ÚVOD DO INFORMATIKY

Pavel Haluza
Jiří Rybička
Tomáš Hála

ÚVOD DO INFORMATIKY

Pavel Haluza
Jiří Rybička
Tomáš Hála

Konvoj®

Poslední aktualizace tohoto textu (v2.00) byla provedena 23. září 2024.

© Pavel Haluza, Jiří Rybička, Tomáš Hála, Brno 2024
© KONVOJ, spol. s r. o., Brno 2024

ISBN 978-80-7302-189-4

Předmluva

Milí studenti,

dostává se Vám do rukou učební text do předmětu *Úvod do informatiky*, který se řadí mezi povinné předměty prvního ročníku bakalářských informaticky zaměřených studijních programů na Provozně ekonomické fakultě Mendelovy univerzity v Brně. Cílem tohoto učebního textu je uvedení do problematiky využití informačních a komunikačních technologií se zaměřením na reprezentaci dat různého typu v počítači, metody kódování a zabezpečení dat zejména při přenosu a také na prostředky pro hromadné zpracování dat. Text se soustřeďuje na základní principy, funkce a využití programového vybavení.

Jednotlivé kapitoly na sebe navazují, nedoporučujeme tedy v textu přeskakovat a vybírat náhodně jen některá témata. Pro vlastní kontrolu, zda jste materiál správně pochopili, doporučujeme odpovědět na kontrolní otázky a vypracovat příklady, zvláště ty neřešené. Většina těchto příkladů bude řešena v rámci cvičení, kde k nim získáte správné výsledky.

Nejedná se o typ textu, který lze číst podobně jako beletrii – nad mnoha částmi je potřebné se zamyslet a pochopit různé principy a souvislosti. Není tedy vhodné měřit pokrok v učení počtem „přečtených“ stránek, ale spíše schopností řešit nastíněné problémy a otázky. Jen tak bude mít pro Vás studium požadovaný efekt.

Přejeme Vám mnoho úspěchů nejen při studiu.

Autoři

Obsah

1 Úvod **9**
1.1 Základní pojmy . . . 9
1.2 Ergonomie práce s počítačem . . . 11
1.3 Údržba počítače . . . 13

2 Číselné soustavy **17**
2.1 Nepoziční soustavy . . . 17
2.2 Poziční soustavy . . . 20
2.2.1 Polyadické soustavy . . . 23
2.2.2 Převody mezi soustavami . . . 25
2.2.3 Aritmetické operace . . . 27
2.3 Příklady na procvičení . . . 29

3 Teorie informace a informatika **33**
3.1 Základní pojmy . . . 33
3.2 Měření množství informace . . . 36
3.3 Jednotky informace . . . 41
3.4 Příklady na procvičení . . . 43

4 Vnitřní reprezentace dat **49**
4.1 Reprezentace numerických dat . . . 49
4.1.1 Binární řádová mřížka . . . 49
4.1.2 Celá čísla . . . 51
4.1.3 Racionální čísla s pevnou řádovou čárkou . . . 57
4.1.4 Racionální čísla s pohyblivou řádovou čárkou . . . 58
4.2 Reprezentace textových dat . . . 67
4.2.1 Jednobajtové znakové kódy . . . 67
4.2.2 Vícebajtové znakové kódy . . . 71
4.3 Reprezentace grafických dat . . . 77
4.3.1 Barvy a barevné modely . . . 77
4.3.2 Rastrové obrazy . . . 79
4.3.3 Vektorové obrazy . . . 81
4.4 Reprezentace zvukových dat . . . 82
4.4.1 Vzorkování . . . 83
4.4.2 Sekvence MIDI . . . 84
4.5 Příklady na procvičení . . . 85

5 Přenos, kódování a zabezpečení dat **93**
5.1 Přenos dat . . . 93
5.1.1 Spojitý a diskrétní signál . . . 94
5.1.2 Přenosový řetězec . . . 96

5.2 Kódování dat . . . 97
5.2.1 Rovnoměrné kódování . . . 97
5.2.2 Nerovnoměrné kódování . . . 99
5.2.3 Hledání optimálního kódu . . . 101
5.3 Zabezpečení dat při přenosu . . . 110
5.3.1 Zabezpečení proti technické nedokonalosti přenosu . . . 110
5.3.2 Zabezpečení proti neoprávněným osobám . . . 123
5.4 Příklady na procvičení . . . 129

6 Principy uložení dat v souborech 137
6.1 Datové formáty . . . 137
6.2 Souborové formáty . . . 142
6.2.1 Asociace formátů a aplikací . . . 144
6.2.2 Rozpoznávání souborových formátů . . . 146
6.2.3 Základní formáty a jejich vlastnosti . . . 147
6.2.4 Konverze souborových formátů . . . 151
6.3 Komprese dat . . . 152
6.3.1 Druhy kompresí . . . 153
6.3.2 Výpočet kompresního poměru . . . 154
6.3.3 Kompresní metody a jejich vlastnosti . . . 155
6.3.4 Možnosti kompresních programů . . . 161
6.4 Souborové systémy . . . 163
6.4.1 Koncepce souboru a adresáře . . . 163
6.4.2 Organizace dat na discích . . . 164
6.4.3 Souborové systémy pro MS-DOS a Windows . . . 166
6.4.4 Souborové systémy pro Unix/Linux . . . 168
6.4.5 Souborové systémy pro macOS . . . 169
6.5 Příklady na procvičení . . . 170

7 Počítačová kriminalita 175
7.1 Porušování autorského práva a souvisejících práv . . . 175
7.1.1 Kategorie softwaru . . . 176
7.1.2 Softwarové licence . . . 177
7.1.3 Formy distribuce softwaru . . . 177
7.1.4 Ochrana proti nelegálnímu použití software . . . 178
7.2 Útoky na bezpečnost dat a systémů . . . 179
7.2.1 Hacking a cracking . . . 179
7.2.2 Malware . . . 181
7.2.3 Grayware . . . 186
7.2.4 Obrana proti různým formám infiltrace . . . 188
7.3 Techniky sociálního inženýrství . . . 191
7.3.1 Spoofing . . . 191
7.3.2 Phishing . . . 192
7.3.3 Pharming . . . 192

7.4 Kyberstalking a kyberšikana 192
7.5 Šíření nevyžádaných a nepravdivých informací 193
7.5.1 Spam 193
7.5.2 Hoax 195
7.6 Prostředky pro zajištění ochrany dat 196
7.6.1 Zálohování a archivace 196
7.6.2 Synchronizace 200

Použitá a doporučená literatura **210**

Seznam zkratek **211**

Rejstřík **212**

1 Úvod

V úvodní kapitole se seznámíme s problematikou, která nás bude provázet celým předmětem. Představíme si základní pojmy z oblasti výpočetní techniky, jejichž osvojení a pochopení je nezbytným předpokladem úspěchu v dalším studiu. Dále se zaměříme na metody správné práce s počítačem, které eliminují možné zdravotní problémy. V závěru kapitoly uvedeme některé důležité prvky preventivní údržby jednotlivých částí počítače.

Cíl

Cílem této kapitoly je osvojení základních pojmů a principů z výpočetní techniky včetně správných návyků při práci u počítače a péče o počítač.

Návaznosti

Kapitolu lze studovat s minimálními předchozími znalostmi, které lze v dnešní době považovat za součást všeobecného přehledu daného základním a středoškolským vzděláním.

Studijní čas

Čas potřebný k nastudování této nenáročné úvodní kapitoly je asi 2 hodiny.

1.1 Základní pojmy

Nejdůležitějším pojmem v moderní výpočetní technice je **počítač**. Nazýváme tak elektronické zařízení na automatizované zpracování dat podle předem vytvořeného programu. Samotný pojem má základ ve slově „počítat“, a to i v angličtině (*compute* → *computer*), jeho význam se však postupně měnil. Původně označoval člověka, který prováděl výpočty, dnes tímto pojmem označujeme stroj. Zdaleka tím ale nemyslíme pouze počítač, který máme doma pod stolem. Počítač dnes řídí činnosti nejrůznějších zařízení a nachází se všude kolem nás – v automobilech, mobilních telefonech, automatických pračkách, mikrovlnných troubách, průmyslových robotech, letadlech, digitálních fotoaparátech, přehrávačích multimédií, záchodových splachovadlech, zámcích na karty u dveří, dětských hračkách apod. *Počítač*

Počítače dnes zkrátka zasahují téměř do všech lidských činností i do běžného života. Předpokládá se, že jejich vliv se bude nadále zvyšovat a budou lidem poskytovat stále komfortnější služby. Počítače jsou v současné době propojovány pomocí počítačových sítí a využívají celosvětovou síť Internet. Počítačové sítě umožňují sdílení zdrojů (soubory, tiskárny), ale i vzájemnou komunikaci, která je dnes jedním z hlavních moderních komunikačních nástrojů informační společnosti.

Není jednoduché najít v historii první skutečný počítač, za předchůdce dnešních počítačů však můžeme bez obav považovat mechanická počitadla, z nichž ta nejstarší vznikla ještě před naším letopočtem. Těmto nástrojům stejně jako celému vývoji výpočetní techniky se věnuje samostatný učební text.

Počítač je ovládán uživatelem, který mu poskytuje data ke zpracování prostřednictvím jeho vstupních zařízení. Moderní počítač (obr. 1.1) se vždy skládá ze dvou částí – z technického vybavení a z programového vybavení.

Obrázek 1.1: *Moderní počítač*

Hardware

Technické vybavení [1] (*hardware*) představuje všechny fyzicky existující součástky počítače. Patří sem především:

- *počítačová skříň* – bedna z plechu nebo jiných materiálů, ve které je uložena základní deska s procesorem, operační pamětí a sběrnicemi, dále pevný disk a elektrický zdroj;
- *monitor* – zobrazuje informace uživateli, je připojen ke grafické kartě;
- *klávesnice* – zprostředkovává alfanumerický vstup od uživatele;
- *myš* – umožňuje pohybovat kurzorem myši a vyvolávat události stiskem tlačítka;
- další *vstupní* či *výstupní zařízení* – tiskárna, skener, reproduktory apod.

Technickému vybavení počítače včetně aktuálních trendů a parametrů jednotlivých součástek se podrobně věnuje samostatný předmět.

Software

Programové vybavení (*software*) je tvořeno sadou všech dat a počítačových programů, které vykonávají nějakou činnost. Je nezbytné pro provoz počítače a řeší konkrétní úlohy ve spolupráci s uživatelem. Protože je software zpravidla považován za autorské dílo, koncoví uživatelé jej využívají na základě licencí (viz kap. 7.1.2, str. 177).

Software lze rozdělit do dvou kategorií:

- *operační systém* – speciální programy zajišťující chod samotného počítače;
- *aplikační software* – programy pro uživatele nebo zajištění řízení jiného stroje.

Mezi důležité pojmy související se softwarem patří zejména:

- *algoritmus* – jednoznačný návod či postup pro řešení určitého typu úlohy, který v každém kroku určuje, jak dále postupovat;
- *programovací jazyk* – „umělý“ počítačový jazyk jako prostředek pro zápis algoritmů;
- *instrukce* – předpis k provedení jednoduché činnosti, kterou realizuje technické vybavení počítače (přičtení jedničky, uložení hodnoty do paměti apod.);
- *program* – algoritmus zapsaný jako posloupnost instrukcí v programovacím jazyku;
- *programování* – implementace algoritmu, jejímž cílem je vytvořit program;
- *proces* – spuštěný program, který provádí požadovanou činnost.

[1] Slangový výraz pro technické vybavení počítače zní „železo“, což je pravděpodobně odvozeno z anglického ekvivalentu. Ten v USA totiž kromě technického vybavení počítače označuje i prodejny s nářadím a železářským zbožím.

1.2 Ergonomie práce s počítačem

Dospěli jsme do doby, ve které se počítače staly nástrojem denní potřeby. Trávíme u nich dlouhé hodiny a přitom zapomínáme na své zdraví. Závažné potíže způsobené častým a nesprávným vysedáváním u počítače zpravidla nejsou na první pohled znatelné a mnohdy se objevují až s odstupem času. V některých případech už však může být pozdě. Špatné návyky při práci s počítačem přitom mohou poškozovat oči, krk, záda, ruce i životosprávu.

Optimalizaci lidské činnosti zejména vhodnými rozměry a tvary nástrojů, nábytku a jiných předmětů zkoumá nauka jménem *ergonomie*. Její název pochází z řeckých slov *ἔργον* (práce) a *νόμος* (zákon) a jak už název této vědní disciplíny napovídá, zabývá se tím, jak má člověk pracovat u počítače, aby mu to způsobilo co nejmenší zdravotní újmu.

Nesprávné pracovní návyky

Nejčastější skupinou objektivně zjistitelných nemocí z povolání je **poškození z opakovaného namáhání** (*Repetitive Strain Injury*). V souvislosti s prací na počítači se o tomto problému diskutuje poměrně dlouho. Je vyvolán opakovanou monotónní zátěží určitých partií těla a monotónní duševní prací. V posledních letech dochází k vysokému nárůstu počtu případů, proto lze tento problém považovat za jedno z největších zdravotních rizik moderního světa.

Syndrom karpálního tunelu

Jedním z nejčastějších problémů je *syndrom karpálního tunelu*. Jde o postižení nervů v oblasti zápěstí, které bývá způsobeno nevhodnou prací se vstupními zařízeními (časté klikání myší, používání nevhodné klávesnice). Příznaky způsobené tlakem v zápěstí se projevují bolestmi prstů, poklesem jejich citlivosti, otoky nebo častým mravenčením. V pokročilém stavu může dojít až k ochrnování postižené ruky. Operace většinou pomůže, ale jsou známy i případy, kdy už nemohla být obnovena plná citlivost nebo funkce ruky.

Syndromu a obecně bolestem rukou se dá přitom jednoduše předcházet. Stačí, aby ruce při práci na počítači svíraly tupý úhel a zápěstí bylo co nejvíce v rovině. Klávesnici je tedy potřeba mít v přiměřené výšce. Praktická je také gelová podložka pod myš a rozšířený okraj klávesnice. Naprosto nevhodná je výsuvná klávesnice, při které se neúměrně namáhají ruce. Toto řešení bohužel najdeme u většiny rádoby ergonomických počítačových stolů.

Syndrom kubitálního tunelu

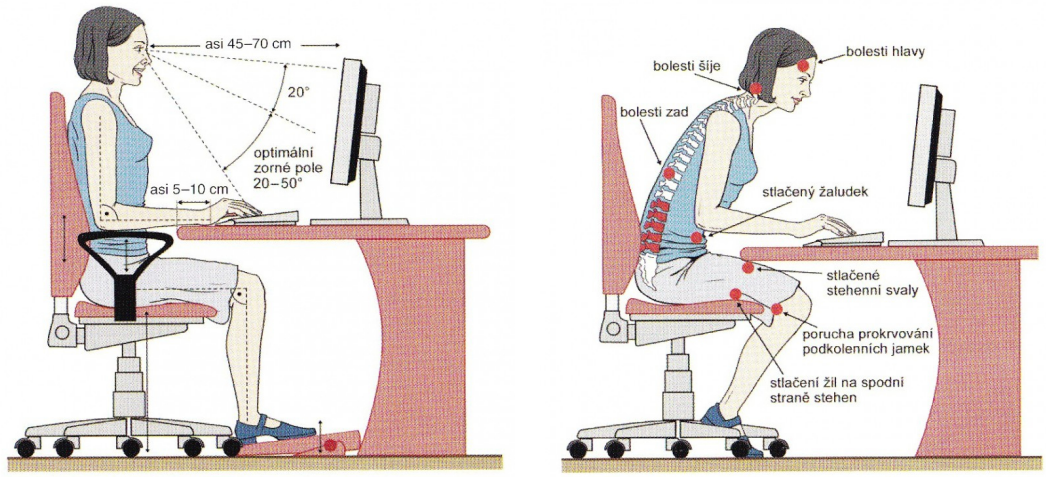
Vysoký počet malých svalů je namáhán také při psaní. Při několikahodinové práci denně dojde až desítkám tisíc jejich použití. V drtivé většině nejsme experti na psaní všemi deseti a máme tedy sklony psát „nezdravě“. Jednou z nejčastějších chyb při psaní je využívání jedné ruky u klávesových zkratek. Jde například o psaní velkých písmen nebo používání častých klávesových zkratek. Používání jedné ruky zbytečně namáhá prstové svaly a v konečném důsledku může vést až k *syndromu kubitálního tunelu*, což je po syndromu karpálního tunelu druhé nejčastější onemocnění postihující nervy v oblasti lokte.

Problémy se zápěstím

Dalším typickým zlozvykem je opírání zápěstí například o okraj stolu. To pak podvědomě nutí zbytečně natáčet ruku k dosažení na některé klávesy. Je lepší mít ruce volně nad klávesnicí a zápěstí co nejvíce v rovině. Vhodné jsou opěrky dlaní, kterými jsou vybaveny některé kvalitnější klávesnice. Tyto opěrky však neslouží k odkládání rukou během psaní, ale právě ve chvíli, kdy nepíšeme a ruce odpočívají.

Uspořádání pracovního místa

Práce s počítačem nese i další rizika, proto je potřeba dodržovat několik základních pravidel a ergonomických návyků, které mohou v mnoha směrech tělu ulevit. Pozornost je potřeba věnovat celému pracovnímu místu, na kterém trávíme většinu doby (obr. 1.2).



Obrázek 1.2: *Správné sezení u počítače (vlevo) a problémy při špatném sezení (vpravo)*

Monitor by měl být v úrovni očí a rozhodně by měl stát přímo před námi. Pokud je umístěn na straně a my k němu musíme otáčet hlavu, zbytečně namáháme oblast krku. Vzdálenost očí od monitoru by měla být minimálně 30 cm, optimálně se uvádí trojnásobek velikosti úhlopříčky (v palcích). Pokud je monitor blíže, působí na oči destruktivní vliv záření vydávaného obrazovkou. Nejde jen o samotné škodlivé „neviditelné" záření, ale také o poškozování oční sítnice vysokou intenzitou viditelného záření. Jestliže je monitor daleko, oči se automaticky pokoušejí zaostřit tak, aby na obrazovce dokázaly rozeznat co nejmenší detaily. Jsou potom neúměrně namáhány a není divu, že brzy začnou bolet. V dnešní době už sice není problém opatřit si méně škodlivý LCD monitor, pokud ale musíme pracovat s CRT monitorem, neměli bychom používat zářivkové osvětlení. Monitor by také nikdy neměl být umístěn proti oknu. Z okna je ale vhodné se občas podívat, donutíme tím oči zaostřit na jinou vzdálenost, zároveň je tak procvičujeme a předcházíme jejich únavě.

Správné sezení

Lokty je potřeba držet při těle a měly by svírat pravý úhel. Předloktí mějme pokud možno rovnoběžně s podlahou a nepodceňujme ani opěrky rukou na židli. Samotná židle by pak měla podporovat rovné držení zad a opěrátko by nemělo být příliš prohnuté. Výšku židle bychom měli nastavit tak, aby nohy svíraly pravý úhel a dotýkaly se země. Pokud se tak neděje, je vhodné pořídit si speciální podložku na nohy. Mezi vhodné cviky pak patří kroužení hlavou a rameny, zvedání a uvolňování ramen nebo předpažení a rozpažení rukou. Pro záda je pak vhodné jejich úplné napřímení.

Škodlivost notebooků

Samostatnou kapitolu představují notebooky, o kterých nelze hovořit jako o ergonomických zařízeních určených k dlouhodobému používání. Původně byly navrženy pro práci na cestách, stále více lidí si je však pořizuje jako náhradu běžného počítače. Velká část studentů je používá k plnění studijních povinností, a to běžně i několik hodin denně.

Zásadním nedostatkem notebooků je umístění klávesnice a displeje velmi blízko u sebe, což prakticky znemožňuje práci ve vzpřímené pozici. Ještě horší je to u netbooků, které jsou pro jakoukoli delší práci zcela nevhodné, přestože výrobci se nám snaží tvrdit opak.

Je zřejmé, že pokud pravidelně dojíždíme a potřebujeme mít svůj počítač stále při sobě, bez notebooku se pravděpodobně neobejdeme. V takovém případě se vyplatí investovat

alespoň do speciálního polohovatelného stojanu, který posune displej výše do úrovně očí. Dále se vyplatí uvažovat o samostatném LCD monitoru. V obou případech docílíme připojením externí klávesnice a myši alespoň částečně ergonomického pracovního prostředí. Doplnit jej můžeme také gymnastickým míčem, který lze používat nejen ke krátkým cvikům v pauzách, ale také jako náhražku nedostatečně kvalitní židle. Existuje i vysoce ergonomické počítačové příslušenství, jakým je například vertikální myš nebo trackball, ale dostupnost těchto zařízení na tuzemském trhu je velmi omezená, takže je potřeba navštívit zahraniční internetové obchody.

Duševní hygiena

Velmi důležitý je pohyb, který také může částečně kompenzovat nedostatečně ergonomický hardware. Pohyb ovšem bývá velmi často zanedbáván a stejně tak krátké přestávky. Doporučuje se rozdělení práce do menších segmentů. Každých 10 minut bychom si měli udělat na 10 až 20 sekund krátkou přestávku a dát ruce pryč z klávesnice. Po každých 30 až 60 minutách je pak vhodné přerušit činnost na 2 až 5 minut, projít se a protáhnout. Vysedávání u počítače bývá často spojeno také s obezitou, a proto bychom měli při práci konzumovat jen zdravá jídla.

1.3 Údržba počítače

Počítač je plný elektroniky, a proto doslova přitahuje prach. Jeho povrch proto musíme pravidelně stírat. Na odstraňování prachu sice existují speciální prostředky, které jsou k sehnání ve specializovaných prodejnách, ale pro základní očištění povrchu monitoru a skříně počítače postačí předem propraná a usušená (nebo polosuchá) prachovka. K čištění obrazovky používáme totéž, co na okna, jen musíme dávat pozor, aby se do žádné části počítače nedostaly tekutiny.

Údržba skříně

Jednou za čas (podle prašnosti prostředí, obvykle jednou za 6–12 měsíců) je třeba provést důkladnější údržbu. Odšroubujeme kryt skříně počítače (několik šroubů vzadu), sejmeme ho a podíváme se, v jakém stavu je vnitřek. Při prvním otevření počítače nás často čeká nepříjemné překvapení, protože zjistíme, že vnitřek je vyplněn změtí izolovaných drátů a pásových kabelů. To může působit značné problémy, protože kabely překážejí proudění vzduchu a některé čipy se při delším a intenzivním provozu mohou přehřívat. Proto první, co uděláme, je opatrné uspořádání a svázání kabelů tak, aby ve skříni byl dostatek místa. Kabely svazujeme pokud možno izolovanými drátky, a to tak, aby nebyly příliš sevřeny. Dalším nepříjemným překvapením bývá vrstva prachu na čemkoli, co se ve skříni nachází. Prach je velmi nebezpečný všemu, co pracuje s elektřinou, protože tvoří termoizolační vrstvu. Tato vrstva brání čipu v chlazení, navíc je čip elektricky vodivý a velmi si rozumí se statickou elektřinou. Na oprašování vnitřku počítače rozhodně nepoužíváme prachovku. Existují speciální foukací balónky, kterými prach odfoukáváme pryč, ale musíme dávat pozor kam. Hodně choulostivý je například zdroj, a by jej proto mračna prachu v žádném případě neměla zasáhnout. Mnohem lepší je vysavač. Prach jím odstraňujeme opatrně tak, abychom se trubicí nedotkli žádné součástky (statická elektřina, mechanické poškození apod.), a nemusíme se přitom starat, kam je foukán.

V napájecím zdroji je vestavěn kondenzátor a akumulátor, a proto může být nebezpečný i ve vypnutém stavu. Zacházíme s ním velmi opatrně a rozhodně se jej nepokoušíme

rozebrat. Základní deska (*motherboard*) je nejdůležitější částí počítače. Na desce je upevněn procesor, paměťové čipy, vedou jí tištěné obvody apod., proto o ni musíme přiměřeně důkladně pečovat. Základní desku je nutné čas od času zbavit prachu (obr. 1.3), ale to není všechno. Čipy na deskách jsou buď přiletovány, nebo zasazeny do patic. Vlivem tepla se čipy ze svých patic vysunují, proto je musíme opatrně zatlačit zpět. Při této akci dáváme pozor především na to, abychom nezlomili nožičky nebo jiné součásti čipu, které zasunujeme, a také abychom nezlomili základní desku. Pokud to jde, tak při zasouvání základní desku podložíme (alespoň prstem ruky), nebo ji celou vymontujeme.

Po několika letech provozu je nutná důkladnější preventivní údržba. Pokud si nejsme něčím jisti, necháme to raději na odborníkovi, protože při rozebírání počítače může dojít k různým nečekaným problémům. Každý správce má svůj postup, navíc jednotlivé součástky lze vymontovat pouze v určitém pořadí, které se může u jednotlivých počítačů lišit.

Obrázek 1.3: *Počítač, který nutně potřebuje vyčistit*

Údržba klávesnice

Klávesnice vyžaduje speciální údržbu. Dbáme na to, aby se nad ní nesvačilo, protože drobky se špatně odstraňují, voda je zase nebezpečím pro kovové součásti (koroze, elektřina apod.) a tekutiny obsahující cukr a kyseliny dokážou spolehlivě a rychle zoxidovat cokoliv. Prach a drobky lze zvenčí odstranit například vysavačem. Pokud se do klávesnice dostane voda, odpojíme ji, vytřepeme co nejvíce vody a necháme důkladně usušit (nepoužíváme sušičku ani mikrovlnnou troubu, stačí pouze ponechat v suché místnosti). Jestliže se na klávesnici vylije sladká nebo kyselá šťáva, zapomeneme na nebezpečnost vody a klávesnici ihned strčíme pod její proud. Po důkladném propláchnutí opět necháme usušit.

Uvnitř počítače se nedotýkáme ničeho kovového, protože částečky kůže, potu, tuku apod. mohou odstartovat oxidaci, neboť obsahují kyseliny. Na stejné podložce, na které je umístěn pevný disk, nesmí být žádný zdroj vibrací. Jde například o některé typy tiskáren.

Teplo a koroze

Teplo škodí počítačům především tím, že urychluje korozi (při nárůstu teploty o 10 °C se rychlost koroze přibližně zdvojnásobuje) a zkracuje životnost součástek. Každý počítač má ventilátor (obvykle vestavěný ve zdroji), který však často nestačí, a někdy si za to můžeme i sami. Příliš vysoká teplota uvnitř může být způsobena velkou vrstvou prachu, která tepelně izoluje, slabým ventilátorem, slabým napájecím zdrojem nebo vysokou pokojovou teplotou. Pokud nestačí ventilátor, můžeme pořídit nový výkonnější (obvykle bývá na procesoru). Počítač nikdy neumísťujeme do blízkosti topných těles. Tepelný šok (náhlé zvýšení teploty) je obzvlášť destruktivní. Proto se doporučuje při přesunu počítače z chladu do tepla (například u notebooků) nechat jej předem zahřát na pokojovou teplotu a až poté zapnout.

Řešením může být také nechávat počítač neustále zapnutý, protože k nejhorečnější činnosti a tím také k tepelnému šoku dochází při zapnutí počítače.

Cigaretový prach

Kouření v blízkosti počítače zkracuje jeho životnost přibližně o 40 %. Nejvíce je ohrožen pevný disk, o kterém se říká, že je uzavřen ve vzduchotěsné schránce. Jenže molekuly popílku v cigaretovém kouři jsou mnohem menší než molekuly vzduchu. Výrobce s tím nepočítá, navíc kdyby schránky na disky měly být „kouřotěsné“, byla by jejich výroba mnohem dražší. Molekula kouře pronikne do schránky pevného disku, a protože na světě fungují jisté zákonitosti ohledně gravitace, elektřiny a statické elektřiny, je doslova přitahována k povrchu disků (pevný disk je ve skutečnosti několik disků ve svazku nad sebou). Čtecí a zápisová hlava se pohybuje na nízkém vzduchovém polštáři nad povrchem disku. Jestliže se v její cestě nahromadí dostatečné množství molekul, dojde ke kolizi a následnému poškození dat (v horším případě i povrchu disku), protože hlava může při své vysoké rychlosti velmi prudce narazit na povrch. Nebo na povrch nenarazí, ale vleče molekuly s sebou, čímž vznikají zajímavé rýhy, na kterých jsou ovšem data ztracena.

Magnetismus

Magnetické a elektromagnetické pole škodí především datům uloženým na magnetických pamětech (pevný disk, disketa). Proto v blízkosti počítače a disket se zdržíme pokusu o umístění magnetů a čehokoliv, co je může obsahovat. Ve starších telefonech je zvonění působeno kladívkem vybuzeným elektromagnetem, proto by tyto telefony neměly být v blízkosti disků. Motor tiskárny obvykle nebývá stíněn a může také produkovat elektromagnetické pole. Určité nebezpečí představují detektory kovů, rentgenové přístroje a různé detektory na letištích. V blízkosti počítače také rovněž nepoužíváme magnetické šroubováky. Motory v mechanikách pevných a pružných disků neprodukují příliš silné magnetické pole, a jsou proto v pořádku. Žádné větší motory ani topná tělesa by neměly být zapojeny do elektrické zásuvky, která napájí počítač. Jde například o vařiče, ledničky, někdy může dělat problémy i obyčejná varná konvice.

Sluneční záření

Počítači škodí také přímé sluneční záření. Na vnitřní straně obrazovky starších CRT monitorů je vrstva látky obsahující především fosfor. Úkolem této vrstvy je chemicky reagovat na elektrony záření vysílaného třemi náboji uvnitř monitoru a tím zobrazovat to, co uživatel vidět má. Při těchto chemických procesech se vrstva pomalu a nenápadně poškozuje, proto třeba na starších monitorech můžeme vidět vypálená místa. Sluneční záření má na tuto vrstvu podobný vliv. Pokud vystavíme obrazovku slunečnímu záření, obraz postupně bledne a kvalita zobrazení klesá.

Kontrolní otázky

1. Co je to počítač a z jakých částí se skládá?
2. Co je příčinou poškození z opakovaného namáhání a jak mu lze předcházet?
3. Kde by měl při práci s počítačem být umístěn monitor?
4. Proč může být pravidelné používání notebooků škodlivé?
5. Jak často a jakým způsobem zbavujeme počítač prachu a jiných nečistot?

Shrnutí

V této úvodní kapitole jsme si objasnili význam pojmu počítač a popsali jsme jednotlivé komponenty moderních osobních počítačů. Zaměřili jsme se i na zdravotní problémy, které může způsobovat dlouhodobá práce s počítačem, pokud nebudeme dodržovat základní pravidla. V poslední části kapitoly byla představena některá doporučení k preventivní údržbě jednotlivých komponent počítače.

Slovníček pojmů

Hardware: technické vybavení počítače, souhrn fyzicky existujících součástek počítače, které tvoří počítačová skříň s vybavením (zdroj, základní deska, procesor, sběrnice, paměti), vstupní a výstupní zařízení.

Počítač: stroj na automatizované zpracování dat podle předem vytvořeného programu.

Poškození z opakovaného namáhání: skupina objektivně zjistitelných nemocí z povolání způsobených nesprávnými návyky při práci s počítačem.

Software: programové vybavení počítače, souhrn dat a počítačových programů, obvykle chráněno licencí.

Řešení kontrolních otázek

1. *Co je to počítač a z jakých částí se skládá?*

Počítač je stroj na automatizované zpracování dat podle předem daného programu, skládá se z technického vybavení (hardware) a programového vybavení (software).

2. *Co je příčinou poškození z opakovaného namáhání a jak mu lze předcházet?*

Příčinou je monotónní zátěž určitých partií těla, typickým projevem je syndrom karpálního nebo kubitálního tunelu. Problému lze předcházet správnou polohou rukou a zápěstí při psaní, lze využít praktických ergonomických pomůcek, např. gelové podložky pod myš.

3. *Kde by měl při práci s počítačem být umístěn monitor?*

Monitor by měl být v úrovni očí přímo před námi, ideálně ve vzdálenosti až dvojnásobku úhlopříčky obrazovky. Neměl by být umístěn proti oknu ani vystaven slunečnímu záření.

4. *Proč může být pravidelné používání notebooků škodlivé?*

Notebooky mají klávesnici a displej blízko u sebe, což znemožňuje práci ve vzpřímené pozici. Vhodným řešením může být pořízení polohovatelného stojanu, samostatného monitoru a klávesnice.

5. *Jak často a jakým způsobem zbavujeme počítač prachu a jiných nečistot?*

Údržbu bychom měli provádět jednou až dvakrát ročně, je vhodné odstraňovat prach nejen z povrchu skříně, ale i z jejich útrob, kde může působit jako termoizolační vrstva a způsobovat přehřívání jednotlivých komponent. Čistíme opatrně vysavačem. Klávesnici v případě nutnosti odpojíme, omyjeme pod proudem čisté vody a poté ji necháme důkladně vyschnout v suché místnosti.

2 Číselné soustavy

Úvod

Číselná soustava je způsob vyjádření určitého počtu základních jednotek. K vyjádření velikosti **čísla** se užívá elementárních symbolů nazývaných **číslice** a jejich kombinací. Způsobu přiřazení určité kombinace symbolů danému počtu základních jednotek říkáme kód.

Číslo a číslice

Všechny moderní počítače pracují ve dvojkové soustavě. Proč tomu tak je? Čím se počítání v této soustavě liší od počítání v soustavě, na kterou jsme odmalička zvyklí? Nejen na tyto otázky nalezneme odpověď v následujícím textu.

A proč se vůbec musíme číselnými soustavami zabývat? Jednoduše proto, že se bez nich zcela jistě neobejdeme. Číselných soustav existuje teoreticky nekonečně mnoho, každý z nás v běžném životě používá jednu z nich (desítkovou) a v informatice se dnes už nikdo neobejde bez dalších tří. Princip práce je však pro všechny číselné soustavy naprosto shodný. S trochou nadsázky lze proto konstatovat, že kdo zvládá počítání v desítkové soustavě, neměl by mít problém počítat v kterékoli jiné soustavě.

Počet elementárních symbolů v dané soustavě určuje **základ soustavy** (*radix*), který není v soustavě nikdy obsažen. Počet dostupných symbolů v dané číselné soustavě je proto vždy roven základu.

Základ soustavy

Cíl

Cílem této kapitoly je seznámení s různými možnostmi vyjádření čísel. Seznámíme se s číselnými soustavami, ukážeme si, jak mezi nimi můžeme převádět a jakým způsobem lze provádět aritmetické operace v libovolné soustavě. Zaměříme se také na soustavy, které jsou v informatice důležité. Podle způsobu určení hodnoty čísla z dané reprezentace rozdělíme soustavy na poziční a nepoziční.

Návaznosti

Pro studium této kapitoly je nutné zejména zvládat aritmetiku ze základní a střední školy.

Studijní čas

Čas potřebný k nastudování této kapitoly je asi 4 hodiny.

2.1 Nepoziční soustavy

Nepoziční číselné soustavy jsou spíše historickou záležitostí. Dnes už se prakticky nepoužívají, i když existuje jedna výjimka. Důležitou vlastností těchto soustav je skutečnost, že hodnota symbolu není dána jeho umístěním v sekvenci symbolů. Nepoziční soustavy rovněž neobsahují symboly pro nulu a záporná čísla. Jejich velkou výhodou je jednoduché sčítání a odčítání, naopak nevýhodou je obvykle příliš dlouhý zápis čísel, která výrazně převyšují hodnotu největšího symbolu soustavy.

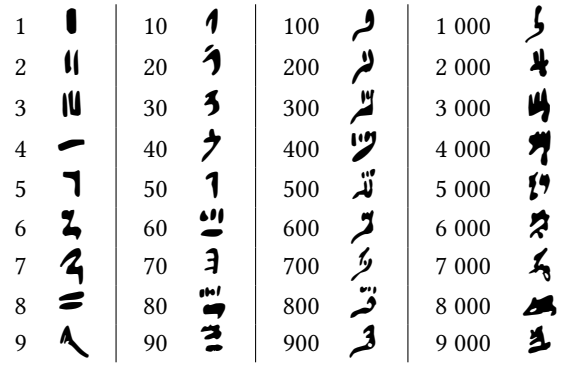
Egypt

Egypťané používali od nejstarších dob nepoziční desítkovou soustavu. Nejstarší záznamy pocházejí z období kolem roku 3000 př. n. l. V *hieroglyfickém* písmu existovaly znaky pro číslo 1 a pro mocniny čísla 10 až po 10^6 (obr. 2.1). V zápisu čísla se uplatňoval aditivní přístup, tj. znak se opakoval tolikrát, kolik jednotek příslušného řádu bylo zapotřebí.

1	10	100	1 000	10 000	100 000	1 000 000

Obrázek 2.1: *Egyptské hieroglyfické číslice*

Zatímco hieroglyfy byly určeny pro monumentální zdobné nápisy a náboženskou sféru, pro běžné použití souběžně existovalo zjednodušené *hieratické* písmo (obr. 2.2), z něhož se později vyvinulo písmo *démotické*.[2] Používalo se více než tisíc let a během této doby prošlo mnoha vývojovými změnami.[3]



1		10		100		1 000	
2		20		200		2 000	
3		30		300		3 000	
4		40		400		4 000	
5		50		500		5 000	
6		60		600		6 000	
7		70		700		7 000	
8		80		800		8 000	
9		90		900		9 000	

Obrázek 2.2: *Egyptské hieratické číslice*

Posledním vývojovým stádiem egyptštiny je koptština[4], kterou se v Egyptě prokazatelně hovořilo ještě v 17. století. Koptské písmo tvoří řecká alfabeta doplněná o několik znaků démotického písma (obr. 2.4 vlevo) a má několik dialektů.

Řecko

Ve starém Řecku používali nejprve v 10. stol. př. n. l. *herodiánské číslice*, které odpovídaly zkratkám číslovek (obr. 2.3).

Ι	Π	Δ	Η	Χ	Μ
(*ἴος*)	(*πέντε*)	(*δέκα*)	(*ἑκατόν*)	(*χίλιοι*)	(*μύριοι*)
1	5	10	100	1 000	10 000

Obrázek 2.3: *Řecké herodiánské číslice*

Někdy na přelomu letopočtu se začínají objevovat *iónské číslice*, které využívají písmena alfabety (stejně jako u koptštiny, viz srovnání na obr. 2.4). Číslice jsou zapsány pomocí velkých písmen řecké abecedy, k nimž je připsán pruh nebo čárka, aby se odlišily od písmen. Bylo použito 27 písmen řecké abecedy včetně tří starých znaků. Tímto způsobem bylo možné zapsat přirozená čísla do 999, s určitou drobnou úpravou pak i vyšší čísla.

[2] Původ pojmenování písem lze najít ve staré řečtině: hieroglyfické jako složenina slov *ἱερός* (posvátný) a *γλύφω* (rytina), hieratické od slova *ἱερατικά* (kněžský), démotické od slova *δημοτικός* (lidový).

[3] Je to právě démotické písmo, které se spolu s majestátními hieroglyfy a řeckým překladem textu nachází na slavné Rosettské desce objevené v roce 1799 během Napoleonova tažení do Egypta.

[4] Pojmenování jazyka je odvozeno z arabského *qibṭ* (قبط), to má původ ve starořeckém *Αἴγυπτος* (Egypťané).

1	ⲁ	10	ⲓ	100	ⲣ	1	Α′	(alfa)	10	Ι′	(ióta)	100	Ρ′	(rhó)
2	ⲃ	20	ⲕ	200	ⲥ	2	Β′	(beta)	20	Κ′	(kappa)	200	Σ′	(sigma)
3	ⲅ	30	ⲗ	300	ⲧ	3	Γ′	(gama)	30	Λ′	(lambda)	300	Τ′	(tau)
4	ⲇ	40	ⲙ	400	ⲩ	4	Δ′	(delta)	40	Μ′	(mí)	400	Υ′	(ypsilon)
5	ⲉ	50	ⲛ	500	ⲫ	5	Ε′	(epsilon)	50	Ν′	(ný)	500	Φ′	(fí)
6	ⲋ	60	ⲝ	600	ⲭ	6	Ϛ′	(stigma)	60	Ξ′	(ksí)	600	Χ′	(chí)
7	ⲍ	70	ⲟ	700	ⲯ	7	Ζ′	(zéta)	70	Ο′	(omikron)	700	Ψ′	(psí)
8	ⲏ	80	ⲡ	800	ⲱ	8	Η′	(éta)	80	Π′	(pí)	800	Ω′	(omega)
9	ⲑ	90	ϥ	900	ϯ	9	Θ′	(théta)	90	Ϙ′	(koppa)	900	Ϡ′	(sampí)

Obrázek 2.4: *Koptské číslice (vlevo) a řecké iónské číslice (vpravo)*

Nejznámější a dodnes používané jsou *římské číslice*. Jsou součástí číselné soustavy, kterou Římané převzali od Etrusků. Soustava se používala i po rozpadu Římské říše. K zápisu jednotlivých čísel byly původně používány samostatné symboly, které byly ve středověku modifikovány do dnešní podoby. Ačkoli již ve 14. století bylo zavedeno používání arabských číslic, v mnoha případech se i v dnešní době římské číslice používají (číselníky hodin, označování století, měsíců a čtvrtletí, číslování položek, odstavců, kapitol apod.). *Řím*

Základem soustavy římských číslic je sedm symbolů: I, V, X, L, C, D, M.[5] Jednoduché pravidlo pro práci s římskými čísly říká, že větší číslice vždy předcházejí menším a výsledné číslo je dáno součtem hodnot všech symbolů. Ve středověku byly pro zkrácení zápisu doplněny složené symboly, u kterých je toto pravidlo porušeno, tedy menší číslice předchází větší a odčítá se od ní (obr. 2.5). Existuje šest takových symbolů: IV, IX, XL, XC, CD a CM.

1	I	10	X	100	C
2	II	20	XX	200	CC
3	III	30	XXX	300	CCC
4	IV	40	XL	400	CD
5	V	50	L	500	D
6	VI	60	LX	600	DC
7	VII	70	LXX	700	DCC
8	VIII	80	LXXX	800	DCCC
9	IX	90	XC	900	CM

Obrázek 2.5: *Římské číslice*

Kontrolní otázky

1. Jaké výhody a nevýhody mají nepoziční číselné soustavy?
2. Na jakém principu je založena řecká iónská číselná soustava a jaký je její původ?
3. Které symboly používá soustava římských číslic a co tyto symboly vyjadřují?
4. Jak se v soustavě římských číslic vyjádří hodnoty 135, 298, 444, 705, 1999 a 2019?

[5] Pro snadné zapamatování existuje celá řada mnemotechnických pomůcek v podobě říkanek, například „Ivan Vedl Xénii Lesní Cestou Do Města“ nebo „Ivan, Vašek, Xénie Lijí Cín Do Mumie“.

2.2 Poziční soustavy

U pozičních číselných soustav je hodnota každého symbolu dána jeho pozicí v sekvenci symbolů, tím je dána váha pro výpočet celkové hodnoty čísla.

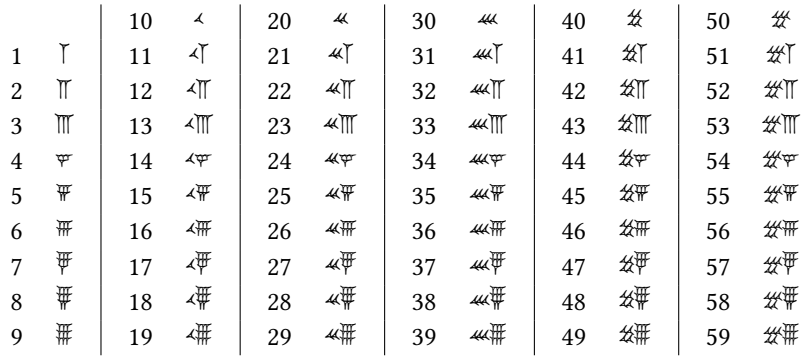
Význam nuly

Nezbytným předpokladem pro používání pozičních soustav bylo objevení symbolu pro nulu. Rozlišujeme však dva druhy nuly podle jejího významu: poziční nulu (vyplňuje prázdné místo a může místo ní být i prázdné místo) a nulu ve významu „žádné množství“, která se začala používat až mnohem později.

Než se budeme věnovat pozičním číselným soustavám používaným v současnosti, podíváme se i na některé významné historické soustavy a osvětlíme si původ číslic, které používáme pro zápis čísel dnes.

Babylón

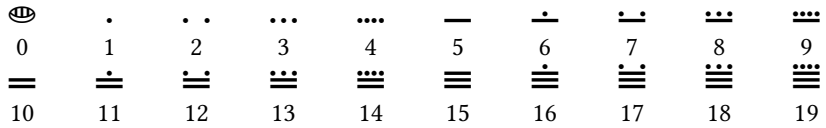
Sumerové a později Babylóňané ve starověké Mezopotámii používali klínopisné číslice (obr. 2.6). Klíny vyrývali rákosovým rydlem do vlhkých hliněných tabulek, které posléze vypalovali na slunci v pecích. Staří Asyřané a Babylóňané byli výbornými matematiky a astronomy a protože potřebovali zapisovat velká čísla, vymysleli přibližně v polovině 3. tis. př. n. l. k jejich zapisování poziční soustavu o základu 60. Zbytky této soustavy se nám zachovaly dodnes – úhly a hodiny dělíme na šedesát minut. Během několika dalších století se ustálil zápis čísel jen pomocí dvou znaků s hodnotami 1 a 10. Symbol pro prázdný řád sice měli (dva šikmé klíny), ale nepoužívali jej k vyjádření nuly v místě nejnižšího řádu.



Obrázek 2.6: *Babylónské klínopisné číslice*

Mayové

Již ve 4. stol. př. n. l. Mayové používali poziční soustavu o základu 20 s pozůstatky dřívější pětkové soustavy a se znakem pro nulu, který byl doplněn v polovině 4. stol. (obr. 2.7). K zápisu čísel používali buď hieroglyfy, nebo kombinace teček a čar, nula byla vyjádřena schematickým obrazem mušle. Při zápisu větších přirozených čísel se řády psaly pod sebe, přičemž nejnižší řád byl dole. Pro astronomické výpočty si Mayové soustavu upravili tak, že ve třetím řádu místo 400 zařadili 360, což odpovídá 1 roku s 18 měsíci po 20 dnech. Zajímavé je, že dvacítkovou soustavu používali také Aztékové a dodnes ji používají Inuité.

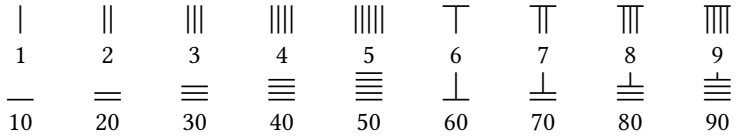


Obrázek 2.7: *Mayské číslice*

Čína

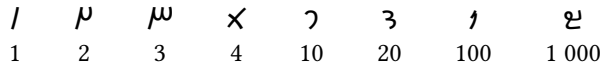
Nejstarší čínská čísla se objevují na magických kostkách ze 14. až 11. stol. př. n. l. a také

na keramických a bronzových předmětech a mincích z 10. až 3. stol. př. n. l. Již od 4. stol. př. n. l. a možná i dříve Číňané používali k vyjádření čísel tyčinky. Existovalo 18 znaků pro čísla 1 až 9 a desítky od 10 do 90. Obě skupiny znaků se liší jen uspořádáním tyčinek (obr. 2.8). Symbol pro nulu zpočátku chyběl a bylo místo něj prázdné místo. Do Číny se nula dostala z Indie, kde se poprvé objevila v 8. století př. n. l. V tištěných čínských matematických spisech se symbol nuly ve tvaru kroužku objevil ve 13. století.



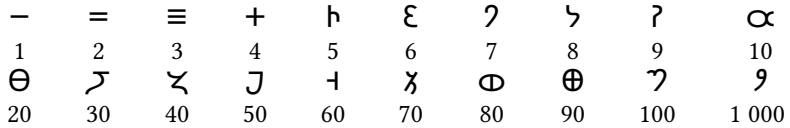
Obrázek 2.8: *Čínské číslice*

Největší vliv na vytváření současných zápisů čísel měly indické číslice. Před poziční soustavou se v Indii používaly různé nepoziční číselné soustavy, které ovšem zanikly. Například na území Gandháry na rozhraní dnešního východního Afghánistánu a severního Paňdžábu v Pákistánu byly od 4. stol. př. n. l. až do 3. stol. n. l. používány číslice staroindického písma *kharóšthí* (खरोष्ठी), které znázorňuje obr. 2.9. *Indie*



Obrázek 2.9: *Indické číslice v písmu kharóšthí*

Desítková soustava existovala v Indii od nejstarších dob. Zřejmě nejrozšířenější byly číslice v písmu *bráhmí* (ब्राह्मी, resp. ve znacích písma 𑀩𑀺𑀭𑀳𑀺𑀮𑀺), které se bez podstatných změn užívaly od 3. stol. př. n. l. déle než tisíc let ve velké části Indie. Existovaly zde zvláštní symboly pro jednotky, desítky, sta a tisíce, větší násobky byly vyjadřovány multiplikativně. Přestože se nejedná o symboly poziční soustavy, staly se základem pro moderní číslice (obr. 2.10). V různých zemích se však postupem času tvarově lišily.



Obrázek 2.10: *Indické číslice v písmu bráhmí*

Existence speciálních symbolů pro číslice 1 až 9 je důležitým rysem indické matematiky, který byl předpokladem pro vznik poziční desítkové soustavy. Byl to velmi významný vědecký i kulturní úspěch národů žijících na území Indie. Teprve tato soustava umožnila provádění početních operací v písemné podobě tak jednoduše, že bylo možné konkurovat počítání na početní desce. Proces vzniku desítkové poziční soustavy byl dlouhý a nebyl dosud zcela objasněn. Patrně se zde užívala od 6. stol., v 7. stol. již prokazatelně existovala a zprávy o ní se začaly šířit na západ. Koncem 8. stol. byla známa v Bagdádu a arabští matematici ji rychle přijali.

Nulu vyjadřovali Indové zpočátku tečkou, kterou později vytlačil všeobecně používaný kroužek. Indické slovo *śūnya* (शून्य, „nic“) Arabové přeložili jako *ṣifr* (صِفْر), z toho pak vzniklo

přes latinské *zephirum* (poprvé použito v roce 1202) nejen všeobecně používané *zero*, ale také slovo *cifra*, které v češtině někdy používáme jako synonymum ke slovu *číslice*.

Nelze přesně zjistit, zda byla nula v Indii objevena, nebo ji Indové převzali od jiných národů. Tvar symbolů pro číslice se v Indii měnil od místa k místu a také postupem času. Z mnoha zápisů se nejvíce používaly číslice písma *dévanágarí* (देवनागरी), které je v Indii základním písmem dodnes (obr. 2.11). Pravděpodobně vznikly úpravou písma *bráhmí* (obr. 2.10).

०	१	२	३	४	५	६	७	८	९
0	1	2	3	4	5	6	7	8	9

Obrázek 2.11: *Indické číslice v písmu dévanágarí*

Arabský svět

Ve starých arabských textech se čísla vyjadřovala slovy nebo řeckou alfabetou. Slovní zápis se vedle číselného udržel i později, zatímco řecký zápis ve 12. stol. vymizel. V 8. stol. se objevila arabská numerace podobná řecké, která se opírala o arabskou abecedu a během dvou století se postupně rozšířila. Avšak již v první polovině 10. stol. se začíná šířit indická poziční soustava v písmu *dévanágarí* (obr. 2.11). První dvě známé práce, které se o ní zmiňují, jsou v arabštině a pocházejí z počátku 9. stol. Jejich autory jsou perský matematik a astronom Abū ʿAbd Allāh Muḥammad ibn Mūsā al-Khwārizmī[6] (* cca 780, † cca 850) a arabský filozof, matematik, astrolog a lékař Abū Yūsuf Yaʿqūb ibn ʿIsḥāq aṣ-Ṣabbāḥ al-Kindī[7] (* cca 801, † cca 873).

Od 10. století se pro zápis čísel v poziční soustavě používaly *východoarabské* číslice (*arqām hindiyyah*, أرقام هندية), které vznikly modifikací číslic písma *bráhmí* (obr. 2.10). Existují ve dvou variantách (obr. 2.12). Přibližně ve stejné době se však na Pyrenejském poloostrově a v severozápadní Africe objevily *západoarabské*[8] číslice. Nazývaly se *ghubār* (غبار), což znamená písek či prach. O jejich rozšíření v Evropě se významně zasloužil italský matematik Leonardo Fibonacci (* cca 1170, † cca 1250). Rukopisnými opisy se tvary číslic různě obměňovaly, k ustálení jejich tvaru přispělo až rozšíření knihtisku v 15. století. Východoarabské číslice se dodnes udržely na Arabském poloostrově, v Egyptě, Iráku, Íránu, Jordánsku, Libanonu či Sýrii, západoarabské číslice se užívají například v Maroku.

٠	١	٢	٣	٤	٥	٦	٧	٨	٩	(standardní tvar)
٠	١	٢	٣	۴	۵	۶	۷	۸	۹	(perská varianta)
0	1	2	3	4	5	6	7	8	9	

Obrázek 2.12: *Východoarabské číslice*

Jedničková soustava

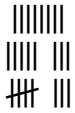
Speciálním případem poziční soustavy je **jedničková soustava**, ve které je každé číslo vyjádřeno opakováním jednoho stejného symbolu. Při pohledu na obr. 2.13 si jistě vzpome-

[6] Jeho jméno (arabsky أبو عبد الله محمد بن موسى الخوارزمي) bylo psáno jako „al-Gorizmí“, později „Algoritmí“, z čehož vznikl *algoritmus*, ačkoli princip algoritmizace byl známý již dříve. Je také autorem spisu *Al-Kitāb al-muḫtaṣar fī ḥisāb al-ǧabr wa'l-muqābala* (الكتاب المختصر في حساب الجبر والمقابلة), kde z arabského „al-ǧabr“ pochází dnešní slovo *algebra*. Oblast Khwārizm se rozkládá okolo dnešního města Chiva v Uzbekistánu.

[7] Jeho jméno (arabsky أبو يوسف يعقوب بن إسحاق الصبّاح الكندي) bylo zkracováno na latinské „Alkindus“. Je považován za otce arabské filozofie, dohlížel na překlady řeckých vědeckých a filozofických textů do arabštiny. Je také považován za jednoho z otců kryptografie, položil základy kryptoanalýzy. Je autorem nejméně 260 knih, které přispěly k rozvoji geometrie, medicíny, filozofie, logiky a fyziky.

[8] Bývají nazývány také jako indo-arabské či hindsko-arabské, příp. nepřesně pouze arabské, a jsou právě ty číslice, které dnes tvoří nejrozšířenější systém symbolického zápisu čísel.

neme, kde jsme se s touto soustavou už setkali. Sice nevykazuje schopnosti plnohodnotné poziční soustavy, ale rozhodně ji nelze řadit mezi nepoziční soustavy. Tato soustava tvořila základ pro zápis prvních tří číslic indo-arabské, římské nebo čínské soustavy.



Obrázek 2.13: *Jedničková soustava*

2.2.1 Polyadické soustavy

Nejrozšířenějším typem pozičních číselných soustav jsou **polyadické soustavy**, ve kterých lze číslo vyjádřit součtem mocnin základu dané soustavy vynásobených příslušnými platnými číslicemi. Základů může být obecně více[9], ale nejčastěji pracujeme v soustavách s jedním základem, kterým může být libovolné přirozené číslo $z > 1$. Taková soustava se potom nazývá z-adická a počet různých symbolů pro vyjádření čísel je u ní roven z.

Základ soustavy

Libovolné číslo můžeme v poziční číselné soustavě o jednom základu $z > 1$ vyjádřit dvěma ekvivalentními způsoby:

Způsoby zápisu

1. *pozičním zápisem*
$$(a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-m})_z;$$
2. *polynomem*
$$\sum_{i=-m}^{n} a_i \cdot z^i.$$

Proměnná a_i představuje řádovou číslici na pozici i ($0 \leq a_i < z$), proměnná n označuje nejvyšší řád s nenulovou číslicí ($n \in \mathbb{N}_0$), proměnná m označuje nejnižší řád s nenulovou číslicí ($m \in \mathbb{N}_0$). Řádová čárka odděluje celou a zlomkovou část čísla (viz str. 25). Není-li uvedena, jedná se o celé číslo, kde poslední platnou číslicí je a_0.

Příklad 2.1

Mějme číslo A v soustavě o základu $z = 10$ dané koeficienty $a_2 = 1$, $a_1 = 2$, $a_0 = 5$, $a_{-1} = 4$ a číslo B v soustavě o základu $z = 5$ dané koeficienty $a_3 = 4$, $a_1 = 3$, $a_0 = 1$, $a_{-1} = 2$, $a_{-2} = 4$. Použijeme-li poziční zápis čísel, dostáváme:

$$A = (125{,}4)_{10} \qquad B = (4031{,}24)_5$$

Je-li zřejmé, v jaké soustavě je číslo vyjádřeno, nemusíme základ soustavy uvádět. Pokud vyjádříme obě čísla polynomem, dostáváme:

$$A = 1 \cdot 10^2 + 2 \cdot 10^1 + 5 \cdot 10^0 + 4 \cdot 10^{-1}$$
$$B = 4 \cdot 5^3 + 0 \cdot 5^2 + 3 \cdot 5^1 + 1 \cdot 5^0 + 2 \cdot 5^{-1} + 4 \cdot 5^{-2}$$

[9] Z historie známe mnoho různých základů, například tucet (12), mandel (15), kopa (60), veletucet (144), velekopa (3600) nebo staroanglická libra dělená na šilinky (20), pence (240) a farthingy (960).

Záporná čísla

Definice polyadických soustav uvažují pouze nezáporná čísla. Je tomu jednoduše proto, že záporná čísla nebyla dlouho potřeba. Například v Evropě se záporná čísla začala používat až v 15. století. Chceme-li zapsat záporné číslo, musíme použít předem dohodnutý způsob, kterým takové číslo rozlišíme od kladného. Běžně za tímto účelem používáme unární operátor minus zapsaný před číslem. Počítač při práci se zápornými čísly používá *číselné kódy*, kterými transformuje omezenou množinu celých čísel do omezené množiny nezáporných čísel. S problematikou se blíže seznámíme v následujících kapitolách (viz str. 52), do té doby se zápornými čísly pracovat nebudeme.

Důležité soustavy

Je zřejmé, že číselných soustav může být nekonečně mnoho. V souvislosti s výpočetní technikou budeme často kromě desítkové soustavy používat ještě další tři. Jejich charakteristiky jsou uvedeny v tab. 2.1.

Tabulka 2.1: *Významné číselné soustavy*

Název soustavy	Základ	Číslice – symboly pro zápis čísel	Typické použití
Dvojková	2	0, 1	všechny moderní počítače
Osmičková	8	0, 1, 2, 3, 4, 5, 6, 7	přístupová práva v OS Unix
Desítková	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	běžné počty
Šestnáctková	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	MAC adresy, odstíny barev

Rozsah soustavy je vždy určen hodnotami z intervalu od 0 po $z - 1$, kde z je základ soustavy. Dokazuje to jen skutečnost, kterou jsme zmínili dříve – základ soustavy v ní nikdy není obsažen. V případě šestnáctkové soustavy nastává drobný problém s tím, že potřebujeme rozlišit 16 různých symbolů, ale známe jen 10 číslic. Proto chybějící číslice nahradíme písmeny anglické abecedy. Podobu několika prvních čísel vyjádřených ve výše uvedených soustavách znázorňuje tab. 2.2.

Tabulka 2.2: *Vyjádření čísel 0 až 11 v různých soustavách*

$z = 2$	0	1	10	11	100	101	110	111	1000	1001	1010	1011
$z = 8$	0	1	2	3	4	5	6	7	10	11	12	13
$z = 10$	0	1	2	3	4	5	6	7	8	9	10	11
$z = 16$	0	1	2	3	4	5	6	7	8	9	A	B

Moderní počítače zpracovávají jen číselné údaje vyjádřené ve dvojkové soustavě. Je tomu tak proto, že logické obvody počítačů pracují pouze se dvěma stavy (zapnuto a vypnuto), které lze technicky snadno rozlišit. Nejmenší paměťovou jednotkou počítače je proto buňka, která dokáže uchovat právě takovou informaci (viz kap. 3.4, str. 40).

Dvojková soustava ale není jen výtvorem moderní doby. „Objevil“ ji už ve 3. tis. př. n. l. čínský císař Fú Xī (伏羲), za novodobé průkopníky jsou považováni anglický astronom, matematik a přírodovědec Thomas Harriot (* cca 1560, † 1621), španělský teolog, spisovatel a filozof Juan Caramuel y Lobkowitz (* 1606, † 1682) a především německý filozof, vědec a matematik Gottfried Wilhelm von Leibniz (* 1646, † 1716), kterého inspiroval soubor čínských textů *Yì Jīng*[10] (易經) datovaný do 2. tisíciletí př. n. l.

[10] Dílo známé také jako *Kniha proměn* se dodnes používá nejen jako orákulum, ale také jako filozofický a kosmologický systém, který hraje důležitou roli v taoismu, ze kterého ve velké míře vycházela čínská kultura.

2.2.2 Převody mezi soustavami

Hledáme-li ekvivalentní zápis daného čísla v číselné soustavě o jiném základu, musíme provést převod mezi oběma soustavami. Při převodu hledáme hodnoty koeficientů a_i v polynomu cílové soustavy. Z praktického hlediska jsou užitečné převody:

- ze soustavy o základu 10 do libovolné soustavy;
- z libovolné soustavy do soustavy o základu 10;
- mezi soustavami o mocninách stejného základu.

Převod z desítkové soustavy

Každé číslo vyjádřené v poziční soustavě (s výjimkou jedničkové) může mít celočíselnou a zlomkovou část. Tyto části jsou odděleny znakem nazvaným *řádová čárka* (v desítkové soustavě jí říkáme desetinná). V anglosaských zemích a ve většině programovacích jazyků se místo desetinné čárky používá tečka.[11] Při převodu čísel ze soustavy o základu 10 do soustavy o libovolném základu musíme číslo nejdříve rozložit na dvě části, které pak převádíme odlišným způsobem.

Celou část čísla, tj. koeficienty a_i, kde $0 \leq i \leq n$, převádíme tak, že postupně dělíme základem cílové soustavy a zapisujeme zbytky po dělení (operace modulo). Výpočet končí, když je výsledek celočíselného dělení roven nule. Ekvivalent celé části v cílové soustavě pak představují zbytky po dělení zapsané v opačném pořadí.

Zlomkovou část čísla, tj. koeficienty a_i, kde $-m \leq i < 0$, převádíme tak, že postupně násobíme základem cílové soustavy a zapisujeme celočíselné výsledky. Výpočet končí, když je desetinná část rovna nule. Ekvivalent zlomkové části v cílové soustavě pak představují celočíselné výsledky násobení.

Příklad 2.2

Převeďte číslo 149,75 z desítkové soustavy do soustavy o základu $z = 8$.

$$
\begin{array}{cccc}
 & (149{,}75)_{10} & & \\
 & \swarrow \quad \searrow & & \\
149 : 8 = 18 & a_0 = 5 & 0{,}75 \cdot 8 = 6{,}0 & a_{-1} = 6 \\
18 : 8 = 2 & a_1 = 2 & & \\
2 : 8 = 0 & a_2 = 2 & & \\
 & \searrow \quad \swarrow & & \\
 & (225{,}6)_8 & &
\end{array}
$$

Při převodu zlomkové části můžeme narazit na problém s přesností. Zatímco ve zdrojové soustavě je každé číslo přesné, v cílové soustavě může být neúplné. To se projeví při aritmetických operacích, kdy může být výsledek libovolné operace zatížen chybou. Pokud má číslo v cílové soustavě nekonečný nebo periodický rozvoj, musíme zvolit požadovanou přesnost na daný počet číslic zlomkové části.

[11] Jako oddělovač tisíců se potom v anglosaských zemích používá čárka, příp. zúžená mezera podobně jako v češtině. Číslo „1,234.56“ uvedené v anglickém textu potom odpovídá číslu „1 234,56“ v českém textu. V některých anglických textech (zejména technického charakteru) se lze setkat i s vynecháním 0 u čísel menších než 1, například „.4“ je totéž jako „0.4“. Způsob oddělování celé a zlomkové části, resp. jednotlivých řádů, proto vždy volíme podle jazyka, ve kterém své myšlenky zapisujeme.

Příklad 2.3

Převeďte číslo 265,1 z desítkové soustavy do soustavy o základu $z = 2$.

$$(265{,}1)_{10}$$

$$\begin{array}{rlrl}
265 : 2 = 132 & a_0 = 1 & 0{,}1 \cdot 2 = 0{,}2 & a_{-1} = 0 \\
132 : 2 = 66 & a_1 = 0 & 0{,}2 \cdot 2 = 0{,}4 & a_{-2} = 0 \\
66 : 2 = 33 & a_2 = 0 & 0{,}4 \cdot 2 = 0{,}8 & a_{-3} = 0 \\
33 : 2 = 16 & a_3 = 1 & 0{,}8 \cdot 2 = 1{,}6 & a_{-4} = 1 \\
16 : 2 = 8 & a_4 = 0 & 0{,}6 \cdot 2 = 1{,}2 & a_{-5} = 1 \\
8 : 2 = 4 & a_5 = 0 & 0{,}2 \cdot 2 = 0{,}4 & a_{-6} = 0 \\
4 : 2 = 2 & a_6 = 0 & 0{,}4 \cdot 2 = 0{,}8 & a_{-7} = 0 \\
2 : 2 = 1 & a_7 = 0 & 0{,}8 \cdot 2 = 1{,}6 & a_{-8} = 1 \\
1 : 2 = 0 & a_8 = 1 & \vdots &
\end{array}$$

$$(100001001{,}00011001\ldots)_2$$

Převod do desítkové soustavy

Při převodu čísel ze soustavy o libovolném základu do desítkové soustavy využijeme polynomiálního zápisu čísla. Provedeme-li s polynomem naznačené aritmetické operace, dostaneme bez ohledu na základ soustavy vždy hodnotu tohoto čísla v desítkové soustavě. Dostaneme-li převodem číslo s periodickým nebo nekonečným rozvojem, opět musíme zvolit požadovanou přesnost.

Příklad 2.4

Převeďte čísla $(314)_5$, $(\mathrm{F1})_{16}$, $(3{,}21)_4$ a $(2{,}1)_3$ do desítkové soustavy.

$$\begin{aligned}
(314)_5 &= 3 \cdot 5^2 + 1 \cdot 5^1 + 4 \cdot 5^0 = 3 \cdot 25 + 1 \cdot 5 + 4 \cdot 1 = (84)_{10} \\
(\mathrm{F1})_{16} &= 15 \cdot 16^1 + 1 \cdot 16^0 = 15 \cdot 16 + 1 \cdot 1 = (241)_{10} \\
(3{,}21)_4 &= 3 \cdot 4^0 + 2 \cdot 4^{-1} + 1 \cdot 4^{-2} = 3 \cdot 1 + 2 \cdot 0{,}25 + 1 \cdot 0{,}0625 = (3{,}5625)_{10} \\
(2{,}1)_3 &= 2 \cdot 3^0 + 1 \cdot 3^{-1} = 2 \cdot 1 + 1 \cdot 0{,}3333\ldots = (2{,}3333\ldots)_{10}
\end{aligned}$$

Hornerův polynom

Úspornější převod nabízí **Hornerův**[12] **polynom**, ve kterém upravíme polynom tak, aby se základ soustavy vyskytoval vždy v první mocnině. Opět musíme zvlášť vyjádřit celou část C a zlomkovou část Z:

$$C = \Big(\big((a_n \cdot z + a_{n-1}) \cdot z + \ldots + a_2\big) \cdot z + a_1\Big) \cdot z + a_0$$

$$Z = \Big(\big((a_{-m} \cdot z^{-1} + a_{-m+1}) \cdot z^{-1} + \ldots + a_{-2}\big) \cdot z^{-1} + a_{-1}\Big) \cdot z^{-1}$$

Příklad 2.5

Převeďte pomocí Hornerova polynomu číslo $(\mathrm{C7A})_{16}$ do desítkové soustavy.

$$(\mathrm{C7A})_{16} = (12 \cdot 16 + 7) \cdot 16 + 10 = (192 + 7) \cdot 16 + 10 = 199 \cdot 16 + 10 = 3194$$

[12] William George Horner (* 1786, † 1837) byl britský matematik. Metodu, kterou používal k řešení algebraických rovnic, však již o 500 let dříve znalí čínští matematikové Qín Jiǔsháo (秦九韶; * 1202, † 1261) a Zhū Shìjié (朱世傑; * 1249, † 1314).

Příbuzné soustavy

Specialitou jsou převody mezi soustavami o různých základech, z nichž ani jeden není deset. Jednou z osvědčených možností je převod ve dvou krocích přes desítkovou soustavu, ale v určitých případech existuje i jednodušší cesta. Jedná se o situace, kdy obě soustavy mají jako základ mocninu stejného čísla, například 2^n. Takové soustavy mají pro použití v počítači hlavní význam. Velmi efektivně lze mezi takovými soustavami převádět, když si uvědomíme, že jedna číslice v čísle vyjádřeném v soustavě o základu z^n reprezentuje n číslic v čísle vyjádřeném v soustavě o základu z.

Příklad 2.6

Převeďte číslo $(100011100101101)_2$ do soustavy o základu $z = 16$. Protože $16 = 2^4$, jedna číslice šestnáctkové soustavy odpovídá čtyřem číslicím ve dvojkové soustavě. Rozdělíme proto číslo zprava na jednotlivé čtveřice a každou pak převádíme samostatně.

$$\begin{array}{cccc} (\ 0100 & 0111 & 0010 & 1101\ )_2 \\ (\ \ \ 4 & \ \ \ 7 & \ \ \ 2 & \ \ \ \mathrm{D}\ )_{16} \end{array}$$

Prefix soustavy

Některé aplikace (např. vědecká kalkulačka) podporují zápis čísel v nedesítkových soustavách. Pokud ale napíšeme nedesítkové číslo tam, kde se očekává desítkové, musíme uvést *prefix* soustavy. Obvykle se používá `0b` pro dvojkové číslo, `0o` nebo `0` pro osmičkové číslo, `0d` pro desítkové číslo a `0x` pro šestnáctkové číslo. Tvar prefixu se může v jednotlivých aplikacích lišit, je proto vhodné vždy nejdříve prostudovat nápovědu k aplikaci.

2.2.3 Aritmetické operace

Aritmetické operace lze bez obtíží provádět v soustavě o libovolném základu. S trochou nadsázky můžeme tvrdit, že kdo chápe principy počítání v desítkové soustavě, je schopen počítat v libovolné jiné soustavě.

Sčítání

Čísla v soustavě o základu z sečteme tak, že postupujeme od nejnižšího řádu k nejvyššímu a pro každý řád určíme hodnotu součtu, kterou vyjádříme jako $a \cdot z + b$, kde $a \in \mathbb{Z}_0^+$ je počet celých násobků základu a $b \in \mathbb{N}$ je zbytek. Zřejmě musí platit, že $b < z$. Hodnotu b potom zapíšeme do výsledku jako součet daného řádu, hodnotu a přeneseme do vyššího řádu. Zároveň platí, že součet dvou n-ciferných čísel nikdy nebude mít více než $n + 1$ číslic.

Příklad 2.7

Sečtěte v soustavě o základu $z = 10$ čísla 679 a 138.

$$\begin{array}{rll} (679)_{10} & \text{součet v řádu 0:} & 17 = 1 \cdot 10 + 7 \\ (138)_{10} & \text{součet v řádu 1:} & 11 = 1 \cdot 10 + 1 \\ \hline (817)_{10} & \text{součet v řádu 2:} & 8 = 0 \cdot 10 + 8 \end{array}$$

Příklad 2.8

Sečtěte v soustavě o základu $z = 6$ čísla 423, 1341 a 521.

$$\begin{array}{rll} (423)_{6} & \text{součet v řádu 0:} & 5 = 0 \cdot 6 + 5 \\ (1341)_{6} & \text{součet v řádu 1:} & 8 = 1 \cdot 6 + 2 \\ (521)_{6} & \text{součet v řádu 2:} & 13 = 2 \cdot 6 + 1 \\ \hline (3125)_{6} & \text{součet v řádu 3:} & 3 = 0 \cdot 6 + 3 \end{array}$$

Odčítání

Při odčítání postupujeme opět od nejnižšího řádu k nejvyššímu a pro každý řád určíme hodnotu rozdílu, kterou vyjádříme jako $-a \cdot z + b$, kde $a \in \mathbb{Z}_0^+$ je počet celých násobků základu, $b \in \mathbb{N}$ je zbytek a opět musí být $b < z$. Hodnotu b zapíšeme jako rozdíl daného řádu, hodnotu a vrátíme do vyššího řádu, odkud jsme si ji „vypůjčili“.

Příklad 2.9

Odečtěte v soustavě o základu $z = 8$ od čísla 42714 číslo 17043.

$$\begin{array}{r} (42714)_8 \\ -\,(17043)_8 \\ \hline (23651)_8 \end{array} \qquad \begin{array}{rr} \text{rozdíl v řádu 0:} & 1 = -0 \cdot 8 + 1 \\ \text{rozdíl v řádu 1:} & -3 = -1 \cdot 8 + 5 \\ \text{rozdíl v řádu 2:} & 6 = -0 \cdot 8 + 6 \\ \text{rozdíl v řádu 3:} & -5 = -1 \cdot 8 + 3 \\ \text{rozdíl v řádu 4:} & 2 = -0 \cdot 8 + 2 \end{array}$$

Násobení probíhá na stejném principu, jen je nutné dávat pozor na to, že dílčí výsledky jsou ve stejné soustavě jako činitelé. Dělení lze provádět také, ale prakticky se nepoužívá, dokonce ani počítač nedělí.

Operace v počítači

Počítač ve skutečnosti umí pouze sčítat a všechny ostatní operace si na sčítání převádí. Součet probíhá tak, jak bylo popsáno výše, pouze s tím rozdílem, že paměť počítače je limitována počtem bitů. Odčítání probíhá jako přičítání opačného čísla, například výpočet $47 - 35$ se ve skutečnosti realizuje jako součet $47 + (-35)$. Násobení je převedeno na opakované sčítání, dělení se zbytkem se realizuje jako opakované „odčítání“. Bližší informace lze nalézt v kapitole věnované reprezentaci čísel v paměti počítače (viz str. 54).

Kontrolní otázky

5. Jakým způsobem je u pozičních číselných soustav určena hodnota každé číslice?
6. Jaký je rozdíl mezi poziční nulou a nulou ve významu „žádné množství“?
7. Odkud pocházejí arabské číslice, které používáme pro zápis čísel dnes?
8. Kolik pozičních číselných soustav existuje a v kterých může existovat číslo 176?
9. Jakými způsoby lze vyjádřit číslo v poziční soustavě?
10. Které číselné soustavy jsou důležité pro informatiku?
11. Jakým způsobem lze převést číslo z jedné soustavy do druhé?
12. K čemu lze použít Hornerův polynom?
13. Jaká pravidla platí pro sčítání a odčítání v libovolné soustavě?
14. Jak jsou aritmetické operace realizovány v paměti počítače?

2.3 Příklady na procvičení

Příklad 2.10 Vyjádřete daná čísla zapsaná v římské soustavě pomocí desítkové soustavy:

a) XXVI
b) CXCVIII
c) DLIV
d) CMXCVI
e) MMCMXCIX
f) MMXI
g) MMMMLXX
h) MMMMMMMMMMCMXCVIII

Příklad 2.11 Vyjádřete daná čísla zapsaná v desítkové soustavě pomocí římské soustavy:

a) 58
b) 93
c) 139
d) 452
e) 901
f) 999
g) 1458
h) 3796

Příklad 2.12 Vyjádřete čísla s koeficienty a_i o základu z pozičním zápisem i polynomem:

a) $z = 9$, $a_0 = 1$, $a_1 = 7$, $a_2 = 4$
b) $z = 8$, $a_0 = 5$, $a_1 = 3$, $a_2 = 1$, $a_{-1} = 5$
c) $z = 2$, $a_0 = 1$, $a_3 = 1$
d) $z = 10$, $a_{-1} = 4$, $a_{-2} = 1$, $a_{-4} = 3$
e) $z = 16$, $a_0 = 13$, $a_1 = 8$, $a_2 = 10$

Příklad 2.13 Převeďte daná čísla z desítkové soustavy do soustavy o základu z:

a) 103, $z = 7$
b) 261,25, $z = 2$
c) 417,82, $z = 16$
d) 309,03, $z = 8$
e) 883,75, $z = 3$
f) 827, $z = 10$

Příklad 2.14 Převeďte daná čísla ze soustavy o základu z do desítkové soustavy:

a) 420, $z = 5$
b) 10011001, $z = 2$
c) C13E,A8, $z = 16$
d) 30424, $z = 8$
e) 461,36, $z = 7$
f) 737, $z = 10$

Příklad 2.15 Převeďte daná čísla do desítkové soustavy s využitím Hornerova polynomu:

a) 561,21, $z = 8$
b) 101110,01, $z = 2$
c) ABC12,C4, $z = 16$
d) 761,43, $z = 8$
e) C000,0B, $z = 13$
f) ABBA, $z = 16$

Příklad 2.16 Převeďte daná čísla ze soustavy o základu z_1 do soustavy o základu z_2:

a) 1011101, $z_1 = 2$, $z_2 = 8$
b) 3102, $z_1 = 4$, $z_2 = 2$
c) 11110100101, $z_1 = 2$, $z_2 = 16$
d) 1C3D, $z_1 = 16$, $z_2 = 2$
e) 1002212, $z_1 = 3$, $z_2 = 9$
f) 110111001,10111, $z_1 = 2$, $z_2 = 16$

Příklad 2.17 Zjistěte výsledek operace mezi dvěma čísly v soustavě o základu z:

a) 101100101 + 1101001011, $z = 2$
b) A1B2 + F3E4, $z = 16$
c) 74321 + 625, $z = 8$
d) 322 + 123, $z = 4$
e) 40322 − 2004, $z = 5$
f) 364 − 172, $z = 8$
g) 718 − 366, $z = 9$
h) C6 − A3, $z = 16$
i) 100 · 222, $z = 3$
j) ABCD · 4321, $z = 16$
k) 1001 · 1110, $z = 2$
l) 3325 · 21077, $z = 8$

Tabulka 2.3: *Základní číslice poziční desítkové soustavy v různých částech světa*

0	1	2	3	4	5	6	7	8	9	Výskyt
〇*	一	二	三	四	五	六	七	八	九	čínské, japonské a korejské číslice
〇	丨	亓	亓	乂	𠄌	亠	亠	亠	文	čínské číslice sūzhōu [a]
영*	일	이	삼	사	오	육	칠	팔	구	korejské číslice [b]
	Α′	Β′	Γ′	Δ′	Ε′	Ϛ′	Ζ′	Η′	Θ′	řecké číslice [c]
	א	ב	ג	ד	ה	ו	ז	ח	ט	hebrejské číslice
٠	١	٢	٣	٤	٥	٦	٧	٨	٩	arabské číslice
۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	perské číslice
০	১	২	৩	৪	৫	৬	৭	৮	৯	indické číslice bráhmí
००	१	२	३	४	५	६	७	८	९	indické číslice dévanágarí
૦	૧	૨	૩	૪	૫	૬	૭	૮	૯	gudžarátské číslice
੦	੧	੨	੩	੪	੫	੬	੭	੮	੯	paňdžábské číslice gurmukhí
০	১	২	৩	৪	৫	৬	৭	৮	৯	káthijávárské číslice
০	১	২	৩	৪	৫	৬	৭	৮	৯	bengálské číslice
০	১	২	৩	৪	৫	৬	৭	৮	৯	urijské číslice
০	১	২	৩	৪	৫	৬	৭	৮	৯	manipurské číslice
౦	౧	౨	౩	౪	౫	౬	౭	౮	౯	telugské číslice
೦	೧	೨	೩	೪	೫	೬	೭	೮	೯	kannadské číslice
০	১	২	৩	৪	৫	৬	৭	৮	৯	santalské číslice
௦	௧	௨	௩	௪	௫	௬	௭	௮	௯	tamilské číslice
൦	൧	൨	൩	൪	൫	൬	൭	൮	൯	malajálamské číslice
᠐	᠑	᠒	᠓	᠔	᠕	᠖	᠗	᠘	᠙	mongolské číslice
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	tibetské číslice
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	lepčské číslice
᥆	᥈	ᥑ	ᥒ	ᥓ	ᥔ	ᥕ	ᥖ	ᥗ	ᥘ	limbské číslice
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	čamské číslice
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	khmerské číslice
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	thajské číslice
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	laoské číslice
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	thajské číslice tham
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	thajské číslice hora
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	thajské číslice lü
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	barmské číslice
𖺀	𖺁	𖺂	𖺃	𖺄	𖺅	𖺆	𖺇	𖺈	𖺉	šanské číslice
๐	๑	๒	๓	๔	๕	๖	๗	๘	๙	karenské číslice
᭐	᭑	᭒	᭓	᭔	᭕	᭖	᭗	᭘	᭙	balijské číslice
ꦐ	ꦑ	ꦒ	ꦓ	ꦔ	ꦕ	ꦖ	ꦗ	ꦘ	ꦙ	jávanské číslice
᮰	᮱	᮲	᮳	᮴	᮵	᮶	᮷	᮸	᮹	sundajské číslice
𞅀	𞅁	𞅂	𞅃	𞅄	𞅅	𞅆	𞅇	𞅈	𞅉	mandingské číslice
𞄰	𞄱	𞄲	𞄳	𞄴	𞄵	𞄶	𞄷	𞄸	𞄹	somálské číslice
ᘜ	ᘝ	ᘞ	ᘟ	ᘠ	ᘡ	ᘢ	ᘣ	ᘤ	ᘥ	číslice jazyka vai
	፩	፪	፫	፬	፭	፮	፯	፰	፱	etiopské číslice

Poznámky:

* nula se též vyjadřuje znakem 零 v čínštině a znakem 공 v korejštině

[a] též *huāmǎ*, *cǎomǎ*, *jīngzǐmǎ*, *fānzǐmǎ* nebo *shāngmǎ*, v současnosti slouží pouze k vyjádření cen zboží v obchodech

[b] v současnosti slouží pouze k vyjádření datumu, měny, telefonních čísel apod.

[c] v současnosti slouží pouze jako pořadová čísla (podobně jako římské číslice)

Shrnutí

Tato kapitola nás seznámila s různými způsoby zápisu čísel v číselných soustavách. Zmínili jsme historicky důležité nepoziční soustavy a poté poziční soustavy používané v současnosti. Ukázali jsme si, jak vyjádřit libovolné číslo v libovolné číselné soustavě a jakým způsobem lze převádět mezi různými soustavami. V závěru jsme se naučili principy sčítání a odčítání v libovolné soustavě a dozvěděli jsme se, jak provádí aritmetické operace počítač.

Slovníček pojmů

Celá část čísla: řády reprezentované koeficienty a_i, kde $0 \leq i \leq n$.
Číselná soustava: způsob reprezentace čísel.
Číslice: elementární symboly používané k zápisu čísla.
Číslo: hodnota vyjadřující určitý počet základních jednotek.
Hornerův polynom: zjednodušená podoba zápisu čísla, kde se základ soustavy vyskytuje vždy v první mocnině.
Jedničková soustava: speciální případ poziční soustavy, ve které je každé číslo vyjádřeno opakováním jednoho stejného symbolu.
Nepoziční soustava: číselná soustava, ve které je váha každé číslice pevně stanovena bez ohledu na jejím umístění v čísle.
Polyadická soustava: číselná soustava, ve které lze číslo vyjádřit součtem mocnin základu dané soustavy vynásobených hodnotou příslušných číslic.
Poziční soustava: číselná soustava, ve které je váha číslice dána jejím umístěním v čísle.
Základ soustavy: počet elementárních symbolů, které jsou v soustavě dostupné.
Zlomková část čísla: řády reprezentované koeficienty a_i, kde $-m \leq i < 0$.

Řešení kontrolních otázek

1. *Jaké výhody a nevýhody mají nepoziční číselné soustavy?*

Výhodou nepozičních soustav je snadné sčítání a odčítání, nevýhodou je dlouhý zápis čísel, ve kterých počet číslic výrazně převyšuje hodnotu největší číslice.

2. *Na jakém principu je založena řecká iónská číselná soustava a jaký je její původ?*

Řecká iónská číselná soustava se vyznačuje tím, že k vyjádření číslic používá písmena alfabety. Je odvozena z koptštiny, posledního vývojového stádia egyptštiny.

3. *Které symboly používá soustava římských číslic a co vyjadřují?*

Soustava římských číslic obsahuje sedm symbolů s následujícím významem: I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1 000. Pro zkrácení zápisu existují složené symboly s následujícím významem: IV = 4, IX = 9, XL = 40, XC = 90, CD = 400, CM = 900.

4. *Jak se v soustavě římských číslic vyjádří hodnoty 135, 298, 444, 705, 1999 a 2019?*

135 = CXXXV, 298 = CCXCVIII, 444 = CDXLIV, 705 = DCCV, 1999 = MCMXCIX
(ne MXMIX ani MIM, neboť XM ani IM nejsou povolené složené symboly), 2019 = MMXIX.

5. Jakým způsobem je u pozičních číselných soustav určena hodnota každé číslice?

Hodnota každé číslice (její váha) v čísle je u poziční číselné soustavy dána umístěním číslice na příslušném řádu čísla.

6. Jaký je rozdíl mezi poziční nulou a nulou ve významu „žádné množství“?

Poziční nula je v poziční soustavě nezbytná, protože vyplňuje „prázdné místo“ u neobsazeného řádu. Nula ve významu „žádné množství“ je jednociferným číslem, které bylo používáno mnohem později.

7. Odkud pocházejí arabské číslice, které používáme pro zápis čísel dnes?

Arabské (správně indo-arabské) číslice vznikly v Indii někdy na přelomu letopočtu. Jsou odvozeny od číslic v indickém písmu bráhmí. Přibližně v 10. století se pod názvem ghubār rozšířily do Evropy a do arabských zemí v severozápadní Africe (odtud označení západoarabské).

8. Kolik pozičních číselných soustav existuje a v kterých může existovat číslo 176?

Existuje nekonečně mnoho pozičních číselných soustav. Číslo 176 může existovat v libovolné soustavě o základu $z \geq 8$.

9. Jakými způsoby lze vyjádřit číslo v poziční soustavě?

Pro vyjádření čísel v poziční soustavě můžeme použít poziční zápis nebo ekvivalentní polynomiální zápis.

10. Které číselné soustavy jsou důležité pro informatiku?

Všechny moderní počítače pracují ve dvojkové soustavě ($z = 2$), v informatice se často setkáme také se soustavou osmičkovou ($z = 8$) a šestnáctkovou ($z = 16$).

11. Jakým způsobem lze převést číslo z jedné soustavy do druhé?

Při převodu z desítkové soustavy rozdělíme číslo na celou a zlomkovou část a každou převádíme samostatně. Celou část postupně dělíme základem soustavy a zapisujeme zbytky, které pozpátku tvoří číslice celé části výsledného čísla. Zlomkovou část postupně násobíme základem soustavy a zapisujeme celočíselné výsledky, které tvoří číslice zlomkové části výsledného čísla. Při převodu do desítkové soustavy vyjádříme hodnotu polynomu tím, že provedeme naznačené aritmetické operace.

12. K čemu lze použít Hornerův polynom?

Hornerův polynom usnadňuje převod čísel do desítkové soustavy tím, že základ soustavy je uveden vždy v první mocnině.

13. Jaká pravidla platí pro sčítání a odčítání v libovolné soustavě?

Postupujeme od nejnižšího řádu k nejvyššímu a u každého určíme součet (resp. rozdíl), který vyjádříme jako $a \cdot z + b$ (resp. $-a \cdot z + b$). Hodnotu proměnné b zapíšeme do výsledku, hodnotu proměnné a přeneseme (resp. vrátíme) do vyššího řádu.

14. Jak jsou aritmetické operace realizovány v paměti počítače?

Sčítání standardním postupem na určitém počtu bitů, odčítání jako přičítání opačného čísla, násobení jako opakované sčítání, dělení se zbytkem jako opakované „odčítání“.

3 Teorie informace a informatika

Úvod

Žijeme v době informační exploze. Každý den jsme nuceni zpracovávat obrovské množství informací. K tomu je zapotřebí rozumět pojmu informace, dokázat rozlišit informace od dat a umět změřit množství informace ve zprávě. Tato kapitola poskytuje odpovědi na otázky, které vyvstanou při řešení uvedených problémů, a uvede je do souvislosti s reprezentací dat v paměti počítače.

Cíl

Cílem této kapitoly je osvěžit kombinatoriku a výpočet pravděpodobnosti, umět spočítat množství vlastní informace konkrétní realizace nějakého jevu a entropii určitého subjektu v určité situaci a také znát jednotky informace a umět převádět mezi jejich násobky podle platných norem.

Návaznosti

Pro studium této kapitoly je nutná orientace v matematice, zejména v oblasti funkcí, kombinatoriky a pravděpodobnosti, a to na úrovni střední školy.

Studijní čas

Čas potřebný k nastudování této kapitoly je asi 5 hodin.

3.1 Základní pojmy

Mnohokrát bylo uvedeno, že počítač je stroj na zpracování dat. Co ale jsou ta data? A jaký je rozdíl mezi daty a informacemi? Pojďme si nejprve udělat pořádek v pojmech (obr. 3.1).

Údaje

Hodnoty získané měřením, pozorováním nebo pouhým zaznamenáním reálné skutečnosti nazýváme **údaje**. Pro zpracování je nutné údaje nejprve zjistit a následně zaznamenat. Způsobů zaznamenání je mnoho, například údaj o zobrazené hodnotě na displeji digitální váhy můžeme zaznamenat v podobě čísla v desítkové soustavě v kilogramech, stejně dobře ale můžeme použít jinou jednotku (i když váha člověka vyjádřená v tunách nemá pravděpodobně ten správný psychologický efekt) nebo jinou číselnou soustavu (například římská čísla).

Data

Jakékoli vyjádření skutečnosti schopné přenosu, uchování, interpretace či zpracování nazýváme **data**. Jedná se tedy o kvalitativně nebo kvantitativně formalizované údaje. V okamžiku, kdy si údaje z displeje váhy dohodnutým způsobem poznačíme, stávají se z nich data, která můžeme dále zpracovávat (například si můžeme vést statistiku přírůstků a úbytků hmotnosti). Přestože sama o sobě jsou data nehmotná, pro jejich uložení a zpracování nutně potřebujeme hmotné médium. V prvotní fázi se spokojíme s médiem v podobě papíru, efektivnější ale bude počítačové zpracování, zejména při větších objemech dat.

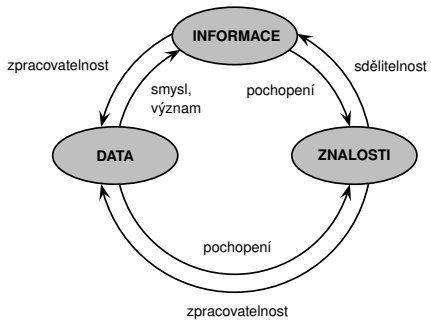
Pro počítačové zpracování je nutné data popsat formálně správným způsobem. K tomuto účelu byl zaveden **datový typ**. Přidělením datového typu určujeme *velikost prostoru v paměti*, na kterém budou data uložena v binárním kódu, a *způsob interpretace* tohoto uloženého binárního kódu. Z těchto dvou vlastností vyplývá množina povolených hodnot, ze způsobu interpretace povolených hodnot pak dále vyplývá i množina povolených operací. Datovým typem tedy rozumíme množinu hodnot a množinu povolených operací.

Informace

Nyní se dostáváme k pojmu **informace**. Informace vznikají zpracováním dat. Jsou složeny z dat, která snižují neznalost a vyvolávají změnu stavu či chování příjemce. Změna stavu po přijetí zprávy je tím větší, čím větším „překvapením" je obdržená informace pro příjemce. Je tedy zřejmé, že každou informaci lze považovat za součást dat, ale zdaleka ne všechna data musejí nést informaci. Množství informace ve zprávě je navíc relativní vzhledem k určitému příjemci a určité situaci. Zpráva o aktuální hmotnosti bude informací jen pro toho, kdo o takovou zprávu měl zájem a nebyl si výsledkem předem jist.

Znalosti

Kolekci pojmů doplníme ještě o **znalosti**. Ty představují ucelený komplex informací o nějaké objektivní realitě. Jsou výsledkem poznávacího procesu, předpokladem uvědomělé činnosti a umožňují porozumět skutečnosti. Například ze zprávy o aktuální hmotnosti můžeme při porovnání s předchozími hodnotami usoudit, zda jsme přibrali, nebo shodili.



Obrázek 3.1: *Souvislost mezi základními pojmy*

Definice informace

Pojem **informace** dosud nebyl pevně ukotven. Existuje mnoho definic podle toho, co jejich autoři považovali za důležité. Samotný pojem pochází z latinského *informo*, což znamená přinášet zprávu, oznámení, poučení. Americký matematik, filozof a zakladatel kybernetiky Norbert Wiener (* 1894, † 1964) ji definoval takto: *„Informace je název pro obsah toho, co se vymění s vnějším světem, když se mu přizpůsobujeme a působíme na něj svým přizpůsobováním."* Namátkově vyberme několik dalších příkladů definice téhož pojmu:

- Terminologický slovník informatiky: *„Informace je obsah jakéhokoli oznámení, údaje o čemkoli, s určením pro přenos v prostoru a čase. V nejširším slova smyslu je to obsah vztahů mezi materiálními objekty, projevující se změnami těchto objektů."*
- Slovník cizích slov: *„Informace je obsah zprávy, sdělení, objasnění, vysvětlení, poučení."*
- Oborová encyklopedie: *„Informace jsou údaje, čísla, znaky, povely, instrukce, příkazy, zprávy apod. Za informace považujeme také podněty a vjemy přijímané a vysílané živými organismy."*
- Ilustrovaný encyklopedický slovník: *„Informace – zpráva, sdělení, údaj, informování, poučení. Má přibližně dvojí význam: kvalitativní ve smyslu zprávy, sdělení jejího obsahu; kvantitativní v teorii informace a kybernetice."*

Při studiu informace se uplatňuje mnoho různých přístupů. V kybernetických systémech jsou rozhodující informační procesy, tj. procesy přijímání, uchovávání, zpracování a vydání informace. I když informace není ani fyzikální objekt, látka nebo energie, je ji možno vytvářet, přenášet a zpracovávat materiálními objekty.

Teorie informace

Již v prvních publikacích zabývajících se informacemi byly studovány otázky kvantifikace a přenosu informací. Jednu takovou publikaci napsal již v roce 1948 americký elektronik a matematik Claude Elwood Shannon (* 1916, † 2001), zakladatel **teorie informace**, jehož objevy přinesly překvapivé výsledky. Ukázal, že pro měření množství informace obsaženého v kvalitativně různých typech přenosu lze použít se stejným úspěchem jedné a téže míry. Nejdůležitější témata v teorii informace jsou tato:

- zavedení míry množství informace;
- kódování a dekódování zpráv;
- přenos zpráv.

Informatika

Informatika je vědní obor zabývající se informacemi z hlediska kvalitativního, tj. jejich získáváním, zpracováním a hodnocením ve vztahu k příjemci. Základními prostředky používanými v informatice jsou počítače, zejména jejich aplikační programové vybavení. K zaznamenání informací může sloužit textový procesor, ke zpracování informací použijeme tabulkový kalkulátor, k archivaci kartotéku, ke zveřejnění prezentační software a k předávání informací poštovního klienta.

Informatika existovala i dříve, jen se tak nenazývala – vzpomeňme na hliněné destičky, papyrus, knihy apod. Dnes informatika představuje široký a progresivní obor. Ve vyspělých zemích jsou informace stejně ceněným hospodářským zdrojem jako kapitál, půda, suroviny a pracovní síla. V oblasti získávání, zpracování a využití informací pracuje více než polovina zaměstnaného obyvatelstva těchto zemí. Studium tohoto oboru tedy nabízí nejen zajímavý program, ale i dobrou perspektivu uplatnění v praxi.

Informační systémy

Dominantní úlohu ve studiu a hlavně využívání informatiky dnes hrají informační systémy (i když samozřejmě existovaly i dříve). Je-li systém definován jako komplex prvků a vazeb ve vzájemné interakci, pak **informační systém** je dynamický systém, jehož vazby tvoří informace a prvky představují místa jejich transformace. Informační systém slouží k poskytování potřebných informací v požadovaném rozsahu, lhůtách, podrobnostech i formě. Mezi jeho dílčí úlohy patří zejména sběr informací, jejich přenos, redukce, archivace, zpracování a distribuce. S množstvím a potřebou pružně využívat informací významně stoupla také nutnost využívání informačních a komunikačních technologií. Jednotlivé složky počítačového informačního systému jsou popsány v tab. 3.1 na str. 36.

Kontrolní otázky

1. Co jsou údaje, data, informace a znalosti?
2. Kdy se z dat stávají informace?
3. Čím se zabývá teorie informace a čím informatika?
4. Co je informační systém a k čemu slouží?

Tabulka 3.1: *Složky informačního systému*

Prvek	Význam
Hardware	Tvoří fyzickou vrstvu informačního systému. To zahrnuje počítače, jejich periferní zařízení, počítačové sítě a další infrastrukturu.
Software	Zahrnuje základní softwarové vybavení (operační systém a ovladače zařízení) a aplikační software. Aplikační software tvoří programy používané pro podporu podnikových funkcí a procesů.
Databáze	Představují organizovanou sbírku faktů a informací. Často jsou považovány za nejdůležitější a nejcennější část informačního systému.
Telekomunikace	Umožňují přenos signálů přes počítačovou síť.
Lidé	Protože lidé obvykle tvoří rozdíl mezi úspěchem a neúspěchem, mohou být považováni za klíčovou součást informačního systému. Účelem informačního systému je poskytovat informace rozdílným skupinám uživatelů. Tyto skupiny tvoří manažeři a jiní interní zaměstnanci (úředníci, technici, prodejci), ale také uživatelé z vnějšího prostředí (zákazníci, partneři apod.). Úspěch informačního systému obvykle závisí na akceptaci operací a výstupů jeho uživateli. Na úspěchu systému se podílejí také profesionálové jako systémoví analytici, návrháři, programátoři nebo IT manažeři.
Procedury	Zahrnují strategie, politiky, metody a pravidla pro používání informačního systému a popisují úlohy, které všichni uživatelé provádějí. Procedury jsou obvykle popsány v manuálech nebo jiných referenčních příručkách.

3.2 Měření množství informace

Formalizace v teorii informace je založena na pravděpodobnostně-statistickém vyjádření informace. Claude Elwood Shannon vyšel z toho, že zprávy, které se přenášejí pomocí nějakého zařízení, patří do kategorie náhodných jevů. Proto svoji formalizaci založil na následující definici: *„Informace je míra množství neurčitosti nebo nejistoty o nějakém náhodném ději odstraněná realizací tohoto děje.“*

Informační entropie

Je zřejmé, že každá realizace náhodně probíhajícího děje odstraňuje část nejistoty o tomto ději a poskytuje o něm určité množství znalostí. Z formalizačního hlediska je tedy nutné postihnout míru neurčitosti nějakou kvantifikovanou veličinou. Shannon uto veličinu zavedl a pojmenoval ji **entropie** (název si „vypůjčil“ z termodynamiky).

Základním problémem při výpočtu informační entropie je kvantifikace rozšíření okruhu znalostí příjemce. To je spojeno s individuálními vlastnostmi příjemce, takže jedna a tatáž zpráva může někomu zvýšit okruh znalostí a někomu jinému nemusí. Využijeme poznatků z pravděpodobnosti. Víme, že každá zpráva poskytuje informace o nějaké události, která má určitou pravděpodobnost. Čím větší je ta pravděpodobnost, tím snazší je pro nás odhadnutí výsledku a tím menší informaci také ve zprávě obdržíme.

Jev a jeho realizace

Pod pojmem **jev** budeme chápat libovolný náhodný proces s n možnými realizacemi, například tah sportky, účast studentů na přednášce nebo semafor na křižovatce. Každý projev takového jevu nazveme **realizací**, například vytažení konkrétních čísel, konkrétní počet osob v přednáškové místnosti nebo svítící barvu na semaforu.

Užitečné vzorce

Je zřejmé, že v teorii informace se při určování pravděpodobnosti realizace nějakého jevu neobejdeme bez kombinatorické matematiky. Uveďme proto nyní stručný přehled užitečných vzorců.

Variace

Variace k-té třídy z n prvků je každá uspořádaná k-tice vytvořená z celkového počtu n prvků, přičemž při výběru *záleží na pořadí jednotlivých prvků.*

Variace bez opakování je k-tice sestavená z n prvků tak, že žádný z prvků se ve výběru neopakuje. Počet k-členných variací z n prvků (kde $k \leq n$) vypočítáme takto:

$$V_k(n) = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-k+1) = \frac{n!}{(n-k)!}$$

Variace s opakováním je k-tice sestavená z n prvků tak, že každý prvek se v ní vyskytuje nejvýše k-krát. Počet k-členných variací s opakováním z n prvků vypočítáme takto:

$$V'_k(n) = n^k$$

Permutace

Permutace n-prvkové množiny je jedno z možných uspořádání prvků této množiny, přičemž výsledná uspořádaná n-tice má *stejný počet prvků jako původní množina.* Obvykle uvažujeme **permutace bez opakování**. Počet všech možných výběrů z n-prvkové množiny (pro $n \in \mathbb{N}_0$) vypočítáme takto:

$$P(n) = n!$$

Někdy bereme v úvahu také **permutace s opakováním**, ve kterém se některé prvky vyskytují vícekrát. Počet takových permutací z n-prvkové množiny, kde se jednotlivé prvky opakují k-krát, pak vypočítáme takto:

$$P'(k_1, k_2, \ldots, k_n) = \frac{(k_1 + k_2 + \ldots + k_n)!}{k_1! \cdot k_2! \cdot \ldots \cdot k_n!}$$

Kombinace

Kombinace k-té třídy z n prvků je každá neuspořádaná k-tice vytvořená z celkového počtu n prvků, přičemž při výběru *nezáleží na pořadí jednotlivých prvků.*

Kombinace bez opakování je k-tice sestavená z n prvků tak, že žádný z prvků se ve výběru neopakuje. Počet k-členných kombinací z n prvků vypočítáme takto:

$$C_k(n) = \binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$$

Kombinace s opakováním je k-tice sestavená z n prvků tak, že každý prvek se ve výběru může objevit nejvýše k-krát. Počet k-členných kombinací s opakováním z n prvků vypočítáme takto:

$$C'_k(n) = \binom{n+k-1}{k} = \frac{(n+k-1)!}{k! \cdot (n-1)!}$$

Výběr funkce

Abychom mohli vyčíslit množství informace ve zprávě, potřebujeme najít vhodnou matematickou funkci. Uvažujme jev X, který má celkem n možných realizací, jejichž pravděpodobnosti jsou postupně $p(x_1), p(x_2), \dots, p(x_n)$. Množství informace pak vypočteme jako $f(n)$. Taková funkce musí mít tyto požadované vlastnosti:

1. Je-li $n = 1$, pak se jedná o jev jistý a $p(x) = 1$. Pak požadujeme, aby $f(n) = 0$.
2. Jsou-li X a Y dva jevy, které se realizují současně (paralelně, bez vzájemných vazeb), pak $p(x, y) = p(x) \cdot p(y)$. V takovém případě požadujeme, aby míra neurčitosti složeného jevu XY byla dána součtem neurčitostí jevů X a Y, tj. $f(n_x, n_y) = f(n_x) + f(n_y)$.
3. Jsou-li X a Y dva jevy, jejichž počet realizací je n_x a n_y, a platí $n_x > n_y$, pak požadujeme, aby $f(n_x) > f(n_y)$.

Vhodným typem funkcí splňujících tyto podmínky jsou *logaritmy*. Pokud uvažujeme počítač s pamětí o kapacitě n bitů, pak do takové paměti můžeme uložit celkem 2^n hodnot. Toto číslo vyjadřuje počet možných realizací. Počet bitů nutných k zobrazení těchto realizací se pak dá psát jako logaritmus o základu dva z počtu realizací. Zdá se proto, že nejvhodnější je zvolit dvojkový logaritmus, jednotkou pak bude bit.

Počítání s logaritmy

Výpočet logaritmu o základu dva lze provádět i na běžné kalkulačce s podporou desítkových logaritmů. Vyjdeme přitom ze vztahu

$$\log_a x = \frac{\log_b x}{\log_b a} = \log_a b \cdot \log_b x,$$

při dosazení požadovaných základů pak dostáváme

$$\log_2 x = \log_2 10 \cdot \log x \doteq 3{,}322 \cdot \log x.$$

Vlastní informace

Předpokládejme množinu všech možných realizací náhodného jevu X, jednu z takových realizací x a pravděpodobnost této realizace $p(x)$. Každému $x \in X$ pak lze přiřadit reálné číslo $I(x)$, které nazveme **vlastní informací** (někdy též *částečnou informací*) o realizaci x. Toto číslo představuje množství informace obsažené ve výsledku x a jeho hodnotu v bitech[13] vypočteme ze vztahu[14]

$$I(x) = -\log_2 p(x).$$

Příklad 3.1

Chceme-li zjistit, jakou vlastní informaci nese zpráva o výsledku losování určitých 5 čísel z 20, postupujeme tak, že aplikujeme vztah pro výpočet vlastní informace, určíme pravděpodobnost vytažení konkrétní pětice a dosadíme do vzorce:

$$I(x) = -\log_2 p(x) = -\log_2 \frac{1}{\binom{20}{5}} = -\log_2 \frac{1}{15\,504} \doteq 13{,}92 \text{ bitů}$$

[13] Pro vyjádření množství informace se používají také jednotky *shannon* (Sh) při použití dvojkového logaritmu, *nat* (nat) při použití přirozeného logaritmu a *hartley* (Hart) při použití desítkového logaritmu na počest inženýra Ralpha Vintona Lyona Hartleyho (* 1888, † 1970). Platí 1 Sh = 1 bit = 0,693 nat = 0,301 Hart.

[14] Je-li všech n realizací jevu X stejně pravděpodobných, pak pravděpodobnost každé realizace x je $p(x) = 1/n$ a můžeme odvodit $I(x) = -\log_2 1/n = -\log_2 n^{-1} = \log_2 n$. Znaménko minus je pouze „kosmetická“ záležitost, která zaručuje, že velikost informace bude vždy nezáporná.

Hodnotu vlastní informace můžeme použít i pro zhodnocení úspornosti uložení dat v paměti počítače. Představme si, že bychom výsledek losování pětice čísel z 20 chtěli uložit do paměti počítače. Jakým způsob reprezentace zvolíme? Můžeme čísla zapsat do textového souboru a oddělit je mezerou, pak bychom potřebovali 9–14 B podle toho, zda by byla jednociferná, nebo dvouciferná. Kdybychom je ukládali vždy jako dvouciferná (tj. jednociferná doplníme zleva nulou), nepotřebovali bychom oddělovač a stačilo by nám 10 B. Výsledek předchozího příkladu ale ukazuje, že existuje způsob, jak pětici uložit mnohem úsporněji. Řešením je zakódování konkrétní pětice do posloupnosti nul a jedniček, která by měla 14 cifer a stačil by nám proto prostor o velikosti 2 B. O reprezentaci různých typů dat v paměti počítače a možnostech jejich kódování pojednávají následující kapitoly (viz str. 49).

Nyní můžeme přistoupit k určení průměrného množství informace ve zprávě, tj. k výpočtu informační entropie. Jak již víme, číslo $I(x)$ vyjadřuje množství informace obsažené ve výsledku jedné realizace náhodného jevu. Nás ale zajímá i zjištění množství informace o náhodné veličině. Entropii $H(X)$ můžeme považovat za informaci o náhodné veličině X obsaženou v této náhodné veličině, popřípadě za maximální informaci, jakou lze o veličině X vůbec získat. Proto entropii někdy nazýváme *úplnou informací* o náhodné veličině X. S ohledem na teorii pravděpodobnosti je entropie střední hodnotou vlastní informace.

Výpočet entropie

Předpokládejme množinu všech možných realizací jevu $X = \{x_1, x_2, \dots, x_n\}$. Dále uvažujme pravděpodobnosti jednotlivých realizací $p(x_1), p(x_2), \dots, p(x_n)$. Informační **entropii** náhodného jevu X pak vypočteme ze vztahu

$$H(X) = -\sum_{i=1}^{n} p(x_i) \cdot \log_2 p(x_i) = \sum_{i=1}^{n} p(x_i) \cdot I(x_i),$$

kde $I(x_i)$ je vlastní informace obsažená v realizaci x_i.

Příklad 3.2

Předpokládejme, že soutěžící v televizní soutěži dostane otázku a k ní čtyři nabídnuté odpovědi. Správnou odpověď nezná a ani žádnou z nabízených variant nepreferuje. Nejistota soutěžícího v situaci, kdy správná odpověď může být se stejnou pravděpodobností kterákoli ze čtyř nabídnutých, je rovna $H(X) = -4 \cdot 0{,}25 \cdot \log_2 0{,}25 = -\log_2 0{,}25 = 2$.

Soutěžící proto nejspíše požádá o nápovědu „50 na 50“ a z nabízených odpovědí mu nyní zůstanou už jen dvě. Správná může být opět kterákoli z nich, proto nejistota soutěžícího v této situaci je rovna $H(X) = -2 \cdot 0{,}5 \cdot \log_2 0{,}5 = -\log_2 0{,}5 = 1$.

Nyní si soutěžící vybere jednu z nabízených variant a odpoví na otázku. Vzápětí se dozví, zda byla odpověď správná. Bez ohledu na to, zda vyhrál, nyní už správnou odpověď zná, takže hodnota informační entropie je rovna $H(X) = -1 \cdot \log_2 1 = -\log_2 1 = 0$.

Tím, že se soutěžící dozvěděl správnou odpověď, klesla jeho neznalost v této oblasti z původních 2 bitů k nule, obdržel tedy informaci o velikosti 2 bity.

Příklad 3.3

Podobně bychom mohli kvantifikovat množství informace, kterou obdržel pohádkový Jiřík, když měl určit, která z dívek je Zlatovláska. Když zdolal všechny překážky, ocitnul se před 12 zahalenými dívkami a měl si vybrat tu, jež se stane jeho budoucí královnou. V této nezáviděníhodné situaci existuje 12 stejně pravděpodobných realizací jevu X. Hodnota informační entropie je v tuto chvíli rovna $H(X) = -12 \cdot 1/12 \cdot \log_2 1/12 = -\log_2 1/12 \doteq 3{,}59$.

Díky dobrotě srdce Jiříkova v předchozím dění pohádky mu však zlatá muška sednuvší na třetí pannu zleva změnila podmínky soutěže na mnohem příznivější situaci, kdy z 12 možných realizací byla rázem jedna jistá a zbylých 11 nemožných. Jiřík měl naprostou jistotu, že zvolí Zlatovlásku, proto hodnota informační entropie je rovna $H(X) = -\log_2 1 = 0$. Povšimněme si, že v tuto chvíli je zcela jedno, zda okolo Zlatovlásky stojí dalších 11 dívek, či zda tam Zlatovláska stojí zcela osamocena.[15]

Připusťme však alternativní pohádku, kdy je letní parno, plno much kolem hradu i uvnitř a znavený Jiřík si není jistý, zda velká zlatá moucha přiletěvší náhle na první dívku je opožděnou pravou zlatou muškou, nebo jen obyčejnou masařkou. Musí si vybrat mezi dvěma dívkami, tedy jeho neznalost je rovna $H(X) = -2 \cdot 0{,}5 \cdot \log_2 0{,}5 = 1$.

Kdyby se však velká a tlustá moucha Jiříkovi nezdála, mohla by být volba o něco jednodušší. Určitá pochybnost zůstává, takže můžeme předpokládat, že dívka, na níž ukazuje masařka, s 99% pravděpodobností není Zlatovláska. Opět nezáleží na pořadí dívek, proto je Jiříkova neznalost rovna $H(X) = -\left(1/100 \cdot \log_2 1/100 + 99/100 \cdot \log_2 99/100\right) \doteq 0{,}08$.

Příklad 3.4

Na železničním návěstidle je možné nastavit návěst „Stůj“, která svítí 80 % času, a dalších pět různých návěstí s přibližně stejnou pravděpodobností. Vypočtěte entropii tohoto zdroje zpráv.

Tento jev má šest možných realizací s pravděpodobnostmi $p(x_1) = 0{,}8$, $p(x_2) = 0{,}04$, $p(x_3) = 0{,}04$, …, $p(x_6) = 0{,}04$. Dosadíme tyto hodnoty do vzorce a vypočteme hodnotu informační entropie.

$$H(X) = -\left(0{,}8 \cdot \log_2 0{,}8 + 5 \cdot 0{,}04 \cdot \log_2 0{,}04\right) \doteq 1{,}19$$

Nejmenší míra informace

Z předchozích příkladů je zřejmé, že entropie jevu X nabývá nejvyšší hodnoty při stejné pravděpodobnosti výskytu jednotlivých realizací x_i. Pak platí $H(X) = -\log_2 p(x_i)$. **Nejmenší míru informace** proto můžeme odvodit od jevu, který má pouze dvě stejně pravděpodobné realizace. Potom $H(X) = -2 \cdot 0{,}5 \cdot \log_2 0{,}5 = -\log_2 0{,}5 = 1$, z čehož vyplývá, že nemůže existovat informace o velikosti menší než 1 bit.

Kontrolní otázky

5. Jak zní Shannonova definice informace?
6. Jaký je rozdíl mezi náhodným jevem a jeho realizací?
7. Ve kterých jednotkách vyjadřujeme množství informace?
8. Kterým vztahem spočítáme vlastní informaci o realizaci nějakého jevu?
9. Kterým vztahem spočítáme informační entropii nějakého jevu?
10. Čemu je rovna nejmenší velikost informace a z čeho je odvozena?

[15] Jiřík by pak „vybíral“ z jednoho kandidáta stejným způsobem jako občané při totalitních volbách.

Pro zájemce uvedeme na závěr této podkapitoly ještě jeden příklad ze života, který může být užitečný zejména pro sázkaře. Vysvětlíme si princip stanovení sázkového kurzu na základě poznatků z teorie informace.

Příklad 3.5

Sázková kancelář vypisuje sázky na sportovní utkání mezi týmem A a týmem B. Na základě dlouhodobých zkušeností a složitých výpočtů odhaduje, že v základní hrací době vyhraje s pravděpodobností 45 % tým A, remíza nastane s pravděpodobností 20 % a tým B vyhraje s pravděpodobností 35 %. Jak na základě těchto informací sestaví a vypíše sázkové kurzy na jednotlivé varianty?

Pokud by sázková kancelář použila principy z teorie informace, stačilo by spočítat vlastní informace pro jednotlivé realizace, tedy

$$I(A) = -\log_2 0{,}45 = 1{,}15, \quad I(B) = -\log_2 0{,}2 \doteq 2{,}32, \quad I(0) = -\log_2 0{,}35 \doteq 1{,}51$$

a z výsledků sestavit sázkové kurzy. Na první pohled je zřejmé, že remíza je nejméně pravděpodobným výsledkem, a proto je kurz nejpříznivější. Šťastlivec, který by si na tuto variantu vsadil třeba 100 000 Kč, by v případě správného tipu vyhrál 232 000 Kč.

Ve skutečnosti se sázkové kurzy stanovují jiným způsobem, ale princip závislosti kurzu na pravděpodobnosti je samozřejmě zachován. Sázkové kanceláře vypočítávají kurz (v ideálním spravedlivém případě) jako převrácenou hodnotu pravděpodobnosti, kterou ještě sníží o marži. Uvažujeme-li 10% marži, dojdeme k následujícím výsledkům:

$$\frac{1}{0{,}45} \cdot 0{,}9 = 2{,}00, \quad \frac{1}{0{,}20} \cdot 0{,}9 = 4{,}50, \quad \frac{1}{0{,}35} \cdot 0{,}9 = 2{,}57.$$

Šťastlivec, který by si vsadil na remízu 100 000 Kč, by v případě správného tipu za předpokladu spravedlivě stanoveného kurzu vyhrál 450 000 Kč.

3.3 Jednotky informace

Bit, bajt a násobky

Základní jednotkou informace je **bit** (*binary digit*, b), který nabývá hodnoty buď 0, nebo 1. Z historických důvodů platí, že 8 bitů tvoří jeden **bajt** (*byte*, B).

Násobky základních jednotek informace (tab. 3.2) uvádí norma IEC 60027-2, která platí od roku 2005. Pro úplnost dodejme, že velikosti běžných pevných disků osobních počítačů se dnes pohybují v řádu stovek GiB. Podle údajů z roku 2021 je v celosvětovém digitálním vesmíru uloženo asi 79 ZiB dat.

Na rozdíl od soustavy SI není v informatice mezi jednotlivými násobky základní jednotky hodnota 1 000, ale 1 024. Je tomu tak proto, že počítač nepracuje v desítkové soustavě, kde by hodnotu 1 000 bylo možné vyjádřit jako mocninu nižší hodnoty (10^3). Ve dvojkové soustavě je této mocnině nejblíže právě hodnota 1 024, která představuje mocninu 2^{10}.

Marketing v praxi

Desítkové předpony v souladu se soustavou SI byly dříve používány v chybném významu a ještě dnes se mnohdy setkáme s jejich chybným použitím pro vyjádření množství informace. Například 1 kilobajt (kB) představuje 1 000 bajtů a nikoli 1 024 bajtů. Toho mnoho výrobců a prodejců zneužívá a nabízí produkty s klamavým označením.

Tabulka 3.2: *Přehled základních jednotek informace podle normy IEC 60027-2*

Jednotka	Zkratka	V bajtech	Přepočet
Kibibajt	KiB	2^{10} B	1 024 B
Mebibajt	MiB	2^{20} B	1 024 KiB
Gibibajt	GiB	2^{30} B	1 024 MiB
Tebibajt	TiB	2^{40} B	1 024 GiB
Pebibajt	PiB	2^{50} B	1 024 TiB
Exbibajt	EiB	2^{60} B	1 024 PiB
Zebibajt	ZiB	2^{70} B	1 024 EiB
Yobibajt	YiB	2^{80} B	1 024 ZiB

Příklad 3.6

Představme si nabídku pevného disku s inzerovanou kapacitou 600 GB. Kolik paměti má uživatel ve skutečnosti k dispozici?

Pokud bychom chtěli vyjádřit kapacitu pevného disku v bajtech, dostali bychom se k číslu 644 245 094 400, neboť uvažujeme správné použití „počítačových“ jednotek GiB, přestože na obalu je uvedeno GB. Ve skutečnosti ale bude kapacita o 44 246 094 400 B nižší, neboť prodejce s vysokou pravděpodobností používá přepočet s násobky 1 000.

Pokud je na obalu disku uvedena kapacita 600 GB, zkušený informatik dobře ví, že má k dispozici jen 558 GiB. Co si ale pomyslí běžný uživatel operačního systému Windows, když mu Průzkumník oznámí, že kapacita disku, na jehož obalu je inzerováno 600 GB, je pouze 558 GB? Rozdíl činí cca 42 GB, což představuje téměř 9 disků DVD.

Novější distribuce operačních systémů macOS a Linux už dlouhá léta uvádějí veškeré údaje o velikostech pamětí ve správných jednotkách, ale operační systém Windows stále urputně trvá na desítkových předponách, které navíc používá v chybném významu.

Aby byla absurdita ještě větší, kromě norem SI (desítkové předpony) a IEC (dvojkové předpony) existuje ještě norma JEDEC (*Joint Electron Device Engineering Council*), která se vztahuje zejména na operační paměti a používá desítkové předpony, avšak s dvojkovým významem.

Kontrolní otázky

11. Co je základní jednotkou informace?

12. Kolik bitů tvoří jeden bajt?

13. Jak se jmenují první tři násobky velikosti informace?

3.4 Příklady na procvičení

Kombinatorika

Příklad 3.7 Mějme číslice 1, 2, 3, 4, 5. Kolik různých pěticiferných čísel lze s jejich pomocí zapsat, pokud se každá číslice použije právě jednou? Kolik takových čísel bude začínat číslicí 5? A kolik jich bude sudých?

Příklad 3.8 Čtyři české a tři slovenské knihy je třeba uspořádat na poličce tak, aby byly seřazeny nejprve české a poté slovenské knihy. Kolika způsoby se to dá udělat?

Příklad 3.9 Patnáct svatebčanů se nemohlo dohodnout, kdo kde bude stát na svatební fotografii. Ženich navrhl, aby učinili všechny možné sestavy svatebčanů na fotografiích. Kolik fotografií je třeba udělat? Kolik času by to zabralo, pokud pořízení jedné fotky trvá 10 sekund a fotografuje se nepřetržitě?

Příklad 3.10 Mějme množinu znaků $\{a, b, c\}$. Kolik různých čtyřznakových řetězců, ve kterých se znak a vyskytuje dvakrát, lze z této množiny vytvořit?

Příklad 3.11 Osm studentů má na kolejích připraveno ubytování ve třech pokojích. Dva jsou třílůžkové, jeden dvoulůžkový. Kolika způsoby můžeme studenty rozdělit do pokojů?

Příklad 3.12 Kolika způsoby je možné hodit třemi kostkami součet 11?

Příklad 3.13 Šestimístný kód uzavírání trezoru v bance je vytvořen ze stejných číslic jako číslo 220096. Kolik je možností vytvoření příslušného kódu, pokud se číslice může opakovat nejvýše dvakrát? Jak dlouho by zloději trvalo vyzkoušet všechny možnosti, pokud nad každou stráví 5 sekund?

Příklad 3.14 Přijímací zkouška na vysokou školu se skládá ze čtyř úkolů. Z každého úkolu musí uchazeč získat minimálně 2 body, podle kvality řešení také 3 nebo 4 body. K přijetí stačí dosáhnout 13 bodů. Kolika způsoby může uchazeč složit přijímací zkoušku, aby byl přijat?

Příklad 3.15 Mějme množinu znaků $\{a, b, c, d\}$. Kolik různých dvouznakových řetězců můžeme vytvořit, pokud se žádný znak nesmí v řetězci opakovat?

Příklad 3.16 Do výboru zvolili 7 poslanců. Kolika způsoby se z nich dá vybrat předseda, místopředseda a tajemník?

Příklad 3.17 Osm studentů si slíbilo, že si z prázdninových cest vzájemně pošlou pohlednice. Kolik pohlednic bylo rozesláno?

Příklad 3.18 V tombole je šest různých výher. Kolika způsoby můžeme postupně (s přihlédnutím k pořadí) vylosovat tři výherce?

Příklad 3.19 Mějme množinu $\{0, 1\}$. Kolik různých zpráv o délce 6 znaků z ní lze vytvořit?

Příklad 3.20 Mějme množinu $\{a, b, c, d\}$. Kolik různých zpráv o délce 5 znaků z ní lze vytvořit?

Příklad 3.21 Hodíme třemi kostkami – bílou, modrou a žlutou. Kolik různých výsledků můžeme dostat?

Příklad 3.22 Kolik různých značek teoreticky existuje v Morseově abecedě, sestavují-li se tečky a čárky do skupin po jedné až pěti?

Příklad 3.23 V rovině je 6 různých bodů (žádné 3 body neleží na jedné přímce). Kolik různých úseček dostaneme pospojováním všech těchto bodů navzájem?

Příklad 3.24 Na kružnici je rozmístěno 9 bodů. Kolik různých trojúhelníků, jejichž vrcholy jsou tyto body, existuje?

Příklad 3.25 Zkoušející má připravených 20 příkladů z matematiky a 30 z ekonomiky. Na písemku chce dát 3 příklady z matematiky a 2 z ekonomiky. Kolik má možností sestavení různých zadání?

Příklad 3.26 Ve třídě je 19 chlapců a 11 dívek. Zjistěte, kolika způsoby lze z nich vybrat tříčlennou skupinu, v níž jsou a) pouze chlapci, b) pouze dívky, c) dva chlapci a jedna dívka.

Příklad 3.27 Kolika způsoby je možné rozdat 32 hracích karet 4 hráčům?

Příklad 3.28 V cukrárně mají pět druhů zmrzlin. Otec chce pro rodinu koupit 15 porcí. Kolika způsoby může zmrzlinu koupit?

Příklad 3.29 V železničním depu je 20 osobních vozů, 7 lůžkových vozů a 5 poštovních vozů. Kolik různých pětivozových souprav lze sestavit, jestliže nezáleží na pořadí vozů v soupravě?

Příklad 3.30 Kolika způsoby můžeme rozdělit 20 volných vstupenek na premiéru komedie Babovřesky mezi 10 důchodkyň, které mají v úmyslu vzít do kina vnoučata?

Teorie informace

Příklad 3.31 Potřebujeme unikátně očíslovat milion osob. Kolikaciferná čísla budeme potřebovat, použijeme-li soustavu a) desítkovou, b) dvojkovou, c) osmičkovou, d) šestnáctkovou?

Příklad 3.32 Kolik otázek je třeba v průměru položit, abychom se dozvěděli datum narození člověka (den v roce), pokud odpovědi jsou pouze ano/ne a tázaný odpovídá pravdivě? Uvažujte, že každý čtvrtý rok je přestupný.

Příklad 3.33 Vypočtěte, kolik bitů je nejméně potřeba pro uložení čísla -675 do paměti počítače. Uvažujte dvojkový doplňkový kód.

Příklad 3.34 Vypočtěte pravděpodobnost realizace jevu X, která nese vlastní informaci o velikosti a) 5 bitů, b) 8 bitů, c) 12 bitů, d) 16 bitů.

Příklad 3.35 Vypočtěte vlastní informaci, kterou nese tato zpráva: *„Počítač Šmudla-11 pracuje v 16 hodin v systému Linux.“* Přitom víte, že máte k dispozici 400 počítačů, ty mohou pracovat ve třech operačních systémech nebo mohou být vypnuty a časový údaj je s přesností na hodiny.

Příklad 3.36 V týdenním rozvrhu je celkem 42 bloků, každý se může nacházet v některé ze 48 učeben a může jej učit jeden ze 110 učitelů. Zjistěte, jakou vlastní informaci nese zpráva: *„Ing. Bžouda je v pátek ve 13 hodin v Q07 a po něm nastupuje tamtéž RNDr. Kokoška.“*

Příklad 3.37 Vypočtěte, kolika bity je potřeba vyjádřit informaci o skutečnosti, že z padesáti žárovek je jedna vadná.

Příklad 3.38 Z předmětu Teoretické základy informatiky získá student jedničku s pravděpodobností 0,25, z předmětu Algoritmizace získá jedničku s pravděpodobností 0,5. Vypočtěte vlastní informaci situace, kdy jeden student získá jedničku z obou předmětů.

Příklad 3.39 Na silničním semaforu svítí vždy 60 s červená, 15 s červená se žlutou, 60 s zelená, 10 s žlutá. Vypočtěte velikost vlastní informace, kterou nese svit červené.

Příklad 3.40 Je dána věta: *„Babička jede do města.“* Zjistěte velikost vlastní informace obsažené v této větě. Víte, že počet osob, které připadají v úvahu, je 50, počet činností, které mohou konat, je 100, počet míst, kde se děj může odehrávat, je 10.

Příklad 3.41 Je dána věta: *„Sestra šla večer do kina.“* Zjistěte vlastní informaci zprávy za předpokladu, že daný člověk má ještě bratra, sestra může konat 20 činností a může jít na 15 různých míst. Jakou vlastní informaci bude mít tato zpráva pro člověka, který ví, že sestra nechodí jinam než do kina?

Příklad 3.42 Vypočtěte vlastní informaci, kterou nese zpráva o prvních čtyřech místech v závodu lyžařů ve slalomu. Na startu bylo 32 závodníků. Předpokládejte, že každý z nich mohl vyhrát se stejnou pravděpodobností.

Příklad 3.43 Z 64 semifinalistů soutěže postoupilo do finále 8 soutěžících. Vypočtěte množství informace v této zprávě.

Příklad 3.44 Jistý král jisté země má v hradním sklepě sbírku 930 lahví jistého bylinného likéru. Jeho političtí protivníci z jisté politické strany však usilují o to, aby se jej zbavili, a proto pošlou hradního šaška, aby láhve otrávil. Hradní stráž však šaška chytila hned poté, co stihl otrávit první láhev. Nepodařilo se jim ale zjistit, která to byla. Výslechem se zjistilo, že jed je tak prudký, že i při nízké koncentraci by krále spolehlivě zabil, navíc se jeho účinky projeví až po uplynutí 72 hodin. Král se proto rozhodl, že použije některé z doživotně odsouzených vězňů, aby jeho likér ochutnali. Kolik vězňů bude potřebovat, aby spolehlivě zjistil, ve které láhvi se jed nachází, a mohl už za 72 hodin zase v klidu popíjet?

Příklad 3.45 Závodu se zúčastnilo 10 závodníků. Vypočtěte množství informace ve zprávě, která obsahuje jména závodníků, kteří se umístili na prvních třech místech, pokud:
a) zpráva neobsahuje informace o jejich konkrétním umístění,
b) zpráva obsahuje informace o jejich konkrétním umístění.

Příklad 3.46 V koši jsou 2 černé, 2 červené a 2 bílé míčky. Vypočtěte informační entropii barvy náhodně (poslepu) vybraného míčku. Poté vypočtěte entropii barvy dalšího poslepu vybraného míčku (po provedení předchozího výběru).

Příklad 3.47 Potkají se dva studenti po zkoušce. První říká: *„Mám za jedna.“* Druhý na to: *„Taky jsem zkoušku udělal.“* Kolik informace každá zpráva obsahuje? Určete odstraněnou neurčitost.

Příklad 3.48 Vypočtěte množství informace ve zprávě, která redukuje počet variant určitého jevu z 64 na 16.

Příklad 3.49 Dostali jste nabídku účasti ve hře v kostky. Máte na výběr mezi dvěma hracími kostkami K_1 a K_2, které nejsou symetrické a mají následující pravděpodobnosti:
$K_1 = \left(2^{-3}, 2^{-3}, 2^{-3}, 2^{-3}, 2^{-3}, 3 \cdot 2^{-3}\right)$, $K_2 = \left(2^{-2}, 2^{-2}, 2^{-3}, 2^{-3}, 2^{-3}, 2^{-3}\right)$.
Preferujete-li symetrii kostky, kterou si vyberete? Bude výhodnější, když se před začátkem hry vylosuje kostka, se kterou se bude hrát?

Příklad 3.50 Zdroj zpráv produkuje zprávy ze souboru $\{x_1, x_2, x_3, x_4\}$ s pravděpodobnostmi postupně 0,2; 0,3; 0,4 a 0,1. Vypočtěte entropii tohoto zdroje zpráv. Dále se pokuste určit redundanci zdroje.

Příklad 3.51 Ze zkoušky získalo 8 studentů A, 8 studentů B, 16 studentů C, 16 studentů D, 8 studentů E a 8 studentů F. Určete entropii známky náhodně vybraného studenta.

Příklad 3.52 V zásilce je 24 vajec velikosti S, 96 vajec velikosti M, 48 vajec velikosti L a 24 vajec velikosti XL. Určete entropii velikosti náhodně vybraného vejce.

Příklad 3.53 Vyjádřete množství informace v uvedených jednotkách:

a)	1 MiB	(KiB)	d)	256 kb	(B)	g)	0,5 GiB (MiB)
b)	2 KiB	(B)	e)	100 B	(b)	h)	128 kb (KiB)
c)	64 b	(B)	f)	512 B	(KiB)		

Shrnutí

Na informace můžeme pohlížet ze dvou hledisek. Vědní obor informatika se zabývá jejich získáváním, zpracováním a hodnocením. Vědní obor teorie informace se zabývá otázkami měření množství informace ve zprávě, přenosem a kódováním dat. Objasnili jsme si pojmy údaje, data, informace a znalosti a seznámili se s Shannonovou etodou měření množství informace ve zprávě na základě poznatků z kombinatoriky a statistiky. Naučili jsme se počítat vlastní informaci o realizaci nějakého jevu a informační entropii celého jevu. Odvodili jsme základní jednotku informace a seznámili se s násobky používanými v informatice.

Slovníček pojmů

Bajt: jednotka informace, 8 bitů.
Bit: základní jednotka informace.
Data: kvalitativně nebo kvantitativně formalizované údaje.
Datový typ: množina povolených hodnot a povolených operací s nimi.
Entropie (v teorii informace): veličina udávající míru neurčitosti nebo nejistoty o nějakém jevu, průměrné množství informace ve zprávě (také úplná informace).
Informace: data, která snižují neznalost a vyvolávají změnu stavu či chování příjemce.
Informace (podle Shannona): míra množství neurčitosti nebo nejistoty o nějakém ději odstraněná realizací tohoto děje.
Informační systém: dynamický systém, jehož vazby tvoří informace a prvky představují místa jejich transformace.
Informatika: vědní obor zabývající se informacemi z hlediska kvalitativního.
Jev: libovolný náhodný proces s více možnými realizacemi.
Realizace jevu: každý konkrétní projev jevu.
Teorie informace: vědní obor zabývající se informacemi z hlediska kvantitativního.
Údaje: hodnoty získané měřením, pozorováním nebo zaznamenáním reálné skutečnosti.
Vlastní informace: množství informace obsažené ve výsledku jedné realizace jevu (také částečná informace).
Znalosti: ucelený komplex informací o nějaké objektivní realitě.

Řešení kontrolních otázek

1. *Co jsou údaje, data, informace a znalosti?*

Údaje jsou hodnoty získané měřením, pozorováním nebo zaznamenáním reálné skutečnosti. Kvalitativně nebo kvantitativně formalizované údaje nazýváme data. Informace jsou data, která snižují neznalost a vyvolávají změnu stavu či chování příjemce. Ucelený komplex informací o nějaké objektivní realitě nazýváme znalosti.

2. *Kdy se z dat stávají informace?*

Data se stávají informacemi v okamžiku jejich zpracování, kdy příjemci přinášejí něco nového a snižují tím jeho neznalost či nejistotu. Všechny informace jsou data, ale ne všechna data jsou informacemi. Pěkným příkladem toho, jak se z dat stávají informace, je dotaz v jazyce SQL položený nad databází uživatelů, kterým zjišťujeme jejich průměrný věk.

3. *Čím se zabývá teorie informace a čím informatika?*

Teorie informace zkoumá informace z hlediska kvantitativního (měření množství informace, kódování a přenos dat), informatika zkoumá informace z hlediska kvalitativního (získávání, zpracování a hodnocení).

4. *Co je informační systém a k čemu slouží?*

Informační systém je dynamický systém, jehož vazby tvoří informace a prvky tvoří místa jejich zpracování. Slouží k poskytování potřebných informací v požadovaném rozsahu, lhůtách, podrobnostech i formě.

5. Jak zní Shannonova definice informace?

Informace podle Shannona je míra množství neurčitosti nebo nejistoty o nějakém ději, která byla odstraněna realizací tohoto děje.

6. Jaký je rozdíl mezi náhodným jevem a jeho realizací?

Náhodný jev je libovolný náhodný proces s mnoha možnými realizacemi. Realizace jevu je jeden jeho konkrétní projev.

7. Ve kterých jednotkách vyjadřujeme množství informace?

V informatice vyjadřujeme množství informace v bitech, při výpočtu množství informace používáme logaritmy o základu 2.

8. Kterým vztahem spočítáme vlastní informaci o realizaci nějakého jevu?

Vlastní informace o realizaci nějakého jevu je rovna zápornému dvojkovému logaritmu pravděpodobnosti dané realizace, tedy $I(x) = -\log_2 p(x)$.

9. Kterým vztahem spočítáme informační entropii nějakého jevu?

Informační entropii nějakého jevu spočítáme tak, že sečteme velikosti vlastních informací všech realizací tohoto jevu vynásobených jejich pravděpodobnostmi, tedy podle vztahu $H(X) = \sum_{i=1}^{n} p(x_i) \cdot I(x_i)$.

10. Čemu je rovna nejmenší velikost informace a z čeho je odvozena?

Nejmenší velikost informace je rovna jednomu bitu a je odvozena z entropie jevu, který má pouze dvě stejně pravděpodobné realizace.

11. Co je základní jednotkou informace?

Základní jednotkou informace je jeden bit.

12. Kolik bitů tvoří jeden bajt?

Jeden bajt tvoří osm bitů.

13. Jak se jmenují první tři násobky velikosti informace?

Základním násobkem je jeden bajt (B), 1 024 bajtů je jeden kibibajt (KiB), 1 024 kibibajtů je jeden mebibajt (MiB) a 1 024 mebibajtů je jeden gibibajt (GiB).

4 Vnitřní reprezentace dat

Úvod

Jak již dobře víme, veškerá data v počítači jsou reprezentována výhradně pomocí nul a jedniček. Pro člověka však musí být tyto hodnoty vhodně *zobrazeny*, aby jim mohl správně porozumět. Zobrazení stejné posloupnosti nul a jedniček ale může být provedeno nekonečně mnoha způsoby. Tím, že nějaké posloupnosti binárních hodnot v paměti počítače přisoudíme konkrétní význam, *interpretujeme* zobrazené údaje a současně tak *implementujeme* jejich datový typ. Modelujeme tímto způsobem objektivní realitu, protože veškerá data jsou zobrazena pro vstup (v binární podobě) i pro výstup (v podobě srozumitelné lidem).

Cíl

Cílem této rozsáhlé kapitoly je důkladné pochopení principů reprezentace jednotlivých typů dat v paměti počítače. Ukážeme si, jak lze v počítači uložit celá i racionální čísla, texty, obrázky i hudbu, dále srovnáme různé možnosti reprezentace z hlediska použitelnosti a poznatky uvedeme do souvislosti s datovými typy v programovacích jazycích.

Návaznosti

Pro studium této kapitoly je nutné ovládat aritmetické operace ve dvojkové soustavě, orientovat se v principech teorie informace a znát alespoň základní jednoduché datové typy používané v běžných programovacích jazycích nebo v tabulkových kalkulátorech.

Studijní čas

Čas potřebný k nastudování této kapitoly je asi 10 hodin.

4.1 Reprezentace numerických dat

Zobrazení hodnot v paměti počítače je *limitováno* rozsahem paměťových buněk. To znamená, že každá posloupnost nul a jedniček je zobrazena na předem daném prostoru. Velikost takového prostoru (počet bitů paměti) úzce souvisí s datovým typem, který těmto údajům přisuzujeme. Máme-li například k dispozici 1 bajt paměti, tzn. 8 bitů, můžeme na tomto prostoru ukládat nejvýše osmiciferné binární posloupnosti, např. 10011101.

4.1.1 Binární řádová mřížka

Označení řádů

Formát zobrazitelných čísel jednoznačně definuje **binární řádová mřížka** (obr. 4.1), která v závislosti na umístění řádové čárky určuje rozsah zobrazitelných řádů čísla – od nejnižšího řádu m až po nejvyšší řád n. Nejnižší řád bývá označován jako LSB (*Least Significant Bit*), nejvyšší řád bývá označován jako MSB (*Most Significant Bit*).

bity:	0	0	1	0	1	1	0	1
řády:	5	4	3	2	1	0	−1	−2
	↑							↑
	n							$-m$

Obrázek 4.1: *Binární řádová mřížka zobrazující hodnotu 001011,01*

Vlastnosti mřížky

Mezi základní vlastnosti každé binární řádové mřížky v souvislosti s její velikostí (počtem bitů) patří zejména:

- *délka mřížky* – vyjadřuje počet rozlišitelných hodnot (obsazených řádů), značíme l;
- *jednotka* – představuje nejmenší kladné zobrazitelné číslo, značíme ϵ;
- *modul* – představuje nejmenší číslo, které již zobrazitelné není, značíme M;
- *pozice řádové čárky* – může být pevná či pohyblivá.

Operační paměť počítače je rozdělena do adresovatelných buněk o velikosti 1 bajtu, kterým říkáme *slabiky*. Délka řádové mřížky je proto vždy celým násobkem slabik.

Příklad 4.1

Určete nejnižší a nejvyšší řád, délku, modul a jednotku následujících mřížek:

a)

$n = 0, \quad m = 7, \quad l = 8, \quad M = (10{,}0000000)_2 = 2^1, \quad \epsilon = (0{,}0000001)_2 = 2^{-7}$

b)

$n = 7, \quad m = 0, \quad l = 8, \quad M = (100000000)_2 = 2^8, \quad \epsilon = (00000001)_2 = 2^0$

Jak vidíme z příkladu, pro základní vlastnosti řádové mřížky reprezentující číslo v libovolné poziční číselné soustavě o základu z obecně platí následující pravidla:

- délka mřížky je dána počtem řádů nalevo a napravo od řádové čárky, tj. $l = m + n + 1$;
- jednotka je číslo, které má jedničku pouze v nejnižším řádu, tj. $\epsilon = z^{-m}$;
- modul je číslo, které má jedničku v řádu, který již není v mřížce obsažen, tj. $M = z^{n+1}$.

Přetečení

V případě, kdy délka řádové mřížky je menší než počet řádů čísla, které do ní chceme vložit, toto číslo nebude v mřížce zobrazitelné. Budou uloženy jen ty řády, které se do mřížky vejdou, ostatní řády budou ignorovány. Dojde k **přetečení** (*overflow*) a ztrátě přesnosti v reprezentaci čísla, které pak bude uloženo s chybou, bude zaokrouhleno nebo bude zaměněno za číslo s opačným znaménkem. Ztráta nejnižších řádů zlomkové části není obvykle podstatná, neboť jejich číselná hodnota je malá a můžeme ji zanedbat. Naproti tomu ztráta nejvyšších řádů celé části čísla znamená prakticky úplné znehodnocení číselného údaje, protože jediný takto ztracený řád má vyšší hodnotu než zbytek zobrazené části čísla.

Pokud sčítáme operandy s různými znaménky (nebo odčítáme operandy se stejnými znaménky), výsledek nemůže být větší než žádný z operandů a k přetečení proto dojít nemůže. Pokud ale sčítáme operandy se stejnými znaménky (nebo odčítáme operandy s různými znaménky), k přetečení dojít může. Poznáme to snadno při přenosu do znaménkového bitu[16] (u sčítání), resp. při výpůjčce ze znaménkového bitu (u odčítání), kdy zjistíme, že na

[16] Tzv. znaménkový bit slouží u datových typů se znaménkem k reprezentaci znaménka. Blíže o něm bude pojednáno v podkapitole věnované reprezentaci záporných čísel (viz str. 52).

výsledek potřebujeme o 1 bit více a znaménkový bit tak místo znaménka reprezentuje jen další bit výsledku. Hardware detekuje přetečení tak, že přenos do nejvyššího bitu je různý od přenosu z něj (u sčítání), resp. výpůjčka z prvního bitu za nejvyšším bitem je různá od výpůjčky z nejvyššího bitu (u odčítání).

Příklad 4.2

Co se stane, pokud se pokusíme do osmibitového paměťového prostoru s řádovou čárkou za nejnižším bitem uložit číslo 275? Dojde k přetečení, neboť na prostoru 8 bitů lze reprezentovat nejvýše 256 hodnot. Řádová mřížka bude vypadat následovně:

1	0	0	0	1	0	0	1	1	‖

Nejvyšší bit čísla není kam uložit a bude ignorován, v paměti bude uloženo číslo 19.

Opakem přetečení je **podtečení** (*underflow*). K němu dochází v případě, kdy je číslo, které chceme do řádové mřížky zapsat, velmi malé a téměř se blíží nule, tedy první nenulová číslice je příliš daleko za řádovou čárkou a do mřížky se nevejde. Číslo je potom zaokrouhleno a uloženo jako nula. Tento problém nastává při reprezentaci racionálních čísel (viz str. 57).

Podtečení

Příklad 4.3

Co se stane, pokud se pokusíme do osmibitového vyhrazeného paměťového s řádovou čárkou za nejvyšším bitem uložit číslo 0,00390625 (= 2^{-8})? Dojde k podtečení, protože k reprezentaci čísla 2^{-8} potřebujeme 9 bitů. Řádová mřížka bude vypadat následovně:

0	‖ 0	0	0	0	0	0	0	1	

Nejnižší bit čísla není kam uložit a bude ignorován, v paměti bude uloženo číslo 0.

4.1.2 Celá čísla

Celá čísla jsou v počítači reprezentována s *pevnou řádovou čárkou*, což znamená, že řádová čárka v mřížce je umístěna vždy na pevné pozici vpravo za nejnižším bitem.

Celočíselné údaje můžeme reprezentovat datovými typy, které dělíme podle délky řádové mřížky na typy s nižší (*short*) a vyšší (*long*) přesností a podle přítomnosti znaménka na typy bez znaménka (*unsigned*) a se znaménkem (*signed*). V programovacích jazycích nemusí být všechny kombinace celočíselných datových typů vždy zastoupeny a velikost typů v paměti se může u různých implementací jazyků lišit.

Celočíselné datové typy

Máme-li k dispozici n bitů paměti, pak na tomto prostoru můžeme reprezentovat hodnoty v intervalu 0 až $2^n - 1$ v datovém typu bez znaménka, nebo v intervalu -2^{n-1} až $2^{n-1} - 1$ v datovém typu se znaménkem (tab. 4.1).

Rozsah hodnot

Celá čísla bez znaménka (tedy prvky z množiny $\mathbb{N}_0$) jsou reprezentována v binárním kódu bez úprav na vymezeném paměťovému prostoru. Řády, které nejsou obsazeny, jsou v mřížce doplněny nulami. Budeme-li uvažovat čtyřbitová[17] celá čísla bez znaménka (tab. 4.2), můžeme na daném prostoru zobrazit hodnoty 0 až 15 (tedy $2^4 - 1$).

Kladná čísla

[17] Přestože v počítači jsou hodnoty ukládány do prostoru o velikosti násobků bajtů (tedy nejméně na 8 bitech), ve snaze uvádět úplný výčet reprezentovatelných hodnot a současně nekomplikovat ilustraci příliš velkým množstvím hodnot si můžeme dovolit prostor omezit pouze na 4 bity bez újmy na obecnosti.

Tabulka 4.1: *Přehled celočíselných datových typů*

Velikost	Rozsah hodnot	Znaménko	Pascal	C++
1 B	-2^7 až $2^7 - 1$	ano	`shortint`	`signed char`
1 B	0 až $2^8 - 1$	ne	`byte`	`unsigned char`
2 B	-2^{15} až $2^{15} - 1$	ano	`smallint`	`int`
2 B	0 až $2^{16} - 1$	ne	`word`	`unsigned`
4 B	-2^{31} až $2^{31} - 1$	ano	`longint`	`long`
4 B	0 až $2^{32} - 1$	ne	`longword`	`unsigned long`

Tabulka 4.2: *Reprezentace čtyřbitových čísel bez znaménka*

Kód	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Význam	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Záporná čísla

Z kapitoly věnované číselným soustavám (viz str. 24) již víme, že základním problémem při reprezentaci záporných čísel v počítači je vyjádření znaménka pomocí binárního kódu. Používáme *číselné kódy*, které transformují omezenou množinu celých čísel do omezené množiny nezáporných čísel. Existují tři takové způsoby kódování:

- *přímé kódování* – mřížku rozdělíme na dvě části a budeme odděleně reprezentovat znaménko a absolutní hodnotu čísla;
- *aditivní kódování* – změníme způsob interpretace všech čísel tak, že před uložením zvýšíme hodnotu čísla o zvolenou konstantu, čímž se „zbavíme“ záporných čísel;
- *doplňkové kódování* – změníme způsob interpretace záporných čísel tak, že znaménko bude přímou součástí hodnoty čísla.

Přímý kód

Jedinou pro člověka čitelnou formou zápisu, která dokáže rozlišit kladné a záporné hodnoty, je **přímý kód** (*sign magnitude*). Hardware, který by takový kód podporoval, by ale musel být zbytečně složitý a vlivem toho i nepříjemně pomalý.

Základním principem přímého kódu je reprezentace znaménka pomocí nejvyššího řádu mřížky, tzv. *znaménkového bitu*. Ten nabývá hodnoty 0 v případě, že je číslo kladné, nebo hodnoty 1, je-li číslo záporné (tab. 4.3). Lidé chápou záporná čísla stejným způsobem s tím rozdílem, že jedničku ve znaménkovém bitu nahrazují symbolem *minus*.

Tabulka 4.3: *Reprezentace čtyřbitových čísel se znaménkem v přímém kódu*

Kód	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Význam	0	1	2	3	4	5	6	7	−0	−1	−2	−3	−4	−5	−6	−7

Tento způsob vyjádření nezachovává relace mezi kladnými a zápornými čísly, protože kladná čísla rostou zleva doprava, zatímco záporná čísla rostou zprava doleva. Při pohledu na reprezentaci čísel v kódu vidíme, že sice platí $5 > 2$, ale současně $-5 > -2$ a dokonce $-5 > 2$. Největším problémem přímého kódu je dvojí reprezentace nuly, kdy čísla 0 a −0 se chápou jako různá a dokonce platí $-0 > 0$.

Z původních n bitů vyhrazených pro reprezentaci čísla jsme jeden použili na reprezentaci znaménka, pro samotné číslo tak zůstává pouze $n-1$ bitů. Na takovém prostoru můžeme zobrazit pouze 2^{n-1} hodnot, tedy o polovinu méně než na n bitech. Na druhou stranu každé takto zobrazitelné číslo může, ale nemusí mít znaménko. Místo rozsahu 0 až $2^n - 1$ jsme proto nyní schopni zobrazit rozsah $-2^{n-1} + 1$ až $2^{n-1} - 1$.

Příklad 4.4

Jak vyjádříme číslo −53 v přímém kódu na prostoru 8 bitů? Nejdříve převedeme číslo 53 do dvojkové soustavy, výsledek doplníme zleva nulami, abychom vyplnili celou délku mřížky, a nakonec upravíme znaménkový bit. Reprezentací čísla −53 v přímém kódu je posloupnost 10110101.

Z historického hlediska je velmi důležitý **inverzní kód** (*one's complement*). Přestože dnes neexistuje stroj pracující v tomto kódu, využijeme jej při hledání reprezentace čísla ve dvojkovém doplňkovém kódu.

Inverzní kód

Překlad termínu pro inverzní kód z angličtiny zní *jedničkový doplněk*, což přesně vystihuje princip tohoto kódu. Kladná čísla jsou reprezentována beze změny, u záporných čísel najdeme pro každý řád jeho doplněk do jedničky. Uvědomíme-li si, že binární kód je složen pouze z nul a jedniček, je zřejmé, že doplňkem nuly je jednička a doplňkem jedničky je nula. Stačí tedy provést inverzi všech bitů čísla, jak vyplývá z českého označení kódu (tab. 4.4).

Tabulka 4.4: *Reprezentace čtyřbitových čísel se znaménkem v inverzním kódu*

Kód	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Význam	0	1	2	3	4	5	6	7	−7	−6	−5	−4	−3	−2	−1	−0

Tento způsob vyjádření zachovává relace mezi kladnými a zápornými čísly v tom smyslu, že kladná i záporná čísla rostou zleva doprava. Lze porovnávat dvě kladná čísla ($5 > 2$) i dvě záporná čísla ($-2 > -5$). Problém s dvojí reprezentací nuly ale stále zůstává, interval zobrazitelných hodnot je opět $-2^{n-1} + 1$ až $2^{n-1} - 1$.

Příklad 4.5

Jak vyjádříme číslo −53 v inverzním kódu na prostoru 8 bitů? Nejdříve převedeme číslo 53 do dvojkové soustavy, výsledek doplníme zleva nulami, abychom vyplnili celou délku mřížky, a nakonec provedeme inverzi všech bitů čísla. Reprezentací čísla −53 v inverzním kódu je posloupnost 11001010.

Význam **dvojkového doplňkového kódu** (*two's complement*) tkví v tom, že odstraňuje dvojí reprezentaci nuly (viz tab. 4.5). Tvoří se stejně jako inverzní kód, ale k reprezentaci záporného čísla je navíc přičtena jednička. Jedná se o nejpoužívanější číselný kód pro celá čísla, na kterém jsou založeny prakticky všechny operace na moderních počítačích.

Dvojkový doplňkový kód

Tabulka 4.5: *Reprezentace čtyřbitových čísel se znaménkem ve dvojkovém doplňkovém kódu*

Kód	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Význam	0	1	2	3	4	5	6	7	−8	−7	−6	−5	−4	−3	−2	−1

Příklad 4.6

Jak vyjádříme číslo −53 ve dvojkovém doplňkovém kódu na prostoru 8 bitů? Nejdříve převedeme číslo 53 do dvojkové soustavy, výsledek doplníme zleva nulami, abychom vyplnili celou délku mřížky, provedeme inverzi všech bitů čísla a k výsledku přičteme jedničku. Reprezentací čísla −53 ve dvojkovém doplňkovém kódu je posloupnost 11001011.

Podívejme se nyní na způsob, kterým dvojkový doplňkový kód řeší problém dvou nul. Předpokládejme prostor 8 bitů. Číslo 0 na něm vyjádříme jako 00000000. Kdybychom chtěli vyjádřit číslo −0, dostali bychom nejdříve inverzí posloupnost 11111111 a po přičtení jedničky výsledek 100000000. Pozorný čtenář si v tuto chvíli uvědomí, že k reprezentaci čísla −0 potřebujeme 9 bitů. Připomeňme, že reprezentace v paměti je limitována délkou řádové mřížky, která v tomto případě má pouze 8 bitů. Devátý bit je projevem *přetečení* a bude ignorován, protože jej není kam zapsat. Tím máme zaručeno, že čísla 0 a −0 budou v počítači správně reprezentována stejným způsobem v podobě posloupnosti 00000000.

Vedlejším efektem jediné reprezentace nuly je možnost reprezentace většího počtu hodnot. Ve dvojkovém doplňkovém kódu můžeme na prostoru n bitů zobrazit 2^{n-1} kladných hodnot (včetně nuly) a stejný počet záporných hodnot. Protože číslo −0 již zobrazit nelze, zobrazíme místo něj číslo -2^{n-1} a tím získáme interval -2^{n-1} až $2^{n-1} - 1$.

Překlad anglického termínu pro dvojkový doplněk nabízí jiný způsob interpretace algoritmu. Hledáme-li reprezentaci čísla $-X$ na prostoru n bitů, pak ve skutečnosti hledáme reprezentaci *doplňku* čísla X do hodnoty 2^n, která už na tomto prostoru sama zobrazitelná není (tvoří modul řádové mřížky, viz str. 50). Hledaným doplňkem čísla X je číslo $2^n - X$.

Příklad 4.7

Ukažme si uvedený postup opět na příkladu reprezentace čísla −53 ve dvojkovém doplňkovém kódu na prostoru 8 bitů. Na tomto prostoru zobrazíme 2^8 hodnot v intervalu −128 až 127. Při hledání dvojkového doplňku čísla 53 stačí vyjádřit číslo $2^8 - 53 = 203$, které je určeno posloupností 11001011. K záměně mezi čísly −53 a 203 dojít nemůže, neboť číslo 203 neleží v intervalu zobrazitelných hodnot.

Odčítání v počítači

Na str. 28 se lze dočíst, že počítač všechny aritmetické operace převádí na sčítání. V okamžiku, kdy víme, jak jsou v počítači reprezentována záporná čísla, můžeme ukázat, jak sčítání a odčítání probíhá.

Příklad 4.8

0101 (5)	0101 (5)	1100 (−4)	1011 (−5)	0101 (5)
0010 (2)	1111 (−1)	0110 (6)	1011 (−5)	0101 (5)
0111 (7)	10100 (4)	10010 (2)	10110 (6)	1010 (−6)

První příklad ukazuje běžný a korektní součet dvou kladných hodnot. Druhý příklad ukazuje, jak probíhá odčítání. Místo výpočtu výrazu 5 − 1 se ve skutečnosti realizuje výpočet výrazu 5 + (−1), kde číslo −1 je vyjádřeno v doplňkovém kódu. Výsledek je teoreticky pětibitový, ale již víme, že přenos mimo řádovou mřížku je ignorován a hodnota 0100 skutečně reprezentuje číslo 4. Podobně je tomu i ve třetím příkladu. Čtvrtý a pátý

příklad demonstrují, co se může stát, když výsledek sčítání neleží v zobrazitelném intervalu. U čtvrtého příkladu má být správný výsledek −10, ale protože na 4 bitech lze reprezentovat pouze hodnoty v intervalu −8 až 7, je číslo −10 nahrazeno svým doplňkem, který získáme jako $2^4 - 10 = 6$. Podobně v pátém příkladu je výsledek 10, ale místo něj je zobrazen doplněk $-(2^4 - 10) = -6$.

Zamysleme se nyní nad výsledkem součtu čísla a jeho doplňku. Je zřejmé, že obě hodnoty budou mít stejnou binární reprezentaci, jejich součtem proto vždy dosáhneme přetečení, při kterém na uvedeném prostoru zůstane pouze posloupnost nul. Z matematického hlediska proto můžeme doplněk chápat jako *inverzní číslo*.

Aditivní kód

Poslední možností pro reprezentaci celých čísel se znaménkem je **aditivní kód** (*biased*, *offset binary*, též *excess-K*). Je konstruován tak, aby nula byla v binární reprezentaci posunuta přibližně do poloviny intervalu, proto bývá někdy nazýván kódem s posunutou nulou. Principem je přičtení konstanty k hodnotě každého čísla. Existují dvě varianty – *aditivní kód typu 0* s posunem o konstantu 2^{n-1} (tab. 4.6) a méně používaný *aditivní kód typu 1* s posunem o konstantu $2^{n-1} - 1$ (tab. 4.7).

Tabulka 4.6: *Reprezentace čtyřbitových čísel se znaménkem v kódu s posunutou nulou (aditivní kód typu 0)*

Kód	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Význam	−8	−7	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6	7

V tomto kódu se zápis kladného čísla liší od běžného způsobu reprezentace. Nezáporná mají v nejvyšším bitu jedničku, záporná čísla nulu. Díky tomu jsou záporná čísla v binární reprezentaci skutečně menší než kladná. Pozorný čtenář si zajisté povšimnul, že když bychom provedli inverzi nejvyššího bitu u všech hodnot, dostaneme dvojkový doplňkový kód. Interval zobrazitelných hodnot v aditivním kódu typu 0 je opět -2^{n-1} až $2^{n-1} - 1$.

Sčítání a odčítání lze provádět stejným způsobem jako u čísel bez znaménka, pouze nesmíme zapomínat, že každá hodnota je posunuta o konstantu k. Místo výrazu $x + y$ tak ve skutečnosti počítáme výraz $(x + k) + (y + k)$, jehož výsledkem je z matematického hlediska číslo $x + y + 2k$. My však požadujeme, aby součet byl posunut stejně jako sčítance pouze o jedno k, proto musíme od výsledku jedno k odečíst, abychom dostali $x + y + k$. Obdobně u odčítání místo výrazu $x - y$ počítáme $(x + k) - (y + k)$, což dává výsledek $x - y$, ale abychom dostali požadované $x - y + k$, musíme k výsledku jedno k přičíst. Násobení je složitější.

Příklad 4.9

Ve čtyřbitovém aditivním kódu s posunem o konstantu $k = 2^3$ budeme výpočet výrazu $-5 + 2$ provádět tak, že hodnoty −5 a 2 posuneme o k, sečteme a od výsledku jedno k odečteme. Obdobně výpočet výrazu $5 - 3$ provedeme tak, že hodnoty 5 a 3 posuneme o k, odečteme a k výsledku jedno k přičteme:

$$\begin{array}{rrl} -5 + 8 = 3 & 0011 & (-5) \\ 2 + 8 = 10 & 1010 & (2) \\ \hline -3 + 8 = 5 & 0101 & (-3) \end{array} \qquad \begin{array}{rrl} 5 + 8 = 13 & 1101 & (5) \\ 3 + 8 = 11 & -\ 1011 & (3) \\ \hline 2 + 8 = 10 & 1010 & (2) \end{array}$$

Pro úplnost uvedeme i podobu reprezentace čísel aditivním kódem typu 1. Jak je patrné z tab. 4.7, v tomto případě bude mít číslo nula v nejvyšším bitu nulu. Interval zobrazitelných hodnot pak bude $-2^{n-1} + 1$ až 2^{n-1}.

Tabulka 4.7: *Reprezentace čtyřbitových čísel se znaménkem v kódu s posunutou nulou (aditivní kód typu 1)*

Kód	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Význam	−7	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6	7	8

Rozšíření řádové mřížky

Pokud zjistíme, že na uvedeném prostoru nelze zobrazit požadované číslo, můžeme rozšířit řádovou mřížku přidáním dalších řádů. Prostor v paměti potřebný pro reprezentaci takového čísla zvětšíme tím, že zvolíme datovým typ s větším rozsahem. Úpravu zajistí aritmetická jednotka. Co se ale v tom okamžiku stane s reprezentací hodnot?

Příklad 4.10

Rozšíříme binární řádovou mřížku pro reprezentaci čtyřbitových hodnot tak, aby v ní bylo možné reprezentovat osmibitové hodnoty.

- Číslo v přímém kódu doplníme tak, aby v nejvyšším bitu zůstalo znaménko, tzn. přidáme nuly až za nejvyšší bit:
  $$0100 \Longrightarrow 0\underline{0000}100 \qquad 1001 \Longrightarrow 1\underline{0000}001$$
- Číslo v inverzním a dvojkovém doplňkovém kódu doplníme zleva o bity stejné hodnoty jako u nejvyššího bitu:
  $$0001 \Longrightarrow 0\underline{0000}001 \qquad 1110 \Longrightarrow 1\underline{1111}110$$
- Číslo v aditivním kódu typu 0 s posunem o konstantu 2^{n-1} upravíme tak, že za nejvyšší bit přidáme bity s opačnou hodnotou než u nejvyššího bitu:
  $$0101 \Longrightarrow 0\underline{1111}101 \qquad 1011 \Longrightarrow 1\underline{0000}011$$

Desítkový doplněk

Princip doplňkového kódu můžeme aplikovat i na jiné soustavy včetně desítkové, jak ukazuje tab. 4.8. Předpokládejme trojciferná desítková čísla, jejichž znaménko je určeno první číslicí zleva. Má-li tato číslice hodnotu 0–4, číslo je kladné, jinak je záporné.

Tabulka 4.8: *Reprezentace trojciferných číslic desítkovým doplňkový kódem*

Kód	000	001	002	...	497	498	499	500	501	502	...	997	998	999
Význam	0	1	2	...	497	498	499	−500	−499	−498	...	−3	−2	−1

Trojciferných desítkových čísel je 10^3, z toho polovina je kladných (včetně nuly) a druhá polovina záporných. Interval zobrazitelných hodnot je potom −500 až 499.

Příklad 4.11

Ukažme si nyní, jak bychom v takovém kódu určili hodnotu výrazu 241 − 198. Číslo −198 vyjádříme doplňkem $10^3 - 198 = 802$, poté obě čísla sečteme a ignorujeme přenos do nejvyššího řádu:

$$\begin{array}{rl} 241 & (241) \\ 802 & (-198) \\ \hline 1043 & (43) \end{array}$$

Z historie známe případy počítačů, které pracovaly ve dvojkově kódované desítkové soustavě, tzv. **kódu BCD** (*Binary Coded Decimal*). V dnešní době se stále můžeme s takovými stroji setkat. Tento způsob reprezentace je používán v kalkulačkách, digitálních měřicích přístrojích apod. Základním principem je uložení číslic desítkové soustavy do půlslabiky (*nibble*, 4 bity), jiné číslice (10 až 15) se nesmí vyskytovat. Existují dvě možnosti řešení: *Kód BCD*

- *zhuštěný tvar* – v jedné slabice jsou uloženy dvě číslice kódu BCD;
- *nezhuštěný tvar* – v jedné slabice je uložena jen jedna číslice kódu BCD.

Případné znaménko se v kódu BCD zobrazuje takovou kombinací bitů, která se nepoužívá pro číslice. Znaménko plus bývá například zobrazeno jako 1010 a znaménko minus jako 1011. Umístění znaménka ve slově může být libovolné a určuje ho programátor. V každém případě ale znaménko zabírá stejný počet bitů jako jednotlivé číslice.

Příklad 4.12

Ukažme, jak bude v kódu BCD reprezentováno číslo 38. Číslice 3 je reprezentována posloupností 0011, číslice 8 posloupností 1000. Ve zhuštěném tvaru na jednom bajtu tedy bude číslo 38 reprezentováno jako posloupnost 00111000, v nezhuštěném tvaru na dvou bajtech bude číslo reprezentováno jako posloupnost 00000011 00001000.

Pojďme si nyní shrnout všechny důležité poznatky o reprezentaci celých čísel v paměti počítače. Dozvěděli jsme se, že v *jedné slabice* (1 bajtu, 8 bitech) můžeme reprezentovat celkem 256 hodnot: *Shrnutí*

- čísla bez znaménka z intervalu 0 až 255;
- čísla se znaménkem ve dvojkovém doplňkovém kódu z intervalu −128 až 127;
- číslo se znaménkem v aditivním kódu s posunem o k z intervalu $-k$ až $255 - k$;
- číslo v kódu BCD z intervalu −9 až 99 ve zhuštěném tvaru.

Ve *dvou slabikách* (2 bajtech, 16 bitech) můžeme reprezentovat 65 536 hodnot:

- čísla bez znaménka z intervalu 0 až 65 535;
- čísla se znaménkem ve dvojkovém doplňkovém kódu z intervalu −32 768 až 32 767;
- číslo se znaménkem v aditivním kódu s posunem o k z intervalu $-k$ až $65\,535 - k$;
- číslo v kódu BCD z intervalu −999 až 9999 ve zhuštěném tvaru.

O významu binární posloupnosti vždy rozhoduje přiřazený *datový typ*.[18]

4.1.3 Racionální čísla s pevnou řádovou čárkou

Většina výpočtů v počítači probíhá v oboru racionálních čísel, která tvoří vlastní podmnožinu reálných čísel, tj. $\mathbb{Q} \subset \mathbb{R}$. Jsou to reálná čísla, která lze vyjádřit zlomkem (tedy podílem dvou celých čísel). Každé reálné číslo lze s libovolnou přesností nahradit racionálním číslem.

Při práci s racionálními čísly s pevnou řádovou čárkou (*fixed point*) je binární řádová mřížka rozdělena na úsek reprezentující celou část čísla (n bitů) a úsek reprezentující zlomkovou část čísla (m bitů). Zobrazitelné hodnoty leží v intervalu -2^{n-1} až $2^{n-1} - 2^{-m}$. Celá čísla, o kterých pojednávala kap. 4.1.2, jsou pouze speciálním případem racionálních čísel s pevnou řádovou čárkou pro $m = 0$. Stejně jako u celých čísel lze použít přímý i doplňkový kód, aby bylo možné reprezentovat i záporná čísla.

[18] Datové typy pro reprezentaci čísel v kódu BCD dnes nejsou v programovacích jazycích běžně k dispozici. Některé jazyky mají podporu kódu BCD jen v podobě emulátorů. Soudobý hardware kód BCD nepodporuje.

Přesnost zobrazení

Přesnost zobrazení vždy závisí na počtu bitů použitých pro reprezentaci, resp. na umístění řádové čárky. Všechna čísla jsou tak zobrazena se stejnou *absolutní přesností*. V důsledku toho jsou pak malá čísla zobrazována s nízkou *relativní přesností*. Při vědeckých výpočtech ale požadujeme, aby byla všechna čísla uložena se stejnou relativní přesností. Proto je reprezentace racionálních čísel v pevné řádové čárce vhodná pouze pro úlohy s omezenými nároky na přesnost nebo tam, kde hardware nepodporuje operace v pohyblivé řádové čárce.

Problém přesnosti můžeme názorně demonstrovat na příkladu reprezentace čísel na prostoru 8 bitů při umístění řádové čárky uprostřed, tj. pro $m = n = 4$. Nejmenší záporné číslo, které při takovém umístění řádové čárky můžeme zobrazit, je v doplňkovém kódu reprezentováno posloupností 10000000 a je to číslo −8. Největší kladné číslo, které můžeme zobrazit, je reprezentováno posloupností 11111111, která představuje číslo 7,9375.

Příklad 4.13

Znázorníme na prostoru 8 bitů s řádovou čárkou uprostřed čísla 4,625 a 4,6. Reprezentací čísla 4,625 je posloupnost 01001010, reprezentací čísla 4,6 je posloupnost 01001001. Vyjádříme-li z posloupnosti zpětně hodnotu v desítkové soustavě, zjistíme, že číslo 4,6 bylo uloženo nepřesně. Z jeho reprezentace byly uloženy všechny bity celé části, ale zlomková část byla zkrácena na 4 bity, což představuje číslo 4,5625.

$$4{,}625 = 0100{,}1010 = 1 \cdot 2^2 + 1 \cdot 2^{-1} + 1 \cdot 2^{-3} = 4{,}625$$
$$4{,}6 \quad = 0100{,}1001 = 1 \cdot 2^2 + 1 \cdot 2^{-1} + 1 \cdot 2^{-4} = 4{,}5625$$

4.1.4 Racionální čísla s pohyblivou řádovou čárkou

Vědecká notace

Reprezentaci racionálních čísel s pohyblivou řádovou čárkou (*floating point*) definuje standard IEEE[19] 754. Ten zavádí zápis čísel v **semilogaritmickém tvaru** tzv. vědeckou notací, kterou lze libovolné číslo vyjádřit kombinací základu soustavy a dvou dalších hodnot.

První hodnota, tzv. **mantisa**, upravuje číslo do tvaru, ve kterém je řádová čárka umístěna za první nenulovou číslici. Platí $1 \leq m < z$, kde m je mantisa a z základ soustavy.

Druhá hodnota, tzv. **exponent**, vyjadřuje počet řádů, o které bylo nutné řádovou čárku posunout. Může být kladný (čárka se posunula doleva), nebo záporný (posun doprava).

Každé racionální číslo pak můžeme vyjádřit přehledným způsobem jako součin mantisy a základu soustavy umocněného exponentem, např. číslo 0,000000004945 můžeme zapsat jako $4{,}945 \cdot 10^{-9}$.

Reprezentace v paměti

Paměťový prostor pro binární reprezentaci racionálních čísel s pohyblivou řádovou čárkou je rozdělen na tři části, ve kterých se ukládají znaménko, mantisa a exponent (obr. 4.2). Exponent je reprezentován v aditivním kódu s posunem o $2^{n-1} - 1$.

Znaménko	Exponent	Mantisa

Obrázek 4.2: *Organizace paměťového prostoru pro reprezentaci racionálního čísla s pohyblivou řádovou čárkou*

[19] IEEE (*Institute of Electrical and Electronics Engineers*) je mezinárodní organizace elektroinženýrů a informatiků. Patří společně s ISO, IEC a ANSI k nejvýznamnějším standardizačním organizacím. Zájemci mohou nalézt podrobnější informace na webových stránkách `http://www.ieee.org` nebo `www.ieee.cz`.

Normalizace mantisy

Protože v binární reprezentaci je první nenulovou číslicí vždy jednička, mantisa by vždy začínala jedničkou. Z tohoto důvodu se tato jednička až na výjimky neukládá (tzv. skrytý bit) a ušetřené místo je využito pro reprezentaci dalšího bitu zlomkové části, což zvyšuje přesnost zobrazení.

Formáty čísel

Standard IEEE 754 z roku 1985 rozlišuje několik standardizovaných formátů racionálních čísel, které se liší potřebným paměťovým prostorem a tím i přesností zobrazení hodnot. Pro znaménko je vždy využit 1 bit, zbytek bitů je rozdělen mezi exponent a především mantisu. **Jednoduchá přesnost** (*single precision*) reprezentuje číslo na 32 bitech, z nichž je 23 bitů vyhrazeno pro mantisu. **Dvojitá přesnost** (*double precision*) reprezentuje číslo na 64 bitech, z nichž mantisa zabírá 52 bitů. Existuje také **rozšířená přesnost** (*extended precision*), která reprezentuje číslo na 80 bitech, přičemž mantisa zabírá 64 bitů. V tomto formátu je však skrytý bit explicitně zobrazen jako nejvyšší bit mantisy.

V roce 2008 byly doplněny další standardizované formáty, které uspokojují stále vzrůstající nároky na přesnost. **Čtyřnásobná přesnost** (*quadruple precision*) reprezentuje číslo na 128 bitech, z nichž je 112 bitů rezervováno pro mantisu. Komu ani to nestačí, může využít **osminásobnou přesnost** (*octuple precision*), kde mantisa zabírá 236 bitů z celkových 256 bitů. Naopak pro případy, kdy není nutná vyšší přesnost, nabízí standard **poloviční přesnost** (*half precision*), kde mantisa zabírá 10 bitů z celkových 16 bitů. Tento formát je vhodný pro výpočty v oblasti grafiky nebo neuronových sítí.

Algoritmus převodu

Algoritmus převodu racionálního čísla do reprezentace s pohyblivou řádovou čárkou se skládá z těchto pěti kroků (platí pro všechny formáty s výjimkou extended precision):

1. Převedeme absolutní hodnotu čísla do binární reprezentace obvyklým způsobem.
2. Normalizujeme výsledek (mantisu vyjádříme dvojkově, exponent prozatím desítkově).
3. Převedeme exponent do aditivního kódu posunem o danou konstantu.
4. Zkrátíme mantisu na požadovanou přesnost a provedeme finální zaokrouhlení.
5. Nastavíme hodnotu znaménkového bitu.

Zaokrouhlení mantisy

Zaokrouhlování dle standardu IEEE 754 využívá tří **pomocných bitů**, které hardware uchovává nad rámec těch bitů, které lze ještě uložit do mantisy:

- *guard bit* – chrání před ztrátou významného bitu při přenosu do vyšších řádů;
- *round bit* – umožňuje zaokrouhlení výsledku po normalizaci;
- *sticky bit* – představuje logický součet všech zbývajících bitů.

Podívejme se nyní, jak se s pomocnými bity pracuje při zaokrouhlování. Připomeňme, že racionální číslo je vyjádřeno ve tvaru:

$$1\ ,\ b_1\ b_2\ b_3\ \ldots\ b_{n-1}\ b_n\ g\ r\ s\ x\ x\ x\ x\ \ldots\ x,$$

kde $b_1, b_2, \ldots b_n$ jsou bity mantisy, které lze reprezentovat, g, r, s jsou pomocné bity a x jsou další případné bity, jejichž hodnota není podstatná. Při rozhodování o způsobu zaokrouhlení racionálního čísla pak vycházíme z hodnoty pomocných bitů.

Je-li hodnota neuložených bitů menší než 0,5, pak má guard bit hodnotu 0 a zaokrouhlení lze provést jednoduše odříznutím těchto bitů:

$$1\ ,\ b_1\ b_2\ b_3\ \ldots\ b_{n-1}\ b_n\ \underbrace{0\ x\ x\ x\ x\ x\ x\ \ldots\ x}_{\text{odřízneme}}$$

Je-li hodnota neuložených bitů rovna přesně 0,5, pak má guard bit hodnotu 1, ostatní bity napravo od něj hodnotu 0 a pro další postup je rozhodující nejnižší bit mantisy b_n:

$$1\ ,\ b_1\ b_2\ b_3\ \dots\ b_{n-1}\ 0\ \underbrace{1\ 0\ 0\ 0\ 0\ 0\ 0\ \dots\ 0}_{\text{odřízneme}}$$

$$1\ ,\ b_1\ b_2\ b_3\ \dots\ b_{n-1}\ 1\ \underbrace{1\ 0\ 0\ 0\ 0\ 0\ 0\ \dots\ 0}_{\text{inkrementujeme mantisu}}$$

Je-li hodnota neuložených bitů větší než 0,5, pak má guard bit hodnotu 1 a ostatní bity napravo od něj neobsahují samé nuly. V takovém případě vždy provedeme inkrement mantisy:

$$\begin{array}{l} 1\ ,\ b_1\ b_2\ b_3\ \dots\ b_{n-1}\ b_n\ 1\ 0\ 1\ x\ x\ x\ x\ \dots\ x \\ 1\ ,\ b_1\ b_2\ b_3\ \dots\ b_{n-1}\ b_n\ \underbrace{1\ 1\ x\ x\ x\ x\ x\ \dots\ x}_{\text{inkrementujeme mantisu}} \end{array}$$

Příklad 4.14

Podívejme se, jak bude v paměti počítače reprezentováno racionální číslo 64,2 ve standardizovaném formátu single precision.

1. Nejdříve převedeme číslo 64,2 do binárního kódu a získáme nekonečnou posloupnost 1000000,00110011001100110011..., neboť zlomková část má periodický rozvoj.
2. Výsledek normalizujeme do tvaru $1{,}00000000110011001100110011\ldots \cdot 2^6$.
3. Upravíme exponent přičtením 127 a získáme číslo 6 + 127 = 133, které je reprezentováno binární posloupností 10000101.
4. Vezmeme část mantisy za řádovou čárkou a zaokrouhlíme (zkrátíme) ji na požadovanou přesnost 23 bitů. Získáme posloupnost 00000000110011001100110.
5. Nastavíme znaménkový bit na 0 a poskládáme výslednou reprezentaci čísla.

0	10000101	00000000110011001100110

$6{,}42 \cdot 10^1$

Příklad 4.15

Nyní ukažme, jak bude se stejnou přesností reprezentováno číslo −12,5.

1. Nejdříve převedeme číslo 12,5 do binárního kódu a získáme posloupnost 1100,1.
2. Výsledek normalizujeme do tvaru $1{,}1001 \cdot 2^3$.
3. Upravíme exponent přičtením 127 a získáme číslo 3 + 127 = 130, které je reprezentováno posloupností 10000010.
4. Vezmeme část mantisy za řádovou čárkou a upravíme na požadovanou přesnost 23 bitů. Protože tato část mantisy má pouze 4 bity, musíme ji rozšířit přidáním 19 nulových bitů. Získáme posloupnost 10010000000000000000000.
5. Nastavíme znaménkový bit na 1 a poskládáme výslednou reprezentaci čísla.

1	10000010	10010000000000000000000

$-1{,}25 \cdot 10^1$

Příklad 4.16

Pro ukázku zaokrouhlení budeme uvažovat reprezentaci čísla 0,1.

1. Nejdříve převedeme číslo 0,1 do binárního kódu a získáme nekonečnou posloupnost 0,0001100110011001100110011001100..., neboť zlomková část má periodický rozvoj.
2. Výsledek normalizujeme do tvaru $1{,}10011001100110011001100\ldots \cdot 2^{-4}$.
3. Upravíme exponent přičtením 127 a získáme číslo $-4 + 127 = 123$, které je reprezentováno posloupností 01111011.
4. Vezmeme část mantisy za řádovou čárkou a upravíme na požadovanou přesnost 23 bitů. Získáme posloupnost 10011001100110011001100. Protože pomocné bity mají hodnotu 111, provedeme zaokrouhlení pomocí inkrementu mantisy. Dostaneme posloupnost 10011001100110011001101.
5. Nastavíme znaménkový bit na 0 a poskládáme výslednou reprezentaci čísla.

0	01111011	10011001100110011001101

$1{,}0 \cdot 10^{-1}$

Pro úplnost dodejme, že standardizované formáty umožňují také reprezentaci tzv. *denormalizovaných čísel*. Taková čísla jsou reprezentována libovolnou nenulovou mantisou a nulovým exponentem. Tento způsob reprezentace se využívá pro velmi malé hodnoty, které nejdou uložit normalizovaně. Mantisa je uložena *s pevnou řádovou čárkou* umístěnou za nejvyšším bitem, který má v tomto případě nulovou hodnotu.

Denormalizovaná čísla

Příklad 4.17

Uveďme nyní pro ilustraci několik příkladů normalizovaných a denormalizovaných čísel zobrazitelných ve standardu single precision. Jsou mezi nimi i nejmenší normalizované a nejmenší denormalizované číslo.

$^0/_1$	00000010	00000000000000000000000
$^0/_1$	00000001	00000000000000000000000
$^0/_1$	00000000	10000000000000000000000
$^0/_1$	00000000	01000000000000000000000
$^0/_1$	00000000	00000000000000000000001

$\pm 1{,}0 \cdot 2^{2-127} = \pm 2^{-125}$

$\pm 1{,}0 \cdot 2^{1-127} = \pm 2^{-126}$

$\pm 0{,}1 \cdot 2^{1-127} = \pm 2^{-127}$

$\pm 0{,}01 \cdot 2^{1-127} = \pm 2^{-128}$

$\pm 0{,}m \cdot 2^{1-127} = \pm 2^{-149}$

Jsou-li v binární reprezentaci mantisy i exponentu samé nuly (znaménko může být libovolné), je takové číslo zobrazeno jako *nula*. V semilogaritmickém tvaru je takové číslo vyjádřeno jako $\pm 1{,}0 \cdot 2^{e_{\min}-1}$, kde $e_{\min}$ je nejnižší možná mocnina exponentu.[20]

Vyhrazené hodnoty

[20] Připomeňme, že řádová čárka v mantise je umístěna za první nenulovou číslicí, kterou navíc při reprezentaci zanedbáváme (tzv. skrytý bit). Tím jsme se ale připravili o možnost zakódování nuly. V binární reprezentaci čísla nula není žádná nenulová číslice, řádovou čárku tedy není kam umístit. Proto je za nulu považováno každé „příliš malé“ číslo, které má v binární reprezentaci na prvních 126 bitech zlomkové části samé nuly. Při pokusu o reprezentaci bude takové číslo vždy zatíženo podtečením a bude zaokrouhleno na nulu.

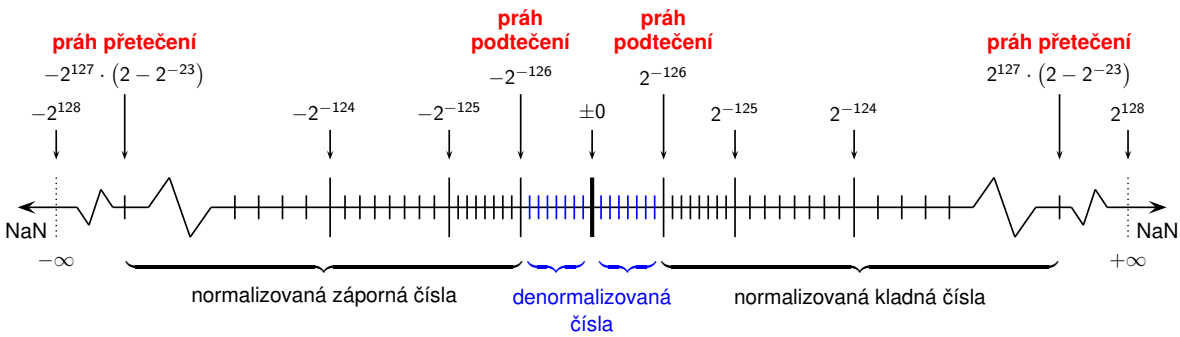
Je-li číslo příliš velké, je zobrazeno jako *nekonečno*. V semilogaritmickém tvaru je takové číslo vyjádřeno jako $\pm 1{,}0 \cdot 2^{e_{\max}+1}$, kde $e_{\max}$ je nejvyšší možná mocnina exponentu.

Když se pokusíme provést operaci, jejíž výsledek není definován (%, $\log 0$, $\sqrt{-1}$ atd.), dostaneme hodnotu NaN (*Not a Number*). Ta je reprezentována libovolnou nenulovou mantisou (nulová představuje nekonečno) a exponentem $e_{\max} + 1$.

Tabulka 4.9: *Přehled reprezentovatelných hodnot včetně denormalizovaných čísel a vyhrazených hodnot*

$^0/_1$	00...00	00...00	±0 vyhrazená hodnota
$^0/_1$	00...00	00...01 ⋮ 11...11	$\pm 0{,}m \cdot 2^{1-b}$ denormalizované číslo
$^0/_1$	00...01 ⋮ 11...10	00...01 ⋮ 11...11	$\pm 1{,}m \cdot 2^{e-b}$ normalizované číslo

$^0/_1$	11...11	00...00	±Inf vyhrazená hodnota
$^0/_1$	11...11	00...01 ⋮ 01...11	sNaN vyhrazená hodnota
$^0/_1$	11...11	10...00 ⋮ 11...11	qNaN vyhrazená hodnota



Obrázek 4.3: *Zobrazitelné hodnoty ve formátu single precision podle standardu IEEE 754*

Aritmetické operace

Během výpočtů dochází neustále k zaokrouhlování a následné ztrátě přesnosti, kterou způsobuje omezená přesnost vyjádření na daném prostoru a aproximace skutečných hodnot přibližnými. Důsledkem je potom neplatnost asociativních, komutativních a distributivních zákonů.

Předpokládejme zápis čísla n v semilogaritmickém tvaru $n = m_n \cdot z^{e_n}$, kde m_n je mantisa čísla n, z je základ soustavy a e_n je exponent čísla n. Pak pro čísla x a y vyjádřené s pohyblivou řádovou čárkou obecně platí:

$$\left.\begin{aligned} x + y &= \left(m_x \cdot z^{e_x - e_y} + m_y\right) \cdot z^{e_y} \\ x - y &= \left(m_x \cdot z^{e_x - e_y} - m_y\right) \cdot z^{e_y} \end{aligned}\right\} \quad e_x \le e_y$$

$$x \cdot y = \left(m_x \cdot m_y\right) \cdot z^{e_x + e_y}$$

$$\frac{x}{y} = \left(\frac{m_x}{m_y}\right) \cdot z^{e_x - e_y}$$

Algoritmus sčítání racionálních čísel s pohyblivou řádovou čárkou se skládá z následujících čtyř kroků:

1. Srovnání exponentů – odečteme exponenty a mantisu čísla s menším exponentem posuneme doprava o počet bitů, který je roven rozdílu exponentů.
2. Provedení aditivní operace – sečteme mantisy obou čísel.
3. Normalizace výsledku – určíme počet nul mezi řádovou čárkou a první platnou číslicí součtu mantis, posuneme součet doleva o tolik míst, kolik nul bylo nalezeno za řádovou čárkou a původní exponent zmenšíme o počet nalezených nul.
4. Zaokrouhlení výsledku – ošetříme pomocné bity, zaokrouhlíme dle zvolené metody a pokud při zaokrouhlení došlo k přenosu, znovu normalizujeme.

Algoritmus pro násobení je velmi jednoduchý: exponenty sečteme, mantisy vynásobíme, výsledek normalizujeme a zaokrouhlíme. Znaménkový bit součinu získáme aplikací operace xor na znaménkové bity činitelů.

Příklad 4.18

Podívejme se nyní, jak proběhne součet hodnot 1234,5 a 2,75. Nejdříve hodnoty vyjádříme ve standardizovaném formátu single precision:

0	10001001	00110100101000000000000	1234,5
0	10000000	01100000000000000000000	2,75

Nyní provedeme porovnání exponentů a posun menšího z čísel tak, aby jeho exponent odpovídal exponentu většího z čísel:

$$
\begin{aligned}
1234{,}5 &= 1{,}00110100101000000000000 \cdot 2^{10} \\
2{,}75 &= 1{,}01100000000000000000000 \cdot 2^{1} \\
&= 0{,}10110000000000000000000 \cdot 2^{2} \\
&= 0{,}01011000000000000000000 \cdot 2^{3} \\
&\ddots \\
&= 0{,}00000000101100000000000000000000 \cdot 2^{10}
\end{aligned}
$$

Sečteme obě čísla:

$$
\begin{aligned}
1234{,}5 &= 1{,}00110100101000000000000 \cdot 2^{10} \\
2{,}75 &= 0{,}00000000101100000000000000000000 \cdot 2^{10} \\
\hline
&= 1{,}00110101010100000000000000000000 \cdot 2^{10}
\end{aligned}
$$

Výsledek je v normalizovaném tvaru, není tedy potřeba provádět další úpravy. Pomocný guard bit má hodnotu 0, zaokrouhlení proto provedeme odříznutím nepotřebných bitů. Dostáváme výsledek, který je v tomto případě přesný stejně jako oba sčítance:

0	10001001	00110100101000000000000	1234,5
0	10000000	01100000000000000000000	2,75
0	10001001	00110101010100000000000	1237,25

Příklad 4.19

Je zřejmé, že v mnoha případech dojde ke ztrátě přesnosti již v okamžiku reprezentace samotných sčítanců. Jasnou představu o aproximaci hodnot a jejím důsledku na přesnost výpočtů získáme při pohledu na součet čísel 0,33 a 979,7. Nejdříve zobrazíme obě hodnoty ve formátu single precision. Již v této fázi můžeme pozorovat nepřesnost zobrazení, což lze snadno ověřit zpětným převodem hodnot do desítkové soustavy:

0	01111101	01010001111010111000011	0,330000013113
0	10001000	11101001110110011001101	979,7700012207

Nyní provedeme porovnání exponentů a posun menšího z čísel tak, aby jeho exponent odpovídal exponentu většího z čísel:

$$
\begin{aligned}
0{,}33 &= 1{,}01010001111010111000011 \cdot 2^{-2} \\
&= 0{,}10101000111101011100001 1 \cdot 2^{-1} \\
&= 0{,}01010100011110101110000 11 \cdot 2^{0} \\
&= 0{,}00101010001111010111000011 \cdot 2^{1} \\
&= 0{,}00010101000111101011100001 1 \cdot 2^{2} \\
&= 0{,}00001010100011110101110000 11 \cdot 2^{3} \\
&= 0{,}00000101010001111010111000011 \cdot 2^{4} \\
&= 0{,}00000010101000111101011100001 1 \cdot 2^{5} \\
&= 0{,}00000001010100011110101110000 11 \cdot 2^{6} \\
&= 0{,}00000000101010001111010111000011 \cdot 2^{7} \\
&= 0{,}00000000010101000111101011100001 1 \cdot 2^{8} \\
&= 0{,}00000000001010100011110101110000 11 \cdot 2^{9} \\
979{,}7 &= 1{,}11101001110110011001101 \cdot 2^{9}
\end{aligned}
$$

Sečteme obě čísla:

$$
\begin{aligned}
0{,}33 &= 0{,}00000000001010100011110101110000 11 \cdot 2^{9} \\
979{,}7 &= 1{,}11101001110110011001101 \cdot 2^{9} \\
\hline
&= 1{,}11101010000000111101011101110000 11 \cdot 2^{9}
\end{aligned}
$$

Výsledek je v normalizovaném tvaru, není tedy potřeba provádět další úpravy. Pomocné bity mají hodnotu 101, proto zaokrouhlíme inkrementem mantisy. Dostáváme výsledek, který je zatížen ztrátou přesnosti i v důsledku aproximovaných hodnot obou sčítanců:

0	01111101	01010001111010111000011	0,330000013113
0	10001000	11101001110110011001101	979,7700012207
0	10001000	11101010000000111101100	980,030029297

Příklad 4.20

Násobení racionálních čísel s pohyblivou řádovou čárkou si ukážeme na příkladu součinu hodnot 2,5 a 12,75. Nejdříve hodnoty vyjádříme ve standardizovaném formátu single precision:

0	10000000	01000000000000000000000	2,5
0	10000010	10011000000000000000000	12,75

Nyní sečteme exponenty (pozor na aditivní kód a práci s posunem):

$$\begin{array}{rl} 10000000 & 128 - 127 = 1 \\ 10000010 & 130 - 127 = 3 \\ \hline 100000010 & 258 - 254 = 4 \end{array}$$

Poté následuje násobení mantis:

$$\begin{array}{rcll} 2{,}5 & = & 1{,}01000000000000000000000 \cdot 2^1 & 101000\ldots \\ \cdot\ 12{,}75 & = & 1{,}10011000000000000000000 \cdot 2^3 & \cdot\ 110011\ldots \\ \hline & & & 101000\ldots \\ & & & 101000\ldots \\ & & & 000000\ldots \\ & & & 000000\ldots \\ & & & 101000\ldots \\ & & & 101000\ldots \\ \hline 31{,}875 & = & 1{,}11111110000000000000000 \cdot 2^4 & 11111111000\ldots \end{array}$$

Součin je v normalizovaném tvaru, není proto potřeba provádět další úpravy. Pomocné bity mají nulovou hodnotu, zaokrouhlení tedy provedeme jednoduše odříznutím nepotřebných bitů. Na závěr ještě zjistíme hodnotu znaménkového bitu (0 xor 0 = 0) a dostáváme výsledek:

0	10000000	01000000000000000000000	2,5
0	10000010	10011000000000000000000	12,75
0	10000011	11111110000000000000000	31,875

Racionální datové typy

Nyní se podíváme ještě na běžné datové typy pro reprezentaci racionálních čísel s pohyblivou řádovou čárkou (tab. 4.10), které vycházejí ze standardu IEEE 754. Rozsah zobrazitelných hodnot ve formátu single precision přehledně ilustruje obr. 4.3 (str. 62).

Tabulka 4.10: *Přehled racionálních datových typů*

Velikost	Rozsah hodnot	Znam.	Šířka expon.	Šířka mantisy	Pascal	C++
4 B	2^{-126} až $\left(2-2^{-23}\right)\cdot 2^{127}$	ano	8 b	23 b	`single`	`float`
8 B	2^{-1022} až $\left(2-2^{-52}\right)\cdot 2^{1023}$	ano	11 b	52 b	`double`	`double`
10 B	2^{-16382} až $\left(2-2^{-64}\right)\cdot 2^{16383}$	ano	15 b	64 b	`extended`	`long double`

Kontrolní otázky

1. K čemu slouží binární řádová mřížka a jakými vlastnostmi je popsána?
2. Kdy může dojít k přetečení nebo podtečení a jak se tyto stavy projevují?
3. Jakým způsobem a na jakém prostoru jsou v paměti počítače obvykle reprezentována celá čísla bez znaménka?
4. Jaký je princip přímého kódování celých čísel se znaménkem?
5. Jaký je princip inverzního kódování celých čísel se znaménkem?
6. Z jakého důvodu se dnes už nepoužívá přímé a inverzní kódování?
7. Jaké vlastnosti má reprezentace záporných čísel dvojkovým doplňkovým kódem?
8. Jaký je interval zobrazitelných hodnot v celočíselném datovém typu na n bitech, uvažujeme-li čísla se znaménkem ve dvojkovém doplňkovém kódu?
9. Na jakém principu je založen aditivní kód a k čemu se používá?
10. K jakému účelu slouží kód BCD a jak jsou v něm reprezentována čísla?
11. Pro která čísla je výhodnější reprezentace s pohyblivou řádovou čárkou?
12. Jak vypadá racionální číslo zapsané v semilogaritmickém tvaru?
13. Co vyjadřují mantisa a exponent racionálního čísla?
14. Které standardizované formáty slouží pro reprezentaci racionálních čísel s pohyblivou řádovou čárkou a jak jsou tyto formáty organizovány v paměti?
15. Jaký je rozsah zobrazitelných hodnot ve formátu single precision?
16. Které vyhrazené hodnoty mohou být výsledkem operace nad racionálními čísly s pohyblivou řádovou čárkou a jak jsou tyto hodnoty reprezentovány v paměti?
17. Co jsou denormalizovaná čísla a jak jsou reprezentována v paměti?

4.2 Reprezentace textových dat

Data v paměti počítače jsou reprezentována v podobě binárních sekvencí. Pozorného čtenáře proto nepřekvapí, že při práci s texty je nezbytné každý znak *zakódovat* do jednoznačné binární posloupnosti. Soubor kódů se označuje jako **znaková sada** (*character set*). *Znaková sada*

Znaky v počítači můžeme podle jejich významu a způsobu použití rozdělit do dvou skupin. Početnější skupinu tvoří **zobrazitelné znaky** (*printable characters*), které slouží pro zápis textové informace v podobě písmen, číslic, interpunkčních znamének, matematických symbolů a dalších znaků vyjádřených textově. *Zobrazitelné znaky*

Druhou skupinu tvoří **řídicí znaky** (*control characters*), které byly původně určeny pro ovládání periferních zařízení (tiskárna, dálnopis apod.), dnes jimi také ovládáme programy. Nejsou to znaky v obvyklém významu, ale spíše povely, které mohou být v každém systému implementovány odlišným způsobem. Typickými příklady řídicích znaků jsou escape, tabulátor nebo znak konce řádku. *Řídicí znaky*

4.2.1 Jednobajtové znakové kódy

Nejstarší znakové kódy potřebovaly pro reprezentaci každého znaku 1 bajt paměti, bylo jimi tedy možné rozlišit až 256 různých znaků.

Okolo let 1963–1964 navrhla firma IBM osmibitový znakový kód **EBCDIC** (*Extended Binary Coded Decimal Interchange Code*), který vycházel z kódu používaného pro děrné štítky a kódu BCD pro periferní zařízení IBM. Kód byl určen pro sálové počítače. První čtveřice bitů vyjadřovala kategorii znaku, druhá čtveřice určovala konkrétní znak. Existovalo více variant pro různé země, v některých asijských zemích se používala také rozšířená verze kódu, která pro zobrazení čínských, japonských či korejských znaků používala 2 bajty. *EBCDIC*

Na pozicích #0 až #63 a #255 byly umístěny řídicí znaky, na pozicích #64 až #254 se nacházely zobrazitelné znaky. Velkou nevýhodou tohoto kódu bylo uspořádání znaků anglické abecedy, které netvořily spojitou posloupnost. To znemožňovalo použití regulárních výrazů, například `[a-z]` pro výběr všech malých písmen abecedy.

Příklad 4.21

Reprezentace řetězce „Hello World!“ ve znakovém kódu EBCDIC (číselné kódy jednotlivých znaků jsou uvedeny v šestnáctkové soustavě):

```
C8 85 93 93 96 40 E6 96 99 93 84 4F
 H  e  l  l  o  ␣  W  o  r  l  d  !
```

V roce 1968 byl dokončen vývoj znakového kódu **ASCII** (*American Standard Code for Information Interchange*), který byl v roce 1972 standardizován pod označením ISO 646. Kód byl původně sedmibitový, osmý bit byl použit pro zabezpečení při přenosech. *ASCII*

Řídicí znaky jsou umístěny na pozicích #0 až #31 a #127, ostatní pozice zabírají zobrazitelné znaky. Rozložení znaků znázorňuje tab. 4.11. Souřadnice řádku a sloupce tvoří šestnáctkově vyjádřený kód znaku, například mezera $20_{16} = 32_{10}$, znak „@“ $40_{16} = 64_{10}$.[21]

[21] V operačním systému Windows lze každý zobrazitelný znak z kódu ASCII zapsat kombinací klávesy Alt a čísla znaku v desítkové soustavě. V operačním systému Linux lze pro zápis libovolného znaku z kódu Unicode použít kombinaci kláves Ctrl + Shift + u a čísla znaku v šestnáctkové soustavě.

Tabulka 4.11: *Rozložení znaků ve znakovém kódu ASCII (ISO 646)*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	␣	!	”	#	$	%	&	’	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Malá a velká písmena jsou od sebe vzdálena o 32 pozic a v binární reprezentaci se liší jen jedním bitem – srovnejme například znaky „C“ (#67, 01000011) a „c“ (#99, 01100011).

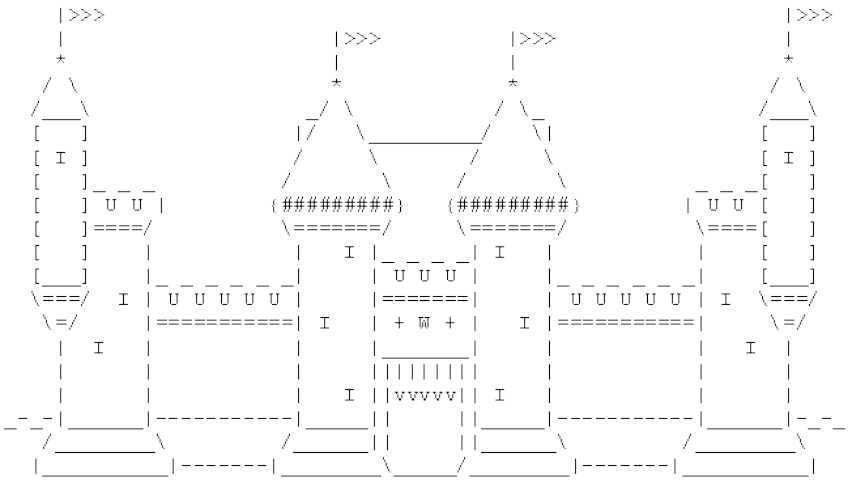
Příklad 4.22

Reprezentace řetězce „Hello World!“ ve znakovém kódu ASCII (číselné kódy jednotlivých znaků jsou uvedeny v šestnáctkové soustavě):

```
48 65 6C 6C 6F 20 57 6F 72 6C 64 21
 H  e  l  l  o  ␣  W  o  r  l  d  !
```

Základní část ASCII

Tato tzv. **základní část** tabulky ASCII (*standard character set*) je společná pro všechny světové jazyky používající latinskou abecedu. S rozvojem IBM PC však brzy začalo být zřejmé, že 95 zobrazitelných znaků nemůže stačit. Umožňují totiž plnohodnotně psát texty pouze v jazycích, které nemají specifické znaky s diakritickými znaménky. Nicméně i s takto omezenou množinou znaků lze vytvářet zajímavé útvary, jak ukazuje obr. 4.4.



Obrázek 4.4: *Kresba složená ze znaků pomocí techniky zvané „ASCII art“*

Žádný světový jazyk[22] se bez **národních znaků** (znaků s diakritikou) neobejde, původně ale nebyl pro jejich reprezentaci navržen žádný znakový kód. Pro zajištění podpory národních znaků v kódu ASCII byl proto nakonec použit osmý (původně kontrolní) bit, což umožnilo reprezentaci dalších 128 zobrazitelných znaků. Tzv. **rozšířená část** tabulky ASCII (*extended character set*) obsahuje národní znaky jednotlivých jazyků, použití je však zcela nejednotné. Problém spočívá v tom, že jazyků je mnoho a každý používá jinou množinu národních znaků. Proto existuje mnoho variant rozšířené části podle jednotlivých národních prostředí. A aby to bylo ještě složitější, pro jeden jazyk může existovat i více variant kódu. *Rozšířená část ASCII*

Čeština a slovenština potřebují 44 národních znaků. Postupem času pro ně vzniklo mnoho variant znakových kódů, které se liší pouze jiným uspořádáním znaků v tabulce. Některé z nich se používají dodnes (viz tab. 4.13 na str. 70). Znakové sady vytvářely naprosto nekoordinovaně národní i mezinárodní standardizační organizace, firmy i jednotlivci.

Kód bratrů Kamenických[23] (též MJK, KEYBCS2, CP895) vznikl v roce 1985 a byl určen pro osobní počítače s operačním systémem MS-DOS. Vychází ze znakové sady CP437 používané na počítačích IBM PC, ve které jsou znaky na pozicích #128 až #171 nahrazeny českými a slovenskými národními znaky na bázi vizuální podobnosti. *Kameničtí*

Kód *PC Latin 2* (též IBM Latin 2, CP852) byl určen pro počítače s operačním systémem MS-DOS používající středoevropské jazyky. Obsahuje proto národní znaky z češtiny, slovenštiny, polštiny, maďarštiny, ale také z albánštiny, rumunštiny, srbochorvatštiny apod. *PC Latin 2*

V rámci RVHP byl v Sovětském svazu vyvinut národní standard KOI-8 (Код Обмена Информацией, 8 бит), jehož varianta *KOI-8 ČS2* byla normou ČSN 36 9103 v r. 1989 doporučena pro češtinu a slovenštinu. Zajímavostí je, že obsahuje zvláštní znaky pro velké a malé „ch“. Kód se používal v počítačích se systémem CP/M, dnes se již aktivně nevyužívá. *KOI-8*

Komise ISO publikovaly v letech 1987–1988 prvních osm kódových tabulek standardu ISO 8859. Dalšími návrhy byl počet tabulek zvýšen až na šestnáct (tab. 4.12), později byla jedna zrušena (resp. nahrazena vícebajtovým kódem).

Tabulka 4.12: *Varianty znakového kódu ISO 8859*

Kód	Popis
ISO 8859-1	jazyky západní Evropy
ISO 8859-2	jazyky východní Evropy
ISO 8859-3	jazyky jižní Evropy
ISO 8859-4	jazyky severní Evropy
ISO 8859-5	slovanské jazyky s cyrilicí
ISO 8859-6	arabština
ISO 8859-7	řečtina
ISO 8859-8	hebrejština

Kód	Popis
ISO 8859-9	turečtina
ISO 8859-10	jazyky severní Evropy, zdokonalená verze
ISO 8859-11	thajština
ISO 8859-12	dévanágarí, v roce 1997 zrušeno
ISO 8859-13	baltské jazyky
ISO 8859-14	keltské jazyky
ISO 8859-15	jazyky západní Evropy, zdokonalená verze
ISO 8859-16	jazyky jihovýchodní Evropy

Varianta *ISO Latin 2* (ISO 8859-2) je určena pro většinu středoevropských a východoevropských jazyků, které používají latinku nebo latinskou transkripci, mezi něž patří i čeština a slovenština. Lze ji ale použít i pro němčinu a finštinu. Tuto standardizovanou znakovou sadu dříve používaly operační systémy třídy Unix. *ISO 8859-2*

[22] Prakticky se národním znakům v současnosti nevyhne ani angličtina – potřebuje je pro desítky slov přejatých z jiných jazyků, zejména francouzštiny.

[23] Autory svého času populárního kódu jsou Marian Kamenický (* 1943) a Jiří Kamenický (* 1949).

Tabulka 4.13: *Základní a rozšířený kód ASCII – různé varianty použitelné (nejen) pro češtinu a slovenštinu*

hexadecimálně	desítkově	ASCII	hexadecimálně	desítkově	ASCII	hexadecimálně	desítkově	ISO Latin 1 (ISO 8859-1)	Windows Latin 1 (CP1252)	ISO Latin 2 (ISO 8859-2)	Windows-1250 (CP1250)	Mac OS Macintosh CE	Kamenčtí (KEYBCS2)	KOI-8 ČS2 (KOI8CS)	IBM Standard (CP437)	IBM PC Latin 1 (CP850)	IBM PC Latin 2 (CP852)	hexadecimálně	desítkově	ISO Latin 1 (ISO 8859-1)	Windows Latin 1 (CP1252)	ISO Latin 2 (ISO 8859-2)	Windows-1250 (CP1250)	Mac OS Macintosh CE	Kamenčtí (KEYBCS2)	KOI-8 ČS2 (KOI8CS)	IBM Standard (CP437)	IBM PC Latin 1 (CP850)	IBM PC Latin 2 (CP852)
00	0	NUL	40	64	@	80	128	⁂	€	⁂	€	Ä	Č	Ç	Ç	Ç	Ç	c0	192	À	À	Ŕ	Ŕ	ŋ	└	┌	└	└	└
01	1	SOH	41	65	A	81	129	·		·		Ā	ü	ù	ü	ü	ü	c1	193	Á	Á	Á	Á	Ń	┴	á	┴	┴	┴
02	2	STX	42	66	B	82	130	‚	‚	‚	‚	ā	é	è	é	é	é	c2	194	Â	Â	Â	Â	¬	┬	└	┬	┬	┬
03	3	ETX	43	67	C	83	131	□	ƒ	□		É	ď	§	â	â	â	c3	195	Ã	Ã	Ă	Ă	√	├	č	├	├	├
04	4	EOT	44	68	D	84	132	„	„	„	„	Ą	ä	ß	ä	ä	ä	c4	196	Ä	Ä	Ä	Ä	ń	─	ď	─	─	─
05	5	ENQ	45	69	E	85	133	□	…	□	…	Ö	Ď	£	à	à	ů	c5	197	Å	Å	Ĺ	Ĺ	Ň	┼	ě	┼	┼	┼
06	6	ACK	46	70	F	86	134	□	†	□	†	Ü	Ť	Æ	å	å	ć	c6	198	Æ	Æ	Ć	Ć	∆	╞	ŕ	╞	ã	Ã
07	7	BEL	47	71	G	87	135	□	‡	□	‡	á	č	ω	ç	ç	ç	c7	199	Ç	Ç	Ç	Ç	«	╟	─	╟	Ã	ã
08	8	BS	48	72	H	88	136	□	ˆ	□		ą	ě	Å	ê	ê	ł	c8	200	È	È	Č	Č	»	╚	ü	╚	╚	╚
09	9	HT	49	73	I	89	137	□	‰	□	‰	Č	Ě	æ	ë	ë	ë	c9	201	É	É	É	É	…	╔	í	╔	╔	╔
0a	10	LF	4a	74	J	8a	138	□	Š	□	Š	ä	Ľ	ω	è	è	Ő	ca	202	Ê	Ê	Ę	Ę		╩	ů	╩	╩	╩
0b	11	VT	4b	75	K	8b	139	□	‹	□	‹	č	Í	å	ï	ï	ő	cb	203	Ë	Ë	Ë	Ë	ň	╦	Í	╦	╦	╦
0c	12	FF	4c	76	L	8c	140	□	Œ	□	Ś	Ć	ľ		î	î	î	cc	204	Ì	Ì	Ě	Ě	Ő	╠	ľ	╠	╠	╠
0d	13	CR	4d	77	M	8d	141	ť		ť	Ť	ć	Í		ì	ì	Ź	cd	205	Í	Í	Í	Í	Ď	═	ó	═	═	═
0e	14	SO	4e	78	N	8e	142	ž	Ž	ž	Ž	é	Ä	¤	Ä	Ä	Ä	ce	206	Î	Î	Î	Î	ő	╬	ň	╬	╬	╬
0f	15	SI	4f	79	O	8f	143	ź		ź	Ź	Ż	Á	ò	Å	Å	Ć	cf	207	Ï	Ï	Ď	Ď	Ō	╧	ó	╧	¤	¤
10	16	DLE	50	80	P	90	144	□		□		ź	É	ì	É	É	É	d0	208	Đ	Đ	Đ	Đ	–	╨	ô	╨	ð	đ
11	17	DC1	51	81	Q	91	145	□	‘	□	‘	Ď	ž		æ	æ	Ľ	d1	209	Ñ	Ñ	Ń	Ń	—	╤	ä	╤	Đ	Đ
12	18	DC2	52	82	R	92	146	□	’	□	’	í	Ž	¡	Æ	Æ	Í	d2	210	Ò	Ò	Ň	Ň	“	╥	ř	╥	Ê	Ď
13	19	DC3	53	83	S	93	147	□	“	□	“	ď	ô	Ň	ô	ô	ô	d3	211	Ó	Ó	Ó	Ó	”	╙	š	╙	Ë	Ë
14	20	DC4	54	84	T	94	148	□	”	□	”	Ě	ö	¿	ö	ö	ö	d4	212	Ô	Ô	Ô	Ô	‘	╘	ť	╘	È	ď
15	21	NAK	55	85	U	95	149	□	•	□	•	ē	Ó	ñ	ò	ò	Ľ	d5	213	Õ	Õ	Ő	Ő	’	╒	ú	╒	ı	Ň
16	22	SYN	56	86	V	96	150	□	–	□	–	É	ů	▌	û	û	ľ	d6	214	Ö	Ö	Ö	Ö	÷	╓	╞	╓	Í	Í
17	23	ETB	57	87	W	97	151	□	—	□	—	ó	Ú	▄	ù	ù	Ś	d7	215	×	×	×	×	◊	╫	é	╫	Î	Î
18	24	CAN	58	88	X	98	152	□	˜	□		ė	ý	▀	ÿ	ÿ	ś	d8	216	Ø	Ø	Ř	Ř	ō	╪	à	╪	Ï	ě
19	25	EM	59	89	Y	99	153	□	™	□	™	ô	Ö		Ö	Ö	Ö	d9	217	Ù	Ù	Ů	Ů	Ŕ	┘	ý	┘	┘	┘
1a	26	SUB	5a	90	Z	9a	154	□	š	□	š	ö	Ü		Ü	Ü	Ü	da	218	Ú	Ú	Ú	Ú	ŕ	┌	ž	┌	┌	┌
1b	27	ESC	5b	91	[	9b	155	□	›	□	›	õ	Š		¢	ø	Ť	db	219	Û	Û	Ű	Ű	Ř	╗	┬	╗	╗	╗
1c	28	FS	5c	92	\	9c	156	□	œ	□	ś	ú	Ľ		£	£	ť	dc	220	Ü	Ü	Ü	Ü	‹	▄		▄	▄	▄
1d	29	GS	5d	93	]	9d	157	ť		ť	Ť	Ě	Ý		¥	Ø	Ł	dd	221	Ý	Ý	Ý	Ý	›	▌		▌	¦	Ţ
1e	30	RS	5e	94	^	9e	158	ž	ž	ž	ž	ě	Ř		₧	×	×	de	222	Þ	Þ	Ţ	Ţ	ř	▐		▐	Ì	Ů
1f	31	US	5f	95	_	9f	159	□	Ÿ	□	ź	ü	ť		ƒ	ƒ	č	df	223	ß	ß	ß	ß	Ŗ	▀	┼	▀	▀	▀
20	32		60	96	`	a0	160					†	á	≡	á	á	á	e0	224	à	à	ŕ	ŕ	ŗ	α	╕	α	Ó	Ó
21	33	!	61	97	a	a1	161	¡	¡	Ą	ˇ	°	í	±	í	í	í	e1	225	á	á	á	á	Š	β	Á	ß	ß	ß
22	34	"	62	98	b	a2	162	¢	¢	˘	˘	Ę	ó	≥	ó	ó	ó	e2	226	â	â	â	â	‚	Γ	╜	Γ	Ô	Ô
23	35	#	63	99	c	a3	163	£	£	Ł	Ł	£	ú	≤	ú	ú	ú	e3	227	ã	ã	ã	ã	„	π	Č	π	Ò	Ń
24	36	$	64	100	d	a4	164	¤	¤	¤	¤	§	ň	⌠	ñ	ñ	Ą	e4	228	ä	ä	ä	ä	š	Σ	Ď	Σ	õ	ń
25	37	%	65	101	e	a5	165	¥	¥	Ľ	Ą	•	Ň	⌡	Ñ	Ñ	ą	e5	229	å	å	Í	Í	Ś	σ	Ě	σ	Õ	ň
26	38	&	66	102	f	a6	166	¦	¦	Ś	¦	¶	Ů	÷	ª	ª	Ž	e6	230	æ	æ	ć	ć	ś	µ	Ŕ	µ	µ	Š
27	39	'	67	103	g	a7	167	§	§	§	§	ß	Ô	≈	º	º	ž	e7	231	ç	ç	ç	ç	Á	τ	╒	τ	þ	š
28	40	(	68	104	h	a8	168	¨	¨	¨	¨	®	š	°	¿	¿	Ę	e8	232	è	è	č	č	Ť	Φ	Ů	Φ	Þ	Ř
29	41	)	69	105	i	a9	169	©	©	Š	©	©	ř	√	⌐	®	ę	e9	233	é	é	é	é	ť	Θ	Í	Θ	Ú	Ú
2a	42	*	6a	106	j	aa	170	ª	ª	Ş	Ş	™	ŕ	²	¬	¬	¬	ea	234	ê	ê	ę	ę	Í	Ω	Ů	Ω	Û	ŕ
2b	43	+	6b	107	k	ab	171	«	«	Ť	«	ę	Ř	½	½	½	ź	eb	235	ë	ë	ě	ě	Ž	δ	Ľ	δ	Ù	Ű
2c	44	,	6c	108	l	ac	172	¬	¬	Ź	¬	¨	¼	¼	¼	¼	Č	ec	236	ì	ì	ě	ě	ž	∞	Ľ	∞	ý	ý
2d	45	-	6d	109	m	ad	173					≠	§	§	¡	¡	ş	ed	237	í	í	í	í	Ū	∅	Ö	φ	Ý	Ý
2e	46	.	6e	110	n	ae	174	®	®	Ž	®	ġ	«	«	«	«	«	ee	238	î	î	î	î	Ó	ε	Ň	ε	¯	ţ
2f	47	/	6f	111	o	af	175	¯	¯	Ż	Ż	ļ	»	»	»	»	»	ef	239	ï	ï	ď	ď	Ô	∩	Ó	∩	´	´
30	48	0	70	112	p	b0	176	°	°	°	°	¡	░	α	╕	╕	╕	f0	240	ð	ð	đ	đ	ū	≡	Ô	≡		
31	49	1	71	113	q	b1	177	±	±	ą	±	Ī	▒	β	╖	╖	╖	f1	241	ñ	ñ	ń	ń	Ů	±	Ä	±	±	˝
32	50	2	72	114	r	b2	178	²	²	˛	˛	≤	▓	Γ	╖	╖	╖	f2	242	ò	ò	ň	ň	Ú	≥	Ř	≥	‗	˛
33	51	3	73	115	s	b3	179	³	³	ł	ł	≥	│	π	│	│	│	f3	243	ó	ó	ó	ó	ů	≤	Š	≤	¾	˘
34	52	4	74	116	t	b4	180	´	´	´	´	ī	┤	Σ	┤	┤	┤	f4	244	ô	ô	ô	ô	Ű	⌠	Ť	⌠	¶	˘
35	53	5	75	117	u	b5	181	µ	µ	ľ	µ	Ķ	╡	σ	╡	Á	Á	f5	245	õ	õ	ő	ő	ũ	⌡	Ú	⌡	§	§
36	54	6	76	118	v	b6	182	¶	¶	ś	¶	∂	╢	µ	╢	Â	Â	f6	246	ö	ö	ö	ö	Ų	÷	┤	÷	÷	÷
37	55	7	77	119	w	b7	183	·	·	ˇ	·	∑	╖	γ	╖	À	Ě	f7	247	÷	÷	÷	÷	ų	≈	É	≈	¸	¸
38	56	8	78	120	x	b8	184	¸	¸	¸	¸	ł	╕	Φ	╕	©	Ş	f8	248	ø	ø	ř	ř	Ý	°	À	°	°	°
39	57	9	79	121	y	b9	185	¹	¹	š	ą	Ļ	╣	Θ	╣	╣	╣	f9	249	ù	ù	ů	ů	ý	·	Ý	·	¨	¨
3a	58	:	7a	122	z	ba	186	º	º	ş	ş	ļ	║	Ω	║	║	║	fa	250	ú	ú	ú	ú	ķ	˙	Ž	•	·	·
3b	59	;	7b	123	{	bb	187	»	»	ť	»	Ľ	╗	δ	╗	╗	╗	fb	251	û	û	ű	ű	Ż	√	┴	√	¹	ű
3c	60	<	7c	124	|	bc	188	¼	¼	ź	Ľ	ľ	╝	ω	╝	╝	╝	fc	252	ü	ü	ü	ü	Ł	ⁿ		ⁿ	³	Ř
3d	61	=	7d	125	}	bd	189	½	½	˝	˝	Ĺ	╜	φ	╜	¢	Ż	fd	253	ý	ý	ý	ý	ż	²		²	²	ř
3e	62	>	7e	126	~	be	190	¾	¾	ž	ľ	Í	╛	ε	╛	¥	ż	fe	254	þ	þ	ţ	ţ	Ģ	■		■	■	■
3f	63	?	7f	127	DEL	bf	191	¿	¿	ż	ż	Ņ	┐	η	┐	┐	┐	ff	255	ÿ	ÿ	˙	˙	˘					

Při přípravě grafického rozhraní operačního systému Windows se upustilo od znakové sady používané v systému MS-DOS. Byla použita znaková sada ISO 8859-1, která byla mírně modifikována. Vzniklo tak zcela nesystémově devět nových variant kódování (tab. 4.14).

Tabulka 4.14: *Varianty znakového kódu Windows*

Kód	Popis
Windows-1250	jazyky střední a východní Evropy
Windows-1251	jazyky používající cyrilici
Windows-1252	jazyky západní Evropy
Windows-1253	řečtina
Windows-1254	turečtina

Kód	Popis
Windows-1255	hebrejština
Windows-1256	arabština
Windows-1257	baltské jazyky
Windows-1258	vietnamština

Znaková sada *Windows-1250* (též CP1250) podporuje středoevropské jazyky a němčinu a velmi se podobá standardu ISO 8859-2. Mezi Windows-1250 a ISO 8859-2 najdeme pouze čtrnáct rozdílů, u češtiny a slovenštiny dokonce pouze šest: znaky ž, š, ť, Ž, Š a Ť. Obecně však neexistuje jednoznačná konverze mezi znakovými sadami ISO 8859 a Windows. *Windows 1250*

Znaková sada *Macintosh Central European* (též MacCE, Apple-CE) je určena pro operační systém macOS. Od ostatních kódování se liší, vznikla pod dohledem firmy Apple a její struktura je přizpůsobena firemním standardům. *MacCE*

Příklad 4.23

Reprezentace řetězce „Příliš žlutý kůň!“ v různých variantách rozšířeného znakového kódu ASCII (číselné kódy jednotlivých znaků jsou uvedeny v šestnáctkové soustavě):

Kameničtí:	50	A9	A1	6C	69	A8	20	91	6C	75	74	98	20	6B	96	A4	21
PC Latin 2:	50	FD	A1	6C	69	E7	20	A7	6C	75	74	EC	20	6B	85	E5	21
KOI-8 ČS2:	50	D2	C9	6C	69	D3	20	DA	6C	75	74	D9	20	6B	CA	CE	21
ISO 8859-2:	50	F8	ED	6C	69	B9	20	B9	6C	75	74	FD	20	6B	F9	F2	21
Windows-1250:	50	F8	ED	6C	69	9A	20	9E	6C	75	74	FD	20	6B	F9	F2	21
MacCE:	50	DE	92	6C	69	E4	20	EC	6C	75	74	F9	20	6B	F3	CB	21
	P	ř	í	l	i	š	␣	ž	l	u	t	ý	␣	k	ů	ň	!

Jak je vidět z příkladu, rozdíly mezi jednotlivými variantami jsou pouze v umístění národních znaků, ostatní znaky patří do základní části tabulky, a proto budou při použití libovolné varianty rozšířeného kódu ASCII vždy zobrazeny stejně.

4.2.2 Vícebajtové znakové kódy

Na konci 80. let 20. století se jednobajtové kódy dostaly do krize. Podporovaly sice téměř libovolný jazyk, ale nebylo možné v jednom dokumentu použít více jazyků současně (například češtinu a francouzštinu).

Vznikla proto naléhavá potřeba *sjednotit* různé znakové kódy pro národní abecedy, protože tehdejší chaotický stav způsoboval značné problémy při spolupráci aplikací a přenosech dat mezi programy a různými platformami. Ve stejné době začal také proces komputerizace „exotických“ jazyků zejména z jihovýchodní Asie, jejichž národní znaky by se ani do tabulky nevešly.

V rámci hledání nových možností proto vznikly hned dva souběžné projekty: Unicode a UCS. Přibližně v roce 1991 projekty spojily své úsilí na vytvoření jednotné kódové tabulky pro všechny světové jazyky. Oba projekty existují a publikují standardy samostatně, ale kódové tabulky jsou kompatibilní a jejich rozšiřování je koordinováno.

BMP

Byla popsána množina nejpoužívanějších znaků ze všech světových jazyků, která obsahuje 2^{16} prvků. Tato množina představuje základní rovinu (*Basic Multilingual Plane*) a je podporována všemi variantami vícebajtových kódů (viz tab. 4.15, str. 74).

UCS

Standard **UCS** (*Universal Character Set*, ISO 10646) definuje univerzální znakovou sadu, která obsahuje kompletní soubor znaků pro reprezentaci všech známých jazyků. Obsahuje též grafické, typografické, matematické a vědecké symboly. Kódovací prostor 31 bitů umožňuje rozlišit 2 147 483 648 znaků rozdělených do 128 skupin po 24 bitech. Každá skupina je pak rozdělena na 256 rovin (*planes*) po 16 bitech a každá rovina na 256 řad s 256 sloupci. Většina používaných znaků je přitom umístěna v první části první skupiny (tj. v BMP na 16 bitech). Znaky přidané mimo BMP mají obvykle velmi specifický účel. Doporučuje se proto využívat pouze 21 bitů kódu, což i tak nabízí dostatečný prostor pro více než milion znaků.

Unicode

Standard **Unicode** poskytuje prostor pro reprezentaci právě těch 1 114 112 znaků, které lze zobrazit na 21 bitech znakové sady ISO 10646. Tento prostor je rozdělen na 17 částí, každá o velikosti 2^{16}. První část odpovídá BMP podle standardu ISO 10646. Původní návrh Unicode počítal pouze s BMP, ale postupem času se ukázalo, že pro pokrytí všech používaných abeced to nemůže stačit.

Oba standardy obsahují stejné znaky na stejných pozicích, i když Unicode vydává nové verze častěji. Každá nová verze obohacuje tabulku o nové znaky, ale nikdy nemění již umístěné znaky. Zatímco standard ISO 10646 je prakticky „jen“ rozšířením standardu ISO 8859, standard Unicode se zabývá také implementací algoritmů pro jazyky psané zprava doleva (např. arabština), podporou obousměrně sázených textů (např. směs hebrejštiny a latinky), algoritmy pro řazení a porovnávání textů apod.

Jména znaků

Každému znaku je přiřazen jednoznačný *číselný kód* a *oficiální jméno*, např. znak „A“ má kód `U+0041` (šestnáctkově) a jméno „Latin capital letter A“. Kódy znaků jsou částečně zpětně kompatibilní, neboť znaky `U+0000` až `U+007F` (prvních 128 znaků) jsou shodné se znaky v základní části znakové sady ASCII, znaky `U+0080` až `U+00FF` (dalších 128 znaků) odpovídají standardizované znakové sadě ISO 8859-1.

Kombinované znaky

Standard ISO 10646 dále definuje množinu tzv. *kombinovaných znaků*, které mají význam pouze v kombinaci s jiným znakem a společně tak utvářejí určitý symbol. Příkladem mohou být i česká diakritická znaménka. Část kombinovaných znaků má svůj samostatný kód, který vznikl historicky při kódování národních abeced. Pro účely matematických a dalších symbolů je možné vytvářet další kombinace, přidávat akcenty a diakritiku nad libovolný znak (např. šipku nad matematickou proměnnou). V souvislosti s podporou kombinovaných znaků existují tři implementační úrovně (*levels*) standardu ISO 10646:

- úroveň 1 – nejsou podporovány žádné kombinované znaky a také znaky *hangŭl*[24];
- úroveň 2 – jsou podporovány pouze pevně definované kombinované znaky;
- úroveň 3 – jsou podporovány libovolné varianty kombinovaných znaků.

[24] *Hangŭl* (한글) je jihokorejský termín pro „moderní“ variantu korejského písma z 15. stol., které nahradilo do té doby používané čínské znaky. V Severní Koreji je totéž písmo nazýváno *chosŏn'gŭl* (조선글).

Ne vždy potřebujeme využít celý kódovací prostor, většinou stačí jen několik málo znaků. Rozlišujeme proto několik různých metod, kterými lze znaky kódovat. Některé jsou definovány přímo ve standardu Unicode, zejména jde o kódy UCS-2 a UCS-4 a také kódy **UTF** (*UCS Transformation Format*), z nichž nejvýznamnější jsou UTF-8, UTF-16 a UTF-32. *Metody kódování*

Kód *UCS-2* představuje původní způsob zápisu znaků z projektu Unicode. Nabízí kódovací prostor 16 bitů, na kterém lze zobrazit všechny znaky z BMP, znaky mimo BMP reprezentovat nelze. Kód *UTF-16* rozšiřuje UCS-2 o podporu znaků mimo BMP. Teoretický rozsah je `U+00000000` až `U+0010FFFF`, znaky z intervalu `U+00000000` až `U+0000FFFF` (BMP) jsou reprezentovány na 2 bajtech (totožně v obou kódech), znaky z intervalu `U+00010000` až `U+0010FFFF` (mimo BMP) jsou v UTF-16 reprezentovány na 4 bajtech. *UCS-2, UTF-16*

Kód *UCS-4* zaznamenává každý znak na 4 bajtech, což umožňuje podporu všech znaků z UCS-2 včetně těch mimo BMP. Znaky z UCS-2 se zobrazují beze změn, ale na dvojnásobném prostoru, což je zbytečně neefektivní. Kódy *UTF-32* a UCS-4 jsou ekvivalentní. Teoretický rozsah zobrazitelných hodnot je `U+00000000` až `U+7FFFFFFF`, ale prakticky stačí pouze rozsah `U+00000000` až `U+0010FFFF`, tedy oněch doporučených 21 bitů. Výhodou tohoto kódu je stejná reprezentace všech znaků, nevýhodou je příliš velká neúspornost. *UCS-4, UTF-32*

Všechny zmíněné kódy (tj. UTF-16 i UTF-32) se mohou objevit ve dvou variantách, které se liší použitým pořadím bajtů. Ve variantě *Little Endian* se do paměti ukládá nejdříve nejméně významný bajt (LSB), za něj ostatní bajty až po nejvíce významný bajt (MSB). U varianty *Big Endian* je tomu naopak. Little Endian má výhodu v tom, že jedna a ta samá hodnota může být z paměti načtena v různé délce bez změny adresy, protože je jen doplněna nulami na požadovanou délku. Programátoři, kteří pracují s vyššími programovacími jazyky, však tuto vlastnost využívají velmi zřídka, proto se ponechává kompilátoru. *Endianita*

Příklad 4.24

Znak `U+4A3B2C1D` by byl na prostoru 4 bajtů uložen následovně:

Little Endian:

1D	2C	3B	4A

Big Endian:

4A	3B	2C	1D

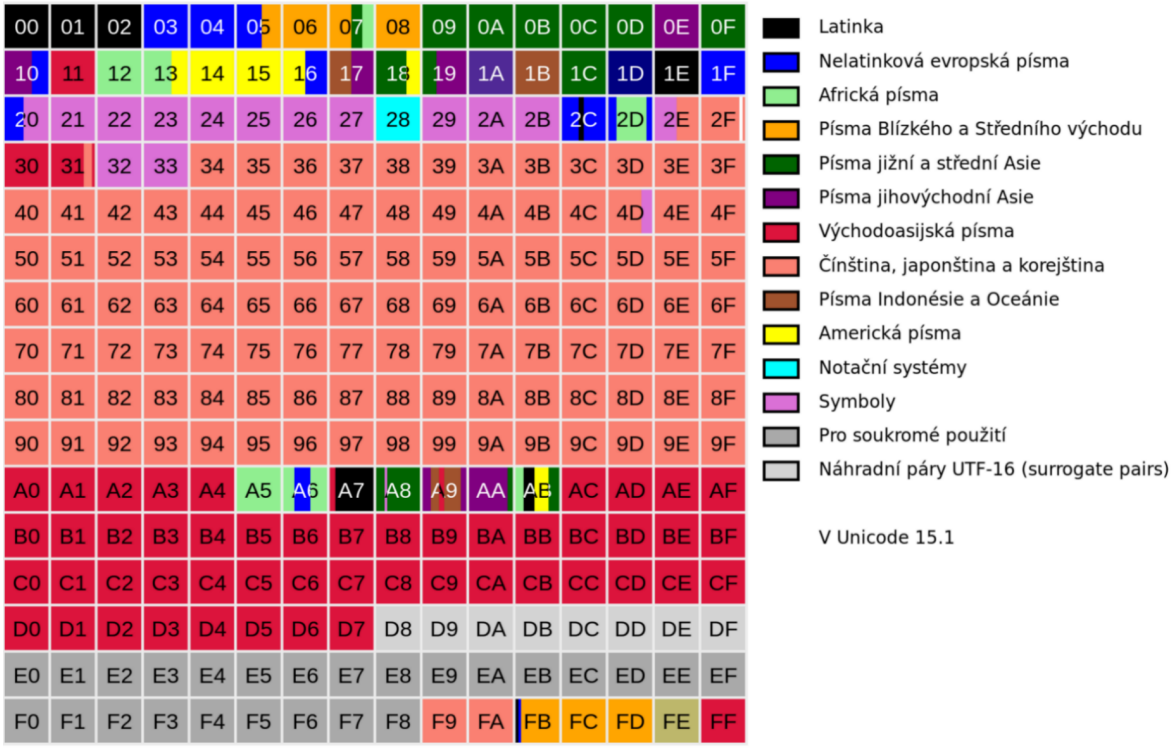
Kód *UTF-8* odstraňuje nevýhody kódu Unicode, protože je úsporný, zpětně kompatibilní s kódem ASCII a nemá problémy s endianitou. V současnosti se jedná o nejpoužívanější formu UCS kódování. Má podporu ve všech webových prohlížečích, doporučuje se pro tvůrce poštovních klientů a je také standardním kódováním v operačních systémech, programovacích jazycích a mnoha dalších softwarových aplikacích. Kód reprezentuje 1 112 064 znaků standardu Unicode, délka reprezentace znaků se pohybuje od 1 do 6 bajtů[25]: *UTF-8*

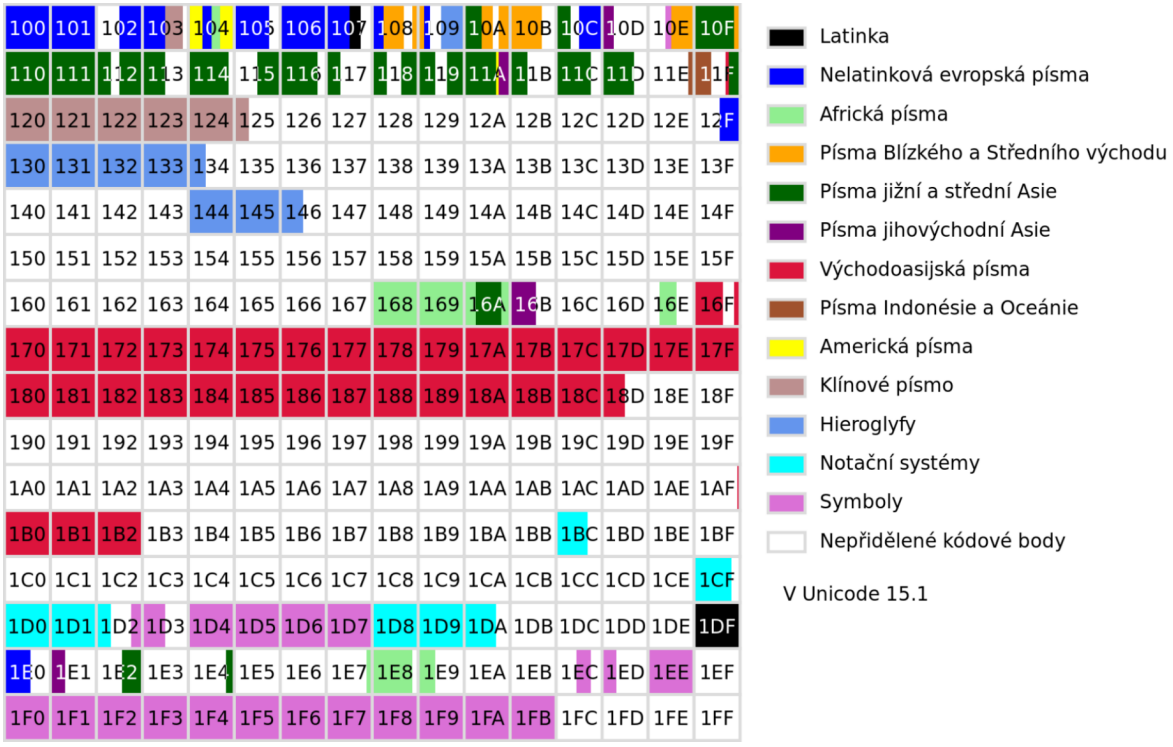
- znaky `U+0000` až `U+007F` (128 znaků ze základní části ASCII) jsou kódovány na 1 bajtu v podobě binární posloupnosti 0xxxxxxx;
- znaky `U+0080` až `U+07FF` (1 920 běžných národních znaků) jsou kódovány na 2 bajtech v podobě binární posloupnosti[26] 110xxxxx 10xxxxxx;
- znaky `U+0800` až `U+FFFF` (zbytek BMP, např. € = `U+20AC`) jsou kódovány na 3 bajtech v podobě binární posloupnosti 1110xxxx 10xxxxxx 10xxxxxx;
- znaky `U+010000` až `U+10FFFF` (ostatní znaky mimo BMP) jsou kódovány na 4 bajtech v podobě binární posloupnosti 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx.

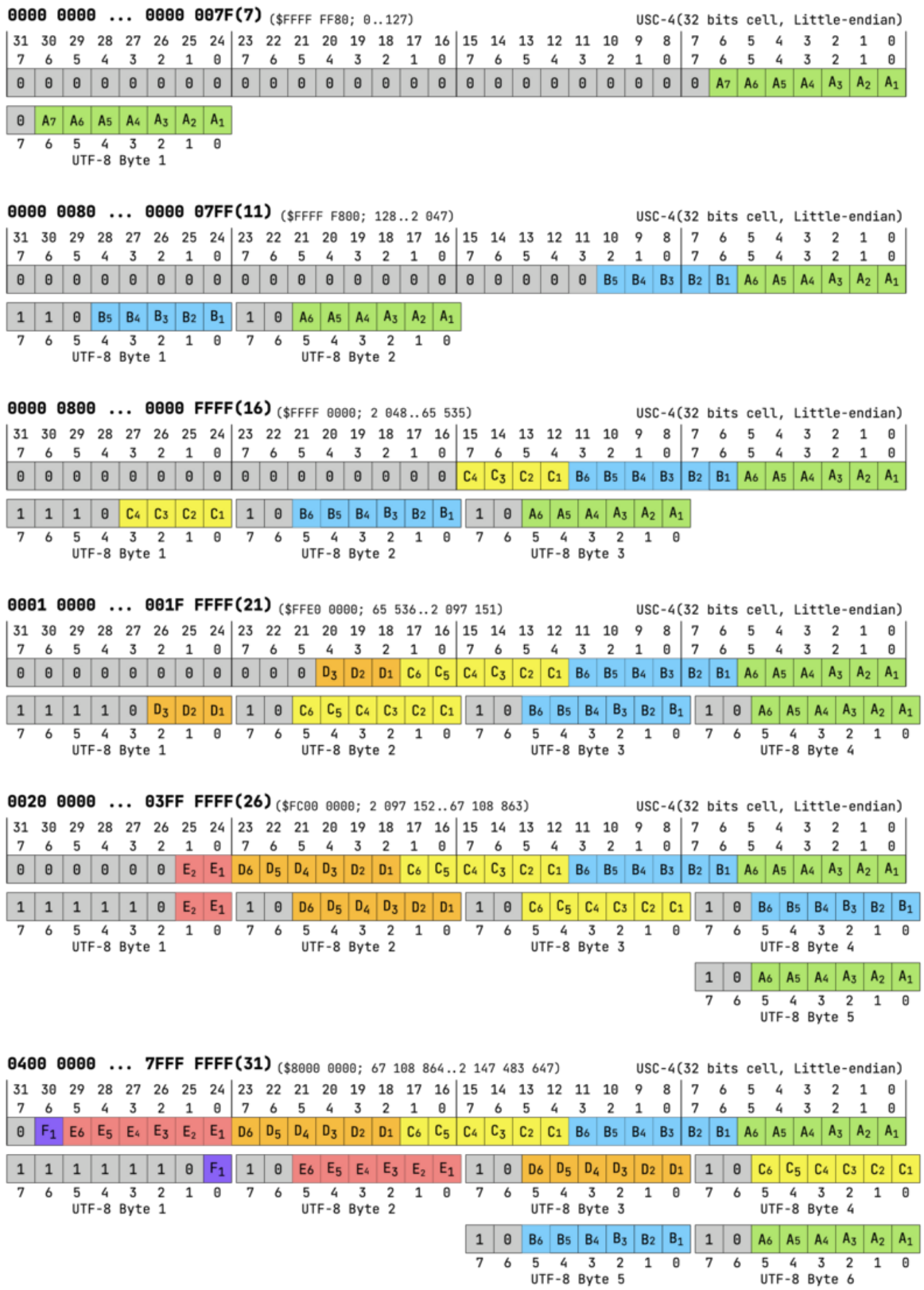
[25] Znaky kódované na 5 a 6 bajtech zatím nebyly vůbec definovány.
[26] Povšimněme si souvislosti mezi počtem binárních jedniček na začátku posloupnosti a počtem bajtů.

Tabulka 4.15: *Aktuální verze standardu Unicode 15.1 (září 2023), přehled jednotlivých rovin a podrobný obsah rovin BMP a SMP (každá buňka zahrnuje 256 kódových pozic pro jednotlivé znaky)*

Rovina 0	Rovina 1	Rovina 2	Rovina 3	Roviny 4–13	Rovina 14	Roviny 15–16
0000–FFFF	10000–1FFFF	20000–2FFFF	30000–3FFFF	40000–DFFFF	E0000–EFFFF	F0000–10FFFF
Basic Multilingual Plane (BMP)	Supplementary Multilingual Plane (SMP)	Supplementary Ideographic Plane (SIP)	Tertiary Ideographic Plane (TIP)		Supplementary Special-purpose Plane (SSP)	Supplementary Private Use Area planes (SPUA-A/B)







Obrázek 4.5: *Podrobné schéma kódování standardu UTF-8*

BOM

Na začátku každého souboru ve vícebajtovém kódu by měla být umístěna specifická značka, tzv. **BOM** (*Byte Order Mark*), která označuje endianitu (pořadí bajtů v souboru). Pro kód UTF-8 není nutná, ale usnadňuje identifikaci znakové sady. Podobu této značky uvádí tab. 4.16.

Tabulka 4.16: *Přehled BOM pro vícebajtové znakové kódy*

Znakový kód	Podoba BOM
UTF-8	EF BB BF
UTF-16 BE	FE FF
UTF-16 LE	FF FE
UTF-32 BE	00 00 FE FF
UTF-32 LE	FF FE 00 00

Příklad 4.25

Reprezentace řetězce „Béďa“ v různých kódováních včetně identifikace endianity pomocí BOM (číselné kódy jednotlivých znaků uvádíme v šestnáctkové soustavě):

ASCII / ISO 8859-2:				42		E9		EF		61	(4 B)
UTF-16 BE:		FE	FF	00	42	00	E9	01	0F	00	61 (10 B)
UTF-16 LE:		FF	FE	42	00	E9	00	0F	01	61	00 (10 B)
UTF-8:	EF	BB	BF		42	C3	A9	C4	8F		61 (9 B)
					B		é		ď		a

Na základě délek reprezentace z předchozího příkladu by se mohlo na první pohled zdát, že kódování UTF-8 nepřináší žádnou podstatnou výhodu. Je však potřeba si uvědomit, že z 9 B v kódu UTF-8 se 3 B používají k reprezentaci BOM. Vzhledem k tomu, že délka BOM je konstantní a že běžný český text obsahuje méně než 30 % dvoubajtových znaků, je zřejmé, že při dostatečném počtu znaků se kód UTF-8 stává jednoznačně nejúspornější *vícebajtovou* variantou pro reprezentaci textů s národními znaky.

Kontrolní otázky

18. Jakým způsobem jsou znaky reprezentovány v paměti počítače?
19. Jaký je rozdíl mezi řídicím a zobrazitelným znakem?
20. Z kterých částí se skládá tabulka znakového kódu ASCII?
21. Co je to národní znak a jaké problémy přináší jeho reprezentace?
22. Které jednobajtové znakové sady jsou použitelné pro češtinu a slovenštinu?
23. Proč se dnes jednobajtové znakové kódy nahrazují vícebajtovými a který vícebajtový kód je optimální?
24. Jaký je při reprezentaci vícebajtových znakových kódů v počítači rozdíl mezi variantou Little Endian a Big Endian?

4.3 Reprezentace grafických dat

Zpracování grafické informace představuje velmi širokou oblast použití počítačových technologií. Komunikace prostřednictvím obrazu velmi usnadňuje a zpřehledňuje přenos informací, zjednodušuje orientaci a rozšiřuje možnosti ovládání stroje.

Základním principem, od něhož se odvíjejí veškeré možnosti zpracování, je zakódování grafické informace pomocí čísel ukládaných do paměti počítače, abychom ji následně mohli podle potřeby zpracovat a zobrazit. Obrázky takto ukládané v počítači můžeme rozdělit do dvou zásadně rozdílných skupin – na *rastrové* a *vektorové*.

4.3.1 Barvy a barevné modely

Světlo, UV, IR

Viditelné světlo je jednou z mnoha forem elektromagnetického záření. Lidské oko vnímá takové záření jen v úzkém frekvenčním pásmu přibližně od $3{,}9 \cdot 10^8$ do $7{,}9 \cdot 10^8$ MHz. Nejnižším frekvencím z tohoto pásma odpovídá červená barva, nejvyšším odpovídá fialová barva. Pásmo viditelného světla sousedí s pásmem ultrafialového záření (*UltraViolet*, v kratších vlnových délkách) a pásmem infračerveného záření (*InfraRed*, v delších vlnových délkách). Přestože infračervené světlo není vidět, je možné jej vnímat jako teplo prostřednictvím receptorů v pokožce. Ultrafialové záření se zase projevuje zvýšením pigmentace pokožky, tedy opálením.

Barva

To, co my vnímáme jako *barvu*, tedy ve skutečnosti není nic jiného než elektromagnetické záření na určité frekvenci. Lidské oko je schopno vnímat jen asi $4 \cdot 10^5$ různých odstínů barev a nejcitlivější je na barevné přechody. Chceme-li dosáhnout interpretace některé barvy z počítače, musíme technicky vyřešit vytvoření příslušného odstínu smícháním vybraných základních barev a informaci o tom uložit v počítači.

Barevné modely

Ke specifikaci jednotlivých barev používáme **barevný model**. Každý barevný model je jednoznačně popsán těmito vlastnostmi:

- základní barvy, z nichž se budou skládat všechny odstíny;
- poměry, v jakých bude možné základní barvy skládat;
- způsob míchání základních odstínů.

RGB

Aditivní model RGB je jedním z nejčastěji používaných modelů. Základními barvami jsou červená (*Red*), zelená (*Green*) a modrá (*Blue*). Tyto základní barvy se míchají aditivním způsobem tak, že se přidávají k černé[27] barvě. Čím větší intenzitu použijeme, tím světlejší odstín dostáváme. Každá ze složek může nabývat 256 intenzit v rozmezí od 0 do 255. Smícháme-li všechny tři složky v maximální intenzitě, obdržíme čistou bílou barvu. Mají-li všechny složky naopak intenzitu nulovou, dostáváme čistou černou barvu.

Smícháním dvou základních odstínů dostáváme odvozené odstíny (červená a zelená dávají žlutou, červená a modrá dávají purpurovou, zelená a modrá dávají azurovou). Smícháním všech tří složek o stejné intenzitě dostáváme šedou barvu, jejíž intenzita odpovídá intenzitám použitých složek (čím vyšší číslo, tím blíže je odstín k bílé[28] barvě).

Barevný model RGB se nejčastěji používá pro interpretaci barev na monitoru či projektoru. Nepotřebuje vnější světlo, protože barvy a jejich složky jsou chápány jako svítící body. Uložení informace o barvě v počítači tak představuje uložení tří čísel reprezentujících

[27] Vnímá-li oko barvu jako černou, znamená to, že z daného směru nepřichází žádné světlo.
[28] Bílá, černá a šedá jsou neutrální (achromatické) barvy, které se liší pouze úrovní jasu a sytosti (viz str. 80).

intenzity jednotlivých barevných složek daného odstínu. Tímto způsobem můžeme rozlišit až 256^3 = 16 777 216 odstínů.

RGBA Můžeme se setkat i s variantou modelu RGB, která uvažuje navíc ještě intenzitu průhlednosti odstínu vzniklého smícháním základních složek, tzv. *alfa kanál*. Tento upravený model se označuje **RGBA**.

CMY **Subtraktivní model CMY** používá jako základní barvy azurovou (*Cyan*), purpurovou (*Magenta*) a žlutou (*Yellow*). Tyto barvy se míchají subtraktivním způsobem, tedy odečítají se od bílé. Čím větší intenzitu použijeme, tím tmavší odstín dostáváme. Každá ze složek může (stejně jako v RGB) nabývat 256 intenzit v intervalu od 0 po 255. Smícháme-li všechny tři složky v maximální intenzitě, teoreticky obdržíme čistou černou barvu. Pokud však všechny složky mají intenzitu nulovou, dostáváme čistou bílou barvu.

Smícháním dvou základních odstínů dostáváme odvozené odstíny (azurová a purpurová dávají modrou, azurová a žlutá zelenou, purpurová a žlutá červenou). Z uvedeného vidíme, že modely RGB a CMY jsou navzájem *doplňkové*. Odvozená barva v jednom modelu je základní barvou v druhém modelu. Dualita obou modelů je patrná z tab. 4.17.

CMYK Zatímco barevný model RGB lze v praxi poměrně dobře realizovat zařízením s vlastním světlem, model CMY, který se používá pro tisková zařízení, potřebuje vnější světlo. Reprezentuje totiž míchání barev na papíře, tímto způsobem však v praxi nedostáváme přesně tytéž odstíny, jaké jsou v teoretickém modelu předepsány. Proto se v reálných zařízeních používá nikoliv trojice, ale čtveřice základních barev. Ke třem základním složkám modelu CMY se přidává ještě složka černá (*Key*)[29], hodnoty se vyjádří relativně a tento doplněný barevný model je pak znám pod zkratkou **CMYK**. Černá složka tedy umožňuje správný tisk šedých a černých bodů, kterých není možné dosáhnout mícháním barevných složek.[30]

Tabulka 4.17: *Příklady barev v modelech RGB, CMY a CMYK a schéma převodu mezi RGB a CMYK*

Barva	**Model RGB**			**Model CMY**			**Model CMYK**			
červená	255	0	0	0	255	255	0	1	1	0
zelená	0	255	0	255	0	255	1	0	1	0
modrá	0	0	255	255	255	0	1	1	0	0
azurová	0	255	255	255	0	0	1	0	0	0
purpurová	255	0	255	0	255	0	0	1	0	0
žlutá	255	255	0	0	0	255	0	0	1	0
0% šedá (černá)	0	0	0	255	255	255	0	0	0	1
25% šedá	63	63	63	192	192	192	0	0	0	0,75
50% šedá	127	127	127	127	127	127	0	0	0	0,50
100% šedá (bílá)	255	255	255	0	0	0	0	0	0	0

$$R' = \frac{R}{255}, \quad G' = \frac{G}{255}, \quad B' = \frac{B}{255}$$

$$K = 1 - \max(R', G', B')$$

$$C = \frac{1 - R' - K}{1 - K}$$

$$M = \frac{1 - G' - K}{1 - K}$$

$$Y = \frac{1 - B' - K}{1 - K}$$

$$R = 255 \cdot (1 - C) \cdot (1 - K)$$

$$G = 255 \cdot (1 - M) \cdot (1 - K)$$

$$B = 255 \cdot (1 - Y) \cdot (1 - K)$$

[29] Mnoho zdrojů dává písmeno „K“ v označení modelu CMYK do souvislosti s černou barvou (*blacK*), ale ve skutečnosti tomu tak není. Označení pochází od klíčovacích značek, které se tisknou černě a slouží pro zarovnání barev při soutisku jednotlivých složek. Jako mnemotechnická pomůcka to ale může být užitečné.

[30] Svou roli hrají i ekonomické důvody. I když budeme uvažovat plnobarevný tisk, stejně nejčastější barvou bude černá. Kdybychom chtěli černou barvu získat mícháním, museli bychom použít všechny základní barvy v maximální intenzitě. Tímto způsobem by ale barevné náplně do tiskárny příliš dlouho nevydržely. Vztahy pro převod mezi RGB a CMYK uvedené v tab. 4.17 jsou však pouze teoretické. Využití čtvrté barvy dělá převod nejednoznačný a lze tedy převést RGB na CMYK mnoha různými způsoby. V praxi se potom volí způsob, který nejlépe vyhovuje konkrétnímu tiskovému zařízení.

4.3.2 Rastrové obrazy

Rastrový obraz můžeme získat nejen kreslením v nějakém programu, ale především skenováním nebo snímáním digitálním fotoaparátem či kamerou. Nejmenší jednotkou rastrového obrazu je elementární obrazový bod nazvaný **pixel** (*picture element*). Posloupnost informací o jednotlivých bodech obrazu bývá často označována pojmem *bitová mapa*, nebo zkráceně *bitmapa*, což pochází z nejjednodušší podoby rastrových dat, kdy jednomu pixelu odpovídá jeden bit paměti. Na obr. 4.6 je znázorněn rastrový obrázek zvětšený tak, aby v něm bylo možné identifikovat jednotlivé pixely. *Pixel*

Obrázek 4.6: *Rastrový obraz a jeho jednotlivé pixely*

Rastrové obrazy jsou podstatně rozšířenější než vektorové, protože většina počítačových zařízení snímá nebo zobrazuje jednotlivé body do kreslicí plochy. S rastrovými obrazy se tak setkáme u všech monitorů a také u většiny tiskáren. U každého rastrového obrazu rozpoznáváme dva základní parametry, a to hustotu obrazu a jeho barevnou hloubku.

Hustota obrazu (též rozlišení, *resolution*) je definována jako počet elementárních bodů na délkovou jednotku. Protože většina počítačových zařízení pochází ze zemí, kde se délkové jednotky vyjadřují v palcích[31], vyjadřuje se hustota obrazu nejčastěji počtem pixelů na palec v jednotkách dpi (*dots per inch*). *Hustota*

Je zřejmé, že čím větší hustotu obrázek má, tím se zdá být kvalitnějším. Monitory pracují s hustotami jen okolo 100 dpi, běžné laserové tiskárny jsou schopny tisknout s hustotou 600 až 1 200 dpi, vysoce kvalitní osvitové jednotky ale mají možnost pracovat s hustotou obrazu kolem 5 000 dpi.

Výstupní zařízení zajišťují kresbu pixelů různě – kapičkou inkoustu, několika zrnky toneru nebo svítícími body. Každý takto zobrazený pixel má nejen svou skutečnou barvu, ale také nenulový rozměr – hovoříme o **fyzickém pixelu**. Naproti tomu informace o pixelu uložená v počítači nese pouze barvu a rozměr se předpokládá nulový – hovoříme o **logickém pixelu**. Na každém zařízení se logický pixel musí zobrazit pomocí fyzického pixelu. V tom ovšem může hrát roli řada faktorů, získaný výsledek je pak dán možnostmi příslušného zařízení a může být odlišný od našeho očekávání. V této souvislosti proto uvažujeme **efektivní hustotu**, která vyjadřuje schopnost výstupního zařízení zobrazit body určité kvality. Při velkých rozdílech mezi logickým a fyzickým pixelem můžeme dostat obraz, jehož efektivní hustota bude nízká a kvalita zcela nedostatečná. *Efektivní hustota*

[31] Palec jako délková jednotka se užíval v mnoha různých kulturách. Jeho velikost různě kolísala mezi 23 mm a 27 mm. V tomto případě máme na mysli anglický palec (*inch*), který je od roku 1959 definován přesně jako 25,4 mm. Odvozené jednotky se nazývají stopa (12 palců), yard (3 stopy) a míle (1 760 yardů).

Příklad 4.26

Uvažujme obraz s 256 úrovněmi šedé barvy o hustotě 600 dpi. Při jeho zobrazení na monitoru dostaneme vcelku kvalitní výsledek, tisk na laserové tiskárně o hustotě 600 dpi však dopadne nesrovnatelně hůře. Proč? Laserová tiskárna (podobně jako jiná zařízení) je schopna tisknout pouze úplně černé tiskové body, nikoli odstíny šedé. Efektu šedé barvy dosahuje tím, že prostřídává potištěné a nepotištěné body, které se pak z dostatečné vzdálenosti jeví jako šedé. Různé odstíny šedé jsou dosahovány různě hustými rastry z černých bodů. Jeden pixel předlohy tedy musí být na tiskárně reprezentován rastrem o velikosti například 4 × 4 tiskové body. Efektivní hustota tisku potom nebude 600 dpi, ale pouze 150 dpi, což je pochopitelně nedostačující.

Barevná hloubka

Barevná hloubka (*color depth*) představuje množství informace o každém pixelu. Podle počtu bitů pro vyjádření barvy každého bodu rozlišujeme několik typů rastrových obrazů:

- *Monochromatický obraz (monochrome)* – 1 bit na pixel. U tohoto typu obrazu se pouze uchovává informace o tom, zda každý bod má či nemá barvu. Barva je pouze jedna, ale nemusí to být nutně černá. V oblasti zpracování a zejména tisku obrazů má tento typ obrazu zásadní význam. Monochromatický obraz tiskne každá černobílá laserová tiskárna, z monochromatických separací se tisknou i barevné obrazy.
- *Obraz s barevnou paletou (indexed color)* – 2, 4, 8 nebo 16 bitů na pixel. U tohoto typu obrazu nabývá každý pixel určitého počtu různých barev vybraných z plné barevné škály – 4, 16, 256 nebo 65 536 barev.
- *Obraz s odstíny šedi (gray scale)* – 8 bitů na pixel. Každý pixel nabývá jednoho z 256 odstínů šedi od černé až po bílou. Těmito obrazy se reprezentují černobílé fotografie nebo schémata v publikacích tištěných jednou barvou, nejčastěji černou. Počet odstínů je dostatečný pro to, aby obraz černobílé fotografie v počítači a především po vytištění byl pro lidské oči nerozeznatelně podobný originálu.
- *Obraz s pravými barvami (true color)* – 24 bitů na pixel. Barva každého bodu je složena ze základních barev tak, jak je definováno v modelu RGB. Tyto obrazy jsou určeny pro reprezentaci barevných fotografií. Vzhledem k počtu barevných odstínů, které vnímají lidské oči, jsou miliony dostupných barev dostatečné k tomu, aby byl obraz nerozeznatelný od originálu. Zde ovšem hraje podstatnou roli správné nastavení výstupních zařízení, aby barvy uložené v počítači byly správně a přesně interpretovány. Obrazy s hloubkou 32 bitů na pixel (*super true color*) odpovídají modelu RGBA nebo CMYK.

Paměťová náročnost

Ze zadaného rozměru obrazu, jeho typu a hustoty lze vypočítat velikost paměti, kterou obraz zabere.[32] Uvažujme celkový počet pixelů p, skutečné rozměry obrazu v palcích r_x a r_y, hustotu v dpi d_x a d_y a barevnou hloubku v bitech h. Velikost nekomprimovaného rastrového obrazu (v bitech) pak můžeme spočítat ze vztahu

$$V = p \cdot h = r_x \cdot d_x \cdot r_y \cdot d_y \cdot h.$$

Protože často platí $d_x = d_y = d$ a navíc chceme výsledek v bajtech, můžeme psát přímo

$$V = p \cdot {}^{h}\!/_{8} = r_x \cdot r_y \cdot d^2 \cdot {}^{h}\!/_{8}.$$

[32] Jedná se o maximální velikost obrazu v nekomprimované podobě. Velikost komprimovaného obrazu závisí na motivu a způsobu komprese. O této problematice bude pojednáno později (viz kap. 6.3, str. 152).

Příklad 4.27

Nekomprimovaný rastrový obraz v pravých barvách o rozměrech 1 600 × 1 200 pixelů potřebuje k uložení barvy každého bodu 24 bitů, tj. 3 bajty. Dosadíme do vztahu:

$$V = 1\,600 \cdot 1\,200 \cdot 3\ \mathrm{B} = 5\,760\,000\ \mathrm{B} \doteq 5{,}49\ \mathrm{MiB}$$

Příklad 4.28

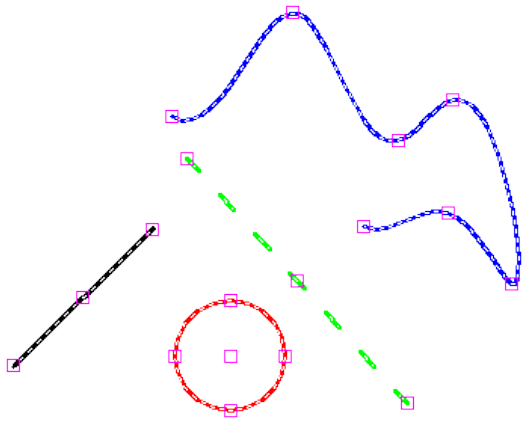
Černobílá fotografie o rozměrech 13 × 9 cm naskenovaná v odstínech šedi s hustotou 300 dpi potřebuje k uložení barvy každého bodu 8 bitů, tj. 1 bajt. Převedeme rozměry fotografie na palce, dostaneme 5,12 × 3,54 palců a dosadíme:

$$V = 5{,}12 \cdot 3{,}54 \cdot 300^2 \cdot 1\ \mathrm{B} = 1\,631\,232\ \mathrm{B} \doteq 1{,}56\ \mathrm{MiB}$$

4.3.3 Vektorové obrazy

Obrazový objekt

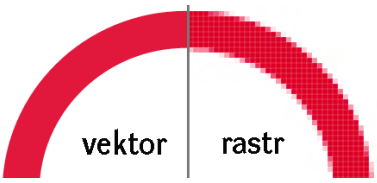
Vektorový obraz je v mnoha směrech doplňkem obrazu rastrového. Základním elementem obrazu není pixel, ale **obrazový prvek** (dříve vektor), který reprezentují geometrické objekty (body, úsečky, křivky, kružnice, mnohoúhelníky, texty apod., viz obr. 4.7). Obrazové objekty jsou popsány svými *atributy*, které udávají jejich velikost a umístění v kreslicí ploše. Číselné hodnoty rozměrů a souřadnic představují relativní hodnoty, jejichž fyzická velikost závisí na globálním parametru – *měřítku*.



Obrázek 4.7: *Vektorový obraz s vyznačenými referenčními body jednotlivých objektů*

Kromě měřítka lze hromadně nastavit například typ a barvu kreslicí čáry, způsob zakončení otevřených čar, druh a barvu výplně uzavřených tahů, požadované písmo textových objektů apod. Popis vektorového obrazu může být realizován mnoha způsoby, například příkaz `\put(1,5){\line(1,0){50}}` v typografickém systému TEX vykreslí vodorovnou čáru o délce 50 jednotek s počátkem na souřadnicích [1; 5]. Podle nastavených parametrů a hodnot jednotlivých objektů pak probíhá základní operace nad vektorovým obrazem, tj. jeho vykreslení v rastrové podobě, tzv. *rastrování*.

Vlastnosti vektorových obrazů je přímo předurčují k použití pro tvorbu ilustrací, diagramů, schémat, grafů apod. Grafy pocházející z tabulkových procesorů jsou obvykle také vektorovými obrazy. Výjimku tvoří Excel, produkt firmy Microsoft, která si s dodržováním standardů nikdy příliš hlavu nelámala. Výhodou vektorových obrazů je přitom je možnost libovolné úpravy obrazu bez ztráty kvality (viz obr. 4.8), práce s každým objektem v obrázku odděleně a také to, že výsledná paměťová náročnost vektorového obrazu je obvykle mnohem menší než u rastrového obrazu. Nevýhodou je zpravidla obtížnější pořízení kvalitního obrazu a u složitějších obrazů též vyšší nároky na paměť a procesor. Vektorová grafika také není vhodná pro práci s velkým množstvím barevných ploch, které najdeme například na barevných fotografiích.



Obrázek 4.8: *Srovnání kvality vektorového a rastrového obrazu při změně měřítka*

Kontrolní otázky

25. K čemu slouží barevné modely a které vlastnosti je popisují?
26. Které barvy tvoří základ modelů RGB a CMYK?
27. Čím se liší rastrový a vektorový obraz?
28. Které parametry rastrového obrazu mají vliv na jeho velikost v paměti?
29. Co vyjadřuje efektivní hustota rastrového obrazu?
30. Co je to bitová hloubka rastrového obrazu?
31. Jak velká je informace o barvě každého pixelu v monochromatickém obrazu a jak velká je v obrazu s pravými barvami?
32. Které atributy můžeme nastavovat objektům ve vektorovém obrazu?
33. Pro které typy obrazů je vhodná vektorová grafika a pro které vhodná není?

4.4 Reprezentace zvukových dat

Infrazvuk, ultrazvuk

Zvuk je mechanické vlnění v látkovém prostředí, které je schopno vyvolat sluchový vjem. Jeho frekvence, které je člověk schopen vnímat, jsou značně individuální a leží v intervalu přibližně od 16 Hz do 20 kHz. Mechanické vlnění mimo tento frekvenční rozsah sluchový vjem nevyvolává. Frekvenci nižší než 16 Hz má *infrazvuk*, který slyší například sloni. Frekvence vyšší než 20 kHz – *ultrazvuk* – mohou vnímat psi, delfíni či netopýři.

V počítači se zvuk reprezentuje výhradně v digitální podobě, je proto nutné analogový signál převést na digitální. Proces hledání diskrétní reprezentace spojitého signálu nazýváme **digitalizace**. Systémy digitálního zvuku zahrnují analogově-digitální převodník (ADC), digitálně-analogový převodník (DAC), digitální paměť, procesní a přenosové složky. Zásadní výhodou digitálního zvuku je jeho snadná uložitelnost, přenos a vyhledání. Digitální reprezentace navíc umožňuje přidání metadat, komprese a dalšího zpracování bez zkreslení.

Digitalizace zvuku

4.4.1 Vzorkování

Princip digitalizace analogového zvuku spočívá v odečítání hodnoty signálu a jejím záznamu v binární podobě. Odebírání vzorku z přijímaného signálu, tzv. **vzorkování**, se provádí v přesně definovaných pravidelných časových intervalech. Určujícími parametry digitálního zvuku jsou vzorkovací frekvence a jemnost rozlišení jednotlivých hodnot.

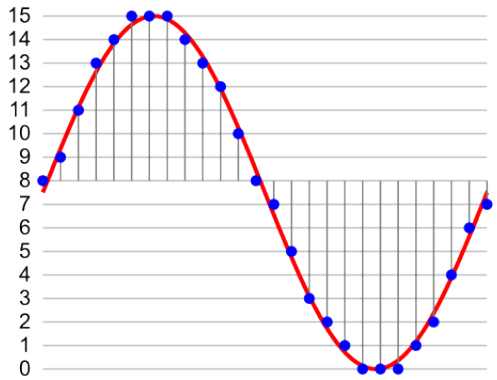
Vzorkování

Podle *Nyquistovy-Shannonovy vzorkovací věty*[33] musí být vzorkovací frekvence alespoň dvojnásobná oproti mezní frekvenci (nejvyšší frekvenční složce) zaznamenaného signálu. V opačném případě dochází k deformaci signálu, tzv. *aliasingu*. Jak již bylo uvedeno, lidské ucho slyší frekvence v rozsahu od 16 Hz do 20 kHz. Vzorkovací frekvence v telekomunikacích je 8 kHz, neboť se přenášejí pouze signály ve standardním telefonním pásmu od 0,3 do 3,4 kHz. U záznamu CD se používá vzorkovací frekvence 44,1 kHz, což odpovídá více než dvojnásobku toho, co průměrně zdravé lidské ucho ještě uslyší. Pro většinu současných aplikací se doporučuje vzorkovací frekvence 48 kHz.

Vzorkovací frekvence

Naměřenou hodnotu signálu je nutno **kvantovat**, tedy zaokrouhlit na nejbližší celé číslo. Čím více paměti máme k dispozici, tím jemnější může být rozlišení jednotlivých hodnot a tím menší zaokrouhlení pak nastává. Zaokrouhlujeme ale vždy, čímž vzniká určitý kvantizační šum. Na obr. 4.9 je znázorněno kvantování signálu do vzorků o velikosti 4 bity, při kterém jsme schopni rozlišit 16 hodnot. Velikost vzorku je obvykle 8 nebo 16 bitů, potom jsme schopni rozlišit 256, resp. 65 536 hodnot signálu.

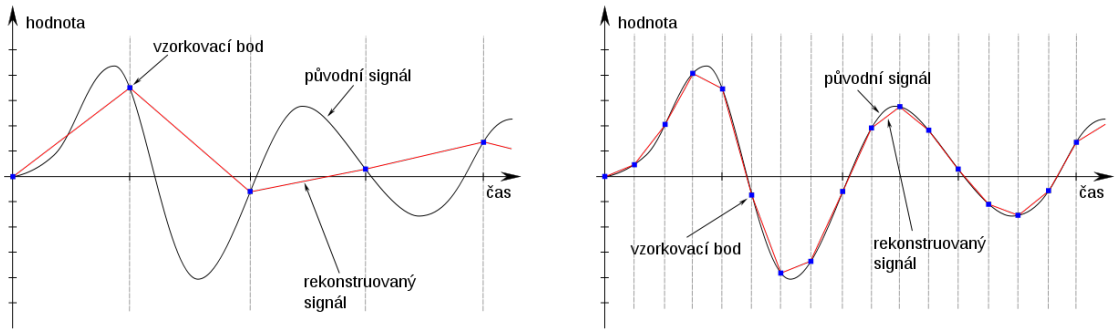
Kvantování



Obrázek 4.9: *Kvantování signálu při velikosti vzorku 4 bity*

[33] Teorém nese jména amerického informatika a fyzika švédského původu Harryho Theodora Nyquista (* 1889, † 1976) a nám již dobře známého matematika Clauda Elwooda Shannona. Nezávisle na nich byl ale objeven dalšími vědci, proto bývá označován také jako Shannonův-Kotělnikovův, Whittakerův-Shannonův-Kotělnikovův nebo dokonce Whittakerův-Nyquistův-Kotělnikovův-Shannonův teorém.

Pokud v rozporu s Nyquistovým-Shannonovým teorémem zvolíme nízkou vzorkovací frekvenci, bude zvuk při rekonstrukci znehodnocen deformací (viz obr. 4.10).



Obrázek 4.10: *Rekonstrukce signálu při nízké vzorkovací frekvenci (vlevo) a vyšší vzorkovací frekvenci (vpravo)*

Paměťová náročnost

Zvuková data mají poměrně vysoké nároky na paměťový prostor. Například na standardním zvukovém CD není možné zaznamenat více než 74 minut.[34] Uvažujeme-li délku nahrávky v sekundách d, počet kanálů k, velikost vzorku v bitech v a vzorkovací frekvenci v Hz f, pak velikost nekomprimované zvukové nahrávky (v bitech) spočítáme ze vztahu

$$V = d \cdot k \cdot f \cdot v.$$

Příklad 4.29

Spočítáme paměťovou náročnost nekomprimované zvukové stereofonní nahrávky o délce 3 minuty se vzorkovací frekvencí 44,1 kHz a velikostí vzorků 16 bitů. Při digitalizaci nahrávky zaznamenáváme po dobu 180 sekund hodnoty signálu pro oba výstupní kanály. Každou sekundu tímto způsobem získáme 44 100 hodnot signálu, které reprezentujeme v podobě 16bitových (2bajtových) čísel, díky tomu můžeme rozlišit 65 536 různých hodnot signálu. Paměťovou náročnost v bajtech tedy spočítáme ze vztahu:

$$2 \cdot 180 \cdot 44\,100 \cdot 2\ \mathrm{B} = 31\,752\,000\ \mathrm{B} \doteq 31\ \mathrm{MiB}$$

Komprese zvuku

Výsledkem digitalizace zvuku jsou velmi rozsáhlé soubory, jejichž komprese je prakticky nevyhnutelná. Používá se většinou ztrátová komprese, při které se ze vstupního signálu odeberou informace, jež člověk neslyší nebo si je neuvědomuje. Tím je možné dosáhnout kompresního poměru až 11 : 1. Lze využít i neztrátovou kompresi, tam se kompresní poměr pohybuje okolo 1,7 : 1. S principem komprese dat se seznámíme v následujících kapitolách (viz kap. 6.3, str. 152, resp. kompresní zvukové formáty na str. 150).

4.4.2 Sekvence MIDI

Druhou možností reprezentace zvuků v počítači jsou **sekvence MIDI** (*Musical Instrument Digital Interface*). Ty obsahují pouze stručný digitální popis výšky jednotlivých tónů, jejich intenzity, délky a nejrůznějších doprovodných efektů. Výstupní zvuková zařízení (zvuková

[34] Toto číslo není náhoda. Firma Sony při vývoji kompaktních disků v roce 1979 trvala na této kapacitě, aby bylo možné na jednom CD zaznamenat celou Beethovenovu Devátou symfonii.

karta, klávesy) z těchto informací umějí vytvářet zvuky. Nereprezentuje se tedy samotný zvuk, ale pouze velmi stručné pokyny pro jeho vytvoření. Proto nelze zaznamenat například lidský hlas ani hudební nástroj, který by syntetizátor nedokázal „zahrát“.

Existuje 128 sekvencí MIDI rozdělených do 16 skupin podle odpovídajícího „nástroje“: klavíry, laděné bicí, varhany, kytary, basy, smyčce, zvuky souboru, žestě, plátkové nástroje, píšťaly, elektronické sólové zvuky, elektronické doprovodné zvuky, elektronické zvukové efekty, etnické zvuky, perkusní zvuky a další zvukové efekty (např. výstřel).

Typickým příkladem zvuku popsaného sekvencí MIDI byly nejstarší vyzváněcí melodie na mobilních telefonech. Jejich výhodou je minimální paměťová náročnost, na druhou stranu převod zvuku do sekvence MIDI je poměrně náročný.

Kontrolní otázky

34. Jaký je princip digitalizace zvukového signálu?

35. Kterými parametry popisujeme digitální zvuk?

36. Jaká je doporučená vzorkovací frekvence zaznamenávaného zvuku?

37. Jaký je princip kvantování zvukového signálu?

38. K čemu slouží sekvence MIDI?

4.5 Příklady na procvičení

Příklad 4.30 Ukažte, jak je dané číslo reprezentováno v paměti počítače v datovém typu bez znaménka na prostoru n bitů:

a) 177, $n = 8$ c) 259, $n = 8$ e) 120, $n = 8$
b) 293, $n = 16$ d) 841, $n = 16$

Příklad 4.31 Vyjádřete dané ve dvojkovém doplňkovém kódu na prostoru n bitů:

a) −41, $n = 8$ c) −17, $n = 8$ e) −130, $n = 8$
b) −256, $n = 16$ d) −566, $n = 16$

Příklad 4.32 Ukažte, jak v paměti počítače na 8 bitech proběhnou následující výpočty:

a) 105 − 27 c) 53 − 19 e) 96 − 74
b) 30 − 11 d) 117 − 78

Příklad 4.33 Vyjádřete číslo v aditivním kódu s posunem o $k = 2^{n-1}$ na prostoru n bitů:

a) −77, $n = 8$ c) −159, $n = 8$ e) −120, $n = 8$
b) 293, $n = 16$ d) 441, $n = 16$

Příklad 4.34 Vyjádřete dané číslo v kódu BCD v obou tvarech (zhuštěném i nezhuštěném):

a) 26 b) 149 c) 5 d) 1892 e) 63

Příklad 4.35 Vyjádřete dané číslo ve formátu single precision (32 b) dle standardu IEEE:

a) 29,417 b) −6,385 c) 114,53 d) −81,0 e) 20,126

Příklad 4.36 Ukažte, jak v paměti počítače proběhnou následující výpočty (single precision):

a) 3,75 + 5,125 b) 0,6 + 0,1 c) −0,3 · 500,25

Příklad 4.37 Předpokládejte textový soubor, jehož obsahem je následující věta: *„Příliš žluťoučký kůň úpěl ďábelské ódy.“* Rozhodněte a zdůvodněte, ve kterém znakovém kódu byl soubor připraven, pokud uvnitř vypadá následovně:

a) `50 F8 ED 6C 69 B9 20 BE 6C 75 BB 6F 75 E8 6B 7D 20 6B F9 F2 ...`
b) `EF BB BF 50 C5 99 C3 AD 6C 69 C5 A1 20 C5 BE 6C 75 C5 A5 6F ...`
c) `50 F8 ED 6C 69 9A 20 9E 6C 75 9D 6F 75 E8 6B 7D 20 6B F9 F2 ...`
d) `FF FE 50 00 59 01 ED 00 6C 00 69 00 69 01 20 00 7E 01 6C 00 ...`
e) `50 A9 A1 6C 69 A8 20 91 6C 75 9F 6F 75 87 6B 98 20 6B 96 A4 ...`
f) `FE FF 00 50 01 59 00 ED 00 6C 00 69 01 69 00 20 01 7E 00 6C ...`

Příklad 4.38 Vypočítejte velikost souboru vzniklého konverzí:

a)	textový soubor, ISO 8859-2,	1024 B	⟶	textový soubor, Windows 1250
b)	textový soubor, UTF-16 LE,	1024 B	⟶	textový soubor, ISO 8859-2
c)	textový soubor, Windows 1250,	2048 B	⟶	binární soubor, formát PDF
d)	textový soubor, UTF-32 BE,	4096 B	⟶	textový soubor, UTF-32 LE
e)	textový soubor, ASCII,	512 B	⟶	textový soubor, UTF-8

Příklad 4.39 Určete barvy podle reprezentace v RGB a nalezněte reprezentaci v CMY:

a) `00FF00` b) `FF0011` c) `010203` d) `00FFFF` e) `696969`

Příklad 4.40 Vypočítejte, kolik bajtů paměti zaberou nekomprimovaná obrazová data:

a)	obraz v pravých barvách,	800 × 600 px	
b)	monochromatický obraz,	640 × 480 px	
c)	obraz v pravých barvách s průhledností,	10 × 15 cm,	600 dpi
d)	obraz v odstínech šedi,	320 × 240 px	
e)	obraz v pravých barvách,	9 × 13 cm,	300 dpi

Příklad 4.41 Vypočítejte, kolik bajtů paměti zaberou nekomprimovaná zvuková data:

a)	délka nahrávky 10 s,	vzorkovací frekvence 32 kHz,	vzorek 8 b,	mono
b)	délka nahrávky 20 s,	vzorkovací frekvence 16 kHz,	vzorek 32 b,	stereo
c)	délka nahrávky 30 s,	vzorkovací frekvence 24 kHz,	vzorek 16 b,	mono
d)	délka nahrávky 40 s,	vzorkovací frekvence 16 kHz,	vzorek 8 b,	stereo
e)	délka nahrávky 50 s,	vzorkovací frekvence 48 kHz,	vzorek 16 b,	mono

Příklad 4.42 Předpokládejte, že zvuková data získaná vzorkováním stereofonní nahrávky o délce 10 sekund za použití vzorkovací frekvence 32 kHz a velikosti vzorku 8 bitů byla zkomprimována do formátu MP3 s datovým tokem 128 kbps. Pokuste se intuitivně určit kompresní poměr (výpočtu se budeme věnovat v kap. 6.3.2 na str. 154).

Shrnutí

Tato velmi důležitá kapitola nám umožnila pochopit principy ukládání různých typů dat do paměti počítače. Seznámili jsme se s reprezentací celých a racionálních čísel, vysvětlili si problémy s vyjádřením znaménka a naučili se používat semilogaritmický tvar čísel s pohyblivou řádovou čárkou. Pak jsme pozornost obrátili k reprezentaci znaků a ukázali, jakým způsobem lze řešit komplikace spojené s národními znaky. Dále jsme se zabývali reprezentací obrazových dat, u kterých jsme objasnili rozdíly mezi rastrovou a vektorovou grafikou. V závěru jsme se seznámili s principem digitalizace zvuku.

Slovníček pojmů

Aditivní kód: kód pro reprezentaci exponentu racionálních čísel, který ke každé hodnotě přičítá určitou konstantu (bias) tak, aby nula ležela uprostřed intervalu.

ASCII: původně sedmibitový znakový kód, který byl později pro reprezentaci národních znaků rozšířen na osmibitový.

Barevná hloubka: množství informace o barvě každého pixelu v rastrovém obrazu.

Barevný model: teoretická pomůcka pro míchání barevných odstínů v počítači.

BCD: kód pro binární reprezentaci desítkových číslic na prostoru 4 nebo 8 bitů.

BOM: jednoznačná identifikace způsobu uspořádání bajtů v textových souborech s vícebajtovým kódováním národních znaků.

CMY, CMYK: subtraktivní barevné modely používané pro reprezentaci barev při tisku.

Digitalizace zvuku: nalezení diskrétní reprezentace spojitého zvukového signálu.

Doplňkový kód: nejpoužívanější číselný kód pro reprezentaci záporných čísel.

Dvojitá přesnost: způsob reprezentace racionálních čísel s pohyblivou řádovou čárkou na prostoru 64 bitů.

EBCDIC: historický osmibitový znakový kód používaný v periferních zařízeních IBM.

Exponent (ve vědeckém tvaru racionálního čísla): počet řádů, o které je nutné posunout řádovou čárku.

Efektivní hustota tisku: vyjádření schopnosti výstupního zařízení zobrazovat body určité kvality.

Fyzický pixel: skutečně vykreslený obrazový bod na papíře nebo obrazovce.

Hustota obrazu: rozlišení obrazu, počet pixelů na jednotku délky.

Inverzní kód: číselný kód pro reprezentaci záporných čísel s využitím inverze všech bitů.

ISO 10646: standardní znaková sada pro vícebajtové kódování.

Jednoduchá přesnost: způsob reprezentace racionálních čísel s pohyblivou řádovou čárkou na prostoru 32 bitů.

Kvantování signálu: zaokrouhlení naměřené hodnoty signálu na nejbližší celé číslo.

Logický pixel: obrazový bod nesoucí informaci o své barvě s nulovým rozměrem.

Mantisa (ve vědeckém tvaru racionálního čísla): racionální číslo zapsané tak, že řádová čárka je umístěna za první nenulovou číslicí.

Národní znak: specifický znak určité abecedy, např. s diakritickým znaménkem.

Obrazový prvek: elementární prvek vektorových obrazů reprezentující geometrické útvary.

Pixel: elementární obrazový bod v rastrových obrazech.
Podtečení: nežádoucí stav, při kterém je číslo zapisované do řádové mřížky tak malé, že je zaokrouhleno na nulu.
Přetečení: nežádoucí stav, při kterém je délka řádové mřížky menší než počet řádů čísla, které do ní chceme zapsat.
Přímý kód: číselný kód pro reprezentaci záporných čísel s využitím znaménkového bitu.
Rastrový obraz: obraz složený z elementárních obrazových bodů určité barvy.
RGB, RGBA: aditivní barevné modely používané pro reprezentaci barev na obrazovkách a projektorech.
Rozšířená část ASCII: nadstavba kódu ASCII obsahující soubor národních znaků dané skupiny příbuzných jazyků.
Rozšířená přesnost: způsob reprezentace racionálních čísel s pohyblivou řádovou čárkou na prostoru 80 bitů.
Řádová mřížka: jednoznačná definice formátu zobrazitelných čísel na určitém prostoru.
Řídicí znak: znak (povel) určený pro ovládání periferních zařízení a programů.
Sekvence MIDI: způsob vyjádření zvuku pomocí popisu výšek jednotlivých tónů, jejich intenzity a délky a dalších doprovodných efektů.
Semilogaritmický tvar čísla: tzv. vědecká notace, ve které je každé číslo vyjádřeno znaménkem a součinem mantisy a základu soustavy umocněného exponentem.
Unicode: standardní znaková sada pro vícebajtové kódování.
UTF: způsob kódování znaků ve standardu UCS, různé varianty.
Vektorový obraz: obraz složený z geometrických útvarů popsaných jejich atributy.
Vzorkování signálu: technika pro digitalizaci zvukového signálu spočívající v pravidelném odečítání hodnoty signálu podle vzorkovací frekvence a její binární reprezentaci na určitém prostoru.
Základní část ASCII: původní standardní znaková sada pro světové jazyky používající latinskou abecedu, bez podpory národních znaků.
Znaková sada: soubor kódů pro reprezentaci znaků v paměti počítače.
Zobrazitelný znak: znak sloužící pro viditelný zápis textové informace.

Řešení kontrolních otázek

1. *K čemu slouží binární řádová mřížka a jakými vlastnostmi je popsána?*

Binární řádová mřížka slouží k jednoznačné definici formátu zobrazitelných čísel v počítači. Je popsána svou délkou (počtem obsazených řádů), jednotkou (nejmenším kladným zobrazitelným číslem), modulem (nejmenším již nezobrazitelným číslem) a pozicí řádové čárky (pevná či pohyblivá).

2. *Kdy může dojít k přetečení nebo podtečení a jak se tyto stavy projevují?*

K přetečení dochází, když číslo, které chceme zapsat do řádové mřížky, má více řádů, než je délka mřížky. Řády, které se nevlezou do mřížky, jsou ignorovány, díky tomu je číslo znehodnoceno. K podtečení dochází v případě, kdy je číslo velmi malé a první nenulová číslice za řádovou čárkou je na pozici, která se již do mřížky nevleze. Číslo je pak zaokrouhleno na nulu.

3. Jakým způsobem a na jakém prostoru jsou v paměti počítače obvykle reprezentována celá čísla bez znaménka?

Celá čísla bez znaménka (přirozená čísla) jsou v paměti počítače reprezentována v binárním kódu na paměťovém prostoru určeném délkou řádové mřížky. Řády, které nejsou obsazeny, jsou doplněny nulami.

4. Jaký je princip přímého kódování celých čísel se znaménkem?

V přímém kódu je znaménko vyjádřeno nejvyšším řádem mřížky, tzv. znaménkovým bitem. Kladná čísla mají v tomto bitu nulu, záporná čísla jedničku. Binární reprezentace absolutní hodnoty čísla je v obou případech stejná.

5. Jaký je princip inverzního kódování celých čísel se znaménkem?

V inverzním kódu je místo záporného čísla zobrazen jeho jedničkový doplněk, který získáme tak, že invertujeme všechny bity na určeném prostoru. Kladná čísla budou mít v nejvyšším bitu nulu, záporná čísla jedničku.

6. Z jakého důvodu se dnes už nepoužívá přímé a inverzní kódování?

Obě kódování mají problém s kladnou a zápornou nulou, kterou kódují jako dvě samostatné hodnoty. Přímý kód navíc nezachovává relace mezi kladným a záporným číslem, takže je obtížné porovnat velikost v binární reprezentaci.

7. Jaké vlastnosti má reprezentace záporných čísel dvojkovým doplňkovým kódem?

Dvojkový doplňkový kód reprezentuje každé záporné číslo $-X$ jeho doplňkem, který je roven číslu $2^n - X$, kde 2^n je modul řádové mřížky. Tento způsob reprezentace nemá kladnou a zápornou nulu a je v současnosti nejpoužívanější.

8. Jaký je interval zobrazitelných hodnot v celočíselném datovém typu na n bitech, uvažujeme-li čísla se znaménkem ve dvojkovém doplňkovém kódu?

Na prostoru n bitů lze ve dvojkovém doplňkovém kódu reprezentovat celá čísla v intervalu od -2^{n-1} do $2^{n-1} - 1$.

9. Na jakém principu je založen aditivní kód a k čemu se používá?

Aditivní kód je založen na přičítání určité konstanty (bias) ke všem reprezentovaným hodnotám tak, aby nula byla posunuta přibližně do poloviny intervalu zobrazitelných hodnot. Obvykle se přičítá konstanta 2^{n-1}, příp. $2^{n-1} - 1$. Výhodou je snadné binární porovnávání reprezentovaných hodnot. Tento kód se používá pro reprezentaci exponentu racionálních čísel s pohyblivou řádovou čárkou.

10. K jakému účelu slouží kód BCD a jak jsou v něm reprezentována čísla?

Kód BCD slouží k binární reprezentaci desítkových číslic. Tento kód byl používán v počítačích od firmy IBM a také ve starších typech kalkulaček se sedmisegmentovými displeji. Principem je kódování dekadických číslic na prostoru 4 bitů (v nezhuštěném tvaru 8 bitů). Případné znaménko je reprezentováno stejně dlouhou binární posloupností, která se nepoužívá pro číslice.

11. Pro která čísla je výhodnější reprezentace s pohyblivou řádovou čárkou?

Reprezentace s pohyblivou řádovou čárkou je výhodnější pro racionální čísla, kde tímto způsobem můžeme ovlivňovat přesnost jejich zobrazení, zejména v jejich zlomkové části.

12. Jak vypadá racionální číslo zapsané v semilogaritmickém tvaru?

Semilogaritmický tvar čísla, tzv. vědecká notace, umožňuje libovolné číslo zapsat kombinací znaménka a součinu mantisy se základem soustavy umocněným exponentem.

13. Co vyjadřují mantisa a exponent racionálního čísla?

Mantisa vyjadřuje upravený tvar čísla, ve kterém je řádová čárka posunuta za první nenulovou číslici. Exponent vyjadřuje počet řádů, o které bylo nutné řádovou čárku posunout. Kladná hodnota exponentu znamená posun vlevo, záporná hodnota znamená posun vpravo.

14. Které standardizované formáty slouží pro reprezentaci racionálních čísel s pohyblivou řádovou čárkou a jak jsou tyto formáty organizovány v paměti?

Standard IEEE 754 rozlišuje tři formáty pro reprezentaci racionálních čísel: single precision (1 + 8 + 23 b), double precision (1 + 11 + 52 b) a extended precision (1 + 15 + 64 b).

15. Jaký je rozsah zobrazitelných hodnot ve formátu single precision?

Ve formátu single precision je rezervován 1 bit na znaménko, 8 bitů na exponent v rozsahu od 2^{-126} po $\left(2 - 2^{-23}\right) \cdot 2^{127}$ a 23 bitů na mantisu, která umožňuje reprezentovat číslo s přesností na 23 binárních číslic za řádovou čárkou.

16. Které vyhrazené hodnoty mohou být výsledkem operace nad racionálními čísly s pohyblivou řádovou čárkou a jak jsou tyto hodnoty reprezentovány v paměti?

Jsou-li v reprezentaci mantisy i exponentu samé nuly, je takové číslo zobrazeno jako nula. Jsou-li v reprezentaci exponentu samé jedničky a v reprezentaci mantisy samé nuly, je takové číslo zobrazeno jako nekonečno. Jsou-li v reprezentaci exponentu samé jedničky a v reprezentaci mantisy alespoň jedna jednička, je hodnota obvykle prohlášena za „nečíslo“ – NaN.

17. Co jsou denormalizovaná čísla a jak jsou reprezentována v paměti?

Denormalizovaná čísla jsou reprezentována nenulovou mantisou a nulovým exponentem. Mantisa je uložena s pevnou řádovou čárkou, která je umístěna za nejvyšší bit. Takový způsob reprezentace je použit pro velmi malé hodnoty, které by při uložení s pohyblivou řádovou čárkou byly zaokrouhleny na nulu.

18. Jakým způsobem jsou znaky reprezentovány v paměti počítače?

Znaky jsou v paměti počítače reprezentovány přirozenými čísly. Soubor kódů se nazývá znaková sada.

19. Jaký je rozdíl mezi řídicím a zobrazitelným znakem?

Zobrazitelný znak slouží pro zápis textové informace v podobě písmen, číslic, interpunkční znamének a dalších textově vyjádřitelných symbolů. Řídicí znak slouží k ovládání periferních zařízení a programů, jsou to povely, které mohou být v každém systému implementovány odlišným způsobem.

20. Z kterých částí se skládá tabulka znakového kódu ASCII?

Tabulka ASCII je složena ze základní části (33 řídicích a 95 zobrazitelných znaků) a rozšířené části (128 národních znaků), která má různé podoby.

21. Co je to národní znak a jaké problémy přináší jeho reprezentace?

Národní znak je zobrazitelný znak, který je specifický pro určitý jazyk (nebo skupinu příbuzných jazyků). Problém spočívá v tom, že obecně existuje více znakových kódů pro jeho reprezentaci, které se navzájem liší uspořádáním znaků v kódové tabulce.

22. Které jednobajtové znakové sady jsou použitelné pro češtinu a slovenštinu?

Z jednobajtových znakových sad pro češtinu a slovenštinu lze v dnešní době doporučit standard ISO 8859-2, příp. Windows-1250, z historických kódů je použitelný kód bratrů Kamenických nebo PC Latin 2.

23. Proč se dnes jednobajtové znakové kódy nahrazují vícebajtovými a který vícebajtový kód je optimální?

Vícebajtové kódy umožňují použití jednotné tabulky pro všechny existující jazyky. S ohledem na vlastnosti vícebajtových kódů je optimální variantou UTF-8, ve kterém je každý znak reprezentován nejčastěji na 1–2 bajtech a speciální znaky na 3–4 bajtech.

24. Jaký je při reprezentaci vícebajtových znakových kódů v počítači rozdíl mezi variantou Little Endian a Big Endian?

Varianty se liší pořadím bajtů v reprezentaci. Ve variantě Little Endian se nejdříve ukládá nejméně významný (poslední) bajt a za něj ostatní bajty až po nejvíce významný (první) bajt, ve variantě Big Endian se nejdříve ukládá nejvíce významný bajt (první) a za něj ostatní bajty až po nejméně významný bajt (poslední).

25. K čemu slouží barevné modely a které vlastnosti je popisují?

Barevné modely slouží ke specifikaci jednotlivých barev v počítači. Každý model je určen základními barvami, ze kterých se skládají všechny odstíny, poměry, v jakých je možné barvy skládat, a způsobem míchání základních odstínů.

26. Které barvy tvoří základ modelů RGB a CMYK?

Základními barvami aditivního modelu RGB jsou červená, zelená a modrá. Základními barvami subtraktivního modelu CMYK jsou azurová, purpurová, žlutá a černá.

27. Čím se liší rastrový a vektorový obraz?

Rastrový obraz se skládá z elementárních obrazových bodů (pixelů) určité barvy. Vektorový obraz se skládá z obrazových objektů (vektorů) reprezentovaných geometrickými útvary a jejich atributy.

28. Které parametry rastrového obrazu mají vliv na jeho velikost v paměti?

Velikost nekomprimovaného rastrového obrazu v paměti je ovlivněna jeho rozměry a hustotou (součinem získáme počet pixelů) a bitovou hloubkou.

29. Co vyjadřuje efektivní hustota rastrového obrazu?

Efektivní hustota obrazu vyjadřuje schopnost výstupního zařízení zobrazit body určité kvality. V důsledku rozdílné velikosti logického (v obrazu) a fyzického (na obrazovce nebo papíru) pixelu bývá efektivní hustota tisku výrazně menší než hustota obrazu.

30. Co je to bitová hloubka rastrového obrazu?

Bitová hloubka představuje množství informace o barvě každého pixelu v rastrovém obrazu. Podle počtu bitů rozlišujeme obrazy monochromatické, s barevnou paletou, s odstíny šedi a v pravých barvách.

31. Jak velká je informace o barvě každého pixelu v monochromatickém obrazu a jak velká je v obrazu s pravými barvami?

V monochromatickém obrazu má informace o barvě každého pixelu velikost 1 bit. V obrazu s pravými barvami má tatáž informace velikost 24 bitů u modelu RGB, příp. 32 bitů u modelu CMYK.

32. Které atributy můžeme nastavovat objektům ve vektorovém obrazu?

Objektům ve vektorovém obrazu lze nastavovat souřadnice na kreslicí ploše, rozměry, měřítko, typ a barvu kreslicí čáry, druh a barvu výplně uzavřených tahů, písmo textových objektů apod.

33. Pro které typy obrazů je vhodná vektorová grafika a pro které vhodná není?

Vektorová grafika je vhodná pro ilustrace, diagramy, schémata, grafy apod. Není vhodná pro barevné fotografie a obecně pro obrazy s velkým množstvím barevných ploch.

34. Jaký je princip digitalizace zvukového signálu?

Digitalizace zvukového signálu spočívá v nalezení diskrétní reprezentace spojitého signálu. Pomocí vzorkování jsou v pravidelných intervalech odčítány hodnoty signálu, které jsou následně vyjádřeny binární posloupností.

35. Kterými parametry popisujeme digitální zvuk?

Digitální zvuk je popsán vzorkovací frekvencí a velikostí vzorků, tedy jemností rozlišení jednotlivých hodnot.

36. Jaká je doporučená vzorkovací frekvence zaznamenávaného zvuku?

Doporučená vzorkovací frekvence by podle Nyquistovy-Shannonovy věty měla být alespoň dvojnásobná ve srovnání s mezní frekvencí zaznamenávaného zvuku.

37. Jaký je princip kvantování zvukového signálu?

Kvantování spočívá v zaokrouhlení naměřené hodnoty signálu na nejbližší celé číslo. Jemnost rozlišení jednotlivých hodnot je dána počtem bitů pro uložení vzorku.

38. K čemu slouží sekvence MIDI?

Sekvence MIDI umožňují popsat zvuk výškou jednotlivých tónů, jejich intenzitou, délkou a doprovodnými efekty. Výstupní zvuková zařízení z těchto informací umějí složit výsledný zvuk.

5 Přenos, kódování a zabezpečení dat

Úvod

Po prostudování předchozích kapitol již máme poměrně jasnou představu o tom, jakým způsobem jsou data různého typu reprezentována v paměti počítače. Obvykle v této souvislosti máme na mysli operační paměť, v dalších kapitolách se budeme zabývat možnostmi reprezentace dat v souborech. Než se ale k této problematice dostaneme, budeme se věnovat přenosu dat a souvisejícím otázkám jejich kódování a zabezpečení.

Cíl

Cílem této kapitoly je pochopení problémů souvisejících s přenosem dat. Ukážeme si výhody digitálního signálu oproti analogovému signálu, seznámíme se s metodami kódování dat a naučíme se stanovit optimální binární kód. V závěru kapitoly se budeme věnovat možnostem zabezpečení dat proti úmyslnému i neúmyslnému poškození během přenosu.

Návaznosti

Pro studium této kapitoly je nutné chápat binární číselnou soustavu, principy reprezentace dat v paměti počítače a orientovat se v teorii informace, zejména v oblasti výpočtu informační entropie.

Studijní čas

Čas potřebný k nastudování této kapitoly je asi 10 hodin.

5.1 Přenos dat

Signál

Pokud chceme s daty jakkoliv pracovat, nevyhneme se nikdy jejich přenosu do místa zpracování. Příjem a zpracování dat je vždy spojen s nějakým fyzikálním pochodem. Fyzikální veličinu, která představuje stavy nebo procesy fyzikálního média, nazýváme **signál**. Matematicky lze signál popsat funkcí $s = f(x, y, z, t)$, kde s je libovolný signál vyjádřený nezávislými souřadnicemi místa (x, y, z) a časovým parametrem t. Signály jsou pak vyjádřeny jako časoprostorová funkce, přičemž ne všechny parametry se musí vždy projevit.

Podle časového parametru t můžeme signály rozdělit na *diskrétní* (jsou definovány jen pro určité časové okamžiky, např. telegrafní zprávy) a *spojité* (jsou definovány pro všechny časové okamžiky, např. telefonní hovory), příp. na *statické* (nezávisí na čase, např. kniha, mapa) a *dynamické* (jsou funkcemi času, např. sportovní přenos v televizi). Je zřejmé, že jednu a tutéž informaci lze přenášet pomocí různých signálů o různé fyzikální podstatě. Je také pravda, že tentýž typ signálu lze použít k přenosu různých informací.

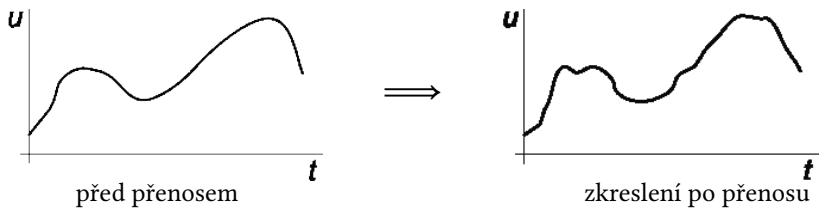
5.1.1 Spojitý a diskrétní signál

Podívejme se proto nyní na způsob přenosu informace pomocí nějaké fyzikální veličiny, tedy signálu. Představme si, že chceme přenášet hudbu. V koncertní síni jsou instalovány mikrofony, vzdálený posluchač má k dispozici reproduktory. Víme, že mikrofon je zařízení se schopností převést zvuk na elektrický signál, reproduktor zase naopak převede elektrický signál na zvuk.

Spojitý signál

Zazní-li nějaký tón, na výstupu mikrofonu se objeví elektrický signál o určitém napětí. Toto napětí se mění v přímé závislosti na snímaném zvuku. Pro další použití je nezbytné tento signál zesílit, pak jej přenést (pro jednoduchost budeme předpokládat pouze přenos vodičem) a přivést do reproduktoru. Tam se elektrický signál opět změní ve výsledný zvuk. Reproduktor zde zpracovává elektrický signál v každém okamžiku, každá přenesená hodnota napětí se tedy používá pro přenášení informace. Protože elektrické napětí je veličina, která může nabývat nekonečně mnoha hodnot v určitém intervalu a každá z těchto hodnot je použita pro přenos informace, mluvíme o **spojitém (analogovém) signálu**.

Při každé transformaci nebo úpravě signálu však dochází ke *zkreslení*. Průběh elektrického signálu, který přivádíme do reproduktoru, se již nerovná průběhu signálu vytvořeného mikrofonem. Posluchač to vnímá například jako omezení čistoty zvuku, omezení hloubek nebo výšek či změnu barvy zvuku. Při přenosu a zpracování se navíc k příslušnému signálu přidávají všudypřítomné šumy, případně i jiné rušení (elektromagnetické vlny vznikající činností elektrických zařízení, různé výboje). To vše je na výstupní straně převedeno na zvuk – různé šumy, praskot, pískání apod. Spojitý signál je tedy při přenosu vždy zkreslen (obr. 5.1).



Obrázek 5.1: *Zkreslení spojitého signálu*

Od dob, kdy se začal elektrický signál používat pro přenos zvukové informace, se vyvíjela stále dokonalejší zařízení, která přidávala minimální zkreslení a šumy. Avšak nikdy nebyly tyto negativní průvodní vlastnosti spojitého signálu zcela eliminovány. Proto se hledaly metody, jak přenášet informaci jiným způsobem, aby bylo možné všechny poruchy a nedokonalosti přenosového kanálu nějak odfiltrovat.

Teprve nástup výpočetní techniky a technologií postavených na číslicových obvodech znamenal revoluční zvrat v přenosu informací. Pro vysvětlení základního principu tohoto přenosu se vrátíme k našemu příkladu s přenosem hudby.

Diskrétní signál

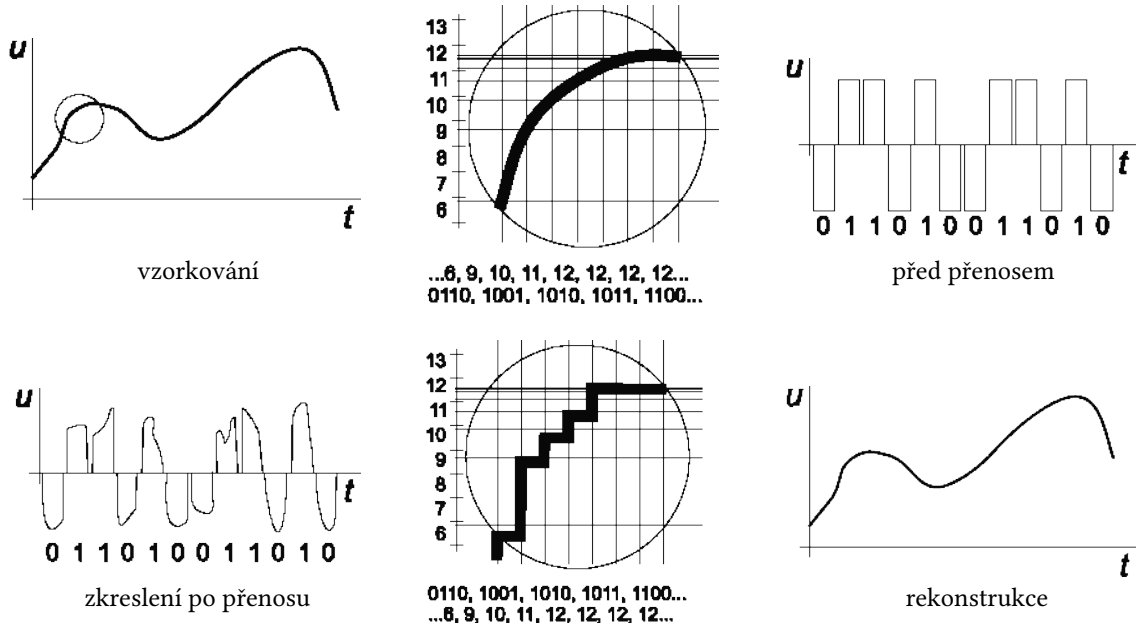
Elektrické napětí z mikrofonu nechceme dál přenášet, protože jej nechceme zkreslit. Proto si signál přesně popíšeme a přeneseme jen tento popis. Například každých 50 milisekund změříme napětí a jeho okamžitou hodnotu vyjádříme pomocí celého čísla v intervalu od 0 do 65 535. Tím získáme **diskrétní (digitální) signál**, neboť z původního signálu vybereme jen některé hodnoty. Výběr je proveden jen v určitých okamžicích, určité reálné

hodnoty navíc „zaokrouhlíme“ jen na číslo příslušné hladiny. Problematice *vzorkování* jsme se věnovali již v kap. 4.4.1 (viz str. 83) v souvislosti s reprezentací zvukových dat v paměti počítače.

Získané vzorky přeneseme běžným způsobem, jako by to byl analogový signál. Na výstupní straně (před reproduktorem) přenášená čísla znovu převedeme na příslušnou napěťovou úroveň a získáme tak původní průběh signálu. Můžeme namítnout, že výběrem se informace o původním signálu nepřenáší všechna a že nelze beze zbytku na cílové straně rekonstruovat vše. Při dostatečně vysoké frekvenci vzorkování však získáme zanedbatelně malý rozdíl výsledného a původního signálu, rozhodně menší, než by způsobilo rušení při přenosu a zpracování analogového signálu.

Princip přenosu

Na první pohled se může zdát, že uvedený způsob přenosu musí být daleko citlivější na rušení, neboť přenos je velmi komplikovaný a přenášená čísla se mohou zkreslit stejně jako původní signál. V praxi tomu tak ovšem vůbec není. Přenos čísel je uskutečňován signálem, který se sice zkreslí stejně jako každý jiný, ale číselná hodnota se při přenosu nemění. Můžeme si to představit tak, že na začátku přenosu je číslo zapsáno krásnými hladkými tahy, po přenosu se tyto tahy zkreslí, jsou sice kostrbaté, ale kostrbatá nula je pořád nula a kostrbatá jednička je zase jednička. Po zjištění správné hodnoty se signál může opět rekonstruovat přesně v původním tvaru (obr. 5.2). Jestliže se při přenosu objeví tak velké zkreslení, že se chybně interpretuje číslice, je možné tomu čelit implementací některé z metod zabezpečení informace proti nedokonalostem přenosového kanálu. Těmto metodám se budeme věnovat v kap. 5.3.



Obrázek 5.2: *Přenos diskrétního signálu – nahoře vzorkování před přenosem, dole rekonstrukce po přenosu*

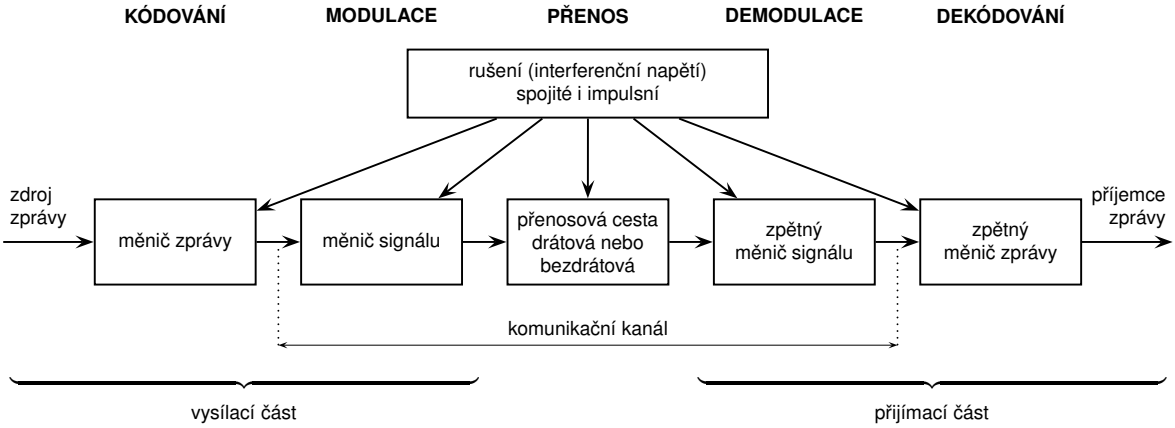
Diskrétní signál lze tedy přenášet bez zkreslení a v kvalitě, která je pro spojitý signál nedosažitelná. Proto se ve všech moderních přenosových technologiích používá jen digitálních metod přenosu – mobilní telefony, družicové přenosy, dnes také digitální televizní a rozhlasové vysílání. Mezi digitální přenosy lze zahrnout veškeré výměny počítačových

dat. Rozvoj technického vybavení počítačů směřuje do stavu, kdy veškerá komunikační technika přejde na počítačové, tedy digitální zpracování všech signálů. Snad jedinou výhodou analogového signálu proti digitálnímu je jednoduchost přenosu a zpracování, protože se pracuje s daleko nižšími frekvencemi. Stoupající technologická úroveň však tuto nevýhodu poněkud stírá, neboť konstrukce zařízení pro zpracování digitálního signálu je již velmi dobře zvládnutá, poměrně levná a široce použitelná.

5.1.2 Přenosový řetězec

Komunikace

Při přenosu a výměně informací vzniká mezi komunikujícími objekty informační vazba, která umožňuje *komunikaci*. Aby bylo možné komunikaci realizovat, musí existovat *komunikační kanál*, po kterém se informace přenášejí. Princip přenosu lze znázornit obecným schématem zvaným **přenosový řetězec** (obr. 5.3).



Obrázek 5.3: *Přenosový řetězec*

Informace musí být schopná komunikačním kanálem projít, proto je obvykle potřeba ji transformovat, tj. vyjádřit znaky v jiném jazyku, zapsat jinou abecedou. Přiřazení znaků jedné abecedy znakům jiné abecedy se nazývá *kódování*, opačný postup pak *dekódování*. Této problematice se budeme věnovat v kap. 5.2.

V průběhu přenosu se může objevit různé *rušení* způsobující, že data nesoucí informaci nebudou doručena ve stejné podobě, ve které byla vyslána. Zabezpečení dat proti tomuto rušení se budeme věnovat v kap. 5.3 (viz str. 110).

V přenosovém řetězci je tedy potřebné informaci k přenosu připravit (zakódovat), vyslat ji do přenosového kanálu, přijmout přijímacím zařízením a následně opět dekódovat do podoby použitelné pro další zpracování v cílovém počítači.

Kontrolní otázky

1. Jaký je rozdíl mezi spojitým a diskrétním signálem?
2. Proč je možné diskrétní signál přenášet bez zkreslení?
3. K čemu slouží přenosový řetězec a z jakých částí se skládá?

5.2 Kódování dat

Z hlediska optimalizace je vhodné, aby každá přenášená zpráva obsahovala co nejvíce informace. Proto při přenosu zpráv využíváme principu kódování informací. Ten spočívá v transformaci informace vyjádřené v nějakém jazyku do informace v jazyku jiném. Vzájemné přiřazování znaků dvou abeced nazýváme **kódování** a inverzní postup **dekódování**. Předpis, který toto přiřazení umožňuje, nazýváme **kód**. Příkladem může být kód pro reprezentaci záporných čísel, znakový kód ASCII nebo Morseova[35] abeceda pro přenos dat telegrafem.

Kódování a entropie

Za efektivní je považován takový kód, jehož výsledkem bude nejmenší počet informačních prvků, a tím i nejkratší doba přenosu. Je důležité, aby každý přenášený znak obsahoval co největší množství informace. Z pohledu teorie informace je tedy hospodárný takový kód, jehož každý kódový znak má *maximální entropii*, tj. všechny znaky mají ve zprávě stejně velkou četnost výskytu, dalším kritériem je střední délka kódového slova.

Redundance kódu

Kvalitativně vyjadřuje hospodárnost kódu **redundance** (nadbytečnost), kterou můžeme vypočítat podle vztahu

$$R = 1 - \frac{H}{H_{\max}},$$

kde H je skutečná entropie jazyka a $H_{\max}$ je teoretická maximální entropie při použití téže abecedy (všechny znaky abecedy jsou stejně možné a jejich vzájemný výskyt není závislý). Pro kódy, jejichž výsledkem jsou počítačově zpracovatelná data, tedy posloupnost nul a jedniček, je teoretická entropie vždy $H_{\max} = 1$, redundance je potom $R = 1 - H$. Evropské jazyky mají redundanci větší než 0,5, některé výpočty dokonce berou v úvahu interval 0,68 až 0,75.

5.2.1 Rovnoměrné kódování

Blokové kódování

Tento typ kódování je založen na jednoduchém způsobu, kdy je každý původní znak zakódován do binárního kódu se stejně dlouhou posloupností nul a jedniček. Délka kódu odpovídá entropii původních znaků. Kódování je obvykle jednodušší a rychlejší na zpracování, ale někdy také méně hospodárné. Je známé také jako **blokové** nebo **Baudotovo**[36] **kódování**.

Příklad 5.1

Mějme zdroj informace, který produkuje čtyři nezávislé znaky A, B, C, D. Tyto znaky chceme zakódovat rovnoměrným kódem. Počet bitů potřebných k zakódování znaků zjistíme tak, že vypočítáme střední délku kódového slova, tj. informační entropii zdroje:

$$H(X) = -4 \cdot (0{,}25 \cdot \log_2 0{,}25) = 2 \text{ bity/znak}.$$

Každý znak tedy bude zakódován do posloupnosti dvou binárních číslic. Na konkrétní podobě kódu příliš nezáleží, musíme jen zajistit, aby bylo přiřazení jednoznačné. Zakódujeme například znak A jako 00, znak B jako 01, znak C jako 10 a znak D jako 11.

[35] Samuel Finley Breese Morse (* 1791, † 1872) byl americký vynálezce, sochař a malíř. V roce 1837 získal patent na elektrický telegraf, podílel se také na tvorbě přenosové abecedy.

[36] Jean-Maurice-Émile Baudot (* 1845, † 1903) byl francouzský telegrafní inženýr, autor mezinárodního pětibitového kódu pro dálnopis. Proto bývá často rovnoměrné kódování pojmenováno na jeho počest.

Zaměřme se nyní na efektivnost rovnoměrného kódu. Chceme-li zjistit jeho redundanci, musíme nejdříve spočítat informační entropii. K tomu nám ale chybí ještě jedna velmi důležitá proměnná – pravděpodobnost výskytu jednotlivých znaků ve zprávě.

Příklad 5.2

Stejná pravděpodobnost

Uvažujme zprávu složenou ze znaků A, B, C, D se stejnou pravděpodobnost výskytu. Budeme uvažovat zprávu o délce například 1 000 znaků. Podle dané pravděpodobnosti by taková zpráva byla složena z 250 A, 250 B, 250 C a 250 D. Četnosti znaků původní zprávy a znaků kódu zaneseme do tabulky:

Znak	Četnost znaků zprávy		Kód	Četnost znaků kódu	
	relativní	absolutní		nuly	jedničky
A	0,250	250	00	500	0
B	0,250	250	01	250	250
C	0,250	250	10	250	250
D	0,250	250	11	0	500
Zpráva	×	1 000	×	1 000	1 000

Z tabulky je zřejmé, že zpráva o délce 1 000 znaků bude složena z 2 000 kódových znaků, ve kterých bude počet nul a jedniček stejný. Spočítáme informační entropii kódu:

$$H(X) = -\left(\frac{1\,000}{2\,000} \cdot \log_2 \frac{1\,000}{2\,000} + \frac{1\,000}{2\,000} \cdot \log_2 \frac{1\,000}{2\,000}\right) = 1{,}000.$$

Kód má maximální entropii, je velmi efektivní a jeho redundance je nulová. Je to dáno také tím, že všechny čtyři kódové kombinace pro dvouciferná binární čísla byly využity.

Příklad 5.3

Různá pravděpodobnost

Mějme nyní rozdílnou pravděpodobností výskytu jednotlivých znaků. Ve zprávě složené z 1 000 znaků se objeví 500 A, 250 B, 125 C a 125 D. Jak se změní hospodárnost kódu? Četnosti znaků zprávy a kódových znaků zaneseme do tabulky:

Znak	Četnost znaků zprávy		Kód	Četnost znaků kódu	
	relativní	absolutní		nuly	jedničky
A	0,500	500	00	1 000	0
B	0,250	250	01	250	250
C	0,125	125	10	125	125
D	0,125	125	11	0	250
Zpráva	×	1 000	×	1 375	625

Z tabulky vyplývá, že zpráva o délce 1 000 znaků bude složena z 2 000 kódových znaků, ve kterých budou převažovat nuly. Dosadíme do vzorce a spočítáme entropii kódu:

$$H(X) = -\left(\frac{1\,375}{2\,000} \cdot \log_2 \frac{1\,375}{2\,000} + \frac{625}{2\,000} \cdot \log_2 \frac{625}{2\,000}\right) = 0{,}896.$$

Kód není efektivní, relativní redundance dosahuje 1 − 0,896 = 0,104, tj. více než 10 %.

Pozornému čtenáři zajisté neušlo, že stačí provést drobnou změnu v kódových znacích a výsledek může být zcela odlišný.

Příklad 5.4

Upravíme předchozí příklad tak, že k původním znakům zprávy přiřadíme jiné kódové znaky. Četnosti znaků zprávy a kódových znaků zaneseme do tabulky:

Znak	Četnost znaků zprávy		Kód	Četnost znaků kódu	
	relativní	absolutní		nuly	jedničky
A	0,500	500	01	500	500
B	0,250	250	10	250	250
C	0,125	125	11	0	250
D	0,125	125	00	250	0
Zpráva	×	1 000	×	1 000	1 000

Z tabulky vidíme, že zpráva o délce 1 000 znaků bude složena z 2 000 kódových znaků, ve kterých bude počet nul a jedniček stejný. Spočítáme informační entropii kódu:

$$H(X) = -\left(\frac{1\,000}{2\,000} \cdot \log_2 \frac{1\,000}{2\,000} + \frac{1\,000}{2\,000} \cdot \log_2 \frac{1\,000}{2\,000}\right) = 1{,}000.$$

Dostali jsme optimální kód i přesto, že jsme různě pravděpodobné znaky zprávy zakódovali rovnoměrným způsobem.

Problémem rovnoměrného kódování je především to, že přiřazuje všem znakům dané abecedy stejně dlouhý kód. Jak je patrné z uvedených příkladů, je tato vlastnost výhodná v případě, že všechny znaky zprávy jsou stejně pravděpodobné. V běžném textu se ale často některé znaky vyskytují podstatně více než jiné. Logická úvaha pak vede k tomu, že by tyto znaky bylo z hlediska optimalizace přenosu lepší zakódovat kratším kódem. K tomu ovšem musíme využít nerovnoměrné kódování. Předchozí příklad sice ukázal, že i pro různé pravděpodobnosti výskytu znaků lze v některých případech nalézt optimální rovnoměrný kód, nicméně je to zbytečné, protože vždy lze nalézt nerovnoměrný kód se stejnou nebo vyšší efektivitou.

5.2.2 Nerovnoměrné kódování

Prefixové kódování

Princip nerovnoměrného kódování je založen na tom, že každému znaku je přiřazen jinak dlouhý kód. Pokud žádný ze symbolů kódové abecedy není prefixem jiného (delšího) symbolu téže abecedy, nazývá se toto kódování **prefixové**[37]. Pro konstrukci a zpracování je takové kódování obtížnější, ale může být maximálně hospodárné. Nejčetnější znaky mívají kód nejkratší, znaky s menší četností výskytu mají kód delší. Nerovnoměrné kódování lze řešit mnoha způsoby. Podívejme se, zda bude hospodárnost takového kódu závislá na pravděpodobnosti výskytu jednotlivých znaků.

[37] Pokud je nějaký kód prefixový, je možné řetězce symbolů tohoto kódu jednoznačně dekódovat, aniž by mezi jednotlivými symboly musely být oddělovače. Pozornému čtenáři jistě neuniklo, že tuto vlastnost má také každý blokový kód.

Příklad 5.5

Stejná pravděpodobnost

Mějme zprávu složenou ze znaků A, B, C, D, ve které mají všechny znaky stejnou pravděpodobnost výskytu. Uvažujme nerovnoměrný kód, který znak A kóduje jako 0, znak B jako 10, znak C jako 110 a znak D jako 111. Dále uvažujme zprávu o délce 1 000 znaků. Podle dané pravděpodobnosti bude taková zpráva složena z 250 A, 250 B, 250 C a 250 D. Četnosti znaků zprávy a kódových znaků zaneseme do tabulky:

Znak	Četnost znaků zprávy		Kód	Četnost znaků kódu	
	relativní	absolutní		nuly	jedničky
A	0,250	250	0	250	0
B	0,250	250	10	250	250
C	0,250	250	110	250	500
D	0,250	250	111	0	750
Zpráva	×	1 000	×	750	1 500

Vidíme, že zpráva o délce 1 000 znaků bude složena z 2 250 kódových znaků, ve kterých budou převažovat jedničky. Spočítáme informační entropii tohoto kódu:

$$H(X) = -\left(\frac{750}{2\,250} \cdot \log_2 \frac{750}{2\,250} + \frac{1\,500}{2\,250} \cdot \log_2 \frac{1\,500}{2\,250}\right) = 0{,}918.$$

Kód je sice poměrně hospodárný, ale jeho redundance je vyšší než v případě rovnoměrného kódu: 1 − 0,918 = 0,082, tj. 8,2 %. Střední délka kódového slova je 2,25 bitů/znak.

Příklad 5.6

Různá pravděpodobnost

Mějme nyní rozdílnou pravděpodobností výskytu jednotlivých znaků. Ve zprávě složené z 1 000 znaků se objeví 500 A, 250 B, 125 C a 125 D. Jaký to bude mít vliv na hospodárnost kódu? Četnosti znaků zprávy a kódových znaků zaneseme do tabulky:

Znak	Četnost znaků zprávy		Kód	Četnost znaků kódu	
	relativní	absolutní		nuly	jedničky
A	0,500	500	0	500	0
B	0,250	250	10	250	250
C	0,125	125	110	125	250
D	0,125	125	111	0	375
Zpráva	×	1 000	×	875	875

Vidíme, že zpráva o délce 1 000 znaků bude složena z 1 750 kódových znaků, ve kterých bude stejný počet nul a jedniček. Spočítáme informační entropii tohoto kódu:

$$H(X) = -\left(\frac{875}{1\,750} \cdot \log_2 \frac{875}{1\,750} + \frac{875}{1\,750} \cdot \log_2 \frac{875}{1\,750}\right) = 1{,}000.$$

Tento kód je maximálně hospodárný, jeho relativní redundance je nulová. Střední délka kódového slova je 1,75 bitů/znak.

Z předchozích příkladů je zřejmé, že jsou-li pravděpodobnosti znaků ve zprávě různé, je nerovnoměrné kódování obvykle hospodárnější. Čím větší jsou rozdíly v pravděpodobnostech výskytu, tím je nerovnoměrné kódování efektivnější. Z toho plyne, že chceme-li najít optimální (nejkratší) kód, musíme hledat mezi nerovnoměrnými kódy. Naučíme se proto dva důležité algoritmy, které vznikly už v 50. letech 20. století a dnes jsou úspěšně aplikovány při kompresi dat různého druhu. Také si ukážeme, jak ověřit, zda pro dané požadavky na délku kódových slov lze prefixový kód vůbec sestrojit.

5.2.3 Hledání optimálního kódu

Při hledání optimálního kódu využijeme některého z existujících algoritmů, nebo můžeme na základě požadavků na délky kódových slov vytvářet kód ručně.

Příklad 5.7

Předpokládejme zdroj zpráv se čtyřmi kódovými znaky a uvedenými pravděpodobnostmi a k němu tři různé nerovnoměrné kódy:

Znak	$p(x)$	Kód 1	Kód 2	Kód 3
A	0,5	0	0	0
B	0,3	01	1	10
C	0,1	011	01	110
D	0,1	111	001	111

Který z těchto tří kódů je nejlepší? Výpočtem entropie zjistíme, že nejméně efektivní je kód 2 (entropie 0,961), navíc jej není možné jednoznačně dekódovat. Kód 1 a kód 3 mají stejnou entropii (0,997). Jsou ale stejně efektivní? Pozornému čtenáři jistě neušlo, že mezi nimi je jeden podstatný rozdíl. Pojďme si to ukázat na příkladu kódové zprávy AABAC, která je vyslána směrem k příjemci v podobě binární posloupnosti `00010011` (kód 1), resp. `00100110` (kód 3). Příjemce ve snaze dekódovat zprávu pocítí významný rozdíl mezi těmito kódy, neboť zatímco kód 3 lze dekódovat znak po znaku průběžně, kód 1 je možné dekódovat až po přijetí celé zprávy a musí se přitom postupovat od konce. Kód 3 je efektivnější, protože byl zkonstruován jako prefixový.

Nabízí se logická otázka: Lze navrhnout prefixový kód za všech okolností? Odpověď přináší následující příklad.

Příklad 5.8

Předpokládejme zdroj zpráv s 10 kódovými znaky (A–J), jejichž pravděpodobnost je 0,15 u znaků A a B, 0,1 u znaků C–H a 0,05 u znaků I a J. Chceme sestrojit binární prefixový kód tak, aby délka kódového slova byla 2 bity pro A a B, 3 bity pro C–H a 4 bity pro I a J. Pro znaky A a B zvolíme prefix `0x` (tedy například `00` pro A, `01` pro B), pro znaky C–H pak musíme zvolit prefix `1xx`, ale zbývající dva bity neposkytují dostatek kombinací pro šest různých hodnot. Vidíme, že prefixový kód s těmito vlastnostmi nemůže existovat.

Kraftova nerovnost

Binární prefixový kód pro n kódových znaků s délkami kódových slov $d_1, d_2, \ldots, d_n$ existuje právě tehdy, pokud platí **Kraftova nerovnost**:[38]

$$2^{-d_1} + 2^{-d_2} + \ldots + 2^{-d_n} \leq 1.$$

McMillanova věta

Dále platí, že *každý jednoznačně dekódovatelný kód splňuje Kraftovu nerovnost.* Toto tvrzení je známé jako **McMillanova**[39] **věta** a vyplývá z něj, že každý jednoznačně dekódovatelný kód buď musí být prefixový, nebo musí existovat jiný kód nad stejnou kódovou abecedou se stejnými délkami kódových slov, který prefixový je.

Příklad 5.9

Ověříme Kraftovu nerovnost pro kód z předchozího příkladu: $2 \cdot 2^{-2} + 6 \cdot 2^{-3} + 2 \cdot 2^{-4} = 1{,}375$. Vidíme, že neplatí, prefixový kód pro zadané délky kódových slov skutečně neexistuje.

Příklad 5.10

Zkusme zvolit jinou délku kódových slov, například 2 bity pro znaky A a B a 4 bity pro znaky C–J. Ověříme Kraftovu nerovnost: $2 \cdot 2^{-2} + 8 \cdot 2^{-4} = 1$, prefixový kód tedy existuje a může vypadat například takto (opět uvažujeme zprávu o délce 1 000 znaků):

Znak	Četnost znaků zprávy		Kód	Četnost znaků kódu	
	relativní	absolutní		nuly	jedničky
A	0,150	150	00	300	0
B	0,150	150	01	150	150
C	0,100	100	1000	300	100
D	0,100	100	1100	200	200
E	0,100	100	1010	200	200
F	0,100	100	1001	200	200
G	0,100	100	1110	100	300
H	0,100	100	1101	100	300
I	0,050	50	1011	50	150
J	0,050	50	1111	0	200
Zpráva	×	1 000	×	1 600	1 800

Zpráva o délce 1 000 znaků bude složena z 3 400 kódových znaků, v nichž budou mírně převažovat jedničky. Dosadíme do vzorce a spočítáme informační entropii:

$$H(X) = -\left(\frac{1\,600}{3\,400} \cdot \log_2 \frac{1\,600}{3\,400} + \frac{1\,800}{3\,400} \cdot \log_2 \frac{1\,800}{3\,400}\right) = 0{,}997$$

Kód je efektivní s redundancí pouze 0,3 %. Střední délka kódového slova je 3,4 bitů/znak.

Nyní již máme poměrně jasnou představu o vlastnostech a výhodách binárních prefixových kódů. Stále ale nevíme, jakým způsobem postupovat při konstrukci kódu tak, aby byl co nejefektivnější. Pojďme se nyní proto věnovat metodám konstrukce optimálních binárních prefixových kódů.

[38] Leon Gordon Kraft (* 1923, † 1989) publikoval tento vztah v roce 1949 ve své disertaci na MIT.
[39] Brockway McMillan (* 1915, † 2016) byl americký vládní úředník a vědec.

Shannonovo-Fanovo kódování

Jednou z metod prefixového kódování je **Shannonovo-Fanovo**[40] **kódování**. Algoritmus hledání kódových znaků lze popsat těmito kroky:

1. Znaky uspořádáme sestupně podle pravděpodobnosti jejich výskytu ve zprávě.
2. Vypočteme kumulativní pravděpodobnosti jednotlivých znaků.
3. Rozdělíme znaky do dvou skupin tak, aby kumulativní pravděpodobnosti v obou skupinách byly blízké 0,5. Jedné skupině přiřadíme nulu, druhé jedničku.
4. Předchozí krok opakujeme tak dlouho, dokud existují vícečlenné skupiny znaků.

Příklad 5.11

Mějme zprávu složenou ze znaků x_1, x_2, ..., x_5, které se vyskytují s pravděpodobnostmi $p(x_1) = 0{,}3$, $p(x_2) = 0{,}24$, $p(x_3) = 0{,}2$, $p(x_4) = 0{,}15$ a $p(x_5) = 0{,}11$. Znaky x_i a jejich pravděpodobnosti $p(x_i)$ zaneseme do tabulky a dopočítáme kumulativní pravděpodobnosti $s(x_i)$. Pak opakovaně rozdělujeme znaky do dvou skupin tak dlouho, až získáme pouze jednočlenné skupiny.

Znak	$p(x)$	$s(x)$	Skupiny			Kód
x_1	0,30	1,00	0	0		00
x_2	0,24	0,70		1		01
x_3	0,20	0,46	1	0		10
x_4	0,15	0,26		1	0	110
x_5	0,11	0,11			1	111

Abychom mohli spočítat redundanci tohoto Shannonova-Fanova kódu, dopočítáme četnosti výskytu nul a jedniček v kódových znacích. Opět budeme uvažovat zprávu o délce 1 000 znaků.

Znak	Četnost znaků zprávy		Kód	Četnost znaků kódu	
	relativní	absolutní		nuly	jedničky
x_1	0,300	300	00	600	0
x_2	0,240	240	01	240	240
x_3	0,200	200	10	200	200
x_4	0,150	150	110	150	300
x_5	0,110	110	111	0	330
Zpráva	×	1 000	×	1 190	1 070

Zpráva o délce 1 000 znaků bude složena z 2 260 kódových znaků, v nichž budou mírně převažovat nuly. Spočítáme informační entropii kódu:

$$H(X) = -\left(\frac{1\,190}{2\,260} \cdot \log_2 \frac{1\,190}{2\,260} + \frac{1\,070}{2\,260} \cdot \log_2 \frac{1\,070}{2\,260}\right) = 0{,}998$$

Kód je velmi efektivní, jeho redundance dosahuje pouze 0,2 %. Střední délka kódového slova je 2,26 bitů/znak.

[40] Jedná se o metodu neztrátové komprese, kterou v roce 1949 navrhli již známý americký elektronik a matematik Claude Elwood Shannon, italsko-americký informatik Roberto Mario Fano (* 1917, † 2016) a americký matematik Warren Weaver (* 1894, † 1978). Metoda je používána například u formátu ZIP.

Huffmanovo kódování

Na podobném principu je založeno také **Huffmanovo**[41] **kódování**. Algoritmus hledání kódových znaků lze popsat těmito kroky:

1. Znaky uspořádáme sestupně podle pravděpodobnosti jejich výskytu ve zprávě.
2. Vezmeme dva znaky s nejmenší pravděpodobností a přiřadíme jim nulu a jedničku. Sečteme jejich pravděpodobnosti a výsledek (tzv. redukovaný znak) zařadíme podle velikosti mezi ostatní.
3. Předchozí krok opakujeme tak dlouho, až dojdeme u dvou nejmenších hodnot pravděpodobnosti k součtu 1.
4. Hodnoty kódových znaků získáme zpětným postupem a „sbíráním“ nul a jedniček.

Příklad 5.12

Mějme zprávu složenou ze znaků x_1, x_2, ..., x_5, které se vyskytují s pravděpodobnostmi $p(x_1) = 0{,}3$, $p(x_2) = 0{,}24$, $p(x_3) = 0{,}2$, $p(x_4) = 0{,}15$ a $p(x_5) = 0{,}11$. Sestrojíme kód podle Huffmanova algoritmu. Postup hledání kódových znaků je patrný z následujícího schématu:

x_1	0,30	
x_2	0,24	
x_3	0,20	
x_4	0,15	1 ↗
x_5	0,11	0

x_1	0,30	
x_{45}	0,26	
x_2	0,24	1 ↗
x_3	0,20	0

x_{23}	0,44	
x_1	0,30	1 ↗
x_{45}	0,26	0

x_{145}	0,56	1
x_{23}	0,44	0

Kódy jednotlivých znaků zpětně odečteme, zapíšeme do tabulky a spočítáme informační entropii a redundanci kódu. Opět budeme uvažovat zprávu o délce 1 000 znaků.

Znak	Četnost znaků zprávy		Kód	Četnost znaků kódu	
	relativní	absolutní		nuly	jedničky
x_1	0,300	300	11	0	600
x_2	0,240	240	01	240	240
x_3	0,200	200	00	400	0
x_4	0,150	150	101	150	300
x_5	0,110	110	100	220	110
Zpráva	×	1 000	×	1 010	1 250

Zpráva o délce 1 000 znaků bude složena z 2 260 kódových znaků, v nichž budou mírně převažovat jedničky. Spočítáme informační entropii kódu:

$$H(X) = -\left(\frac{1\,010}{2\,260} \cdot \log_2 \frac{1\,010}{2\,260} + \frac{1\,250}{2\,260} \cdot \log_2 \frac{1\,250}{2\,260}\right) = 0{,}992$$

Vidíme, že v tomto případě má Huffmanův kód mírně horší efektivitu, redundance dosahuje 0,8 %. Střední délka kódového slova je však stejná jako při použití Shannonova-Fanova kódu, a to 2,26 bitů/znak.

[41] Metodu vynalezl v roce 1951 americký informatik David Albert Huffman (* 1925, † 1999), Fanův student na MIT. Jedná se rovněž o metodu neztrátové komprese, která se ale paradoxně používá v kombinaci s jinými algoritmy ztrátové komprese například ve formátech JPEG, MP3 nebo WMA.

Příklad 5.13

Zkusme konstrukci Huffmanova kódu podle zadání z příkladu 5.8 s tím, že nebudeme omezovat délku kódového slova. Po provedení všech redukcí a sesbírání dílčích výsledků dostáváme následující kód:

Znak	Četnost znaků zprávy		Kód	Četnost znaků kódu	
	relativní	absolutní		nuly	jedničky
A	0,150	150	110	150	300
B	0,150	150	101	150	300
C	0,100	100	100	200	100
D	0,100	100	1111	0	400
E	0,100	100	1110	100	300
F	0,100	100	001	200	100
G	0,100	100	000	300	0
H	0,100	100	011	100	200
I	0,050	50	0101	100	100
J	0,050	50	0100	150	50
Zpráva	×	1 000	×	1 450	1 850

Zpráva o délce 1 000 znaků bude složena z 3 300 kódových znaků, v nichž budou mírně převažovat jedničky. Spočítáme informační entropii kódu:

$$H(X) = -\left(\frac{1\,450}{3\,300} \cdot \log_2 \frac{1\,450}{3\,300} + \frac{1\,850}{3\,300} \cdot \log_2 \frac{1\,850}{3\,300}\right) = 0{,}989$$

Kód je velmi efektivní s redundancí pouze 1,1 %. Střední délka kódového slova je 3,3 bitů/znak.[42] Toto řešení není jediné možné, redukované znaky měly často stejnou pravděpodobnost jako ostatní znaky v kódu a šlo je tedy zařadit více způsoby.

Původ redundance

Ve většině případů se nepodaří najít optimální kód s nulovou redundancí. Důvodem je (zejména u Shannonova-Fanova kódování) technický problém půlení intervalu na dva stejné díly. Můžeme však redundanci minimalizovat například tím, že interval půlíme stále stejným způsobem, tj. horní polovině přiřazujeme stále nulu, spodní polovině jedničku (či naopak). Huffmanovo kódování také není jednoznačné, záleží na tom, kam umístíme redukované znaky v případě, kdy má více znaků stejnou pravděpodobnost výskytu. Obecně ale vede Huffmanovo kódování k lepším výsledkům než Shannonovo-Fanovo kódování.

Při zpracování dat však může být redundance i výhodná, protože zpracování urychluje. S odlišnou formou redundance, která vzniká přidáním zabezpečovacích prvků (hesel, kontrolních součtů apod.), se seznámíme v následující kapitole.

Vraťme se ještě na chvíli k reprezentaci textových dat, kterým jsme se věnovali v kap. 4.2. Nyní již umíme zjistit efektivnost kódu výpočtem entropie, a tak nám nic nebrání aplikovat tyto poznatky i na znakové kódy. Zaměříme se na jednobajtový kód ASCII s rozšířením ISO 8859-2 a vícebajtové kódy UTF-8 a UTF-16 ve variantě Big Endian.

[42] Pozornému čtenáři neunikla zajímavá vlastnost Huffmanova kódu. Když sečteme pravděpodobnosti všech redukovaných znaků a přičteme 1, dostaneme střední délku kódového slova.

Příklad 5.14

Uvažujme úryvek ze Starých pověstí českých:

Když zemřel kníže Křesomysl, vystřídal ho na trůně na Vyšehradě Neklan. Ten vládl českému lidu moudře a mírumilovně. S Litoměřici a s Lemuzy žil v míru a pokoji a vojvodové těchto krajů si ho vážili. Na severu a na západě měl však zlého souseda, vojvodu Vlastislava, který vládl v kraji hrdých Lučanů podél řeky Oharky a horní Mže. Neklan byl mírný a dobrosrdečný, Vlastislav divoký a vášnivý. Neklan bázlivého srdce, Vlastislav bojechtivý. Nic nedbal dobré sousedské vůle ani práva ani spravedlnosti, krev proléval bez milosrdenství. Své sousedy krutě utiskoval a často se svou divokou družinou táhl i do země Lemuzů, Litoměřiců a Čechů, kde vybíral daň, která mu nepatřila.

Úryvek se skládá z 675 znaků, z toho je 108 mezer a 17 interpunkčních znaků. Provedeme frekvenční analýzu, zjistíme četnosti jednotlivých znaků v textu a zaneseme je do tabulky spolu s binární reprezentací znaků ve všech třech kódech s vyznačenými rozlišovacími bity. Pro srovnání jsou v tabulce uvedeny i tzv. referenční četnosti získané z dat Centra zpracování přirozeného jazyka Masarykovy univerzity v Brně. Aby bylo možné provést srovnání, jsou relativní četnosti znaků v úryvku vztaženy k celkovému počtu písmenných znaků (bez mezer a interpunkce).

Znak	Absolutní četnost (úryvek)	Relativní četnost (úryvek)	Relativní četnost (referenční)	Kód znaku ISO 8859-2	Kód znaku UTF-8	Kód znaku UTF-16 BE
a	45	0,0818	0,0670	01100001	01100001	00000000 01100001
A	0			01000001	01000001	00000000 01000001
á	9	0,0164	0,0213	11100001	11000011 10100001	00000000 11100001
Á	0			11000001	11000011 10000001	00000000 11000001
b	8	0,0145	0,0167	01100010	01100010	00000000 01100010
B	0			01000010	01000010	00000000 01000010
c	8	0,0145	0,0261	01100011	01100011	00000000 01100011
C	0			01000011	01000011	00000000 01000011
č	4	0,0091	0,0102	11101000	11000100 10001101	00000001 00001101
Č	1			11001000	11000100 10001100	00000001 00001100
d	28	0,0509	0,0361	01100100	01100100	00000000 01100100
D	0			01000100	01000100	00000000 01000100
ď	0	0,0000	0,0002	11101111	11000100 10001111	00000001 00001111
Ď	0			11001000	11000100 10001110	00000001 00001110
e	36	0,0655	0,0783	01100101	01100101	00000000 01100101
E	0			01000101	01000101	00000000 01000101
é	9	0,0164	0,0118	11101001	11000011 10101001	00000000 11101001
É	0			11001001	11000011 10001001	00000000 11001001
ě	10	0,0182	0,0149	11101100	11000100 10011011	00000001 00011011
Ě	0			11001100	11000100 10011010	00000001 00011010
f	0	0,0000	0,0039	01100110	01100110	00000000 01100110
F	0			01000110	01000110	00000000 01000110
g	0	0,0000	0,0034	01100111	01100111	00000000 01100111
G	0			01000111	01000111	00000000 01000111
h	13	0,0236	0,0230	01101000	01101000	00000000 01101000
H	0			01001000	01001000	00000000 01001000
i	30	0,0545	0,0457	01101001	01101001	00000000 01101001
I	0			01001001	01001001	00000000 01001001
í	8	0,0145	0,0310	11101101	11000011 10101101	00000000 11101101
Í	0			11001101	11000011 10001101	00000000 11001101
j	6	0,0109	0,0198	01101010	01101010	00000000 01101010
J	0			01001010	01001010	00000000 01001010
k	20	0,0400	0,0375	01101011	01101011	00000000 01101011
K	2			01001011	01001011	00000000 01001011

(pokračování na další straně)

(pokračování z předchozí strany)

Znak	Absolutní četnost (úryvek)	Relativní četnost (úryvek)	Relativní četnost (referenční)	Kód znaku ISO 8859-2	Kód znaku UTF-8	Kód znaku UTF-16 BE
l L	35 5	0,0727	0,0410	01101100 01001100	01101100 01001100	00000000 01101100 00000000 01001100
m M	16 1	0,0309	0,0326	01101101 01001101	01101101 01001101	00000000 01101101 00000000 01001101
n N	22 5	0,0491	0,0668	01101110 01001110	01101110 01001110	00000000 01101110 00000000 01001110
ň Ň	1 0	0,0018	0,0007	11110010 11010010	11000101 10001000 11000101 10000111	00000001 01001000 00000001 01000111
o O	37 1	0,0691	0,0828	01101111 01001111	01101111 01001111	00000000 01101111 00000000 01001111
ó Ó	0 0	0,0000	0,0003	11110011 11010011	11000011 10110011 11000011 10010011	00000000 11110011 00000000 11010011
p P	7 0	0,0127	0,0345	01110000 01010000	01110000 01010000	00000000 01110000 00000000 01010000
q Q	0 0	0,0000	0,0001	01110001 01010001	01110001 01010001	00000000 01110001 00000000 01010001
r R	25 0	0,0455	0,0398	01110010 01010010	01110010 01010010	00000000 01110010 00000000 01010010
ř Ř	8 0	0,0145	0,0119	11111000 11011000	11000101 10011001 11000101 10011000	00000001 01011001 00000001 01011000
s S	30 2	0,0582	0,0462	01110011 01010011	01110011 01010011	00000000 01110011 00000000 01010011
š Š	3 0	0,0055	0,0082	10111001 10101001	11000101 10100001 11000101 10100000	00000001 01100001 00000001 01100000
t T	19 1	0,0345	0,0555	01110100 01010100	01110100 01010100	00000000 01110100 00000000 01010100
ť Ť	0 0	0,0000	0,0004	10111011 10101011	11000101 10100101 11000101 10100100	00000001 01100101 00000001 01100100
u U	20 0	0,0364	0,0313	01110101 01010101	01110101 01010101	00000000 01110101 00000000 01010101
ú Ú	0 0	0,0000	0,0015	11111010 11011010	11000011 10111010 11000011 10011010	00000000 11111010 00000000 11011010
ů Ů	7 0	0,0127	0,0057	11111001 11011001	11000101 10101111 11000101 10101110	00000001 01101111 00000001 01101110
v V	33 4	0,0673	0,0438	01110110 01010110	01110110 01010110	00000000 01110110 00000000 01010110
w W	0 0	0,0000	0,0007	01110111 01010111	01110111 01010111	00000000 01110111 00000000 01010111
x X	0 0	0,0000	0,0009	01111000 01011000	01111000 01011000	00000000 01111000 00000000 01011000
y Y	10 0	0,0182	0,0175	01111001 01011001	01111001 01011001	00000000 01111001 00000000 01011001
ý Ý	7 0	0,0127	0,0094	11111101 11011101	11000011 10111101 11000011 10011101	00000000 11111101 00000000 11011101
z Z	8 0	0,0145	0,0212	01111010 01011010	01111010 01011010	00000000 01111010 00000000 01011010
ž Ž	6 0	0,0109	0,0102	10111110 10101110	11000101 10111110 11000101 10111101	00000001 01111110 00000001 01111101
mezera	108	×	×	00100000	00100000	00000000 00100000
tečka	8	×	×	00101110	00101110	00000000 00101110
čárka	9	×	×	00101100	00101100	00000000 00101100

Znaky vyskytující se v úryvku uspořádáme podle četnosti a umístíme do tabulky:

Znak	Četnost znaků v úryvku		Četnost znaků kódu ISO 8859-2		Četnost znaků kódu UTF-8		Četnost znaků kódu UTF-16 BE	
	relativní	absolutní	nuly	jedničky	nuly	jedničky	nuly	jedničky
mezera	0,1600	108	756	108	756	108	1 620	108
a	0,0667	45	225	135	225	135	585	135
o	0,0548	37	74	222	74	222	370	222
e	0,0533	36	144	144	144	144	432	144
l	0,0519	35	140	140	140	140	420	140
v	0,0489	33	99	165	99	165	363	165
i	0,0444	30	120	120	120	120	360	120
s	0,0444	30	90	150	90	150	330	150
d	0,0415	28	140	84	140	84	364	84
r	0,0370	25	100	100	100	100	300	100
n	0,0326	22	66	110	66	110	242	110
k	0,0296	20	60	100	60	100	220	100
u	0,0296	20	60	100	60	100	220	100
t	0,0281	19	76	76	76	76	228	76
m	0,0237	16	48	80	48	80	176	80
h	0,0193	13	65	39	65	39	169	39
ě	0,0148	10	30	50	80	80	110	50
y	0,0148	10	30	50	30	50	110	50
á	0,0133	9	36	36	81	63	108	36
é	0,0133	9	27	45	72	72	99	45
čárka	0,0133	9	45	27	45	27	117	27
b	0,0119	8	40	24	40	24	104	24
c	0,0119	8	32	32	32	32	96	32
í	0,0119	8	16	48	56	72	80	48
ř	0,0119	8	24	40	64	64	88	40
z	0,0119	8	24	40	24	40	88	40
tečka	0,0119	8	32	32	32	32	96	32
p	0,0104	7	35	21	35	21	91	21
ů	0,0104	7	14	42	42	70	63	49
ý	0,0104	7	7	49	42	70	63	49
j	0,0089	6	24	24	24	24	72	24
ž	0,0089	6	12	36	36	60	54	42
L	0,0074	5	25	15	25	15	65	15
N	0,0074	5	20	20	20	20	60	20
č	0,0059	4	16	16	36	28	48	16
V	0,0059	4	16	16	16	16	48	16
š	0,0044	3	9	15	27	21	36	12
K	0,0030	2	8	8	8	8	24	8
S	0,0030	2	8	8	8	8	24	8
Č	0,0015	1	5	3	10	6	13	3
M	0,0015	1	4	4	4	4	12	4
ň	0,0015	1	3	5	10	6	13	3
O	0,0015	1	3	5	3	5	11	5
T	0,0015	1	5	3	5	3	13	3
Zpráva		675	2 813	2 587	3 170	2 814	8 205	2 595
UTF signatura		1	0	0	4	20	1	15

Použijeme-li znakový kód ASCII s rozšířením ISO 8859-2, úryvek českého textu složený z 675 znaků bude v paměti zabírat 675 bajtů, tj. 5 400 bitů, z nichž 2 813 bitů bude mít hodnotu 0 a 2 587 bitů bude mít hodnotu 1. Když tyto charakteristiky dosadíme do vztahu pro výpočet entropie, dostáváme

$$H(X_1) = -\left(\frac{2\,813}{5\,400} \cdot \log_2 \frac{2\,813}{5\,400} + \frac{2\,587}{5\,400} \cdot \log_2 \frac{2\,587}{5\,400}\right) = 0{,}999.$$

Při použití vícebajtového znakového kódu UTF-8 bude stejný úryvek v paměti zabírat 751 bajtů, tj. 6 008 bitů (uvažujeme přítomnost 24bitové signatury), z nichž 3 174 bitů bude mít hodnotu 0 a 2 834 bitů bude mít hodnotu 1. Po dosazení dostáváme

$$H(X_2) = -\left(\frac{3\,174}{6\,008} \cdot \log_2 \frac{3\,174}{6\,008} + \frac{2\,834}{6\,008} \cdot \log_2 \frac{2\,834}{6\,008}\right) = 0{,}998.$$

U dvoubajtového znakového kódu UTF-16 ve variantě Big Endian bude stejný úryvek zabírat v paměti 1 352 bajtů, tj. 10 816 bitů (uvažujeme přítomnost 16bitové signatury), z nichž 8 206 bitů bude mít hodnotu 0 a 2 610 bitů bude mít hodnotu 1. Po dosazení dostáváme

$$H(X_3) = -\left(\frac{8\,206}{10\,816} \cdot \log_2 \frac{8\,206}{10\,816} + \frac{2\,610}{10\,816} \cdot \log_2 \frac{2\,610}{10\,816}\right) = 0{,}797.$$

Výsledky předchozího příkladu potvrzují skutečnost, že jednobajtové znakové kódy jsou mimořádně úsporné. Víme ale, že jejich velkým problémem je nejednotnost. Z vícebajtových kódů dosahuje velmi podobných kvalit nerovnoměrný kód UTF-8, zatímco rovnoměrný dvoubajtový kód UTF-16 vykazuje velkou nehospodárnost.[43]

Kontrolní otázky

4. Co vyjadřuje redundance kódu a jak ji můžeme spočítat?
5. Jaký je rozdíl mezi blokovým a prefixovým kódem?
6. Které z kódů ISO 8859-2, UTF-8 a UTF-16 jsou blokové a které nerovnoměrné?
7. Jak u blokového kódování zjistíme délku kódových znaků?
8. Jak ověříme, zda lze sestrojit prefixový kód s požadovanými vlastnostmi?
9. Za jakých okolností je blokový kód hospodárnější než nerovnoměrný?
10. Jakou redundanci a entropii by měl mít optimální kód?

[43] Na tomto místě je nutné podotknout, že vypočtené charakteristiky jsou pouze orientační. Kdybychom totiž chtěli spočítat skutečnou entropii (a z ní odvozenou redundanci) znakového kódu, museli bychom uvažovat všechny znaky, které lze v daném kódu reprezentovat. V každém jazyku se však vyskytuje jiná, relativně malá podmnožina znaků. V rámci celého kódu proto četnosti jednotlivých znaků nemohou být konstantní a lze tak spočítat entropii pouze pro konkrétní jazyk s konkrétní podmnožinou národních znaků.

5.3 Zabezpečení dat při přenosu

Rušení

Přenosu uložených dat do místa jejich zpracování se prakticky nikdy nelze vyhnout – ať už se jedná o přenos z disku do operační paměti počítače nebo přenosy dat v rámci počítačové sítě. V průběhu přenosu se může objevit různé *rušení*, které způsobí, že data nesoucí informaci nemusí být doručena ve stejné podobě, ve které byla odeslána. Veškeré rušení můžeme rozdělit do dvou kategorií:

1. rušení *nezáměrné*, které plyne z technické nedokonalosti přenosového kanálu;
2. rušení *záměrné*, které je způsobeno třetími osobami ve snaze získat nebo dokonce modifikovat přenášenou informaci.

Zabezpečením tedy lze chápat nejen mechanismy umožňující na přijímací straně zjistit nebo i opravit chybu vzniklou při přenosu, ale také mechanismy zabraňující přečtení a modifikaci informace neoprávněnou osobou.

V každém případě se zabezpečením zvětšuje objem dat, aniž by se zvětšil objem vlastní užitečné informace, z tohoto hlediska je tedy zabezpečení chápáno jako redundance.

5.3.1 Zabezpečení proti technické nedokonalosti přenosu

Chyba

Během přenosu dochází k útlumu signálu vlivem působení vnějších technických vlivů (atmosférické poruchy, činnost jiných strojů a zařízení apod.). Uvažujeme-li přenosy dat složené z jedniček a nul, pod pojmem **chyba** budeme chápat záměnu binární jedničky za nulu, nebo opačně. Takovou chybu můžeme jednoduše opravit, stačí provést inverzi příslušného bitu. K tomu ale potřebujeme nejdříve konkrétní bit lokalizovat, což už tak jednoduché není.

Při opakovaných přenosech můžeme zjišťovat, za jak dlouho chyba nastala. **Četnost chyb** je relativní a u každého zařízení jiná. Některá zařízení mohou pracovat správně přesto, že četnost chyb se pohybuje řádově ve stovkách za hodinu, jiné zařízení je již při výskytu jedné chyby za den chápáno jako nespolehlivé.

Při přenosu určitého bloku dat můžeme také zjišťovat, kolik chyb se v něm vyskytlo. Hovoříme pak o **násobnosti chyby**. Obvykle platí, že pravděpodobnost výskytu jednonásobné chyby je řádově větší než pravděpodobnost výskytu dvojnásobné chyby, ta je opět řádově větší než u výskytu trojnásobné chyby atd. Proto se zabýváme pouze zabezpečením jednonásobných chyb, vícenásobné chyby se zabezpečovat prakticky nevyplatí.

Detekce a oprava chyby

V jednodušších případech stačí, budeme-li pouze vědět, že při přenosu nastala nějaká chyba – použijeme některou z metod **detekce** chyby. Pak musíme být buď schopni data zpracovat s chybami, nebo musíme vyžádat opakovaný přenos téhož úseku dat. Ve složitějších případech musíme zvolit metody pro **opravu** chyby, které dokáží chybu nejen lokalizovat, ale následně inverzí příslušného bitu také opravit.

Detekční kódy

Jednoduchá parita

Mezi typické metody detekce jednonásobných chyb patří zabezpečení **paritou**. Princip spočívá v tom, že ke každému úseku dat (například ke každému bajtu) je přidán další bit tak, aby celkový počet jedničkových bitů byl vždy sudý (sudá parita), nebo lichý (lichá parita).

Zabezpečení devátým bitem se dříve užíval při ukládání informace do operační paměti osobních počítačů. Detekce chyby dat v paměti je považováno za fatální chybu a projeví se chybovým hlášením. Data po Internetu se přenáší zabezpečená paritou v osmém bitu – samotná data jsou potom sedmibitová.

Příklad 5.15

Předpokládejme úsek dat ve tvaru 11010110 zabezpečený jednoduchou lichou paritou v nejvyšším bitu. Ukažme si, jak se při přenosu projeví výskyt chyby s lichou a sudou násobností:

před přenosem	po přenosu	násobnost chyb	paritní bit	porovnání
11010110	11010110	0	1	OK
11010110	11010010	1	0	detekce chyby
11010110	11000100	2	1	OK
11010110	10111110	3	0	detekce chyby
11010110	10000000	4	1	OK

Jak je zřejmé z příkladu, zabezpečení paritou je schopno odhalit přítomnost jednonásobné chyby. Vyskytne-li se chyba dvojnásobná, výpočet bitu parity na přijímací straně dává stejný výsledek jako na odesílací straně, proto se přenos mylně bere za bezchybný. Obecně pro zabezpečení paritou platí, že chyby s lichou násobností jsou detekovány, chyby se sudou násobností nikoli.

Kombinace parit

Nestačí-li jednoduchá detekce jedním paritním bitem, můžeme obecně zavést libovolný počet paritních bitů zabezpečujících různé kombinace datových bitů. Tím se zvýší schopnost detekce chyb, dokonce při dostatečném počtu zabezpečovacích bitů budeme schopni chybu nejen detekovat, ale i lokalizovat, a tedy i opravit.

Příklad 5.16

Uvažujme úsek dat 10101101 zabezpečený kombinovanou sudou paritou v nejnižších bitech takovým způsobem, že první paritní bit bude pokrývat první čtveřici datových bitů, druhý paritní bit druhou čtveřici paritních bitů, třetí paritní bit bude pokrývat liché datové bity, čtvrtý paritní bit sudé datové bity. Ukažme si, jak při takovém zabezpečení bude možné detekovat jednonásobné a vícenásobné chyby při přenosu.

před přenosem	po přenosu	násobnost chyb	paritní bity	porovnání
10101101 0110	10101101 0110	0	0110	OK
10101101 0110	10111101 0110	1	1111	detekce chyby
10101101 0110	10101001 1110	2	0011	detekce chyby
10101101 0110	11111100 0110	3	0011	detekce chyby
10101101 0110	10011111 1110	4	0011	detekce chyby

Mohlo by se zdát, že zabezpečení kombinovanou paritou dokáže odhalit chyby libovolné násobnosti. Bohužel tomu tak není. Stačí zanést chybu na „vhodné“ místo a kontrolní výpočet může dopadnout zcela jinak. Zkusme nasimulovat výskyt jedné chyby v datových bitech, která bude zastíněna dvěma chybami v paritních bitech:

před přenosem	po přenosu	násobnost chyb	paritní bity	porovnání
10101101 0110	10111101 1111	3	1111	OK

Jedním z používaných mechanismů je kombinace jednoduché parity, nazývané v této souvislosti **příčná parita** (*transversal parity*), a tzv. **podélné parity** (*longitudinal parity*). Zabezpečovaná data se rozdělí na úseky požadované délky (obvykle 8, 16 nebo 32 bitů) a v nich se paritní bity vypočítávají za všechny stejnolehlé bity celého bloku dat. Stejnolehlé bity se sečtou bez přenosu do vyššího řádu, zapíše se tedy jen poslední číslice.[44] V případě sudého počtu jedniček je výsledkem nula, v opačném případě jednička. Vznikne tím další úsek o stejné délce, který se připojí k přenášeným datům. Podélná parita patří do skupiny zabezpečení pomocí kontrolních součtů.

Příklad 5.17

Předpokládejme úsek dat 0101010 1010110 0001010 0110101, který je zabezpečen kombinací sudé příčné parity v nejvyšším bitu a sudé podélné parity:

původní data	data s příčnou paritou	výpočet podélné parity
0101010	10101010	10101010
1010110	01010110	01010110
0001010	00001010	00001010
0110101	00110101	00110101
		11000011

Nastane-li při přenosu chyba, bude detekována příčným paritním bitem v příslušném řádku a podélným paritním bitem v příslušném sloupci. Na průsečíku řádku a sloupce tedy můžeme lokalizovat chybný bit, který lze opravit. Chyby s vyšší násobností opravit nelze, ale výhodou je, že i tyto chyby budou alespoň detekovány. Mějme na paměti, že pravděpodobnost vícenásobné chyby je relativně zanedbatelná.

Příklad 5.18

Předpokládejme přenos zabezpečeného úseku dat z předchozího příkladu v podobě 10101010 01010110 00001010 00110101 11000011. Při přenosu došlo k chybě a příjemce obdržel úsek 10101010 01011110 00001010 00110101 11000011. Poznáme, zda došlo při přenosu k chybě? A pokud ano, jsme schopni chybu lokalizovat a opravit?

před přenosem		po přenosu	kontrola parity
10101010		10101010	10101010
01010110	$\longrightarrow$	01011110	1101X110
00001010		00001010	00001010
00110101		00110101	00110101
11000011		11000011	11001011

Kontrolním výpočtem příčné a podélné parity zjistíme nesoulad v paritních bitech druhého bloku dat a v pátém sloupci. Na průsečíku řádku a sloupce se nachází chybný bit, který je třeba invertovat a tím chybu opravit.

Bohužel ale ani tato metoda není stoprocentní. Kdybychom totiž nevěděli, že došlo k jednonásobné chybě, a měli bychom tedy pouze na základě obdrženého úseku dat rozhodnout o jeho správnosti, byli bychom sice schopni lokalizovat konkrétní „chybný“ bit, ale jeho opravou bychom data znehodnotili.

[44] Výpočet odpovídá matematické operaci exkluzivní disjunkce (XOR). Tato funkce se často používá v kryptografii (viz str. 124), v nejnovějších návrzích hashovacích funkcí je jednou z nejdůležitějších operací.

Příklad 5.19

Předpokládejme podobnou situaci jako v předchozím příkladu s tím rozdílem, že příjemce obdrží úsek dat 10101010 11010110 00001010 00110101 11001011, ve kterém jsou dvě velmi nepříjemné chyby v paritních bitech, datové bity jsou bezchybné. Jak se tato skutečnost projeví při kontrole dat na straně příjemce?

před přenosem		po přenosu	kontrola parity
10101010		10101010	10101010
01010110	$\longrightarrow$	11010110	1101X110
00001010		00001010	00001010
00110101		00110101	00110101
11000011		11001011	11001011

Je zřejmé, že při přenosu nastala dvojnásobná chyba na pozici paritního bitu příslušného řádku a sloupce, v důsledku toho by pak onen bit byl považován za chybný, přestože jeho hodnota je správná. Naštěstí, jak už dobře víme, pravděpodobnost dvojnásobné chyby je výrazně menší než pravděpodobnost jednonásobné chyby a pravděpodobnost výskytu dvojnásobné chyby v paritních bitech je proto zanedbatelná.

Kontrolní součet

Zatímco zabezpečení paritou je velmi jednoduché a rychlé, ale nedetekuje někdy dostatečný počet chyb, zabezpečení **kontrolním součtem** (*checksum*) je poněkud komplikovanější, zato obvykle s vyšší schopností detekovat chyby. Mezi kontrolní součty patří:

- *Podélná parita (longitudinal parity)* – součet bloku dat bez přenosů do vyšších řádů, při kterém pro každý řád zabezpečovaných dat dostáváme jeden zabezpečovací bit.
- *Cyklický součet (Cyclic Redundancy Check, CRC)* – často používaný systém zabezpečení, jehož výpočet je založen na zjištění zbytku po dělení dvou binárních polynomů. Jeden polynom je představován zabezpečovanými daty, druhý je vhodně zvolen. Podle úrovně zabezpečení se vypočítávají CRC o šířce 8 bitů, 16 bitů nebo 32 bitů.
- *Otisk (hash)* – složitější algoritmy výpočtu zabezpečovací informace s dalšími možnostmi. Do této kategorie lze zařadit také jednosměrné šifrovací metody SHA (*Secure Hash Algorithm*), MD5 (*Message-Digest Algorithm 5*) apod. (viz str. 125).
- *Další speciální případy* – výpočty kontrolních číslic, například ISBN[45] u knihy, identifikační číslo podnikatelských subjektů[46], číslo účtu v bance[47], VIN, označení lokomotiv Českých drah[48] nebo u rodných čísel[49].

[45] ISBN obsahuje informace o typu publikace, jazykovém regionu, vydavatelství a číslu publikace v rámci vydavatelství. V kódu ISBN-13 se kontrolní číslice zvolí tak, aby zbytek po dělení váženého součtu všech 13 číslic deseti byl nulový, přičemž jako váhy se používají střídavě hodnoty 1 a 3.

[46] Jednotlivé číslice se násobí sestupnými vahami od 8 po 2, ze součtu dílčích součinů se vypočte zbytek po dělení 11 a ten se odečte od 11. Kontrolní číslici pak tvoří zbytek po dělení výsledku deseti.

[47] Sestupnými vahami od 15 do 1 umocníme číslo 2 a určíme zbytek po dělení 11. Získanými zbytky vynásobíme postupně jednotlivé číslice, dílčí součiny sečteme a dále postupujeme stejným způsobem jako u identifikačního čísla podnikatelských subjektů.

[48] Ze šesti číslic označení lokomotivy se vypočte kontrolní číslice tak, že číslice na lichých místech směrem odzadu se násobí dvěma a ciferné součty výsledků se sečtou, číslice na sudých místech směrem odzadu se sečtou a přičtou k předchozímu součtu, vzniklý součet se potom odečte od nejbližší vyšší desítky.

[49] K šesti číslicím vyjadřujícím den, měsíc a rok narození jsou u osob narozených počínaje 1. 1. 1954 přidávány čtyři číslice za lomítkem tak, aby celé rodné číslo bylo beze zbytku dělitelné 11. Z technických důvodů se nepřidělují čísla, jejichž zbytek po dělení 11 je 10. Toto pravidlo bylo ustáleno v roce 1985.

Příklady použití kontrolních číslic

Příklad 5.20

Ověříme, zda ISBN těchto skript (978-80-7302-189-4) je platné.

	Prefix			Skupina		Vydavatel				Vydání			Kontrola
ISBN	9	7	8	8	0	7	3	0	2	1	8	9	4
Váhy	1	3	1	3	1	3	1	3	1	3	1	3	–
Součiny	9	21	8	24	0	21	3	0	2	3	8	27	–

Součet součinů je 126 a protože (10 – 126 mod 10) mod 10 = 4, je toto ISBN platné.

Příklad 5.21

Ověříme, zda číslo bankovního účtu 7200300237 je platné.

Váhy (v)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	–
2^v mod 11	10	5	8	4	2	1	6	3	7	9	10	5	8	4	2	–
Číslo účtu	0	0	0	0	0	0	7	2	0	0	3	0	0	2	3	7
Součiny	0	0	0	0	0	0	42	6	0	0	30	0	0	8	6	–

Součet součinů je 92 a protože (11 – 92 mod 11) mod 10 = 7, je toto číslo účtu platné.

Příklad 5.22

Dopravní podnik města Brna má od roku 2016 ve flotile zájezdový autobus Volvo 9700, kterému bylo přiděleno evid. č. 2901. Ověříme, zda VIN (identifikační číslo) tohoto autobusu ve tvaru YV3T2U82XG1179402 je platné.

S kódováním VIN je to poněkud složitější. Nejdříve musíme provést záměnu písmen za čísla podle následující tabulky:

A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	1	2	3	4	5	7	9	2	3	4	5	6	7	8	9

Dále vynásobíme čísla na jednotlivých pozicích kódu VIN vahami podle následující tabulky (kontrolní číslice se nachází na deváté pozici):

Pozice	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Váha	8	7	6	5	4	3	2	10	–	9	8	7	6	5	4	3	2

Nyní můžeme přejít ke kontrole správnosti čísla:

Kód VIN	Y	V	3	T	2	U	8	2	X	G	1	1	7	9	4	0	2
Číselně	8	5	3	3	2	4	8	2	–	7	1	1	7	9	4	0	2
Váha	8	7	6	5	4	3	2	10	–	9	8	7	6	5	4	3	2
Součiny	64	35	18	15	8	12	16	20	–	63	8	7	42	45	16	0	4

Součet součinů je 373 a protože 373 mod 11 = 10, je tento VIN platný. Pokud je zbytek po dělení roven 10, zapisuje se místo kontrolní číslice znak X.

Příklad 5.23

Ověříme, zda IČO 47915391 je platné.

IČO	4	7	9	1	5	3	9	1
Váhy	8	7	6	5	4	3	2	–
Součiny	32	49	54	5	20	9	18	–

Součet součinů je 187 a protože (11 – 187 mod 11) mod 10 = 1, je toto IČO platné.

Příklad 5.24

Ověříme, zda rodná čísla 600730/0651, 780123/3540 a 906010/7892 jsou platná.

U rodných čísel přidělovaných v období 1954–1985 je nutno spočítat zbytek po dělení prvních devíti číslic rodného čísla a čísla 11. Je-li tento zbytek 10, pak poslední číslice je 0, jinak je poslední číslice rovna zbytku po dělení.

Protože 600730065 mod 11 = 1, poslední číslice musí být 1. První rodné číslo je platné.
Protože 780123354 mod 11 = 10, poslední číslice musí být 0. Druhé rodné číslo je platné.

U rodných čísel přidělovaných po roce 1985 stačí pouze ověřit dělitelnost celého čísla 11. Kontrolu lze provést i zpaměti, stačí sečíst číslice na lichých a sudých místech a ověřit, zda rozdíl součtů je dělitelný 11.

Pro třetí rodné číslo dostáváme (32 – 10) mod 11 = 0, tedy rodné číslo je platné.

Příklad 5.25

Ověříme, zda motorová lokomotiva řady 751 058-9 (do roku 1988 vedena pod označením T478.1058), tzv. „Bardotka“, má platné evidenční číslo.

Číslo lokomotivy	7	5	1	0	5	8	9
Váhy	1	2	1	2	1	2	–
Součiny	7	10	1	0	5	16	–
Ciferné součty	7	1	1	0	5	7	–

Součet ciferných součtů je 21, po odečtení součtu od nejbližší vyšší desítky dostáváme 30 – 21 = 9. Evidenční číslo lokomotivy je tedy platné.

Samoopravné kódy

Potřebujeme-li chybu na přijímací straně opravit (nelze například připustit opakované posílání chybných dat) a nestačí nám k tomu kombinace příčné a podélné parity, musíme použít jiný způsob zabezpečení.

Opakovací kód

Nejjednodušší cestou je posílat stejná data opakovaně – například třikrát, na přijímací straně se vyhodnotí shodnost přijatých trojic. K tomu slouží **opakovací kód**. Je-li ze tří přijatých hodnot alespoň dvojice shodná, bere se jako správná hodnota. Jsou-li všechny tři hodnoty různé, jedná se o vícenásobnou chybu, kterou nelze opravit, ale pouze detekovat. Kód může mít více variant, které se liší délkou posloupnosti a počtem opakování.

Příklad 5.26

Předpokládejme, že chceme přenášet znak 1, který zabezpečíme opakovacím kódem tak, že jej pošleme jako 111. Podívejme se, jak se projeví chyba a jak ji můžeme opravit na straně příjemce.

znak	odeslané slovo	přijaté slovo	násobnost chyb	porovnání	oprava
1	111	111	0	OK	1
1	111	110	1	detekce chyby	1
1	111	101	1	detekce chyby	1
1	111	011	1	detekce chyby	1
1	111	001	2	detekce chyby	0
1	111	010	2	detekce chyby	0
1	111	100	2	detekce chyby	0
1	111	000	3	OK	0

Je zřejmé, že jednonásobnou chybu lze opakovacím kódem s trojznakovými kódovými slovy detekovat i opravit. Později odvodíme vztah mezi požadovanou schopností opravy násobných chyb a nutným počtem kontrolních prvků v kódových slovech.

Hammingovy kódy

Řešení problému opravy chyb přináší **Hammingovy**[50] **kódy**. Vysílaná data jsou kódována do speciálně vybraných posloupností, tzv. *povolené kombinace*. Dojde-li při přenosu k chybě, příjemce obdrží posloupnost, která nepatří mezi povolené, tzv. *zakázanou kombinaci*. Kódy jsou konstruovány tak, že u každé zakázané kombinace lze vypočítat takovou povolenou kombinaci, která je k ní nejblíže (předpokládá se nejnižší násobnost chyby).

Hammingovy kódy (n, k) patří do třídy *lineárních binárních kódů*. Hodnoty v závorce určují délku bloku (počet bitů kódového slova) a délku zprávy (počet datových bitů, které jsou ve slově obsaženy). Podle počtu paritních bitů $c \geq 2$ můžeme vytvářet kódová slova s délkou bloku $n = 2^c - 1$ a délkou zprávy $k = 2^c - c - 1$. *Redundanci* bezpečnostního kódu vypočítáme podle vztahu $R = 1 - k/n$, kde k/n vyjadřuje *informační poměr kódu*.

Příklad 5.27

Podívejme se na různé varianty Hammingových kódů v závislosti na počtu paritních bitů a dosažených hodnot redundance. Hledáme řešení rovnice $2^c = c + k + 1$, kde c je daný počet kontrolních (paritních) bitů a k je hledaný počet informačních (datových) bitů.

paritních bitů	celkem bitů	datových bitů	označení kódu	redundance kódu
2	3	1	Hamming $(3, 1)$[51]	$1 - 1/3 = 0{,}667$
3	7	4	Hamming $(7, 4)$	$1 - 4/7 = 0{,}429$
4	15	11	Hamming $(15, 11)$	$1 - 11/15 = 0{,}267$
5	31	26	Hamming $(31, 26)$	$1 - 26/31 = 0{,}161$
6	63	57	Hamming $(63, 57)$	$1 - 57/63 = 0{,}095$
7	127	120	Hamming $(127, 120)$	$1 - 120/127 = 0{,}055$
8	255	247	Hamming $(255, 247)$	$1 - 247/255 = 0{,}031$
9	511	502	Hamming $(511, 502)$	$1 - 502/511 = 0{,}018$

[50] Richard Wesley Hamming (* 1915, † 1998) byl americký matematik, jehož práce má mimořádný význam v oblasti informatiky a telekomunikací. Zabýval se teorií informace, kódováním a obzvláště samoopravnými kódy. Pracoval v Bellových laboratořích a vyučoval na Stanfordově i Princetonské univerzitě.

Kódová slova v Hammingových kódech jsou generována následujícími pravidly:

1. Všechny bitové pozice, jejichž číslo je rovno mocnině 2, jsou použity pro paritní bit.
2. Všechny ostatní bitové pozice náleží datovým bitům přenášeného slova.
3. Každý paritní bit je vypočítán z některých datových bitů. Pozice paritního bitu udává sekvenci bitů, které jsou v kódovém slově zjišťovány a které přeskočeny.

Tabulka 5.1: *Pozice datových a paritních bitů v kódových slovech Hammingových kódů*

Pozice bitů		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Kódované bity		p_1	p_2	b_1	p_3	b_2	b_3	b_4	p_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}
Pokrytí paritním bitem	p_1	✓		✓		✓		✓		✓		✓		✓		✓
	p_2		✓	✓			✓	✓			✓	✓			✓	✓
	p_3				✓	✓	✓	✓					✓	✓	✓	✓
	p_4								✓	✓	✓	✓	✓	✓	✓	✓

Tabulka 5.1 znázorňuje pozice datových (b_1–b_{11}) a paritních (p_1–p_4) bitů v kódových slovech o délce do 15 bitů, tedy pro Hammingův kód (15, 11). Pro delší kódová slova postupujeme analogicky dle výše uvedených pravidel. Podíváme-li se do tabulky pozorně a v podbarvených buňkách si představíme 1, zjistíme, že v jednotlivých sloupcích tabulky není nic jiného, než binárně zakódované pozice jednotlivých bitů 1–15 z kódového slova.

Hodnotu paritního bitu získáme výpočtem exkluzivní disjunkce (xor) nad zbývajícími vyznačenými bity kódového slova:

$$\begin{aligned} p_1 &= b_1 \oplus b_2 \oplus b_4 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} \\ p_2 &= b_1 \oplus b_3 \oplus b_4 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11} \\ p_3 &= b_2 \oplus b_3 \oplus b_4 \oplus b_8 \oplus b_9 \oplus b_{10} \oplus b_{11} \\ p_4 &= b_5 \oplus b_6 \oplus b_7 \oplus b_8 \oplus b_9 \oplus b_{10} \oplus b_{11} \end{aligned}$$

Příklad 5.28

Čtyřbitová data zabezpečíme samoopravným Hammingovým kódem (7, 4). Do tabulky doplníme všech 16 hodnot, které můžeme na 4 bitech rozlišit, a dopočítáme k nim jednotlivé paritní bity. Získáme tak 16 povolených kódových kombinací.

Data	p_1	p_2	b_1	p_3	b_2	b_3	b_4
0000	0	0	0	0	0	0	0
0001	1	1	0	1	0	0	1
0010	0	1	0	1	0	1	0
0011	1	0	0	0	0	1	1
0100	1	0	0	1	1	0	0
0101	0	1	0	0	1	0	1
0110	1	1	0	0	1	1	0
0111	0	0	0	1	1	1	1

Data	p_1	p_2	b_1	p_3	b_2	b_3	b_4
1000	1	1	1	0	0	0	0
1001	0	0	1	1	0	0	1
1010	1	0	1	1	0	1	0
1011	0	1	1	0	0	1	1
1100	0	1	1	1	1	0	0
1101	1	0	1	0	1	0	1
1110	0	0	1	0	1	1	0
1111	1	1	1	1	1	1	1

Než si ukážeme způsob, kterým Hammingovy kódy detekují a následně opravují chybu v datech, vysvětlíme si dva důležité pojmy.

[51] Pozornému čtenáři neuniklo, že pod označením Hamming (3, 1) se skrývá již známý opakovací kód.

Ham. váha

Hamming. vzdálenost

Hammingova váha $\|u\|_H$ kódového slova u je rovna počtu symbolů ve slově u, které se liší od 0. Máme-li dvě kódová slova u a v, můžeme určit počet pozic, ve kterých se tato dvě slova liší. Počet změn (z nuly na jedničku nebo opačně), které musí být provedeny, abychom ze slova u dostali slovo v, vyjadřuje **Hammingova vzdálenost** $d_H(u, v) = \|u \oplus v\|_H$.

Příklad 5.29

Mějme dvě kódová slova $u = 1001\underline{1}00$ a $v = 1001\underline{0}00$. Hammingova váha kódového slova je $\|u\|_H = 3$, resp. $\|v\|_H = 2$. Hammingova vzdálenost kódových slov u a v je $d_H(u, v) = 1$, rozdílné bity ve slovech jsou podtrženy. Ověříme $u \oplus v = 1001100 \oplus 1001000 = 0000100$, pak tedy $\|0000100\|_H = 1$.

Nastane-li při přenosu chyba a příjemce obdrží některou zakázanou kombinaci, je podle Hammingovy vzdálenosti zjištěna nejbližší povolená kombinace a dojde k opravě dat na tuto povolenou kombinaci. Nejprve si ukážeme, které kombinace jsou povolené a které zakázané, a poté si objasníme princip detekce zakázané kombinace a její opravy.

Příklad 5.30

Uvažujme Hammingův kód $(7, 4)$. Jak již víme z příkladu 5.28, existuje 16 povolených kombinací. Celkový počet kódových kombinací na 7 bitech je 128, tedy zbývajících 112 kombinací je zakázaných. Tabulka ilustruje všechny povolené a zakázané kombinace, jejichž Hammingova vzdálenost od nejbližší povolené kombinace je rovna 1 (uvažujeme pouze jednonásobné chyby).

Datové bity	0000	0001	0010	0011	0100	0101	0110	0111
Povolená kombinace	0000000	1101001	0101010	1000011	1001100	0100101	1100110	0001111
Zakázané kombinace	0000001 0000010 0000100 0001000 0010000 0100000 1000000	0101001 1001001 1100001 1101000 1101011 1101101 1111001	0001010 0100010 0101000 0101011 0101110 0111010 1101010	0000011 1000001 1000010 1000111 1001011 1010011 1100011	0001100 1000100 1001000 1001101 1001110 1011100 1101100	0000101 0100001 0100100 0100111 0101101 0110101 1100101	0100110 1000110 1100010 1100100 1100111 1101110 1110110	0000111 0001011 0001101 0001110 0011111 0101111 1001111

Datové bity	1000	1001	1010	1011	1100	1101	1110	1111
Povolená kombinace	1110000	0011001	1011010	0110011	0111100	1010101	0010110	1111111
Zakázané kombinace	0110000 1010000 1100000 1110001 1110010 1110100 1111000	0001001 0010001 0011000 0011011 0011101 0111001 1011001	0011010 1001010 1010010 1011000 1011011 1011110 1111010	0010011 0100011 0110001 0110010 0110111 0111011 1110011	0011100 0101100 0110100 0111000 0111101 0111110 1101100	0010101 1000101 1010001 1010100 1010111 1011101 1110101	0000110 0010010 0010100 0010111 0011110 0110110 1010110	0111111 1011111 1101111 1110111 1111011 1111101 1111110

Předpokládejme například, že bylo odesláno kódové slovo 1100110, při přenosu nastala chyba a příjemce obdrží kombinaci 1100100. Tato kombinace je vyhodnocena jako zakázaná a je opravena na nejbližší povolenou kombinaci 1100110.

K detekci a opravě chyby při přenosu je nutný výpočet speciální kontrolní posloupnosti na straně příjemce, tzv. **syndromu slova**. Postup je stejný jako u výpočtu paritních bitů, jen do operace vstupuje i samotný paritní bit. Pro Hammingův kód (15, 11) dostáváme:

Syndrom slova

$$\begin{aligned}
s_1 &= p_1 \oplus b_1 \oplus b_2 \oplus b_4 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} \\
s_2 &= p_2 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11} \\
s_3 &= p_3 \oplus b_2 \oplus b_3 \oplus b_4 \oplus b_8 \oplus b_9 \oplus b_{10} \oplus b_{11} \\
s_4 &= p_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus b_8 \oplus b_9 \oplus b_{10} \oplus b_{11}
\end{aligned}$$

Tímto způsobem k paritním bitům $p_1, p_2, \dots, p_m$ získáme kontrolní bity $s_1, s_2, \dots, s_m$. Syndrom slova je potom tvořen posloupností bitů $s = s_m s_{m-1} \dots s_2 s_1$. Pro binární lineární kód (n, k), který má $n - k$ paritních bitů, dostáváme jeden z 2^{n-k} možných syndromů. Syndrom představuje binárně interpretovanou pozici chybného bitu. Je-li nulový, přenos byl bezchybný.

Příklad 5.31

Uvažujme přijatou kódovou kombinaci 1100100 z příkladu 5.30. Spočítáme kontrolní bity:

$$\begin{aligned}
s_1 &= p_1 \oplus b_1 \oplus b_2 \oplus b_4 = 1 \oplus 0 \oplus 1 \oplus 0 = 0 \\
s_2 &= p_2 \oplus b_1 \oplus b_3 \oplus b_4 = 1 \oplus 0 \oplus 0 \oplus 0 = 1 \\
s_3 &= p_3 \oplus b_2 \oplus b_3 \oplus b_4 = 0 \oplus 1 \oplus 0 \oplus 0 = 1
\end{aligned}$$

Syndrom slova $s = s_3 s_2 s_1 = 110$, což je binární reprezentace čísla 6. Při přenosu tedy došlo k chybě, kombinace 1100100 je zakázaná a je nutné ji opravit na povolenou kombinaci 1100110 inverzí šestého bitu zleva.

Pokud by během přenosu nastala dvojnásobná chyba, nebude možné ji ani detekovat, natož opravit. Ukážeme si to na příkladu a poté zformulujeme požadavky na kód, který by byl schopen detekovat a opravit chyby požadované násobnosti.

Příklad 5.32

Předpokládejme odeslané kódové slovo 1000011 a výskyt dvojnásobné chyby, kdy příjemce obdrží slovo 1100111. Spočítáme kontrolní bity:

$$\begin{aligned}
s_1 &= p_1 \oplus b_1 \oplus b_2 \oplus b_4 = 1 \oplus 0 \oplus 1 \oplus 1 = 1 \\
s_2 &= p_2 \oplus b_1 \oplus b_3 \oplus b_4 = 1 \oplus 0 \oplus 1 \oplus 1 = 1 \\
s_3 &= p_3 \oplus b_2 \oplus b_3 \oplus b_4 = 0 \oplus 1 \oplus 1 \oplus 1 = 1
\end{aligned}$$

Syndrom slova $s = s_3 s_2 s_1 = 111$, byla detekována chyba v sedmém bitu. Nejbližší povolenou hodnotou je 1100110, což neodpovídá odeslanému kódovému slovu.

Pod pojmem **minimální vzdálenost kódu** K rozumíme nejmenší Hammingovu vzdálenost všech různých kódových slov, tedy formálně:

Minimální vzdálenost kódu

$$d_H(K) = \min \{ d_H(u, v) \mid u, v \in K \wedge u \neq v \}.$$

Pohledem na jednotlivá kódová slova v příkladech 5.28 a 5.30 se snadno přesvědčíme, že Hammingovy kódy (n, k) mají minimální vzdálenost $d_H(K) = 3$. Tyto kódy se vyznačují schopností detekce a opravy jednonásobných chyb.

Perfektní kódy

Lineární kód označíme jako **perfektní** pro opravu t-násobných chyb, pokud pro každou zakázanou kombinaci z existuje právě jedna povolená kombinace p taková, že $d(z, p) \leq t$. Tedy každý syndrom detekuje jedinečnou zakázanou kombinaci a každá zakázaná kombinace je detekována jedinečným syndromem. Perfektní kódy mají ze všech opravných kódů pro t-násobné chyby minimální redundanci. Takových perfektních kódů však neexistuje mnoho. Můžeme zmínit pouze:

- Hammingovy kódy pro opravu jednonásobných chyb,
- Golayovy[52] kódy pro opravu trojnásobných chyb,
- opakovací kódy s počtem opakování $n = 2t + 1$ pro opravu t-násobných chyb.

V tomto učebním textu se podrobněji zabýváme pouze třídou Hammingových kódů.

Příklad 5.33

Ukažme pro ilustraci konstrukci opakovacího kódu pro přenos čtyřbitových dat s možností opravy dvojnásobných chyb. Dle vztahu z předchozího odstavce pro $t = 2$ odvodíme počet opakování $n = 2t + 1 = 5$ pro každý přenášený znak. Čtyřbitová data 1011 by tedy byla tímto způsobem zakódována do 20bitové posloupnosti 11111 00000 11111 11111. Pokud se při přenosu vyskytne nejvýše dvojnásobná chyba, budou v každé pětici nejvýše dva chybné bity a ze zbývajících tří bitů bude stále možné určit správnou hodnotu.

Limity detekce a opravy

Schopnost lineárního kódu detekovat či opravit t-násobné chyby je odvozena od jeho minimální Hammingovy vzdálenosti d_H. Pokud $t < d_H/2$, chybu lze opravit. Pokud $t = d_H/2$, chybu lze pouze detekovat. Z toho plyne, že pro opravu t-násobných chyb musí být kód konstruován tak, aby $d_H \geq 2t + 1$. Vztah je ilustrován několika příklady v tabulce 5.2.

Tabulka 5.2: *Maximální schopnosti kódu detekovat nebo opravit chybu*

Minimální Hammingova vzdálenost	Počet detekovaných chyb	Počet opravených chyb	
1	0	0	
2	1	0	
3	1	1	⟵ Hammingovy kódy (n, k)
4	2	1	⟵ rozšířené Ham. kódy $(n + 1, k)$
5	2	2	
6	3	2	

Zkrácení Ham. kódů

V příkladu 5.27 na str. 116 jsme uvedli parametry Hammingových kódů (n, k) v závislosti na počtu paritních bitů. Tyto kódy umožňují obecně kódovat jen k datových bitů, které pro nějaké c vyhovují rovnici $2^c = k + c + 1$. Například pro 4 paritní bity dostáváme rovnici $2^4 = k + 5$, jejímž řešením je $k = 11$. Potřebujeme tedy Hammingův kód $(15, 11)$. V případě, kdy nám stačí menší počet datových bitů, můžeme kód zkrátit. Stačí najít takové c, které je řešením nerovnice $2^c \geq k + c + 1$, nepotřebné datové bity vypustíme.

[52] Marcel Jules Édouard Golay (* 1902, † 1989) byl švýcarský matematik, fyzik a teoretický informatik, který aplikoval matematiku na reálné vojenské a průmyslové problémy. Je autorem binárních a ternárních perfektních kódů pro opravu chyb, které jsou zobecněním Hammingových kódů. Tyto kódy byly použity například v sondách Voyager a vedly k pokroku v teorii konečných grup.

Příklad 5.34

Chceme zkonstruovat Hammingův kód pro šestibitové zprávy. Hledáme řešení nerovnice $2^c \geq 6 + c + 1$, kde nejmenší vyhovující $c = 4$. Potřebujeme kód se 4 paritními bity, dle tabulky na str. 116 Hammingův kód $(15, 11)$, který můžeme zkrátit na kód $(10, 6)$.

Pozice bitů	1	2	3	4	5	6	7	8	9	10
Kódované bity	p_1	p_2	b_1	p_3	b_2	b_3	b_4	p_4	b_5	b_6

Rozšíření Ham. kódů

Lineární kód s lichou minimální Hammingovou vzdáleností d_H lze doplnit o další paritní bit, který zabezpečí celé kódové slovo sudou paritou. Minimální Hammingova vzdálenost kódu pak bude $d_H + 1$ a kód tak bude mít schopnost detekce chyby vyšší násobnosti.[53]

Takto můžeme sestrojit **rozšířené Hammingovy kódy**, jejichž minimální Hammingova vzdálenost bude $d_H = 4$. Tyto kódy jsou schopny opravovat jednoduché chyby a současně detekovat dvojité chyby (viz tab. 5.2), nebo pouze detekovat trojité chyby (viz tab. 5.4). Pozice datových a paritních bitů rozšířeného Hammingova kódu $(8, 4)$ znázorňuje tab. 5.3.

Tabulka 5.3: *Pozice datových a paritních bitů v kódových slovech rozšířených Hammingových kódů*

Pozice bitů		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Kódované bity		p_1	p_2	b_1	p_3	b_2	b_3	b_4	p_4
Pokrytí paritním bitem	p_1	✓		✓		✓		✓	
	p_2		✓	✓			✓	✓	
	p_3				✓	✓	✓	✓	
	p_4	✓	✓	✓	✓	✓	✓	✓	✓

Hodnotu přidaného paritního bitu p_4 získáme výpočtem exkluzivní disjunkce (xor) nad všemi bity kódového slova:

$$p_4 = p_1 \oplus p_2 \oplus b_1 \oplus p_3 \oplus b_2 \oplus b_3 \oplus b_4$$

Příklad 5.35

Čtyřbitová data z příkladu 5.28 zabezpečíme rozšířeným Hammingovým kódem $(8, 4)$. Do tabulky doplníme všech 16 hodnot, které můžeme na 4 bitech rozlišit, a dopočítáme k nim jednotlivé paritní bity včetně nového bitu p_4.

Data	p_1	p_2	b_1	p_3	b_2	b_3	b_4	p_4
0000	0	0	0	0	0	0	0	0
0001	1	1	0	1	0	0	1	0
0010	0	1	0	1	0	1	0	1
0011	1	0	0	0	0	1	1	1
0100	1	0	0	1	1	0	0	1
0101	0	1	0	0	1	0	1	1
0110	1	1	0	0	1	1	0	0
0111	0	0	0	1	1	1	1	0

Data	p_1	p_2	b_1	p_3	b_2	b_3	b_4	p_4
1000	1	1	1	0	0	0	0	1
1001	0	0	1	1	0	0	1	1
1010	1	0	1	1	0	1	0	0
1011	0	1	1	0	0	1	1	0
1100	0	1	1	1	1	0	0	0
1101	1	0	1	0	1	0	1	0
1110	0	0	1	0	1	1	0	1
1111	1	1	1	1	1	1	1	1

[53] Rozšíření kódu se sudou minimální Hammingovou vzdáleností tuto vzdálenost nezvýší.

Výpočet syndromu slova probíhá stejně jako u Hammingových kódů (n, k). Podívejme se nyní, jak se při kontrole na straně příjemce projeví chyby různé násobnosti.

Příklad 5.36

Předpokládejme rozšířený Hammingův kód $(8, 4)$ a zprávu 1001, která dle tabulky v příkladu 5.35 bude zakódována do posloupnosti 00110011. Ukážeme všechny situace, které mohou při přenosu nastat: bezchybný přenos, přenos s různým počtem chyb v datových a kontrolních bitech a také v bitu celkové parity.

přijaté slovo	násobnost chyb	syndrom	celková parita	porovnání	oprava
00110011	0	000	0	OK	není nutná
00110010	1	000	1	detekce chyby	lze
01110011	1	010	1	detekce chyby	lze
00100010	2	100	0	detekce chyby	nelze
10111011	2	100	0	detekce chyby	nelze
01011011	3	100	1	detekce chyby	nelze
01100010	3	111	1	detekce chyby	nelze

Na základě pozorování v příkladu 5.36 můžeme nyní zformulovat obecný postup pro kontrolu přijatých dat zabezpečených rozšířenými Hammingovými kódy.

Tabulka 5.4: *Možnosti detekce a opravy chyb u rozšířených Hammingových kódů*

Celková parita	**Syndrom slova**	**Rozhodnutí**
sudá	0	bezchybný přenos
	$\neq 0$	detekce 2 chyb, nelze opravit
lichá	0	detekce 1 chyby, oprava bitu celkové parity
	$\neq 0$	detekce 1 chyby, oprava bitu indikovaného syndromem **nebo** detekce 3 chyb, nelze opravit

Z tab. 5.4 vyplývá, že detekce trojnásobných chyb znemožňuje opravu jednonásobných chyb. To přesně odpovídá tvrzení, které jsme formulovali na str. 121.

Shrnutí Ham. kódů

Hammingovy kódy opravující jednoduché chyby při přenosu mají široké využití v různých odvětvích informatiky od hardwaru (RAM, pevné disky) přes počítačové sítě (Bluetooth, Ethernet, Wi-Fi) až po satelitní komunikaci a digitální audio přenosy. Přestože se v tomto textu nevěnujeme dalším lineárním kódům pro opravu vícenásobných chyb, na závěr této podkapitoly zobecníme požadavky na parametry takových kódů. Uvažujeme-li binární lineární kód (n, k), pak při požadované délce kódového slova n, minimálním počtu datových bitů k a minimálním počtu kontrolních bitů $n - k$ v kódovém slově hledáme řešení nerovnice:

$$
\begin{aligned}
2^{n-k} &\geq 1 + n && \text{pro detekci a opravu jednoduchých chyb,}\\
2^{n-k} &\geq 1 + n + \binom{n}{2} && \text{pro detekci a opravu dvojitých chyb,}\\
&\vdots\\
2^{n-k} &\geq 1 + n + \binom{n}{2} + \ldots + \binom{n}{t} && \text{pro detekci a opravu } t\text{-násobných chyb.}
\end{aligned}
$$

Grayův kód

Pro podporu opravy chyb v digitální komunikaci (pozemní a kabelové televize, odpovídače sekundárního radaru v letadlech, inkrementální snímače polohy) a také třeba při grafické metodě minimalizace logických funkcí pomocí Karnaughovy mapy se uplatňuje samoopravný **Grayův**[54] **kód** (též známý jako *kód konstantní změny*). Jedná se o zrcadlový binární kód, ve kterém se každé dvě po sobě jdoucí kódové hodnoty liší v bitovém vyjádření změnou pouze jedné bitové pozice. Při přechodu mezi dvěma kódovými slovy proto může nastat nepřesností chyba rovnající se nejvýše jedničce. Při realizaci logického obvodu se tak můžeme vyhnout vzniku nežádoucích stavů způsobených současnou změnou více bitů.

V tab. 5.5 je znázorněn postup konstrukce čtyřbitového Grayova kódu, z uvedených vztahů pro kódování a dekódování však můžeme snadno odvodit obecný postup aplikovatelný při konstrukci n-bitového Grayova kódu.

Tabulka 5.5: *Schéma konstrukce čtyřbitového Grayova kódu*

Znak	Přímý kód (BCD)				Grayův kód			
	b_0	b_1	b_2	b_3	g_0	g_1	g_2	g_3
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Kódování:

$$\begin{aligned} g_0 &= b_0 \\ g_1 &= b_0 \oplus b_1 \\ g_2 &= b_1 \oplus b_2 \\ g_3 &= b_2 \oplus b_3 \end{aligned}$$

Dekódování:

$$\begin{aligned} b_0 &= g_0 \\ b_1 &= b_0 \oplus g_1 \\ b_2 &= b_1 \oplus g_2 \\ b_3 &= b_2 \oplus g_3 \end{aligned}$$

5.3.2 Zabezpečení proti neoprávněným osobám

Mechanismy zabezpečení informací proti záměrnému rušení, tj. proti neoprávněným osobám, sledují obvykle dva různé cíle:

1. zabezpečení proti *neoprávněnému čtení* – přenášená informace nesmí být čitelná jiné osobě než oprávněnému příjemci;
2. zabezpečení proti *neoprávněné modifikaci* – na cílové straně musí být zjistitelné, zda informace nebyla při přenosu změněna, čímž se dokazuje pravost odesílatele.

[54] Frank Gray (* 1887, † 1969) byl fyzik a výzkumník v Bellových laboratořích v New Jersey. Kód si nechal patentovat v roce 1953. Jeho využití je nejen v elektronice, ale má také mnoho aplikací v matematice.

Kryptologie

Proti neoprávněnému čtení se používají techniky šifrování. Věda, která se zabývá šifrováním ze všech úhlů pohledu, se nazývá **kryptologie**[55] a jejími hlavními disciplínami jsou kryptografie a kryptoanalýza.

Kryptografie

Kryptografie studuje šifrovací algoritmy, kryptografické nástroje, hardwarové implementace šifrovacích algoritmů, kryptografické protokoly apod. Zabývá se tím, jak informaci transformovat do „nečitelné" podoby a také jak ji transformovat zpět se znalostí správného postupu. Jejím úkolem je zašifrovat zprávu, a znemožnit tím získání obsažené informace nepovolanou třetí stranou, které by se případně mohlo podařit informaci zachytit.

Steganografie

Zvláštní pozornost zasluhuje **steganografie**[56], která je podoborem kryptografie. Jejím úkolem není utajit význam zprávy, ale skrýt její samotnou existenci. Mezi rekvizity patří nejrůznější tajné inkousty, duté hole, dvojitá dna kufrů apod. Moderní elektronická steganografie umožňuje ukrývat zprávy v datech nesoucích informace zcela jiného druhu, například obrazy, zvuky nebo videa.

Kryptoanalýza

Kryptoanalýza je „protivníkem" kryptografie, protože zkoumá, jak šifrovanou informaci převést do čitelné podoby bez znalosti klíče. Zabývá se analýzou síly šifrovacích metod, možnostmi průniku do kryptografických systémů a jejich diskreditace.

Šifrování, tedy nahrazení otevřeného textu zdánlivě nesmyslným, nesrozumitelným a nečitelným kódem, lze vyřešit principiálně dvěma způsoby: *transpozicí* (změnou pořadí znaků zprávy) a *substitucí* (systematickou náhradou znaků zprávy zcela jinými znaky).

Transpoziční šifry

Transpoziční šifry mohou být velmi účinné, uvědomíme-li si, že změnu pořadí znaků zprávy můžeme provést $n!$ způsoby, kde n je počet znaků zprávy. Již pro krátké třicetiznakové zprávy dostáváme astronomických $2{,}65 \cdot 10^{32}$ možností permutací znaků. Tento prostor řešení nelze efektivně prohledat ani při použití nejmodernější výpočetní techniky. Otevřený text musí být zbaven všech prvků, které by mohly tvořit nápovědu při dešifrování – české znaky s diakritikou jsou nahrazeny znaky bez diakritiky, jsou odstraněny mezery a interpunkce.[57]

Substituční šifry

Substituční šifry jsou založeny na systematické náhradě znaků otevřené zprávy za jiné znaky nebo symboly. Použijeme-li pro záměnu určitého znaku vždy jeden a tentýž šifrový symbol, jedná se o tzv. **monoalfabetickou šifru**. Jsou-li šifrové znaky nahrazující původní znak při každém výskytu střídány, jde o tzv. **polyalfabetickou šifru**.

Monoalfabetické šifry jsou dnes považovány za snadno prolomitelné, i když testování všech kombinací možných náhrad by vedlo k prohledávání prostoru $4 \cdot 10^{26}$ klíčů. Prolomení monoalfabetické šifry se však provádí jinak. Vycházíme při tom z faktu, že v otevřené zprávě jsou v určitém jazyku četnosti znaků nerovnoměrné. Ze zašifrované zprávy zjistíme relativní četnosti jednotlivých znaků pomocí tzv. **frekvenční analýzy** a pokusíme se je porovnat s četnostmi otevřených znaků.

Polyalfabetické šifry nejsou frekvenční analýzou prolomitelné. Znakům otevřené zprávy jsou přiřazovány různé šifrové znaky podle určitých pravidel řízených většinou nějakým heslem. Frekvenční spektrum šifrových znaků je tedy zcela odlišné od původního

[55] Termín pochází z řeckých slov *κρυπτός* (tajný) a *λόγος* (věda).

[56] Termín pochází z řeckých slov *στεγανός* (schovaný) a *γράφειν* (psát). Neplést se stenografií (těsnopisem).

[57] V sivosuoltsi s vzýukemm na Cmabridge Uinervtisy vlšyo njaveo, že nzeáelží na pořďaí psíemn ve solvě. Jedniá dleůitžá věc je, aby blya pnvrí a psoelndí pímesna na srpváénm mstíě. Zybetk mžůe být tonátlí sěms a my to přoád bez porlbméů peřčetme. Je to porto, že ldiksý mezok netče kdažé pensímo, ale svolo jkao cleek.

frekvenčního spektra otevřené zprávy. Prolomení této šifry je poměrně komplikované – je možné postupovat přes analýzu šifrované zprávy, odkud lze za určitých okolností odhadnout délku hesla řídícího přiřazování alternativních znaků. Pak se úloha transformuje na několik úloh monoalfabetických, které se prolomí frekvenční analýzou.

Moderní kryptografie

V dnešní době vede rozsáhlé využití výpočetní techniky ke stále se zvyšujícímu objemu zpracovávaných dat. Tato data jsou často důvěrného charakteru a není žádoucí, aby k nim měla přístup nepovolaná osoba. Pro utajení dat je možné používat různé prostředky, ať už technické či programové, nicméně druhy utajení a zabezpečení jsou rozdílné z hlediska bezpečnosti i z hlediska financí. Fyzické zabezpečení, jako jsou speciální linky pro přenos dat nebo úschova dat v chráněných a hlídaných prostorách, je finančně velmi náročné a neposkytuje stoprocentní ochranu. Na druhé straně se však díky snadno dostupné výpočetní technice stává šifrování dat levnou záležitostí a vzhledem k dlouhému vývoji a matematické propracovanosti šifrovacích systémů také velmi bezpečnou. Není tedy divu, že se v dnešní době kryptografie těší velké pozornosti v soukromém i tržním prostředí, nemluvě o armádě.

Šifrovací klíč

Kryptografické systémy používané v současnosti jsou založeny výhradně na matematických principech. Všechny používané šifry jsou podrobeny důkladné analýze a je poměrně přesně známa síla šifer a předpokládaný čas potřebný k jejich prolomení. Důležitou roli v případě šifer hraje **klíč** tvořený obvykle posloupností čísel, pomocí níž se manipuluje se zprávou, a je tak možné provést šifrování či dešifrování. V moderních šifrovacích metodách jsou zpráva i klíč reprezentovány posloupností bitů. Délka klíče je obvykle v desítkách bitů, běžně se používají 128bitové klíče. Proces šifrování je řešen matematickou operací, do níž vstupují data a klíč, výsledkem je šifra. Dešifrování je rovněž řešeno matematickou operací, do níž vstupují šifra a klíč, výsledkem jsou otevřená data (obr. 5.4). Vzniklé zašifrované hodnoty jsou bez znalosti klíče zdánlivě zcela nesmyslné a při jejich získání neoprávněnou osobou není informace čitelná.

Obrázek 5.4: *Obecný šifrovací postup*

Jednosměrné šifry

Jednu z podob moderního šifrování představují **jednosměrné šifry**. Jak již napovídá název, pracují pouze jedním směrem, proto zašifrovanou zprávu již nelze dešifrovat. Tento systém, ač se na první pohled může zdát neužitečný, se používá velmi často, a to k ověření autentičnosti dat či k digitálnímu podpisu. Jako příklad jednosměrných šifer uveďme tzv. *hashovací funkce*. Tyto funkce vytvářejí na základě vstupních dat kontrolní součet, tzv. **otisk** (*hash*), který má obvykle mnohem menší délku než šifrovaná vstupní data. Mezi

typické jednosměrné šifrovací algoritmy patří MD5[58] (*Message-Digest Algorithm 5*) nebo SHA[59] (*Secure Hash Algorithm*). Tímto způsobem jsou zabezpečena i hesla uživatelů v Univerzitním informačním systému. Při přihlašování do systému se ze zadaného hesla vypočte otisk, který musí souhlasit s otiskem uloženým v databázi (samotná hesla se neukládají).

Obousměrné šifry

U **obousměrných šifer** jsme schopni při znalosti správného klíče získat ze zašifrovaných dat původní data. Je-li pro šifrování i dešifrování použit stejný klíč, mluvíme o symetrickém šifrování. V případě, že klíče na šifrování a dešifrování jsou různé, jde o asymetrické šifrování. Používají-li se současně obě techniky, mluvíme o hybridním šifrování.

Symetrické šifrování

Metoda **symetrického šifrování** používá jeden klíč a má jednu podstatnou nevýhodu – klíč se musí nějakým způsobem předat příjemci, protože bez něj je zabezpečená zpráva zcela nečitelná a nerozluštitelná. Výhodou naopak je, že způsob šifrování i dešifrování je velmi jednoduchý, pro každou zprávu lze snadno vytvořit zcela individuální klíč o různé délce, čímž je v praxi zajištěna naprostá nerozluštitelnost a neprolomitelnost klíče. V současnosti se obvykle používají klíče o délce 128 bitů, u citlivých dat 256 bitů.

Asymetrické šifrování

Asymetrické šifrování používá dva klíče – jeden se nazývá *soukromý*, druhý se označuje jako *veřejný*. Pokud nějakou zprávu zašifrujeme jedním klíčem, může být dešifrována pouze druhým klíčem a naopak. Klíče tvoří dvojice (*keypair*) a musejí být vytvořeny současně. Další podmínkou je, aby z jednoho klíče nešel v žádném případě odvodit klíč druhý. Nejznámější a nejrozšířenější asymetrickou šifrou je RSA[60] (*Rivest-Shamir-Adleman*).

Soukromý klíč je předán oprávněnému držiteli zcela diskrétně, veřejný klíč je dostupný komukoliv. Při zabezpečení informace se tedy distribuuje pouze veřejný klíč, který sám o sobě nemá žádnou cenu. Zašifruje-li však kdokoli nějakou zprávu veřejným klíčem příjemce, má jistotu, že tuto zprávu přečte právě jen příjemce, který vlastní příslušný soukromý klíč. Zašifruje-li někdo určitou zprávu svým soukromým klíčem, může ji sice kdokoli rozluštit pomocí příslušného veřejného klíče, ale tím je dokázáno, že odesílatelem je majitel daného soukromého klíče.

Certifikační autorita

V obou uvedených případech však potřebujeme vědět, které fyzické osobě nebo organizaci patří soukromý klíč. K tomuto osvědčení autenticity určitého uživatele slouží tzv. **certifikační autorita**. Jedná se o organizaci, která se stará o vytváření a distribuci asymetrických klíčů a zároveň vede informace o fyzických majitelích soukromých částí klíčů. Na požádání je pak schopna sdělit, kdo je fyzickým majitelem soukromého klíče příslušného k určitému veřejnému klíči. Při přidělování soukromého klíče se zpravidla uživatel osobně dostaví do úřadovny certifikační autority, kde dostane klíč na nějakém datovém médiu. Odpadá tedy jakýkoli přenos soukromého klíče přes síťová spojení. Každá certifikační autorita zároveň určuje, jak dlouho je daný klíč platný a zabezpečuje jeho změnu při ztrátě či prolomení. Akreditovanými poskytovateli certifikačních služeb v České republice jsou První certifikační autorita, a. s., PostSignum QCA (Česká pošta, s. p.) a eIdentity, a. s.

[58] Algoritmus vytvořil v roce 1991 Ronald Linn Rivest (* 1947). Ze vstupní zprávy je vypočten hash o délce 128 bitů. V roce 2004 byl algoritmus plně prolomen, proto již není k ověření autentičnosti zprávy použitelný.

[59] Algoritmus byl vytvořen Národní bezpečnostní agenturou v USA v roce 1993. Jedná se ve skutečnosti o rodinu několika algoritmů. Varianta SHA-2 produkuje hash o délce 256, 384 nebo 512 bitů. Od roku 2012 je známa varianta SHA-3, která je založena na odlišném principu a zaručuje vyšší bezpečnost.

[60] Algoritmus představili v roce 1977 Ronald Linn Rivest, Adi Shamir (* 1952) a Leonard Max Adleman (* 1945). Je založen na myšlence, že je jednoduché vynásobit dvě velká prvočísla, ale rozklad takového součinu na prvočinitele (tzv. faktorizace) je časově náročný.

Elektronický podpis

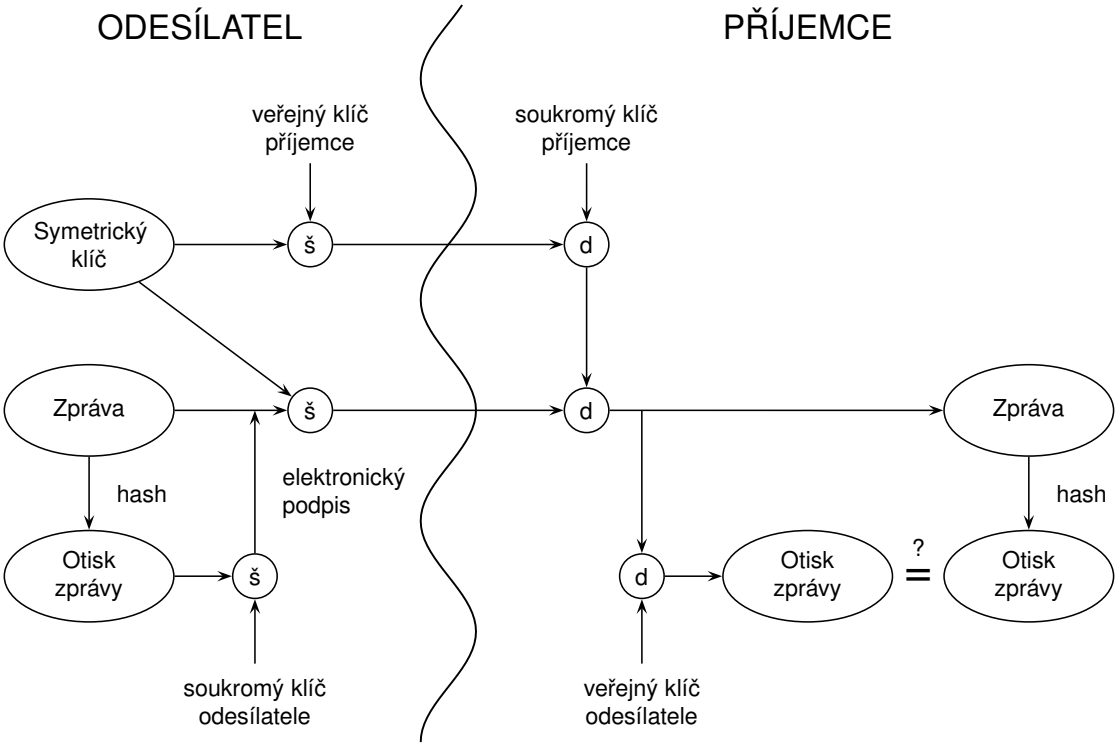
Skutečnost, že dokument nebyl při přenosu modifikován neoprávněnou osobu, se prokazuje **elektronickým podpisem** (*e-signature*). Dle zákona č. 227/2000 Sb., o elektronickém podpisu, se elektronickým podpisem rozumí údaje v elektronické podobě, které jsou připojeny k datové zprávě nebo jsou s ní logicky spojeny a které slouží jako metoda k jednoznačnému ověření identity podepsané osoby ve vztahu k datové zprávě. Tzv. **zaručený elektronický podpis** musí splňovat požadované vlastnosti:

1. Podpis je jednoznačně spojen s podepisující osobou.
2. Je možná identifikace podepisující osoby ve vztahu k datové zprávě.
3. Podpis byl přiřazen k datové zprávě prostředky, které má podepisující osoba pod svou výhradní kontrolou.
4. Podpis je připojen k datové zprávě způsobem, který umožňuje odhalit změnu dat.

Elektronický podpis je vyjádřen blokem bajtů určité délky, která nezávisí na délce či obsahu podepisovaného dokumentu, ale závisí na typu použitého algoritmu. Podpis umožňuje ověřit několik skutečností:

- *autenticitu zprávy* – příjemce ví, kdo je autorem zprávy;
- *integritu zprávy* – příjemce ví, zda byl obsah zprávy během přenosu změněn či ne;
- *nepopiratelnost zprávy* – odesílatel nemůže popřít autorství zprávy s daným obsahem;
- *časové ukotvení zprávy* – podpis může obsahovat časové razítko vydané důvěryhodnou třetí stranou.

Princip komunikace pomocí zpráv zabezpečených proti neoprávněnému čtení a modifikaci při přenosu je znázorněn na obr. 5.5.



Obrázek 5.5: *Princip odeslání a přijetí bezpečné a podepsané zprávy*

Odeslání bezpečné zprávy

Má-li dojít k odeslání bezpečné a podepsané zprávy, musí odesílatel učinit následujících šest kroků:

1. vygenerovat otisk dokumentu;
2. zašifrovat otisk soukromým klíčem odesílatele;
3. připojit zašifrovaný otisk k původnímu dokumentu;
4. generátorem náhodných čísel vytvořit tajný symetrický klíč pro jednu relaci;
5. zašifrovat tajný klíč veřejným klíčem příjemce dokumentu a odeslat šifru příjemci;
6. zašifrovat tajným klíčem zprávu (tj. původní dokument s připojenou šifrou otisku dokumentu) a odeslat šifru příjemci.

Přijetí bezpečné zprávy

Aby si příjemce mohl bezpečnou a podepsanou zprávu přečíst, musí učinit následujících šest kroků:

1. dešifrovat přijatou šifru tajného symetrického klíče soukromým klíčem příjemce;
2. dešifrovat přijatou šifru zprávy tajným symetrickým klíčem;
3. rozdělit zprávu na dvě části – původní dokument a šifru otisku;
4. dešifrovat šifru otisku dokumentu veřejným klíčem odesílatele;
5. vygenerovat vlastní otisk přijatého dokumentu;
6. porovnat přijatý otisk dokumentu a vlastní vygenerovaný otisk.

V případě shody při srovnání obou otisků je dokument považován za autentický, důvěrný a celistvý. Je ověřena totožnost odesílatele a máme jistotu, že zpráva nebyla po cestě modifikována.

Význam takového zabezpečení neustále vzrůstá, neboť sílí tlak firem a peněžních ústavů na obchodování po Internetu. Vzhledem k tomu, že při přenosu dat po síti není možné odstranit neoprávněné čtení, zůstává šifrování jedinou alternativou zabezpečení. Prolomení šifry bez znalosti klíče však není zcela nemožné, i když je pracné. Při stoupajícím výpočetním výkonu počítačů připadá v úvahu i použití tzv. *hrubé síly*, kdy se zkusmo aplikují všechny možné hodnoty klíče dané šířky, tj. například v případě 40bitového klíče je to $2^{40} \doteq 10^{13}$ možností, které lze prověřit řádově za několik dní. Proto se používají podstatně delší klíče, u kterých není v silách dnešní výpočetní techniky správnou kombinaci v reálném čase najít.

Kontrolní otázky

11. Jaký je princip detekčních kódů?

12. V čem spočívá výhoda kombinované parity?

13. Co je to kontrolní součet? Které druhy kontrolních součtů se používají?

14. K čemu slouží kontrolní číslice a jakým způsobem se získává?

15. Jaký je princip samoopravných kódů?

16. Jaké vlastnosti má opakovací kód a kde se používá?

17. Jak schopnosti mají Hammingovy kódy a kde se prakticky používají?
18. Jaké výhody přináší rozšířené Hammingovy kódy?
19. Jaké vlastnosti mají perfektní kódy?
20. Kde se používá Grayův kód a na jakém principu je založen?
21. Čím se zabývá kryptografie a čím kryptoanalýza?
22. Jaký je rozdíl mezi transpoziční a substituční šifrou?
23. Na jakém principu jsou založeny moderní kryptografické metody?
24. Jaký je rozdíl mezi jednosměrným a obousměrným šifrováním?
25. Čím se liší symetrické a asymetrické šifrování?
26. K čemu slouží elektronický podpis a jak jej lze získat?
27. Jakým způsobem lze poslat a přijmout bezpečnou a podepsanou zprávu?

5.4 Příklady na procvičení

Příklad 5.37 Rozhodněte, zda může existovat binární prefixové kódování desítkových číslic, jehož kódové znaky mají délku nejvýše 5 a číslice nula má kód 0. Pokud ano, najděte jej.

Příklad 5.38 Rozhodněte, zda může existovat binární prefixový kód se čtyřmi kódovými znaky a délkou kódových slov 1, 2, 3 a 3 bity. Pokud existuje, sestrojte jej.

Příklad 5.39 Zjistěte, kolik symbolů musí mít kódová abeceda k zakódování 25 znaků do blokového kódu s kódovými znaky délky 2.

Příklad 5.40 Předpokládejte, že při hře „Člověče, nezlob se!“ se jednotlivé hodnoty hodu kostkou kódují tímto kódem: 000, 001, 010, 011, 100, 101. Vypočtěte entropii takového kódu.

Příklad 5.41 Stanovte optimální binární kód pro pět různých znaků vyskytujících se ve zdroji se stejnou pravděpodobností. Vypočtěte entropii takového kódu.

Příklad 5.42 Zpráva obsahuje 5 různých znaků s pravděpodobností výskytu postupně 0,4; 0,25; 0,2; 0,1; 0,05. Nalezněte optimální binární kód, zjistěte střední délku kódového slova a určete entropii kódu.

Příklad 5.43 Zpráva obsahuje 5 různých znaků s pravděpodobností výskytu postupně 0,38; 0,18; 0,16; 0,16; 0,12. Nalezněte optimální binární kód, zjistěte střední délku kódového slova a ověřte, který kód je efektivnější.

Příklad 5.44 V továrně vyrábí 8 druhů sušenek. První dva druhy se vyrábí rychlostí 2 balíčky za sekundu, další dva druhy se vyrábí rychlostí 1 balíček za sekundu, zbylé druhy se vyrábí rychlostí 1 balíček za 2 sekundy. Vypočítejte entropii, pokud každý balíček má vlastní rovnoměrný binární kód.

Příklad 5.45 Klimatizace ve skladu jablek se může nacházet ve čtyřech stavech: topení (`0`), temperování (`10`), chlazení (`110`), vypnuto (`111`). Každá změna je zaznamenána do textového souboru. Z prvních čtyř bajtů v takovém souboru zjistěte, kolikrát během sledovaného období nastal stav temperování: `11101011 01000110 11011101 01100110`.

Příklad 5.46 Regulátor jízdy zařazuje po hodinovou dobu práce lokomotivy její motory sériově (10 min.), paralelně (20 min.), sérioparalelně (5 min.), s odporníkem (5 min.) a s rekuperací (20 min.). Navrhněte optimální kód pro vyjádření stavu regulátoru.

Příklad 5.47 Předpokládejte, že zpráva může obsahovat číslice 0–9 s přibližně stejnou pravděpodobností výskytu. Zvolte kód s minimální redundancí a zjistěte, kolik nejméně bitů bude zabírat prostor v paměti, do něhož byla uložena zpráva skládající se z posloupnosti číslic 371.

Příklad 5.48 Zabezpečte:
a) `7A28 4E1C` sudou paritou v osmém (nejvyšším) bitu
b) `9AF8 EE1C` lichou paritou v devátém (nejvyšším) bitu
c) `5A17 541C` sudou paritou v osmém (nejvyšším) bitu
d) `287A 301F` lichou paritou v osmém (nejvyšším) bitu

Příklad 5.49 Vypočtěte kontrolní součet (podélnou paritou) o zadané délce z úseku dat:
a) `7A28 4E1C` (8 b)
b) `7A28 4E1C 7A28 4E1C` (16 b)
c) `5A17 541C` (8 b)
d) `A87A B01F 2061 7CCB` (16 b)

Příklad 5.50 Je dán textový soubor obsahující řetězec „`AHOJ STUDENTE`“. Vypočítejte 8bitový kontrolní součet (podélnou paritou) prvních pěti znaků v souboru.

Příklad 5.51 Při přenosu byla získána čtyři desítková čísla 119, 210, 165, 238. Zjistěte, zda při přenosu nenastala chyba, víte-li, že poslední z hodnot představuje kontrolní součet.

Příklad 5.52 Jsou dána data `9D17 DF97 9D17 DF97` a jejich 16bitový kontrolní součet `A3B5`. Zjistěte, kde nastala chyba.

Příklad 5.53 Jsou dána data `5A17 549C` zabezpečená v osmém (nejvyšším) bitu sudou paritou. Jejich kontrolní součet je `05`. Zjistěte, kde nastala chyba, a napište opravená data.

Příklad 5.54 Je dána hodnota `5F`. Rozhodněte a zdůvodněte, jaký bude kontrolní součet:
a) 511 takových hodnot
b) 511 takových hodnot, mezi nimiž se dvakrát vyskytla hodnota `6D`

Příklad 5.55 Ověřte správnost následujících dat:
a) ISBN 978-80-2572-491-0
b) IČO 06688217
c) VIN vozidla TMBFH25J4C3046434
d) číslo lokomotivy 749 102-0

Příklad 5.56 Zjistěte, která číslice patří místo otazníku v ISBN 978-80-2572-283-?.

Příklad 5.57 Ověřte správnost svého rodného čísla a čísla běžného účtu v bance.

Příklad 5.58 Určete Hammingovu vzdálenost uvedených hexadecimálních hodnot:

a) `0000` a `0101`
b) `100110` a `000110`
c) `9D17` a `DF97`
d) `A7453C` a `A85534`

Příklad 5.59 Sestrojte opakovací kód pro data o délce k bitů opravující t-násobné chyby:

a) $k = 5,\ t = 1$
b) $k = 5,\ t = 2$
c) $k = 4,\ t = 1$
d) $k = 4,\ t = 2$
e) $k = 4,\ t = 3$
f) $k = 3,\ t = 2$
g) $k = 3,\ t = 3$
h) $k = 2,\ t = 2$
i) $k = 2,\ t = 3$

Příklad 5.60 Uvažujte opakovací kód pro čtyřbitová data. Nalezněte a opravte chyby:

a) `10011 11101 01010 10111`
b) `11111 00110 10111 10011`
c) `00100 10110 10011 11111`
d) `01100 10001 01111 10110`

Příklad 5.61 Vypočítejte minimální počet kontrolních bitů Hammingova kódu (n, k):

a) $k = 1$
b) $k = 2$
c) $k = 3$
d) $k = 4$
e) $k = 5$
f) $k = 6$
g) $k = 7$
h) $k = 8$
i) $k = 9$
j) $k = 10$
k) $k = 11$
l) $k = 12$

Příklad 5.62 Sestrojte Hammingův samoopravný kód pro data o délce k bitů:

a) $k = 1$
b) $k = 2$
c) $k = 3$
d) $k = 4$
e) $k = 5$
f) $k = 6$
g) $k = 7$
h) $k = 8$

Příklad 5.63 Uvažujte Hammingův kód $(7, 4)$. Zkontrolujte data a opravte chyby:

a) `1001101`
b) `1010010`
c) `1110000`
d) `1111110`
e) `0000010`
f) `0100101`
g) `1010110`
h) `0010101`

Příklad 5.64 Sestrojte rozšířený Hammingův kód pro data o délce k bitů:

a) $k = 1$
b) $k = 2$
c) $k = 3$
d) $k = 4$
e) $k = 5$
f) $k = 6$
g) $k = 7$
h) $k = 8$

Příklad 5.65 Uvažujte rozšířený Hammingův kód $(8, 4)$. Zkontrolujte data a opravte chyby:

a) `10010111`
b) `01100111`
c) `10110111`
d) `11011111`
e) `01000100`
f) `11011100`
g) `10011110`
h) `11000000`

Příklad 5.66 Vypočítejte redundanci rozšířeného Hammingova kódu $(8, 4)$.

Příklad 5.67 Sestrojte Grayův kód pro data o délce n bitů a spočítejte jeho redundanci:

a) $k = 2$
b) $k = 3$
c) $k = 4$
d) $k = 5$

Příklad 5.68 Je dán klíč `01101` (binárně) a zpráva `9D17 DF97` (hexadecimálně). Zašifrujte zprávu zleva operací `xor`.

Příklad 5.69 Je dána šifra `3CA7 45F6` (hexadecimálně) a klíč `100110` (binárně). Dešifrujte zprávu zleva operací `xor`.

Shrnutí

Tato kapitola se věnovala přenosu dat. Objasnili jsme rozdíly mezi analogovým a digitálním signálem a seznámili se s principem digitalizace signálu a výhodami, které přináší. Zabývali jsme se různými možnostmi a metodami kódování dat, hledáním optimálního (nejkratšího) kódu a způsobem hodnocení jeho kvality. Dále jsme se věnovali rušivým vlivům při přenosu dat, které mohou být způsobeny technickými nedokonalostmi přenosového kanálu nebo záměrnými zásahy třetích osob s cílem přečíst nebo dokonce modifikovat obsah zprávy. Ukázali jsme principy detekčních a samoopravných kódů a stanovili technické možnosti v oblasti detekce a opravy chyb požadované násobnosti. V závěru kapitoly jsme se seznámili s moderními metodami šifrování dat a vysvětlili princip zasílání a přijímání bezpečných a podepsaných zpráv.

Slovníček pojmů

Asymetrické šifrování: metoda, při které jsou pro šifrování a dešifrování použity různé šifrovací klíče.
Certifikační autorita: organizace, která zabezpečuje distribuci klíčů pro asymetrické šifrování a vede informace o jejich majitelích.
Dekódování: inverzní postup ke kódování.
Detekční kód: kód pro odhalování chyb v datech.
Diskrétní signál: veličina nabývající určitých hodnot v diskrétních časových okamžicích.
Elektronický podpis: otisk zprávy zašifrovaný soukromým klíčem odesílatele.
Frekvenční analýza: jednoduchá metoda pro dešifrování monoalfabetických šifer.
Hammingův kód: samoopravný lineární kód pro opravu jednonásobných chyb.
Hammingova vzdálenost: počet rozdílných bitů mezi dvěma porovnávanými hodnotami.
Jednoduchá parita: detekční metoda využívající jednoho paritního bitu.
Jednosměrné šifrování: metoda, při níž není možné zašifrovanou zprávu již dešifrovat.
Kombinovaná parita: kombinace detekující vícenásobné chyby při přenosu s využitím více paritních bitů, v určitých případech umožňující i lokalizaci a opravu chyby.
Kontrolní součet: metoda zabezpečení dat umožňující detekci vícenásobných chyb.
Kód: předpis umožňující přiřazování znaků při kódování.
Kódování: vzájemné přiřazování znaků dvou abeced.
Kryptoanalýza: vědní obor zabývající se dešifrováním zpráv bez znalosti klíče, analýzou síly šifrovacích metod a možnostmi průniku do šifrovacích systémů.
Kryptografie: vědní obor zabývající se šifrovacími algoritmy a jejich implementací.
Kryptologie: vědní obor zabývající se všemi aspekty šifrování.
Monoalfabetická šifra: šifra, v níž je každý výskyt konkrétního znaku zaměněn vždy za tentýž šifrový symbol.
Nerovnoměrné kódování: postup kódování založený na tom, že každý znak je zakódován různě dlouhou posloupností nul a jedniček podle daného algoritmu.
Obousměrné šifrování: metoda, u níž lze zašifrovanou zprávu se znalostí klíče dešifrovat.
Optimální kód: kód s maximální entropií a s minimální redundancí.
Otisk (hash): kontrolní součet určité délky pro jednosměrné šifrování.

Perfektní kód: samoopravný kód s minimální redundancí.
Polyalfabetická šifra: šifra, u níž je konkrétní znak zaměňován za různé šifrové symboly.
Prefixový kód: jednoznačně dekódovatelný kód, ve kterém žádné kódové slovo není prefixem jiného kódového slova.
Přenosový řetězec: schéma vyjadřující princip přenosu dat od zdroje k cíli.
Redundance kódu: nadbytečnost, která vyjadřuje hospodárnost kódu.
Rovnoměrné kódování: postup kódování založený na tom, že každý znak je zakódován stejně dlouhou posloupností nul a jedniček.
Samoopravný kód: metoda založená na existenci povolených a zakázaných kódových kombinací, která umožňuje detekci, lokalizaci i opravu chyb při přenosu.
Signál: fyzikální veličina představující stavy nebo procesy fyzikálního média.
Soukromý klíč: jeden z páru klíčů pro asymetrické šifrování, který zná pouze jeho majitel.
Spojitý signál: veličina nabývající nekonečně mnoha hodnot v určitém intervalu.
Steganografie: vědní obor zabývající se utajováním existence zpráv.
Substituční šifra: šifra založená na náhradě znaků zprávy zcela jinými znaky.
Symetrické šifrování: metoda, při které je pro šifrování i dešifrování použit stejný klíč.
Syndrom slova: kontrolní posloupnost umožňující lokalizaci jednonásobných chyb.
Šifrovací klíč: posloupnost čísel použitá pro šifrování a dešifrování v počítačové podobě.
Transpoziční šifra: šifra založená na změně pořadí znaků ve zprávě.
Veřejný klíč: jeden z páru klíčů pro asymetrické šifrování, který je veřejně přístupný.

Řešení kontrolních otázek

1. Jaký je rozdíl mezi spojitým a diskrétním signálem?

Spojitý (analogový) signál je dán spojitou (nebo po částech spojitou) funkcí spojitého času. Tím se liší od signálu diskrétního (digitálního), který je dán funkcí definovanou pouze v diskrétních časových okamžicích a tvoří tak posloupnosti funkčních hodnot.

2. Proč je možné diskrétní signál přenášet bez zkreslení?

Protože přenášíme pouze dvě hodnoty (0 a 1), které lze při přenosu spolehlivě rozlišit i v případě výskytu rušení.

3. K čemu slouží přenosový řetězec a z jakých částí se skládá?

Přenosový řetězec je obecné schéma popisující přenos dat od zdroje k cíli. Skládá se z kódování dat, jejich modulace, přenosu, demodulace a dekódování na straně příjemce.

4. Co vyjadřuje redundance kódu a jak ji můžeme spočítat?

Redundance vyjadřuje hospodárnost kódu. Počítáme ji podle vztahu $R = 1 - \frac{H}{H_{\max}}$, kde H je skutečná entropie a $H_{\max}$ je teoretická maximální entropie při použití stejné abecedy.

5. Jaký je rozdíl mezi blokovým a prefixovým kódem?

Prefixový kód je prosté kódování, u kterého žádné kódové slovo není prefixem jiného kódového slova. Blokový kód je prosté kódování, u kterého mají všechna kódová slova stejnou délku. Protože blokový kód musí být prostým zobrazením, je nutně také prefixovým kódem.

6. Které z kódů ISO 8859-2, UTF-8 a UTF-16 jsou blokové a které nerovnoměrné?

Kódy ISO 8859-2 a UTF-16 jsou blokové, kód UTF-8 je nerovnoměrný.

7. Jak u blokového kódování zjistíme délku kódových znaků?

Počet bitů potřebných pro zakódování jednotlivých znaků zjistíme výpočtem informační entropie jevu, který je zdrojem informace.

8. Jak ověříme, zda lze sestrojit prefixový kód s požadovanými vlastnostmi?

Zjistíme platnost Kraftovy nerovnosti. Pokud existuje binární prefixový kód s n kódovými znaky a délkami kódových slov $d_1, d_2, \ldots, d_n$, pak platí $2^{-d_1} + 2^{-d_2} + \ldots + 2^{-d_n} \leq 1$.

9. Za jakých okolností je blokový kód hospodárnější než nerovnoměrný?

Blokový (rovnoměrný) kód může být hospodárnější (nebo alespoň stejně hospodárný) než nerovnoměrný pouze tehdy, je-li počet znaků roven mocnině 2 (pak nevzniká redundance vlivem nevyužitých kombinací) a jejich pravděpodobnosti výskytu ve zprávě jsou shodné.

10. Jakou redundanci a entropii by měl mít optimální kód?

Optimální (nejkratší) kód by měl mít minimální redundanci a maximální entropii.

11. Jaký je princip detekčních kódů?

Ke každému úseku dat jsou přidány další (tzv. paritní) bity, které umožňují detekovat konkrétní úsek, ve kterém nastala chyba. U jednoduché parity je doplněn jeden bit tak, aby počet jedniček v datech byl lichý, resp. sudý. U kombinované parity je použito více paritních bitů, které zabezpečují datové bity na různých pozicích.

12. V čem spočívá výhoda kombinované parity?

Kombinovaná parita používá více paritních bitů současně, nebo kombinaci příčné a podélné parity, což umožňuje nejen detekci chyby, ale i její lokalizaci a následnou opravu.

13. Co je to kontrolní součet? Které druhy kontrolních součtů se používají?

Kontrolní součet je hodnota vypočtená z celého bloku dat určitým algoritmem. Existuje mnoho variant – podélná parita, cyklický součet (CRC), otisky (hashe) apod. Každý kontrolní součet má jiné vlastnosti a možnosti použití.

14. K čemu slouží kontrolní číslice a jakým způsobem se získává?

Kontrolní číslice zabezpečuje kontrolu správnosti zadání určitého jednoznačného identifikátoru (rodné číslo, číslo účtu, IČO, VIN apod.). Získává se výpočtem, ve kterém se jako vstupní hodnoty použijí všechny zbývající číslice identifikátoru. Používají se různé algoritmy aplikující poznatky z oblasti dělitelnosti přirozených čísel.

15. Jaký je princip samoopravných kódů?

Princip je založen na existenci povolených a zakázaných kódových kombinací. Pokud příjemce obdrží zakázanou kombinaci, existuje k ní právě jedna povolená kombinace, na kterou je automaticky opravena. Data je nutné doplnit o potřebný počet kontrolních bitů tak, aby bylo možné detekovat a opravit chybu požadované násobnosti.

16. Jaké vlastnosti má opakovací kód a kde se využívá?

Opakovací (koktavý) kód posílá každý znak zprávy opakovaně n-krát. Jedná se o nejjednodušší případ samoopravného kódu. Má velkou redundanci a lze jej využít pouze tam, kde je opakované zasílání technicky realizovatelné.

17. Jak schopnosti mají Hammingovy kódy a kde se prakticky používají?

Hammingovy kódy slouží k detekci a opravě jednonásobných chyb při přenosu. Mají široké využití v různých odvětvích informatiky od hardwaru (RAM, pevné disky) přes počítačové sítě (Bluetooth, ethernet, Wi-Fi) až po satelitní komunikaci a digitální audio přenosy.

18. Jaké výhody přináší rozšířené Hammingovy kódy?

Rozšířené Hammingovy kódy obsahují navíc jeden kontrolní bit, který zabezpečuje celé kódové slovo sudou paritou. To umožňuje opravu jednonásobné chyby a současně detekci dvojnásobné chyby, nebo detekci trojnásobné chyby při přenosu.

19. Jaké vlastnosti mají perfektní kódy?

Perfektní kódy mají při daných vlastnostech minimální možnou redundanci. Mezi perfektní kódy řadíme Hammingovy kódy pro opravu jednonásobných chyb, Golayovy kódy pro opravu trojnásobných chyb a opakovací kódy pro opravu t-násobných chyb.

20. Kde se používá Grayův kód a na jakém principu je založen?

S Grayovým kódem se setkáme v digitální komunikaci (pozemní a kabelové televize, odpovídače sekundárního radaru v letadlech, inkrementální snímače polohy) nebo při grafické metodě minimalizace logických funkcí pomocí Karnaughovy mapy. Kód se vyznačuje tím, že každé dva po sobě jdoucí kódové znaky se liší změnou pouze v jedné bitové pozici.

21. Čím se zabývá kryptografie a čím kryptoanalýza?

Kryptografie se zabývá šifrovacími algoritmy a jejich implementací. Kryptoanalýza se zabývá dešifrováním bez znalosti klíče. Jedná se o dvě disciplíny kryptologie.

22. Jaký je rozdíl mezi transpoziční a substituční šifrou?

Transpoziční šifry fungují na principu záměny pořadí znaků ve zprávě. Substituční šifry jsou založeny na principu záměny znaků ve zprávě za zcela jiné znaky.

23. Na jakém principu jsou založeny moderní kryptografické metody?

Současné kryptografické metody jsou založeny na matematice. Základem všeho je šifrovací klíč, což je číselná posloupnost umožňující šifrování i dešifrování zprávy s použitím určité matematické operace.

24. Jaký je rozdíl mezi jednosměrným a obousměrným šifrováním?

Při použití jednosměrného šifrování je zprávu možné zašifrovat, ale není možné ji dešifrovat. U obousměrného šifrování lze se znalostí klíče zprávu zašifrovat i dešifrovat.

25. Čím se liší symetrické a asymetrické šifrování?

Symetrické šifrování používá pro šifrování i dešifrování stejný klíč. Asymetrické šifrování používá dva různé klíče (veřejný a soukromý).

26. K čemu slouží elektronický podpis a jak jej lze získat?

Elektronický podpis zaručuje autenticitu, integritu a nepopiratelnost zprávy. Technicky se jedná o otisk zprávy zašifrovaný soukromým klíčem odesílatele.

27. Jakým způsobem lze poslat a přijmout bezpečnou a podepsanou zprávu?

Odesílatel opatří zprávu elektronickým podpisem a obojí následně zašifruje vygenerovaným symetrickým klíčem. Symetrický klíč pak zašifruje veřejným klíčem příjemce a pošle jej spolu se zprávou. Příjemce dešifruje svým soukromým klíčem symetrický klíč, přečte si zprávu a veřejným klíčem odesílatele dešifruje otisk zprávy. Souhlasí-li zaslaný otisk zprávy s otiskem znovu vypočteným na straně příjemce, máme jistotu, že zpráva nebyla během přenosu přečtena ani modifikována neoprávněnou osobou.

6 Principy uložení dat v souborech

Úvod

Až dosud jsme se věnovali *vnitřní* reprezentaci různých dat v počítači. Víme, jak jsou reprezentována čísla, znaky, zvuky a obrazy. Nyní již máme dostatek informací k tomu, abychom se mohli věnovat *vnější* reprezentaci dat, tj. uspořádáním těchto dat v souborech a všem aspektům s tím souvisejícím.

Cíl

Cílem této kapitoly je orientace v základních souborových formátech pro různé typy dat. Ukážeme si rozdíly mezi textovým a binárním formátem dat, naučíme se vybírat optimální souborový formát pro konkrétní účel a seznámíme se s principem a důsledky konverzí souborových formátů. Dále se budeme zabývat metodami ztrátové a neztrátové komprese dat a jejich aplikací v různých formátech. Na závěr se podíváme na princip organizace souborů na disku prostřednictvím souborových systémů, které tvoří prostředníka mezi operačním systémem a daty v souborech.

Návaznosti

Pro studium této kapitoly je nutné orientovat se v možnostech vnitřní reprezentace dat, datových typech a kódování, zejména pak kódování národních znaků.

Studijní čas

Čas potřebný k nastudování této kapitoly je asi 9 hodin.

6.1 Datové formáty

Datový typ

Význam pojmu **datový typ** známe již z kapitoly 3.1 (viz str. 33). Jedná se o množinu povolených hodnot a kolekci povolených operací s těmito hodnotami. S několika datovými typy pro reprezentaci numerických hodnot jsme se poté setkali v kapitole 4 (viz str. 52 a 66) v souvislosti s celými a racionálními čísly. Určitý datový typ je často svázán s programem, který s příslušnými daty pracuje. Tím je současně svázán také s programovacím jazykem, ve kterém je napsán zdrojový text programu.

Každý programovací jazyk disponuje určitým repertoárem konkrétních datových typů, tyto typy mají svá jména, přesné definice a jsou doplněny o množinu zcela exaktně definovaných operací (operátorů, funkcí). Vzhledem k této jednoznačnosti je v praxi možné určit data tím, že řekneme, jakého jsou datového typu v určitém programovacím jazyce. Přestože každý programovací jazyk má množinu datových typů jinou, lze mezi nimi rozpoznat řadu společných rysů.

Datový formát

Data reprezentovaná určitým datovým typem musíme být také schopni uložit. Způsob uložení jednotlivých hodnot datového typu či uspořádání dat v paměti počítače vyjadřuje **datový formát**. Na rozdíl od tzv. jednoduchých datových typů tedy datový formát určuje navíc i strukturu dat. Data téhož datového typu tak mohou mít různý formát zobrazení.

Příklad 6.1

Tabulkové kalkulátory umožňují nastavení formátu zobrazení hodnot v buňkách. Tatáž hodnota (reprezentovaná stejnou binární posloupností) může být zobrazena v různých podobách. V tabulce vidíme různé formáty zobrazení hodnoty 20,851 v programu Excel z balíku Microsoft Office, obdobně to vypadá i v programu Calc z balíku Libre Office.

Zobrazení v buňce	Zvolený datový formát
20,851	obecný
20,9	číslo (jedno desetinné místo)
20,85	číslo (dvě desetinná místa)
20,851	číslo (tři desetinná místa)
21	celé číslo
20,85 Kč	měna
2085%	procenta
20 6/7	zlomky
2,09E+01	semilogaritmický tvar
20. 1. 1900	datum
20. leden 1900	datum
20:25:26	čas
20,851	text (zarovnání vlevo)
000 21	PSČ

Při běžné práci s daty se nevyhneme rozhodování o vhodném datovém formátu pro konkrétní účely. Volíme ze dvou variant podle toho, jakým způsobem jsou data zakódována:

- *binární kódování* je připraveno pro aritmetické a logické operace v paměti počítače;
- *textové kódování* umožňuje přímé čtení a editaci dat člověkem v textovém editoru.

Příklad 6.2

Předpokládejme, že potřebujeme zpracovávat údaje o osobách zahrnující jejich věk. Ten očekáváme v podobě čísla. Který datový formát ale zvolíme?

1. Věk převedeme do dvojkové soustavy a vyjádříme jako celé číslo bez znaménka na prostoru 8 bitů (i když by stačilo jen sedm). Rozsah hodnot pak bude v intervalu 0–255. S těmito čísly můžeme provádět aritmetické operace (můžeme například počítat průměrný věk osob). Například věk 26 let převedeme do binární posloupnosti `00011010`, kterou pro přehlednost vyjádříme šestnáctkově[61] jako `1A`.
2. Číslice vyjádříme samostatně jako znaky v rozšířeném kódu ASCII na 8 bitech (opět by stačilo jen sedm; viz tab. 4.13, str. 70). Číslici 2 odpovídá posloupnost `00110010` (znak #50), číslici 6 posloupnost `00110110` (znak #54). Číslo 26 zapsané tímto způsobem pak v paměti zabere 2 bajty a bude vyjádřeno šestnáctkově jako `32 36`.

[61] Výhody zápisu v šestnáctkové soustavě jsou zřejmé: úspornost zobrazení díky vyjádření každého bajtu právě dvěma číslicemi a také to, že v případě potřeby lze z tohoto tvaru snadno získat dvojkové zobrazení.

Příklad 6.3

Uvažujme podobnou situaci, kdy potřebujeme zpracovávat příjmení osob v podobě řetězců složených z jednotlivých znaků například ze znakové sady ASCII rozšířené o kód ISO 8859-2, v němž jsou obsaženy i znaky české abecedy. Každý znak pak bude zabírat 8 bitů. Který datový formát můžeme nyní použít?

1. Řetězce budou mít proměnlivou délku a zaberou tak jen tolik prostoru, kolik je nezbytně nutné. V takovém případě je ale potřebné znát skutečnou délku každého řetězce. To lze zajistit dvěma způsoby:
   a) délka řetězce bude uložena v prvním bajtu zleva, například:

	'C'	'i'	'm'	'r'	'm'	'a'	'n'
07	43	69	6D	72	6D	61	6E

   b) bude použit dohodnutý znak s významem „konec řetězce“, například:

'C'	'i'	'm'	'r'	'm'	'a'	'n'	
43	69	6D	72	6D	61	6E	00

2. Řetězce budou mít vyhrazen pevně daný prostor bez ohledu na to, kolik znaků se v nich skutečně nachází. Neobsazená políčka budou doplněna dohodnutými znaky, například mezerami:

'C'	'i'	'm'	'r'	'm'	'a'	'n'	' '	' '	' '
43	69	6D	72	6D	61	6E	20	20	20

Z předchozích příkladů je zřejmé, že jedny a tytéž údaje mohou být uloženy mnoha způsoby. Představme si nyní, že bychom chtěli tyto údaje vidět například na terminálu počítače nebo na tiskárně. Který ze způsobů uložení věku a příjmení popsaných v příkladech 6.2 a 6.3 můžeme poslat tiskárně?

Tiskárna i terminál počítače pracují tak, že přijmou jeden bajt a ten buď chápou jako kód zobrazitelného znaku (pak jej zobrazí, resp. vytisknou), nebo jako kód řídicího znaku (pak provedou povel, který je tímto znakem nařízen).

Pokud bychom tiskárně poslali věk v podobě čísla 1A, tiskárna by si oprávněně myslela, že kód 26 je nějaký řídicí znak a provedla by nějakou akci (ta je bohužel skoro u každého zařízení jiná). Podobně špatně by dopadlo zaslání příjmení ve tvaru řetězce s proměnlivou délkou upřesněnou prvním bajtem ve tvaru 07. Tiskárna by obdržela kód 7, což je povel pro zvukový signál (pípnutí).

Pro viditelné zobrazení znaků tedy musíme údaje interpretovat nějak jinak. Informace, která má být viditelná, musí být zobrazena jako posloupnost těch správných zobrazitelných znaků, kromě toho ještě musíme jednotlivé údaje od sebe odlišit, aby nesplynuly, tj. musíme mezi ně vložit nějaké oddělovače.

Příklad 6.4

Chceme vytisknout údaje o osobě na řádku tak, aby příjmení a věk byly zarovnány pod sebou. Přechod na další řádek zajistí znak pro konec řádku. Pro korektní práci s údaji je nutné zajistit, aby příjmení a věk byly zpracovány ve dvou odlišných formátech.

1. V operační paměti budeme mít příjmení uloženo jako řetězec zakončený dohodnutým znakem a věk jako číslo:

'C'	'i'	'm'	'r'	'm'	'a'	'n'		26
43	69	6D	72	6D	61	6E	00	1A

2. Na tiskárnu pošleme příjmení v podobě řetězce pevné délky a věk v podobě znaků zastupujících jednotlivé číslice:

'C'	'i'	'm'	'r'	'm'	'a'	'n'	' '	' '	' '	'2'	'6'
43	69	6D	72	6D	61	6E	20	20	20	32	36

Formátová specifikace

Přesný popis formátu uložení dat, tedy význam jednotlivých bitů či bajtů reprezentujících jednotlivá data, definuje **formátová specifikace**.

Příklad 6.5

Formátová specifikace použitých formátů z příkladu 6.4 může vypadat například takto:

1. Příjmení je tvořeno řetězcem s předem neznámým počtem znaků v kódu ASCII s rozšířením ISO 8859-2, kde každý znak zabírá 1 bajt. Řetězec je ukončen oddělovacím znakem s hodnotou 00. Následuje věk uložený binárně jako celé číslo bez znaménka, které zabírá právě 1 bajt.
2. Příjmení je tvořeno řetězcem 10 znaků v kódu ASCII s rozšířením ISO 8859-2, kde každý znak zabírá 1 bajt. Je-li příjmení kratší než 10 znaků, je zprava doplněno mezerami. Následuje věk zapsaný pomocí znaků základního kódu ASCII tak, že každá jeho číslice zabírá 1 bajt.

Porovnáme-li obě možnosti zobrazení použité v příkladech 6.2, 6.3 a 6.4, zjistíme, že údaje se stejným významem mohou mít nejen diametrálně odlišný tvar zobrazení, ale také odlišné vlastnosti. V prvním případě jsou data připravena v podobě vhodné ke zpracování, výpočtům apod., ale mohou zabírat jiný prostor (někdy větší, jindy menší) a nejsou přímo čitelná. Druhý případ tvoří svými vlastnostmi doplněk případu prvního. Uvedené možnosti zobrazení nebyly vybrány náhodně. Jedná se o představitele dvou různých skupin metod zobrazení dat v paměti počítače (obr. 6.1).

Textový formát

Jsou-li data vhodná pro zobrazení a přímé čtení člověkem, jedná se o **textový formát**. Intuitivně bychom se mohli domnívat, že všechny prvky tohoto formátu musejí být složeny výhradně ze zobrazitelných znaků. Potom by ale soubor s takovými daty nemohl mít více než jeden řádek, protože znak pro konec řádku je z kategorie řídicích znaků. Intuitivní pracovní definici textového formátu proto musíme upravit tak, že všechny prvky textového formátu jsou složeny ze zobrazitelných znaků, mezi nimiž mohou být jako oddělovače použity konce řádků a na konci dat může být nejvýše jeden znak konce souboru.

Binární formát

Jsou-li alespoň některé prvky formátu řešeny jiným způsobem (tj. řídicími znaky), pak se jedná o **binární (netextový) formát**. Data v tomto formátu nejsou přímo čitelná člověkem[62] a jsou připravena pro aritmetické a logické operace v paměti počítače.

```
\kapitola{Přenos, kódování a~zabezpečení dat}

\uvod{Úvod}
Po prostudování předchozích kapitol již máme poměrně
jasnou představu o~tom, jakým způsobem jsou data různého
typu reprezentována v~paměti počítače. Obvykle v~této
souvislosti máme na mysli operační paměť, v~dalších
kapitolách se budeme zabývat možnostmi reprezentace dat
v~souborech. Než se ale k~této problematice dostaneme,
budeme se věnovat přenosu dat a~souvisejícím otázkám
jejich kódování a~zabezpečení.
```

```
%□□□□
10 0 obj
<<
/Filter/FlateDecode
/Length 471
>>
stream
x□T□n°1o□□°□o)$EQ√□□P )□□□mE□□□ui□L□□□□□□-j□°!=□□□G□□W @□*□□□
□,!閭□□7o□       (GdqºAk□□□%R])x°□By?*°□d5i°□Kf#□F4□□□□HN□G□□□;
ə□□□□i`□□□cV9□□□□□□c□-□c□ÞS□□6:□°°□*□□□□J□□□□X□°°*°°,
endstream
```

Obrázek 6.1: *Ukázka dat v textovém (vlevo) a binárním (vpravo) formátu*

Konec řádku a souboru

V souvislosti s textovým formátem dat je nutné poznamenat, že řídicí znaky pro konec řádku a konec souboru se liší podle použitého operačního systému. V oblasti osobních počítačů se lze v současnosti setkat se dvěma typy operačních systémů a každý z nich řeší problém po svém (tab. 6.1). V tabulce jsou použity zkratky řídicích znaků (viz také tab. 4.13, str. 70): LF (*Line Feed*) = posuv o řádek; CR (*Carriage Return*) = návrat vozíku pracovního stroje; EOT (*End Of Transmission*) = konec textu; ESC (*ESCape*) = únik.

Tabulka 6.1: *Znaky pro konec řádku a souboru v různých současných operačních systémech*

Operační systém	Konec řádku	Název znaku	Konec souboru	Název znaku
Unix, Linux, macOS	0A	LF	04	EOT
Windows	0D 0A	CR LF	1A	ESC

Odlišné tvary konců řádků způsobují uživatelům potíže. Představme si, že bychom chtěli soubor s konci řádků podle jednoho operačního systému zobrazit pomocí aplikace podporující konce řádku pouze podle druhého operačního systému. Výsledek je znázorněn na obr. 6.2.

soubor se správně interpretovaným znakem konce řádku	soubor s koncem řádku typickým pro OS typu Unix zobrazený pod OS Windows	soubor s koncem řádku typickým pro OS Windows zobrazený pod OS typu Unix
Praha Brno Plzeň	Praha■Brno■Plzeň	Praha■ Brno■ Plzeň

Obrázek 6.2: *Zobrazení textového souboru s chybnou interpretací konců řádků*

Protože každý operační systém implementuje znak konec řádku v jiné podobě, je teoreticky možné podle podoby tohoto znaku zjistit, pro který operační systém byl soubor připraven. Kvalitnější textové editory mohou ale často zpracovávat různé typy konců řádků a také je bez obtíží změnit, takže tato metoda není zcela spolehlivá.[63]

[62] Zajímavým paradoxem je znak tabulátor. Soubor s takovým oddělovačem je jistě čitelný člověkem, ale de facto není textový, neboť tabulátor je řídicím znakem s kódem 09. Důvod zařazení tabulátoru mezi řídicí znaky lépe pochopíme, když si uvědomíme jeho chování v různých situacích. Zejména pro jeho využití jako oddělovače bývá často považován za další povolený řídicí znak v textovém formátu.

[63] Pro úplnost dodejme, že Poznámkový blok (v OS Windows 10) po 25 letech konečně dokáže správně interpretovat soubor s unixovými konci řádků. Neumožňuje však tvar konců řádků změnit.

Kontrolní otázky

1. Jaký je rozdíl mezi datovým typem a datovým formátem?
2. Co určuje formátová specifikace?
3. Co musí být splněno, aby byla data v textovém formátu?
4. Jak je v textových souborech implementován konec řádku?

6.2 Souborové formáty

Textové i binární formáty uložení dat mají široké praktické využití a setkáme se s nimi na každém kroku. Každý program, který používáme, pracuje s daty nejen v operační paměti, ale také v souborech na discích a jiných paměťových zařízeních. Ze souborů jsou data čtena, program je zpracovává a do souborů opět vypisuje.

Souborový formát

Pojem **souborový formát** je odvozen od pojmu *datový formát* a vyjadřuje přesný popis způsobu uložení dat v souborech. Soubory, které uchovávají data v textovém formátu, nazveme textové. Soubory uchovávající data i v netextovém formátu nazveme binární.

Datové soubory programů určitého zaměření mohou být často v obojím formátu – formáty mají ovšem rozdílné vlastnosti (viz tab. 6.2) a doplňují se, což zvyšuje možnost použití daného programového vybavení a pro praktickou práci má zásadní význam.

Tabulka 6.2: *Srovnání vlastností textových a binárních souborových formátů*

Textový formát	**Binární formát**
Všechna data jsou přímo čitelná člověkem	Alespoň část dat není přímo čitelná člověkem
Pro zpracování je obvykle nutná změna podoby dat	Vhodný pro přímé zpracování
Vysoká přenositelnost, lze zpracovat mnoha obecnými programy a službami operačního systému	Nízká přenositelnost, závisí na určitém programu, který zná přesnou a detailní strukturu dat
Vysoká odolnost vůči porušení či ztrátě informací	Náchylné k chybám, poškození často nelze opravit
Není napadnutelné virem (virus se nemá kde ukrýt)	Napadnutelné virem, lze ukrýt mnoho informací
Většinou nižší úspornost zobrazení	Někdy maximálně úsporný (ale i naopak)

Způsobů uložení informací (datových formátů) je používáno velké množství. Pro usnadnění odhadu, o jaký formát dat se jedná, se místo rozsáhlé a složité specifikace formátu obvykle používají jména formátů – JPEG, DOCX, BMP, PDF, HTML a mnoho dalších, s nimiž se postupně seznámíme.

Otevřený formát

Souborové formáty můžeme rozdělit do dvou skupin. **Otevřené (svobodné) formáty** se vyznačují volně dostupnou specifikací, což umožňuje efektivní zpracování uložených dat. Takové formáty jsou pak ideálním prostředkem pro výměnu informací. Příkladem jsou všechny textové formáty a z těch binárních JPEG, PNG, PDF apod.

Uzavřený formát

Uzavřené (proprietární) formáty se naproti tomu vyznačují utajovanou specifikací, což umožňuje získat monopol pro jejich zpracování a zároveň silně omezuje možnosti využití uložených dat. Typickým příkladem je souborový formát CDR.

Textové soubory můžeme dále rozdělit také podle toho, zda mohou obsahovat národní znaky. Obsahuje-li textový soubor pouze znaky z dolní poloviny ASCII tabulky (tj. na pozicích do #126 včetně), označíme takový formát jako **čistý (prostý) text** (*plain text*).[64] Tento formát neobsahuje žádné národní znaky, nemohou proto nastat problémy s nekorektním zobrazením obsahu souboru (viz příklad 6.6). Takové řešení je sice elegantní, avšak bohužel v praxi méně časté.

Čistý text

Příklad 6.6

Přestože čeština používá 30 národních znaků, dokážeme se bez nich obejít. Důkazem budiž fiktivní dopis mistra Jana Husa (* cca 1369, † 1415), kterým se omlouvá českému národu za zavedení nabodeníček a ukazuje cestu, jak se jim vyhnout. Celý text dopisu je složen výhradně ze znaků bez diakritiky:

Krajani a druzi moji,

ach, smutno mi je, trudno, neveselo, chmurami jsem zavalen, hanbou jsem zdrcen a jest mi za co pykat. Maje na mysli jen dobro (o tom snad nikdo nepochybuje), bez rozmyslu jsem vnesl do jazyka velkou lotrovinu, a tak jsem zavinil historickou nehodu. Dlouho se to tutlalo. Teprve tato epocha naplno vyjevila celou moji vinu, drzost a zradu. Uznal jsem to, beru to na sebe a, posypav popelem, kaju se. To v troufalosti ducha a usilovav pouze o to, aby bylo lze rychleji rozmluvy, ano i knihy, skripta a lejstra zapisovati, vymyslel jsem potrhlou a krutou fintu. Brkem z husy jsem litery a, c, d, e, i, n, o, r, s, t, u, y, z pobodal a zle poranil.

„Jsou to jako bobky od much, co nad linkami ulevily si v letu, fuj!“ podotkli k diakritice kritici z univerzity, ledva ohrnuli nos. Pravopis jazyka jsem tak onehdy podle pochroumal. Tehdy se lid proti tomu moc neohrazoval a vzal to celkem s klidem. Ale myslel si svoje a brblal jako dnes: „Pche, zase jeden trouba, aby se zviditelnil, reformu navrhl. Ale co bychom s panstvem vedli spor a dlouze darmo blafali. Zase holt dali shora befelem akademickou stolici nedoukovi a my si to zase odneseme. Aby ho kat spral, magora!“

A tak se tak najednou psalo a postupem doby se z toho triku zvyk stal. Co jsem to provedl! Jejda, to jsem tomu dal, to jsem to zpackal! Zaujat plky z kazatelny o Bohu, do toho kvalt a doby shon, bez koncepce, bez projektu, bez ohledu na normalizaci, nedbaje na dopady, zavrhl jsem kompatibilitu s latinskou abecedou. Byl to ode mne podraz, omyl, chyba, hloupost a nedbalost.

Omluvou mi snad je, dejme tomu, jen moje neznalost toho, kudy se pokrok v budoucnu bude pohybovat. Nemaje potuchy, co jednou bude ASCII znamenat, znesnadnil jsem vlasti vstup do Evropy a zatarasil jinak pohodlnou cestu do e-mailu i do Internetu. Pomni jen ty zmatky bez konce kolem klaviatur anebo reprezentace ISO Latin 2 nebo toho, co vymysleli ti dva sourozenci – jak se jenom jmenovali – ti, co po nich kdekdo hodil kamenem (viz str. 69). Za ten skutek by krajani-informatici i po letech dozajista souhlasili s onou klatbou, kterou na mou hlavu koncil uvalil, a osudem, co mi ten zrzoun Zikmund podfukem v Kostnici v plamenech uchystal. A nejeden znalec by bezpochyby i dnes na tu hranici svoje poleno donesl a nadto by mi beze studu na holou zadnici s vervou nasekal.

Ouha, ale co dnes poradit vlastenci, chce-li cosi kamsi poslat a je nucen oholit znaky o diakritiku, slova zoufale zohavit, smysl zprznit a pochroumat tak jazyk, jindy pln kouzla a vtipu? Uznejte sami: Tato mluva sama plyne sladce ze srdce, je to vskutku drahocennost, dozajista vlasti poklad a lidu chlouba. Ale mnohdy je to i potvora a mrcha jako had s jedem v zubech. Navenek i ve skrytu je jazyk pln vtipu, schopen kroutit se kdykoli pro radost i z nutnosti jak z vrby proutek. Ohnout svede ho perem nejen akademik, mudrc a pisatel, ale i hubou halama, moula, komik i floutek.

Dumal jsem tuze dlouho, zda mohu malinko napravit, co jsem v minulosti zpackal, a lze-li vyklouznout z pasti diakritiky.

A ejhle – z toho kruhu se vymanit lze! Ten zapletenec jsem rozmotal – inu, za jistou nevelkou cenu. Vynech, zanedbej, obejdi, rovnou do pekel uvrhni ta slova, co jsem zbabral, a spisuje cos, vol jen ta, co zbyla mnou netknuta! Neboj se a nelekej, je jich kupodivu spousta! Vylov si je z mozku, vychytej z knih, zaslechni je v proudech slov na ulici, na rynku i v pajzlu, i tam v rozhlase, naslouchej, po libosti se slovy si hrej, ale dbej, abys nemluvil a nepsal jako robot, co nikdy nenabyl ducha a citu! S rozmyslem a odvahou cpi ta slova na listy, do stroje i do e-mailu! Tahleta nemohou vystrkovat chocholky diakritiky, budou v ASCII hezky zabalena, a proto se jim cestou nic nestane. Linkami, rourami, interprety a editory lehce prolezou a neoloupeny o smysl toho, cos napsal, donesou vzkazy kamkoli bez vady a kazu.

Pak bude na odiv, co za kejkle tenhle jazyk dovede – a o tom je samosebou i tento list. Jeho obmyslnou podstatu jste dozajista hnedlinko zkraje prokoukli, nebo snad ne?

S pozdravem, pokorou a notnou omluvou

Mistr Jan

[64] Mnoho zdrojů užívá pojem *plain text* i v jiné souvislosti. Často se lze setkat s rozdělením textových souborů na *neformátované* (ty bývají označovány jako plain text) a *formátované* (těm se říká dokumenty a skládají se z vlastního textu a formátovacích značek).

Rozšířený text

Obsahuje-li textový soubor i národní znaky (tj. znaky od #128 výše nebo ve vícebajtovém kódování), označíme takový formát jako **rozšířený text** (*extended text*). Pro správné zobrazení obsahu souboru je obvykle nutné upřesnit kódování národních znaků. Takové řešení je velmi časté a uživatelům způsobuje četné problémy.

Národní znaky způsobují komplikace také při výběru vhodné rodiny písma v dokumentech. Existuje mnoho pěkných písem, která ale neobsahují všechny potřebné znaky a symboly, a proto je nemůžeme použít. Poznáme to snadno tak, že chybějící národní znak v daném fontu je obvykle nahrazen národním znakem z jiného fontu. Chceme-li si být naprosto jisti, že zvolený font podporuje všechny české znaky, můžeme použít tzv. pangram.[65]

6.2.1 Asociace formátů a aplikací

Rozšíření souboru

Pro usnadnění orientace v množství souborových formátů se místo rozsáhlé a složité specifikace konkrétního formátu doplňují jména souborů o **rozšíření**v (*extension*, příponu), které je od jména souboru odděleno tečkou. Obvykle používáme přípony odpovídající konkrétním souborovým formátům, například `.jpg` nebo `.jpeg` pro soubor s rastrovými grafickými daty, `.htm` nebo `.html` pro webové stránky, `.mp3` pro komprimovaná zvuková data apod.

Je však potřeba si uvědomit, že *rozšíření souboru automaticky neurčuje jeho formát,* ale právě naopak – podle formátu souboru lze určit vhodné rozšíření jeho jména. V žádném případě ovšem nelze jen podle rozšíření rozpoznat formát dat uložených v souboru. Pravidla pro rozšíření nejsou přesně dána a nikdo nám nemůže zabránit v tom, abychom si pojmenovali náš soubor zcela libovolně. Pouze si musíme pamatovat, co je v tomto souboru uloženo. Když takový soubor někomu předáme a neřekneme, co je jeho obsahem, příjemce nebude na první pohled vědět, co s ním může udělat. V unixových systémech dokonce soubory ani rozšíření mít nemusejí.

V operačních systémech firmy Microsoft je každému souboru zcela mechanicky přisouzen formát výhradně podle rozšíření jeho jména. Je pochopitelné, že netrefíme-li se příponou do správného formátu, pravděpodobně nepůjde soubor běžným způsobem zpracovat, což není příliš inteligentní. Navíc si uživatel může k určitému rozšíření zvolit zpracovávající program – při chybné volbě je pravděpodobné, že se data budou interpretovat nesprávně.

Princip asociace

Uvedeným problémům se lze vyhnout dodržováním určitých pravidel. V operačním systému usnadňuje uživatelům práci tzv. **asociační tabulka**.[66] V ní můžeme vidět všechna rozšíření, která do ní byla vložena buď již při instalaci samotného systému, nebo při instalaci jednotlivých programů. U každého rozšíření můžeme sledovat, co přibližně znamená a jaký program bude schopen zpracovávat soubor tohoto formátu. Přiřazení vhodného programu

[65] Pangram (z řeckého παν γράμμα, každé písmeno) je věta či úsek textu obsahující všechna písmena abecedy. Jedná se zpravidla o slovní hříčku, jejímž obsahem je co nejkratší text s touto vlastností. V angličtině je typickým pangramem věta „The quick brown fox jumps over a lazy dog“, v češtině „Nechť již hříšné saxofony ďáblů rozzvučí síň úděsnými tóny waltzu, tanga a quickstepu“. Mnohem známější je ale varianta zaměřená pouze na výskyt znaků s diakritikou, například věty „Příliš žluťoučký kůň úpěl ďábelské ódy“, „Loď čeří kýlem tůň obzvlášť v Grónské úžině“ a mnoho dalších.

[66] Tabulku lze najít v nastavení systému, konkrétní umístění se však liší: Start → Nastavení → Systém → Výchozí aplikace → Zvolit výchozí aplikace podle typu souboru (Windows 10). V některých linuxových distribucích s prostředím KDE je přístup k tabulce jednodušší: Nastavení systému → Výchozí aplikace, příp. Asociace souborů (Kubuntu).

k danému rozšíření budeme nazývat *asociace aplikace*. Ke každému rozšíření je možné v tabulce přidat jedinou aplikaci. Pokud toto přiřazení nevyhovuje, je možné nahradit přiřazenou aplikaci jiným programem. K této operaci je však nezbytné mít odpovídající oprávnění.

Je-li k danému rozšíření provedena asociace, můžeme pak aktivací souboru v libovolném místě spustit asociovanou aplikaci, v níž se soubor otevře. Aktivace může proběhnout různými způsoby, například:

- stažením souboru přes webový prohlížeč;
- dvojklikem v souborovém manažeru;
- výběrem v dokumentech.

Možnosti asociace

Orientace jen podle přípony se může laikům jevit jako usnadnění práce, ale v praxi často vede ke zmatkům, které zpracovávání dat naopak komplikují. Přidání jediné aplikace k danému rozšíření je totiž jen jednou z variant, které v praxi mohou nastat. Podívejme se na všechny možnosti vztahů mezi formátem a aplikací:

- *1 formát ⟶ 1 aplikace.* K danému formátu existuje právě jedna vhodná aplikace. Jedná se o zcela ideální stav bez jakýchkoliv problémů, avšak v praxi málo realizovatelný. Lze jej najít u nativních formátů grafických editorů. Příkladem může být asociace mezi formátem CDR a aplikací CorelDRAW, formátem CPT a aplikací Corel Photo-Paint (ze stejného softwarového balíku), formátem PSD a aplikací Adobe Photoshop nebo mezi formátem XCF a volně dostupným rastrovým grafickým editorem GIMP.
- *Více formátů ⟶ 1 aplikace.* Asociace je často provedena s jednou aplikací, chceme-li však soubor zpracovat aplikací jinou, v daném případě vhodnější, nemůžeme to provést jednoduchou aktivací souboru, ale spuštěním daného programu a v něm otevřením daného souboru. Protože asociační tabulka se aktualizuje vždy při instalaci nějakého programu, je asociace provedena s tou aplikací, která byla nainstalována jako poslední. Je tedy vhodné změnit asociaci tak, aby byl s daným rozšířením automaticky aktivován program, který budeme používat nejčastěji. Typickým příkladem je asociace formátů DOC, DOCX a RTF s textovým procesorem Microsoft Word nebo formátů XLS, XLSX a XLSM s tabulkovým kalkulátorem Microsoft Excel.
- *1 formát ⟶ více aplikací.* Problematický případ a nepříjemné řešení, které vede ke spuštění posledně instalované aplikace, nebo k výběru aplikace z nabídky. Příkladem může být asociace formátu JPEG s aplikacemi Malování, Adobe Photoshop, Corel Photo-Paint a mnoha dalšími v závislosti na tom, zda a v jakém pořadí jsou v systému instalovány.
- *1 formát ⟶ žádná aplikace.* K danému rozšíření neexistuje žádná asociace – při pokusu aktivovat soubor systém požádá o výběr vhodné aplikace, a to buď nainstalované v daném počítači, nebo dostupné v internetovém prostoru. Pokud nemáme tušení, o jaký formát souboru se jedná, prakticky nemáme šanci vhodnou aplikaci vybrat. Tento případ nastává často tehdy, obdržíme-li soubor vytvořený v programu, který není v našem počítači nainstalován.
- *Žádný formát ⟶ 1 aplikace.* K určité aplikaci neexistuje žádné asociované rozšíření. Tento případ buď vůbec nevadí (například kalkulačka nezpracovává žádné soubory), nebo ukazuje na chybu instalace nebo uživatele, kdy instalovaný program pozbyl položku v asociační tabulce.

Přenositelnost formátu

Nyní již můžeme zavést pojem **přenositelnost formátu**, který budeme pracovně definovat jako množství programů schopných zpracovat tento formát. Důležitým faktorem je podpora zpracování formátu v různých operačních systémech. Přenositelnost je také úzce svázána s otevřeností formátu, ale závisí také na majiteli formátu (srovnejme např. formáty DOCX, ODT a PDF). Přenositelnost textových formátů je obecně daleko větší, binární otevřené souborové formáty jsou rovněž přenositelné.

6.2.2 Rozpoznávání souborových formátů

Poměrně běžný je případ, kdy musíme pracovat se souborem, který nelze žádnou aplikací otevřít. To může být způsobeno například nenávratným poškozením souboru nebo chybějícím záznamem o daném formátu v asociační tabulce, což způsobí, že náš operační systém takový formát „nezná“.

Mnohdy úplně stačí, když se uživatel při pojmenování souboru dopustí překlepu a místo rozšíření `.docx` použije třeba `.doxc`. Prakticky žádný operační systém si pak s takovým souborem v grafickém uživatelském rozhraní nedokáže sám poradit. Můžeme se však sami pokusit zjistit, o co se jedná.

Hlavička formátu

Většinu souborových formátů lze spolehlivě identifikovat podle **hlavičky** (*file header*). Ta bývá zapsána na několika počátečních bajtech souboru. Mnoho běžně používaných formátů má jednoznačnou identifikaci, u některých formátů lze alespoň identifikovat příbuznou skupinu. Inteligentní software (včetně operačního systému) se při práci se soubory orientuje právě podle hlavičky. Rozšíření souboru pak ztrácí smysl a může být libovolné.

Postup v Unixu

Pracujeme-li v operačním systému třídy Unix, máme situaci výrazně jednodušší. Můžeme použít standardní příkaz `file`, který analyzuje několik počátečních bajtů souboru (obvykle postačí 512), v nichž je uložena hlavička formátu. Na základě zjištěných informací se příkaz pokusí určit, o který formát se jedná. První verze byla vytvořena již v roce 1973 a od té doby je neustále aktualizován s ohledem na současné souborové formáty i moderní způsoby kódování národních znaků. Příkaz disponuje obsáhlým seznamem formátů, rozpoznávání proto bývá často úspěšné. Pokud ne, lze alespoň poznat, zda se jedná o soubor textový či binární. Podle toho lze následně zvolit další kroky.

Pro rozpoznání kódování národních znaků lze použít také velmi zdařilý program `enca`, který v roce 2000 vytvořili čeští vývojáři.[67] Další možností je konvertor `cstocs`[68] použitelný pro češtinu a slovenštinu.

Postup ve Windows

V operačním systému Windows je situace složitější, protože zde chybí nástroj, který by alespoň vzdáleně připomínal příkaz `file`. Ani zobrazení vlastností souboru nám v tomto případě nepomůže, neboť systém místo analýzy obsahu souboru vychází z použitého rozšíření. Musíme proto využít znalostí získaných z předchozích kapitol a postupovat takto:

1. Zjistíme, zda se jedná o soubor textový či binární (viz str. 140). K tomu můžeme využít příkaz `type`, který spustíme z příkazového řádku, příp. se můžeme pokusit otevřít soubor v Poznámkovém bloku (Notepad) či podobném textovém editoru.

[67] Autorem programu je David Nečas (* 1976) z Masarykovy univerzity v Brně, udržuje jej Michal Čihař (* 1980). Název je zkratkou sousloví *Extremely Naive Charset Analyser*.

[68] Autorem původní implementace pro Unix je Jan Kasprzak (* 1974) z Masarykovy univerzity v Brně, modul pro Perl vytvořil Jan Pazdziora (* 1974).

2. Jedná-li se o rozšířený textový formát (viz str. 144), můžeme rozpoznat kódování národních znaků (viz str. 69) a podle použitých znaků pro konce řádků také operační systém, v němž soubor pravděpodobně vznikl (viz str. 141). Tak můžeme získat další užitečné informace.
3. U binárních formátů je situace velmi problematická. Někdy pomůže otevření souboru v textovém editoru a průzkum několika prvních znaků (bajtů) hlavičky, kde bývá název formátu uveden v textové podobě. Prakticky tím simulujeme činnost unixového příkazu `file`. Někdy je nutné odhadnout, o jaký soubor by se mohlo jednat, a zkusit jej otevřít v některých vybraných programech. Jsme ovšem omezeni tím, které programy máme nainstalovány.

Analýza přes web

Poměrně novou alternativou je analýza souboru přes webové rozhraní. Přes formulář na webových stránkách `http://checkfiletype.com/` stačí nahrát soubor a počkat na odpověď serveru. Úspěšnost rozpoznání správného souborového formátu je poměrně vysoká.

6.2.3 Základní formáty a jejich vlastnosti

V praxi se používají stovky souborových formátů, z nichž některé jsou velmi specifické. Formáty se navíc neustále vyvíjejí, a tak nemůžeme mít přehled o všech. Uveďme si proto alespoň stručný přehled často používaných formátů a jejich významných vlastností.

Formáty pro text

V kategorii zpracování textů máme k dispozici rozmanitou skupinu souborových formátů. Setkáváme se s nimi prakticky denně a zdaleka neslouží jen k ukládání textu, ale také pro další objekty. Dokumenty v těchto formátech lze prohlížet v mnoha různých programech, které jsou obvykle dostupné zdarma. Za zmínku stojí tyto otevřené formáty:

- **TXT** (*Text Format*) – nejjednodušší formát obsahující pouze text bez formátovacích značek, možnostmi srovnatelný s psacím strojem. Je čitelný prakticky v libovolném textovém editoru při správném nastavení kódování národních znaků (viz kap. 4.2, str. 67) a tvaru konce řádků (viz tab. 6.1, str. 141).
- **DOC** (*Word Document*) – binární formát firmy Microsoft, který se postupem času bohužel stal velmi oblíbeným pro ukládání i výměnu informací. Relativně správně je však zpracovatelný pouze ve Wordu a i zde je třeba dávat pozor na odlišnosti jednotlivých verzí. S menší či větší úspěšností je zpracováván i alternativními kancelářskými programy. S nástupem formátu DOCX jeho využití postupně klesá.
- **DOCX** (*Word Open XML Document*) – nástupce formátu DOC ve Wordu od verze 2007. Je binární, ale jde o komprimovaný soubor, jehož vnitřní struktura je tvořena převážně textovými soubory ve formátu XML, příp. dalšími potřebnými soubory (obrázky).
- **DOCM** (*Word Open XML Macro-Enabled Document*) – podobný jako DOCX, ale může navíc obsahovat vložená makra, která lze v dokumentu použít.
- **RTF** (*Rich Text Format*) – textový formát firmy Microsoft, použitelný v procesu automatizovaného zpracování dokumentů (například při generování výstupů z aplikací). Nemůže obsahovat makra a nelze jej chránit heslem, ale nemůže obsahovat viry.
- **ODT** (*OpenDocument Text*) – podmnožina skupiny binárních formátů ODF pro ukládání a výměnu dokumentů vytvořených kancelářskými aplikacemi. Je určen pro volně dostupné kancelářské balíky (např. OpenOffice.org), vychází ze standardu XML. Když Microsoft „vyvíjel“ formáty DOCX, XLSX a PPTX, pouze okopíroval tento princip.

- **HTML** (*HyperText Markup Language*) – jeden z nejrozšířenějších textových formátů používaný na webu. Má jednoduchou strukturu umožňující snadné vytvoření, editaci v libovolném editoru a lze jej i snadno automaticky generovat. Vyjadřovací možnosti mohou být podpořeny kaskádovými styly, které nastavují vzhled dokumentu.
- **XML** (*Extensible Markup Language*) – důležitý textový formát, který není určen ani tak pro přímé uložení dat a jejich zpracování některou aplikací, ale jeho výhoda spočívá v automatizovaném zpracování. Jsou na něm založeny moderní formáty Open Document (standard ODF – formáty ODT, ODS, ODP apod.), Office Open XML (formáty DOCX, XLSX, PPTX, MDBX), ale také třeba formát SVG pro vektorovou grafiku. V ČR se využívá také pro komunikaci se státní správou (kontrolní hlášení k DPH, výkazy pro správu sociálního zabezpečení apod.).
- **PS** (*PostScript*) – textový formát, který je spíše programovacím jazykem popisujícím vzhled stránky určené k tisku. Popis stránky není závislý na zařízení, formát se proto vyznačuje dobrou přenositelností.
- **PDF** (*Portable Document Format*) – standard firmy Adobe pro ukládání a zpracování dokumentů, snadno přenositelný, nezávislý na hardwaru i softwaru. Vychází z formátu PS, ale pro minimalizaci objemu obsahuje datové objekty v komprimované podobě, a proto je binární. K prohlížení lze použít volně dostupné programy, v současné době jej dokáže přímo zobrazit většina webových prohlížečů.

Tabulkové formáty

V oblasti zpracování tabulkových dat je situace poněkud jednodušší. Vybírat můžeme především z těchto otevřených formátů:

- **XLS** (*Excel Spreadsheet*) – binární formát firmy Microsoft určený pro tabulkový procesor Excel. Má podobné vlastnosti jako formát DOC, lze jej zpracovávat i ve volně dostupných tabulkových kalkulátorech.
- **XLSX** (*Office Open XML Workbook*) – nástupce formátu XLS v Excelu 2007. Podobně jako u formátu DOCX se jedná o komprimovaný soubor se součástmi dokumentů ve formátu XML, případně s dalšími soubory, **XLSM** může navíc obsahovat makra.
- **ODS** (*OpenDocument Spreadsheet*) – podmnožina skupiny binárních formátů ODF pro volně dostupné kancelářské balíky, vychází se standardu XML.
- **CSV** (*Comma-separated Values*) – textový formát, který nemusí mít ani hlavičku, jeho název je odvozen od struktury. Uspořádání dat připomíná tabulku, ve které každý řádek obsahuje data k jednomu záznamu. Záznam se skládá z položek, které jsou vzájemně odděleny čárkou (nebo jiným dohodnutým znakem), tím vznikají „sloupce“ tabulky dat. Používá se pro přenos dat (např. mezi databází a tabulkovým kalkulátorem) a pro jejich automatizované zpracování.

Rastrové formáty

V oblasti zpracování grafických dat je situace poněkud nepřehledná. Zejména musíme rozlišit rastrové a vektorové formáty. Chceme-li pracovat s rastrovou grafikou, vystačíme si obvykle s otevřenými binárními formáty, z nichž stojí za povšimnutí tyto:

- **BMP** (*Windows Bitmap*) – jednoduchý, dobře přenositelný formát, který při ukládání obrazových dat obvykle nepoužívá žádnou kompresi. Velikost výsledného souboru tak lze snadno odvodit (viz str. 80), je však nutné ještě připočítat velikost hlavičky formátu, která se liší podle verze a použité bitové hloubky. Kvůli vysoké paměťové náročnosti je tento formát zcela nevhodný pro použití na webu.

- **PCX** (*PC Paintbrush File Format*) – licenční formát pro Malování, postupně se rozšířil i do jiných aplikací. Dnes málo využívaný z důvodu nízké účinnosti komprese.
- **JFIF** (*JPEG File Interchange Format*) – oblíbený formát, který je znám spíše pod označením **JPEG** (*Joint Photographic Experts Group*), což je název použité metody ztrátové komprese (viz str. 160). Je ideální pro ukládání fotografických snímků nebo maleb realistických scenérií s hladkými přechody v tónu a barvě.
- **GIF** (*Graphics Interchange Format*) – formát využívající metodu neztrátové komprese, která byla až do roku 2004 chráněna patentem. Je použitelný pro obrázky, které obsahují nízký počet barev (loga, grafy apod.), a také pro malé animace.
- **PNG** (*Portable Network Graphics*) – formát vytvořený jako náhrada licenčního formátu GIF. Nabízí více barev, možnost průhlednosti (model RGBA, viz str. 78) a lepší neztrátovou metodu komprese, nepodporuje animace. Je ideální pro jednoduché obrázky (ikony, navigační prvky, loga apod.), není vhodný pro fotografické snímky.
- **TIFF** (*Tagged Image File Format*) – kontejnerový formát, který dokáže nést různé obrázky uvnitř jednoho souboru, používat různé barevné hloubky, metody komprese apod. Je také neoficiálním standardem pro ukládání snímků určených k tisku. Jako jeden z mála grafických formátů podporuje vícestránkové soubory.
- **WebP** – moderní formát vyvinutý firmou Google v roce 2010 jako náhrada formátů JPEG, PNG a GIF zejména na webu. Formát podporuje ztrátovou i neztrátovou kompresi, průhlednost i animace.
- **RAW** – třída formátů pro ukládání surových (*raw*) dat ze snímače digitálního fotoaparátu. Neobsahují přímo obrázek, ale všechny potřebné informace k jeho vytvoření.

Vektorové formáty

Pro vektorová data jsou k dispozici nejen obecné formáty spojené s různými volně dostupnými programy, ale také mnoho specifických formátů, které jsou vázány na konkrétní softwarový produkt (často licenční):

- **CDR** (*CorelDRAW File Format*) – uzavřený binární formát určený nativně pro aplikaci CorelDRAW. Částečně jej lze zpracovat i jinými vektorovými editory.
- **SVG** (*Scalable Vector Graphics*) – důležitý formát a značkovací jazyk pro popis dvourozměrných vektorových dat pomocí XML. V současnosti je základním otevřeným formátem pro vektorovou grafiku na Internetu.
- **EPS** (*Encapsulated PostScript*) – varianta formátu PS, která umožňuje ukládat pouze jednu stránku, resp. tisknutelnou oblast obsahující pouze jeden obrázek.
- **AI** (*Adobe Illustrator Artwork*) – uzavřený binární formát určený nativně pro aplikaci Adobe Illustrator. Lze jej v omezené míře zpracovat i jinými vektorovými editory.
- **DWG** (*AutoCAD Drawing*) – uzavřený binární formát určený nativně pro výkresy v aplikaci AutoCAD[69]. Díky rozšíření programu AutoCAD je spolu s formátem DXF považován za standard v oblasti výměny dvourozměrných výkresových dat.
- **DXF** (*Drawing Interchange Format*) – otevřený formát vytvořený pro usnadnění výměny dat mezi aplikací AutoCAD a dalšími programy. Podporuje jak textovou, tak binární podobu uložených dat.

[69] CAD (*Computer-Aided Design*) aplikace obsahují grafické, geometrické, matematické a inženýrské nástroje pro kreslení plošných výkresů a modelování objektů a dějů reálného světa.

- **DWF** (*Design Web Format*) – binární formát pro efektivní distribuci návrhových dat. Obsahuje elektronické výkresy s veškerými výkresovými styly, měřítky a výkresovými listy, lze je tak snadno tisknout na jakémkoli výstupním zařízení. Data jsou komprimována, soubory jsou proto menší a přenositelnější než originální výkresy ve formátu CAD.
- **WMF** (*Windows Metafile*) – otevřený binární formát vytvořený pro OS Windows. Jeho rozšířením později vznikl formát **EMF** (*Enhanced Windows Metafile*). Oba formáty mohou obsahovat vektorová i rastrová data a existují k nim i komprimované verze známé pod označením **WMZ**, resp. **EMZ**.
- **SWF** (*Small Web Format*) – binární formát pro Adobe Flash. Je dominantním formátem pro zobrazení animované vektorové grafiky na webu, může být také uplatněn v různých aplikacích. Specifikace je dostupná za omezujících licenčních podmínek.
- **ODG** (*OpenDocument Graphics*) – otevřený binární formát z množiny formátů ODF, vychází ze standardu XML.

Prezentační formáty

Speciální oblast zpracování grafických dat tvoří prezentace. Pro tyto účely jsou nejčastěji využívány tyto formáty:

- **PPT** (*PowerPoint Presentation*) – binární formát firmy Microsoft určený pro aplikaci PowerPoint. Má stejné vlastnosti jako příbuzné formáty DOC a XLS.
- **PPTX** (*Office Open XML Presentation*) – nástupce formátu PPT v PowerPointu 2007. Podobně jako u formátů DOCX a XLSX se jedná o komprimovaný soubor se součástmi dokumentů ve formátu XML, případně s dalšími soubory.
- **ODP** (*OpenDocument Presentation*) – otevřený binární formát z množiny formátů ODF, vychází ze standardu XML.
- **PDF** (*Portable Document Format*) – tento již zmíněný všestranný formát může obsahovat také animace či jiné interaktivní prvky (formuláře, testy apod.).

Zvukové formáty

V oblasti multimédií je situace velmi rozmanitá. Společnou vlastností většiny formátů je použití komprese, která výrazně snižuje objem uložených dat (podrobněji v kap. 6.3, str. 152). Pro ukládání pouze zvukové informace jsou nejčastěji užívány tyto formáty:

- **MP3** (*MPEG Audio Layer III*) – ztrátový formát založený na principu odstranění informací, jež člověk neslyší nebo si je neuvědomuje. Výsledná kvalita závisí především na zvoleném datovém toku, svou roli hrají i další parametry volitelné při kompresi.
- **AAC** (*Advanced Audio Coding*) – standard ztrátové komprese zvuku, vyvinutý jako logický následovník formátu MP3 pro vysoké datové toky.
- **WMA** (*Windows Media Audio*) – vyvinut firmou Microsoft původně jako náhrada MP3, dnes spíše soupeří s AAC. Je rozšířený u stolních přehrávačů hudby a DVD.
- **FLAC** (*Free Lossless Audio Codec*) – otevřený formát využívající neztrátovou kompresi, která snižuje objem dat přibližně o 40 %.
- **ALAC** (*Apple Lossless Audio Codec*) – formát pro neztrátovou kompresi hudby vyvinutý společností Apple. Na rozdíl od ostatních formátů není náročný na dekompresi, což umožňuje využití v přístrojích s omezeným zdrojem energie (např. iPod).
- **WAV** (*Waveform Audio File Format*) – formát vytvořený společnostmi Microsoft a IBM pro ukládání zvuku na PC. Obsahuje nekomprimovaná data kódovaná pulsně kódovou modulací (princip vzorkování, viz kap. 4.4.1, str. 83).

Pro kombinace textových, obrazových a zvukových dat se používají tzv. multimediální kontejnery. V jednom souboru je obsaženo více proudů multimediálních dat se vzájemnou synchronizací a uživatel si při přehrávání může vybrat, kterou kombinaci chce použít. Mezi typické formáty tohoto typu patří:

Multimediální kontejnery

- **AVI** (*Audio Video Interleave*) – formát uvedený firmou Microsoft již v roce 1992. Dnes je považován za zastaralý formát se značnými nedostatky.
- **MP4** (*MPEG-4 Part 14*) – moderní a otevřená varianta k zastaralému formátu AVI. Může obsahovat navíc menu, více titulků i zvukových stop.
- **MPG** (*MPEG Program Stream*) – hlavní formát pro DVD video, vyznačuje se snadnou implementací v komerčních zařízeních.
- **TS** (*MPEG Transport Stream*) – používá se v prostředí, kde není zaručena bezchybnost přenosu dat (např. DVB, streamování po Internetu).
- **VOB** (*Video Object*) – je založen na MPG formátu, ale s dalšími omezeními a specifikacemi ve vlastních datových tocích.
- **MKV** (*Matroska*[70] *Video*) – otevřený svobodný formát zahrnující většinu moderních video a audio formátů, podporuje také prostorový zvuk. Je podporován většinou přehrávačů. Existuje též ve variantě **MKA** (*Matroska Audio*) pouze pro zvuková data.
- **RIFF** (*Resource Interchange File Format*) – souborový formát firmy Microsoft pro ukládání multimediálních zvukových a obrazových předloh. Mezi formáty, které tento kontejner obsahuje, patří nejen WAV a AVI, ale také vektorový grafický formát CDR.

- **WebM** – otevřený formát společnosti Google navržený speciálně pro streamování videa. Jedná se o jednoduchý formát s nízkou výpočetní náročností kodeků umožňující přehrávání videa na takřka libovolném zařízení včetně tabletů a netbooků.

6.2.4 Konverze souborových formátů

Pod pojmem **konverze** rozumíme změnu datového formátu beze změny informačního obsahu. V ideálním případě tak dochází ke změně způsobu uložení dat, ale jejich smysl zůstává zachován. V praxi tomu ale často bývá jinak a dochází buď ke ztrátě, nebo k nabytí informace. Příčina tohoto stavu tkví v tom, že některé informace nejsou ve zdrojovém, resp. cílovém formátu podporovány a při konverzi jsou doplněny, resp. odstraněny.

Protože data je neustále potřeba někam přenášet a přizpůsobovat příjemci, setkáváme se s konverzemi v praxi velmi často. Možnosti jsou následující:

Možnosti konverzí

- konverze *textového* formátu na *textový* formát – například z HTML do TXT;
- konverze *textového* formátu na *binární* formát – například z RTF do DOC;
- konverze *binárního* formátu na *textový* formát – například z DOC do HTML;
- konverze *binárního* formátu na *binární* formát – například z BMP do JPEG.

V některých případech může být konverze problematická. Vždy je třeba zvážit, které dva formáty je možné mezi sebou konvertovat, abychom dosáhli použitelného výsledku.

[70] Matrjoška (Матрёшка) je ruský výraz pro tradiční vícevrstvou dutou panenku vyráběnou od roku 1890.

Provedení konverze

Konverzi provádíme v ideálním případě nějakým programem, který je schopen číst data ve zdrojovém formátu a vypisovat je v cílovém formátu. Často k tomu stačí některý z běžných programů, například textový procesor nebo grafický editor. Nejčastěji jsou schopnosti čtení spojeny se službou „Otevřít“, schopnosti vypisování pak se službou „Uložit“, resp. „Uložit jako“. V obou případech se objeví standardní dialogové okno, v němž je pro nás nejdůležitější seznam podporovaných souborových formátů.

Konvertory

V některých případech je výhodnější použít specializovaný program, tzv. **konvertor**. Ten slouží ke konverzím mezi souborovými formáty a obvykle pracuje dávkově, čehož lze využít při hromadném zpracování dat. Jako příklad můžeme uvést linuxový program `convert` z programového balíku ImageMagick, který dokáže konvertovat mezi více než 100 grafickými formáty, nebo již zmíněné (viz str. 146) programy `enca` a `cstocs` pro konverzi mezi kódováním národních znaků.

Kontrolní otázky

5. Co je to souborový formát?
6. Jaké jsou výhody otevřených formátů?
7. Jaké komplikace mohou nastat při práci s rozšířeným textovým formátem?
8. V čem spočívá princip asociace formátu a aplikace?
9. Na čem závisí přenositelnost formátu? Které formáty jsou dobře přenositelné?
10. Používají se pro zpracování textů pouze textové formáty?
11. K čemu slouží konverze souborových formátů a jak ji lze provést?

6.3 Komprese dat

Data v počítači jsou obvykle ukládána s určitou redundancí, tedy na daleko větším prostoru, než je nezbytně nutné. Redundance vzniká například nedokonalým kódováním dat, nutností rychlého přístupu k datům nebo přidáním zabezpečovacích prvků. Toto na první pohled neúsporné ukládání tak může být pro efektivní práci s daty velmi výhodné.

Cílem **komprese** (též komprimace, *compression*) je zmenšení objemu dat při zachování všech důležitých informací, které obsahují původní data. Toho lze docílit zejména odstraněním redundance (nadbytečných informací) a irelevance (nepodstatných informací), což vede ke zvýšení entropie dat. Komprese je nutná při práci s některými typy dat a také při přenosech dat po síti s omezenou datovou propustností. Navíc se snižuje riziko, že se během přenosu přeruší spojení a celou akci bude třeba opakovat. Komprese také zvyšuje přehlednost uložení souborů na pevném disku. Soubor, který vznikne kompresí, nazýváme **archiv** (*archive*).

Existuje ještě jeden velmi dobrý důvod pro kompresi. Soubory jsou na discích zaznamenávány do alokačních bloků (clusterů), které jsou nejmenší jednotkou paměti na datovém nosiči. Velikost clusteru se u různých datových nosičů liší v závislosti na konstrukci a celkové kapacitě disku (podrobněji viz str. 165). Typická velikost clusteru je například 4 KiB. To v praxi znamená, že každý libovolně velký soubor zabere na disku místo odpovídající nejbližšímu vyššímu násobku velikosti clusteru. Toto plýtvání místem uvnitř alokovaných oblastí se označuje jako **vnitřní (souborová) fragmentace** (*file fragmentation*). Malé řídce používané soubory je proto vhodné sdružovat v jednom archivu, protože i bez použití komprese můžeme výrazně zmenšit prostor, který data na disku zabírají.

Vnitřní fragmentace

Příklad 6.7

Uvažujme milion malých textových souborů, z nichž každý má velikost pouhý 1 bajt, a datové úložiště s alokačními bloky o velikosti 4 KiB. Pokud bychom těchto milion souborů ukládali samostatně, spotřebovali bychom milion alokačních bloků. Přestože velikost souborů je 1 000 000 B, na disku by ve skutečnosti zabraly prostor o velikosti 3,81 GiB. Pokud ale soubory umístíme do jednoho archivu, potřebujeme jen 245 alokačních bloků, tedy pouhých 980 KiB.

6.3.1 Druhy kompresí

Komprese můžeme rozdělit do několika kategorií podle následujících kritérií:

- *možnost plného obnovení původních dat* – neztrátová × ztrátová;
- *okamžik, ve kterém komprese nastává* – off-line × on-line;
- *přihlížení k informační hodnotě komprimovaných dat* – logická × fyzická;
- *časová náročnost procesu komprese a dekomprese* – symetrická × asymetrická;
- *schopnost přizpůsobit se charakteru dat* – adaptivní × neadaptivní.

Neztrátovou kompresi (*lossless compression*) použijeme v případě, kdy nemůžeme dopustit jakoukoli ztrátu dat a požadujeme možnost rekonstrukce komprimovaného souboru do původní podoby. Víme, že u binárního formátu vede ztráta byť jediného bitu obvykle ke katastrofě v podobě další nepoužitelnosti tohoto souboru (viz tab. 6.2, str. 142).

Neztrátová komprese

Existují ale datové soubory, ve kterých ztráta některých dat vůbec nevadí. Jsou to soubory s nadbytečným množstvím informací (typicky obrazová nebo zvuková data), na které může být s výhodou aplikována **ztrátová komprese** (*lossy compression*). Průměrné lidské smysly ztrátu vůbec nepostřehnou, zvláště jedná-li se například o prezentaci obrazu v nízkém rozlišení na monitoru nebo poslech zvukových dat na zařízení nízké kvality.

Ztrátová komprese

Komprese off-line (*off-line compression*) se používá zejména pro účely archivace dat za použití specializovaných programů, které na požádání zkomprimují zvolené soubory a později je na požádání zase dekomprimují.

Off-line komprese

Některé programy však mohou provádět **on-line kompresi** (*on-line compression*) pro úsporu místa při běžné práci. Například grafické editory ukládají obrazová data do souborů v komprimované podobě. Celý proces běží automaticky bez zatěžování uživatele, někdy je on-line komprese přímou součástí operačního systému.

On-line komprese

Logická komprese

Logická komprese (*logical compression*) používá logické substituce (tj. sekvence znaků nahrazuje jinou, úspornější řadou). Konkrétním příkladem jsou zkratková slova ČEDOK (Česká dopravní kancelář) nebo OSN (Organizace spojených národů).

Fyzická komprese

Fyzická komprese (*physical compression*) se snaží na základě analýzy vstupních dat nalézt kratší kód. Proces probíhá bez zřetele na logiku dat, se kterými se manipuluje. Vytváří se nová sekvence znaků (bitů, bajtů apod.), jejíž vztah k původním datům lze rozpoznat výhradně s použitím dekompresního algoritmu. Bez znalosti tohoto algoritmu je informační hodnota komprimovaných dat nulová.

Symetrická komprese

Pokud je doba (a tím většinou i počet a druh operací) potřebná pro kompresi a dekompresi dat přibližně stejná, jedná se o **symetrickou kompresi** (*symmetric compression*). Této vlastnosti se využívá při on-line kompresi.

Asymetr. komprese

Asymetrická komprese (*asymmetric compression*) je založena na tom, že kompresní algoritmus provede větší množství operací při kompresi dat než při jejich dekompresi (nebo naopak). Krátká doba dekomprese je výhodná například tam, kde soubor dat ukládáme na disk sice v komprimované podobě, ale často s ním pracujeme. Opačný případ, tedy rychlá komprese a zdlouhavá dekomprese, je užitečná při častých změnách cílových dat, například při periodickém zálohování.

Adaptivní komprese

Použití **adaptivní komprese** (*adaptive compression*) umožňuje dosáhnout určité nezávislosti na komprimovaných datech. Takové algoritmy neobsahují žádné statické slovníky řetězců. Pro každý komprimovaný soubor dat si budují nový slovník dynamicky v průběhu kódování. Kompresním algoritmům se podrobněji věnuje kapitola 6.3.3.

Neadapt. komprese

Neadaptivní komprese (*non-adaptive compression*) je určena výhradně pro specifická data. Takové algoritmy většinou obsahují předdefinované slovníky nebo řetězce znaků, o kterých je známo, že jejich pravděpodobnost výskytu v souborech dat je vysoká.

6.3.2 Výpočet kompresního poměru

Kompresní poměr (*compression ratio*) se používá pro hodnocení kvality zhuštění informace. Udává se jako poměr dat před kompresí (tzv. hrubých dat) o objemu h a komprimovaných dat (tzv. čistých dat) o objemu k a vyjadřuje se různým způsobem:

- *jaký násobek* hrubých dat tvoří čistá data – počítáme jako $h/k : 1$;
- *na kolik procent* se data kompresí zmenšují – počítáme jako $k/h \cdot 100$;
- *o kolik procent* se data kompresí zmenšují – počítáme jako $(1 - k/h) \cdot 100$.

Příklad 6.8

Data o velikosti 248 KiB byla zkomprimována do souboru o velikosti 188 KiB. Po dosazení hodnot do vztahu zjistíme, že data se zmenšila o 25 % na 75 % své původní velikosti.[71] Dosažený kompresní poměr je $248/188 : 1$, tedy přibližně $1{,}3 : 1$.

Režijní informace

Aby bylo možné zpětně rekonstruovat původní data, je nezbytné někam zaznamenat metadata o původních souborech (názvy, časy, umístění apod.) a použité metodě komprese.

[71] Všimněme si souvislosti mezi hodnotami „o kolik procent“ a „na kolik procent“ – jejich součet musí vždy dát ze zřejmých důvodů 100 %. Čtenář, který by snad měl problém rozlišit mezi těmito hodnotami, nechť se zamyslí nad rozdílem mezi snížením platu o 10 % a snížením platu na 10 %.

Tento záznam o struktuře archivu se nazývá **režijní informace** (*overhead information*). Její velikost je obvykle zanedbatelná a je závislá na počtu souborů v archivu.

Jsou-li data uložena v podobě, která neumožňuje jejich zhuštění (např. je jich příliš málo), dochází při pokusu o kompresi ke zvýšení jejich objemu o režijní informaci. Výsledný archiv je potom větší, než byla původní data. Tomuto nežádoucímu jevu, při kterém se data kompresí zvětšují, říkáme **záporná komprese** (*negative compression*). *Záporná komprese*

6.3.3 Kompresní metody a jejich vlastnosti

Abychom lépe pochopili princip komprese dat, seznámíme se podrobněji s jednou z nejjednodušších neztrátových metod. Poté se zaměříme na složitější neztrátové i ztrátové metody a jejich využití.

Metodu **proudového kódování** (*Run-Length Encoding*) lze použít na libovolná data. Na charakteru dat ovšem silně závisí dosažený kompresní poměr. Základním principem metody je kódování posloupnosti stejných hodnot v hrubých datech, tzv. proudů (běhů). Nejsou-li v hrubých datech dostatečně rozsáhlé běhy, může dojít i k záporné kompresi. Výhodou této neztrátové metody je jednoduchost a rychlost komprese i dekomprese. Každý proud je zkomprimován do podoby jednoho paketu RLE, který obsahuje: *RLE*

1. *proudové číslo (opakovač)*, který udává počet znaků proudu často snížený o jedničku;
2. *proudovou hodnotu*, jež se shoduje s hodnotou opakujícího se znaku v proudu.

Příklad 6.9

Předpokládejme sekvenci znaků `AAAAkkRRRRRm`. Byly rozpoznány čtyři proudy, kterým byly přiřazeny pakety `3A`, `1k`, `4R` a `0m`. Sekvence bude zakódována do podoby `3A1k4R0m`.

Metoda RLE není vhodná pro kompresi textových souborů, protože kompresní poměr by byl v takovém případ velmi špatný. Podobně by dopadla komprese většiny binárních souborů. Použití je proto možné pouze pro jednoduché obrázky s malou barevnou hloubkou (maximálně 256 barev). Takové obrázky mají často jednobarevné pozadí, které zaručuje dostatečnou velikost jednotlivých proudů. Naopak pro kompresi fotografií se metoda RLE nehodí, neboť fotografie na první pohled nesplňují podmínku dlouhých proudů.

Při kompresi dat reprezentujících rastrový obraz se začíná číst a kódovat od levého horního rohu a postupuje se po řádcích nebo sloupcích jednotlivých pixelů do pravého dolního rohu. Praktické přístupy ke kódování rastrových obrazů se dělí podle typu základních datových prvků tvořících proudy. Kódovat můžeme na úrovni bitové, bajtové nebo pixelové.

Bitová úroveň kódování rozeznává pouze proud binárních jedniček a proud binárních nul. Taková metoda je účinná prakticky pouze na monochromatické obrazy, které obsahují dostatečně dlouhé proudy jedniček i nul k tomu, aby metoda v této modifikaci dosáhla dobrého kompresního poměru. Paket je v takovém případě tvořen jedním bajtem logicky rozděleným na dvě části. Nejvýznamnější bit každého bajtu určuje proudovou hodnotu a sedm méně významných bitů představuje proudové číslo udávající počet opakování proudové hodnoty snížený o jedničku. Je zřejmé, že délka proudu musí být v rozsahu 1–128 znaků, neboť do sedmi bitů lze zapsat číslo z intervalu 0–127.[72] Pokud původní nekomprimovaná

[72] To je také důvod, proč se jako proudové číslo obvykle používá počet znaků proudu snížený o jedničku.

data obsahují proud delší než 128 znaků, musí být tento proud při kódování rozdělen na dva nebo více proudů. Je vhodné připomenout, že při kompresi tato metoda nerozeznává hranice jednotlivých bajtů, neboť ty nejsou v tomto případě zajímavé.

Bajtová úroveň kódování rozeznává proudy opakujících se bajtových hodnot a naopak si nevšímá dělení na bity nebo hranic 16bitových (příp. 32bitových) slov. Paket RLE se v tomto případě skládá ze dvou bajtů. První bajt udává proudové číslo v rozsahu 0–255 a druhý bajt udává proudovou hodnotu ve stejném rozsahu. Velikost proudu, kterému bude odpovídat jeden paket, může dosahovat délky až 256 znaků. Metoda RLE na úrovni bajtů je vhodná pro obrázky s osmibitovou barevnou hloubkou, tj. s nejvýše 256 barevnými odstíny. Komprese na úrovni bajtů je použita také v příkladu 6.9.

Pro dosažení lepších kompresních poměrů bývá vhodné doplnit bajtovou metodu o schopnost ukládání jednobajtových proudů do kódovaného toku dat. V této modifikaci se pro vyjádření proudového čísla používá pouze sedm (nebo i šest) méně významných bitů prvního bajtu paketu. Maximální velikost proudu je tak snížena na 128 (nebo 64) znaků. Nejvýznamnější bit (nebo dva bity) je indikátorem typu proudu. Pokud je tento bit nastaven na 1, jedná se o klasický dvoubajtový paket RLE, kde druhý bajt obsahuje proudovou hodnotu. Při nastavení nejvýznamnějšího bitu prvního bajtu na hodnotu 0 (nebo dvou bitů na hodnotu 00) následuje za proudovým číslem přesný proud znaků, který se čte v té podobě, v jaké je zapsán. Jedná se o tzv. *přímý paket*. Používáním přímých paketů lze zabránit záporné kompresi.

Pixelová úroveň se používá u složitějších obrázků, kdy je každý pixel reprezentován více než jedním bajtem, typicky u obrázků v pravých barvách.[73] Velikost paketu se v tomto případě liší podle počtu bajtů na pixel. Pro obrázky v pravých barvách bude paket čtyřbajtový, kde první bajt bude reprezentovat proudové číslo a následující tři bajty jedinou proudovou hodnotu. Kódovací metoda zůstává stejná jako u jednobajtové úrovně a opět lze využít modifikaci s indikátorem přímého paketu.

Existuje i další modifikace metody RLE, která se snaží vyhnout záporné kompresi, ke které dochází při střídavých datech předlohy. Základní myšlenkou je použití tří bajtů jako jediného paketu RLE. První bajt v tomto případě obsahuje tzv. *příznakovou hodnotu* (též escape sekvenci, viz dále), následující dva bajty tvoří „klasický“ paket RLE složený z proudového čísla a proudové hodnoty. Kódování dat předlohy probíhá tak, že pokud kompresní program zjistí jedno, dvou nebo tříbajtový proud, zapíší se tyto hodnoty do komprimovaného toku dat přímo. Odpadá tak zápis dalších bajtů. Při dekompresi se nejprve posoudí, zda první bajt paketu RLE obsahuje příznakovou hodnotu. Pokud ano, expanduje se paket podle údajů proudového čísla a proudové hodnoty. Pokud ne, přidá se tento bajt přímo do nekomprimované podoby souboru. Nevýhodou této modifikace je zvýšení minimální velikosti proudu pro „klasický“ paket RLE na čtyři bajty, což může negativně ovlivnit kompresní poměr některých dat. Další problém nastává, pokud nekódovaný tok dat obsahuje hodnotu, která se shoduje s hodnotou vyhrazenou pro příznak. Tomu lze zabránit tím, že se jako příznaková hodnota zvolí bajt, který se v datech předlohy nevyskytuje. Aplikaci této metody ilustruje příklad 6.10 (jsou použita stejná textová data jako v příkladu 6.9).

[73] Informaci o použité barevné hloubce lze obvykle získat z hlavičky souborového formátu.

Příklad 6.10

Uvažujme sekvenci znaků `AAAAkkRRRRRm`. Byly rozpoznány čtyři proudy, kterým byly přiřazeny pakety `3A`, `kk`, `4R` a `m`. Sekvence bude zakódována do podoby `#3Akk#4Rm`, kde znak # reprezentuje příznakovou hodnotu.

Příklad 6.11

Předpokládejme úsek zkomprimovaných dat o délce 32 bitů v podobě posloupnosti 10010001 01110110 11111010 00111010. Jak bude tento úsek reprezentován při použití metody RLE na bitové, bajtové a pixelové úrovni a jaký bude dosažený kompresní poměr?

Při použití na bitové úrovni data reprezentují úsek 319 bitů, který může představovat například monochromatický obraz složený z 319 bodů, z nichž prvních 18 je černých, následuje 119 bílých, 123 černých a 59 bílých. Data byla kompresí zmenšena o 90 %.

$$\underbrace{\underline{10010001}}_{18\times\ 1}\ \underbrace{\underline{01110110}}_{119\times\ 0}\ \underbrace{\underline{11111010}}_{123\times\ 1}\ \underbrace{\underline{00111010}}_{59\times\ 0}$$

Při použití na bajtové úrovni data reprezentují úsek 397 bajtů, který může reprezentovat například obraz v odstínech šedi složený z 397 bodů, z nichž prvních 146 bodů má odstín 46% šedé a zbylých 251 bodů má odstín 23% šedé. Data byla kompresí zmenšena o 99 %.

$$\underbrace{\underline{10010001}\ 01110110}_{146\times\ 118}\ \underbrace{\underline{11111010}\ 00111010}_{251\times\ 58}$$

Pokud budeme uvažovat indikátor přímého paketu na bajtové úrovni, pak data reprezentují úsek 141 bajtů, který může odpovídat obrazu v odstínech šedi složený z 18 bodů odstínu 46% šedé a 123 bodů odstínu 23% šedé. Data byla kompresí zmenšena o 97 %.

$$\underbrace{\underline{1}\underline{0010001}\ 01110110}_{18\times\ 118}\ \underbrace{\underline{1}\underline{1111010}\ 00111010}_{123\times\ 58}$$

Při použití na pixelové úrovni data reprezentují úsek 438 bajtů, který může odpovídat obrazu v pravých barvách složenému ze 146 bodů odstínu #76FA3A. Data byla kompresí zmenšena o 99 %.

$$\underbrace{\underline{10010001}\ 01110110\ 11111010\ 00111010}_{146\times\ \#76FA3A}$$

Pokud budeme uvažovat indikátor přímého paketu na pixelové úrovni, pak data reprezentují úsek 54 bajtů, který může odpovídat obrazu v pravých barvách složenému z 18 bodů odstínu #76FA3A. Data byla kompresí zmenšena o 93 %.

$$\underbrace{\underline{1}\underline{0010001}\ 01110110\ 11111010\ 00111010}_{18\times\ \#76FA3A}$$

Nyní se budeme věnovat složitějším kompresním metodám. Podle principu, na kterém jsou založeny, je můžeme rozdělit do těchto skupin:

- *statistické* – založené na četnosti výskytu znaků v komprimovaném souboru;
- *slovníkové* – založené na kódování všech vyskytujících se posloupností;
- *transformační* – založené na ortogonálních nebo jiných transformacích.

Statistické metody

Mezi neztrátové statistické algoritmy patří již dříve zmíněné metody nerovnoměrného kódování dat. **Shannonovo-Fanovo kódování** (viz str. 103) i **Huffmanovo kódování** (viz str. 104) pracují na základě různých četností znaků v kódovaných datech a jsou určeny především pro kompresi textových souborů. Zajímavou neztrátovou kompresní metodou je **aritmetické kódování**[74] (*arithmetic coding*). Jedná se o formu entropického kódování s proměnlivou délkou kódového slova. Na rozdíl od výše zmíněných metod však aritmetické kódování kóduje celý vstupní text do jednoho čísla n, kde $0 \leq n < 1$. Metoda je hardwarově velmi náročná, proto se prosazuje velmi pomalu. Algoritmus nejprve zjistí pravděpodobnosti prvků ze vstupní abecedy a podle toho rozdělí základní interval $\langle 0; 1)$. Po přečtení prvního znaku vybere podinterval náležící přečtenému znaku, tento interval znovu rozdělí a pokračuje rekurzivně do přečtení posledního znaku zprávy. Tímto způsobem získáme interval pravděpodobnosti reprezentující danou zprávu. Poté stačí vybrat libovolné číslo z tohoto intervalu.

Příklad 6.12

Chceme zakódovat zprávu $m = abac$, ve které jednotlivé znaky mají pravděpodobnost výskytu $p(a) = 1/2$, $p(b) = p(c) = 1/4$. Sestavíme kódovací tabulku:

Symbol	a	b	a	c
a	⟨0 ; 0,5)	⟨0 ; 0,25)	⟨0,25 ; 0,3125)	⟨0,25 ; 0,28125)
b	⟨0,5 ; 0,75)	⟨0,25 ; 0,375)	⟨0,3125 ; 0,34375)	⟨0,28125 ; 0,296875)
c	⟨0,75 ; 1)	⟨0,375 ; 0,5)	⟨0,34375 ; 0,375)	⟨0,296875 ; 0,3125)

Pro zakódování zprávy $m = abac$ můžeme tedy použít libovolné číslo ležící v intervalu $\langle 0{,}296875; 0{,}3125)$. Nyní nalezneme binární reprezentaci takového čísla.

Binárně	Hledaný interval	Desítkově
0,0…	⟨0 ; 0,5)	0
0,01…	⟨0,25 ; 0,5)	0,25
0,010…	⟨0,25 ; 0,375)	0,25
0,0100…	⟨0,25 ; 0,3125)	0,25
0,01001…	⟨0,28125 ; 0,3125)	0,28125
0,010011…	⟨0,296875 ; 0,3125)	0,296875

Zprávu lze zakódovat do posloupnosti 010011 se střední délkou kódového slova 1,5 bitu. Získali jsme optimální kód s nulovou redundancí (viz kap. 5.2, str. 97). Zájemci mohou sestrojit Shannonův-Fanův nebo Huffmanův kód a porovnat výsledky.

[74] Princip kódování popsal již počátkem 60. let 20. stol. americký informatik Peter Elias (* 1923, † 2001), jeden z průkopníků v oblasti teorie informace. Algoritmus publikovali v roce 1976 nezávisle na sobě finský informatik Jorma Johannes Rissanen (* 1932, † 2020) a americký informatik Richard Clark Pasco (* 1950) jako svoji disertační práci na Stanfordu.

Slovníkové metody

Jednou z nejrozšířenějších neztrátových metod je **LZW**[75] (*Lempel-Ziv-Welch*). Základním principem tohoto algoritmu je vyhledávání opakovaných posloupností v hrubých datech a jejich kódování kratším kódem. Pomocí odkazů na tyto posloupnosti dat algoritmus postupně buduje adaptivní datový slovník. Komprese dat probíhá podle následujícího schématu:

- Pokud se posloupnost bajtů (řetězec) ve vytvářeném slovníku nevyskytuje, je tato posloupnost přidána do slovníku a v nezměněné formě zapsána do komprimovaného výstupního toku dat.
- Pokud se vstupní posloupnost bajtů ve slovníku již nachází, zapíše se do výstupního toku dat pouze zástupná slovníková hodnota odpovídající nalezené vstupní posloupnosti. Díky tomu, že zástupná hodnota je vždy menší než čtená posloupnost, dochází ke kompresi dat.

Dekomprese je inverzním procesem, při kterém algoritmus čte komprimovaný tok dat a stejným způsobem vytváří datový slovník posloupností (řetězců) bajtů. Díky tomu není nutné slovník ukládat, neboť dekompresní algoritmus si jej vytvoří z dat sám. Další výhodou této metody je velmi příznivý kompresní poměr, který se pohybuje od hodnoty 1,5 : 1 až do 10 : 1 i více. Komprese i dekomprese jsou rychlé a nejsou paměťově náročné. Nevýhodou může být rychlý nárůst slovníkových kódů odkazujících na řetězce původního souboru, což vede v některých případech k zaplnění paměti určené pro slovníkové kódy. Z toho pak vyplývá smazání aktuálního slovníku a jeho nové vytváření bez možnosti použití „zapomenutých“ odkazů. V případě větších slovníků zase narůstá doba vyhledání řetězce v tabulce.

Ruský programátor Igor Viktorovič Pavlov je autorem modifikace metody LZW, která je pomalejší, ale dosahuje lepšího kompresního poměru. Metoda je známá jako **LZMA** (*Lempel-Ziv-Markov Chain Algorithm*) a zdokonaluje algoritmus LZ77 o Markovovy[76] řetězce, které byly vyvinuty již dávno pro účely popisu diskrétních náhodných procesů.

Kompresní metoda **Deflate**[77] využívá kombinaci slovníkové metody LZ77 a Huffmanova kódování. Setkáme se s ní například u formátů PNG, ODF nebo PDF.

Jako posledního zástupce neztrátových metod uveďme ještě skupinu metod **CCITT**[78], reprezentovanou standardy CCITT Group 3 a CCITT Group 4 pro kompresi faksimilních přenosů (fax) a pro kompresi obrazových dat monochromatických dokumentů. Tyto neadaptivní kompresní metody využívají ke snížení redundance v datovém proudu podobného principu jako Huffmanovo kódování.

[75] Izraelští vědci Abraham Lempel (* 1936, † 2023) a Ja'akov Ziv (* 1931, † 2023) vypracovali v letech 1977 a 1978 kompresní algoritmy, které vešly ve známost jako LZ77 a LZ78. V roce 1984 na jejich práci navázal americký vědec Terry Archer Welch (* 1939, † 1988), který modifikoval kompresní algoritmus LZ78 pro potřeby diskových řadičů.

[76] Andrej Andrejevič Markov (* 1856, † 1922) byl ruský matematik, který se zabýval teoriemi stochastických procesů. Jeho syn, taktéž Andrej Andrejevič Markov (* 1903, † 1979) je považován za zakladatele sovětské školy konstruktivní matematiky a logiky.

[77] Americký programátor Phillip Walter Katz (* 1962, † 2000) ji vytvořil pro svůj program PKZIP a souborový formát ZIP.

[78] Mezinárodní poradní sbor pro telegraf a telefon (*Comité Consultatif International Téléphonique et Télégraphique*) byl založen v roce 1956 a je orgánem Mezinárodní telekomunikační unie (ITU) se sídlem v Ženevě. Shromažďuje materiály týkající se mezinárodní spolupráce v oblasti techniky telefonních a telegrafních spojů a zpracovává je pro potřeby členských zemí ITU. V roce 1993 se přejmenoval na ITU-T.

Transformační metody

Jednou z nejpokrokovějších ztrátových metod pro kompresi grafických dat je **JPEG** (*Joint Photographic Experts Group*). Na rozdíl od dosud popsaných metod není JPEG přesně definovaným algoritmem, ale spíše sadou kompresních metod, které mohou být přizpůsobeny konkrétním požadavkům uživatele. Metoda byla navržena pro práci jak s obrazy v odstínech šedi, tak s obrazy s velkou barevnou hloubkou. Horších výsledků dosahuje při práci s vektorovými obrazy a jednoduchou grafikou. Celá kompresní technologie je založena na faktu, že lidské oko dokáže hůře rozeznat malé změny barev blízkých bodů než rozdíly v intenzitě a jasu. Z toho důvodu je také tento standard vhodnější pro barevné obrázky než pro předlohy skládající se pouze z mnoha odstínů šedé.

Nezanedbatelnou výhodou metody JPEG je možnost volby faktoru kvality (*Q-factor*), který ovlivňuje kvalitu obrázků po kompresi. Hodnoty faktoru se pohybují od 1 do 100, kde menší hodnota znamená menší objem komprimovaných dat za cenu horší kvality obrázku a vyšší hodnota naopak. Výsledkem použití této metody by mělo být nalezení takové hodnoty faktoru, který by zaručoval největší kompresní poměr při současném zachování požadované kvality obrázku. V praxi se ukazuje, že snížení kvality na 75 % většina uživatelů vůbec nezaznamená, přitom kompresní poměr v takovém případě může být až 25 : 1.

Kompresní algoritmus JPEG využívá diskrétní Fourierovu[79] transformaci[80]. Postup při kompresi se skládá z pěti navazujících kroků:

1. *Transformace barev* do barevného modelu YC_BC_R. Toto schéma umožňuje aplikovat kompresi s lepším výsledkem při stejné kvalitě obrazu. V další fázi jsou jasové složky Y a barevné složky C_B a C_R zpracovány odděleně.
2. *Redukce barev* – snižování objemu dat zprůměrováním barevných složek C_B a C_R sousedních pixelů. Využívá se toho, že lidské oko je citlivější na změny jasu než na změny odstínů barev. Tím lze ušetřit 33–50 % paměťového prostoru.
3. *Rozdělení do bloků* – obrazová data každého barevného kanálu jsou rozdělena do čtverců o rozměrech 8 × 8 pixelů, hodnoty jasu v každém bloku jsou transformovány z intervalu $\langle 0; 2^p - 1\rangle$ na interval $\langle 2^{p-1}; 2^{p-1} - 1\rangle$. Následně je aplikována diskrétní kosinová transformace (DCT) dle vztahu

$$F(u,v) = \frac{1}{4} \cdot C(u) \cdot C(v) \cdot \left[\sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cdot \cos \frac{(2x+1)u\pi}{16} \cdot \cos \frac{(2y+1)v\pi}{16} \right],$$

   kde x, y jsou souřadnice původní matice, u, v jsou souřadnice transformované matice a $C(i)$ nabývá hodnoty $1/\sqrt{2}$ pro $i = 0$ a hodnoty 1 pro $i > 0$.
4. *Kvantování koeficientů* – v této fázi se stanovuje kvalita a tedy i stupeň komprese obrazu. Lidské oko je citlivé na malé změny jasu, nedokáže však odlišit přesnou hodnotu rozdílů odstínů vysokých frekvencí. Kvantizace redukuje amplitudu vysokofrekvenčních koeficientů, které nepřispívají ke kvalitě obrazu. Účelem je také odstranění informace, která není pozorovatelná.
5. *Kódování* – matice koeficientů je zapisována do výstupního souboru metodou cik-cak. Aplikuje se obvykle aritmetické nebo Huffmanovo kódování. Využívá se nulové hodnoty většiny vysokofrekvenčních koeficientů (v okolí pravého dolního rohu).

[79] Jean-Baptiste Joseph Fourier (* 1768, † 1830) byl francouzský matematik a fyzik, který se proslavil zkoumáním Fourierových řad a jejich aplikací. Je také objevitelem skleníkového efektu.

[80] Jedná se o integrální transformaci, která převádí signály z časové oblasti do oblasti frekvenční.

Při dekompresi se používá inverzního postupu. Vzhledem k tomu, že inverzní postup není z matematického hlediska o moc jednodušší, nebývá doba potřebná pro dekompresi o mnoho kratší než při původním kódovacím procesu. Inverzní DCT je popsána vztahem

$$f(x,y) = \frac{1}{4} \cdot \left[\sum_{u=0}^{7} \sum_{v=0}^{7} C(u) \cdot C(v) \cdot F(u,v) \cdot \cos \frac{(2x+1)u\pi}{16} \cdot \cos \frac{(2y+1)v\pi}{16} \right].$$

Tabulka 6.3 shrnuje informace o kompresních metodách a dává je do souvislosti s konkrétními grafickými rastrovými souborovými formáty.

Tabulka 6.3: *Kompresní metody používané v grafických rastrových souborových formátech*

Formát	Metoda	Ztrátovost	Poznámka
BMP	RLE	ne	pouze pro palety do 256 barev
PCX	RLE	ne	
JFIF	JPEG	ano	
GIF	LZW	ne	
PNG	Deflate	ne	
TIFF	RLE, LZW, CCITT	ne	metodu lze zvolit, včetně ztrátové JPEG

Vývojem standardů pro kompresi videa a připojeného audiosignálu se zabývá komise **MPEG** (*Moving Picture Experts Group*), která pracuje pod organizací ISO. Existuje několik norem pro kompresi videa MPEG, které vyhovují různým oblastem využití, například:

- MPEG-1 – navržena pro přenosové rychlosti a využití v souvislosti s CD. Pro kompresi připojeného audiosignálu existují tři kódovací schémata (vrstvy, *layers*).[81]
- MPEG-2 – navržena pro dálkové a satelitní přenosy při zachování televizní kvality.
- MPEG-3 – navržena původně pro podporu HDTV, ale po úpravách byla tuto oblast schopna pokrýt i norma MPEG-2, proto se od ní upustilo.
- MPEG-4 – navržena pro přenos videa a připojeného audiosignálu pro web, CD, DVD, hlasovou a videokomunikaci a digitální televizní vysílání.
- MPEG-7 – standard pro rychlé vyhledávání multimediálních dat podle klíče.

Kompresní formáty MPEG využívají převážně ztrátovou kompresi pomocí transformačních kodeků. Vzorky obrazu nebo zvuku se rozdělí na drobné segmenty, transformují se na frekvenční prostor a poté kvantizují. Výsledná data se dále kódují entropicky. Typickou aplikací jsou multimediální kontejnery MPEG Program/Transport Stream (viz str. 151).

6.3.4 Možnosti kompresních programů

Funkce komprimátorů

Programy pro kompresi dat, tzv. **komprimátory**, jsou k dispozici ve všech operačních systémech a jejich funkce jsou velmi podobné. Kromě vlastní komprese však nabízejí ještě celou řadu funkcí – jsou například schopny podle různých kritérií vybírat soubory vkládané do archivu, přidávat poznámky k archivu, zašifrovat a zabezpečit archiv heslem, vytvářet archivy v několika částech podle požadované velikosti výstupního média, při dekompresi vytvářet adresářové struktury podle originálu, manipulovat se soubory podle data a času

[81] Zvuková data komprimovaná metodou MPEG Audio Layer III známe ze souborů s rozšířením `.mp3`.

jejich poslední modifikace, testovat integritu archivu, opravit některé chyby v archivu nebo se o to alespoň pokusit, vytvářet samorozbalovací archivy. Prakticky všechny kompresní programy se dají použít i jako programy archivační, repertoár jejich funkcí se u různých typů příliš neliší. Základním rozdílem zůstává uživatelské rozhraní (systém nabídek v dialogových oknech, nebo parametry z příkazového řádku).

Volba účinnosti

Při kompresi lze obvykle zvolit její účinnost – čím větší kompresní poměr, tím déle komprese trvá. Je také možnost zvolit poměr roven jedné, kdy se ze skupiny souborů pouhým spojením bez komprese vytvoří jeden celek – vhodné pro usnadnění manipulace s více soubory. Velikost kompresního poměru je na stejných datech u různých komprimátorů mírně odlišná, je to důsledek různě koncipovaných implementací metod LZW nebo LZMA. Komprimátory také obvykle udávají kompresní poměr bez započtení režijní informace, která je ale u větších objemů dat zanedbatelná.

Formáty

Existuje mnoho formátů pro ukládání komprimovaných dat. Mezi nejznámější a nejpoužívanější se řadí formáty ZIP a RAR. Názvy formátů většinou korespondují s názvy komprimátorů: ZIP, GZ (`gzip`), TGZ nebo TAR.GZ (`tar` + `gzip`), 7Z (7-Zip), ARJ. Všechny archivní formáty jsou z principu binární soubory.

Programy ve Windows

Nejpoužívanějšími kompresními programy na osobních počítačích v prostředí OS Windows jsou WinZIP, WinRAR, WinARJ a 7-Zip. Operační systém Windows umí od verze XP v souborovém manažeru přímo pracovat s archivy formátu ZIP (zobrazovat jako běžné složky, vytvářet ze skupiny souborů komprimovanou složku). Každý z těchto komprimátorů používá svůj nativní formát ukládání dat, ale většinou je schopen číst a někdy i produkovat formáty jiné. Starší programy byly orientovány většinou na jediný formát archivu.

Programy v Unixu

V prostředí operačního systému Unix jsou k dispozici standardní programy `tar` (jen pro archivaci) a `gzip`, dále programy `zip`, resp. `unzip` s formátem kompatibilním s prostředky operačního systému Windows. Modernější programy provozované v systému Windows jsou již také schopny bez problémů zpracovávat archivy získané programy `tar` a případně komprimované programem `gzip`.

Kontrolní otázky

12. K čemu se používá ztrátová komprese?
13. Co je vnitřní fragmentace a jak ji lze eliminovat?
14. Jaký je princip fyzické komprese?
15. Co udává kompresní poměr a jak jej lze spočítat?
16. Jaký je princip kompresní metody RLE?
17. Čím se liší metoda LZW od metody RLE?
18. Které kompresní metody se používají pro obrazová data?
19. Co je záporná komprese a kdy vzniká?
20. Jaké služby nabízejí komprimátory při vytváření archivů?

6.4 Souborové systémy

Souborový systém (*file system*) je způsob organizace souborů ukládaných na elektronická paměťová média. Definuje způsob vytváření, ukládání a přístupu k souborům a adresářům, hlídá volné místo apod. Tvoří také rozhraní mezi vnější pamětí a operačním systémem. *Souborový systém*

Dnes se bez souborového systému prakticky neobejdeme. Software, který realizuje souborový systém, bývá obvykle přímo součástí operačního systému. Důvody pro používání souborového systému jsou především tyto:

1. *Omezená kapacita paměti.* V operační paměti je možné pracovat pouze s omezeným množstvím dat. Soubory naproti tomu umožňují uložit obrovské množství dat.
2. *Nutnost perzistence dat.* Obsah paměti je při ukončení procesu ztracen. Data v souboru přetrvávají, dokud není soubor změněn nebo smazán.
3. *Sdílení a výměna dat.* K jednomu souboru může přistupovat více procesů současně, také je umožněn přenos dat mezi počítači.

Každý souborový systém má své možnosti a omezení, proto jsou v různých operačních systémech podporovány různé souborové systémy. Odlišnosti v jednotlivých souborových systémech lze spatřovat především v následujících bodech: *Odlišnosti*

- rozlišování velkých a malých písmen v názvech souborů a adresářů;
- maximální délka názvu souboru či adresáře a povolené znaky v názvu;
- maximální velikost souboru;
- možnost omezení přístupu k datům pomocí přístupových práv;
- možnost nastavení diskové kvóty uživatelům (přiděleného prostoru na disku);
- možnost automatické komprese a šifrování ukládaných souborů;
- možnost nastavování atributů (vlastností) souborů;
- schopnost žurnálování transakcí.

Moderní **žurnálovací souborový systém** (*journaling file system*) si uchovává informace o operacích, které provedl, a v případě výpadku je schopen se rychle dostat zpět do konzistentního stavu. Principem žurnálování je uchovávání chronologického záznamu ukládaných operací, do kterého se zapisují všechny prováděné činnosti. Pokud dojde například k výpadku napájení, je po restartu nekonzistence opravena návratem do předchozího zaznamenaného stavu za pomoci záznamů z žurnálu. *Žurnálování*

6.4.1 Koncepce souboru a adresáře

Pod pojmem **soubor** (*file*) chápeme pojmenovanou konečnou posloupnost bajtů s definovaným umístěním ve vnější paměti. Reprezentuje základní množinu dat určitého typu. Každý soubor má jméno, aby bylo možné se na něj odkazovat po dobu, která značně přesahuje dobu běhu programu, který jej vytvořil. Soubor tak představuje abstrakci, která umožňuje uložení informace na disk a její pozdější čtení a současně odstiňuje uživatele od technických podrobností práce s disky. *Soubor*

Mezi atributy (příznaky) souborů patří například jméno, typ, velikost, umístění, oprávnění, vlastník a také data a časy vytvoření, modifikace a přístupu k souboru. Soubory můžeme rozdělit do tří typů:

- *běžné soubory* – z pohledu operačního systému se jedná pouze o pojmenovaný souvislý blok dat, jejichž vnitřní struktura je určena programem, který je vytváří;
- *adresáře* – systémové soubory vytvářející logickou strukturu souborového systému;
- *speciální soubory* – znaková a bloková zařízení[82] počítače, pojmenované roury a sockety sloužící ke komunikaci procesů, symbolické odkazy apod.

Adresář

Pod pojmem **adresář** (*directory*, *folder*, *catalog*) chápeme pojmenovanou množinu jiných adresářů a souborů. Technicky se jedná jen o zvláštní typ souboru. V operačních systémech MS-DOS a Windows je adresář nazýván **složkou** a uživatel jej jako soubor nevidí. Adresáře pomáhají udržovat organizační strukturu souborů a uchovávají také jejich atributy. Tvoří hierarchickou strukturu, tzv. adresářový strom, jehož počátečním elementem je tzv. kořenový adresář. Typickými operacemi nad adresářem jsou hledání souboru, vypsání obsahu adresáře (seznamu souborů), přejmenování, vytvoření a smazání souboru.

Kvůli jednoznačnosti nemohou v jednom adresáři existovat dvě položky se shodným jménem (v tomto případě se nerozlišuje mezi souborem a adresářem). Některé souborové systémy neumožňují vznik podadresářů, jiné naopak umožňují vznik neorientovaných cyklů (smyček) pomocí symbolických odkazů.[83]

Cesta

Umístění adresáře v hierarchickém uspořádání v podobě čitelné člověkem určuje **cesta** (*path*). Může být *absolutní* (vyjádřená od kořene souborového systému), nebo *relativní* (vztažená k aktuálnímu adresáři). Relativní cesta může obsahovat zástupné znaky pro kořenový, aktuální, domovský a rodičovský adresář, v OS Windows také název disku.

6.4.2 Organizace dat na discích

Oddíl

Disky jsou obvykle rozděleny na **diskové oddíly** (*partition*). Souborový systém se rozkládá vždy jen na konkrétním oddílu, nikoli tedy nutně na celém disku. To umožňuje mít na jednom disku více nezávislých souborových systémů různého typu. Vytvoření více diskových oddílů má řadu nesporných výhod, například:

- Na jednom počítači je možné mít více operačních systémů. Při startu počítače si vybereme ten, který chceme používat (ostatní můžeme provozovat virtuálně).
- Operační systém a data programů jsou odděleny od uživatelských souborů. To umožňuje vytvářet záložní obrazy pouze z operačního systému a instalovaného softwaru.
- Je možné mít oddělené místo pro stránkování paměti.[84]

[82] Znaková zařízení pracují vždy s jednotlivými znaky (např. klávesnice, tiskárna, terminál). Bloková zařízení pracují vždy s celými bloky dat (např. jednotky vnějších pamětí). V operačních systémech bývají implementovány různě, například v operačním systému Windows jsou bloková zařízení označována písmeny.

[83] Pevný odkaz (*hard link*) se vyznačuje tím, že konkrétní data v souborovém systému jsou přístupná pomocí více jmen, přičemž není podstatné, které jméno je pro jejich dosažení použito. Symbolický odkaz (*symbolic link, symlink, soft link*) je speciální soubor, který obsahuje pouze odkaz na jiný soubor nebo adresář. Při otevření se pracuje s obsahem souboru určeným odkazem, při mazání se smaže pouze odkaz. Operační systémy typu Unix podporují oba typy odkazů, v operačním systému Windows existovala pouze určitá obdoba symbolických odkazů v podobě zástupců, skutečné odkazy jsou podporovány až od verze Vista a nejsou zpětně kompatibilní. Detaily ponechme do navazujících předmětů.

[84] Stránkování paměti (*paging*) je metoda správy paměti, při níž počítač ukládá a načítá data ze sekundárního úložiště (disku) pro použití v hlavní paměti. Je důležitou součástí implementace virtuálních pamětí v moderních operačních systémech.

- Softwarová cache paměť může být oddělena od ostatních souborů. Tato paměť může dynamicky zvětšovat svou velikost a eventuálně i zaplnit souborový systém.
- Data jsou chráněna izolací souborů. Logická poškození disku zasahují vždy jeden oddíl, ostatní zůstanou nepoškozeny. Možnost zcela smazat systémový oddíl navíc zjednodušuje reinstalaci operačního systému.
- Zvýší se celkový výkon počítače v případě, kdy je menší souborový systém efektivnější. Přístup k souborům na jednom velkém oddílu může být pomalejší než při použití dvou menších oddílů.

Na druhou stranu je třeba také zmínit, že vytvoření více oddílů na jednom disku přináší i určité nevýhody. Například:

- Snižuje se celkové místo dostupné pro ukládání dat na disk. Operační systém musí vytvořit na každém oddílu speciální oblast pro správu souborového systému.
- Snižuje se celkový výkon disku v systémech, kde je často k datům přistupováno paralelně na více oddílech, neboť čtecí a zápisová hlava disku je nucena se přesouvat podle požadavků mezi různými místy, aby mohla přistupovat k datům na každém oddílu. Tento problém se netýká disků typu SSD, které na rozdíl od klasických pevných disků neobsahují pohyblivé mechanické části.
- Není možné využít celou kapacitu disku. Tento problém se týká souborů, jejichž velikost je menší než kapacita disku, ale současně větší než velikosti dílčích oddílů.
- Zpomalí se přesun dat mezi různými diskovými oddíly jednoho pevného disku. Operační systém musí zkopírovat celý obsah přesouvaných dat i přesto, že data zůstávají fyzicky na stejném disku.

Sektor

Nejmenším teoreticky adresovatelným blokem vnější paměti je **sektor** (*sector*). Formát IBM definuje sektory pevného disku o velikosti 512 B, ale novější disky s pokročilým formátováním (*Advanced Format Drive*) mají fyzické sektory o velikosti 4 KiB.

Cluster

Nejmenší logickou částí souborového systému a tedy nejmenším skutečně adresovatelným blokem vnější paměti je **alokační blok** (*cluster*). Jeho velikost se liší v závislosti na použitém souborovém systému a především na kapacitě pevného disku. Alokační blok zabírá prostor vždy nejméně jednoho sektoru, v praxi 2^n sektorů.

Volba velikosti clusteru je vždy kompromisem mezi rychlostí práce souborového systému a efektivním využitím místa na disku. Již víme, že při ukládání malých souborů do příliš velkých alokačních bloků dochází k **vnitřní (souborové) fragmentaci** (*file fragmentation*), kterou lze eliminovat například použitím komprese (viz kap. 6.3, str. 152).

Disková fragmentace

Každý soubor je na disku rozdělen na části o velikosti alokačního bloku, přičemž v současnosti se nepoužívají bloky menší než 4 KiB. Jednotlivé části souboru ale nemusejí být na disku uloženy za sebou a mohou se mezi nimi nacházet volné oblasti. Volné místo však není kompaktní, práce s daty se tím zpomaluje. Tento jev nazýváme **vnější (disková) fragmentace** (*free space fragmentation*). Dopady vnější fragmentace lze zmírnit použitím větších alokačních jednotek, ovšem za cenu nárůstu vnitřní fragmentace. Efektivnějším řešením se v tomto případě jeví použití nástroje pro defragmentaci, který je součástí operačního systému. Defragmentaci lze provádět v určitých intervalech (typicky ve Windows), nebo lze data ukládat rovnou tak, aby byla fragmentace snížena (např. v Linuxu, viz str. 168).

Příklad 6.13

Následující příklad nám objasní, proč vždy nejsou jednotlivé části souboru uloženy bezprostředně za sebou a proč se na disku vyskytují volné oblasti.

1. Předpokládejme pevný disk s 16 clustery, které jsou na počátku prázdné.

2. Vytvoříme soubory A, B, C a D s potřebným počtem clusterů (1, 4, 2 a 3):

A_1	B_1	B_2	B_3	B_4	C_1	C_2	D_1	D_2	D_3						

3. Smažeme soubor C:

A_1	B_1	B_2	B_3	B_4			D_1	D_2	D_3						

4. Vytvoříme nový soubor E se třemi clustery:

A_1	B_1	B_2	B_3	B_4	E_1	E_2	D_1	D_2	D_3	E_3					

5. Smažeme soubory A a E:

	B_1	B_2	B_3	B_4			D_1	D_2	D_3						

6. Vytvoříme soubor F se čtyřmi clustery:

F_1	B_1	B_2	B_3	B_4	F_2	F_3	D_1	D_2	D_3	F_4					

7. Po provedené defragmentaci disku:

B_1	B_2	B_3	B_4	F_1	F_2	F_3	F_4	D_1	D_2	D_3					

CDFS, UDF

Při ukládání dat na optické disky CD je využíván souborový systém **CDFS** (*Compact Disc File System*), který je znám spíše pod označením standardu ISO 9660. Z tohoto standardu je odvozen také tzv. obraz disku (*ISO image*), i když u DVD a Blu-ray disků je používán otevřený formát souborového systému známý jako **UDF** (*Universal Disk Format*), který byl navržen jako náhrada ISO 9660. Formát zajišťuje výměnu dat mezi různými operačními systémy a snižuje množství omezení, kterými ISO 9660 trpí. Může být použit pro téměř všechna známá média (CD, DVD, Blu-ray ve všech variantách).

V následujícím textu se stručně seznámíme se souborovými systémy podporovanými v různých operačních systémech. Budou nás zajímat především jejich základní vlastnosti, možnosti použití a případná omezení. Podrobnější výklad problematiky ponechme do dalších předmětů.

6.4.3 Souborové systémy pro MS-DOS a Windows

FAT

Souborový systém **FAT** (*File Allocation Table*) je velmi jednoduchý, a proto je podporován prakticky všemi operačními systémy. Kvůli své jednoduchosti a rozšíření je velmi často používán i na výměnných médiích (například na disketě nebo USB flash disku).

Základem systému je alokační tabulka obsahující informace o obsazení clusterů jednotlivými soubory a adresáři. Soubory mají jména ve tvaru 8 . 3 (osm znaků pro název, tři znaky pro rozšíření). Souborový systém nepodporuje nastavení oprávnění k souborům a nemá ani ochranu proti poškození dat. Disk je rozdělen na několik úseků: boot sektor s informacemi o svazku, dvě FAT (navzájem se zálohující), kořenový adresář a samotná data.

V roce 1980 byla vytvořena první verze známá jako *FAT12*. Byla určena pro první verzi operačního systému QDOS, předchůdce systému MS-DOS. Tato verze obsahovala pouze kořenový adresář a neuměla vytvářet podadresáře. Vyznačovala se 12bitovým adresováním[85] (odtud název) a clustery o velikosti 8 KiB. Bylo tedy možné adresovat maximálně 4 096 clusterů a velikost oddílu byla maximálně 32 MiB. Dnes se tato verze již nepoužívá a není ani podporována žádným současným operačním systémem.

V roce 1983 byla vytvořena verze *FAT16*, ale první opravdu použitelná varianta byla k dispozici až v roce 1987. Jak již napovídá název, adresování bylo 16bitové a velikost clusteru až 32 KiB. Bylo tedy možné adresovat maximálně 65 536 clusterů a velikost oddílu byla maximálně 2 GiB. Operační systém Windows NT 4.0 používal clustery o velikosti 64 KiB, což umožňovalo zvýšení maximální velikosti oddílu na 4 GiB[86].

V roce 1995 vyšlo pro operační systém Windows 95 další rozšíření *Virtual FAT* (VFAT), které přineslo revoluční novinku v podobě podpory dlouhých jmen souborů (až 255 znaků). Požadovaného efektu bylo dosaženo tak, že každý soubor s dlouhým názvem obsahuje v seznamu adresářových položek (v seznamu souborů v adresáři) více záznamů umístěných bezprostředně za sebou, přičemž až poslední záznam je standardním záznamem typu FAT. Předchozí záznamy mají příznaky souboru nastavené tak, aby byly staršími operačními systémy ignorovány.

Pro operační systémy Windows 98, 2000, ME a XP byla určena verze *FAT32*, která se objevila v roce 1997. Velikost clusteru byla 4–32 KiB, velikost oddílu byla omezena systémem BIOS na maximálně 8 GiB. V důsledku toho mohla být velikost disku teoreticky až 2 TiB, což v té době bylo nemyslitelné. Maximální velikost souboru byla 4 GiB, což znemožňovalo například uložení obrazu DVD. Tato verze souborového systému již obsahovala záložní FAT, ze které bylo možné data v případě poškození obnovit.

V roce 2006 bylo nezávislými programátory publikováno rozšíření *FAT+*, které zvyšuje maximální velikost souboru až na 256 GiB.

V roce 2007 se objevila proprietární verze *Extended FAT* (exFAT, FAT64), která byla určena pro přenosné flash disky jako náhrada FAT32. Systém podporuje disky do velikosti až 512 TiB, má podporu transakcí, ale nepodporuje šifrování a kompresi. Je nativně podporován v operačním systému Windows od verze 7, Mac OS X od verze 10.6.5, pro Linux existuje uzavřený licencovaný ovladač a dokonce lze najít i open-source podporu pro MS-DOS.

Souborový systém FAT je obecně zatížen celou řadou nepříjemných problémů a chyb, z nichž stojí za povšimnutí především tyto:

- Clustery nenásledují hned za sebou, což zvyšuje diskovou fragmentaci.
- Nejprve se ukládají data, až poté jméno souboru. Dojde-li při ukládání k chybě, není dohledatelný obraz ani samotná data.
- Složka je v kořenovém adresáři zapsána jako soubor s číslem ukazujícím na první cluster. Pokud se toto číslo neuloží, jsou všechna data ve složce ztracena.

[85] To znamená, že bylo možné adresovat 2^{12} = 4 096 clusterů. Uvědomíme-li si, že v každém clusteru může být uložen nejvýše jeden soubor (u větších souborů jeden jejich fragment), snadno si odvodíme, že maximální počet souborů nemůže být větší než počet clusterů.

[86] Počet clusterů a velikost oddílu spolu velmi úzce souvisí. Počet clusterů je omezen adresováním, velikost oddílu je odvozena od počtu a velikosti clusterů. Při 16bitovém adresování lze pracovat až s 65 536 clustery, odtud při velikosti clusteru 64 KiB dojdeme k maximální kapacitě 4 GiB.

- Může dojít k překřížení souborů, kdy více záznamů FAT ukazuje na stejný cluster.
- Snadno může dojít k poškození FAT, ať už použitím softwaru pro správu diskových oddílů, chybou uživatele, selháním programu, virovou nákazou či nekorektní instalací operačního systému.

NTFS První verze souborového systému **NTFS** (*New Technology File System*) byla vytvořena již v roce 1993 pro Windows NT 3.1, zatím poslední verze z roku 2001 je používána od Windows XP výše. Podporuje 64bitové adresování a velikost clusteru od 0,5 KiB až po 64 KiB, nejčastěji však 4 KiB. Důsledkem je teoretická velikost oddílu až $2^{64} - 1$ clusterů, velikost souboru až $2^{64} - 2^{10}$ B a maximální počet souborů $2^{32} - 1$, tj. přes 4 miliardy. Velmi nepříjemným je omezení maximální délky specifikace souboru na 255 znaků v UTF-16, a to včetně cesty. Systém nerozlišuje mezi velkými a malými písmeny. Je podporováno žurnálování, komprese a šifrování souborů, ale kompresní algoritmy jsou navrženy pro clustery do velikosti 4 KiB. Systém podporuje uživatelské kvóty a tzv. ACL (*Access Control List*) pro přidělování práv k souborům a adresářům. Adresář je v NTFS chápán jako specifický soubor nesoucí odkazy na ostatní soubory a adresáře, které obsahuje. Oproti FAT32 je systém NTFS bezpečnější, spolehlivější a méně náchylný k chybám.

6.4.4 Souborové systémy pro Unix/Linux

Struktura souborových systémů pro unixové operační systémy se skládá z těchto částí:

- *zaváděcí blok* – malý program sloužící k zavedení jádra operačního systému do paměti;
- *superblok* – popisuje parametry a stav souborového systému, udává velikost a počet volných bloků a čas poslední kontroly;
- *seznam i-uzlů* – každý soubor je popsán právě jedním i-uzlem, každý i-uzel má unikátní číslo, ale neobsahuje název souboru (ten s číslem i-uzlu spojuje až adresář);
- *seznam datových bloků* – zde jsou uloženy soubory.

Každý záznam v i-uzlu obsahuje informace o typu souboru (běžný soubor, adresář, link apod.), přístupová práva pro tři úrovně uživatelů (vlastníka, skupinu a ostatní), počet odkazů na soubor (i-uzel), vlastníka souboru a jeho skupinu, velikost souboru v bajtech, datum vytvoření, poslední změny a posledního přístupu a adresy datových bloků.

Pro operační systémy typu Unix je k dispozici celá řada různých souborových systémů. Většinou se jedná o vyspělé systémy s vhodnými vlastnostmi, které podporují transakční zpracování a ochranu dat, diskové kvóty a mechanismy proti fragmentaci. Většinou ale chybí podpora komprese a šifrování dat. Za zmínku stojí zejména tyto:

UFS
- **UFS** (*Unix File System*) – tradiční systém pro Unix, který přímo vychází z historického unixového systému (ten ani neměl žádný název).

JFS
- **JFS** (*Journaled File System*) – vyvinut v roce 1990 pro použití na serverech, největší důraz je proto kladen na propustnost a spolehlivost.

ext
- **ext** (*extended file system*) – vyvíjen od roku 1992, vychází z UFS. Od prvních verzí prakticky nedošlo k zásadním změnám. Od verze *ext3* nabízí žurnálování, stromovou implementaci indexů souborů v adresáři, omezení fragmentace již při ukládání souborů na disk a možnost změny velikosti souborového systému za běhu. Od verze *ext4*

podporuje nanosekundová časová razítka, optimalizaci alokací a on-line defragmentaci na úrovni souborů nebo celého systému. Každá novější verze je zpětně i dopředně kompatibilní. Limity pro clustery, oddíly a soubory jsou uvedeny v tab. 6.4.

Tabulka 6.4: *Limity souborového systému ext*

Verze	Rok	Velikost clusteru	Velikost oddílu	Velikost souboru
ext2	1993	1–8 KiB	2–32 TiB	16 GiB až 64 TiB
ext3	2001	1–4 KiB	4–16 TiB	16 GiB až 2 TiB
ext4	2008	4 KiB	1 EiB	16 TiB

- **XFS** (*Extended File System*) – výkonný 64bitový žurnálovací systém určený především pro disková pole na velkých serverech. První verze pochází z roku 1993. *XFS*
- **ReiserFS** – od roku 2001, může být implementován i pro operační systém Windows. Jedná se o první žurnálovací systém, který byl přidán do jádra Linuxu. *ReiserFS*
- **ZFS** (*Zettabyte File System*) – od roku 2005, vyvinut pro operační systém Solaris. Systém dokáže pracovat s oddílem o velikosti až 256 ZiB (2^{78} B). *ZFS*
- **btrfs** (*B-tree file system*) – vyvíjí Oracle od roku 2012, má přinést některé vlastnosti ZFS do prostředí Linuxu, ve kterém ZFS nelze použít z důvodu nekompatibilních licencí. *btrfs*

Pro úplnost dodejme, že operační systémy Unix/Linux bez problémů přečtou souborový systém VFAT, specializované programy pak i souborový systém NTFS.

6.4.5 Souborové systémy pro macOS

Od roku 1998 používala společnost Apple souborový systém **HFS+** (*Hierarchical File System Plus*), též známý jako *Mac OS Extended*. Využití našel na počítačích s OS X (iMac, MacBook apod.) a také v přehrávačích iPod. Jedná se žurnálovací systém s ochranou proti fragmentaci, který podporuje disky až do velikosti 8 EiB. Na jednom disku mohou být maximálně 4,3 miliardy souborů. Soubory lze šifrovat a komprimovat na pozadí. Skupiny uživatelů a přístupových práv jsou stejné jako v Unixu. *HFS+*

V roce 2007 byl vytvořen proprietární souborový systém **APFS** (*Apple File System*) pro 64bitová iOS zařízení a pro macOS (od verze 10.13). Cílem bylo odstranění problémů HFS+. Nový systém je optimalizován pro HDD i Flash/SSD disky s důrazem na šifrování. *APFS*

Kontrolní otázky

21. Co je souborový systém a k čemu slouží?
22. Jaký je rozdíl mezi souborem a adresářem? Jak je adresář implementován?
23. Jaký je rozdíl mezi absolutní a relativní cestou?
24. Co je diskový oddíl, sektor a cluster?
25. Proč dochází k diskové fragmentaci a jak ji lze eliminovat?
26. Které souborové systémy jsou v současnosti podporovány v operačních systémech Windows, Unix/Linux a macOS?

6.5 Příklady na procvičení

Příklad 6.14 Stáhněte si archiv s pracovními soubory[87] a proveďte dekompresi do předem připraveného adresáře. Dále postupujte takto:

1. U každého souboru zjistěte, zda je textový, nebo netextový.
2. U každého textového souboru dále proveďte tyto kroky:
   a) Zjistěte, zda jde o čistý text, nebo o rozšířený text, tedy zda obsahuje národní znaky. Pokud je to rozšířený text, zjistěte použité kódování národních znaků.
   b) Proveďte změnu kódování národních znaků u vybraného rozšířeného textového souboru na UTF-8 a ověřte, že změna proběhla korektně.
   c) Zjistěte, pro který operační systém je soubor připraven.
3. Určete souborový formát všech souborů z archivu a přejmenujte je tak, aby jejich rozšíření odpovídalo skutečnosti.
4. Vyhledejte ke každému zjištěnému souborovému formátu asociovanou aplikaci, kterou je možné soubor korektně otevřít.
5. Prohlédněte si soubor v příslušné aplikaci a zjistěte, co se v souboru nachází.

Příklad 6.15 Vyzkoušejte kompresi libovolného delšího nekomprimovaného souboru. Zjistěte název souboru, který se kompresí vytvořil. Archiv si prohlédněte vhodným nástrojem pro zobrazení obsahu souboru. Prozkoumejte, jaký vliv na úspěšnost komprese má volba kompresního softwaru a nastavení rychlosti a účinnosti kompresního algoritmu.

Příklad 6.16 Vytvořte textovým editorem alespoň tři velmi krátké soubory (délka max. 15 znaků, obsahem běžná česká slova). Tyto soubory poté zkomprimujte do jednoho archivního souboru. Porovnejte délky komprimovaných souborů a výsledného archivu. Jak se nazývá jev, ke kterému došlo? Co je příčinou jeho vzniku?

Příklad 6.17 Stáhněte si archiv s pracovními soubory[87] a proveďte dekompresi do předem připraveného adresáře. Stanovte kompresní poměr pro jednotlivé souborové formáty v archivu a zjistěte, které formáty má smysl komprimovat za účelem snížení objemu dat.

Shrnutí

Studium principů uložení dat v počítači jsme zakončili kapitolou věnovanou uložení dat v souborech. Objasnili jsme si pojem souborový formát, rozdělili formáty do dvou skupin a naučili se je rozpoznávat. Dozvěděli jsme se, jak funguje asociace souborových formátů s aplikacemi, známe podstatu konverze formátů a jsme schopni ji provést. Dále jsme se věnovali problematice komprese dat a seznámili se s několika metodami používanými pro různé účely. V závěru kapitoly jsme se stručně zabývali koncepcí souborů a adresářů, způsobem organizace souborů na discích a implementací pod různými operačními systémy.

Slovníček pojmů

Adaptivní komprese: metoda přizpůsobená aktuálním datům na základě jejich analýzy.
Adresář (složka): pojmenovaná množina jiných adresářů a souborů.
Alokační blok (cluster): nejmenší logická část souborového systému složená ze sektorů.
APFS: souborový systém pro operační systém macOS.
Archiv: soubor vytvořený komprimátorem.
Aritmetické kódování: kompresní metoda založená na kódování zprávy do jednoho čísla.
Asociační tabulka: struktura přiřazující k formátu souboru obslužnou aplikaci.
Asymetrická komprese: metoda, při níž je komprese časově náročnější než dekomprese.
CCITT: neztrátová kompresní metoda pro faksimilní monochromatická data.
CDFS: souborový systém pro optické disky (CD).
Cesta: informace o umístění adresáře nebo souboru v hierarchickém uspořádání.
Čistá data: data po kompresi.
Čistý (prostý) text: textový soubor neobsahující národní znaky.
Datový formát: způsob uspořádání dat v paměti počítače, který na rozdíl od jednoduchých datových typů určuje i strukturu dat.
DCT: transformační metoda, která převádí signály z časové oblasti do oblasti frekvenční.
Deflate: neztrátová kompresní metoda pro libovolná data.
Diskový oddíl (partition): disk nebo jeho část, na kterém se rozkládá souborový systém.
Dokument: soubor, který kromě samotného textu obsahuje i formátovací značky.
ext: množina souborových systémů pro linuxové operační systémy.
FAT: souborový systém pro operační systémy MS-DOS a Windows.
Formátová specifikace: popis významu každého bitu nebo bajtu v datovém formátu.
Fyzická komprese: zmenšení objemu dat bez ohledu na jejich význam.
HFS+: souborový systém pro operační systém OS X.
Hlavička formátu: identifikace souborového formátu na prvních bajtech souboru.
Hrubá data: data před kompresí.
JPEG: ztrátová kompresní metoda pro rastrová obrazová data.
Komprese (komprimace): zmenšování objemu dat při zachování důležitých informací.
Kompresní poměr: hodnota udávající poměr objemu hrubých a čistých dat.
Komprimátor: program pro off-line kompresi a archivaci dat.
Kontejner: multimediální soubor obsahující více proudů dat se vzájemnou synchronizací.
Konvertor: specializovaný program umožňující konverzi souborových formátů.
Konverze: převod souboru z jednoho souborovém formátu na jiný.
Logická komprese: zmenšení objemu dat s využitím znalosti jejich významu.
LZMA: neztrátová kompresní metoda založená na Markovových řetězcích.
LZW: neztrátová kompresní metoda založená na kratším vyjádření opakujících se posloupností dat na základě vytvořeného slovníku.
MPEG: ztrátová kompresní metoda pro multimediální data.
Netextový (binární) formát: způsob uložení dat obsahující zobrazitelné i řídicí znaky.
Neztrátová komprese: způsob ukládání dat umožňující jejich úplnou rekonstrukci.
NTFS: souborový systém pro operační systém Windows.
Off-line komprese: komprese prováděná na vyžádání specializovanými programy.

On-line komprese: komprese prováděná při zpracování dat bez vědomí uživatele.
Otevřený (svobodný) formát: formát, ke kterému je volně dostupná specifikace.
Proudové kódování: kompresní metoda založená na kódování posloupnosti stejných hodnot v hrubých datech.
Přenositelnost formátu: možnost zpracování souborového formátu různými verzemi různých aplikací pod různými operačními systémy.
Přípona (rozšíření): součást jména souboru, která naznačuje souborový formát.
Režijní informace: záznam o struktuře archivu.
RLE: neztrátová kompresní metoda zkracující proud stejných znaků.
Rozšířený text: textový formát obsahující národní znaky.
Sektor: nejmenší teoreticky adresovatelný blok vnější paměti.
Soubor: pojmenovaná konečná posloupnost bajtů s definovaným umístěním v paměti.
Souborový formát: datový formát, v němž jsou data uložena v souboru.
Souborový systém: způsob organizace souborů na elektronickém paměťovém médiu.
Stránkování paměti: metoda správy paměti, při níž se data odkládají z primárního úložiště na sekundární (disk).
Symetrická komprese: metoda, při níž je komprese i dekomprese stejně časově náročná.
Textový formát: způsob uložení dat obsahující zobrazitelné znaky, řídicí znak pro konec řádku a nejvýše jeden řídicí znak pro konec souboru.
UDF: souborový systém pro optická média (CD, DVD, Blu-ray).
Uzavřený (proprietární) formát: formát bez volně dostupné formátové specifikace.
Vnější (disková) fragmentace: alokační bloky patřící jednomu souboru rozmístěné neuspořádaně po celém oddílu.
Vnitřní (souborová) fragmentace: nevyužité místo uvnitř alokačních bloků na disku.
Záporná komprese: negativní jev, při kterém objem dat kompresí narůstá.
Ztrátová komprese: způsob ukládání dat, při kterém se nepodstatná data vynechají.
Žurnálování: uchovávání chronologického záznamu všech prováděných činností.

Řešení kontrolních otázek

1. *Jaký je rozdíl mezi datovým typem a datovým formátem?*

Datový formát oproti datovému typu navíc určuje i strukturu dat.

2. *Co určuje formátová specifikace?*

Formátová specifikace definuje význam jednotlivých bitů či bajtů v reprezentaci dat.

3. *Co musí být splněno, aby byla data v textovém formátu?*

Data v textovém formátu jsou připravena pro zobrazení a přímé čtení člověkem. Obsahují zobrazitelné znaky, řídicí znak konce řádku a nejvýše jeden řídicí znak konce souboru.

4. *Jak je v textových souborech implementován konec řádku?*

Implementace konce řádku se liší v závislosti na používaném operačním systému. V OS Windows je implementován dvěma řídicími znaky, v ostatních systémech jedním.

5. *Co je to souborový formát?*

Souborový formát vyjadřuje přesný popis způsobu uložení dat v souborech, může být textový, nebo netextový (binární).

6. Jaké jsou výhody otevřených formátů?

Otevřené formáty mají volně dostupnou specifikaci, což umožňuje jejich efektivní využití a zpracování uložených dat mnoha volně dostupnými programy.

7. Jaké komplikace mohou nastat při práci s rozšířeným textovým formátem?

Rozšířený textový formát obsahuje národní znaky. Pro správné zobrazení obsahu souboru je nutné zvolit správné kódování těchto znaků.

8. V čem spočívá princip asociace formátu a aplikace?

Asociace formátu a aplikace přiřazuje každému formátu obslužnou aplikaci, která s ním umí pracovat. Tento vztah je uchováván v asociační tabulce.

9. Na čem závisí přenositelnost formátu? Které formáty jsou dobře přenositelné?

Přenositelnost formátu závisí na počtu programů, které s ním umějí pracovat. Z tohoto pohledu jsou nejlépe přenositelné otevřené formáty (ideálně textové), které jsou zpracovatelné na libovolné platformě a v libovolném operačním systému.

10. Používají se pro zpracování textů pouze textové formáty?

Nikoli, masivně se používají i binární formáty – například DOCX nebo oblíbený PDF.

11. K čemu slouží konverze souborových formátů a jak ji lze provést?

Konverze slouží ke změně způsobu uložení dat beze změny informační obsahu. Data v jednom formátu se konvertují do jiného formátu s odlišnými vyjadřovacími schopnostmi. Provádí se běžnými službami „Otevřít“ a „Uložit jako“, případně specializovanými programy.

12. K čemu se používá ztrátová komprese?

Ztrátová komprese se používá ke zmenšení objemu audiovizuálních dat, u kterých odstranění některých informací nevadí, neboť průměrné lidské smysly ztrátu nepostřehnou.

13. Co je vnitřní fragmentace a jak ji lze eliminovat?

Vnitřní fragmentace vzniká nevyužitím kapacity posledního clusteru při ukládání souboru na disk. Eliminovat ji lze kompresí většího počtu menších souborů do jednoho archivu.

14. Jaký je princip fyzické komprese?

Fyzická komprese je založena na zmenšování objemu dat bez ohledu na jejich význam. Na základě analýzy vstupních dat se hledá kratší kód pro jejich vyjádření.

15. Co udává kompresní poměr a jak jej lze spočítat?

Kompresní poměr umožňuje hodnotit kvalitu provedené komprese. Udává se jako poměr hrubých a čistých dat a lze jej vyjádřit jako násobek hrubých dat proti čistým, příp. procentuálním úbytkem či zůstatkem hrubých dat.

16. Jaký je princip kompresní metody RLE?

Kompresní metoda RLE je založena na kódování posloupností stejných hodnot, tzv. proudů, kratším kódem. Není adaptivní a je určena pro obrazová data.

17. Čím se liší metoda LZW od metody RLE?

Kompresní metoda LZW je adaptivní a použitelná na libovolná data. Principem je analýza vstupních dat, vyhledávání opakujících se posloupností z nich vytvořením vlastního slovníku a jejich vyjádření kratším kódem na základě vybudovaného slovníku.

18. Které kompresní metody se používají pro obrazová data?

Pro kompresi obrazových dat se používají neztrátové metody RLE, LZW, Deflate, CCITT aj. a ztrátová metoda JPEG.

19. Co je záporná komprese a kdy vzniká?

K záporné kompresi dochází v případě, kdy hrubá data již nelze významně zkomprimovat. Výsledný archiv je potom větší, než byla původní data.

20. Jaké služby nabízejí komprimátory při vytváření archivů?

Komprimátory umožňují kromě vlastní komprese také mnoho dalších funkcí (práce s adresáři, vkládání poznámek, šifrování archivu apod.) a možnost volby účinnosti kompresního algoritmu (lepší kompresní poměr × rychlejší komprese).

21. Co je souborový systém a k čemu slouží?

Souborový systém je způsob organizace souborů ukládaných na elektronická paměťová média. Určuje způsob vytváření, ukládání a přístupu k souborům a adresářům, hlídá volné místo apod. Tvoří rozhraní mezi vnější paměti a operačním systémem.

22. Jaký je rozdíl mezi souborem a adresářem? Jak je adresář implementován?

Soubor je pojmenovaná konečná posloupnost bajtů s definovaným umístěním ve vnější paměti, adresář je pojmenovaná množina jiných adresářů či souborů. Obvykle je adresář implementován jako speciální soubor s odkazy na soubory a adresáře, které obsahuje.

23. Jaký je rozdíl mezi absolutní a relativní cestou?

Absolutní cesta k souboru vychází od kořenového adresáře souborového systému, relativní cesta je vztažena k aktuálnímu pracovnímu adresáři.

24. Co je diskový oddíl, sektor a cluster?

Diskový oddíl je část disku (nebo i celý disk), na kterém se rozkládá souborový systém. Sektor je nejmenší teoreticky adresovatelným blokem vnější paměti. Cluster (alokační blok) je nejmenší logickou částí souborového systému, je složen ze sektorů.

25. Proč dochází k diskové fragmentaci a jak ji lze eliminovat?

K diskové fragmentaci dochází proto, že fragmenty souborů při ukládání na disk nemusejí být umístěny v po sobě jdoucích clusterech. To významně zpomaluje práci se soubory. Eliminace diskové fragmentace je možná použitím nástroje pro defragmentaci, moderní souborové systémy jsou vybaveny podporou defragmentace již při ukládání souboru na disk.

26. Které souborové systémy jsou v současnosti podporovány v operačních systémech Windows, Unix/Linux a macOS?

V systému Windows jsou podporovány souborové systémy FAT a NTFS. Pro operační systémy třídy Unix lze využít celou řadu souborových systémů, nejznámější jsou ext, UFS, XFS, JFS, ReiserFS, ZFS a btrfs. V systému macOS jsou podporovány souborové systémy HFS+ a novější APFS.

7 Počítačová kriminalita

Úvod

V počátcích celosvětové sítě Internet (koncem 80. let 20. století) na bezpečnost dat nikdo nemyslel. Cílem vytvoření sítě bylo sdílení informací bez omezení. Důsledkem tohoto stavu byla spousta různých bezpečnostních děr, které bylo nutné opravit.[88] Prudký rozvoj osobních počítačů a jejich propojení do sítě představuje silnou zbraň, jejíž účinek si v mnoha případech lze jen stěží představit.

Dnes se ale na ochranu dat velmi dbá, a to jak v souvislosti s ochranou soukromí, tak i s autorskými právy. Problém spočívá v tom, že jednotlivé bity nelze rozlišit, změnu jednotlivého bitu nelze přímo zjistit (pouze za cenu zvýšení redundance – viz kap. 5.3, str. 110) a navíc bity lze donekonečna kopírovat.

Zdroje problémů

Problémy mohou přicházet ze tří stran. Uvádí se, že přibližně polovinu všech škod způsobí *lidské chyby*. Za útokem mohou stát také naši *známí nepřátelé* (zneuznaní zaměstnanci nebo obecně lidé, kteří nás nemají v oblibě). Třetí skupinu útočníků tvoří *neznámí hackeři*, kteří mají vyhledávání bezpečnostních děr jako svůj koníček. Většina z nich se řídí hackerskou etikou a jejich počínání je pro bezpečnost velmi užitečné.

Cíl

Cílem této kapitoly je seznámení s různými formami nelegálních aktivit spojených s počítačem. Zaměříme na porušování autorských práv, útoky na bezpečnost systémů, šíření nevyžádaných sdělení, získávání neoprávněných informací a napadání škodlivým softwarem. Na závěr se podíváme na procesy, které zvyšují bezpečnost ukládaných dat – archivace, zálohování a synchronizace.

Návaznosti

Pro studium této kapitoly je nezbytné orientovat se v základních principech činnosti počítače a operačního systému a chápat princip práce se soubory v běžných souborových systémech.

Studijní čas

Čas potřebný k nastudování této kapitoly je asi 8 hodin.

7.1 Porušování autorského práva a souvisejících práv

Programy (a také některé datové soubory) mají svého autora, který je držitelem práv. Autorem může být fyzická osoba i organizace. Je přirozené, že autor chce za své dílo získat finanční odměnu, pokud se autorské odměny dobrovolně nevzdá.

[88] Není to ale jen problém Internetu. Jsou známy případy již ze 60. let 20. století, kdy pachatel pomocí modifikace bankovního počítačového programu neoprávněně získával finanční prostředky na svůj účet.

Počítačové pirátství

Užití programů nebo datových souborů bez vědomí autora a bez zaplacení příslušných poplatků je porušením autorského zákona[89] a také porušením trestního zákona[90]. Označuje se jako **počítačové pirátství** (též softwarové pirátství, *copyright infringement of software*) a má mnoho podob od neoprávněného užití programu nebo dat přes nelegální kopírování až po nelegální placenou distribuci ukradeného softwaru.

Každý softwarový produkt je chráněn licencí, která je uzavřena mezi autorem a uživatelem díla, přičemž autor určuje, jakým způsobem může uživatel s dílem nakládat. Pokud se zamyslíme nad důvody, proč vůbec existují licence na softwarové produkty, nalezneme jich hned několik. Předně je třeba si uvědomit, že software není rohlík, jehož hodnota je minimální. Dalším důvodem je to, že software doopravdy nevlastníme, pouze vlastníme *oprávnění* jej používat. Software se také (na rozdíl od rohlíku) nezkazí a je svým způsobem věčný. Proto musíme každý software používat ve shodě s licencí.

7.1.1 Kategorie softwaru

Komerční software

Programový produkt, který je šířen za úplatu, označujeme jako **komerční** (*commercial software*, též *payware*). Takový software je nutné zakoupit nebo pronajmout a jeho používání je omezeno danou licencí (viz kap. 7.1.2). Často je tak omezen počet instalací software současně, přenositelnost licence či právo modifikace produktu. Do této kategorie softwarových produktů patří například operační systém Microsoft Windows, kancelářský balík Microsoft Office nebo rastrový grafický editor Adobe Photoshop.

Proprietární software

Proprietární software (*proprietary software*), též nesvobodný software nebo software s uzavřeným kódem (*closed-source*) má podmínky užití upraveny licencí a jeho zdrojové kódy nejsou volně dostupné. Uživatelé nejsou oprávněni software volně šířit ani v něm provádět jakékoliv úpravy. Z praktického pohledu se jedná například o možnost instalace softwaru na omezený počet pracovních stanic. Software bývá dodáván s bezpečnostními opatřeními, např. přístupovým klíčem. Proprietární software obvykle řadíme do kategorie komerčního softwaru, ale ne každý komerční software je zároveň proprietární.

Svobodný software

Pod pojmem **svobodný software** (*free software*) rozumíme programový produkt, ke kterému je k dispozici zdrojový kód spolu s právem software používat, modifikovat a distribuovat. Svobodný software nemusí být nekomerční. Je tomu právě naopak – musí být dostupný pro komerční využití. Komerční vývoj není ničím neobvyklým, u svobodného softwaru využívaného k vysoce komerčním účelům (např. Red Hat Enterprise Linux) je placení služeb obvyklé. Mezi nejznámější svobodný software patří operační systém Linux, rastrový grafický editor GIMP, webový prohlížeč Firefox nebo kancelářský balík LibreOffice. Množství svobodného softwaru je obrovské a lze nalézt příklad pro téměř každou oblast použití.

Otevřený software

Otevřený software (*open-source software*) je programový produkt dostupný zdarma i se zdrojovými kódy. Otevřenost zde znamená jak technickou dostupnost kódu, tak i legální dostupnost formou licence, která umožňuje při dodržení podmínek uživatelům zdrojový kód upravovat a využívat ve svém softwaru. Otevřené operační systémy (například Linux nebo na jeho jádře založený Android) pohánějí miliony serverů, stolních počítačů, chytrých telefonů a dalších zařízení.

[89] Zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů. Poslední novela tohoto zákona nabyla účinnosti v roce 2022.
[90] Zákon č. 40/2009 Sb., trestní zákoník. Poslední novela tohoto zákona nabyla účinnosti v roce 2022.

7.1.2 Softwarové licence

V praxi se můžeme setkat s celou řadou různých softwarových licencí k softwarovým produktům. Z následujícího neúplného přehledu nejznámějších licencí[91] je zřejmé, že existuje celá řada možností, jak získat software bezplatně, a přitom jej legálně využívat.

K nejsvobodnějším patří **licence BSD**. Umožňuje volné šíření licencovaného obsahu, přičemž vyžaduje pouze uvedení autora a informace o licenci, spolu s upozorněním na zřeknutí se odpovědnosti za dílo. Zkratka BSD označuje *Berkeley Software Distribution*, obchodní organizaci, která licenci vyvinula a používala při vývoji stejnojmenného operačního systému (známého také jako BSD Unix). Licence dovoluje komerční využití, včetně využití v proprietárním softwaru bez zveřejněného zdrojového kódu. *BSD*

Zkratka **EULA** (*End-User Licence Agreement*) označuje licenci pro uživatele proprietárního softwaru (obvykle komerčního). Jeho zdrojový kód může být otevřený, ale výsledný produkt spadá pod licenci EULA, která znemožňuje editaci a šíření. *EULA*

Licence **GNU GPL** (*General Public License*) se uplatňuje na svobodný software, kde autor dodává i zdrojové kódy, které lze libovolně upravovat.[92] Další šíření upraveného softwaru však lze opět jen pod licencí GNU GPL, a to bezplatně nebo za cenu médií. Je možné se setkat s různými úpravami, např. GNU FDL (*Free Documentation License*). *GNU GPL*

Specialitou je licence **OEM** (*Original Equipment Manufacturer*), která je k danému programovému vybavení získána současně se zakoupením hardwaru či jiného softwarového produktu. OEM verze softwaru může mít stejný rozsah funkcí jako plná verze, ale je poměrně časté, že její funkce jsou omezené. Software poskytovaný v OEM licenci je možné používat pouze na tom počítači, na kterém byl výrobcem nainstalován. Pod tímto druhem licence je často prodáván např. operační systém Microsoft Windows. *OEM*

7.1.3 Formy distribuce softwaru

Volné dílo (*public domain*) se vyznačuje tím, že jeho autor se rozhodl jej poskytnout bez nároku na autorská práva. V českém právním systému se nelze vzdát autorských práv, je pouze možné nabídnout bezúplatnou licenci na libovolné využití díla. Zároveň se předpokládá, že autor, který své dílo takto poskytnul, se nebude autorských práv domáhat. *Volné dílo*

Jako **freeware** označujeme software (většinou proprietární), jehož užívání je zdarma, autor si však obvykle vyhrazuje určitá omezení (např. využití pouze pro nekomerční sféru). Z tohoto důvodu často mluvíme o typu softwarové licence. Přesné znění podmínek je zakotveno v licenční smlouvě, která je často specifická pro každý freeware. *Freeware*

Shareware je označení pro formu distribuce proprietárního softwaru. Často je shareware nabízen ke stažení na webu výrobce, nebo jako příloha na CD či DVD. Uživatel může software po určitou dobu zdarma zkoušet, je však povinen řídit se autorovou specifikací licence. Při dlouhodobějším používání je nutné uhradit cenu programu nebo se registrovat. Volně šířený shareware je zpravidla omezen časově nebo funkčně. Následující formy distribuce bývají často chápány jako různé odvozeniny sharewaru. *Shareware*

[91] Některé z uvedených licencí nejsou (přísně vzato) licencemi v pravém slova smyslu, jedná se spíše o formy distribuce softwaru, proto jsou zařazeny do samostatné podkapitoly.

[92] Navzdory chápání laickou veřejností je rozdíl mezi *freewarem*, *svobodným softwarem* a *otevřeným softwarem*. Freeware je proprietární software, zatímco svobodný software a otevřený software lze používat bez omezení.

Donationware

Donationware je licenční model distribuce softwaru, kdy autor požaduje za poskytnutí softwaru příspěvek pro neziskovou či charitativní organizaci (též *charityware* nebo *careware*), případně dobrovolný příspěvek na svůj účet. Jedná se o specifický druh freeware s nepovinnou platbou. Jako příklad můžeme uvést textový editor `vim`.

Trial

Trial (*trialware*) je programový produkt, který je určen k vyzkoušení. Funkčnost je obvykle časově omezena, po uplynutí daného časového intervalu je nutné zakoupit plnou verzi, nebo program odinstalovat. Některé programy z kategorie shareware plnohodnotně fungují i po vypršení testovacího období.[93]

Demo

Pod pojmem **demoverze** (*demoware*) rozumíme obvykle software, který je dostupný zdarma v omezené verzi pouze jako ukázka funkcí plné verze. Omezení nejčastěji spočívá v absenci některých důležitých funkcí (ukládání souborů, vypalování disků apod.) nebo vkládání identifikačních značek do výstupů (např. u grafických editorů). Stejný termín se používá pro software, který funguje pouze jako demonstrační ukázka předváděná koncovému zákazníkovi s cílem ukázat, jak bude program vypadat a fungovat.

Cardware, beerware

Cardware (též *postcardware*) je forma distribuce, kdy autor umožňuje využití díla, pokud mu uživatel zašle pohlednici. Autorem myšlenky je Aaron Giles (* 1970), autor programu JPEGView. Na podobném principu je postavena forma **beerware**, která (jak napovídá název) přikazuje uživateli koupit autorovi pivo, případně vypít pivo na jeho zdraví. Dodržování těchto požadavků zpravidla nebývá právně vymáháno a obecně ani přísně dodržováno.

Abandonware

Pojem **abandonware** (též *orphanwaren* označuje software, který je již zastaralý. Jedná se převážně o počítačové hry, které již nejsou v prodeji a patrně k tomu nedojde ani v žádné speciální edici. Po legální stránce jde o **warez**[94], přesto jej mnoho softwarových společností toleruje nebo dokonce bere pozitivně. Některé společnosti dokonce starší software uvolňují k volnému užití.

7.1.4 Ochrana proti nelegálnímu použití software

Technicky je obrana proti počítačovému kopírování poměrně složitá záležitost. Počítačová kopie je za normálních podmínek od originálu nerozeznatelná a má všechny vlastnosti originálu včetně kvality. Obranu je tedy nutné založit na jiných principech.

Možnosti obrany

Jednou z možností ochrany je registrace uživatelů a následná vzdálená kontrola licenčního čísla produktu a uživatele například při každé aktualizaci. Další možností je tzv. hardwarový klíč obsahující přístupový kód, který je zaznamenán na trvalou paměť. Tato paměť musí být v okamžiku startu programu připojena k počítači. Třetí možností je speciální druh záznamu na distribuovaném médiu, který se při kopii nepřenese v původní podobě, nebo s ním běžné kopírovací služby nedokážou pracovat.

Podstatné však je, že *každé* technické řešení lze nakonec obejít, jen je zapotřebí určitých znalostí, prostředků a úsilí.

[93] Někdy bývá *trialware* považován za jednu z forem *demoverze*, která je omezena časově. Dalšími formami jsou *crippleware* (omezení funkcí) a *adware* (přítomnost otravné reklamy).

[94] *Warez* je slangový termín označující autorská díla, se kterými je nakládáno v rozporu s autorským zákonem. Podle druhu bývá někdy *warez* členěn na *gamez*, *appz*, *crackz* a *moviez*. Nejčastějším způsobem šíření je dnes hlavně Internet.

Kontrolní otázky

1. Co se rozumí pod pojmem počítačové pirátství?
2. Které zákony se zabývají problematikou počítačové kriminality?
3. Jaký je rozdíl mezi komerčním a proprietárním softwarem?
4. Jaký je rozdíl mezi otevřeným a svobodným softwarem?
5. Jaký význam mají licence k softwarovým produktům?
6. Jaká omezení vyplývají z licencí OEM?
7. Které licence a formy distribuce umožňují bezplatné využití softwaru i dat?
8. Které licence a formy distribuce vyžadují platbu za software i data?
9. V čem se zásadně liší freeware a shareware?
10. Čím se vyznačuje volné dílo (public domain)?
11. Jaké jsou možnosti obrany proti počítačovému pirátství?

7.2 Útoky na bezpečnost dat a systémů

V této podkapitole se zaměříme na trestné činy proti důvěryhodnosti, integritě a dostupnosti počítačových dat a systémů.

7.2.1 Hacking a cracking

Hacking

Jednou z nejvýraznějších oblastí počítačové kriminality je pronikání do systémů, tzv. **hacking**. Jedná se o obejití nebo prolomení ochrany a průnik do systému jinou než standardní cestou. Hackeři se vyvinuli z programátorů snažících se zjistit, jak systém funguje, případně odstranit chyby. Postupem času vzniklo několik typů hackerů, z nichž stojí za zmínku tyto:

- *white hats* – uznávají hackerskou etiku, své útoky provádějí za účelem poznání, často s vědomím a na žádost firem zabývajících se bezpečností systémů;
- *black hats* – při své činnosti hledají majetkový či jiný prospěch, jejich činnost lze kvalifikovat i jako protiprávní zásah do systémua bývají často označováni jako *crackeři*;
- *gray hats* – šedá zóna hackerů stojící na pomezí mezi „bílými“ a „černými“;
- *wannabes* – trapní snaživci, kteří spouštějí připravené hackerské programy, aniž by sami něco naprogramovali nebo měli potřebné znalosti.

Cracking

Jak vyplývá z výše uvedeného seznamu, nejnebezpečnější oblastí je tzv. **cracking**, neboli nedovolené vniknutí do chráněného systému s cílem získat pro sebe nějakou neoprávněnou výhodu či poškodit majitele a legální uživatele systému. Patří sem získávání údajů pomocí trojských koní, technik souvisejících s dešifrováním komunikace, odposlechem v bezdrátové síti aj. Cracker využívá bezpečnostních děr operačního systému a programů pro podporu síťového spojení. Cílem útoku jsou zpravidla finanční instituce nebo jiné subjekty disponující informacemi vysoké ceny.

Čmuchání na síti

Nyní se zaměříme na některé formy crackingu. Zajímavou technikou získávání citlivých informací je sledování a analýza provozu v počítačové síti, tzv. **sniffing (čmuchání)**. Informace získané odchytáváním komunikace v síti jsou následně určeny k dalšímu zneužití. K zachytávání existuje velké množství softwarových i hardwarových prostředků. Oblíbeným prostředkem je síťová karta přepnutá pomocí speciálního programu (snifferu) do promiskuitního módu. Tímto způsobem lze získat veškerý obsah nešifrované komunikace, která prochází přes konkrétní uzel sítě. Spolehlivou a účinnou obranou proti sniffingu je používání šifrované komunikace všude tam, kde je to možné (viz kap. 5.3.2, str. 124).

Softwarový cracking

Při útocích hackerů a crackerů dochází k protiprávnímu zásahu do dat. Příkladem může být **softwarový cracking** s cílem odstranění ochrany programu a jeho následné neoprávněné užívání a šíření. Velmi častý je také vandalismus webu (*defacement*).

Útoky typu DoS

Velmi nebezpečné jsou některé druhy kybernetických útoků, při kterých dochází k závažnému narušení nebo přerušení fungování informačního systému, resp. počítače. Jedná se o tzv. **útoky typu DoS** (*Denial of Services*), které se snaží znepřístupnit službu, počítač nebo celou síť. Toho lze spolehlivě docílit zahlcením serveru velkým množstvím nesmyslných požadavků. Útok z jednoho místa lze řešit použitím firewallu, ale existuje též varianta distribuovaných útoků, na kterých se podílejí tisíce počítačů z celého světa (infikovaných virem nebo úmyslně). Jako příklady útoků typu DoS uveďme tyto:

- *mass mailing list* – zneužití e-mailové adresy pro registraci na mnoha pochybných stránkách, ze kterých začne chodit záplava mailů;
- *e-mail bombing* – zahlcení poštovní schránky automaticky generovanými maily, které může vést až ke zhroucení celého poštovního systému;
- *fork bomb* – forma lokálního útoku, kdy programy spouští donekonečna samy sebe, až vyčerpají systémové prostředky a shodí operační systém.

Lámání hesel

Nejrozšířenější autentizační[95] techniku současnosti představují především hesla a PIN (*Personal Identification Number*). Existují však nástroje, které se snaží tuto ochranu prolomit. Mezi základní techniky patří zejména **útok hrubou silou** (*brute force attack*), při kterém se útočník snaží odhalit heslo pomocí testování všech možných kombinací. Pokud tak činí pomocí předem připraveného seznamu, jedná se o **slovníkový útok** (*dictionary attack*). Tyto techniky používají oprávnění uživatelé (kteří zapomněli heslo), systémoví administrátoři (kteří kontrolují, zda si uživatelé nevolí nevhodná hesla), ale zejména hackeři k získání protiprávního přístupu nebo dešifrování cizích dat. Na Internetu je volně k dispozici celá řada různých „prolamovačů“.

[95] Na tomto místě připomeňme tři související pojmy, které se často pletou: *identifikace* (zjištění totožnosti, např. zadáním uživatelského jména), *autentizace* (ověření totožnosti, např. zadáním hesla) a *autorizace* (oprávnění k provedení operace, důsledek úspěšné identifikace a autentizace).

7.2.2 Malware

Malware

Pro škodlivý software nebo jiná data, jejichž účelem je uškodit počítači nebo jeho obsahu či je jinak zneužít, používáme termín **malware**[96]. Někdy bývá do této kategorie řazen i software, který umí infiltrovat počítač bez souhlasu jeho uživatele, mezi nejvýznamnější představitele však patří počítačové viry, červy a trojské koně. Označení by se nemělo používat pro software, který sice obsahuje chyby, ale byl napsán pro legitimní účely.

Počítačové viry

Původ virů a definice

Pojmem **počítačový virus** označujeme umělý útvar vytvořený člověkem, který se dokáže sám šířit bez vědomí uživatele a provádí nějakou (většinou škodlivou) činnost. První takové programy se tu a tam objevovaly již v 70. letech 20. století,[97] pojem však poprvé definoval až americký informatik Frederick B. Cohen (* 1956) v roce 1983: „Počítačový virus je počítačový program, který může infikovat jiný počítačový program takovým způsobem, že do něj zkopíruje své tělo, čímž se infikovaný program stává prostředkem pro další aktivaci viru.“

Důvody vzniku

Pokud se zamyslíme nad tím, proč vůbec počítačové viry vznikají, nalezneme hned několik důvodů. Vytvářejí je propuštění programátoři velkých softwarových firem, kteří se chtějí pomstít, ale také mladí programátoři, kteří si chtějí vyzkoušet své schopnosti a neuvědomují si celkový dopad své činnosti. Dokonce se můžeme setkat s názorem, že viry produkují přímo firmy vytvářející antivirové programy za účelem zvýšení prodeje svých výrobků. V každém případě jsou viry ideálním prostředkem pro diskreditaci platformy, která není schopna sama sebe uchránit před jejich šířením.

Jména virů

Viry bývají pojmenovány přímo autorem (např. AIDS, Brain, Alabama), podle místa svého odhalení (např. Durban, Suomi, Taiwan), podle prováděné činnosti (např. Flip, Ping-pong, Jo-jo), podle délky, o níž zvětšují infikované soubory (např. 405, 5120, 4096) nebo podle data aktivace (např. Friday 13th, December 24th, Advent).

První virus pro PC

První skutečný virus, který mohl nějak uškodit, vytvořili pro počítače IBM PC s operačním systémem MS-DOS v lednu roku 1986 bratři Basit a Amjad Farooq Alvi v Pákistánu a pojmenovali jej *Brain*. Údajně jej vytvořili proto, aby zabránili pirátství jejich vlastního softwaru. Od té doby se samozřejmě mnoho změnilo, společně s vývojem techniky se vyvíjely i viry. Objevily se viry skrývající se před antivirovými programy, modifikující svůj vlastní kód, napadající i jiné objekty než výkonné programy apod. Je téměř jisté, že s příchodem nových operačních systémů se objeví nové viry, které budou využívat jejich specifických vlastností.

Kategorie virů

Podle napadených oblastí (tedy podle cíle virové nákazy) můžeme počítačové viry rozdělit do několika kategorií:

- **Souborové viry** (*file viruses*) jsou přidávány ke spustitelným souborům. Principem napadení je taková změna původního programu, že při jeho spuštění se nejprve aktivuje

[96] Jedná se o složeninu slov *malicious* (záludný, zákeřný, lstivý, zlomyslný) a *software* a spíše než specifické vlastnosti takového programu popisuje záměr jeho autora. Lze se setkat i s pejorativním termínem *scumware* pocházejícím od slova *scum* (špína).

[97] Zmínku lze najít také ve sci-fi románu Davida Gerrolda z roku 1972 nazvaném *When HARLIE Was One*.

virus, který provede všechny akce potřebné k rozmnožování a případně i destrukci, teprve potom se spustí původní program, aniž by uživatel časovou ztrátu v řádech tisícin sekundy zpozoroval. Podle způsobu infekce můžeme souborové viry dále dělit na přepisující, parazitické a doprovodné.

- **Dokumentové viry** (*macro viruses*) napadají dokumenty v aplikacích kancelářských balíků podporujících makra (MS Word, MS Excel). Princip napadení spočívá v tom, že po načtení zavirovaného dokumentu příslušná aplikace interpretuje makra viru, která při různých doprovodných akcích zabezpečí napadení globální šablony. Zavirovaná makra šablony se potom vkládají do dalších otevíraných dokumentů, a tím je infikují. Historie makrovirů sahá k roku 1995, kdy se objevil první virus „WM/Concept.A“ určený pro Word 6. Jeho autor chtěl údajně pouze demonstrovat, že je něco takového vůbec možné – Concept obsahoval makro určené k nějaké obtěžující nebo destruktivní akci, ale to neobsahovalo žádný kód. V současnosti jsou makroviry stále velmi rozšířeny.
- **Zaváděcí viry** (*boot viruses*) útočí na zaváděcí sektor disku.[98] Virus se díky tomu dostává do operační paměti dříve než operační systém a zajistí si spuštění přímo při startu počítače. Poškození zaváděcího sektoru navíc může vést ke zničení veškerého přístupu na disk. Za své rozšíření vděčí infikovaným přenosným médiím zapomenutým v mechanice. Vlastní infekce spočívá nejčastěji v tom, že se virus zapíše do tabulky oblastí pevného disku a její původní obsah odklidí na nějaké jiné „bezpečné“ místo. Na požadavek čtení tabulky oblastí pevného disku nevrátí její skutečný obsah, ale původní obsah tak, jak vypadal před nákazou.
- **Multipartitní viry** (*multipartite viruses*) kombinují více mechanismů šíření, a mohou tak pracovat jako souborové, zaváděcí i dokumentové viry.

Šíření virů

Princip činnosti počítačového viru lze shrnout do tří mechanismů: šíření, obrana a útok. Základním předpokladem pro úspěch je bezesporu **šíření viru**, tedy infikování dalších programů a počítačů. K tomu je zapotřebí mít vhodné podmínky, což představuje zejména vhodné prostředí (počítač s operačním systémem, který virus zná a je schopen používat) a objekty, které dokáže virus napadnout a které jsou nějakým způsobem šířeny (libovolné spustitelné soubory s rozšířením `.exe`, `.com`, `.sys`, ale také soubory knihoven připojovaných ke spustitelným programům s rozšířením `.dll` a dokumenty MS Office).

Obranné strategie

Útočníci se také snaží vybavit své viry nějakou formou obrany proti antivirovým systémům. Mezi základní patří **pasivní obrana** spočívající v použití takových programových konstrukcí, které znesnadní analýzu viru a ochrání ho před některými technikami hledání. Kromě toho existuje také **aktivní obrana**, která je založena na ochraně vlastního kódu a pokusech o poškození či dokonce likvidaci antivirového systému. Některé viry (tzv. *stealth*) se při pokusu antivirového programu o detekci snaží předstírat, že na disku a v operační paměti je všechno v pořádku. U nejzlomyslnější formy sebeobrany viru, tzv. *pasti*, začínají

[98] V počítačích kompatibilních s IBM PC je zaváděcí sektor (*boot sector*) prvním sektorem pevného disku, diskety či jiného média. Obsahuje hlavní spouštěcí záznam (*Master Boot Record*), na kterém se nachází spustitelný program čtený ihned po zapnutí počítače. Tento záznam obsahuje zavaděč operačního systému, kterému BIOS předává při startu počítače řízení, dále tabulku rozdělení disku na logické oddíly (*partition table*) a číselný identifikátor disku. Jeho velikost je 512 bajtů.

problémy až po odhalení a následném odstranění viru.[99] Obecně lze říci, že každý virus usazený v zaváděcím sektoru může tuto past představovat, i když to autorem viru nebylo původně plánováno. Obnovení původního obsahu zaváděcího sektoru a likvidace viru může způsobit nesoulad mezi současným obsahem disku a tabulkou rozdělení, čímž se po obnově automaticky nenávratně ztratí přístup k některým diskovým oddílům.

Projevy viru

Vlastní **útok** spočívající v provádění jistých (zpravidla škodlivých) akcí na infikovaném počítači se může projevit mnoha způsoby:

- změnami v systémových oblastech nebo v souborech (u všech typů virů);
- výrazným úbytkem volné paměti (u rezidentních virů);
- vytvořením spustitelných souborů s rozšířením `.com` v těch adresářích, kde již existuje soubor stejného jména s rozšířením `.exe` (u doprovodných virů).

Podle těchto typických projevů lze detekovat a odstraňovat známé i neznámé viry. Kromě technických projevů mívají viry ve svém těle zakomponovány i další činnosti:

- *doprovodné efekty* s promyšleným načasováním, například manipulace s obsahem obrazovky či vypsání objevného sdělení;
- *obtěžování uživatele* například úpravou systémového času, „polykáním“ kláves či občasnou záměnou stisknuté klávesy za sousední[100];
- *krádeže* s využitím Internetu a přenášení dat například pomocí elektronické pošty;
- *destrukce dat* například vkládáním slov do náhodného místa v dokumentu.

Mýty o virech

Počítačovým virům bývá často přisuzována řada schopností, kterými ovšem nedisponují. Nejrozšířenější fámou je **poškozování technického vybavení počítače**. K tomu mohlo ojediněle dříve docházet například systematickým čtením z fyzicky opačných konců pevného disku, kdy soustava čtecích hlav nevydržela opakované mnohonásobné přejíždění celého rozsahu. Další poškození mohlo nastat četným opakováním náročných instrukcí procesoru, které vedly k teplotnímu kolapsu, nastavováním nevhodných synchronizačních sekvencí na monitoru a následným tepelným přetížením apod. Vše je realizovatelné jen za předpokladu, že technické vybavení je konstrukčně nedokonalé a je navrženo na horní mezi svých možností. Poškození však mohlo být způsobeno i běžnou činností, možná jen za delší čas. Současné počítačové viry hardware poškodit nemohou, mohou však smazat obsah paměti Flash BIOS u některých základních desek a tím znemožnit chod počítače.

Druhou nejrozšířenější fámou je **infekce při pouhém čtení dat**. Aby se virus mohl stát aktivním, musí procesor začít provádět jeho instrukce. K tomu může dojít pouze spuštěním napadeného souboru nebo pokusem o zavedení systému z infikovaného média. Pouhým čtením jakýchkoli dat (i virových) nelze počítač v žádném případě infikovat. To se tedy týká nejen prohlížení dat na disku či disketě, ale také dalších operací, které čtení využívají (tisk, kopírování souborů ze vzdálených systémů apod.). Viry rozhodně nenajdeme v textových souborech, protože se tyto soubory nedají spustit a navíc jsou přímo čitelné a kontrolovatelné člověkem (viz kap. 6.1, str. 140). Obvykle nemohou být viry ani ve všech datových

[99] Například multipartitní virus One_Half provádí průběžné šifrování obsahu disku. Pokud není jeho odstranění doprovázeno dekódováním disku, některé adresáře se zcela ztratí a u jiných bude obsah zčásti přepsán. Tím ale past ještě nekončí. V případě, že v této situaci použijeme program na opravu disku, data ztratíme úplně. Program určený pro řešení běžných problémů na disku najde místo poškozeného obsahu nesmysly, pokusí se je vzít vážně a na základě těchto dat provede opravu. Výsledkem je zcela nepoužitelná část disku.

[100] Vžsledkem je, že vztosre ovvyklá míea přejleoů a nicbetušícá užuvatel se vzteký.

souborech (obrázky, zvuky, videa apod.), ale mohou zde být sekvence využívající různých chyb a bezpečnostních děr ovládacího softwaru, které následně způsobí podivné chování a někdy i ztrátu dat.

Závěrem ještě dodejme, že pojem *virus* se často chybně používá v obecném smyslu pro všechny typy infiltrací. Počet nově objevených klasických virů od roku 1998 postupně klesá, pozornost útočníků se však stále více zaměřuje na ostatní typy škodlivého softwaru.

Červy

Pojmem **červy**[101] (*worms*) označujeme škodlivé programy, které se šíří prostřednictvím Internetu (nejčastěji e-mailu). Na rozdíl od běžných virů se nepřipojují k žádnému programu ani se na disku dále nešíří. Na hostitelském počítači parazitují v jediném exempláři, přičemž ke svému šíření využívají jeho komunikačního propojení s dalšími počítači.

První počítačové červy na konci 80. let 20. století byly užitečné programy sloužící k monitorování počítačů v síti a optimálně rozdělující zátěž jednotlivých strojů. Později se vlivem chybně naprogramovaného červu ukázala velká hrozba, kdy došlo k nekontrolovatelnému šíření a k zahlcení sítě. Jedním z nejznámějších červů je I Love You, který 4. května 2000 během několika hodin zaplavil Internet, způsobil škody za několik desítek milionů amerických dolarů a ochromil provoz některých internetových služeb.

Projevy červů

Červů existuje mnoho druhů a jejich škodlivost spočívá ve dvou fázích: mohou zahltit síť svým šířením, ale také mohou po napadení systému provádět škodlivé akce podobně jako ostatní viry. Mezi typické projevy patří problémy s fungováním počítače nebo jeho součásti, mazání souborů, šifrování souborů jako nátlak k zaplacení poplatku, prohledávání počítače za účelem získání osobních dat nebo vytváření tzv. „zadních vrátek“ (*backdoor*), která poté mohou být využita jako přímá cesta k infikování počítače dalšími nákazami. Některé červy se aktivují i bez vůle uživatele, využívají při tom bezpečnostních děr poštovních klientů.

Když si počítačová komunita uvědomila svou zranitelnost, podnikla preventivní kroky ke zvýšení zabezpečení počítačů před podobnými útoky. I přesto ale počet nově objevených červů v současné době zůstává stabilní.

Trojské koně

Trojské koně (*Trojan horses*) často řadíme mezi typické představitele malwaru, na rozdíl od virů však není tento typ škodlivého kódu schopen sebereplikace a infekce souborů. Trojský kůň nejčastěji vystupuje jako spustitelný soubor, který neobsahuje nic jiného než škodlivý kód. Navenek se tváří jako neškodný nebo dokonce přitažlivý program, po spuštění však vypustí svůj náklad (obvykle virus nebo spyware). Typickým příkladem je program pojmenovaný `waterfalls.scr`, který se tváří jako volně šiřitelný spořič obrazovky. Kromě zobrazování vodopádu však otevírá porty počítače, čímž poskytuje vzdálený přístup hackerům. Od počítačového viru nebo červu se trojský kůň liší tím, že se škodlivý kód dále nereplikuje. Účinnou formou dezinfekce je proto odstranění dotyčného souboru.

[101] Termín *červ* byl převzat z románu Johna Brunnera *Jezdec na rázové vlně* z roku 1975, který pojednává o programu schopném samostatného šíření, který měl na pokyn autora vyřadit z provozu celou telefonní síť.

Destruktivní trojani

Dříve byly trojské koně vytvářeny jako jednoduché programy předstírající nějakou užitečnou činnost, které místo toho smažou soubory, přepíšou konfiguraci počítače uloženou v paměti CMOS[102] nebo provedou jinou destruktivní akci. V minulosti se několikrát objevil trojský kůň vydávající se dokonce za antivirový program McAfee VirusScan, ve skutečnosti však likvidující soubory na pevném disku. Jako jeden z posledních dodržujících tuto tradici byl trojan Telefoon, který se vydával za kompresní program RAR.

V systému Windows může trojský kůň využít toho, že řada programů (včetně systémového správce souborů) skrývá přípony souborů. Vypadá pak jako soubor s obrázkem, zvukem, archivem nebo čímkoliv jiným, přestože se ve skutečnosti jedná o spustitelný program. Chce-li uživatel obrázek kliknutím zobrazit, je ve skutečnosti spuštěn škodlivý kód.

Klasifikace trojských koní

Současné trojské koně mívají mnoho funkcí, podle kterých bývají také pojmenovány. Můžeme je klasifikovat takto:

- *sniffer* – odposlouchování přístupových jmen a hesel, čísel kreditních karet apod.;
- *keylogger* – sledování stisknutých kláves pro možnost odposlechu hesel;
- *spyware* – sledování uživatele a jeho zvyklostí při surfování na Internetu;
- *backdoor* – síťová služba, kterou může útočník použít pro získání přístupu do systému přes počítačovou síť;
- *spam server* – rozesílání nevyžádané elektronické pošty z napadeného počítače;
- *souborový server* – ukládání nelegálních dat do počítače a jejích sdílení na Internetu;
- *security disabler* – zablokování bezpečnostního softwaru počítače;
- *denial of service attacker* – vysílání nesmyslných požadavků na konkrétní počítač, který je tím vyřazen z činnosti;
- *dialer* – přesměrování infikovaného počítače připojeného přes vytáčené připojení k síti Internet na dražší tarify.

Pravděpodobnost infiltrace trojským koněm se zvýšila od chvíle, kdy si řada počítačových virů začala nést trojské koně s sebou a během svého šíření je začala vypouštět na jednotlivé počítače. Pokud je virus úspěšný, dojde jak k masivnímu rozšíření viru, tak i k vypuštění velkého množství trojských koní. Potom už záleží jen na autorovi daného viru, jak dokáže získané možnosti využít. Počet nově objevených trojských koní je od roku 2003 setrvale vysoký.

Rootkity

Nepříjemným a relativně novým malwarem je tzv. **rootkit**. Jedná se o program, který dokáže zamaskovat svou přítomnost v napadeném počítači tím, že změní funkce jádra operačního systému, aby nebyl vidět při výpisu souborů, běžících procesů, síťových připojení ani při prohledávání registrů. Pro jej lze detekovat pouze před instalací do jádra operačního systému.

[102] CMOS (*Complementary Metal-Oxide-Semiconductor*) je paměť, v níž je uložena informace o konfiguraci počítače. Je neustále napájena z baterie, a není tedy závislá na připojení počítače k elektrické síti. Technologie, která se využívá k výrobě těchto pamětí, se vyznačuje velmi malou spotřebou energie, baterie tedy dokáže udržet správné informace relativně dlouhou dobu.

Skandál hudebního vydavatelství

Známým případem použití rootkitu byl DRM software firmy Sony BMG v roce 2005. Společnost se pokoušela zabránit nelegálnímu kopírování hudebních CD tím, že na ně umístila rootkit, který se automaticky nainstaloval na počítač, do něhož bylo CD vloženo. Rootkit zasahoval do operačního systému a chránil obsah CD před porušováním autorských práv. Měl nepříjemný vedlejší efekt – vytvořil v systému slabiny, které mohly být zneužity hackery. Po skandálu musela být „infikovaná“ CD s ostudou stažena z trhu.

Cookies

V domnělé anonymitě si uživatelé na webu troufají dělat věci, jakých by se ve skutečném životě těžko odvážili. Jedná se například o prohlížení erotických nabídek, návštěvu herny či crackerské stránky nebo stahování chráněných skladeb či filmů. Jejich stopy však lze sledovat poměrně snadno – jsou často „značkováni“ pomocí tzv. **cookies**[103].

Zneužití cookies

Pro počítač nejsou cookies nijak nebezpečné, bezpečnostní riziko je však spojeno s jejich zneužitím. Pokud se crackerovi podaří získat přístup k cizím cookies, může se vydávat za cizího člověka a využívat jeho účet.

7.2.3 Grayware

Aplikace nebo soubory, které nejsou přímo klasifikovány jako *malware*, ale stále negativně ovlivňují výkon počítače a přinášejí významná bezpečnostní rizika, označujeme jako „šedý“ software – **grayware**. Zejména jde o aplikace, které se chovají nepříjemným nebo nežádoucím způsobem, a přesto jsou méně závažné nebo znepokojující než malware.

Grayware zahrnuje spyware, adware, dialery, žertovné programy, vzdálený přístup k nástrojům a jakýkoliv jiný program odlišný od viru navrženého tak, aby poškodil nebo zneužil výkon počítačů. Tento termín je používán přibližně od roku 2004.

Spyware

Obecně lze **spyware** definovat jako nechtěný software, který se instaluje bez vědomí uživatele a nějakým způsobem jej poškozuje – ovlivňuje soukromí uživatele nebo bezpečnost systému, používá systémových zdrojů včetně již instalovaných programů, sbírá a odesílá citlivá data apod. Provádí činnost, která neslouží uživateli, ale tvůrcům jeho programového kódu. Spyware je tedy možné chápat jako určitou odrůdu počítačového viru, jehož cílem není dále se množit, ale pouze provádět nějakou nevítanou činnost v napadeném počítači. Alarmující je skutečnost, že roku 2002 počet nových spywarů strmě narůstá.

Projevy spywaru

Mezi nejnebezpečnější činnosti, které spyware provádí, patří bezesporu shromažďování citlivých dat a veškeré snahy o získání přístupu k bankovním účtům nebo dalším cenným údajům a zdrojům. Mezi nejčastější příznaky výskytu spywaru patří nežádoucí domovská stránka v prohlížeči, pomalý start počítače spojený se zdlouhavým připojováním

[103] Jedná se o malé množství dat, která server pošle prohlížeči a ten je uloží na počítači uživatele. Při každé další návštěvě téhož serveru pak prohlížeč tato data posílá zpět serveru. Technika slouží k rozlišování jednotlivých uživatelů, ukládání předvoleb apod. Označení se do češtiny překládá jako *sušenka* a asociuje zvyklost z USA či Velké Británie nabídnout účastníkům zájmového spolku oblíbenou sušenku pro vytvoření příjemnější atmosféry.

k síti Internet, ve zvýšené míře vyskakující reklamy při surfování, přesměrování telefonní linky u vytáčeného spojení, padání Windows častěji než obvykle, záhadně se objevující nové ikony na ploše aj.

Kategorie spywaru

Spyware lze rozdělit do celé řady kategorií, z nichž některé získaly samostatné pojmenování.[104] Mezi nejznámější patří *keylogger* sledující každé zmáčknutí klávesy na klávesnici a odesílající zaznamenané stisky třetí osobě a *RAT (Remote Administration Tool)*, který umožní neoprávněnému uživateli vzdáleně ovládat napadený počítač.

Možnosti instalace spywaru

Existuje mnoho způsobů, jak se může spyware dostat do počítače. Častým způsobem je instalace spywaru během instalace jiného požadovaného softwaru (zejména volně dostupných programů pro přehrávání, zpracování nebo sdílení multimediálních či jiných souborů). Nejčastější cestou je však webový prohlížeč či elektronická pošta, neboť moderní internetové stránky jsou tvořeny pomocí plnohodnotných programovacích jazyků, díky nimž mají uživatelé Internetu k dispozici celou řadu pokročilých aplikací. Komplexnost těchto nástrojů s sebou bohužel nese také možnost jejich zneužití ze strany autorů spywaru. Zcela „spolehlivým“ zdrojem spywaru je pak prohlížení zjevně rizikových webových stránek (pornografie, warez apod.). Nástroje snažící se skrytě nainstalovat na počítač spyware však mohou být zakomponovány prakticky do libovolného webu.

Adware

Adware (*advertising-supported software*) bývá někdy instalován se souhlasem uživatele jako součást softwaru, u něhož není třeba platit licenční poplatky (typicky freeware, viz str. 177). Uživatel je poté obtěžován reklamami s různou úrovní agresivity – od běžných bannerů až po neustále vyskakující (tzv. pop-up) okna nebo ikony v oznamovací oblasti.

Většina adwaru spadá do šedé zóny mezi neškodným softwarem a spywarem. Často obsahuje *hijacker*, který mění startovací stránku webového prohlížeče nebo přidává oblíbené položky. Na rozdíl od spywaru ale adware neshromažďuje citlivá data bez souhlasu uživatele.

Dialer

Připojení do sítě Internet se dříve realizovalo pomocí telefonní linky a modemu. Pro navázání spojení je potřebné se telefonicky spojit s modemem nejbližšího serveru, který požadované připojení do sítě provede. Telefonní poplatky jsou účtovány za místní spojení.

Dialer[105] je škodlivý program, který mění způsob přístupu k síti Internet tím, že přesměruje místní volání nejbližšího modemu na linky s vysokými sazbami. Uživatel pak za minutu spojení platí řádově desítky až stovky korun. S rostoucím podílem pevných připojení k počítačovým sítím tato forma škodlivého programu pomalu ustupuje.

[104] Stejné nebo velmi podobné kategorie lze najít i u jiných forem škodlivého softwaru (červů, trojských koní apod.). Jde prakticky o tytéž programy, rozdíl je pouze ve způsobu jejich šíření a zdroji infekce.
[105] Termín pochází z anglického *dial* (vytáčet číslo na telefonním přístroji).

7.2.4 Obrana proti různým formám infiltrace

Funkce antivirů

V současnosti jsou známy desetitisíce životaschopných počítačových virů. K jejich detekci, odstranění a prevenci proti případné nákaze slouží **antivirový software**, zkráceně antivir.[106] Mezi jeho základní funkce patří zejména:

- kontrola každého otvíraného souboru;
- kontrola spouštěných programů;
- pravidelná kontrola všech souborů na disku;
- kontrola e-mailů;
- kontrola webových stránek.

Detekce malwaru

Počítačové viry a jiný škodlivý software lze vyhledávat několika různými způsoby. Za zmínku stojí zejména tyto metody detekce:[107]

- **Vyhledávání signatur** – nejčastější a nejpoužívanější metoda. Je založena na předpokladu, že virus v sobě nese určitý jedinečný řetězec bajtů, tzv. signaturu, díky kterému jej lze snadno identifikovat. Vyskytuje-li se takový řetězec v aktivním souboru, obsahuje tento soubor i virus. Vývoj této detekční metody šel kupředu, takže si dnes dokáže poradit i s mutujícími viry, které se mohou měnit a modifikovat. Seznam všech známých signatur je uložen v databázi, kterou antivirový program musí neustále aktualizovat.
- **Heuristická analýza** – rozvinutější metoda, která dokáže najít i virus, pro který dosud nebyly vytvořeny signatury. V případě mohutného virového napadení by měla být tato procedura dostatečně účinná. Základem vyhledávání je předpoklad, že podezřelé projevy programů mají na svědomí počítačové viry. Antivirový program zjišťuje tyto podezřelé projevy postupným prováděním jednotlivých instrukcí testovaného programu. Toto testování je ale bohužel velmi náročné a pomalé, občas může dojít i k planému poplachu. Heuristická analýza také nedokáže odstranit nově nalezený virus.
- **Test kontrolního součtu** – metoda, která preventivně počítá kontrolní součet (viz též kap. 5.3.1, str. 113) pro všechny ohrožené soubory na disku. Při každém použití takového souboru je opět spočítán kontrolní součet, který je následně porovnán s předchozím výsledkem. Pokud se údaje liší, je vyhlášen poplach. Tato metoda je použitelná pouze v případech, kdy se obsah souborů během zpracování nemění. Metodu tedy nelze aplikovat v případě, kdy si například spustitelný soubor do svého těla ukládá různé uživatelské konfigurace.
- **Antivirové štíty a scannery** – programové moduly antivirového systému, které testují všechny otevírané soubory na přítomnost virů a navíc analyzují chování spuštěných programů. Je-li zjištěna podezřelá činnost, štít činnost zastaví a vyzve uživatele k potvrzení dalšího kroku. Bohužel značná část uživatelů bez rozmyslu krok potvrdí, což samozřejmě nahrává virům.

[106] Ačkoli to není na první pohled patrné, moderní antiviry slouží i k eliminaci červů a trojských koní. I to může být důvodem, proč se v laické terminologii často nerozlišuje mezi viry a jiným malwarem. Stále častěji se ale můžeme setkat s označením *anti-malware*.

[107] Žádná metoda však není zcela spolehlivá. Americký informatik Frederick B. Cohen v roce 1987 ukázal, že neexistuje algoritmus, který by spolehlivě detekoval všechny možné viry.

- **Data mining** – nejnovější přístup v detekci malwaru. Algoritmy dolování dat a strojového učení se snaží vyhodnotit potenciální škodlivost souborů pomocí podrobné analýzy jejich typických rysů.

Úspěšnost těchto metod závisí na schopnostech antivirového programu a aktuálnosti databáze počítačových virů, kterou si programy pravidelně stahují z Internetu. V současné době antivirové programy používají více metod vyhledávání, například kombinace vyhledávání signatur a heuristické analýzy. Vhodná je také možnost odeslat podezřelý zavirovaný soubor k analýze přímo firmě, která antivirový program vyrobila. Zajímavé je, že v poslední době odborníci začínají pochybovat o užitečnosti a efektivnosti antivirových programů, neboť bylo zjištěno, že například účinnost detekce malwaru v e-mailech je relativně nízká.

Operace s infikovanými soubory

V okamžiku, kdy antivirový program detekuje virus, uživatel musí vybrat další postup. Obvykle existují tři možnosti dalšího postupu:

1. pokusit se *opravit* (vyléčit) soubory odstraněním viru, je-li to technicky možné;
2. umístit soubory do *karantény*, kdy je nelze používat a je tak zamezeno šíření viru;
3. *smazat* infikované soubory i s virem.[108]

Ochrana před malwarem

První veřejně známé neutralizování rozšířeného viru bylo provedeno v roce 1987. Na podzim roku 1988 byl vytvořen první antivirový program Solomon's Anti-Virus Toolkit, který znamenal předzvěst dalších specializovaných systémů na ochranu počítačů a vzniku firem zabývajících se tímto specifickým odvětvím. Už v roce 1990 byly na trhu dvě desítky takových produktů. V dnešní době máme na výběr z celé řady antivirových systémů, z nichž stojí za povšimnutí zejména tyto:

- *Avast! (AntiVirus Advanced SeT)* – produkt české firmy AVAST Software, která je následníkem společenství vlastníků ALWIL Software. Na trhu je od roku 1988, stal se jedním z nejpopulárnějších antivirů. Pro domácí použití je freeware, ale směle může konkurovat i placeným produktům.
- *AVG (AntiVirus Guard)* – produkt české firmy AVG Technologies (dříve Grisoft), který je určen pro všechny operační systémy. Na trhu je od roku 1992 a v současnosti je pro domácí a nekomerční použití k dispozici zdarma. Pravidelně sklízí úspěchy v nezávislých testech a je držitelem významných ocenění.
- *BitDefender* – kvalitní produkt rumunské společnosti Softwin, na trhu od roku 1996.
- *Dr. Web* – první ruský antivirový program, na trhu od roku 1992. Dokáže detekovat a odstranit červy u e-mailů, internetové viry, trojské koně, polymorfní viry, makroviry, rootkity, spyware apod.
- *ESET NOD32 Antivirus* – slovenský komerční produkt, který byl několikrát oceněn jako nejlepší antivirový program. Na trhu je od roku 1992.
- *F-Secure Anti-Virus* – finský produkt stejnojmenné firmy, na trhu od roku 1988.
- *KAV (Kaspersky Anti-Virus)* – produkt ruské firmy Kaspersky Lab. U nás není příliš rozšířen, ale kvalitou se může srovnávat s nejlepšími aplikacemi tohoto typu.
- *McAfee VirusScan* – produkt společnosti Intel, nabízí řešení pro domácí uživatele i korporátní prostředí. Je součástí komplexního bezpečnostního balíčku.

[108] Ani to ale často nemusí stačit. Je-li například nalezen zaváděcí virus, někdy nepomůže ani formátování pevného disku, protože virus může být zapsán také v zaváděcí tabulce.

- *MSE (Microsoft Security Essentials)* – bezplatný antivirus pro menší podniky, který slouží také jako obrana proti spywaru, rootkitům a trojským koním.
- *Norton AntiVirus* – produkt americké firmy Symantec pro domácí použití.

Ochrana před spywarem

Obrana proti spywaru je záležitostí obezřetnosti každého uživatele, který by si nejdříve měl pečlivě pročíst veškerou dokumentaci včetně licenční smlouvy a prohlášení o zásadách ochrany osobních údajů. Někdy je v nich zahrnutí nežádoucího softwaru do dané instalace dokonce přímo uvedeno. Většina antivirových programů spyware nenajde, je proto nutné použít speciální software – **antispyware**. Doporučuje se používat více těchto programů současně, neboť mohou mít různé databáze vzorků spyware. Mezi nejznámější zástupce této kategorie[109] softwarových produktů patří:

- *AVG Anti-Spyware* – dostupný zdarma i pro komerční účely;
- *Spy Emergency* – slovenský komerční software;
- *Spy Sweeper* – komerční software americké společnosti Webroot;
- *Spyware Doctor* – komerční produkt, podle většiny testů jeden z nejlepších;
- *Spyware Terminator* – dostupný zdarma i pro komerční účely.

Ochrana před adwarem

Na ochranu před adwarem také existuje několik programů, z nichž zřejmě nejznámějším je Ad-Aware od německé společnosti Lavasoft. Tento produkt dokáže detekovat a odstranit také spyware, počítačové viry, dialery, trojské koně, rootkity a mnoho dalších méně známých forem škodlivého softwaru.

Preventivní metody

Detekci a odstraňování virů nebo obecně škodlivého softwaru, který se již dostal na uživatelský počítač, lze charakterizovat jako hašení doutnajícího nebo hořícího požáru. Mnohem bezpečnější je, pokud se na uživatelský počítač škodlivý software vůbec nedostane. Nejpodstatnějším prvkem prevence je vhodné chování uživatele a také správné nastavení některých programů. Mezi osvědčené preventivní metody patří:

- používat legální a pravidelně aktualizované programové vybavení;
- používat antivirové programy, antispyware, firewall[110];
- pravidelně aktualizovat zejména síťové produkty (komunikátory, ovladače apod.);
- používat otevřené souborové formáty;
- zakázat makra u neznámých dokumentů MS Office;
- nevěřit všem zprávám elektronické pošty bez ohledu na adresu odesílatele;
- nespouštět podezřelé programy z Internetu;
- pravidelně zálohovat důležitá data;
- zcela obecně používat zdravý rozum.

Unix, macOS a škodlivý software

Na závěr ještě dodejme, že operační systémy typu Unix (a tedy i macOS) jsou vůči škodlivým kódům méně náchylné, a proto pro svou vlastní potřebu antivirové programy prakticky nepotřebují. Výjimkou jsou servery, které tyto programy používají zejména na ochranu svých klientů. Škodlivé kódy, zejména rootkity, pro tyto operační systémy existují také, ovšem běžný uživatel se s nimi prakticky nesetká a když už, většinou se jedná o kódy dávno zneškodněné. Také uživatelé chytrých telefonů a tabletů s operačním systémem na bázi Unixu (např. Android) mohou být poměrně klidní.

[109] Asi nikoho v tuto chvíli nepřekvapí, že většina programů pro boj se spywarem si dokáže současně poradit i s adwarem stejně dobře jako s různými formami malwaru.

[110] Termínem *firewall* označujeme síťové zařízení, které slouží k řízení a zabezpečování síťového provozu mezi sítěmi s různou úrovní důvěryhodnosti a zabezpečení.

Kontrolní otázky

12. Jaký je rozdíl mezi hackingem a crackingem?
13. Na jakém principu je založen sniffing?
14. Které metody se používají k prolamování hesel?
15. Co je to malware a jaké má formy?
16. Jak pracuje počítačový virus a do kterých kategorií se počítačové viry dělí?
17. Co mají společného a čím se liší počítačový virus a počítačový červ?
18. Na jakém principu je založen trojský kůň?
19. Které činnosti provádí spyware?
20. K čemu slouží antivirový software a které funkce obvykle nabízí?
21. Jaké jsou možnosti prevence proti škodlivému softwaru?

7.3 Techniky sociálního inženýrství

Sociální inženýrství

Pojmem **sociální inženýrství** (*social engineering*) označujeme způsob manipulace lidí za účelem provedení určité akce nebo získání určité informace. Termín je běžně používán ve významu podvodu nebo podvodného jednání za účelem získání utajených informací organizace nebo přístup do informačního systému firmy. Proces využívá slabin a důvěry podváděných bez omezení druhu komunikačního kanálu (osobně, telefonicky, on-line) nebo mezinárodních hranic (Internet tuto bariéru zrušil zcela). Ve většině případů útočník nepřichází do osobního kontaktu s obětí.

V této podkapitole se budeme zabývat vybranými technikami sociálního inženýrství, které bývají z pohledu legislativy označovány jako trestné činy se vztahem k počítači.

7.3.1 Spoofing

Formy spoofingu

Pod označením **spoofing** se skrývají různé útočné techniky, jejichž společným rysem je podvržení údajů. Kromě „neškodných“ kanadských žertíků stojí za zmínku následující formy:

- *IP spoofing* – tzv. útok typu „man-in-the-middle“, kdy útočník kontroluje komunikační kanál a může zadržovat pakety nebo je nahrazovat jinými;
- *DNS spoofing* – podvržení webové stránky (např. formuláře) pomocí útoku na službu překládající doménová jména na adresy IP;
- *e-mail spoofing* – podvržení e-mailové adresy, oblíbená technika spammerů;
- *GPS spoofing* – vyslání falešného signálu GPS, na základě kterého přijímací zařízení chybně vyhodnotí svou pozici.

7.3.2 Phishing

Phishing

Phishing[111] spočívá v získávání důvěrných a citlivých informací (hesla, detaily o kreditních kartách apod.) za účelem získání přístupu a následného obohacení se z cizích zdrojů. Nejčastější a nejznámější formou je zasílání podvodných e-mailových zpráv. V phishingovém e-mailu podvodník upozorňuje, že adresátova banka (nebo jiná instituce poskytující finanční služby on-line) potřebuje ověřit určité údaje a za tímto účelem žádá adresáta o přihlášení. Dále následuje odkaz na podvrženou stránku, kde adresát vyplní požadované údaje (nejčastěji přihlašovací údaje nebo PIN a číslo karty). Po získání přístupu následuje zneužití nebo vykradení účtu oběti.

Nebezpečí této techniky spočívá především v její jednoduchosti, neboť útočník nemusí mít prakticky žádné technické znalosti. Nejefektivnější obranou proti phishingu je použití zdravého rozumu – nerozdávat důvěrné informace automaticky a bez rozmyslu na kdejakou výzvu a v případě pochybností ověřit cílovou adresu IP.

7.3.3 Pharming

Pharming

Daleko nebezpečnější způsob útoku představuje **pharming**[112], kdy útočník nejprve hackuje DNS (což je server odpovědný za přiřazování doménových jmen adresám IP) tak, aby odkazoval na podvržené stránky. Oběť tak v podstatě nemá o podvrhu žádné podezření a ani zkušení uživatelé nemusejí poznat rozdíl. Pharming má následně stejný průběh jako phishing, od nějž se liší právě pouze nutností podvržení DNS. Proti pharmingu je bezmocný antivirový i antispywarový software.

Kontrolní otázky

22. Čím se zabývá sociální inženýrství?

23. Co je to spoofing a jaké může mít formy?

24. Jaký je rozdíl mezi phishingem a pharmingem?

7.4 Kyberstalking a kyberšikana

V psychiatrii či kriminologii je pod pojmem *stalking* chápáno úmyslné pronásledování a obtěžování jiné osoby, které snižuje kvalitu jejího života a ohrožuje její bezpečnost.

Kyberstalking

Kyberstalking (*cyberstalking*) je forma stalkingu využívající kyberprostoru a „přítomnosti“ oběti v něm. Jde zejména o zasílání různých zpráv pomocí instant messengerů (ICQ), chatu, technologií VoIP apod. Nejčastějšími obětmi jsou celebrity, bývalí partneři, politici, zhrzení v lásce apod. Útočník může také využívat prostředky IT (např. keyloggery) k získávání a shromažďování množství osobních informací o oběti.

[111] Tato technika se do češtiny překládá jako *rhybaření* a její označení vzniklo spojením anglických slov *fishing* (rybaření) a *phreaking* (nabourávání telefonních linek).
[112] Do češtiny někdy překládáno jako *farmaření*, označení vznilo spojením slov *farming* a *phishing*.

Kyberšikana

Kyberšikana (*cyberbullying*) využívá elektronické prostředky (mobilní telefony, e-maily, pagery, Internet, blogy apod.). Její typické projevy představují zasílání obtěžujících, urážejících či útočných e-mailů a SMS, vytváření stránek a blogů dehonestujících ostatní, popřípadě může sloužit k posilování klasických forem šikany, nejčastěji prostřednictvím nahrání scény na mobilní telefon a jejího následného rozeslání známým dotyčného nebo vystavení na Internetu. Zejména poslední varianta může být extrémně nebezpečná.

O kyberšikaně se v poslední době mluví zejména v souvislosti s událostmi na českých školách, kdy násilných videí z českých škol je možné najít stovky, možná i tisíce. Nezanedbatelné jsou také důsledky takového počínání. Šikanované oběti se neumí bránit a bojí se zkazit „legraci“. Je znám i případ, že dítě, které záznam z mobilního telefonu poskytlo učitelům, se samo stalo cílem výhrůžek některých spolužáků. Bohužel jsou už evidovány případy s tragickým koncem.

Kontrolní otázky

25. Co je to kyberstalking?

26. Které typické projevy má kyberšikana?

7.5 Šíření nevyžádaných a nepravdivých informací

7.5.1 Spam

Velký problém v oblasti elektronické komunikace v současnosti představují nevyžádaná sdělení všeho druhu (nejčastěji však reklamního charakteru), která se masově šíří Internetem i dalšími komunikačními kanály. Pro tento fenomén používáme označení **spam**.[113]

Etymologie slova spam

Název pochází od značky amerických vepřových konzerv, které se vyrábějí od 30. let minulého století dodnes, a vznikl jako zkratka ze sousloví „kořeněná šunka“ (*spiced ham*). Do souvislosti s nevyžádanou poštou se tato pochutina dostala vlivem populárního britského satirického seriálu Monty Pythonův létající cirkus. V závěrečném skeči 25. epizody se objevuje restaurace, kde všechny položky jídelního lístku obsahují spam, některé i opakovaně. Spam však reálně existuje mnohem déle, první zmínka pochází již z 19. století.[114]

Dnešní spam má mnoho podob (obchodní nabídky, nahotinky apod.) a odhaduje se, že tvoří asi 75 % všech e-mailových zpráv. Existuje také mimo svět počítačů – vzpomeňme si na tuny reklamních letáků ve schránce nebo otravné telefonáty s „výhodnými“ nabídkami pochybných společností.

Roboti

Adresy příjemců jsou do spamových databází získávány často pomocí tzv. robotů. Tyto speciální programy systematicky prohledávají webové stránky a zaznamenávají vše, co připomíná e-mailovou adresu, například řadu číslic či písmen, která obsahuje zavináč.

[113] Lze se setkat i s označením UBE (*Unsolicited Bulk Email*) nebo UCE (*Unsolicited Commercial Email*).

[114] Tehdy povolila telegrafická společnost Western Union zasílání zpráv do více destinací. V roce 1864 byl zaznamenán první hromadně odeslaný nežádoucí telegram. Následkem toho byli bohatí obyvatelé Severní Ameriky až do období velké hospodářské krize zahlcováni nejasnými investičními nabídkami. V Evropě tento problém neexistoval v takovém měřítku, protože telegrafii zde regulovaly národní poštovní kanceláře.

Ochrana mailových adres

Základní obrana tedy spočívá v nezveřejňování e-mailové adresy na webových stránkách. To se lehko řekne, ale realita bývá poněkud horší. Pokud je webová stránka určena pro prezentaci a firma očekává, že na základě ní ji budou kontaktovat zákazníci, pak adresu zveřejnit musí. Používá se k tomu ovšem důmyslný trik: adresa je uvedena v „nečitelném“ tvaru tak, aby ji uživatel bez potíží pochopil, ale aby to byl pro vyhledávací program problém. „Bezpečný“ tvar adresy by mohl být například `jméno(zavináč)firma.cz` nebo `jméno(at)firma(dot)cz`, často se zavináč nebo i kompletní adresa vkládají v podobě rastrového obrázku.

Vyhledávání adres může být také prováděno pomocí malwaru přímo v počítači. Je proto vhodné mít kvalitní antivirový a antispamový software. Při obdržení nevyžádaného e-mailu se silně nedoporučuje otvírat přílohy a klikat na nabízené odkazy. Tím se dává najevo, že adresa je funkční a obsah zprávy je pro příjemce atraktivní.

Program vyhledávající adresy je také může získat sledováním odpovědí ze vzdálených poštovních serverů, na které provádí slovníkový útok (viz str. 180), při kterém se pokouší doručit e-mail na adresy složené z obvyklých jmen a příjmení, oblíbených slov a přezdívek. Tyto adresy jsou ve větším ohrožení, proto se doporučuje jejich rozšíření o další znaky.[115]

Pokud bychom se podívali na další metody, které nás mají ochránit před spamem, v první řadě musíme zmínit restriktivní nastavení poštovního serveru, testování pošty na přítomnost spamu a jeho následné filtrování. Příslušný software lze dnes pořídit zdarma, kvalitní provozovatelé poštovních serverů již mají **antispam** implementován. Jeho principy se liší podle typu infiltrace.

Filtrace podle způsobu dopravy

Při *filtraci podle způsobu dopravy* je příchozí pošta rozlišována podle adresy IP. Využívá ji většina poskytovatelů, i když není 100% účinná. Existují dva principy:

1. **Blacklisting** – adresy, ze kterých již bylo zaznamenáno rozesílání spamu, jsou zapsány na tzv. černé listině (*blacklist*), která je zveřejněna nejčastěji pomocí systému DNS. Tím je předem rozhodnuto, který e-mail je spam, a to na základě analýzy adresy odesílatele nebo adresy IP, ze které dopis přišel. Fakt, že adresa je v blacklistu, má za následek buď přímé odmítnutí e-mailu ještě během poštovní relace, nebo je informace z blacklistu použita při následné filtraci podle obsahu.[116]
2. **Graylisting** – adresy odesílatele a příjemce jsou využívány dynamicky. Poštovní server udržuje databázi, kde je pro každou trojici „adresa IP, odesílatel, příjemce“ uvedeno, zda dopis s těmito atributy má být převzat k dopravě, nebo zda jeho převzetí má být dočasně odmítnuto. První dopis je odmítnut a je zaznamenán čas, kdy k tomu došlo. Následně jsou dopisy se stejnými atributy odmítány. Pokud se původní poštovní server stále pokouší o odeslání, je potvrzen záznam v databázi a dopisy jsou po určitou dobu přijímány a odesílány bez zdržení. V opačném případě je záznam smazán. Roboti se totiž snaží doručit zprávy co nejrychleji, chyby neošetřují a nepokoušejí se o opakované odeslání. Tento způsob se většinou užívá před filtrováním podle obsahu

[115] To je jeden z důvodů, proč v UIS začínají uživatelská jména všech studentů znakem „x“, externí spolupracovníci mají přidělena uživatelská jména začínající „qq“.

[116] Nemusí ale platit, že každá pošta z adresy v blacklistu je spam. Může se stát, že někteří uživatelé potřebují z takového prostoru poštu doručovat. Některé filtrační programy proto mají i tzv. bílou listinu (*whitelist*), na kterou jsou zařazeni odesílatelé, jejichž poštu chce uživatel dostávat. Zprávy z této adresy pak nejsou blokovány.

a zvyšuje tak jeho účinnost. Nevýhodou je občasné zdržení dopisu a případné přijetí dopisů v jiném pořadí, než v jakém byly odeslány. Některé poštovní servery mohou také být nedostatečně vybaveny a neimplementují frontu dopisů k odeslání.

Filtrace podle obsahu dopisu

Při *filtraci podle obsahu dopisu* je příchozí pošta analyzována na přítomnost spamu pomocí dvou typů filtrů:

1. **Filtry založené na pravidlech** – vyhledávají v dopisech typické rysy spamu. Mohou to být některá slova (viagra, enlarge apod.), slovní spojení nebo chyby typické pro spam (datum odeslání v budoucnosti, nedovolené znaky v hlavičce apod.). Za každý rozpoznaný rys je uděleno bodové ohodnocení, body se zpravidla sčítají a pokud součet přesáhne určitou hranici, je dopis označen jako spam. Rozpoznávané rysy jsou definovány pomocí pravidel, která je nutné pravidelně aktualizovat a přizpůsobovat stále vynalézavějším praktikám spammerů.
2. **Bayesovské**[117] **filtry** – založené na učení s využitím znalostí z oblasti umělé inteligence. Filtry se učí metodou „správně–špatně“, kdy jsou programu předkládány e-maily označené jako spam a e-maily, které spamem nejsou.[118] Filtr si ukládá informace o jednotlivých dopisech rozložených na odstavce či slova a podle těchto informací pak v režimu rozpoznávání přiřadí každému předloženému dopisu odpovídající pravděpodobnost, že je to spam. Tento způsob filtrace může používat i laik. Vysoká účinnost spočívá v tom, že sám koncový uživatel označí podle svého individuálního názoru, co je pro něj spam a co ne, a podle toho se filtr „učí“.

7.5.2 Hoax

Je zřejmé, že pomocí Internetu lze šířit nepředstavitelné množství informací. Ne vždy jsou ale tyto informace pravdivé, a v mnoha případech lze dokonce mluvit o šíření závadných a škodlivých informací jednotlivcům i skupinám.

Nejzajímavější, ale zároveň nejnebezpečnější formou je šíření nepravdivých varování, známých jako **hoax**[119]. Jedná se o nevyžádanou zprávu, která uživatele varuje před nějakým nebezpečím, prosí o pomoc, snaží se ho pobavit apod. Hoax většinou obsahuje i výzvu žádající další rozeslání hoaxu mezi přátele, příp. na co největší množství dalších adres, proto se někdy označuje také jako *řetězový e-mail*.

Příklady hoaxu

Hoax dezinformuje masy lidí a vyvolává v nich mylné přesvědčení, je proto mnohem nebezpečnější než spam. Příkladem šíření hoaxu je zpráva o viru, který neodhalí ani antivirový program. Pomocí podrobného návodu vede uživatele ke smazání souboru, ve kterém má být vir skryt. Přitom se jedná o nepostradatelný systémový soubor. Běžní uživatelé se mohou na hoax nachytat a nevědomky a v dobré víře jej následně posílat dál.

Jiným příkladem je zpráva mířící na city s prosbou o darování krve, hledání ztracené osoby nebo přímo vylákání peněz. Některé z těchto zpráv původně opravdu rozeslali lidé ve svízelné životní situaci, ale hoaxy často přežívají mnohem déle, než měl autor v úmyslu. Známý je případ s žádostí o darování krve šířený více než čtyři roky po smrti pacienta.

[117] Thomas Bayes (* 1701, † 1761) byl anglický matematik, statistik, filozof a duchovní. Formuloval větu o souvislosti podmíněné pravděpodobnosti nějakého jevu s opačnou podmíněnou pravděpodobností.
[118] Používá se pro ně označení *ham* (šunka).
[119] Termín pochází ze 17. století a obecně označuje podvod, mystifikaci nebo žertovnou klamnou zprávu.

Ochrana proti hoaxu

Hoaxy se zabývá specializovaný server, který vede databázi většiny známých zpráv a přidává k nim stanovisko dotčené instituce. Je k dispozici na adrese `http://www.hoax.cz`. Zcela obecně platí, že obsahuje-li zpráva výzvu k hromadnému přeposílání, jedná se s vysokou pravděpodobností o hoax.

Pro úplnost ještě dodejme, že do oblasti počítačové kriminality patří také trestné činy související s obsahem. Zejména jde o šíření dětské pornografie a extremismu na Internetu. Této problematice se ale věnovat nebudeme.

Kontrolní otázky

27. Jaký je princip spamu?
28. Jak lze chránit e-mailové adresy před roboty?
29. Na jakých principech pracuje antispamový filtr?
30. Čemu říkáme hoax a proč je nebezpečný?
31. Kde můžeme ověřit, zda je doručená zpráva hoax?

7.6 Prostředky pro zajištění ochrany dat

V souvislosti s přenosem dat (viz kap. 5.3, str. 110) jsme dospěli k závěru, že zabezpečení dat je možné jen za cenu *redundance*, tedy zvýšení jejich objemu. Požadovanou úroveň zabezpečení důležitých dat zajistíme především vytvářením záložních kopií.

7.6.1 Zálohování a archivace

Zálohování

Pod pojmem **zálohování** (*backup*) rozumíme proces, při němž vznikají kopie zdrojových dat. Tímto způsobem chráníme data před poškozením (úmyslným i neúmyslným) nebo ztrátou. Datová záloha je obvykle uložena v jiném úložišti než zdrojová data. Zálohování probíhá nepravidelně (např. v domácnostech) či pravidelně podle rozvrhu (např. ve firmách) a je při něm kladen důraz na možnost rychlé obnovy dat.

Archivace

Chceme-li data dlouhodobě uchovat pro budoucí použití, použijeme **archivaci** (*archiving*). Některá data se archivují povinně podle různých vyhlášek nebo zákonů (například doklady o provedených pracích a jiné účetní dokumenty). Oproti zálohování není při archivaci obvykle kladen důraz na možnost rychlé obnovy dat. Účelem archivace je především dlouhodobé uchování dat, která již nejsou potřeba pro každodenní využití.

Při archivaci dat se můžeme setkat se strategií, která původní data poté, co byla archivována, maže. Příkladem může být e-mailový klient typu MS Outlook, ve kterém jsou staré zprávy při archivaci přesouvány z doručené pošty do archivu.

Archivace a zálohování jsou (i přes zřejmé rozdíly) podobnými procesy – data potřebujeme ve vhodném tvaru a ve vhodné době uložit na vhodné místo, z něhož je musíme být schopni v případě potřeby co nejrychleji a nejjednodušeji opět vzít a použít. Zálohujeme soubory, adresářové podstromy, disky, systémové soubory i celé oblasti.

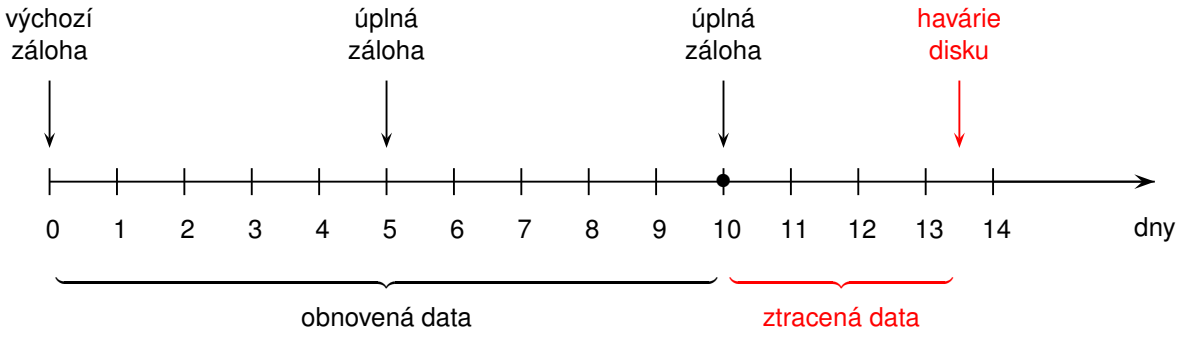
Pro různé podmínky se používají různé strategie zálohování. Volba správné strategie je závislá na tom, jestli je potřeba se zálohami pracovat velmi často, nebo je naopak požadována maximální délka archivace zálohovaných dat. Existují i další kritéria, která odrážejí konkrétní specifické podmínky. Než se pustíme do zálohování, měli bychom mít vždy zcela jasno v odpovědích na následující otázky:

1. Který typ zálohy použijeme?
2. Jaká bude vzdálenost kopie?
3. Jakou zvolíme četnost záloh?
4. Který program použijeme?

Typy záloh

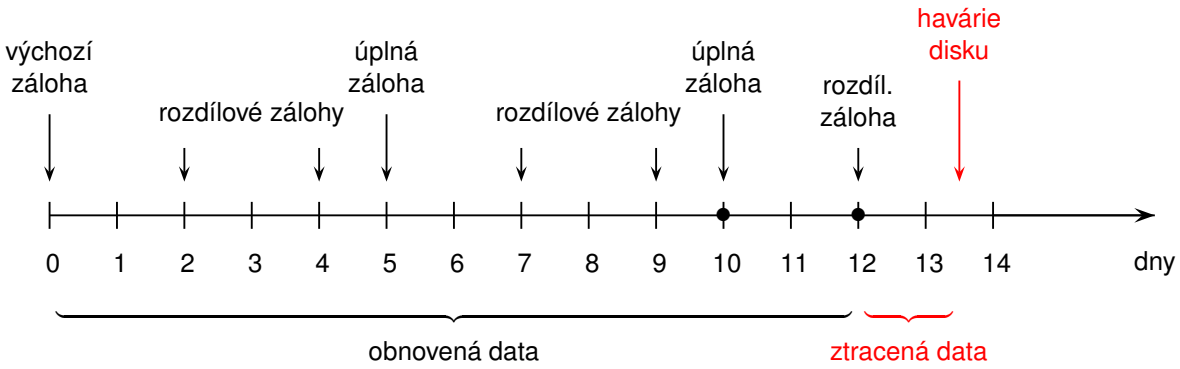
Podle objemu dat určených ke zpracování rozlišujeme několik typů záloh. Základem je **výchozí záloha** (*default backup*), která obsahuje kopii původního systému. Provádí se bezprostředně po první instalaci a zálohuje všechny programy a uživatelské soubory.

Úplná záloha (*full backup*, obr. 7.1) je podobná výchozí záloze, ale provádí se pravidelně. Vždy obsahuje všechna potřebná data, která byla v okamžiku vytvoření zálohy k dispozici. Jedná se o časově i objemově nejnáročnější metodu zálohování, často se vytváří obraz disku. Puntíky na časové ose znázorňují zálohy, které je nutné použít k obnově dat.



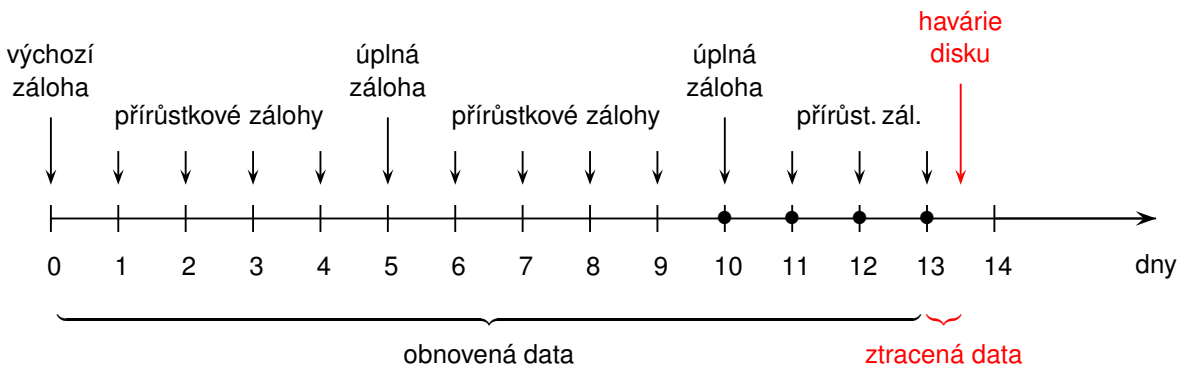
Obrázek 7.1: *Obnova dat při používání pětidenních úplných záloh*

Rozdílová záloha (*differential backup*, obr. 7.2) obsahuje veškerá data, která byla změněna od poslední úplné zálohy. S přibývajícím časem od poslední úplné zálohy narůstá objem zálohovaných dat i čas potřebný k vytvoření zálohy.



Obrázek 7.2: *Obnova dat při používání pětidenních úplných záloh a dvoudenních rozdílových záloh*

Přírůstková záloha (*incremental backup*, obr. 7.3) obsahuje pouze data, která byla změněna od poslední libovolné zálohy (úplné, rozdílové nebo přírůstkové). Výhodou je časová nenáročnost a malý objem zálohovaných dat. Nevýhodou je nutnost obnovy poslední úplné zálohy a všech následujících přírůstků.



Obrázek 7.3: *Obnova dat při používání pětidenních úplných záloh a denních přírůstkových záloh*

Aby byl proces zálohování maximálně efektivní, je nutné připravit **plán záloh** (*backup schedule*), ze kterého bude zřejmé, v kterém okamžiku se má záloha provést. O samotnou realizaci automatického zálohování se postará vhodný software, který také pomůže předejít lidskému selhání.

V případě archivace obsahuje záloha všechna data určená k uchování, neboť požadujeme, aby archiv uchovával stav dat v určitém časovém okamžiku. Nevytváříme proto rozdílové ani přírůstkové zálohy.

Zálohování bývá často spojen s doplňkovými operacemi, které pomáhají zvýšit efektivitu celého procesu. Jedná se například o tyto operace:

- *deduplikace* – odstranění duplicitních složek a souborů ze zálohy;
- *duplikace* – vytvoření více záloh na různých médiích a různých místech pro zvýšení rychlosti při obnově dat a ochraně zálohy před poškozením;
- *komprese* – prakticky ve všech případech (viz také kap. 6.3, str. 152);
- *šifrování* – u datových nosičů s nutností zabezpečení proti neoprávněným osobám (viz též kap. 5.3.2, str. 124), nevýhodou je časová náročnost a snížení účinku komprese.

Vzdálenost kopie

Zálohování dat je realizováno kopií do archivního prostoru. Podle umístění tohoto prostoru můžeme zálohy rozdělit na několik druhů. Je zřejmé, že se vzdáleností záložní kopie od zdrojových dat roste i bezpečnost těchto dat.

Příruční záloha se nachází na stejném disku jako zdrojová data a je ideální ochranou před nechtěnou chybnou operací uživatele. Neochrání však před napadením virem nebo technickou poruchou disku.

Odkládací záloha se nachází na stejném počítači, ale na jiném disku. Může sloužit podobně jako příruční záloha, ale navíc nabízí možnost ochrany před zavirováním nebo technickou poruchou pracovního disku. Tento princip se často využívá při zrcadlení, kdy jsou v počítači dva fyzické disky, na nichž se udržuje neustále stejný tvar záznamu (viz str. 201).

Bezpečnostní záloha se nachází zcela mimo pracovní počítač a je umístěna na archivním médiu nebo na jiném počítači v síti. Při uložení na archivním médiu (diskety, CD,

DVD, Blu-ray, magnetické pásky, externí disky, flash paměti apod.) je třeba dbát na rozdílné schopnosti uchování dat a optimální podmínky skladování.[120] Ukládání kopie na jiný počítač v síti může být provedeno méně často a pomaleji než na disk ve stejném počítači, ale bezpečnost uložení vůči poruchám a napadení daného počítače se tím výrazně zvyšuje. Na lokální síti se s výhodou využívají tzv. NAS (*Network Attached Storage*), v poslední době zažívají velký rozmach vzdálené zálohovací služby na principu cloud computingu. Jsou dostupné v základu většinou zdarma, za příplatek umožňují široké možnosti zálohování a vysokou kapacitu úložného prostoru. Z nejznámějších webových úložišť jmenujme alespoň Dropbox a Microsoft OneDrive (dříve SkyDrive).

Z výše uvedeného je zřejmé, že *čím je fyzická vzdálenost záložní kopie větší, tím je silnější ochrana dat* například před požárem nebo povodní.

Četnost záloh

Podle vzdálenosti archivu lze zhruba říct, že vzdálenost kopie je nepřímo úměrná četnosti zálohování. Pro zálohování nebo archivaci určitých dat tedy musíme zvolit optimální vzdálenost archivu, která na jedné straně zajišťuje požadovanou úroveň bezpečnosti, na druhé straně umožňuje data efektivně kopírovat. Pro příruční zálohy můžeme volit okamžité kopie, tedy kopie při každém uložení nebo změně ostrých souborů, pro kopie na jiné počítače nebo archivní média se volí denní, týdenní či měsíční četnosti.

Pro zálohování i archivaci je nutné vytvořit organizační režim. Možnost ztráty dat bývá často podceňována a dokud nedojde k výraznějšímu incidentu, neexistuje žádný dohodnutý způsob, jak s daty nakládat. Uvádí se, že kompletní ztráta dat může až v 90 % případů vést k ukončení činnosti firmy. Toto nebezpečí si však připouští poměrně málo subjektů.

Softwarová podpora zálohování

Proces zálohování může probíhat ve dvou režimech: *on-line* za běžného chodu počítače, nebo *off-line* mimo běžný provoz počítače, obvykle za pomoci zavedení speciálního média. Kopie ostrých dat do záložního nebo archivního prostoru je velmi častou činností, proto je potřebné ji podporovat v mnoha programech.

Podle charakteru můžeme zálohovaná data rozdělit do čtyř skupin: operační systém, data operačního systému, aplikace, uživatelská data. Při zálohování většího množství dat se obvykle používá specializovaný program (například i v systému Microsoft Windows je součástí instalace), který celý proces zálohování usnadňuje. Pro zálohování většího množství dat je možné použít také specializovaná zařízení (hardware), která pracují poloautomaticky nebo plně automatizovaně. Vzhledem k odlišnosti operačních systémů a aplikací pro osobní počítače a servery existují různé zálohovací software pro všechny typy systémů.

V rámci programového vybavení v operačním systému Windows jsou k dispozici aplikační i systémové služby, které mohou provádět různé zálohovací nebo archivační kroky.

- *Pravidelné ukládání* rozpracovaných dat na disk – lze chápat jako ochranu proti nečekanému výpadku napájení nebo chybě uživatele. V programech lze nastavit vhodnou četnost (řádově v minutách).
- *Vytváření záložních souborů* při uložení – některé editory provedou při uložení rozpracovaných dat zálohu předchozí verze souboru, pokud na disku existovala. Názvy těchto souborů se od hlavních dat odlišují nejčastěji rozšířením `.bak` (liší se podle operačního systému), nebo přidáním vlnky na konec názvu. U některých programů je také možné zvolit, do jakého adresáře a s jakým názvem se záloha provede.

[120] V této souvislosti je třeba zdůraznit, že oblíbené flash disky USB nejsou pro zálohování příliš vhodné a už vůbec bychom je neměli používat k archivaci. Použitá technologie fyzického zapisování do paměti je schopna udržet data bez připojení k napájení pouze několik desítek měsíců.

- *Kopírování souborů* – činnost je prováděna ručně, uživatel tedy musí explicitně kopii zadat, je možné provést libovolný transfer dat.
- *Zálohovací program* `backup` – umožňuje zálohovat nejen uživatelská data ve zvolených složkách, ale i systémové záznamy. Program umožňuje definovat, která data, ze kterých složek a do kterých cílových míst se mají zálohovat, dále nastavit periodický čas kopie apod. Rovněž umožňuje obnovit systémové oblasti, které se ukládají bez zásahu uživatele. Lze tak obnovit činnost systému po narušení nebo chybné instalaci některých programů.
- *Plánovač úloh* – obecný prvek umožňující nařídit spuštění komprimátoru, který zařídí vytvoření archivní souborů a uložení do archivního prostoru. Plánovač definuje časový okamžik nebo periodicitu akce.

V systémech typu Unix jsou k dispozici rovněž obdobné prostředky pro podporu zálohování a archivace.

- *Archivační program* `tar` (*tape archiver*) – standardní výbava unixových systémů umožňující vytvářet archivní soubory ze stanovených souborů a adresářů. V implicitním tvaru jen soustřeďuje data do jednoho celku, původně určeného pro kopii na pásku. Je možné nařídit, aby byl archivní soubor navíc komprimován standardním komprimátorem `gzip` nebo `bzip2`, výsledný archiv má potom obvykle rozšíření `.tar.gz` (též `.tgz`) nebo `.tar.bz2` (též `.tbz2`).
- *Archivační program* `dump` – zálohuje data přímo čtením příslušného blokového zařízení. Výhodou je vysoká rychlost. Program umožňuje inkrementální zálohování, ale je podporován pouze pro souborové systémy ext2 a ext3. Pro souborový systém XFS existuje podobný program `xfsdump`. Pro obnovu souborů z archivu slouží program `restore`.
- *Plánovač úloh* `cron` – softwarový démon, který umožňuje spustit požadovaný proces v požadovanou dobu včetně periodických nastavení. Příkazy a časy jsou definovány obvykle v souboru `crontab`, celosystémová definice v souboru `/etc/crontab`.

Kromě výše uvedených programů si uživatel operačního systému typu Unix může pro účely archivace snadno vytvořit libovolný skript, v jehož těle nařídí volání příslušného zálohovacího procesu. Naplánováním spuštění tohoto skriptu v plánovači `cron` lze archivní proces podle potřeby spustit.

Samostatnou možností ochrany dat je okamžitý zápis každé změny do **žurnálu** (*log*). Provádí se ukládáním bajtů nebo celých bloků dat místo celých změněných souborů. Průběžný záznam změn v žurnálu umožňuje získat obraz dat v minulosti. Tím se liší od prostého zrcadlení dat na druhý disk.

7.6.2 Synchronizace

Synchronizace dat (*data synchronization*) je proces, který zajišťuje konzistenci dat a jejich kopií na všech synchronizovaných úložištích. V současnosti má široké uplatnění v celé řadě aplikací, zejména ve webových prohlížečích a v mobilních zařízeních. Synchronizace může být také užitečná v šifrování pro udržování veřejných klíčů na serverech. Existuje celá řada dostupných synchronizačních nástrojů.

Pro domácí zálohy na externí disky nebo aktualizaci pro dopravu na USB flash discích můžeme využít nástroje pro **synchronizaci souborů** (*file synchronization*, *syncing*). Automatický proces brání kopírování identických souborů, čímž šetří mnoho času v porovnání s manuální kopií. Je rychlejší a méně náchylný k chybám. *Synchronizace souborů*

Správce verzí (*version control*) je určen k řešení situace, kdy více než jeden uživatel se současně pokusí změnit stejný soubor (na rozdíl od synchronizátorů souborů, které jsou optimalizovány pro situace, kdy pouze jedna kopie souboru bude najednou editována). Z tohoto důvodu sice lze správce verzí použít pro synchronizaci souborů, vyhrazené programy však mají menší režii. Typickými zástupci této kategorie programů jsou CVS[121] a Subversion[122]. *Správce verzí*

Distribuované souborové systémy (*distributed filesystems*) mohou být využity k zajištění více verzí synchronizovaných souborů. To obvykle vyžaduje, aby zařízení ukládající soubory byla vždy připojena. Typickým zástupcem je systém Coda[123]. *Distrib. souborové systémy*

K vytváření přesné kopie souboru dat používáme **zrcadlení** (*mirroring*). Zrcadlová stránka na Internetu je přesnou kopií jiné webové stránky. Této vlastnosti se nejčastěji využívá k poskytování více zdrojů téže informace, je to cenný způsob poskytování spolehlivého přístupu k rozsáhlým objemům dat určeným ke stahování. Typickým nástrojem v této kategorii je program `rsync`[124]. *Zrcadlení*

Pro úplnost dodejme, že všechny nástroje udržují množinu souborů synchronizovanou, avšak pouze správa verzí a zrcadlení umožňují pracovat s větším množstvím kopií.

Kontrolní otázky

32. Jaký je rozdíl mezi zálohováním a archivací?
33. Které typy záloh existují a čím se liší?
34. Co vyjadřuje vzdálenost záložní kopie?
35. Podle jakých kritérií stanovujeme četnost záloh?
36. Které programy můžeme použít pro zálohování a archivaci?
37. K čemu se používá synchronizace a jakou může mít podobu?
38. Které programy můžeme použít pro synchronizaci?

[121] Volně dostupného správce verzí CVS (*Concurrent Versions System*) vytvořil v roce 1986 holandský informatik Dick Grune (* 1939). Program je dodáván v základní verzi pro použití v příkazovém řádku, existují však i grafické nadstavby. Systém je také přímo podporován řadou vývojových prostředí.

[122] Apache Subversion (zkráceně SVN) je nástupcem systému CVS. Vyvíjí se od roku 2000 a snaží se zachovat podobný způsob a styl práce jako CVS, ale odstranil některé jeho nedostatky.

[123] Distribuovaný souborový systém Coda (*Constant Data Availability*) vyvíjí na univerzitě Carnegie Mellon od roku 1987 Mahadev Satyanarayanan (* 1953). Program stále ve vývoji, který se zaměřuje na vytvoření robustního produktu pro komerční využití.

[124] Program vytvořil v roce 1996 australský programátor Andrew Tridgell (* 1967). Nástroj se snaží minimalizovat přenosy dat tím, že přenáší pouze rozdílové informace a zároveň pomocí kontrolního součtu porovnává skutečný obsah obou kopií. Data jsou proto (na rozdíl od jiných nástrojů tohoto typu) přenášena v každém směru pouze jednou.

Shrnutí

V závěrečné kapitole jsme si nejdříve vymezili softwarové pirátství a uvedli několik důležitých typů licencí k softwarovým produktům, kterými by se jejich uživatelé měli řídit.

Poté jsme se věnovali trestným činům proti důvěryhodnosti, integritě a dostupnosti počítačových dat a systémů. Kromě hackingu a crackingu jsme se zabývali především malwarem ve všech jeho podobách od počítačových virů přes červy a trojské koně až po rootkity. Pozornost byla věnována také šedé zóně softwaru, kam řadíme spyware, adware a dialery. Nezapomněli jsme na možnosti obrany proti různým formám infiltrace, kterou nabízejí antivirové a antispywarové systémy.

V další části jsme se seznámili se sociálním inženýrstvím a rozebrali tři jeho nejpoužívanější techniky – spoofing, phishing a pharming. Dále jsme se krátce věnovali kyberstalkingu a kyberšikaně a zdůvodnili jejich vysokou společenskou nebezpečnost.

Předposlední podkapitola byla věnována šíření nevyžádaných a poplašných zpráv a také možnostem obrany pomocí filtrace e-mailových zpráv podle různých kritérií.

V poslední podkapitole jsme zakončili problematiku bezpečnosti dat přehledem prostředků pro zajištění jejich ochrany. Objasnili jsme si praktické rozdíly mezi archivací a zálohováním a seznámili se s různými strategiemi zálohování a softwarovou podporou tohoto procesu v různých operačních systémech. Poté jsme se věnovali zajištění konzistence dat a popsali různé možnosti a nástroje pro synchronizaci dat i správu verzí.

Desatero bezpečnosti při práci s počítačem

1. Vědět, co hrozí, a přemýšlet nad riziky.
2. Udržovat operační systém a veškerý software aktualizovaný.
3. Mít zapnutý firewall.
4. Používat antivirový software.
5. Používat antispamovou a antivirovou kontrolu e-mailu.
6. Důvěřovat, ale prověřovat.
7. Mít bezpečné heslo.[125]
8. Nespouštět žádné programy z nedůvěryhodných zdrojů.
9. Při práci v internetovém bankovnictví nemít současně spuštěné rizikové aplikace.
10. Nepovolovat nepoužívané funkce a neinstalovat nepotřebné programy.

Závěrečné motto k zamyšlení: *Stoprocentně zabezpečený počítač nikdy nebyl připojen k Internetu a nikdy do něj nebylo vsunuto záznamové médium.*

[125] Heslo by mělo mít nejméně 8 znaků, nejlépe různých typů (písmena, číslice, interpunkce apod.). Heslo nesmí být slovem žádného přirozeného jazyka a je nutné jej několikrát ročně měnit. V žádném případě nesmí být heslo nikde napsáno.

Slovníček pojmů

Abandonware: software, který je již zastaralý a neprodává se, ale přesto se šíří.
Adware: programy obvykle instalované jako součást jiného bezplatného softwaru.
Aktivní obrana viru: proces ochrany virového kódu s cílem likvidace antivirového systému.
Antispam: program, jehož úkolem je odhalování spamu ve zprávách.
Antispyware: program, jehož úkolem je vyhledávání a likvidace spywaru.
Antivir: program, jehož úkolem je vyhledávání a likvidace malwaru.
Antivirový štít (scanner): modul antivirového systému monitorující běžnou činnost na počítači, který kontroluje přenosy dat a testuje data na přítomnost virových signatur.
Archivace: proces uchování dat pro budoucí použití, často povinný dle platné legislativy.
Bayesovské filtry: filtrační antispamová technika využívající znalosti z umělé inteligence k určení, s jakou pravděpodobností je zpráva spam.
Bezpečnostní záloha: soubor zálohovaných dat nacházející se na archivním médiu nebo jiném počítači v síti.
Beerware: způsob distribuce umožňující využití díla při vypití piva na autorovo zdraví.
Blacklisting: filtrační antispamová technika využívající seznam podezřelých adres.
BSD: licence umožňující volné užití a šíření díla při uvedení autora.
Cardware: způsob distribuce umožňující využití díla při zaslání pohlednice.
Cookies: data identifikující klienta na serveru, jejich podvržením lze zneužívat identitu.
Cracking: nedovolené vniknutí do chráněného systému s cílem získat neoprávněnou výhodu či poškodit majitele a uživatele systému.
Demoverze: komerční software s omezením dostupný zdarma jako ukázka plné verze.
Dialer: škodlivý program, který mění způsob přístupu k síti Internet přes modem přesměrováním na linky s vyšším tarifem.
Dokumentový virus: počítačový virus napadající dokumenty v aplikacích kancelářských balíků podporujících makra.
Donationware: forma distribuce softwaru za malý příspěvek pro charitu.
DoS útok: kybernetický útok s cílem znepřístupnění služby, počítače nebo celé sítě.
EULA: licence pro uživatele proprietárního softwaru s otevřeným kódem, ale bez možnosti jeho editace a šíření.
Freeware: software k užití zdarma, autor si však může vyhradit určitá specifická omezení.
GNU GPL: licence na svobodný software se zdrojovými kódy, které lze volně upravovat.
Grayware: aplikace nebo soubory, nejsou označeny jako malware, ale chovají se nepříjemným nebo nežádoucím způsobem.
Graylisting: filtrační antispamová technika založená na odmítnutí prvního pokusu o doručení zprávy, neboť roboti se o opakované doručení zpravidla nepokoušejí.
Hacking: obejití či prolomení ochrany a průnik do systému jinou než standardní cestou.
Heuristická analýza: proces hledání virů na základě analýzy charakteristických projevů.
Hoax: nevyžádané poplašné zprávy, nepravdivá varování, prosby o pomoc apod., často s výzvou k dalšímu masivnímu šíření.
Komerční software: software poskytovaný za finanční úplatu, zpravidla proprietární.
Kyberstalking: úmyslné pronásledování a obtěžování osob s využitím prostředků ICT.

Kyberšikana: využití ICT pro zasílání obtěžujících, urážejících či útočících zpráv.
Malware: škodlivý software nebo jiná data, jejichž účelem je škodit počítači nebo jeho obsahu či je jinak zneužít.
Multipartitní virus: počítačový virus kombinující více mechanismů šíření.
Odkládací záloha: soubor zálohovaných dat na jiném disku než zdrojová data.
OEM: licence k softwaru získaná se zakoupením hardwaru či jiného softwaru, často s omezenými funkcemi.
Otevřený software: programový produkt dostupný zdarma i se zdrojovými kódy.
Pasivní obrana viru: proces ochrany virového kódu před některými technikami hledání.
Pharming: technika sociálního inženýrství založená na hacknutí DNS serveru a následném vylákání důvěrných a citlivých informací.
Phishing: technika sociálního inženýrství spočívající v získávání důvěrných a citlivých informací za účelem získání přístupu k cizím zdrojům.
Plán záloh: nástroj k automatizaci zálohovacího procesu, často prostřednictvím specializovaného programu.
Počítačové (softwarové) pirátství: užití programů nebo datových souborů bez vědomí autora a bez zaplacení příslušných poplatků.
Počítačový virus: umělý útvar vytvořený člověkem, který se dokáže sám šířit bez vědomí uživatele a provádí škodlivou činnost.
Počítačový červ: škodlivý program, který se šíří prostřednictvím Internetu, nepřipojuje se k žádnému programu ani se na disku dále nemnoží.
Proprietární software: programový produkt bez zdrojových kódů, jehož používání upravuje licence.
Příruční záloha: soubor zálohovaných dat nacházející se na stejném disku jako zdrojová data jako prevence před nechtěnou chybou uživatele.
Přírůstková záloha: soubor dat, která byla změněna od poslední úplné, přírůstkové nebo rozdílové zálohy.
Rootkit: škodlivý kód, který dokáže maskovat svou přítomnost v počítači změnou funkce jádra operačního systému.
Rozdílová záloha: soubor dat, která byla změněna od poslední úplné zálohy.
Shareware: software chráněný autorským zákonem, který je možné volně distribuovat, zpravidla s integrovaným omezením.
Slovníkový útok: útok hrubou silou s využitím předem připraveného seznamu slov.
Sniffing: technika crackingu zaměřená na sledování a analýzu provozu v počítačové síti.
Sociální inženýrství: manipulace lidí za účelem provedení akce nebo získání informace.
Softwarový cracking: protiprávní zásah do dat s cílem odstranění ochrany a jeho následného neoprávněného užívání a šíření.
Souborový virus: počítačový virus napadající spustitelné soubory.
Spam: nevyžádaná zpráva, která se masivně šíří Internetem a dalšími kanály.
Spoofing: technika sociálního inženýrství s cílem podvržení údajů.
Správce verzí: nástroj pro řešení situace, kdy více než jeden uživatel současně pracuje se stejným souborem.
Spyware: nechtěné programy, které se instalují bez vědomí uživatele a poškozují jej.

Svobodný software: programový produkt dostupný se zdrojovými kódy s právem použití, modifikace a distribuce i v komerční sféře.
Synchronizace: proces zajišťující konzistenci dat a všech jejich kopií.
Trial: programový produkt s časovým omezením určený k vyzkoušení.
Trojský kůň: škodlivý kód, který není schopen sebereplikace a infekce souborů.
Úplná záloha: pravidelně vytvářená kopie systému obsahující obraz disku se všemi daty.
Útok hrubou silou: technika k prolomení ochrany testováním všech možných kombinací.
Virová signatura: charakteristický vzorek virového kódu v databázi antivirového programu, podle něhož je možné provádět vyhledávání.
Volné dílo: software dostupný bez nároku na autorská práva, v českém právním systému bezúplatná licence na libovolné využití díla.
Výchozí záloha: kopie původního systému vytvořená bezprostředně po první instalaci.
Warez: autorská díla, se kterými je nakládáno v rozporu s autorským zákonem.
Zálohování: proces ochrany dat před poškozením nebo ztrátou, při němž vznikají kopie zdrojových dat.
Zaváděcí virus: počítačový virus napadající zaváděcí sektor disku.
Zrcadlení: technika používaná k vytváření přesné kopie souboru dat.

Řešení kontrolních otázek

1. *Co se rozumí pod pojmem počítačové pirátství?*

Jako počítačové pirátství označujeme užití programů nebo datových souborů bez vědomí autora a bez zaplacení příslušných licenčních poplatků.

2. *Které zákony se zabývají problematikou počítačové kriminality?*

Počítačovým pirátstvím se zabývá zákon č. 121/2000 Sb. (autorský zákon) a také zákon č. 40/2009 Sb. (trestní zákoník).

3. *Jaký je rozdíl mezi komerčním a proprietárním softwarem?*

Proprietární software se vyznačuje tím, že nemá volně dostupné zdrojové kódy a při jeho použití se uživatel musí řídit podmínkami licenční smlouvy. Komerční software je každý programový produkt, který nelze užívat bezplatně. Ve většině případů mezi komerčním a proprietárním softwarem není rozdíl, ale toto pravidlo neplatí vždy.

4. *Jaký je rozdíl mezi otevřeným a svobodným softwarem?*

Svobodný i otevřený software lze používat bez omezení, avšak zatímco otevřený software lze za všech okolností užívat zdarma, svobodný software musí být dostupný i pro komerční využití.

5. *Jaký význam mají licence k softwarovým produktům?*

Licenční smlouva se uzavírá mezi držitelem autorských práv (autorem) a uživatelem díla a autor v ní určuje, jakým způsobem je možné s dílem nakládat. Uživatel softwarového díla získá s licencí oprávnění k užití díla, nikoli dílo samotné.

6. Jaká omezení vyplývají z licencí OEM?

Licence OEM je k danému softwaru získána současně se zakoupením hardwaru či jiného softwaru. Taková verze softwaru má často omezené funkce a je možné ji používat pouze na počítači, na kterém byl výrobcem nainstalován.

7. Které licence a formy distribuce umožňují bezplatné využití softwaru i dat?

Bezplatné užití díla umožňují zejména licence BSD, GNU GPL a formy distribuce cardware, beerware, demoverze, freeware, shareware, trial (po určitou dobu) a volné dílo (public domain). V některých případech též licence EULA. Obecně se jedná o otevřený software, často také o svobodný software.

8. Které licence a formy distribuce vyžadují platbu za software i data?

Platbu za užití díla vyžadují produkty označované jako donationware (symbolický příspěvek pro neziskovnou či charitativní organizaci), shareware a trial (po uplynutí stanovené doby). Obecně se jedná o komerční software, typicky pod licencí OEM.

9. V čem se zásadně liší freeware a shareware?

Freeware je typ softwaru, jehož užívání je zcela zdarma. Shareware je typ softwaru, který lze volně distribuovat a zdarma vyzkoušet, ale pro další používání je třeba jej zakoupit nebo se zaregistrovat.

10. Čím se vyznačuje volné dílo (public domain)?

Autor díla označeného jako public domain se rozhodl jej poskytnout bez nároku na autorská práva, což však v některých právních systémech (včetně českého) není možné. Proto se public domain chápe jako bezúplatná licence pro libovolné využití díla bez domáhání se autorských práv.

11. Jaké jsou možnosti obrany proti počítačovému pirátství?

Možnosti obrany proti pirátství jsou relativně omezené. Používá se systém registrace uživatelů a vzdálená kontrola licenčního čísla produktu, hardwarový klíč s přístupovým kódem nebo speciální druh záznamu na distribuovaném médiu, který není možné zkopírovat v původní podobě. Všechna technická řešení však lze s určitými znalostmi obejít při vynaložení určitého úsilí a prostředků.

12. Jaký je rozdíl mezi hackingem a crackingem?

Hacking se vyznačuje obcházením nebo prolamováním zabezpečení a pronikání do systémů nestandardní cestou. Cracking je jednou z nejnebezpečnějších forem hackingu. Cílem crackera je získat pro sebe neoprávněnou výhodu nebo poškodit legální uživatele systému.

13. Na jakém principu je založen sniffing?

Principem sniffingu je sledování a analýza provozu v počítačové síti, což umožňuje získání informací a jejich následné zneužití.

14. Které metody se používají k prolamování hesel?

Prolomení hesla lze provést útokem hrubou silou, kdy se testují všechny možné kombinace. Efektivnější postup nabízí slovníkový útok, při kterém se testují pouze smysluplné výrazy podle předem připraveného seznamu.

15. Co je to malware a jaké má formy?

Malware je souhrnné označení pro škodlivý software či jiná data, jejichž účelem je uškodit počítači nebo jeho obsahu či je jinak zneužít. Vyskytuje se v mnoha formách, z nichž nejvýznamnější jsou počítačové viry, červy a trojské koně.

16. Jak pracuje počítačový virus a do kterých kategorií se počítačové viry dělí?

Počítačový virus pracuje ve třech fázích – šíření (infikování dalších programů a počítačů), obrana (pasivní či aktivní), útok (provádění zpravidla škodlivých akcí na infikovaném počítači). Viry se dělí na zaváděcí (napadají zaváděcí sektor disku), souborové (napadají spustitelné soubory), dokumentové (napadají dokumenty kancelářských systémů obsahující makra) a multipartitní (kombinace více mechanismů šíření).

17. Co mají společného a čím se liší počítačový virus a počítačový červ?

Viry i červy patří do kategorie škodlivého softwaru. Na rozdíl od virů se červy šíří prostřednictvím sítě, nepřipojují se k žádnému programu ani se na disku počítače dále nešíří.

18. Na jakém principu je založen trojský kůň?

Trojský kůň není schopen sebereplikace ani infekce souborů. Vystupuje jako neškodný spustitelný soubor, který však obsahuje škodlivý kód. Po spuštění kódu vypustí trojský kůň svůj škodlivý obsah (virus, červ nebo jiný malware).

19. Které činnosti provádí spyware?

Spyware poškozuje uživatele tím, že bez jejich vědomí ovlivňuje jeho soukromí a bezpečnost systému, používá systémových zdrojů, sbírá a odesílá citlivá data apod.

20. K čemu slouží antivirový software a které funkce obvykle nabízí?

Antivirový software slouží k detekci a odstranění malwaru a prevenci proti případné nákaze. Mezi jeho funkce patří kontrola otevíraných souborů a spouštěných programů, pravidelná kontrola všech souborů na disku, kontrola e-mailů a navštěvovaných webových stránek.

21. Jaké jsou možnosti prevence proti škodlivému softwaru?

Mezi osvědčené metody platí především používání legálního a aktualizovaného softwaru, antivirových a antispywarových programů a firewallu, pravidelná aktualizace síťových produktů, preference otevřených souborových formátů, pravidelné zálohování dat a používání zdravého rozumu.

22. Čím se zabývá sociální inženýrství?

Sociální inženýrství manipuluje s lidmi za účelem provedení určité akce nebo získání určité informace. Obvykle se využívá podvodu nebo podvodného jednání za účelem získání utajovaných informací nebo získání přístupu do informačního systému.

23. Co je to spoofing a jaké může mít formy?

Spoofing je útočná technika sociálního inženýrství založená na podvržení údajů. Může mít mnoho forem, od kanadských žertíků přes kontrolu komunikačního kanálu (IP spoofing), podvržení webové stránky (DNS spoofing), podvržení adresy odesílatele e-mailu či vysílání falešných signálů pro navigační systému (GPS spoofing).

24. Jaký je rozdíl mezi phishingem a pharmingem?

Phishing spočívá v získávání důvěrných a citlivých informací za účelem získání přístupu a následného obohacení se z cizích zdrojů. Pharming je nebezpečnější formou útoku, kdy útočník nejdříve hackuje DNS tak, aby oběť neměla o podvrhu žádné tušení.

25. Co je to kyberstalking?

Kyberstalking je zlovolné a úmyslné pronásledování a obtěžování jiné osoby s využitím prostředků ICT. Nejčastěji jde o zasílání zpráv pomocí messengerů, chatu, VoIP technologií apod., útočník také může získávat a shromažďovat množství osobních údajů o oběti.

26. Které typické projevy má kyberšikana?

Kyberšikana se projevuje nejčastěji zasíláním obtěžujících, urážejících nebo útočných zpráv, vytváření webových stránek a blogů s dehonestujícím obsahem, někdy může sloužit k posilování klasických forem šikany.

27. Jaký je princip spamu?

Principem spamu je masové šíření nevyžádaných sdělení všeho druhu (nejčastěji reklamního charakteru) na Internetu i dalšími komunikačními kanály. Zprávy jsou rozesílány automaticky pomocí robotů, kteří také prohledávají webové stránky a vytvářejí databázi e-mailových adres příjemců.

28. Jak lze chránit e-mailové adresy před roboty?

Základní obrana proti robotům spočívá v uvádění e-mailových adres v „nečitelném“ tvaru. „Bezpečný“ tvar adresy by mohl být například `jméno(zavináč)firma.cz` nebo `jméno(at)firma(dot)cz`, často se zavináč nebo i kompletní adresa vkládají v podobě rastrového obrázku.

29. Na jakých principech pracuje antispamový filtr?

Antispamový filtr využívá filtraci příchozích zpráv podle způsobu dopravy (blacklisting, graylisting) a podle obsahu zprávy (filtry vyhledávající typické rysy spamu, bayesovské filtry s využitím umělé inteligence).

30. Čemu říkáme hoax a proč je nebezpečný?

Hoax je označení pro masivně šířené nepravdivé informace, nevyžádané zprávy varující před nebezpečím, prosby o pomoc apod., typicky s požadavkem na další šíření. Nebezpečí spočívá v dezinformaci masy lidí a vyvolávání mylných přesvědčení.

31. Kde můžeme ověřit, zda je doručená zpráva hoax?

Nejlepším způsobem je návštěva specializovaného serveru `http://www.hoax.cz`, kde se nachází databáze většiny známých hoaxů včetně stanoviska dotčených institucí.

32. Jaký je rozdíl mezi zálohováním a archivací?

Zálohování je proces, při kterém vznikají kopie zdrojových dat, která tím chráníme před poškozením či ztrátou. Archivace představuje dlouhodobé uchování aktuálně nepotřebných dat pro budoucí použití. Oproti zálohování není při archivaci kladen důraz na možnost rychlé obnovy dat.

33. Které typy záloh existují a čím se liší?

Základním typem je výchozí záloha, která obsahuje kopii původního systému a provádí se bezprostředně po jeho první instalaci. Úplná záloha má podobný charakter, ale provádí se pravidelně a obsahuje veškerá data, která byla v tom okamžiku k dispozici. Rozdílová záloha obsahuje data, která byla změněna od poslední úplné zálohy – k obnově dat je kromě rozdílové zálohy nutná poslední úplná záloha. Přírůstková záloha obsahuje data, která byla změněna od poslední zálohy libovolného typu (úplné, rozdílové nebo přírůstkové) – k obnově dat je nutná poslední úplná záloha a všechny následné přírůstkové zálohy.

34. Co vyjadřuje vzdálenost záložní kopie?

Podle vzdálenosti záložní kopie od zdrojových dat je možné určit míru bezpečnosti zálohy. Příruční záloha se nachází na stejném disku jako zdrojová data a slouží pouze jako ochrana před chybnou operací uživatele. Odkládací záloha se nachází na jiném disku stejného počítače a slouží navíc jako ochrana před zavirováním nebo poškozením pracovního disku. Bezpečnostní záloha se nachází zcela mimo pracovní počítač (na archivním médiu nebo na síti) a představuje nejvyšší stupeň zabezpečení.

35. Podle jakých kritérií stanovujeme četnost záloh?

Četnost záloh volíme podle charakteru dat a podle vzdálenosti archivu. U příručních záloh se používají okamžité kopie při každé změně zdrojových dat, u odkládacích a bezpečnostních záloh volíme denní, týdenní či měsíční četnosti.

36. Které programy můžeme použít pro zálohování a archivaci?

Zálohování může probíhat on-line za běžného chodu počítače nebo off-line na vyžádání uživatele. Kromě specializovaných programů jsou v každém operačním systému k dispozici aplikační i systémové služby, které mohou provádět archivační nebo zálohovací kroky. V systému Windows je k dispozici program `backup` a systémový plánovač úloh, operační systémy typu Unix obsahují archivační programy `tar` a `dump` a plánovač úloh `cron`. Kromě uvedených prostředků si uživatelé v systémech typu Unix mohou vytvořit vlastní skript.

37. K čemu se používá synchronizace a jakou může mít podobu?

Synchronizace je proces, který zajišťuje konzistenci dat a jejich kopií na synchronizovaných úložištích. Uplatňuje se v celé řadě aplikací včetně šifrování (udržování veřejných klíčů na serverech) a je dostupný v celé řadě různých nástrojů. Může mít podobu synchronizace souborů nebo správce verzí, často se využívá principu zrcadlení.

38. Které programy můžeme použít pro synchronizaci?

Pro synchronizaci existují velké množství specializovaných programů, například správce verzí CVS nebo Subversion, distribuovaný souborový systém Coda nebo program `rsync` využívající zrcadlení.

Použitá a doporučená literatura

Budai, D. *Základy ergonomie: jak si nezničit zdraví u počítače* [on-line]. 16. 3. 2012 [cit. 2014-03-09]. Dostupné na `http://pcrady.cnews.cz/zaklady-ergonomie-jak-si-neznicit-zdravi-u-pocitace`.

Farana, R. *Kódování a komprese dat.* 1. vyd. Ostrava : Ostravská univerzita, 2014. 153 s.

Frekvence písmen, bigramů, trigramů, délka slov [on-line]. [cit. 2023-12-01]. Dostupné na `https://nlp.fi.muni.cz/cs/FrekvencePismenBigramu`.

Haluza, P., Rybička, J., Hála, T. *Historie výpočetní techniky.* 2. vyd. Brno : Konvoj, 2024. 79 s. ISBN 978-80-7302-188-7.

Hesteric, R. *Příklady.EU – cvičení z učiva středních škol – matematika, fyzika a chemie.* [on-line]. [cit. 2020-02-15]. Dostupné na `https://www.priklady.eu/`.

Hlavatá, E. *Terminologický slovník informatiky.* 1. vyd. Praha : UVTEI, 1979. 331 s.

Ilustrovaný encyklopedický slovník. Sv. 1. Praha : Academia, 1980. 970 s.

Janeček, J. *Rozluštěná tajemství: luštitelé, dešifranti, kódy a odhalení.* 2. vyd. Praha : XYZ, 2008. 268 s. ISBN 978-80-86864-96-9.

Klimeš, L. *Slovník cizích slov.* 3. vyd. Praha : SPN, 1985. 791 s.

Krištoufek, K. a kol. *Výpočetní a řídicí technika.* Oborová encyklopedie. 1. vyd. Praha : SNTL, 1982. 376 s.

Morkes, D. *Komprimační a archivační programy.* 1. vyd. Brno : Computer Press, 1998. 178 s. ISBN 80-7226-089-8.

Pezlar, Z., Rybička, J. *Teoretické základy informatiky.* 1. vyd. Brno : Konvoj, 2001. 52 s. ISBN 80-85615-99-1.

Rybička, J. a kol. *Informatika pro ekonomy.* 3. vyd. Brno : Konvoj, 2007. 139 s. ISBN 80-7302-128-6.

Rybička, J., Talandová, P. *Informatika pro ekonomy.* 1. vyd. Praha : Alfa Nakladatelství, 2009. 252 s. ISBN 978-80-87197-24-0.

Singh, S. *Kniha kódů a šifer: utajování od starého Egypta po kvantovou kryptografii.* 2. vyd. Praha : Dokořán, 2011. 380 s. ISBN 978-80-7363-268-7.

Zdroje obrázků

Obrazové materiály byly převzaty z následujících serverů: `kancelarskezidle.com`, `mendelu.cz`, `wikimedia.org`.

Podklad pro obálku převzat ze serveru `pixabay.com`.

Použité fonty

Text byl vysázen systémem XƎLaTeX fontem Libertinus (autoři C. MacIennan a K. Hosny). Pro sazbu v cizích jazycích byly použity fonty rodiny Noto (autor Google).

Seznam zkratek

AAC Advanced Audio Coding
ACL Access Control List
ALAC Apple Lossless Audio Codec
ANSI American National Standards Institute
APFS Apple File System
ASCII American Standard Code for Information Interchange
AVG AntiVirus Guard
AVI Audio Video Interleave
BCD Binary Coded Decimal
BIOS Basic Input-Output System
BMP Basic Multilingual Plane
BOM Byte Order Mark
BSD Berkeley Software Distribution
btrfs B-tree file system
CAD Computer-Aided Design
CCITT Comité Consultatif International Téléphonique et Télégraphique
CDFS Compact Disc File System
CMY Cyan-Magenta-Yellow
CR Carriage Return
CRC Cyclic Redundancy Check
CRT Cathode Ray Tube
CSV Comma-Separated Values
CVS Concurrent Versions System
DCT Discrete Cosine Transform
DNS Domain Name System
DoS Denial of Services
dpi dots per inch
DWF Design Web Format
EBCDIC Extended BCD Interchange Code
EMF Ehnanced Windows Metafile
EOT End of Transmission
EPS Encapsulated PostScript
EULA End-User Licence Agreement
ext extended file system
FAT File Allocation Table
FLAC Free Lossless Audio Codec
GIF Graphic Interchange Format
GIMP GNU Image Manipulation Program
GNU GNU's Not Unix
GPL General Public License
GPS Global Positioning System
HFS+ Hierarchical File System Plus
HTML Hypertext Markup Language
IBM International Business Machines Company
IČO Identifikační číslo organizace
IEC International Electrotechnical Commission
IEEE Institute of Electrical and Electronics Engineers
IP Internet Protocol
ISBN International Standard Book Number
ISO International Organization for Standardization
ITU International Telecommunication Union
JEDEC Joint Electron Devices Engineering Council
JFIF JPEG File Interchange Format
JFS Journaled File System
JPEG Joint Photographic Experts Group
KOI Код Обмена Информацией
LCD Liquid-Crystal Display
LF Line Feed
LSB Least Significant Bit
LZMA Lempel-Ziv-Markov Chain Algorithm
LZW Lempel-Ziv-Welch
MD5 Message-Digest Algorithm 5
MIDI Musical Instrument Digital Interface
MKA/MKV Matroska Audio/Video
MP3 MPEG Audio Layer III
MPEG Moving Picture Experts Group
MS-DOS Microsoft Disk Operating System
MSE Microsoft Security Essentials
MSB Most Significant Bit
NTFS New Technology File System
ODF Open Document Format
OEM Original Equipment Manufacturer
OS Operating System
PDF Portable Document Format
PIN Personal Identification Number
PNG Portable Network Graphics
PS PostScript
RAT Remote Administration Tool
RGB Red-Green-Blue
RIFF Resource Interchange File Format
RLE Run-Length Encoding
RSA Rivest-Shamir-Adleman
RSI Repetitive Strain Injury
RTF Rich Text Format
SHA Secure Hash Algorithm
SI Le Système International d'Unités
SIP Supplementary Ideographic Plane
SMP Supplementary Multilingual Plane
SMS Short Message Service
SSP Supplementary Special-purpose Plane
SPUA Supplementary Private Use Area
SVG Scalable Vector Graphics
SWF Small Web Format
TIFF Tagged Image File Format
TIP Tertiary Ideographic Plane
UBE Unsolicited Bulk Email
UCE Unsolicited Commercial Email
UCS Universal Character Set
UDF Universal Disk Format
UFS Unix File System
UIS Univerzitní informační systém
UTF UCS Transformation Format
VIN Vehicle Identification Number
VOB Video Object
WAV Waveform Audio File Format
WMA Windows Media Audio
WMF Windows Metafile
XCF Experimentaly Computing Facility
XFS Extended File System
XLS Excel Spreadsheet
XML Extensible Markup Language
ZFS Zettabyte File System

Rejstřík

7Z, souborový formát, 162

A
AAC, souborový formát, 150
abandonware, zastaralý nepodporovaný software, 178
absolutní cesta, způsob lokalizace adresáře, 164
ACL, seznam oprávnění, 168
adaptivní komprese, 154, 159
aditivní barevný model, 77, 78, 80
aditivní číselný kód, 52, 55–59, 65
Adleman, Leonard Max, 126
adresář, vymezení pojmu, 164
adware, reklamní software, 178, 187
AI, souborový formát, 149
ALAC, souborový formát, 150
alfa kanál, složka pixelu, 78
alfabeta, řecká abeceda, 18, 22
aliasing, deformace signálu, 83
al-Khwārizmī, Abū ʿAbd Allāh Muḥammad ibn Mūsā (= Algoritmí, أبو عبد الله محمد بن موسى الخوارزمي), 22
al-Kindī, Abū Yūsuf Yaʿqūb ibn ʿIsḥāq aṣ-Ṣabbāḥ (= Alkindus, أبو يوسف يعقوب بن إسحاق الصبّاح الكندي), 22
Android, operační systém, 176, 190
antispam, druh softwaru, 194
antispyware, druh softwaru, 190
antivirový štít, modul antiviru, 188
antivirus, druh softwaru, 188, 189
APFS, souborový systém, 169
aplikační software, 10, 36
Apple, společnost, 71, 150, 169
arabské číslice (= indo-arabské, hindsko-arabské), 22
archiv, soubor s metadaty vzniklý kompresí, 152
archivace dat, *viz* zálohování dat
aritmetické kódování, 158
aritmetické operace
    v číselných soustavách, 25–28
    v počítači, 28, 50, 51, 54, 55, 61–65, 138, 141
ARJ, souborový formát, 162
ASCII (= ISO/IEC 646), znaková sada, 67–73, 76, 97, 105, 109, 138–140, 143
asociace souborového formátu a aplikace, 144–146
asymetrická komprese, 154
asymetrické šifrování, 126–128
atributy
    obrazových objektů, 81
    souborů, 163, 164
autentizace, ověření identity, 180
autorizace, povolení přístupu, 180
AVI, souborový formát, 151

B
babylónské číslice, 20
bajt, jednotka informace, 41
barevná hloubka obrazu, 79, 80, 148, 149, 155, 156, 159
barevná paleta, 80, 161
barevný model, 77–80, 149, 160
    schéma převodu, 78
    vlastnosti, 77
barva, vnímání elektromagnetického záření, 77
Baudot, Jean-Maurice-Émile, 97
Baudotův kód, 97–99
Bayes, Thomas, 195
bayesovské filtry, 195
BCD, číselný kód, 57, 123
beerware, forma distribuce softwaru, 178
Big Endian, varianta vícebajtového kódu, 73, 76, 105, 109
binární (= dvojková) číselná soustava, 17, 24
binární (= netextový) formát dat, 138, 141, 142, 146, 153
binární řádová mřížka, 49–54, 56, 57
    rozšíření o další řády, 56
bit, jednotka informace, 38, 41
bitová mapa, 79, 148
blacklisting, 194
blokové kódování, 97–99
BMP, souborový formát, 142, 148, 151, 161
BMP, základní rovina standardu Unicode, 72, 73, 76
BOM, identifikace znakové sady, 76
bráhmí (𑀩𑀺𑀭𑀳𑀺), staroindické písmo, 21, 22
Brunner, John Kilian Houston, 184
BSD, typ softwarové licence, 177
btrfs, souborový systém, 168

C
Caramuel y Lobkowitz, Juan, 24
cardware, forma distribuce softwaru, 178
CCITT, sada kompresních algoritmů, 159, 161
CDFS, souborový systém, 166
CDR, souborový formát, 142, 145, 149, 151
celá část čísla, 23, 25, 26, 50, 57, 58
certifikační autorita, 126
cluster (= alokační blok), nejmenší jednotka paměti na datovém nosiči, 153, 165–168
CMY, barevný model, 78
CMYK, barevný model, 78, 80
Cohen, Frederick B., 181, 188
cookies, 186
CPT, souborový formát, 145
cracking, 180
CRC, hashovací funkce, 113
CSV, souborový formát, 148

Č
četnost chyby, 110
Čihař, Michal, 146
čínské číslice, 20
číselná soustava, 17–31
    aritmetické operace, 25–28
    nepoziční, 17–19, 21
    polyadická, 23, 24
    poziční, 20–28
    prefix soustavy, 27
    převody mezi soustavami, 25–27

základ soustavy, 17, 19, 20, 23–28
číselný kód, 24, 52–59, 65, 97, 123
aditivní, 52, 55–59, 65
doplňkový, 52–57
inverzní, 53, 56
přímý, 52, 53, 56, 57
číslo, 17, 24, 49–66, *viz také* číselná soustava
celá část čísla, 23, 25, 26, 50, 57, 58
číslice, 17, 18, 20–22, 24, 30
denormalizované, 61
polynomiální zápis, 23, 25, 26
poziční zápis, 23
záporné, 17, 24, 50, 52–58
zlomková část čísla, 23, 25, 26, 50, 57–61

D
data mining, 188
data, vymezení pojmu, 33, 34
datový formát, 137–142, 146
binární (= netextový), 138, 141, 142, 146, 153
specifikace, 140
textový, 138, 140–142, 146
datový typ, 33, 49–52, 56, 57, 65, 66, 137, 138
Deflate, kompresní algoritmus, 159, 161
dekadická (= desítková) číselná soustava, 17, 21, 24
démotické číslice, 18
demoverze, forma distribuce softwaru, 178
denormalizované číslo, 61
desítková číselná soustava, 17, 21, 24
dešifrování, 125–128, *viz také* kryptografie
detekce chyby, 110–113, 115–123
detekční kód, 110–115
dévanágarí (देवनागरी), staroindické písmo, 22
dialer, 185, 187
digitalizace zvukového signálu, 83, 84
diskový oddíl, izolovaná oblast pevného disku, 164, 165, 168, 169
diskrétní (= digitální) signál, 93–95
DOC, souborový formát, 145, 147, 148, 150, 151
DOCM, souborový formát, 147
DOCX, souborový formát, 142, 145–148, 150
dokumentový virus, 182
donationware, forma distribuce softwaru, 177
doplňkový číselný kód (= dvojkový doplněk), 52–58
DoS útok, 180
dpi, jednotka hustoty obrazu, 79–81
dvojková číselná soustava, 17, 24
DWF, souborový formát, 149
DWG, souborový formát, 149
DXF, souborový formát, 149
dynamický signál, 93

E
EBCDIC, znaková sada, 67
efektivní hustota rastrového obrazu, 79, 80
egyptské číslice, 17, 18
elektronický podpis, 127, 128
Elias, Peter, 158
EMZ, souborový formát, 150
endianita, pořadí bajtů ve vícebajtovém kódování, 73, 76, *viz také* BOM
entropie (= úplná informace), 36, 39, 40, 97–105, 109, 152
binárního kódu, 97–105, 109
náhodného jevu, 36, 39, 40, 97
EPS, souborový formát, 149, 151
ergonomie práce s počítačem, 11–13
escape (= ESC), řídicí znak, 67, 141
EULA, typ softwarové licence, 177
exFAT, souborový systém, 167
exponent racionálního čísla, 58–66
ext, souborový systém, 168

F
faktorizace, rozklad součinu na prvočinitele, 126
Fano, Roberto (= Robert) Mario, 103, 158
Farooq Alvi, Amjad (امجد فاروق علوی), 181
Farooq Alvi, Basit (باسط فاروق علوی), 181
FAT, souborový systém, 166
Fibonacci, Leonardo (= Leonardo Pisánský), 22
filtry založené na pravidlech, 195
FLAC, souborový formát, 150
formátová specifikace, 140
Fourier, Jean-Baptiste Joseph, 160
Fourierova transformace, dekompozice funkce do frekvenčních komponentů, 160, 161
fragmentace, nesouvislé uložení dat, 153, 165, 168, 169
freeware, forma distribuce softwaru, 177
frekvenční analýza, 106–108, 124
Fú Xī (伏羲), 24
fyzická komprese, 154
fyzický pixel, 79

G
Gerrold, David (= Friedman, Jerrold David), 181
GIF, souborový formát, 149, 161
Giles, Aaron, 178
GNU GPL, typ softwarové licence, 177
Golay, Marcel Jules Édouard, 120
Google, společnost, 149, 151
grafika v počítači, 76–82, 148–150, 155
Gray, Frank, 123
graylisting, 194
Grayův kód,
grayware, 186
Grune, Dick, 201
GZ, souborový formát, 162

H
hacking, 179, 180
Hamming, Richard Wesley, 116
Hammingova váha, 118
Hammingova vzdálenost, 118
Hammingovy kódy, 116–122
rozšíření kódu, 121
zkrácení kódu, 120
hangŭl (한글), korejské písmo, 72
hardware, technické vybavení počítače, 10
Harriot, Thomas, 24
hartley (Hart), jednotka informace, 38

Hartley, Ralph Vinton Lyon, 38
hashovací funkce, 112, 113, 125
herodiánské číslice, 18
heuristická analýza, 188
hexadecimální (= šestnáctková) číselná soustava, 24
HFS+, souborový systém, 169
hieratické číslice, 18
hieroglyfické číslice, 17, 18
hlavička souborového formátu, 146–148, 156
hoax, 195, 196
Horner, William George, 26
Hornerův polynom, 26
HTML, souborový formát, 142, 147, 151
Huffman, David Albert, 104, 158
Huffmanův kód, 104, 105, 158–160
Hus, Jan, 143
hustota rastrového obrazu, 79–81
hybridní šifrování, 126–128

Ch
chosŏn'gŭl (조선글), korejské písmo, 72
chyba v datech, 110–123

I
IBM, společnost, 67–69, 150, 165, 181, 182
identifikace, určení uživatele, 180
IEEE 754, standard pro reprezentaci čísel s pohyblivou řádovou čárkou, 58–65
indické číslice, 21, 22
informace, 34–36, 38, 40, 42
    měření množství, 38–40, *viz také* entropie
    teorie informace, 35–42
    vymezení pojmu, 34, 36
informační entropie, 36, 39, 40, 97–105, 109, 152
informační systém, 35, 36
informatika, vědní disciplína, 35
inverzní číselný kód (= jedničkový doplněk), 53, 56
iónské číslice, 18
ISO Latin 2 (= ISO/IEC 8859-2), znaková sada, 69–71, 76, 105–109, 139, 140

J
jedničková číselná soustava, 23
jednobajtová znaková sada, 67–71, 105, 109
jednoduchá parita, 110–112
jednosměrná šifra, 125
jednotky informace, 41, 42
jev, libovolný náhodný proces, 36
JFS, souborový systém, 168
JPEG (= JFIF), souborový formát, 104, 142, 145, 149, 151, 161
JPEG, sada kompresních algoritmů, 159–161

K
Kamenický, Jiří, 69
Kamenický, Marian, 69
Kameničtí (= KEYBCS2, CP895), znaková sada, 69–71
Kasprzak, Jan, 146
Katz, Phillip Walter, 159
kharóšthí (खरोष्ठी), staroindické písmo, 21
klíčovací značka, zarovnání barev při soutisku, 78
klínopisné číslice, 20
kód, předpis pro převod informace, 17, 24, 97, 99–109, 115–123, 158–160
kódování dat, 96–109, 158–160
    aritmetické, 158
    nerovnoměrné (= prefixové), 99–109, 158–160
    rovnoměrné (= blokové), 97–99
KOI-8 (Код Обмена Информацией), znaková sada, 69–71
kombinace, neuspořádaný výběr prvků, 37
kombinatorika, obor matematiky, 37
kombinovaná parita, 111–113, 115
kombinovaný znak ve standardu Unicode, 72
komerční software, 176–178
komprese (= komprimace), 80, 83, 84, 100, 103, 104, 148–150, 152–162, 165, 168
    adaptivní, 154, 159
    asymetrická, 154
    fyzická, 154
    logická, 154
    neadaptivní, 154
    neztrátová, 84, 103, 104, 149, 150, 153, 155–159, 161
    off-line, 153
    on-line, 153
    symetrická, 154
    záporná, 155, 156
    ztrátová, 84, 104, 149, 150, 153, 159–161
komprese (= komprimace) dat, 198
kompresní algoritmy, 155–161
    slovníkové, 159
    statistické, 158, 159
    transformační, 159–161
kompresní poměr, 84, 154–157, 159, 160
kompresní software, 159, 161, 162
konec řádku (= EOL, CR/LF), řídicí znak, 67, 140, 141, 147
konec souboru (= EOT), řídicí znak, 140, 141
kontrolní číslice, metoda zabezpečení dat, 113–115
kontrolní součet, detekční kód, 112–115
konverze souborového formátu, 151, 152
koptské číslice, 18
Kotělnikov, Vladimir Alexandrovič (Котельников, Владимир Александрович), 83
Kraft, Leon Gordon, 102
Kraftova nerovnost, 102
kryptoanalýza, vědní disciplína, 22, 124
kryptografie (= šifrování), vědní disciplína, 22, 112, 123–128, 180, 198
kryptologie, vědní disciplína, 124
kvantování signálu, diskretizace oboru hodnot, 83
kybernetika, vědní disciplína, 34
kyberstalking, 192
kyberšikana, 193

L
Leibniz, Gottfried Wilhelm von, 24
Lempel, Abraham (אברהם למפל), 159
Linux, operační systém, 42, 67, 141, 144, 152, 165, 167–169, 176
Little Endian, varianta vícebajtového kódu, 73, 76

logaritmus, funkce, 38
logická komprese, 154
logický pixel, 79
lokalizace adresáře, 164
LZMA, kompresní algoritmus, 159
LZW, kompresní algoritmus, 159, 161

M
MacCE, znaková sada, 70, 71
macOS, operační systém, 42, 71, 141, 167, 169, 190
malware, 181–186
    metody detekce, 188, 189
mantisa racionálního čísla, 58–66
    normalizace, 59, 60, 63–65
    zaokrouhlení, 59, 60, 63–65
Markov, Andrej Andrejevič (Марков, Андрей Андреевич), 159
Markovův řetězec, popis diskrétních náhodných procesů, 159
mayské číslice, 20
McMillan, Brockway, 102
McMillanova věta, 102
MD5, hashovací funkce, 113, 125
MDBX, souborový formát, 148
měření množství informace, 38–40, *viz také* entropie
metody filtrace mailů
    podle obsahu dopisu, 195
    podle způsobu dopravy, 194
Microsoft, společnost, 82, 138, 144, 145, 147, 148, 150, 151, 176, 177, 182, 189, 190, 198, 199
MKV, souborový formát, 151
monoalfabetická šifra, 124
monochromatický obraz, 80, 155, 157
Morse, Samuel Finley Breese, 97
Morseova abeceda, kód používaný v telegrafii, 97
MP3, souborový formát, 104, 150
MP4, souborový formát, 151
MPEG, sada kompresních algoritmů, 161
MPG, souborový formát, 151
MS-DOS, operační systém, 69, 71, 164, 166, 167, 181
multipartitní virus, 182

N
NaN, výsledek aritmetické operace, 62
národní znak, 69–71, 73, 76, 109, 143, 144, 146, 147, 152
násobnost chyby, 110–113, 115, 116, 118
nat, jednotka informace, 38
neadaptivní komprese, 154
Nečas, David, 146
nejnižší řád binárního čísla (= LSB), 49, 50, 73
nejvyšší řád binárního čísla (= MSB), 49, 50, 73
nekonečno, výsledek aritmetické operace, 61, 62
nepoziční číselná soustava, 17–19, 21
nerovnoměrné kódování, 99–109, 158–160
neutrální (= achromatické) barvy, 77
neztrátová komprese, 84, 103, 104, 149, 150, 153, 155–159, 161
NTFS, souborový systém, 168, 169
nula, 17, 20–22, 30, 51–55, 61
    číslo (žádné množství), 17, 20–22, 30, 51–55, 61
    poziční číslice, 17, 20–22, 30, 51, 53
Nyquist, Harry Theodor, 83, 84
Nyquistova-Shannonova vzorkovací věta (= Whittakerův--Nyquistův-Kotělnikovův-Shannonův teorém), 83, 84

O
obousměrná šifra, 126–128
obrazový objekt, element vektorového obrazu, 81
ODF, třída souborových formátů, 147, 148, 159
ODG, souborový formát, 150
ODP, souborový formát, 148, 150
ODS, souborový formát, 148
odstíny šedi, 80, 81, 157, 159
ODT, souborový formát, 146–148
OEM, typ softwarové licence, 177
off-line komprese, 153
oficiální jméno znaku ve standardu Unicode, 72
ochrana proti hoaxu, 195
oktalová (= osmičková) číselná soustava, 24
on-line komprese, 153
opakovací kód, 115, 116
operační systém, 10, 24, 36, 42, 67, 69, 71, 141, 144, 146, 147, 150, 152, 153
oprava chyby, 110–112, 115–123
optimální (= nejkratší) kód, 97–109
osmičková číselná soustava, 24
otevřený (= open source) software, 176, 177
otevřený (= svobodný) souborový formát, 142, 146–151
otisk zprávy, *viz* hashovací funkce

P
paměťová náročnost nekomprimovaných dat
    rastrových obrazových, 80–82, 148
    zvukových, 84
pangram, text obsahující všechna písmena abecedy, 144
parita, detekční kód, 110–113, 115
Pasco, Richard Clark, 158
Pavlov, Igor Viktorovič (Павлов, Игорь Викторович), 159
Pazdziora, Jan, 146
PC Latin 2 (= IBM Latin 2, CP852), znaková sada, 69–71
PCX, souborový formát, 148, 161
PDF, souborový formát, 142, 146, 148, 150, 159
perfektní kódy, 120
permutace, uspořádání prvků množiny, 37
pharming, 192
phishing, 192
Pisánský, Leonardo, *viz* Fibonacci
pixel, element rastrového obrazu, 79–81, 155, 156, 160
PNG, souborový formát, 142, 149, 159, 161
počítač, 9–15
    ergonomie práce, 11–13
    programové vybavení, 10
    technické vybavení, 10
    údržba, 13–15
počítačová kriminalita, 175–196
    obrana proti malwaru, 188–190
    obrana proti nelegálnímu použití software, 178

počítačové pirátství, 176–178
počítačový červ, 184
počítačový virus, 181–184
    kategorie, 181, 182
    obranná strategie, 182, 183
    princip činnosti, 182
    projevy, 183
podtečení, 51, 61
polyadická číselná soustava, 23, 24
polyalfabetická šifra, 124
polynomiální zápis čísla, 23, 25, 26
pomocný bit při zaokrouhlování mantisy, 59, 60, 63–65
poškození z opakovaného namáhání, 11
poziční číselná soustava, 20–28
poziční zápis čísla, 23
PPT, souborový formát, 150
PPTX, souborový formát, 147, 148, 150
pravé barvy, 80, 156, 157
prefix číselné soustavy, 27
prefixové kódování, 99–109
proprietární (= closed source) software, 176, 177
prostý (= čistý) text, 143
přenositelnost souborového formátu, 146
přenosový řetězec, 96
přesnost zobrazení racionálních čísel, 57, 58, 64, 65
přetečení, neplatný výsledek aritmetické operace, 50, 51, 54, 55
převody mezi číselnými soustavami, 25–27
přímý číselný kód, 52, 52, 53, 56, 57
přípona názvu (= rozšíření) souboru, 144, 146
PS, souborový formát, 148, 149
PSD, souborový formát, 145

Q
Q faktor, parametr algoritmu JPEG, 160
Qín Jiǔsháo (= Daogu, 秦九韶, 道古), 26

R
RAR, souborový formát, 162
rastrová grafika, 77, 79–82, 148, 149, 155, 194
rastrování vektorového obrazu, 81
RAW, třída souborových formátů, 149
realizace jevu, 36
redundance kódu, 97–100, 102–105
ReiserFS, souborový systém, 168
rekonstrukce signálu, 84, 95
relativní cesta, způsob lokalizace adresáře, 164
reprezentace dat v paměti (= vnitřní), 49–85, 105–109
    numerických, 49–66
    obrazových, 76–82
    textových, 66–76, 105–109
    zvukových, 82–85
reprezentace dat v souborech (= vnější), 137–170, *viz také* datový formát; souborový formát
režijní informace, metadata v archivním souboru, 154
RGB, barevný model, 77, 78, 78, 80
RGBA, barevný model, 78, 80, 149
RIFF souborový formát, 151
Rissanen, Jorma Johannes, 158
Rivest, Ronald Linn, 126
RLE, kompresní algoritmus, 155–157, 161
rootkit, 185, 186
rovnoměrné kódování, 97–99
rozlišení (= hustota) rastrového obrazu, 79–81
rozpoznávání souborového formátu, 146, 147
rozšířený text, 144, 146
RSA, šifrovací algoritmus, 126
RTF, souborový formát, 145, 147, 151
rušení signálu, 94, 95, 110

Ř
řádová čárka, oddělovač částí čísla, 25, 49–51, 57–61, 65
řecká abeceda, *viz* alfabeta
řecké číslice, 18
řetězový e-mail, 195
řídicí znak, 67, 139–141
římské číslice, 18

S
samoopravný kód, 115–123
Satyanarayanan, Mahadev, 201
scanner, 188
sektor, nejmenší adresovatelný blok vnější paměti, 165
sekvence MIDI, digitální popis signálu, 84, 85
semilogaritmický tvar racionálního čísla, 58, 61, 62
SHA, hashovací funkce, 113, 126
Shamir, Adi (עדי שמיר), 126
shannon (Sh), jednotka informace, 38
Shannon, Claude Elwood, 35, 36, 46, 83, 84, 103, 158
Shannonův-Fanův kód, 103, 105, 158
shareware, forma distribuce softwaru, 177, 178
signál, fyzikální veličina, 93–95
skrytý bit, nejvyšší platný bit mantisy, 59, 61
slabika (= bajt), adresovatelná buňka paměti, 50, 57
slovníkový útok, 180, 194
SMP, doplňková rovina standardu Unicode, 76
sniffing, 180
sociální inženýrství, 191, 192
software, programové vybavení počítače, 10, 181–190
    formy distribuce softwaru, 177, 178
    kategorie, 176
    typy licencí k softwaru, 10, 177
soubor, vymezení pojmu, 163, 164
souborový formát, 142–152, 156, 162
    binární (= netextový), 142, 146–151
    hlavička, 146–148, 156
    konverze, 151, 152
    otevřený (= svobodný), 142, 146–151
    pro archivy, 162
    pro dokumenty, 147, 148
    pro multimédia, 150, 151
    pro prezentace, 150
    pro rastrovou grafiku, 148, 149
    pro tabulky, 148
    pro vektorovou grafiku, 149, 150
    přenositelnost, 146
    rozpoznávání, 146, 147
    textový, 142, 146–149, 151

uzavřený (= proprietární), 142, 149
souborový systém, 163–169
pro macOS, 169
pro MS-DOS a Windows, 166–168
pro Unix/Linux, 168, 169
souborový virus, 181
soukromý klíč, 126–128
spam, 193–195
spamovací robot, 193
spojitý (= analogový) signál, 93–95
spoofing, 191
spyware, 185–187
standardizované formáty racionálních čísel, 59–65
Staré pověsti české, 106
statický signál, 93
steganografie, vědní disciplína, 124
střední délka kódového slova, 97, 100, 102–105
substituce, náhrada znaků zprávy jinými znaky, 124
subtraktivní barevný model, 78, 80
SVG, souborový formát, 148, 149, 151
svobodný (= free) software, 176, 177
SWF, souborový formát, 150
symetrická komprese, 154
symetrické šifrování, 126–128
syndrom karpálního tunelu, 11
syndrom kubitálního tunelu, 11
syndrom slova, kontrolní posloupnost, 119
synchronizace dat, 200, 201

Š
šedesátková číselná soustava, 20
šestnáctková číselná soustava, 24
šifra, 124–128
šifrovací klíč, 125–128
šifrování, *viz* kryptografie

T
tabulátor (= HT), řídicí znak, 67, 141
teorie informace, 35–42
test kontrolního součtu, 188
textový formát dat, 138, 140–142, 146
TGZ (= TAR.GZ), souborový formát, 162
TIFF, souborový formát, 149, 161
transpozice, změna pořadí znaků zprávy, 124
trialware, forma distribuce softwaru, 178
Tridgell, Andrew, 201
trojský kůň, 184, 185
klasifikace, 185
TS, souborový formát, 151
TXT, souborový formát, 147, 151

U
UCS (= ISO/IEC 10646), univerzální znaková sada, 71–76
údaje, vymezení pojmu, 33
údržba počítače, 13–15
UFS, souborový systém, 168
umělá inteligence, 195
unární (= jedničková) číselná soustava, 23
Unicode, univerzální znaková sada, 67, 71–76
Unix, třída operačních systémů, 24, 69, 141, 144, 146, 147, 162, 164, 168, 169, 169, 177, 190, 200
UTF, metoda kódování znaků, 73–76, 105–109
útok hrubou silou, 128, 180
uzavřený (= proprietární) souborový formát, 142, 149

V
variace, uspořádaný výběr prvků, 37
vektorová grafika, 77, 81, 82, 148–151, 159
veřejný klíč, 126–128
VFAT, souborový systém, 167, 169
vícebajtová znaková sada, 71–76, 105, 109
vlastní (= částečná) informace o realizaci jevu, 38, 39
vnější (= disková) fragmentace, 165
vnitřní (= souborová) fragmentace, 153, 165
VOB, souborový formát, 151
volné dílo, forma distribuce softwaru, 177
vyhledávání signatur, 188
východoarabské číslice (= arqām hindiyyah, أرقام هندية), 22
vzorkovací frekvence, 83, 84, 95
vzorkování zvukového signálu, 83, 84, 94, 95, 150

W
warez, autorská díla užívaná v rozporu s autorským zákonem, 178
WAV, souborový formát, 150, 151
Weaver, Warren, 103
WebM, souborový formát, 151
WebP, souborový formát, 149
Welch, Terry Archer, 159
Whittaker, Edmund Taylor, 83
Wiener, Norbert, 34
Windows, operační systém, 42, 67, 71, 141, 144, 146, 150, 162, 164–169, 176, 177, 185, 186, 199
Windows-1250 (= CP1250), znaková sada, 70, 71
WMA, souborový formát, 104, 150
WMF, souborový formát, 150
WMZ, souborový formát, 150

X
XCF, souborový formát, 145
XFS, souborový systém, 168
XLS, souborový formát, 145, 148, 150, 151
XLSM, souborový formát, 145, 148
XLSX, souborový formát, 145, 147, 148, 150, 151
XML, souborový formát, 147, 148

Y
YC_BC_R (= YCbCr, Y'CbCr, YCC), barevný model, 160
Yì Jīng (= Kniha proměn, 易經), 24

Z
zabezpečení dat, 95, 96, 110–128
proti nedokonalostem při přenosu, 95, 96, 110–123
proti neoprávněným osobám, 123–128
synchronizace, 200, 201
zálohování a archivace, 196–200
základ číselné soustavy, 17, 19, 20, 23–28
Zákon č. 40/2009 Sb. (= trestní zákoník), 176
Zákon č. 212/2000 Sb. (= autorský zákon), 176

zálohování dat, 196–200
    typy záloh, 197, 198
západoarabské číslice (= ghubār, غبار), 22
záporná komprese, 155, 156
záporné číslo, 17, 24, 50, 52–58
zaváděcí virus, 182
ZFS, souborový systém, 168
Zhū Shìjié (= Zhū Shìjié Hànqīng, Zhū Sōngtíng, 朱世傑漢卿, 朱松庭), 26
ZIP, souborový formát, 103, 162
Ziv, Ja'akov (יעקב זיו), 159
zkreslení signálu, 94, 95
zlomková část čísla, 23, 25, 26, 50, 57–61
znak v počítači, 66–76, 139–141
    národní, 69–71, 73, 76, 143, 144, 146, 147, 152
    řídicí, 67, 139–141
    zobrazitelný, 67–69, 73, 139, 140
znaková sada (= znakový kód), 67–76, 97, 105–109, 138–140, 143
znalosti, vymezení pojmu, 34
znaménkový bit, 50, 52, 53, 56, 59, 60, 62, 63, 65
zobrazitelný znak, 67–69, 73, 139, 140
ztrátová komprese, 84, 104, 149, 150, 153, 159–161
zvuk, mechanické vlnění, 82

Ž

žurnálování, vytváření záznamů o prováděné činnosti, 163, 168, 169, 200

Pavel Haluza, Jiří Rybička, Tomáš Hála

Úvod do informatiky

Obálka Lucie Pacáková
Sazba Pavel Haluza
Technický redaktor Tomáš Hála
Korektury KONVOJ, spol. s r. o.

Vydalo nakladatelství KONVOJ, spol. s r. o.,
jako svou 304. publikaci. Druhé, rozšířené vydání.
219 stran. Brno 2024

Adresa nakladatelství:
KONVOJ spol. s r. o., Berkova 22, 612 00 Brno
E-mail konvoj@konvoj.cz, http://www.konvoj.cz

ISBN 978-80-7302-189-4