MAGAZÍN
KATEDRY INFORMATIKY

číslo 2 | prosinec 2014 Univerzita Palackého v Olomouci

Úvodní slovo

Vážení čtenáři,

dostává se Vám do rukou již druhé číslo Magazínu katedry informatiky. V tomto vydání bychom Vás rádi seznámili s obory, které u nás lze studovat. Dále Vám Petr Osička představí zajímavou metodu analýzy dat, kterou se na katedře zabýváme a která umožňuje odhalit nové informace a souvislosti v tabulkových datech. Já jsem pro Vás připravil malé ohlédnutí za historií unixových systémů zabývající se okolnostmi, které stály za jejich rozšířením. Závěrem pro Vás máme opět nachystaný i jeden zajímavý problém pro pobavení. Tentokrát se týká toho, jak správně rozeznat barvu ponožek.

Dovolte mi, abych Vám jménem redakce Magazínu i jménem katedry popřál příjemné prožití vánočních svátků a úspěšný start do nového roku.

Přeji vám příjemné čtení.

Petr Krajča

redaktor Magazínu Katedry informatiky Univerzity Palackého v Olomouci

Obsah

2 KATEDRA
Naše studijní obory
Přehled oborů, které se naší katedře dají v současné době studovat

3 VÝZKUM
Rozklady binárních matic
Zajímavé téma výzkumu na naší katedře

6 TÉMA
45 let unixů
Ohlednutí za historií unixových systémů

9 HÁDANKA
Ponožkový kvíz
Problém z oblasti interaktivního dokazování

Naše studijní obory

Posláním univerzity, a tedy i naší katedry, je objevování nových poznatků a předávání již existujících poznatků studentům. S některými výzkumnými tématy, kterým se lidé na katedře věnují, jsme již čtenáře magazínu seznámili. V tomto článku chceme informovat o naší pedagogické činnosti a představit obory, které lze na naší katedře studovat.

Informatika

Informatiku lze studovat v prezenční formě bakalářského, navazujícího magisterského programu a také v doktorském programu. Hlavní důraz při studiu je kladen na pochopení principů informatiky. Studenti získají solidní teoretické základy i praktické dovednosti. V bakalářské etapě se studenti seznámí se základy matematických disciplín a teoretické informatiky, principy návrhu a analýzy algoritmů. Součástí studia je 21 programovacích kurzů, během nichž se studenti naučí programovat v několika programovacích jazycích a používat různá programovací paradigmata. Poznají principy současných počítačových technologií, jako jsou databázové, informační nebo operační systémy, a síťové a webové technologie.

Absolventi navazujícího magisterského studia jsou vysoce kvalifikovanými odborníky schopnými samostatné tvůrčí práce. Mají znalosti z náročných oblastí informatiky, například analýzy dat, umělé inteligence, kryptografie, konstrukce programovacích jazyků a jejich překladačů, moderních operačních systémů.

Aplikovaná informatika

Aplikovanou informatiku lze studovat v bakalářském programu v prezenční i kombinované formě. Oproti oboru Informatika se v menší míře vyučují teoretické předměty a větší důraz je kladen na předměty praktického zaměření.

Podotkněme, že studenti Aplikované informatiky nejsou o možnost studovat teoretické předměty ochuzeni, stejně tak jako nejsou ochuzeni studenti Informatiky o možnost absolvovat předměty praktického rázu. Díky kreditovému systému si je mohou zapsat jako nepovinný předmět. Rozdíly mezi obory jsou dány především skladbou povinných předmětů.

Absolventi bakalářského programu Aplikované informatiky mohou pokračovat v navazujícím magisterském studiu oboru Informatika po složení rozdílových zkoušek.

Bioinformatika

Bioinformatiku lze studovat v prezenčním bakalářském a navazujícím magisterském studiu. Jde o mladý perspektivní obor, který se zabývá metodami pro zpracování dat získaných v biologii, biochemii, medicíně a dalších oborech. Takových dat je díky prudkému rozvoji biotechnologií více a více a poptávka po odbornících schopných například analyzovat genomická data roste. Obor je zabezpečován katedrou informatiky a katedrou biochemie. Studenti jsou v přímém kontaktu s pracovníky předního mezinárodního biotechnologického centra – Centra regionu Haná.

Učitelství

Bakalářský obor Informatika pro vzdělávání a navazující magisterský obor Učitelství výpočetní techniky pro střední školy vychovává učitele informatiky, kteří v počítači vidí víc než pouhý stroj na psaní textu, tvorbu tabulek a vyhledávání informací na internetu.

Rozklady binárních matic

Petr Osička

Analýza dat je významným tématem v informatice a naše katedra se mu řadu let intenzivně věnuje. Jedna z metod, kterou se na katedře zabýváme, je analýza tabulkových dat pomocí rozkladů binárních matic. Metoda spočívá v tom, že data vyjádříme pomocí binární matice a hledáme dvě binární matice, jejichž součin dá matici původní. Na první pohled zajímavá úloha má řadu praktických aplikací, umožňuje například odhalit zajímavé vztahy v datech nebo zredukovat množství dat, se kterými je nutné pracovat.

Metoda rozkladu binární matice pracuje s tabulkovými daty. Tato data mají tvar tabulky, jejíž řádky odpovídají objektům a jejíž sloupce odpovídají vlastnostem objektů, kterým říkáme atributy. Pomocí křížku na odpovídajícím místě tabulky značíme fakt, že objekt má daný atribut. V případě, že objekt daný atribut nemá, necháme místo v tabulce prázdné. V následujícím příkladu reprezentují řádky tabulky pacienty a sloupce tabulky různé symptomy vyskytující se u pacientů.

	horečka	kašel	vyrážka	bolest hlavy	zvracení
Jan	×				×
Markéta	×		×		×
Mirek		×		×	
Martin	×		×		
Radim		×		×	

Tabulková data se vyskytují poměrně často, lze je získat přímým pozorováním, například v předchozím příkladu zaznamenáváním výsledků vyšetření u lékaře; dalším příkladem je tzv. nákupní košík, kdy objekty odpovídají zákazníkům a sloupce kupovanému zboží, tyto informace často obchody sbírají pomocí různých klubových kartiček. Tabulková data lze ale také získat transformací z jiného tvaru, například databázové tabulky.

Data z tabulky si lze představit jako množinu I uspořádaných dvojic řádek–sloupec, pro které je v tabulce na místě odpovídajícím dánemu řádku a danému sloupci křížek. Tabulku z předchozího příkladu tedy lze reprezentovat jako $\{(\text{Jan, horečka}), (\text{Jan, zvracení}), (\text{Markéta, horečka}), \dots\}$.

Formálně je I *binární relace* mezi množinou řádků a množinou sloupců. S relacemi lze provádět celou řadu operací, pro účely tohoto článku je ovšem důležitá operace *skládání binárních relací*. Uvažme tedy relaci A mezi množinami X a Z a relaci B mezi množinami Z a Y. Výsledkem složení relací A a B je relace $A \circ B$ mezi množinami X a Y daná vztahem

$$(x, y) \in A \circ B \text{ právě když existuje } z \in Z \text{ takové, že } (x, z) \in A \text{ a současně } (z, y) \in B.$$

Vyjádřeno slovně, x a y jsou spolu ve výsledné relaci, právě když existuje prostředník z, který je v relaci A s prvkem x a v relaci B s prvkem y.

Podívejme se například na tabulky A a B z následující strany. Tabulka A zachycuje vztah mezi pacienty a onemocněními (jakou nemocí pacient trpí), tabulka B vztah mezi onemocněními a symptomy (jaké symptomy jsou projevem daného onemocnění). Jejich složením dostaneme vztah mezi pacienty a symptomy z tabulky na této straně. Například na řádku Markéta ve sloupci vyrážka je křížek, protože existuje onemocnění, konkrétně neštovice, takové, že Markéta tímto onemocněním trpí (v tabulce A je na řádku Markéta ve sloupci neštovice křížek) a současně je vyrážka projevem neštovic (v tabulce B je na řádku neštovice ve sloupci vyrážka křížek). Podobně

bychom se dopátrali nějakého onemocnění, které spojuje tabulky A a B, pro každý křížek v původní tabulce.

Vstupem výpočtu rozkladu binární matice je relace I mezi množinami X a Y. Tedy například vztah mezi pacienty a symptomy z našeho příkladu. Cílem je nalézt množinu Z, relaci A mezi množinami X a Z, a relaci B mezi množinami Z a Y tak, aby platilo $I = A \circ B$ a Z měla co nejméně prvků.

Pro analýzu dat má rozklad poměrně zajímavý význam. Jak jsme již uvedli, relace I reprezentuje vztah mezi objekty z množiny X a atributy z množiny Y. Můžeme říci, že každý objekt z X je popsán pomocí atributů z Y. Například v tabulce na začátku článku popisujeme pacienty pomocí symptomů. Cílem rozkladu je nalezení množiny Z skrytých atributů, takzvaných *faktorů*, pomocí nichž lze objekty popsat. Cílem je, aby popis byl úsporný, tj. aby faktorů bylo co nejméně. *Chceme tedy vyjádřit obsah tabulky co nejstručněji a umožnit tak snazší pochopení toho, jaká data v tabulce jsou*. Relace A reprezentuje vztah mezi objekty a faktory, tedy popisuje objekty pomocí nově nalezených atributů. Relace B potom zachycuje vztah mezi faktory a původními atributy. Pokud je faktor z v relaci s původním atributem y, říkáme, že y je manifestací faktoru z.

V případě dat o pacientech lze nalezené faktory, tedy prvky množiny Z, interpretovat jako kandidáty na onemocnění. Pro každého pacienta relace A říká, jaké onemocnění daný pacient může mít, zatímco relace B pro každé onemocnění ukazuje, jakými symptomy je manifestováno. Podle definice skládání relací pak pacient x má symptom y (relace I), pokud existuje onemocnění z takové, že pacient x má onemocnění z (relace A) a y je jedním ze symptomů onemocnění z (relace B). Rozložíme-li relaci reprezentovanou tabulkou ze začátku článku, obdržíme faktory, které můžeme interpretovat jako neštovice, nachlazení a virózu. Interpretaci (a jména) faktorů lze odhadnout z relace B. Relace A a B lze zachytit pomocí následujících tabulek.

B	horečka	kašel	vyrážka	bolest hlavy	zvracení
neštovice	×		×		
nachlazení		×		×	
viroza	×				×

A	neštovice	nachlazení	viróza
Jan			×
Markéta	×		×
Mirek		×	
Martin	×		
Radim		×	

Maticová formulace

Pokud řádky a sloupce tabulkových dat očíslujeme, tedy množina objektů bude $X = \{1, 2, \ldots n\}$ a množina atributů $Y = \{1, 2, \ldots m\}$, lze si tabulku (a tedy i binární relaci I mezi X a Y) představit jako binární matici, tj. matici obsahující pouze 0 a 1. Tuto matici dostaneme z tabulky tak, že vynecháme popisy řádků a sloupců, křížky nahradíme jedničkami a prázdná místa nulami. V matici je pak na i-tém řádku v j-tém sloupci 1, pokud jsou i-tý objekt a j-tý atribut v relaci I. Následující matice je reprezentací dat o pacientech.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Součinem binární matice A o n řádcích a p sloupcích a binární matice B o p řádcích m sloupcích je binární matice $A \cdot B$ o n řádcích a m sloupcích, která je dána

$$(A \cdot B)_{ij} = \bigvee_{t=1,\ldots,p} A_{it} B_{tj},$$

kde A_{ij} je prvek matice A na i-tém řádku a j-tém sloupci a $\vee$ odpovídá funkci maxima. Pokud se předchozí vztah podíváme blíže, můžeme si všimnout, že násobení matic odpovídá skládaní relací. Pokud vynásobíme matici odpovídající relaci A a matici odpovídající relaci B, dostaneme matici odpovídající výsledku jejich složení. Stačí si uvědomit, že maximum z binárních hodnot je 1, právě když alespoň jedna z těchto hodnot je 1, a že aritmetické násobení binárních hodnot odpovídá logické spojce „a“. V maticové formulaci je tedy úloha rozkladu definovávana následovně. Pro vstupní matici I nalezněte matice A a B tak, že $I = A \cdot B$ a vnitřní rozměr

součinu (tedy počet sloupců matice A a počet řádků matice B) je co nejmenší. Součin v našem příkladě je tedy

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Algoritmy

Spočítat optimální rozklad je NP-težký problém, tj. neznáme žádný algoritmus, který by dokázal v polynomickém čase (vzhledem k velikosti matice) spočítat rozklad s minimálním počtem faktorů. Existují nicméně algoritmy aproximační, které se snaží optimálnímu rozkladu přiblížit.

Na naší katedře vzniklo několik algoritmů pro výpočet rozkladu, včetně toho v současné době nejrychlejšího a nejkvalitnějšího algoritmu GreEss. Všechny algoritmy jsou založeny na teoretickém vhledu do rozkladů. Ve zbytku kapitoly tyto algoritmy a princip jejich fungování vysvětlíme.

Algoritmy jsou založeny na „geometrickém“ pohledu na rozklad. Vztah, kterým definujeme součin matic, se dá upravit tak, že součin matic A a B dostaneme jako po souřadnicích aplikované maximum z matic získaných maticovým součinem sloupce z matice A a řádku z matice B. Vždy násobíme i-tý sloupec matice A s i-tým řádkem matice B, jako v následujícím příkladu.

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \vee \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Pokud z matice získané součinem sloupce a řádku vybereme pouze řádky a sloupce obsahující nějakou jedničku, dostanem matici plnou jedniček. Tuto matici budeme označovat jako obdélník. K nalezení rozkladu matice I musíme najít takovou množinu obdélníků obsažených v I (tedy podmatic I plných jedniček), že každá jednička z I je obsažena alespoň v jednom obdélníku (řekneme, že je pokrytá obdélníkem).

Algoritmus *GreCon* (zkratka názvu Greedy Concepts, zde slovo concepts označuje obdélníky) vygenerujeme množinu všech maximálních obdélníků obsažených v I, a těmito obdélníky budeme postupně pokrývat jedničky v matici I. Obdélníky jsou maximální v tom smyslu, že po přidání jakéhokoliv řádku nebo sloupce by už obdélník obsahoval nulu. To, že stačí generovat pouze maximální obdélníky (a ne všechny obdélníky), plyne z toho, že jedničky pokryté libovolným obdélníkem, lze pokrýt i libovolným obdélníkem, který původní obdélník obsahuje. Přirozeně, každý obdélník je obsažen v nějakém maximálním obdélníku. V každé iteraci algoritmu vybereme ten obdélník, který pokrývá největší množství doposud nepokrytých jedniček, a k rozkladu ho přidáme. To opakujeme tak dlouho, dokud nepokryjeme všechny jedničky v matici I.

Nevýhodou algoritmu GreCon je jeho velká časová složitost. Počet maximálních obdélníků obsažených v matici může být velký, v nejhorším případě až exponenciální (vzhledem k rozměrům matice) a GreCon je všechny opakovaně prochází. Algoritmus *GreConD* (Greedy Concept on Demand) tomuto předchází tím, že generuje obdélník pokrývající velké množství doposud nepokrytých jedniček na požádání. Při jeho výpočtu začíná algoritmus s prázdným obdélníkem. K němu poté opakovaně přidává další sloupce. Pro přidání vybere vždy ten sloupec, který maximalizuje počet nově pokrytých jedniček. Podotkněmě, že po přidání nového sloupce je nutné odstranit některé řádky z obdélníku, abychom pořád pořád měli matici plnou jedniček (nový sloupec může obsahovat nulu na nevhodném místě). Autory algoritmů GreCon a GreConD jsou Radim Bělohlávek a Vilém Vychodil. Jejich přesný popis lze nalézt v článku, na který odkazujeme na konci článku.

Algoritmus *GreEss* (Greedy Essential elements) nejdříve najde ve vstupní matici tzv. *esenciální* prvky. Je to podmnožina jedniček v matici, jejímž pokrytím libovolnou množinou maximálních obdélníků pokryjeme všechny jedničky v matici. GreEss poté tyto esenciální prvky využívá ke generování obdélníků na požádání po-

dobným způsobem jako GreConD. Algoritmus vytvořili Radim Bělohlávek a Martin Trnečka.

Pro ilustraci uvedeme příklad běhu algoritmu GreCon na matici

$$I = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Každý z obdélníků obsažený v matici I můžeme jednoznačně charakterizovat seznamem řádků a sloupců, které jej tvoří. Například, pokud je obdélník charakterizován množinou řádků $\{2, 4\}$ a množinou sloupců $\{1, 5\}$, pak má jedničky na souřadicích $(2, 1), (2, 5), (4, 1)$ a $(4, 5)$. Všechny maximální obdélníky obsažené v I jsou v následující tabulce.

Obdélník	řádky	sloupce
C_1	$\{1, 2, 3\}$	$\{1, 2\}$
C_2	$\{2\}$	$\{1, 4, 5\}$
C_3	$\{3\}$	$\{1, 2, 3, 4\}$
C_4	$\{2, 4\}$	$\{1, 5\}$
C_5	$\{1, 2, 3, 4\}$	$\{1\}$
C_6	$\emptyset$	$\{1, 2, 3, 4, 5\}$

Postupně vybereme obdélníky C_1, C_4, C_3, pomocí níž sestavíme rozklad. Každý obdélník představuje jeden sloupec v matici A a jeden řádek v matici B. Ve sloupci matice A je každý řádek, který charakterizuje daný obdélník nastaven na 1 (ostatní na 0), v řádku matice B je každý sloupec, který charakterizuje daný obdélník nastaven na 1 (ostatní na 0). Ve výsledku tedy dostáváme rozklad

$$A \cdot B = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Další výzkumná témata

Tématům souvisejícím s rozklady matic se na katedře věnuje celá výzkumná skupina. Kromě již zmiňovaných autorů algoritmů mezi do ní patří Jan Outrata, který použil dekompozici pro předzpracování dat klasifikačních metod. Klasifikace je důležitá úloha, která má široké využití. Předzpracování vstupu pomocí rozkladu významně vylepšilo schopnost správné klasifikace u širokého spektra metod (například rozhodovacích stromů.)

Úlohu rozkladu binární matice lze zobecnit na matice, které obsahují místo nul a jedniček prvky vybrané z jiné škály, například z intervalu $[0, 1]$. Neplatí pak, že objekt daný attribut buď má nebo nemá, ale může jej mít v nějakém stupni. Například prvek na řádku Markéta ve sloupci bolest hlavy v tabulce z našeho příkladu by v tomto případě udával stupeň, ve kterém Markétu bolí hlava. Data podobného typu lze získat i z dotazníků, kdy řádky odpovídají dotazujícím a sloupce jednotlivým otázkám. Škála pak zachycuje možné odpovědi (například: spokojen, částečně spokojen, ...). Rozklad matice má u dat tohoto typu stejný význam jako u binárních dat, pouze je potřeba použít obecnějšího matematického aparátu. Teoretickými základy takových rozkladů se zabývají Jan Konečný a Eduard Bartl.

Rozklady binárních matic na katedře informatiky

- Na výzkum rozkladů matic a souvisejích oblastí členové katedry získali již několik grantů GAČR – hlavní grantové agentury pro základní výzkum. Posledním je grant R. Bělohlávka, který bude řešen v letech 2015-2017.
- Čtenářům, které téma zaujalo, doporučuji přečíst si článek Belohlavek R., Vychodil V.: *Discovery of optimal factors in binary data via a novel method of matrix decomposition*. Journal of Computer and System Sciences 76(1)(2010), 3-20.

45 let unixů

Petr Krajča

Operační systém Unix je s námi už 45 let a i když to tak na první pohled nevypadá, jedná se o nejrozšířenější operační systém vůbec. V tomto článku nechci rozpoutávat další z mnoha diskuzí nad tím, který operační systém je lepší, jestli Windows, Linux nebo třeba OS X, ale rád bych se věnoval příběhu Unixu, který si za svou existenci prošel několika zajímavými zvraty, které zapříčinily, že nějaký unix najdete dnes téměř všude, od superpočítačů až po mobilní telefony. Možná je to trochu paradoxní, ale za úspěchem Unixu, nestály jen jeho dobré technické vlastnosti.

MULTICS

Vše to začalo před padesáti lety, kdy v roce 1965 rozjela firma General Electric (GE) společně s MIT a Bellovými laboratořemi (Bell Labs) projekt počítače GE-645 a operačního systému MULTICS– Multiplexed Information and Computer Services. Tento systém měl v bostonské oblasti nabízet počítačové služby „ze zásuvky“, stejně jako je k dispozici telefon či elektřina. Jednalo se tak s trochou nadsázky o předchůdce toho, čemu dnes někteří říkají cloud. Tento projekt byl z obchodního hlediska velmi dobře promyšlený, v bostonské oblasti sídlí hned několik prestižních univerzit, takže existovala nezanedbatelná poptávka po výpočetním výkonu. V tomto případě ale obchodní řešení předběhlo řešení technické.

Vytvořený systém přinesl řadu na svou dobu přelomových vlastností. Systém virtuální paměti, který nerozlišoval mezi daty v paměti a soubory, možnost měnit hardware a jeho konfiguraci za běhu, souborový systém se stromovou strukturou, návrh, který od začátku počítal se souběžnou prací více uživatelů, takže obsahoval správu oprávnění a mnohé další zajímavé vlastnosti. Všechny tyto vlastnosti si ale vybraly svou daň. Výsledný systém byl příliš komplikovaný a výkon neodpovídal očekávání a nakonec na pokrytí bostonské oblasti službami tohoto počítače nedošlo. Bellovy laboratoře se proto rozhodly z projektu vycouvat.

UNIX

Unix, je ukázkovým příkladem toho, že chybami se člověk učí. Ken Thompson, jeden z vývojářů Bellových laboratoří, který pracoval na MULTICSu, našel vyřazený počítač PDP-7 a začal si pro něj psát zjednodušenou verzi MULTICSu, která později dostala název UNICS, UNiplexed Information and Computing Service, později Unix. Jelikož Unix upoutal pozornost dalších kolegů, časem vznikly již oficiální verze pro mnohem výkonnější počítač PDP-11, který patřil mezi nejrozšířenější počítače té doby.

První verze Unixu byly stejně jako většina operačních systémů té doby napsány v assembleru, což značně limitovalo přenositelnost na jiné počítače. V roce 1972 jej proto Ken Thompson a Dennis Ritchie přepsali do jazyka C, který pro tento účel Ritchie vytvořil z jazyka BCPL.

To byl jeden ze tří hlavních důvodů, proč se Unix rozšířil. Dalším významným faktorem bylo, že telekomunikační společnost AT&T, které Bellovy laboratoře patřily, měla kvůli v té době probíhajícím antimonopolním sporům zakázané podnikat v jiné oblasti než v telekomunikacích a soudní nařízení dokonce požadovalo, aby AT&T poskytlo své technologie, které nesouvisely s telekomunikacemi, každému, kdo si o to požádá. Třetí důvod, proč Unix získal svou popularitu, byl fakt, že řada amerických univerzit používala počítače z rodiny PDP, pro které představoval ve srovnání s tím, co dodával

Ken Thompson (sedící), Dennis Ritchie (stojící), počítač PDP-11 (v pozadí) (Foto: Peter Hamer)

originální výrobce, Unix pokročilejší operační systém. Navíc Unix byl dodáván včetně okomentovaných zdrojových kódů, tedy se dal použít nejen ve výuce, ale nabízel velký potenciál pro postupné rozšiřovaní.

Jedny z prvních rozšíření vznikaly na Kalifornské univerzitě v Berkeley a nesly název Berkeley Software Distribution (BSD). Rozšíření navržená v Berkeley kromě běžných nástrojů, jako jsou překladače, textové editory, apod. zahrnovala i síťový protokol TCP/IP, který se později stal základem dnešního Internetu. Postupně vznikla celá řada unixů a dalo by se říct, že každá počítačová firma měla svůj unix, o kterém by se daly vést dlouhé historické diskuze, my se ale nyní zaměříme na unix, který není Unixem.

Terminologická poznámka: V tomto textu slovem „Unix" označuji původní operační systém z Bellových laboratoří, případně jeho přímé nástupce a slovem „unix" libovolný operační systém unixového typu.

GNU's not Unix

V roce 1983 přišel Richard M. Stallman s projektem GNU, což znamená „GNU's Not Unix". Mělo se jednat o svobodnou implementaci Unixu. Jednou z hlavních motivací pro vznik takového operačního systému byla snaha výrobců nedodávat zdrojové kódy k softwaru a klást různá omezení na to, jak se softwarem zacházet,

Richard M. Stallman (Foto: Anders Brenna, Teknisk Beta)

což začalo být v rozporu se zvyklostmi, které do té doby víceméně platily. Stallman formuloval čtyři svobody, které musí svobodný software splňovat. Uživatel musí mít:

1. svobodu používat program k libovolnému účelu,
2. svobodu studovat, jak program funguje a měnit to,
3. svobodu distribuovat kopie programu,
4. svobodu vylepšovat a měnit program a dát tyto změny veřejnosti.

Aby bylo možné tyto svobody zaručit, je potřeba mít přístup ke zdrojovým kódům. Tyto svobody a požadavky Stallman vtělil do Manifesta GNU a licence GPL (General Public License), pod kterou je software v rámci projektu GNU distribuován.

Sám Stallman pro projekt GNU naprogramoval textový editor Emacs, překladač GCC, debugger GDB a řadu dalších nástrojů. Koncem osmdesátých let byly dokončeny všechny hlavní komponenty, chybělo ale jádro operačního systému. Původní plán upravit jádro z operačního systému BSD nevyšel a Stallman proto navrhl použít jádro Mach vyvíjené na Carnegie Mellon University, ze kterého vzniklo jádro GNU Hurd. Vývoj se však protahoval a dodnes nebyl dokončen. V roce 1991 se ale našlo řešení, které málokdo čekal.

Linux

Příběh Linuxu začal podobně nevinně jako příběh Unixu. V roce 1991 si mladý student Helsinské univerzity, Linus Torvalds, začal psát svůj vlastní operační systém. Chtěl tehdy vytvořit unixový systém pro svůj domácí počítač s procesorem Intel 386, který by byl zdar-

Linus Torvalds (Foto: LINUXMAG.com)

ma, protože unixy tehdejší doby byly cenově nedostupné. V té době používal operační systém MINIX navržený pro výuku, který však měl (nejen z pedagogických důvodů) řadu omezení a bylo jej možné používat pouze pro výukové účely. O tom, že Torvalds ze začátku neměl větší ambice svědčí i oznámení o vzniku nového operačního systému, které začínalo:

```
Hello everybody out there using minix -
I'm doing a (free) operating system (just a
hobby, won't be big and professional like
gnu) for 386(486) AT clones...
```

Svůj vývoj začal na platformě MINIX s využitím nástrojů z GNU. Za začátku byl Linux distribuován pod vlastní licencí, která zakazovala komerční použití, spolu s nástroji z projektu GNU. Časem se ale Torvalds dal přesvědčit a přelicencoval Linux pod licenci GPL, a tak umožnil vznik a další rozvoj (eko-)systému GNU/Linux.

Torvalds k vývoji Linuxu vždy přistupoval jako praktik a praktická řešení dostávala často přednost před teoreticky nebo ideově čistějšími řešeními. O tom svědčí i dnes téměř legendární výměna názorů s prof. Tanenbaumem, autorem operačního systému MINIX, ohledně architektury Linuxu, ve které prof. Tanenbaum prohlásil, že kdyby mu takový operační systém Torvalds donesl, tak za něj nedostane ani zápočet.

Podobně se Torvalds například nebál používat proprietární (nesvobodný) nástroj BitKeeper pro správu zdrojových kódu Linuxu. Richard Stallman poukazoval na to, že to není příliš dobrý nápad a že s nesvobodným programem mohou být v budoucnu potíže. V roce 2005 mu dal autor BitKeeperu zapravdu, když začal diktovat uživatelům různá omezení, když zrušil bezplatnou verzi programu nebo když začal vyžadovat, aby se uživatelé neúčastnily vývoje podobných nástrojů. To se ukázalo jako nepřekonatelná bariéra pro vývojáře Linuxu. Torvalds proto musel hledat vhodnou náhradu. Jelikož nic lepšího neexistovalo, vytvořil vlastní nástroj—Git, který je dnes považován za jeden ze standardních nástrojů pro správu zdrojových kódu.

Tím, že GNU/Linux poskytuje prostředí plnohodnotného operačního systému, které je možné libovolně upravovat dle aktuálních potřeb, stal se z něj výborný základ pro další modifikace a speciální operační systémy. Linux proto dnes najdete téměř všude—od extrémů jako jsou superpočítače z TOP 500, kde má Linux více než tříčtvrtinový podíl, až po mobilní telefony s Androidem a různé „krabičky" s domácím wifi-routery, či diskovými poli.

BSD, OS X a ti další

Historie Unixu je však mnohem košatější. V tomto článku jsem opomněl řadu dalších zajímavých Unixů, ať už komerčních či nekomerčních. Za zmínku by určitě stál třeba systém FreeBSD, který se stal základem operačního systému OS X nebo herních konzolí PlayStation, ale to by vydalo na další, podobně objemný, článek. Možná se k tomuto tématu na stránkách našeho magazínu ještě vrátíme.

1. **Unix Wars** – v průběhu osmdesátých a devadesátých let existovalo velké množství vzájemně ne úplně kompatibilních implementací Unixu, každý výrobce si často dodal své vlastní rozšíření, což vedlo ke sporům o to, co je opravdový Unix, se kterým by ostatní měli být kompatibilní. Nejčastěji se vedly spory mezi implementacemi vycházejícími přímo z Unixu a implementacemi vycházejícími z BSD, které dominovaly v akademické sféře. Vzniklo několik více či méně úspěšných iniciativ, které se pokoušely roztříštěný svět unixů sjednotit. Nejúspěšnější z tohoto pohledu bylo vytvoření standardu POSIX, který se snaží většina unixů dodržovat. Dokonce i Microsoft Windows NT měly až do verze 2000 (včetně) certifikaci, že podporují POSIX.
2. **Y2K38** – Kolem roku 2038 se dá očekávat, že nás média zahrnou dávkou katastrofických zpráv o blížícím se kolapsu civilizace stejně, jako nás zahrnovaly podobnými zprávami koncem devadesátých let v souvislosti s problémem Y2K. Původní autoři Unixu totiž přišli s myšlenkou uchovávat čas jako celé číslo představující počet sekund od půlnoci 1. ledna 1970 a tento způsob reprezentace času převzala řada dalších programů. 19. ledna 2038 však dojde k tomu, že hodnoty se už nevlezou do 32 bitových celočíselných typů a programy a datové formáty, které pracující s touto reprezentací času přestanou pracovat korektně.

Ponožkový kvíz

Petr Osička

Dnešní hádanka je z oblasti interaktivních dokazovacích systémů. K jejímu vyřešení není zapotřeby hlubokých znalostí matematiky, postačí elementární znalost pravděpodobnosti.

Uvažujme dvě osoby, Alici a Boba. Alice je barvoslepá a Bob má různobarevné ponožky. Bob chce o tom, že má různobarevné ponožky, Alici přesvědčit. Ta je ovšem barvoslepá, a ponožky jí připadají stejné. Úkolem je vymyslet scénář, ve kterém figurují jenom Alice, Bob a předměty, které mají lidé běžně u sebe, ve kterém Bob může přesvědčit Alici o různobarevnosti svých ponožek s pravděpodobností libovolně se blížící jedné. Správné řešení hádanky můžete najít na webové stránce magazínu.

Redakce:
Petr Krajča, Petr Osička
Katedra informatiky PřF UP
17. listopadu 12
771 46 Olomouc

web: www.inf.upol.cz/magazin
email: magazin@inf.upol.cz
atom: http://www.inf.upol.cz/atom/magazin
telefon: 585634701
GPS: 49.591870, 17.262336