Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

ŠTÁTNY PEDAGOGICKÝ ÚSTAV
NATIONAL INSTITUTE FOR EDUCATION

Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Operačné systémy a počítačové siete

Predmet: Operačné systémy a počítačové siete

Línia: Vlastný odborový kontext informatiky a informatickej výchovy

Operačný program VZDELÁVANIE

EURÓPSKA ÚNIA
Európsky sociálny fond

Európska únia
Európsky sociálny fond

Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov ES

Operačné systémy a počítačové siete

Garant predmetu:

RNDr. Peter Tomcsányi
KZVI FMFI UK, Bratislava
tomcsanyi@fmph.uniba.sk

Autori:

RNDr. Peter Tomcsányi
KZVI FMFI UK, Bratislava
prof. Ing. Miloš Šrámek, PhD., KAI FMFI UK, Bratislava,
Ing. Peter Palúch, PhD.
KIS FRI, Žilinská univerzita

Identifikácia modulu

Aktivita projektu:	1.3 Ďalšie vzdelávanie kvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ
Línia aktivity:	Vlastný odborový kontext informatiky a informatickej výchovy
Predmet:	Operačné systémy a počítačové siete

Zaradenie modulu

Modul je zaradený v línii Informatika. Nasleduje po základných moduloch predmetu Programovanie a prináša pohľad na hlbšie vrstvy softvéru a hardvéru počítačov nielen z hľadiska skúseného používateľa, ale aj z hľadiska programátora. Pojednáva o teórii a praxi operačných systémov a počítačových sietí. V praktických príkladoch prezentuje dva najrozšírenejšie operačné systémy pracovných staníc - Microsoft Windows a Linux. Svojím hlbším pohľadom do operačného systému Linux zároveň nadväzuje na modul 3DG1, ktorý opisoval Linux z hľadiska bežného používateľa.

Abstrakt modulu

Modul pozostáva zo 4 tematických jednotiek. V prvej časti sa oboznámime s operačným systémom vo všeobecnosti. Táto časť opakuje, prehlbuje a aktualizuje vedomosti z teórie operačných systémov. Ukážky v tejto časti sú zamerané na operačný systém Windows a zahŕňajú aj programovanie. V druhej časti sa zoznámime s vnútornou štruktúrou operačného systému Linux. Dozvieme sa, ako sa teoretické objekty a konštrukcie z prvej časti modulu prakticky implementujú v Linuxe a ako ich prakticky spravovať. V tretej časti sa budeme venovať počítačovým sieťam a internetovým protokolom. Zopakujeme si a prehĺbime vedomosti o základných princípoch sietí, technológiách lokálnych sietí a internetových protokoloch. Štvrtá časť je praktický návod ako si zapojiť vlastnú sieť. Poskytuje praktický návod, ako vybrať zariadenia pre sieť, ako ich zapojiť a hlavne ako nastaviť pracovné stanice tak, aby správne pracovali v danej sieti. Počas vzdelávania v rámci tohto modulu si účastník zopakuje, zaktualizuje a prehĺbi vedomosti o operačných systémoch a počítačových sieťach.

Obsah

Úvod

Cieľom modulu je zopakovanie, aktualizovanie a prehĺbenie vedomostí, ktoré už majú účastníci vzdelávania o operačných systémoch a počítačových sieťach. Modul podáva jednak teoretické vedomosti v tejto oblasti a súčasnú terminológiu, jednak praktické vedomosti z oblasti nastavovania parametrov dvoch najrozšírenejších operačných systémov osobných počítačov.

Teória operačných systémov je doplnená pohľadom programátora – kompletnými ukážkami programov, ktoré používajú danú vlastnosť operačného systému. Ukážky sú k dispozícii v e-learningovom prostredí a sú naprogramované v objektovom Pascale v prostredí Lazarus, vďaka čomu sú použiteľné ako vo Windows, tak aj v Linuxe.

Teória počítačových sietí je doplnená praktickými ukážkami. Obsahuje odkazy na niekoľko rozširujúcich materiálov, ktoré sú k dispozícii v e-learningovom prostredí.

Výučba modulu bude v počítačovej učebni s nainštalovaným Windows, Lazarusom verzie 0.9.28.2 alebo novšej a programom Wireshark verzie 1.2.3 alebo novším. Účastníci budú mať k dispozícii Live DVD Linuxu (distribúcia Ubuntu 9.04) so všetkými potrebnými programami.

Vstupné vedomosti

Požadované prerekvizity

Predpokladáme úspešne absolvované moduly 3DG1, 3DG2, 3Prog1 až 3Prog4.

Predpokladané vstupné vedomosti, skúsenosti a zručnosti

Predpokladáme, že účastník modulu:

- vie pracovať s OS Windows, so súbormi a s prehliadačom,
- vie sa orientovať v grafickom prostredí Linuxu (distribúcia Ubuntu), vie pracovať so súbormi v grafickom prostredí,
- rozumie základným programátorským pojmom a dokáže vytvoriť jednoduché programy v Delphi/Lazaruse a Imagine Logu
- rozumie zápisom čísiel v dvojkovej sústave

Operačné systémy

Čo je to operačný systém

Operačný systém je základným program každého súčasného počítača, bez ktorého by sme si nevedeli predstaviť jeho použitie.

Každý používateľ má skúsenosť s operačným systémom, ale nie je ľahké podať jeho definíciu. Je to aj preto, že operačný systém vykonáva dve rôzne úlohy.

Operačný systém ako vrstva softvéru (pohľad zvonka)

Obrázok 1: Vrstvový model počítača podľa [5]

Autor učebnice [5] opisuje moderný počítač z hľadiska programátora ako zariadenie pozostávajúce zo šiestich vrstiev (viď obrázok 1). Najspodnejšia je „skutočný“ hardvér - logické obvody, najvyššou vrstvou je vyšší programovací jazyk. Táto predstava zhruba zodpovedá aj historickému vývoju počítačov. Prvé elektronické počítače (napríklad ENIAC) mali len čistý hardvér, teda logické obvody implementované pomocou elektrónok, a elektronické moduly sa prepájali do funkčných blokov pomocou káblov a prepínačov. Neskôr sa vyvinuli ďalšie dve úrovne (mikroarchitektúra a strojový kód), ktoré spolu s úrovňou logických obvodov tvoria procesor súčasných počítačov.

Procesor sa dá programovať v programovacom jazyku, ktorý nazývame **strojový kód**. Jeho inštrukcie sú navrhnuté v prvom rade so zreteľom na ich ľahké vykonanie počítačom. Preto programovanie v takomto jazyku nie je pre človeka príjemné. Navyše všetky vstupné a výstupné operácie treba veľmi prácne programovať. Preto už dávno vznikla knižnica (pod)programov, v ktorej boli naprogramované technicky náročné úseky kódu ako napríklad zápis jedného sektora na disk alebo tlač riadku textu na tlačiareň. Postupne sa z nich vyvinula ucelená nová vrstva - operačný systém. Podprogramy operačného systému nazývame **služby** operačného systému alebo aj systémové volania (system calls).

Za prvý vyšší programovací jazyk sa považuje jazyk FORTRAN, ktorý uviedla firma IBM v roku 1956.

Časom sa operačné systémy stali zložitejšími, definovali nové objekty špecifické pre operačný systém (súbor, semafor, stránka, proces ...) a implementovali služby, ktoré programátorovi umožňujú s týmito objektmi pracovať.

Pri pohľade zvonka je operačný systém **služobník** pre vyššie úrovne - jeho prvoradou úlohou je uľahčiť programátorom ich prácu.

Nad úrovňou operačného systému už môže programátor programovať v jazyku, ktorý je zmiešaný z inštrukcií strojového kódu a z volaní operačného systému. To je už jednoduchšie, ale stále dosť zložité. Preto sa časom vyvinuli aj ďalšie vrstvy - vrstva assemblerového jazyka umožňuje zapisovať strojový kód v človeku prístupnejšom textovom zápise avšak ide stále o tie isté jednoduché inštrukcie ako v strojovom kóde. Až vrstva vyššieho programovacieho jazyka dáva programátorovi k dispozícii jazyk, ktorý viac zohľadňuje jeho potreby než vlastnosti a možnosti konkrétneho procesora.

Operačný systém ako správca prostriedkov (pohľad zvnútra)

Druhý pohľad na operačný systém ho charakterizuje ako správcu všetkých súčastí počítača. Moderné počítače pozostávajú z množstva komponentov - procesorov, pamätí, diskov, tlačiarní a pod., o ktoré súperí niekoľko bežiacich programov. V terminológii operačných systémov používame pre bežiaci program pojem **proces** a komponent počítača nazývame **prostriedok**. Teda procesy potrebujú pre svoje správne vykonanie prostriedky. V moderných operačných systémoch beží naraz viaceré procesy a tie súperia o prostriedky. Operačný systém musí rozhodnúť, ako a ktorý prostriedok pridelí ktorému procesu. V závislosti od druhu prostriedku je možné prostriedok buď v jednom čase prideliť len jednému procesu, a toto pridelenie postupne meniť (napríklad procesor, tlačiareň), alebo je možné prideliť z daného prostriedku istú časť jednému procesu a ďalšiu časť inému procesu (napríklad pamäť alebo diskový priestor).

Z pohľadu zvnútra je teda operačný systém **vládca** počítača, prideľuje vyšším vrstvám prostriedky a stará sa o správne fungovanie celého počítača.

Z tohto pohľadu členíme operačný systém na štyri časti. Každá z nich spravuje istý

špecifický druh prostriedkov. Nazývame ich správa procesov, správa pamäte, správa zariadení a správa súborov. V takomto členení sa budeme ďalej v tejto kapitole zoznamovať s jednotlivými časťami operačných systémov.

Skutočný operačný systém plní naraz obe úlohy. Každá správa má časti, ktoré spravujú daný druh prostriedkov (napr. správa pamäte spravuje pamäť). Na druhej strane však na to, aby procesy mohli používať dané prostriedky, musí implementovať služby na ich pridelenie, uvoľnenie, zdieľanie a ochranu (napríklad služby pridelenia a uvoľnenia úseku pamäte konkrétnej veľkosti).

Zadanie 1

Odborníci a v poslednom čase aj laickí používatelia poznajú viaceré operačné systémy po mene. Ktoré z týchto operačných systémov poznáte? Viete kedy sa používali a na aký typ počítačov či iných zariadení boli určené?

OS/360, CP/M, MS-DOS, Mac OS X, Symbian, Windows CE, Windows 7, Solaris, Linux.

Správa procesov

Vieme už, že proces je bežiaci program. Podľa jedného programu môže bežať viac procesov, môžu pracovať s rôznymi dátami a byť v rôznom štádiu rozpracovania. Do správy procesov zahŕňame časti operačného systému, ktoré sa starajú o vznik procesov (spustenie programov), ich život, zánik ako aj komunikáciu medzi procesmi.

Vzťah medzi programom a procesom je taký, ako medzi kuchárskym receptom a varením podľa neho (aj keď kuchárske recepty bývajú menej exaktné než počítačové programy)

Keď v jednom počítači je rozbehnutých naraz viac procesov, hovoríme tomu multiprogramming alebo multitasking. Ak má pritom počítač len jeden procesor, tak musí správa procesov zabezpečovať ich zdanlivo súčasný beh tak, že procesor prideľuje raz jednému, raz zase inému procesoru. Hovoríme tomu pseudoparalelizmus. Ak má počítač k dispozícii viac procesorov, tak niekoľko procesov môže bežať skutočne paralelne. Operačný systém potom zadeľuje viac procesorov medzi viac procesov. Nazývame to multiprocessing.

Zadanie 2

Kedy a prečo ako vlastne vznikol multiprogramming? Operačný systém, v ktorom existuje naraz len jeden proces by bol predsa oveľa jednoduchší.

Prečo používame multiprogramming aj v osobných počítačoch, ktoré majú slúžiť jednému človeku?

Z hľadiska používateľa proces vznikne tak, že spustí nejaký program - dvojkliknutím na jeho ikonu alebo napísaním jeho mena v príkazovom riadku. Aby však toto mohol používateľ urobiť, musí už bežať nejaký iný proces, ktorý sleduje kliknutia myšou na ikony alebo číta riadky príkazov a vykonáva ich. Preto pri bližšom technickom pohľade vidíme, že proces vzniká tak, že jeden už bežiaci proces požiada operačný systém o vznik ďalšieho procesu. To môže spraviť buď ako reakciu na vstup od používateľa alebo aj samostatne keď na vykonanie istej činnosti potrebuje vykonať iný program.

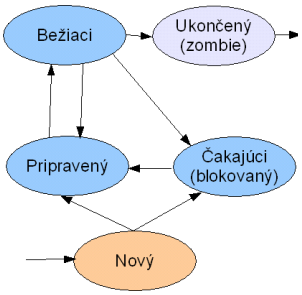
Z toho nám vyplýva že operačný systém musí obsahovať službu "vytvor proces". V ďalšom si ukážeme jej detaily vo Windows, ako funguje takáto služba v Linuxe, sa dozviete v ďalšej kapitole.

Procesy vytvárajú stromovú hierarchiu, v ktorej má proces svojho rodiča - ten proces, ktorý ho vytvoril. Na začiatku preto musí operačný systém vytvoriť špeciálnym spôsobom (aspoň) jeden proces, ktorý nemá rodiča a ktorý je teda koreňom stromu procesov. Okrem tohto koreňového procesu má každý proces svoj rodičovský proces.

Ukončovací kód býva jedno celé číslo. Úspech sa zvykne indikovať nulou, nenulový kód sa považuje za chybu.

Proces môže zaniknúť z niekoľkých dôvodov. Môže dokončiť svoju prácu a vyvolať službu "ukonči tento proces". To je normálne ukončenie, pri ňom môže zároveň odovzdať ukončovací kód (exit code), ktorým oznámi rodičovskému procesu, či proces skončil úspešne. Proces však môže skončiť aj preto, že počas jeho práce vznikne chyba, ktorú nedokáže sám spracovať a operačný systém ho ukončí (chybové

ukončenie). Obvykle je možný aj prípad keď jeden proces požiada operačný systém o ukončenie iného procesu (ukončenie zvonku).



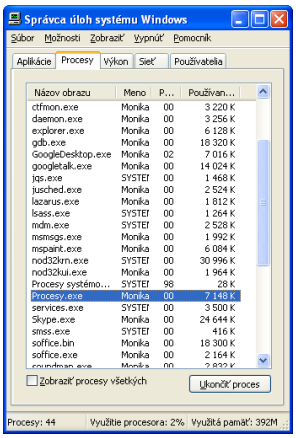
Obrázok 2: Životný cyklus procesu - mierne upravené podľa [6]

Počas svojej existencie je procesu prideľovaný procesor aby mohol vykonávať svoje inštrukcie (stav Bežiaci). Keďže v operačnom systéme väčšinou existuje viac procesov než procesorov, existujú aj situácie, keď proces, ktorý je pripravený bežať, nemá k dispozícii procesor a musí preto počkať, kým mu ho operačný systém pridelí (stav Pripravený). Niekedy však proces čaká na niečo z vlastnej iniciatívy - napríklad čaká na kliknutie myšou alebo iný vstup od používateľa, na správu od iného procesu. Vo všeobecnosti hovorím, že proces čaká na nejakú udalosť. Kým tá udalosť nenastane, proces čaká (stav Čakajúci) a uvoľní tým vlastne procesor pre beh ostatných procesov. K slovu sa dostane vtedy, keď daná udalosť nastane.

Procesy môžu v niektorých operačných systémoch končiť len zo stavu bežiaci, v iných môžu skončiť v ľubovoľnom stave. V niektorých operačných systémoch sa ešte definuje aj stav ukončený (zombie). V tomto stave už proces vlastne neexistuje, zostal len jeho popis v operačnom systéme. Tento stav slúži hlavne na to, aby rodičovský proces mal prístup k ukončovaciemu kódu procesu, ktorý už skončil.

Všetky ukážkové programy k tejto kapitole nájdete v e-learningovom prostredí pod nadpisom Programy k 1. kapitole (**Programy1.zip**). Program Procesy je špecifický pre Windows, jeho Linuxová verzia je uvedená v ďalšej kapitole.

Počas experimentovania s programom Procesy, sledujte zoznam bežiacich procesov v Správcovi úloh systému Windows (Task manager), ktorý vyvoláte stlačením Ctrl+Alt+Del.



Obrázok 3: Správca úloh systému Windows

Zadanie 3

V adresári Procesy je ukážka programu, ktorý na špecifickú činnosť (editovanie obrázku) vyvolá iný program a čaká na jeho ukončenie. Otvorte Procesy.lpi v Lazaruse, spustite ho a stlačte Edituj – spustí sa Skicár, keď v ňom zmeníte obrázok a skončíte, program Procesy zobrazí zeditovaný obrázok. Preštudujte si text programu. Čo sa zobrazuje v textovom poli pod tlačidlom Edituj?

Vlákna (Threads)

Procesy tak ako sme ich opísali existujú v každom modernom operačnom systéme. V ukážke z predošlej podkapitoly vidíme, že vytvorenie procesu je pomerne časovo náročná operácia - operačný systém musí prečítať do pamäte nový vykonateľný súbor, prideliť mu pamäť a vykonať veľa administratívy. Každý proces má svoju vlastnú virtuálnu pamäť, svoje otvorené súbory a mnohé ďalšie nastavenia. Procesy môžu zdieľať dáta len pomocou súborov alebo pomocou posielania správ.

Programátori často potrebujú naprogramovať vykonávanie viacerých paralelných činností v rámci jedného programu. Napríklad súčasné textové editory chcú nielen reagovať na stláčané klávesy, ale zároveň aj kontrolovať pravopis a pri listovaní stránok zobrazovať aj správne zmenšené náhľady obrázkov. Ak by všetko bolo naprogramované v jednom procese, tak by jedna činnosť by musela počkať na skončenie ďalšej, tak by mal používateľ dojem, že všetko ide akosi pomaly a zadrháva sa. Preto je vhodné opísané tri činnosti vykonať paralelne tak, aby všetky tri mohli veľmi úzko spolupracovať a tvorili spolu jeden program.

Z týchto potrieb sa zrodilo vlákno (thread). V moderných operačných systémoch je program vykonávaný vláknami. Proces je potom obalom pre viac vlákien, ktoré majú spoločnú pamäť, otvorené súbory a niektoré ďalšie nastavenia. Vlákna jedného procesu môžu zdieľať dáta oveľa jednoduchšie než procesy a to v globálnych premenných aj v otvorených súboroch. S každým procesom vznikne automaticky jedno vlákno - základné vlákno procesu. Ak proces nevytvorí ďalšie vlákna, tak bude mať len jedno a bude sa vykonávať postupne tak, ako je to popísané v základných kurzoch programovania. Programátor však môže vytvoriť aj niekoľko vlákien, každému z nich dať vykonávať svoj kód (procedúru v zmysle terminológie Pascalu). To nazývame **multithreading**.

Vlákna môžeme použiť v rámci jedného procesu keď potrebujeme robiť činnosti, pri ktorých treba rýchlo reagovať na udalosti (stláčanie klávesov pri písaní textu) ale zároveň sa vyskytujú aj výpočtovo náročnejšie úlohy (kontrola pravopisu prekresľovanie obrázkov, tlač celého dokumentu), ktoré je dobré robiť "na pozadí"

tak, aby sa výrazne nespomalila hlavná činnosť.

Vlákna majú využitie aj v serverových aplikáciách keď má jeden program spracovávať požiadavky prichádzajúce po sieti (napríklad odoslať html stránku). Pri príchode požiadavky je preto jednoduché vytvoriť ďalšie vlákno, ktoré bežiac paralelne s ostatnými vybaví požiadavku konkrétneho klienta a potom skončí.

Donedávna by sme mohli vo výpočte použitia multithreadingu skončiť a vznikol by dojem, že multithreading je špecialita, ktorú potrebuje prakticky ovládnuť len mizivé percento programátorov. S rozšírením viacjadrových procesorov doslova na stôl každého používateľa sa však situácia podstatne zmenila. Viacjadrový procesor sa totiž z hľadiska programátora správa ako viac procesorov. A využiť potenciál viac procesorov v jednom programe sa dá len pomocou multithreadingu.

Aj projektový notebook má dvojjadrový procesor.

Pretože vlákna zdieľajú spoločnú pamäť, pri behu je nutné synchronizovať ich vykonávanie, inak môže vzniknúť situácia, ktorú nazývame alebo súbeh alebo časová závislosť.

Súbeh (časová závislosť, race condition) je situácia keď výsledok práce viacerých spolupracujúcich procesov alebo vlákien závisí od poradia, v akom im bol prideľovaný procesor. Niekedy je výsledok správny, inokedy úplne zlý. (Ale v praxi je skôr 10000-krát správny a raz, práve keď to najmenej potrebujeme, program úplne zlyhá). Súbeh ako aj spôsob obrany proti nemu si ukážeme na konkrétnom programe.

Súbeh môže vzniknúť vždy, keď sa paralelne vykonávajú procesy alebo vlákna, ktoré zdieľajú dáta. Môže nastať aj v programoch Imagine Loga. Jeho podrobnejšie vysvetlenie je v [7].

Zadanie 4

V adresári Mandelbrot je ukážka programu, ktorý kreslí známy fraktál – Mandelbrotovu množinu [9]. Otvorte Mand.lpi (podľa možnosti na počítači s viacjadrovým procesorom). Spustite ho a stlačte tlačidlo Štart. Po čase sa nakreslí obrázok a vypíše sa čas výpočtu v milisekundách. Zmeňte konštantu **pocVlakien** na 2, spustite znovu. Aký bol čas výpočtu?

Zakomentujte prvý a posledný riadok procedúry tMandVlakno.zoberDalsiBod a sledujte, čo sa stane.

Všetky ukážkové programy k tejto kapitole nájdete v e-learningovom prostredí pod nadpisom Programy k 1. Kapitole (**Programy1.zip**). Program Mand funguje bez zmien vo Windows aj v Linuxe.

V programe Mand môžete meniť veľkosť jeho okna, tým zmeníte aj veľkosť kresleného obrázku.

Správa pamäte

Úlohou správy pamäte je prideľovať procesom pamäť. To znamená umožniť procesom žiadať pamäť, uvoľňovať ju a niekedy aj zdieľať medzi niekoľkými procesmi. Zároveň správa pamäte aj chráni pamäť, teda zabezpečuje, aby jeden proces nemohol zasahovať do pamäte iného procesu.

Virtuálna pamäť

Skutočnú operačnú pamäť počítača (fyzická pamäť) si možno predstaviť ako pole bajtov maximálnej veľkosti 2^N bajtov pre nejaké vhodné N. Programátor si ju môže predstaviť ako pole `array [0..2^N -1] of byte`. Inštrukcie strojového kódu adresujú operandy v pamäti tak, že uvedú adresu, na ktorej sa operand nachádza.

Niektoré procesory delia pamäť na iné najmenšie adresovateľné jednotky, najrozšírenejšie procesory umožňujú adresovať jednotlivé bajty.

Keď operačný systém prideľuje každému procesu miesto priamo v reálnej pamäti a každý proces pri svojom vykonávaní adresuje priamo reálnu pamäť, prináša to dva problémy:

- Problém relokácie: Programy buď musia dopredu vedieť na akých adresách sa budú nachádzať keď pobežia alebo ich musí operačný systém modifikovať pred uložením do pamäte tak, aby sa správne odvolávali na tie adresy, ktoré im boli pridelené.
- Problém ochrany: V strojovom kóde môže každý proces vykonať inštrukciu, ktorá adresuje akúkoľvek adresu v reálnej pamäti. To znamená, že môže zasiahnuť aj do oblastí pamäte, ktoré mu nepatria.

V histórii operačných systémov sa prideľovanie priamo reálnej pamäti používalo a problém relokácie a ochrany sa riešil rôznymi, často komplikovanými, kombináciami hardvéru a softvéru, ktoré často neriešili niektorý z problémov úplne dôsledne.

Moderné operačné systémy používajú pri správe pamäte prístup, ktorý nazývame **virtuálna pamäť**. To znamená, že každému procesu sa vytvorí ilúzia jeho vlastnej pamäte. Každý proces potom adresuje len do nej a ľubovoľná adresa, ktorú použije jeho program sa chápe ako virtuálna adresa do virtuálnej pamäte procesu. Virtuálna adresa sa nezmení ani pri umiestnení pamäte procesu na inú adresu vo fyzickej pamäti. Každá virtuálna adresa adresuje len do virtuálnej pamäte daného procesu, preto nemôže v žiadnom prípade zasiahnuť virtuálnu pamäť iného procesu. Teda virtuálna pamäť rieši problém relokácie aj ochrany. Navyše môže byť veľkosť virtuálnej pamäte nezávislá od veľkosti fyzickej pamäte. Môže byť od nej menšia, ale aj väčšia.

Moderné procesory majú vo svojej architektúre zahrnutú podporu pre virtuálnu pamäť. Preto súčasné operačné systémy ju takmer všetky aj implementujú.

Na to, aby mohol operačný systém implementovať virtuálnu pamäť, musí mať podporu v procesore lebo v procese virtualizácie prepočítavajú virtuálne adresy na fyzické. Keby sa tento prepočet vykonával na úrovni operačného systému, bolo by treba na vykonanie každého jedného adresovania do pamäte vykonať niekoľko strojových inštrukcií. To by znamenalo niekoľkonásobné spomalenie programu.

Stránkovanie

Stránkovanie je v súčasnosti najrozšírenejší prístup k virtualizácii pamäte, preto si ho opíšeme do detailov.

Práve možnosť vytvorenia väčšej virtuálnej pamäte než je fyzická pamäť je historickým dôvodom na prvé implementácie virtuálnej pamäte. Prvé počítače mali oproti dnešným veľmi málo pamäte. Preto sa často stávalo, že samotný program na riešenie istej úlohy by bol väčší než celá operačná pamäť počítača. Programátori mali na riešenie tejto situácie vypracované rôzne postupy.

Viac prechodov používali aj prvé kompilátory. Viacprechodové kompilátory sa bežne používajú dodnes, aj keď už nie z dôvodu nedostatku pamäte.

Napríklad rozdelili program na menšie časti, ktoré spracovávali dáta postupne. Hovoríme o viacprechodových programoch. Prvý prechod čítal pôvodné dáta, čiastočne ich spracoval a vytvoril nové dáta a uvoľnil pamäť. Druhý prechod prečítal dáta prvého prechodu a vytvoril svoje dáta až posledný prechod vytvoril výsledné spracované dáta.

Ktoré procedúry patria do ktorého prekryvného modulu, to musel navrhnúť programátor. Keď to navrhol zle a celky sa príliš často v pamäti vymieňali, tak celý program zahlcoval počítač čítaním svojich častí z disku a na jeho vykonávanie zostalo málo času.

Keď sa nedal rozdeliť program na po sebe idúce prechody, mohol sa použiť iný prístup - programátor rozdelil program na funkčné celky (prekryvné moduly). Každý celok obsahoval isté konkrétne procedúry a funkcie a prečítal sa do pamäte len keď bolo potrebné vykonať niektorú jeho časť. Keď niektorý celok nebolo treba, mohol byť v pamäti nahradený iným. Tento spôsob sa nazýval prekrývanie (overlay). Bol vstavaný aj v populárnom kompilátore Turbo Pascal.

Z myšlienky prekrývania bol už len malý krok k virtualizácii pamäte metódou, ktorú dnes nazývame stránkovanie. Ten malý krok spočíval v automatizácii. Pri stránkovaní sa virtuálna pamäť procesu rozdelí bez zásahu programátora na veľa rovnako veľkých úsekov, ktoré nazývame stránky. Počas behu programu budú vo fyzickej pamäti umiestnené len tie stránky, ktoré program potrebuje. To sú tie, ktoré práve adresuje alebo ich adresoval v blízkej minulosti. Ostatné stránky sú uložené na disku. Prepočet virtuálnych adries na fyzické adresy pre stránky, ktoré sú v pamäti, vykonáva samotný procesor. Používa pritom tabuľku stránok, ktorú mu pripravil operačný systém. Keď má ale procesor prepočítať adresu, ktorá odkazuje na stránku mimo pamäte, tak musí požiadať o spoluprácu operačný systém. Ten musí nájsť v pamäti miesto pre stránku, ktorá sa práve adresuje (možno pritom uloží nejakú inú stránku z pamäte na disk) a prečíta ju z disku do pamäte. Potom upraví stránkovú tabuľku a procesor už dokáže prepočítať adresu a program môže bežať ďalej.

Celý tento proces je pre programátora bežiaceho programu neviditeľný - nemusí sa vôbec starať o to, ako a kedy bude ktorá časť pamäte jeho programu naozaj v pamäti a ktorá nie. Autor učebnice [6] má pre stránkovanie pekné prirovnanie: Predstavme si, že futbalový klub chce pokryť svoj štadión umelým trávnikom, ale nemá dosť peňazí pre pokrytie celej plochy štadióna. Preto kúpili menej umelého trávnika a rozstrihali ho na štvorce 2x2 metre. Na celú plochu štadióna potom

nakreslili štvorce 2x2 metre, ktoré určujú možné miesta pre štvorce umelej trávy. Počas zápasu potom rýchlo presúvajú štvorce umelej trávy tak, aby boli práve na tých miestach kde prebieha hra. Teda na tie štvorce štadióna, kde sa nachádza nejaký hráč či rozhodca alebo sa pohybuje lopta. Takto vystačia s menším povrchom umelého trávnika a nikto si počas hry nič nevšimne. V realite futbalového zápasu je to absurdná predstava. V realite operačných systémov je to ale štandardný prístup.

Správa zariadení

Jej prvoradou úlohou je poskytnúť procesom funkcie rôznorodých vstupných a výstupných zariadení takým spôsobom, aby procesy už neboli zaťažené technickými detailmi práce jednotlivých konkrétnych zariadení, ale mohli ich používať na vyššej úrovni abstrakcie. Okrem toho sa správa zariadení stará aj o prideľovanie zariadení procesom (napríklad, aby sa dva procesy nesnažili naraz tlačiť na tlačiarni, lebo ich výstupy by sa pomiešali) a ochranu zariadení (nie každý proces smie pristupovať ku každému zariadeniu).

Hoci správa zariadení predstavuje veľkú časť každého operačného systému, programátori v moderných operačných systémoch často pristupujú k zariadeniam tak, akoby to boli špeciálne súbory. Takže z hľadiska programátora správa zariadení často nie je identifikovateľná. Preto sa ňou v tomto prehľade nebudeme bližšie zaoberať.

Správa súborov

Každý proces uchováva nejaké dáta vo svojej (virtuálnej) operačnej pamäti. Operačná pamäť je však pre niektoré aplikácie príliš malá, pre iné zase vadí to, že jej obsah zanikne spolu s procesom a niekedy je problémom aj zdieľanie dát v pamäti medzi procesmi. Preto potrebujeme v počítači mať aj inú možnosť na dlhodobé uloženie dát, ktoré by spĺňalo tieto kritériá:

- Dlhodobosť: Umožňuje uložiť veľké množstvo dát, väčšie než operačná pamäť
- Perzistencia: Dáta neprestanú existovať keď proces, ktorý ich uložil, skončí.
- Zdieľanie: Dáta nepatria jednému procesu, je ich ľahké zdieľať medzi procesmi.

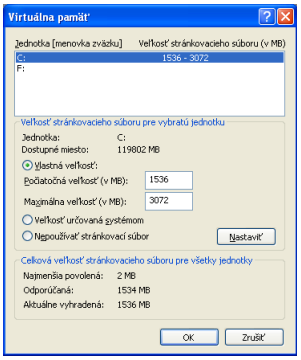
Obvyklé riešenie spĺňajúce uvedené kritériá je uložiť dáta na disky alebo iné podobné externé pamäťové médiá v jednotkách, ktoré nazývame **súbory**. O súbory sa stará operačný systém, konkrétne jeho správa súborov. Hlavnou úlohou správy súborov je poskytovať procesom prístup k súborom. To znamená vytvárať, udržiavať a rušiť súbory ako aj riadiť ich zdieľanie a ochranu.

Disk z hľadiska správy zariadení je pamäť s istou kapacitou, na ktorej sú uložené informácie v blokoch - sektoroch. Drajver zariadenia sprístupňuje disk len ako pole sektorov. Implementuje operácie na zápis sektora, čítanie a niekoľko ďalších pomocných operácií. Ďalej na tomto základe musí stavať správa súborov. Ona musí vedieť akým spôsobom sú v sektoroch disku uložené jednotlivé súbory, kde nájde ich mená, veľkosti, práva používateľov, ako súbor vytvoriť, zrušiť, zväčšiť či zmenšiť. Ako je všetko toto uložené na disku, to je definované v dohode, ktorú nazývame **súborový systém**. Jeden operačný systém môže pracovať naraz s niekoľkými zariadeniami, z ktorých každé má súbory uložené iným spôsobom - je na nich použitý iný súborový systém. Napríklad na systémovom disku je iný súborový systém než na výmennej SD karte z fotoaparátu aj než na DVD médiu.

Súbor

Súbor je abstraktný objekt, ktorú vytvára operačný systém na to, aby programy mohli ukladať svoje dáta. Pre používateľa počítača má však súbor veľmi konkrétny význam a obsah. Každý súbor má svoje meno, ktoré mu dal používateľ (alebo programátor, ak ide o pomocný súbor nejakej aplikácie, ktorý nikdy používateľ nepotrebuje vidieť).

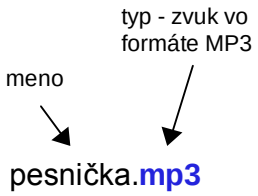
Operačný systém Windows používa vždy stránkovanie. Na odkladanie stránok, pre ktoré nie je miesto v pamäti, používa zvláštny stránkovací súbor (paging file). Jeho veľkosť a umiestnenie si bežne určuje samotný operačný systém. V špeciálnych prípadoch ho môžeme potrebovať zmeniť - napríklad keď ho chceme umiestniť na iný než systémový disk alebo chceme aby sa vôbec nepoužíval (stránkovanie sa tým nevypne bude však používať len dostupnú pamäť).
K nastaveniam virtuálnej pamäte sa dostaneme po pravom kliknutí na Tento počítač, voľba Vlastnosti/ Spresnenie/Výkon/ Nastavenie/Spresnenie/ Virtuálna pamäť/Zmeniť

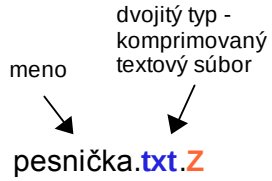


Obrázok 4: Okno nastavení virtuálnej pamäti vo Windows XP

Súbor je pojem, ktorý vznikol s operačnými systémami podobne ako proces, semafor, či drajver, na rozdiel od nich je však známy aj bežným používateľom.

Pre disky existuje niekoľko používaných súborových systémov a každý operačný systém uprednostňuje niektorý z nich ako svoj "vlastný". Napríklad Windows XP preferuje súborový systém NTFS, zatiaľ čo v Ubuntu je štandardom súborový systém ext4 alebo ext3.





Windows XP (a novšie verzie Windows) umožňuje skryť pred používateľom koncovku súboru a typ súboru znázorňuje graficky pomocou priradenej ikony a popisného mena. Po inštalácii je táto vlastnosť zapnutá, preto mnohí používatelia Windows vlastne už nepoznajú koncovky súborov.

V operačných systémoch Windows v snahe o priblíženie terminológie operačného systému používateľom, nahradili dovtedy zaužívaný termín directory slovom folder (v presom preklade šanón, obal na spisy) a priradili mu príslušný obrázok. V slovenskom preklade sa tento pojem preložil ako priečinok, v českom preklade je to složka. V tomto materiáli budeme používať pôvodný odborný názov adresár, o priečinkoch budeme hovoriť len v súvislosti s elektronickou poštou.

Všetky ukážkové programy k tejto kapitole nájdete v e-learningovom prostredí pod nadpisom Programy k 1. Kapitole (**Programy1.zip**). Program Adresare funguje bez zmien vo Windows aj v Linuxe.

Dáta v súbore môžu byť rôzneho typu. V mnohých operačných systémoch sa ujal zvyk označovať typ súboru pomocou koncovky (prípony) súboru - znakmi, ktoré nasledujú za poslednou bodkou v mene súboru. V Unixových operačných systémoch je zvykom dávať súboru aj niekoľko typov keď sa tým označuje, že ide o súbor v archíve alebo súbor preložený prekladačom.

Niektoré programy striktne rozlišujú súbory podľa typu zadaného v mene súboru, iné sa radšej riadia obsahom súboru a z neho vydedukujú typ súboru (väčšinou z prvých niekoľkých bajtov súboru).

Zadanie 5

Nájdite na disku grafický súbor typu jpg, png alebo bmp a zmeňte mu koncovku tak, aby označovala iný typ grafického súboru. Potom skúste súbor otvoriť pomocou niekoľkých programov, (skicár, RNA, Internet Explorer, FireFox, Irfan view). Ktorý z nich ho otvoril, ktorý vyhlásil chybu, ktorý úplne zlyhal?

Adresár, priečinok

Kapacita diskových zariadení bola v počiatkoch ich používania pomerne nízka, pre identifikáciu súboru stačilo jeho meno. Každé médium malo okrem samotných súborov uložený aj zoznam ich mien a atribútov - adresár (Directory). So zväčšujúcou sa kapacitou médií prišla aj požiadavka organizovať a štrukturovať ich obsah. Okrem hlavného adresára média bolo možné vytvoriť podadresáre napr. jeden pre každého používateľa počítača alebo jeden pre každú aplikáciu. V koreňovom adresári média boli potom záznamy o podadresároch a až v podadresároch boli záznamy o súboroch.

Moderné operačné systémy zovšeobecňujú tento prístup do ľubovoľnej hĺbky. Adresáre môžu obsahovať nielen súbory ale aj ďalšie adresáre a tie zase súbory aj adresáre a tak ďalej.

Keď exitujú podadresáre, tak nám už samotné meno neidentifikuje jednoznačne daný súbor. Preto sa používajú kompletné mená súborov, ktoré nazývame aj cesta k súboru. Cesta k súboru popisuje mená adresárov od koreňového adresára disku alebo globálneho koreňa až po meno súboru. Jednotlivé mená v ceste sú oddelené špeciálnym znakom, ktorý sa nemôže použiť v mene súboru ani adresára (vo Windows je to \, v unixovských operačných systémoch je to /).

Hoci je adresár dátová štruktúra, ktorú v prvom rade používa samotný operačný systém, je sprístupnená aj programátorom. V programoch môžeme čítať záznamy v adresároch a tým získavať informácie o súboroch a podadresároch daného adresára.

Zadanie 6

V adresári Adresare je ukážka programu na výpis stromu adresárov. Otvorte Adresare.lpi v Lazaruse a spustite ho. Do textového poľa napíšte celú cestu k nejakému adresáru a stlačte tlačidlo Strom. Preštudujte si text programu.

Operačný systém GNU/Linux

Pod názvom Linux (alebo Windows) si bežne používateľ najskôr predstaví jeho vzhľad, spôsob používania a aplikácie, ktoré v ňom má k dispozícii. Otázky, ktoré skutočne súvisia s vlastným operačným systémom, teda s tým, ako „to“ vo vnútri vlastne pracuje, sa však objavujú oveľa neskôr a väčšina používateľov si snáď žiadnu z tých otázok ani nikdy nepoloží.

V tomto module sa však chceme pozrieť dovnútra a práve preto je dôležité používať presnejší názov GNU/Linux. Súvisí to s tým, že softvér, ktorý je na linuxovom počítači nainštalovaný, môžeme rozdeliť zhruba na dve časti – na jadro a na aplikačný softvér (Obr. 1). Tieto dve časti majú rôznu funkciu, sú navzájom značne nezávislé a majú aj rôzny pôvod. O tom sme sa viac dozvedeli v module 3DG1.

Pre tento kurz bolo vytvorené Live DVD s modifikovaným OS Linux Ubuntu 9.04. DVD je dostupné na stránke http://sospreskoly.org/diskusia/dvd-na-stiahnutie V príkladoch v tomto texte sa predpokladá, že sa nachádzame v prostredí tohto DVD.

Základom operačného systému GNU/Linux je jeho jadro. Jeho tvorcom je Linus Torvalds a nazýva sa Linux. Jadro sprostredkováva spojenie medzi hardvérom počítača (pamäť a vstupno-výstupné zariadenia) a používateľskými programami. Tými sú bežné aplikácie, s ktorými sa stretáva používateľ – nástroje na spracovanie textu, kreslenie, prehrávanie hudby a videa – ale aj obslužné a iné špeciálne programy, s ktorými sa používateľ stretáva už menej (napríklad kompilátor). Väčšina nástrojov z tejto druhej kategórie vznikla v rámci projektu GNU(Gnu is not Unix), ktorý viedol Richard M. Stallman a jeho nadácia FSF (Free Software Foundation) od začiatku 80-tych rokov. Tieto nástroje sú neoddeliteľnou súčasťou operačného systému, bez nich by jadro nemalo ako plniť svoju úlohu. Z tohto dôvodu budeme v tomto texte odlišovať GNU/Linux ako operačný systém pozostávajúci z jadra a používateľských programov od samotného jadra Linux.

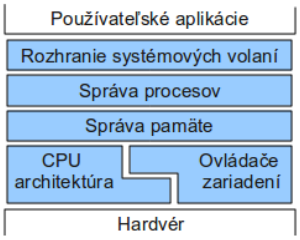
Rozlišovanie jadra a aplikácií je dôležité aj preto, lebo kombinácia GNU + Linux nie je jedinou možnosťou. Existujú aj GNU/Hurd (Hurd je jadro, ktoré vyvíja FSF) alebo GNU/Solaris (Solaris je zase jadro, ktoré vyvíja spoločnosť SUN), teda kombinácie nástrojov GNU s iným jadrom, ako je Linux. Naopak, Linux je bez GNU nástrojov použitý v operačnom systéme Android, ktorý spoločnosť Google vyvíja pre mobilné telefóny a iné podobné zariadenia.

Zadanie 1

Na stránke www.kernel.org v časti *What is Linux* zistite, ktoré počítačové architektúry Linux podporuje.

Jadro Linux

Jadro operačného systému je správcom hardvérových prostriedkov počítača, ktoré sprístupňuje používateľským programom. Veľmi zjednodušená schéma Linuxu je na Obr. 1. Podrobnejšiu schému možno nájsť na stránke http://www.makelinux.net/kernel_map.



Obrázok 1: Zjednodušená štruktúra jadra

Na najnižšej logickej úrovni jadra sa nachádza kód, ktorý priamo spolupracuje s hardvérom (napr. prostredníctvom jeho registrov). Táto časť na jednej strane súvisí s procesorom a jeho architektúrou a na druhej s prídavnými zariadeniami (disky, sieťové adaptéry a mnohé ďalšie. Jadro je napísané v jazyku C, pričom len malá časť, súvisiaca s architektúrou, musí byť napísaná v zodpovedajúcom assembleri. Vďaka tomu je Linux veľmi dobre prenositeľný – dnes ho možno rovnako dobre použiť na mobilných telefónoch, ale aj na najväčších superpočítačoch (http://en.wikipedia.org/wiki/List_of_Linux_supported_architectures).

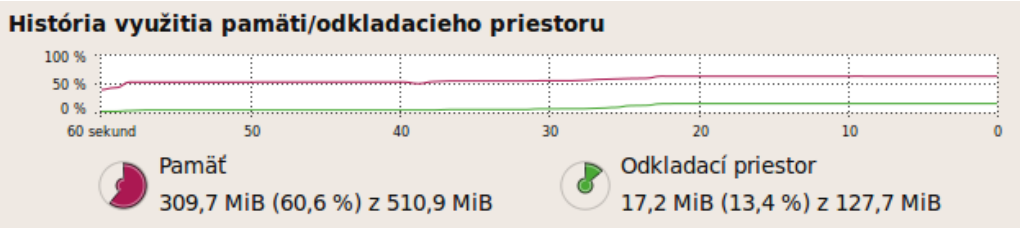
Ovládače zariadení

Najväčšiu časť kódu Linuxu predstavujú ovládače zariadení – Linux dnes podporuje najviac zariadení zo všetkých operačných systémov (http://broadcast.oreilly.com/2008/10/how-linux-supports-more-device.html).
Tieto ovládače sú pritom priamo zaradené do jadra, takže používateľ si ich nemusí sám doinštalovávať. Samozrejme, že toto pravidlo má aj výnimky. Napríklad, viaceré ovládače grafických adaptérov nie sú otvoreným softvérom a treba ich preto doinštalovať. V prípade Ubuntu systém na dostupnosť takýchto ovládačov sám upozorní alebo si ju možno overiť v menu Systém -> Správa -> Ovládače hardvéru. Samotná inštalácia je otázkou niekoľkých kliknutí myšou.

Správa pamäte

Jednou z úloh jadra je mapovanie virtuálneho adresného priestoru do fyzickej operačnej pamäte počítača alebo dokonca aj do pamäťového priestoru na diskovom zariadení. V druhom prípade ide o odkladanie pamäťových stránok na disk do odkladacieho priestoru a začne k nemu dochádzať v prípade, keď sa sumárne požiadavky všetkých bežiacich programov začnú blížiť k veľkosti operačnej pamäte počítača. Tento odkladací priestor sa vytvára pri inštalácii Linuxu, keď sa na tento účel vyčleňuje jeden diskový oddiel. Celkový stav pamäťových požiadaviek bežiacich programov zistíme pomocou programu *Systém -> Správa -> Monitor* systému v záložke *Zdroje* (Obr. 2).

Odkladací priestor má ešte jedno použitie. Pri hibernácii systému (teda pri takom vypnutí počítača, keď sa zachová aktuálny stav bežiacich programov) sa do odkladacieho priestoru uloží obsah celej operačnej pamäte. Odtiaľ sa pri opätovnom zapnutí počítača opäť načíta a programy pokračujú v práci tam, kde pred hibernáciou zastavili. Z tohto využitia aj vyplýva odporúčaná veľkosť odkladacieho priestoru – má byť väčší, ako operačná pamäť.



Obrázok 2: Grafické znázornenie spotreby operačnej pamäte a odkladacieho priestoru v programe Monitor systému.

Virtualizácia pamäte umožňuje obmedzenie práv používateľského programu. Vďaka nej program môže pristupovať len do „svojej“ pamäte a nie do pamäťového priestoru iných programov alebo dokonca do priestoru, ktoré využíva samotné jadro. Jadro samozrejme nepracuje s virtuálnou pamäťou – do operačnej pamäte pristupuje priamo. Tu častokrát na odlíšenie používame termíny *kernelový* a *používateľský* priestor.

Oddelenie používateľského a kernelového pamäťového priestoru má zásadný vplyv na stabilitu systému. Závažná chyba v používateľskej aplikácii, spočívajúca v zápise na pamäťové miesto, kam sa zapisovať nesmie, totiž nemôže viesť k ohrozeniu celého systému – abnormálne sa ukončí len chybná aplikácia. Iné je to, samozrejme v kernelovom priestore, kde taká chyba vedie k zrúteniu celého operačného systému, k tzv. *kernel panic*.

Na bežnú prácu v Linuxe úplne postačujú grafické nástroje. Naopak, profesionálny administrátor Windows sa nezaobíde bez príkazového riadka v programe PowerShell.

Príkazový riadok

Kým prídeme k niektorým vybraným detailom, ktoré súvisia s operačným systémom Linux (a vlastne aj ďalšími unixovými operačnými systémami), priblížime prácu v príkazovom riadku. Príkazový riadok hrá v Linuxe stále dôležitú rolu, a to aj napriek značnej popularite interaktívnych grafických aplikácií.

Z textových príkazov možno zostaviť program, ktorý sa nazýva skript. Medzi systémovými programami v GNU/Linuxe sa so skriptami stretneme veľmi často.

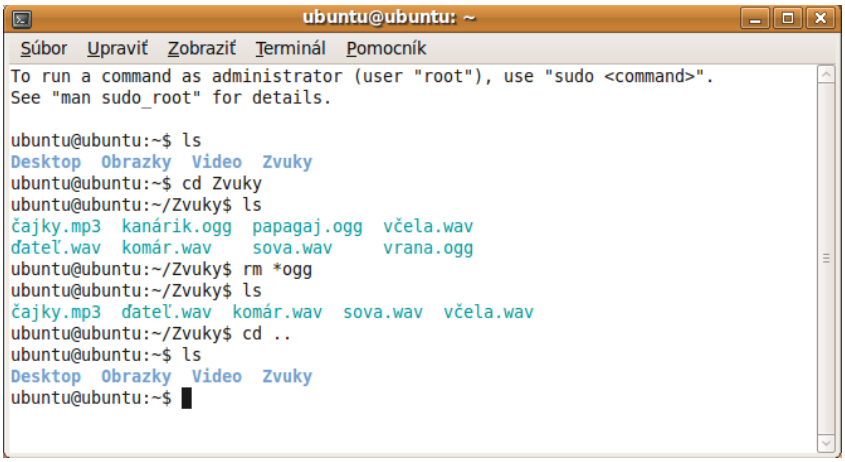
V príkazovom riadku sa zadávajú textové príkazy. Význam tohto prístupu spočíva najmä v ich jednoznačnosti a jednoduchosti. Aj keď mnohé funkcie sú dostupné aj prostredníctvom grafických aplikácií, pri použití textových príkazov sa pri vysvetľovaní môžeme sústrediť na podstatu, ktorá nie je prekrytá nepodstatnými detailmi grafického prostredia. Z tohto dôvodu sa textové príkazy preferujú napr. v návodoch a v dokumentácii operačného systému Ubuntu, pri administrácii serverov (tie obvykle vôbec nemajú grafické rozhranie) a na zjednodušenie prístupu k počítaču pre nevidiacich.

V GNU/Linuxe sa v príkazovom riadku pracuje v termináli. Máme k dispozícii buď

naozajstný terminál, do ktorého sa z grafického režimu prepneme kombináciou klávesov CTRL-ALT-F1 až F6 (je ich teda šesť, naspäť do grafického režimu sa dostaneme cez CTRL-ALT-F7) alebo emulátor terminálu (program `gnome-terminal`), ktorý je dostupný v grafickom režime cez menu *Aplikácie -> Príslušenstvo ->Terminál*.

Na Obr. 3 je uvedená ukážka práce s príkazovým riadkom v termináli v prípade Live DVD. Tu vlastne ide o komunikáciu s programom, ktorý sa nazýva interpreter riadkových príkazov (command-line interpreter, CLI). V tomto prípade ide o program `Bash`, ktorý príkazy používateľa preberá a následne vykonáva.

Interpreter svoju pripravenosť na prijímanie príkazov indikuje tzv. výzvou. V našom prípade ide o text *ubuntu@ubuntu:~$*. Znak *$* je jeho ukončujúci znak, za ktorým možno písať príkaz, ktorý sa odosiela stlačením klávesu *Enter*. Interpreter príkaz analyzuje a, pokiaľ bol syntakticky správny, tak ho aj vykoná. Obvykle pritom ide o spustenie jedného z mnohých, viac-menej jednoúčelových programov.



Obrázok 3: Terminál

Na obrázku je znázornená postupnosť použitých príkazov na vymazanie všetkých zvukových súborov vo formáte ogg v podadresári `Zvuky`. Použité sú tu príkazy `ls` - výpis obsahu adresára (**l**ist), `cd` - zmeň adresár (**c**hange **d**irectory), `rm` - vymaž (**rm**ove). Pre návrat do pôvodného adresára sme použili príkaz '`cd ..`'. Dve bodky sú špeciálnym označením adresára o úroveň vyššie. Ďalším špeciálnym označením je '`.`' (bodka), ktorá reprezentuje aktuálny adresár.

Príkaz má obvykle štruktúru '*názov prepínače meno_súboru ...*'. Názov je meno programu, ktorý danú činnosť prevedie. Prepínače sú parametre, ktoré bližšie definujú príkaz a meno_súboru predstavuje jeden alebo viac súborov, na ktoré sa príkaz vzťahuje. Prepínače začínajú znakom -. Nie všetky príkazy musia mať prepínače a mená súborov. Napríklad, už spomínaný príkaz `ls` môžeme použiť aj takto:

`ls -l`: dlhá forma, vypíše doplnkové informácie o súboroch
`ls -t`: súbory vypíše v poradí podľa času vzniku
`ls -S`: súbory vypíše v poradí podľa veľkosti.
`ls -l sova.wav`: vypíše detaily o súbore sova.wav
`ls -l *.wav`: vypíše detaily o všetkých súboroch s príponou wav.

Prepínače možno spájať: '`ls -lt`' alebo '`ls -lS`'. Pridaním prepínača -r sa obráti poradie: '`ls -rlS`'.

Výzva začína identifikáciou používateľa a počítača v tvare používateľ@systém, ktorá je ukončená znakom ':'. ubuntu@ubuntu v našom prípade teda znamená, že ide o používateľa ubuntu na počítači s menom ubuntu. Za dvojbodkou nasleduje názov aktuálneho adresára. Vlnovka ~ pritom označuje domovský adresár používateľa, v tomto prípade adresár `/home/ubuntu`. Identifikácia je dôležitá - z príkazového riadka sa veľmi jednoducho dá pomocou príkazu `ssh` pripojiť na iný počítač. Táto identifikácia teda zabraňuje situácii, keď používateľ vykoná síce správny príkaz, ale na nesprávnom počítači.

Ďalšie užitočné príkazy:

`cp odkiaľ kam`: skopíruje súbor 'odkiaľ' do adresára 'kam'. Ak 'kam' neexistuje alebo ak to nie je adresár, vytvorí sa identický súbor s novým menom 'kam'.
`mv odkiaľ kam`: presunie súbor 'odkiaľ' do adresára 'kam'. Ak 'kam' neexistuje, zmení sa meno súboru na 'kam'.
`rm subor`: vymaže súbor
`mkdir adresar`: vytvorí nový adresár
`rmdir adresar`: zmaže prázdny adresár. Ak adresár nie je prázdny, treba použiť '`rm -rf adresar`'
`cat meno_suboru`: vypíše obsah súboru na obrazovku.
`man meno_prikazu`: vypíše manuál príkazu (programu). Krátky návod obvykle môžme získať aj pomocou prepínača -h alebo --help. Napríklad: `ls --help`.
`nano meno_suboru`: jednoduchý textový editor
`gedit meno_suboru`: jednoduchý textový editor s vlastným grafickým oknom.

Zadanie 2

V termináli vypíšte obsah adresára (rôzne varianty), vytvorte a následne odstráňte nový adresár, skopírujte súbor do iného adresára, zobrazte textový súbor, otvorte textový súbor v textovom editore `nano` a pod.

Uvedené akcie vykonajte aj pomocou správcu súborov. Pokúste sa opísať postupnosť akcií, ktorú ste vykonali v oboch prostrediach, ďalšej osobe v maili. Ktorý spôsob sa bude ľahšie opisovať?

Správa súborov a prístupové práva

Pod pojmom súborový systém si obvykle predstavujeme priestor na ukladanie dát na diskovom zariadení (pevné a pružné disky, USB kľúče, SSD disky a iné). Na takomto zariadení súbory bývajú usporiadané hierarchicky v stromovej štruktúre (t.j. v adresári môžu byť okrem súborov aj ďalšie adresáre), pričom k ľubovoľnému súboru môžeme pristupovať priamo. Disk, tak ako je vyrobený v továrni, však nie je na takéto ukladanie súborov použiteľný.

Ak v počítači máme viacero diskových zariadení (alebo aspoň oddielov na disku), tak v niektorých operačných systémoch (napr. Windows) každé z nich má svoju vlastný a nezávislý súborový strom. Na počítači s OS GNU/Linux je však len jeden strom, pričom do jeho adresárov môžeme pripájať celé diskové alebo aj nediskové súborové systémy. Koreň tohto stromu sa nazýva *root* a označuje sa lomkou `/`. Lomka je aj oddeľovacím znakom pri zadávaní postupnosti adresárov. Napríklad, adresár `home`, ktorý je priamo v koreni, potom zadáme ako `/home` a adresár, ktorý je v ňom ako `/home/ubuntu`. Takáto postupnosť názvov sa nazýva *cesta*. Cesta môže byť absolútna, ak začína koreňom, alebo relatívna, ak začína v ľubovoľnom adresári. Špeciálne označenie máme pre označenie aktuálneho adresára (`.` - bodka), nadradeného adresára (`..` - dve bodky) a domovského adresára používateľa (vlnovka `~` pre aktuálneho používateľa alebo `~meno` pre ostatných používateľov).

Nediskový súborový systém devfs je pripojený do adresáta /dev a procfs do /proc

Zadanie 3

Pomocou správcu súborov sa zoznámte so súborovým systémom. Nájdite adresár používateľa `ubuntu`.
V správcovi použite náhľad na hierarchiu adresárov.

To isté spravte pomocou terminálu a príkazu `ls`.

Používateľské práva

Jedným zo základných pravidiel bezpečnej práce s počítačom je princíp najmenších privilégií, podľa ktorého používateľ má mať čo najmenšie práva, ktoré však postačujú na vykonanie požadovanej činnosti. V operačných systémoch sa toto pravidlo bežne implementuje rozlišovaním medzi bežným používateľom a administrátorom. Administrátor má všetky práva, zatiaľ čo bežný používateľ len niektoré – obvykle len také, aby mohol pracovať len s istými programami a len s vybranými adresármi a súbormi.

V operačných systémoch unixového typu, a teda aj v Linuxe, sa okrem práv používateľa a administrátora definujú aj práva skupiny používateľov, ktorý sa využíva na dva účely. V prvom rade môže vlastník súboru rozšíriť práva aj pre ďalších (ale nie nevyhnutne všetkých) používateľov mimo vlastníka. Druhým použitím je úprava, kto môže spúšťať vybrané programy a používať rôzne zariadenia počítača. Kvôli tomu môže byť používateľ zaradený aj do viacerých skupín.

V Linuxe každý používateľ patrí aspoň do jednej (svojej vlastnej) skupiny. Na druhej strane, každý súbor je priradený práve jednému používateľovi a jednej skupine a má tri trojice atribútov. Prvá špecifikuje práva *vlastníka*, druhá práva člena *skupiny*,

ktorej súbor patrí a tretia práva *ostatných*. V každej trojici sú tri atribúty – právo na *čítanie* súboru, právo na *zápis* do súboru a právo na *vykonanie* (spustenie) súboru. Právo na zápis automaticky znamená aj právo na úpravu a zmazanie. V prípade adresárov tieto atribúty majú trochu iný význam. Vlastníka, skupinu a atribúty súboru najľahšie zistíme pomocou príkazu `ls -l`. Pre rôzne typy súborov obvykle dostaneme nasledujúce výpisy:

Dátový súbor:
```
-rw-r--r-- 1 ubuntu users 45326 2009-12-14 15:51 minv1.png
```

Vykonateľný súbor (program):
```
-rwxr-xr-x 1 root root 64208 2009-10-06 13:06 rm
```

Súbor zariadenia:
```
crw-rw---- 1 root cdrom 21, 0 2010-01-02 11:58 sg0
```

Adresár:
```
drwxr-xr-x 25 ubuntu ubuntu 4096 2009-12-18 09:28 ubuntu
```

Napríklad, môžeme mať skupinu projekt, ktorej členovia môžu zdieľať (čítať, vytvárať, mazať) vybrané súbory a adresáre patriace skupine projekt. V inej situácii nemusí byť žiaduce, aby používatelia, ktorí nemajú fyzický prístup k počítaču, mohli pracovať s prídavnými zariadeniami (napr. s CD mechanikou). Preto v prípade CD mechaniky je vytvorená skupina cdrom, ktorá zabezpečí, aby s mechanikou mohol pracovať len ten, kto má práva vhodné pre používateľa desktopu.

Pri vysvetľovaní významu polí začneme od konca: sú to názov, čas a dátum vzniku alebo poslednej úpravy a veľkosť (v prípade zariadenia je význam tohto poľa iný). O právach hovoria prvé (atribúty), tretie (vlastník) a štvrté (skupina) pole. Atribúty majú 10 znakov:

1: Typ (-: súbor, d: adresár, c: znakové zariadenie, b: blokové zariadenie)
2 – 4: vlastník, právo na čítanie (r), zápis (w) a vykonanie (x)
5 – 7: skupina, právo na čítanie (r), zápis (w) a vykonanie (x)
8 – 10: ostatní, právo na čítanie (r), zápis (w) a vykonanie (x)

V prípade adresára je význam trochu iný: w – právo vytvárať a mazať súbory v adresári, x - právo vojsť do adresára.

Čo nám teda hovoria práva, ktoré sme zistili v predošlých riadkoch:

- Vlastník ubuntu dátového súboru ho môže čítať a zapisovať (teda meniť alebo zmazať). Príslušníci skupiny users a ostatní ho môžu len čítať (môžu si ho však skopírovať a svoju kópiu už meniť môžu).
- Súbor (program) rm môže ktokoľvek čítať a spúšťať, ale len vlastník (administrátor) ho môže meniť a zmazať.
- Zariadenie sg0 (cdrom mechanika) môže používať každý člen skupiny cdrom.
- Do adresára ubuntu (domovský adresár používateľa ubuntu) môže vojsť ktokoľvek a môže v ňom aj čítať súbory. Vytvárať a mazať súbory však v ňom môže len vlastník.

V skupine cdrom zrejme budú používatelia, ktorí majú možnosť pracovať priamo pri počítači. Tí, ktorí majú k nemu len vzdialený prístup cez internet ho nepotrebujú a teda v tejto skupine asi nebudú. Pre viaceré zariadenia môžu takto byť vytvorené špeciálne skupiny, prostredníctvom ktorých možno detailne nastavovať prístupové práva.

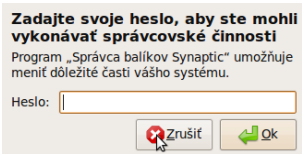
Zadanie 4	Zobrazte si atribúty súborov v rôznych adresároch (domovský adresár, adresár Zvuky, /dev/ … To isté spravte pomocou správcu súborov.
Riešenie	V správcovi súborov treba zvoliť Pohľad zoznam, ktorý sa nachádza vpravo hore. V zozname sa štandardne atribúty neuvádzajú. Treba ich povoliť v menu správcu súborov Upraviť-> Predvoľby, v záložke Stĺpce zoznamu.

Zmena používateľských práv

Na zmenu atribútov slúži príkaz *chmod*. Jeho syntax je

```
chmod [ugoa][+-][rwx] subory
```

pričom zátvorky [] znamenajú, že treba použiť jeden zo znakov vo vnútri. Význam: u-user(vlastník), g-group(skupina), o-other (ostatní), a-all (všetci) + právo pridať, - právo zobrať. r, w, x bolo vysvetlené už skôr.



Obrázok 4: Dialógové okno na zadávanie hesla

Zadanie 5	V adresári Zvuky zrušte právo na čítanie pre wav súbory ostatným používateľom.
Riešenie	Použite príkaz `chmod o-r *.wav` a výsledok si overte pomocou `ls -l` To isté sa dá dosiahnuť aj cez správcu súborov v položke *Vlastnosti* kontextového menu súboru.

Administrátorské práva

Administrátor má všetky práva a v unixových operačných systémoch sa nazýva *root*. V Ubuntu však tento používateľ štandardne nie je aktivovaný, ale aspoň jeden z používateľov (vždy ten, ktorý bol vytvorený ako prvý - na Live DVD je ním používateľ *ubuntu*) môže jeho práva získať pomocou príkazu `sudo`, ktorý sa uvedie v príkazovom riadku pred samotným príkazom, ktorý chceme vykovať. Príkaz `sudo` musíme potvrdiť heslom používateľa. Ak grafická aplikácia vyžaduje administrátorské práva, vyžiada zadanie hesla používateľa v dialógovom okne.

Použitie príkazu `sudo` si ukážeme na príklade, v ktorom vytvoríme novú skupinu (kurz) používateľov. Na Live DVD používateľ ubuntu nemá heslo, a tak si ho príkaz `sudo` ani nepýta. Na nainštalovanom systéme je to samozrejme inak. Po zadaní hesla si ho systém niekoľko minút pamätá.

Zadanie 6	Vytvorte skupinu 'kurz' a zmeňte skupinu súborov ogg súborov v adresári `Zvuky` na 'kurz'
Riešenie	Použite príkazy `sudo addgroup kurz` a `sudo chgrp kurz *.ogg`. Príkazy `addgroup` a `chgrp` najskôr zadajte bez `sudo` a pozorujte, čo sa stane.
Zadanie 7	Pomocou nástroja *Systém -> Správa -> Používatelia a skupiny* vytvorte používateľa 'kurz1' a pridajte mu práva používateľa desktopu. O ktoré práva ide, to si overte v záložke *Používateľské práva*. Vyskúšajte si, aké práva by používateľ mal, ak by bol unprivileged (neprivilegovaný). V príkazovom riadku zistite, do ktorých skupín používateľ kurz1 patrí (príkaz: `groups kurz1`)
Zadanie	Pomocou nástroja *Systém -> Správa -> Používatelia a skupiny* v položke '*Spravovať skupiny*' pridajte používateľa *kurz1* do predtým vytvorenej skupiny 'kurz'.

Riešenie 8

Nájdite skupinu 'kurz', kliknite na *Vlastnosti*, zaškrtnite používateľa 'kurz1' a potvrďte.

Výsledok si opäť overte pomocou príkazu `groups kurz1` a výsledok porovnajte s výsledkom jeho predchádzajúceho použitia.

Vytváranie a správa procesov

Jednou z hlavných úloh jadra je vytváranie nových procesov, zavádzanie programov zo súboru do pamäte a prideľovanie procesorového času jednotlivým procesom.

V tejto časti potrebujeme na experimentovanie viacero programov, ktoré sú dostupné v e-learningovej podpore pod nadpisom Programy k 2.kapitole (**Programy2.zip**).

Spúšťanie programov, popredie a pozadie

Už vieme, že príkazy (programy) sa v termináli spúšťajú jednoducho zadaním ich mena. Interpreter príkazov, ktorý nám toto spúšťanie umožňuje, však program musí pred spustením najskôr nájsť. Ak zadáme, napríklad, `ls`, interpreter hľadá program `ls` v predvolených adresároch, ktorými sú `/bin`, `/usr/bin` a ďalšie. Ak nás zaujíma, o ktorý program vlastne ide, môžeme použiť program `which`. Ten vypíše adresár, kde sa program nachádza. Napríklad:

```
ubuntu@ubuntu:~$ which ls
ubuntu@ubuntu:~$ /bin/ls
```

Súbory s programami v Linuxe nemávajú žiadnu príponu. Ich spustiteľnosť určujú atribúty súboru.

Program `ls` by sme teda mohli spustiť aj zadaním úplnej cesty `/bin/ls`. Interpreter však neprehľadáva aktuálny adresár, a tak nenájde programy, ktoré sme si sami vytvorili. Preto ich musíme zadať aj s cestou. V prípade programu `slucka`, ktorý je v archíve stiahnutom zo stránky modulu, vojdeme do jeho adresára a zadáme

```
./slucka
```

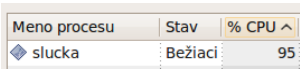
Program slucka je skript, ktorý obsahuje len pár príkazov. Presvedčiť sa o tom môžeme tak, že ho otvoríme v textovom editore.

kde `./` je cesta k programu (`.` je aktuálny adresár – to už vieme). Možno nás prekvapí, že po spustení tohto programu terminál prestane zmysluplne reagovať. Je to tým, že program `slucka` sme, ako hovoríme, *spustili v popredí*. Vtedy program blokuje klávesnicu tým, že sám preberá všetky znaky, ktoré z nej idú. V programe `slucka` je však nekonečná slučka a program inak nerobí nič. Terminál sa uvoľní až potom, keď program skončí. Keďže `slucka` sám nikdy neskončí, musíme ho ukončiť násilne – stlačením CTRL-C. Aby program terminál neblokoval, musíme ho s*pustiť na pozadí* – tak, že za príkaz pridáme znak `&` (ampersand):

```
./slucka &
```

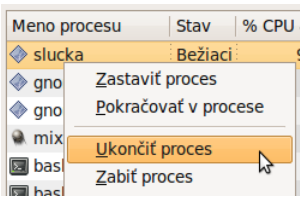
Všetky programy, ktoré boli spustené z menu alebo plochy grafického rozhrania, sú v tomto zmysle spustené na pozadí.

Terminál je naďalej funkčný, ako keby program `slucka` ani nebežal. O tom, že beží, sa môžeme presvedčiť pomocou grafického nástroja *Systém -> Správa -> Monitor systému*, v záložke *Procesy* (Obr. 5). Tu vidíme aj ďalšie údaje o procesoch, napríklad podiel celkového procesorového času, ktorý proces zaberá. Hodnota 95%, ktorá je uvedená pri programe `slucka` na Obr. 5 predstavuje podiel na výkone celého počítača. Ak by mal dva procesory, tak by tu bola zhruba polovičná hodnota.



Obrázok 5: Záznam o bežiacom programe slucka v Monitore systému.

V záložke *Zdroje* je graficky znázornené celkové zaťaženie procesora. V Monitore systému môžeme proces aj ukončiť, a to v kontextovom menu, ktoré otvoríme kliknutím pravým tlačidlom na riadok procesu. V menu vyberieme položku *Ukončiť proces* a potvrdíme.

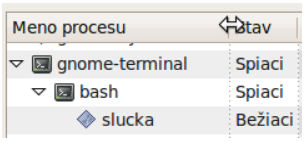


Obrázok 6: Násilné ukončenie procesu.

Vytváranie procesov, volania fork a exec

V tejto časti si na príklade spúšťania programov z programov ukážeme, ako používateľské programy využívajú služby jadra.

Najskôr sa však pozrime na to, čo nám Monitor systému ešte o našom bežiacom



Obrázok 7: Jedna z vetiev stromu procesov, zobrazená Monitorom systému.

Monitor systému závislosť na procese `init` nezobrazuje, presvedčiť sa o tom však môžeme pomocou programu `pstree`, ktorý zadáme v príkazovom riadku.

programe vie ukázať. Opäť spustime program `slucka` na pozadí, otvorme si *Monitor systému* a v položke *Zobraziť* zaškrtnime *Závislosti*. Zobrazenie procesov sa zmení - vidíme, že náš proces vykonávajúci program `slucka` je akoby podradený procesu `bash` a ten je podradený procesu `gnome-terminal`. Je to naozaj tak: program `gnome-terminal` spustil interpreter príkazov `bash` a ten spustil náš program `slucka`. Takéto spúšťanie sa robí to pomocou dvoch volaní jadra - *fork* a *exec*.

Volanie *fork* zduplikuje bežiaci proces. Nový proces pritom je potomkom pôvodného procesu (rodiča), čím vzniká stromové usporiadanie procesov. Tento strom má koreň, proces *init*, ktorý má identifikátor 1 a je vlastne rodičom všetkých ostatných procesov v bežiacom systéme.

Po rozvetvení oba procesy, pôvodný aj nový, pokračujú ďalej vo vykonávaní toho istého programu. Nový proces však dokáže zistiť, že je potomkom - volanie fork v ňom vracia hodnotu 0 - a následne volaním exec obvykle nahradí vykonávaný program novým programom. Táto funkcionalita je dokumentovaná na Obr. 7, kde hlavný program spustí program `Gimp`, počká na jeho ukončenie (volanie jadra wait) a načíta obrázok, ktorý bol v Gimpe vytvorený. Tento program sa nachádza v stiahnutom súbore Programy2.zip v adresári `runproc`.

```
begin
  image1.Picture.SaveToFile('pok.png');
  t := fpFork;           {vetvenie}
  if t = 0 then          {v potomkovi}
     fpExecLP('gimp', ['pok.png']);{potomok: povodny program sa nahradi gimp-om}
  {Na toto miesto uz potomok nepride}
  fpWait(stat); {rodic: cakaj na ukoncenie potomka}
  image1.Picture.LoadFromFile('pok.png');
end;
```

Obrázok 8: Ukážka vetvenia procesov v jazyku Pascal. Volania jadra sú tu reprezentované pascalovskými funkciami fpFork, fpExecLP a fpWait.

Zadanie 9

V programe Lazarus otvorte a preložte projekt, ktorý je v adresári `runproc` z archívu Programy2.zip. Otvorte si *Monitor systému*, zvoľte zobrazenie závislostí procesov a vytvorenú aplikáciu z Lazarusu spustite.

Program najskôr vytvorí okno a po stlačení tlačidla spustí program Gimp. V ňom niečo nakreslite, obrázok uložte a Gimp ukončite. Následne sa obrázok načíta do okna aplikácie. V *Monitore systému* pozorujte vytváranie jednotlivých procesov a ich závislostí po jednotlivých akciách.

Spustenie zopakujte priamo zo správcu súborov (ide o program `projekt1`). Ako sa zmenilo miesto procesu v hierarchii procesov?

Počítačové siete

Počítačové siete a prístup k internetu – sieti sietí – sú dnes bežnou samozrejmosťou. Na mnohé ich služby sme si zvykli a rutinne ich používame – WWW, elektronickú poštu, chat či dokonca videokonferencie, prístup k centralizovaným sieťovým zdrojom (dáta, tlačiarne, skenery), vzdialené ovládanie zariadení a ďalšie. V pozadí všetkých týchto služieb stoja mnohé sieťové technológie, algoritmy a protokoly. Cieľom tejto tematickej jednotky je prehĺbiť znalosti o kľúčových princípoch, technológiách a protokoloch používaných v dnešných počítačových sieťach.

Referenčný model siete – OSI

Referenčné modely, model Open System Interconnection

Pri otázke „Vysvetlite mi v skratke, ako funguje počítačová sieť!“ by bezpochyby aj skúsený sieťový odborník na chvíľku zaváhal. Počítačové siete sú totiž, najmä keď zoberieme úvahy rôznorodosť sieťových služieb a technológií, mimoriadne rozmanitým komplexom technických prostriedkov a programového vybavenia, v ktorom sa na prvý pohľad dá len veľmi ťažko orientovať.

Na zrozumiteľný popis komplexného systému však máme prostriedok, a tým je jeho **model** – opis, ktorý vystihuje podstatné vlastnosti systému a zanedbáva nedôležité detaily. Spoločným rysom modelov komplexných systémov je, že sa v nich modelovaný systém dekomponuje – rozloží na menšie a jednoduchšie popísateľné celky. Model potom opisuje, ako jednotlivé celky pracujú a ako navzájom spolupracujú. Hoci model ako celok môže takisto byť rozsiahly, jednotlivé komponenty, z ktorých sa skladá, sú jednoduchšie a z modelu je zrejmá aj štruktúra celého pôvodného systému.

Obrázok 1: Referenčný model Open System Interconnection

Takéto sprehľadňujúce modely existujú aj pre počítačové siete. Zrejme najpopulárnejším modelom je tzv. **referenčný model Open System Interconnection** (RM OSI), ktorého autorom je Medzinárodná štandardizačná organizácia ISO. RM OSI nepopisuje nijakú konkrétnu sieťovú technológiu. Je abstraktným modelom *ľubovoľnej* siete – opisuje totiž všeobecné procesy, ktoré musia prebiehať v každej sieti nezávisle od jej druhu. Tieto procesy sú v RM OSI roztriedené do siedmich samostatných hierarchicky usporiadaných kategórií, tzv. **vrstiev**. Každá vrstva predstavuje množinu vzájomne príbuzných procesov a služieb, ktoré sú zodpovedné za istú konkrétnu činnosť siete. Vrstva na istej úrovni poskytuje svoje služby vyššej vrstve a sama pre svoju správnu činnosť využíva služby nižšej vrstvy.

Fyzická vrstva RM OSI (Physical Layer)

Úlohou fyzickej vrstvy siete je prenos jednotlivých bitov alebo ich skupín pomocou elektrického, optického alebo rádiového signálu po prenosovom médiu. Fyzická vrstva preto popisuje samotné prenosové médium (metalický či optický kábel, bezdrôtové rádiové spojenie a i.) a jeho vlastnosti, parametre sieťových rozhraní, povahu prenášaného signálu (elektrický, optický, spôsoby kódovania), inými slovami, týka sa všetkých fyzikálnych vlastností a charakteristík prenášaného signálu a prenosového média. Fyzická vrstva samotným prenášaným dátam *nerozumie*, chápe ich len ako prúd bitov.

V súčasnosti sa používajú modulačné techniky, ktoré v jedinej zmene vlastnosti prenášaného signálu (amplitúdy, fázy) vyjadria hodnotu viac bitov naraz.

V reálnej sieti medzi komponenty pracujúce na fyzickej vrstve patrí samotná kabeláž (optická alebo metalická), konektory a zásuvky, prepojovacie panely, opakovače signálu (z angl. repeater), konvertory medzi optickou a metalickou kabelážou (prevodníky, z angl. transceiver), rozbočovače (z angl. hub) či modemy.

Linková vrstva RM OSI (Data Link Layer)

Vďaka fyzickej vrstve je možné po prenosovom médiu odoslať a prijať dáta ako prúd bitov. Ďalšou prirodzenou požiadavkou je, aby prostredníctvom fyzickej vrstvy dokázali navzájom komunikovať navzájom prepojené čiže susedné uzly siete. Prostriedky pre túto komunikáciu poskytuje druhá vrstva RM OSI – linková vrstva.

V linkovej vrstve sa nachádzajú mechanizmy na komunikáciu medzi dvomi susednými uzlami v sieti, t.j. uzlami, ktoré sú pripojené na spoločné komunikačné médium. Z pohľadu linkovej vrstvy sú dáta prenášané ako sled individuálnych správ, ktoré budeme nazývať **rámce** (z angl. frame). Rámec je údajová štruktúra, ktorá obsahuje jednak riadiace informácie potrebné pre jej doručenie, jednak samotné prenášané dáta. Štruktúra rámca je pre rôzne linkové technológie rozličná, no pre konkrétnu technológiu (napr. Ethernet) majú všetky prenášané rámce rovnaký základný formát.

Linková vrstva poskytuje prostriedky pre riadenie prístupu k médiu, adresovanie uzlov na spoločnom médiu, kontrolu správnosti prenosu, spoľahlivosť, spojovanosť a riadenie toku dát. Podrobnejší popis týchto mechanizmov sa nachádza v e-learningovej podpore k tomuto modulu.

Novšie varianty Ethernetu majú nepovinnú podporu pre hrubé riadenie toku dát vďaka tzv. PAUSE rámcom.

Je dôležité poznamenať, že rôzne linkové technológie implementujú z uvedených funkcií linkovej vrstvy len niektoré. Jedna z najpoužívanejších linkových technológií, Ethernet, bežne nepodporuje spoľahlivosť, spojovo orientovaný prenos ani riadenie toku dát. Mnohé z týchto funkcií sú preto implementované aj vo vyšších vrstvách.

Typickými komponentmi linkovej vrstvy sú sieťové karty, prepínače (z angl. switch), bezdrôtové sieťové adaptéry a bezdrôtové prístupové body.

Sieťová vrstva (Network Layer)

Prvé dve vrstvy RM OSI obsahujú všetky potrebné prostriedky na to, aby navzájom úspešne dokázali komunikovať ľubovoľné dva uzly prepojené spoločným médiom.

Sieťová vrstva pridáva dôležité mechanizmy, aby medzi sebou mohli komunikovať ľubovoľné dva uzly nachádzajúce sa *kdekoľvek* v sieťovej infraštruktúre, nielen na spoločnom médiu.

Slovo „sieť“ má množstvo rôznych významov, tu pracujeme so slovom „sieť“ vo význame, v akom ho používa sieťová vrstva.

Typickým príkladom sieťovej adresy je bežné telefónne číslo pevnej stanice – obsahuje predčíslie štátu, predčíslie primárnej oblasti a samotné číslo účastníka.

Predovšetkým, sieťová vrstva zavádza pojem siete. Sieť je množina všetkých uzlov na spoločnom médiu. Sieťová vrstva ďalej umožňuje prideliť sieťam a jednotlivým uzlom v nich tzv. sieťové adresy, ktoré pozostávajú z dvoch častí. Jedna časť sieťovej adresy označuje samotnú sieť a je spoločná pre všetky uzly v nej, druhá časť označuje konkrétny uzol v danej sieti. Týmto spôsobom je jednoznačne identifikovaný ľubovoľný uzol v ľubovoľnej sieti.

Ďalšou dôležitou úlohou sieťovej vrstvy je poskytnúť prostriedky, ako prepojiť viaceré doposiaľ oddelené siete do väčšej súvislej infraštruktúry a ako hľadať cestu medzi dvomi ľubovoľnými komunikujúcimi uzlami – cez ktoré medziľahlé siete a uzly musia tiecť dáta od konkrétneho odosielateľa ku konkrétnemu adresátovi. Pre tento účel sieťová vrstva takisto zavádza vlastnú údajovú štruktúru, v ktorej sa prenášajú jednak informácie identifikujúce konkrétneho koncového odosielateľa a príjemcu, ďalej pomocné riadiace informácie a nakoniec samotné prenášané dáta. Túto údajovú štruktúru nazývame **paket** (z angl. packet). Formát paketu takisto závisí od konkrétneho sieťového protokolu. Pakety sa pri odosielaní vždy vkladajú do dátovej časti rámca.

Typickým zariadením sieťovej vrstvy je smerovač (z angl. router), ktorý prepája navzájom viaceré siete a smeruje tok paketov medzi nimi. Smerovač musí poznať, aké siete sa v jeho okolí nachádzajú a akou trasou je do nich možné doručiť pakety.

Transportná vrstva (Transport Layer)

Sieťová vrstva sa postará o prenos paketov medzi ľubovoľnými uzlami, no nemusí poskytovať ďalšie služby. Nemusí mať prostriedky na to, aby odoslané pakety doručila v tom poradí, v akom boli odoslané, nemusí vedieť zariadiť opakované doručenie zle preneseného či strateného paketu, takisto nemusí mať prostriedky na riadenie toku dát. Jediné, čo sa od hocijakej implementácie sieťovej vrstvy vždy očakáva, je iba snaha doručiť pakety ich príjemcovi.

Všetky tieto chýbajúce vlastnosti sú ponechané na starosť práve transportnej vrstve.

Transportná vrstva sa v princípe stará o to, aby komunikácia medzi dvomi uzlami bola spoľahlivá s možnosťou spojovo orientovanej komunikácie. Ak sa paket prenesie nesprávne alebo sa stratí, transportná vrstva zabezpečí jeho opakované odoslanie. Ak pakety dorazia v inom poradí, transportná vrstva ich preusporiada do pôvodného poradia. Transportná vrstva si vie zároveň prispôsobiť objem naraz prenášaných dát podľa aktuálneho stavu siete a adresáta, a tak regulovať rýchlosť prenosu dát.

Súvislý blok dát odosielaný adresátovi sa na úrovni transportnej vrstvy rozdelí (segmentuje) na menšie časti a každá časť dostane vlastnú transportnú hlavičku, čím vznikne segment.

Transportná vrstva využíva vlastnú údajovú štruktúru s názvom **segment**. Analogicky ako pri predchádzajúcich vrstvách, aj segment má vlastnú hlavičku s riadiacimi informáciami a samotnú dátovú časť. Segmenty sa pri odosielaní vkladá do dátovej časti paketu a v hlavičke paketu sa vyznačí koncový odosielateľ a príjemca dát. Paket sa vloží do dátovej časti rámca a priloží sa k nemu ďalšia hlavička, ktorá identifikuje odosielajúcu a prijímajúcu sieťovú kartu na spoločnom segmente.

Transportná vrstva musí byť implementovaná u každého koncového odosielateľa a adresáta dát. Funkcie transportnej vrstvy sú preto združené v softvérovom sieťovom ovládači.

Relačná vrstva (Session Layer)

Vďaka prvým štyrom vrstvám sa odoslané dáta doručia koncovému adresátovi v takom tvare, v akom boli odoslané. Avšak tieto dáta sú v skutočnosti odoslané z konkrétneho programu a takisto u adresáta ich má spracovať konkrétny aplikačný softvér, napríklad poštový program, ICQ klient alebo WWW server. Musí preto existovať spôsob, ako vyznačiť, pre akú softvérovú aplikáciu bežiacu na koncovom uzle sú dané dáta určené. Naviac existujú softvérové aplikácie, pri ktorých je potrebné v komunikácii dodržať istú schému – kedy má právo ktorý softvér posielať a kedy prijímať správy, či je možné vrátiť sa v dialógu nazad (undo), prípadne ako medzi sebou rozoznať niekoľko súbežných *dialógov* medzi tými istými softvérovými aplikáciami na tých istých uzloch (sťahovanie viacerých súborov naraz z toho istého servera tým istým klientom). Prostriedky pre pokrytie týchto potrieb sú združené v relačnej vrstve. Relačná vrstva je teda zodpovedná za identifikáciu komunikujúcich aplikácií a za riadenie komunikačného vzťahu, tzv. relácie, medzi nimi. Pojem relácie a funkcií relačnej vrstvy je však pomerne abstraktný a pre nás nebude dôležitý.

Prezentačná vrstva (Presentation Layer)

Prezentačná vrstva zabezpečuje funkciu „prostredníka“ medzi dvomi komunikujúcimi aplikáciami, ktorého úlohou je dohodnúť, identifikovať a v prípade potreby konvertovať formát vzájomne prenášaných údajov – napríklad automatické dohodnutie jazyka a kódovania diakritiky medzi webovým browserom a serverom alebo identifikácia typu prenášaného dokumentu (napr. obrázok, dokument, spustiteľný súbor). Medzi ďalšie funkcie prezentačnej vrstvy patrí aj kompresia dát a ich šifrovanie, ak sa deje na úrovni používateľskej aplikácie.

Aplikačná vrstva (Application Layer)

Aplikačnú vrstvu predstavujú jednotlivé aplikácie, s ktorými pracuje používateľ, teda napríklad WWW browsery, prehliadače pošty, P2P programy, prehrávače zvuku a videa šíreného po sieti, chat klienti, softvérové IP telefóny a mnohé ďalšie programy. Aplikačná vrstva je jediná vrstva, ktorá neponúka svoje služby ďalšej nadradenej vrstve, ale s ktorou už pracuje samotný používateľ.

Referenčný model TCP/IP

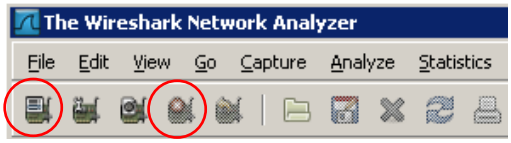
Hoci je RM OSI najpoužívanejším referenčným modelom sietí, je len abstraktným a všeobecným popisom. Konkrétna sieťová technológia môže byť implementovaná s miernymi rozdielmi a úpravami. Dnes najpoužívanejšia rodina protokolov TCP/IP má takisto svoj vlastný model. Podrobnosti o ňom a o jeho vzťahu k RM OSI nájdete v e-learningovej podpore k tomuto modulu.

Čo sme sa naučili

Referenčný model OSI je všeobecný popis služieb a procesov prebiehajúcich v počítačových sieťach. V jeho siedmich vrstvách sú kategorizované jednotlivé činnosti, ktoré sieť vykonáva pri prenose dát. Fyzická vrstva zabezpečuje prenos individuálnych bitov po prenosovom médiu. Linková vrstva poskytuje prostriedky pre komunikáciu uzlov pripojených k spoločnému médiu. Sieťová vrstva zodpovedá za unikátne adresovanie uzlov, ich zoskupovanie do sietí a za doručenie dát medzi ľubovoľnými uzlami. Transportná vrstva sa stará o spoľahlivé doručovanie dát. Relačná vrstva riadi jednotlivé dialógy medzi komunikujúcimi aplikáciami na koncových uzloch. Prezentačná vrstva umožňuje identifikovať a dohodnúť si spoločný formát prenášaných dát. Aplikačná vrstva poskytuje služby, ktoré využíva koncový používateľ siete.

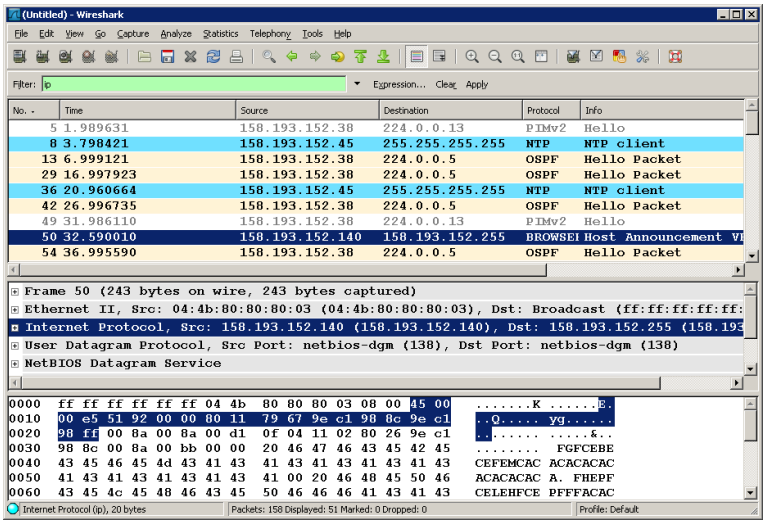
Zadanie 1

Spustite program Wireshark a na titulnej obrazovke kliknite na prvú ikonu v pruhu ikon:



V zobrazenom okne kliknite na tlačidlo „Start“ v tom riadku, v ktorom sa priebežne zvyšujú čísla v stĺpcoch „Packets“ a „Packets/s“, čím spustíte zachytávanie sieťovej prevádzky zo zvolenej sieťovej karty. Navštívte ľubovoľnú webovú stránku a po jej zobrazení kliknite na štvrtú ikonu v pruhu úloh, čím zachytávanie ukončíte.

Prezrite si podrobnosti zachytenej prevádzky v prostrednej časti okna. Prvý pruh obsahuje jednotlivé zachytené pakety. Druhý pruh obsahuje detaily zvoleného paketu. Tretí pruh obsahuje hexadecimálny výpis celého rámca. Po kliknutí na vybranú položku v detailoch paketu sa príslušné bajty tejto položky v hexadecimálnom výpise zvýraznia.



Pomocou riadku Filter pod pruhom ikon obmedzte výpis zachytenej prevádzky – vložte niektorý z výrazov **ip**, **tcp**, **udp**, **arp** a dvakrát stlačte Enter alebo kliknite na tlačidlo „Apply“. Ak sa nezobrazia žiadne pakety, znamená to, že žiadne pakety daného typu neboli zachytené.

Technológie LAN sietí – Ethernet

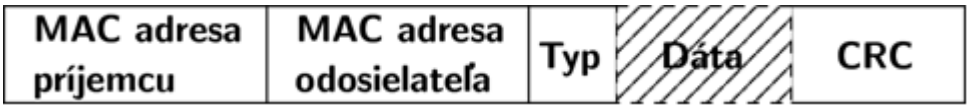
Najpoužívanejšou linkovou technológiou lokálnych sietí je v súčasnosti technológia Ethernet. V priebehu času vzniklo mnoho jej variantov líšiacich sa predovšetkým prenosovou rýchlosťou a použitým prenosovým médiom. V dnešných inštaláciách sa väčšinou používajú varianty s prenosovými rýchlosťami 100 $Mb.s^{-1}$ a 1 $Gb.s^{-1}$ s využitím metalickej kabeláže.

Technológia Ethernet vznikla v 70. rokoch 20. storočia vo firme Xerox, neskôr ju zdokonalilo konzorcium spoločností Digital, Intel a Xerox. Od 80. rokov 20. storočia prevzal na seba vývoj a štandardizáciu inštitút IEEE. Technológia Ethernet patrí medzi technológie 1. a 2. vrstvy RM OSI.

K sieti typu Ethernet sa uzol pripája pomocou svojho sieťového rozhrania (sieťovej karty). Každá ethernetová sieťová karta má od svojho výrobcu pridelenú unikátnu číselnú adresu, ktorá je potrebná pri komunikácii uzlov v spoločnej ethernetovej sieti. Táto adresa sa nazýva **MAC adresa** (Media Access Control) a je tvorená 6-bajtovým číslom. Typicky sa toto číslo uvádza v hexadecimálnom tvare, napríklad 00:0F:EA:7D:A9:68. Prvé tri bajty, v tomto prípade 00:0F:EA, sa nazývajú OUI (Organizationally Unique ID) a identifikujú výrobcu sieťovej karty. Zostávajúce tri bajty, v tomto prípade 7D:A9:68, sú poradové číslo karty vyrobenej daným výrobcom. Každá sieťová karta spracúva len tie rámce, ktoré sú adresované na jej MAC adresu, ostatné ignoruje. Výnimku z tohto pravidla tvoria rámce, ktoré sú adresované príjemcovi s MAC adresou FF:FF:FF:FF:FF:FF. Túto špeciálnu MAC adresu nazývame **adresa obežníka** (**broadcast address**) a rámce s týmto adresátom nazývame linkovými obežníkmi (link-layer broadcast). Linkový obežník musí spracovať *každá* sieťová karta, ktorá ho príjme. Pomocou obežníkov je možné doručovať správy všetkým uzlom siete.

Register hodnôt OUI takisto riadi spoločnosť IEEE. Každý výrobca ethernetových sieťových kariet musí požiadať IEEE o unikátne OUI.

Všetky varianty siete Ethernet využívajú zhodný formát rámca tohto tvaru:



Obrázok 2: Formát rámca typu Ethernet

Význam jednotlivých polí:

- **MAC adresa príjemcu**: adresát. Nachádza sa zámerne na začiatku rámca, aby ju bolo možné čo najrýchlejšie nájsť a spracovať.
- **MAC adresa odosielateľa**: autor rámca.
- **Typ**: Hodnota v tomto poli identifikuje typ dát v dátovej časti rámca. Jednotlivé hodnoty tohto typov a ich významy sú dohodnuté, napríklad číslo 0x0800 udáva, že v dátovej časti sa nachádza IP paket.
- **Dáta**: V tejto časti sa prenášajú samotné dáta.
- **CRC**: Toto pole obsahuje kontrolný súčet celého rámca. Príjemca si vypočíta vlastný kontrolný súčet a porovná ho s hodnotou tohto poľa. Ak sa hodnoty nezhodujú, znamená to, že rámec sa nepreniesol správne a príjemca ho zahodí. Ethernet nezabezpečuje nijakú opravu dát ani opätovné odoslanie.

Na nasledujúcom obrázku získanom z programu Wireshark je zobrazená hlavička ethernetového rámca – MAC adresa príjemcu, MAC adresa odosielateľa a typ dátovej časti. Za typom nasleduje samotná dátová časť rámca, ktorá v tomto prípade obsahuje IP paket, ten v sebe obsahuje TCP segment a vo vnútri TCP segmentu sa nachádza správa protokolu POP3. Kontrolný súčet rámca nie je zobrazený.

```
⊞ Frame 8 (72 bytes on wire, 72 bytes captured)
⊟ Ethernet II, Src: 00:1a:6d:e9:3c:9c (00:1a:6d:e9:3c:9c), Dst: 04:4b:80:80:80:03 (04:4b:80:80:80:03)
  ⊞ Destination: 04:4b:80:80:80:03 (04:4b:80:80:80:03)
  ⊞ Source: 00:1a:6d:e9:3c:9c (00:1a:6d:e9:3c:9c)
    Type: IP (0x0800)
⊞ Internet Protocol, Src: 158.193.134.10 (158.193.134.10), Dst: 158.193.152.140 (158.193.152.140)
⊞ Transmission Control Protocol
⊞ Post Office Protocol
```

Obrázok 3: Rámec typu Ethernet

Meno **prepínač** je odvodené od intuitívnej predstavy, že pri prenose rámca prepínačom sa dočasne prepojí vstupný port s výstupným, a že na prepínači teda dochádza k prepínaniu vstupných a výstupných portov. Prepínač v skutočnosti pracuje na princípe uloženia rámca vo vyrovnávacej pamäti, určení výstupného portu a odoslaní tohto rámca výstupným portom.

Ak sa prepájajú viaceré prepínače navzájom, je dôležité, aby neboli prepojené do kruhu (do slučky), inak sa v tomto kruhu budú napríklad obežníkové rámce prenášať donekonečna a spotrebujú veľkú časť prenosovej kapacity sieťových prvkov. Inteligentné prepínače obsahujú automatickú detekciu a elimináciu slučiek (tzv. protokol Spanning Tree Protocol alebo STP). Detekcia a eliminácie slučiek však nie je bežnou súčasťou všetkých prepínačov.

Kľúčovým stavebným prvkom ethernetových sietí je ethernetový **prepínač**. Prepínač je zariadenie obsahujúce istý počet ethernetových portov (spravidla od 4 do 48), ku ktorým sa pripájajú jednotlivé uzly siete. Úlohou prepínača je vytvárať spoločné komunikačné médium a umožniť pripojeným uzlom komunikovať navzájom.

Prepínač si vedie *tabuľku* svojich portov a MAC adries jednotlivých pripojených staníc. Túto tabuľku si prepínač vytvára z prijatých rámcov a používa ju na doručovanie rámcov k ich príjemcom. Keď niektorým portom vojde rámec, prepínač z tohto rámca prečíta MAC adresu odosielateľa a zaznačí si ju do tabuľky k portu, ktorým rámec vošiel. Týmto spôsobom sa prepínač *učí*, aké adresy majú stanice na jednotlivých jeho portoch.

Ďalej prepínač z rámca prečíta MAC adresu príjemcu a pokúsi sa v tabuľke nájsť port, na ktorom je príjemca pripojený. Ak sa prepínaču podarí vo svojej tabuľke taký port nájsť, odošle rámec von iba týmto portom, inak rozpošle rámec na všetky porty okrem toho, ktorým rámec vošiel.

Prepínač sa teda snaží doručiť rámec podľa možnosti iba na ten port, na ktorom je pripojený adresát. Ak je adresát neznámy, prepínač rozpošle rámec na všetky ostatné porty v dôvere, že adresát – hoci ho nepozná – je pripojený na niektorom z nich. Prepínač je takisto vybavený vyrovnávacou pamäťou, do ktorej sa dočasne ukladajú rámce čakajúce na odoslanie, ak ich nie je možné odoslať okamžite. Pre pripojené uzly je činnosť prepínača úplne neviditeľná. Vo väčších sieťach je možné prepojiť viaceré prepínače navzájom.

Krátky popis bezdrôtových technológií je uvedený v e-learningovej podpore.

Čo sme sa naučili

Technológia Ethernet je dnes najpoužívanejšou technológiou lokálnych sietí. Sieťové karty majú výrobcom nastavenú 6B MAC adresu, ktorú používajú pri vzájomnej komunikácii. Každý rámec typu Ethernet obsahuje MAC adresu príjemcu a odosielateľa, typ prenášaných dát, samotné dáta a kontrolný súčet. Stanice v sieti Ethernet sa pripájajú k prepínaču, ktorý zabezpečuje vzájomné prepojenie medzi stanicami.

Zadanie 2

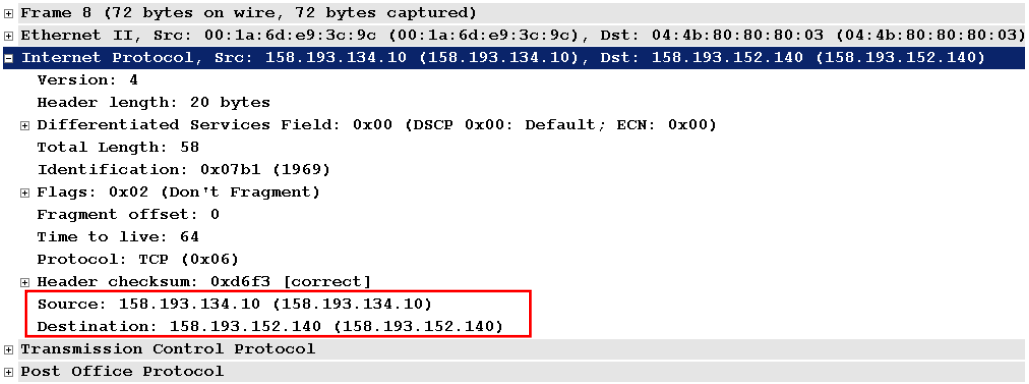
V programe Wireshark zachyťte sieťovú prevádzku, alebo využite dáta zachytené z predchádzajúceho cvičenia. Použitím funkcie Filter si pomocou výrazu **ip** nechajte zobraziť iba rámce nesúce IP pakety. V zachytených rámcoch si detailne všimnite ich štruktúru – MAC adresy príjemcu a odosielateľa, typ obsahu dátovej časti a samotnú dátovú časť. Koľko bajtov zaberá hlavička ethernetového rámca? (6B adresát, 6B odosielateľ, 2B typ, celkovo 14B)

Rodina protokolov TCP/IP

Protokol IP

O význame sieťového protokolu sme sa už zmienili v kapitole o RM OSI. Na tomto mieste si bližšie predstavíme najpoužívanejší sieťový protokol – Internet Protocol (IP) verzie 4.

Protokol IP poskytuje službu prenosu paketov medzi koncovými komunikujúcimi uzlami. IP je tzv. *best-effort* protokol – nerieši otázky garantovaného doručenia, spoľahlivosti, spojovo orientovaného prenosu alebo riadenia toku dát.



Obrázok 4: Hlavička IP paketu

Na obrázku je znázornená hlavička IP paketu, ako ju zobrazuje program Wireshark. Z početných údajov, ktoré obsahuje, sú pre nás podstatné najmä IP adresa odosielateľa a príjemcu, ktoré sa nachádzajú na konci hlavičky. Za nimi nasleduje dátové pole IP paketu. V dátovom poli zobrazeného paketu sa nachádza TCP segment a v TCP segmente správa protokolu POP3.

Zadanie 3	Otvorte si okno príkazového riadku a zistite MAC a IP adresu vašej pracovnej stanice. Pod operačným systémom Windows použite príkaz **ipconfig /all** (nevšímajte si výpisy rozhraní s názvom Tunnel). Pod operačným systémom GNU/Linux použite príkaz **/sbin/ifconfig**.

Zadanie 4	V programe Wireshark zachyťte sieťovú prevádzku, alebo využite dáta zachytené z predchádzajúceho cvičenia. Nechajte si pomocou funkcie Filter použitím výrazu **ip** zobraziť iba IP pakety. Podrobne si všimnite obsah hlavičky IP paketu. Koľko bajtov zaberá IP hlavička? (20B)

Adresovanie v protokole IP

V protokole IP sa ako adresy využívajú **štvorbajtové** čísla, ktoré sa kvôli pohodliu zapisujú v desiatkovej sústave po jednotlivých bajtoch oddelených bodkou, napríklad 87.197.31.42. IP adresy sa prideľujú jednotlivým sieťovým rozhraniam, nie uzlu ako celku. Uzol má teda toľko IP adries, koľko má sieťových rozhraní.

Pri opise sieťovej vrstvy RM OSI sme uviedli, že každá sieťová adresa obsahuje dve časti – **predčíslie siete** a **číslo uzla**. IP adresa tieto dve časti takisto obsahuje, no nie sú bezprostredne viditeľné. Za 30 rokov existencie IP protokolu sa niekoľkokrát

Pôvodný spôsob hľadania predčíslia siete v IP adrese spočíval v tom, že množina všetkých IP adries bola rozdelená na niekoľko tzv. tried, a pre všetky adresy v danej triede bolo stanovené, kde sa v adrese nachádza predčíslie siete a kde číslo uzla. Od používania tohto systému sa upustilo, pretože nebol dostatočne flexibilný a plytval adresami. Podrobnosti o tomto systéme sú popísané v e-learningovej podpore.

zmenil spôsob, ktorým sa v IP adrese tieto dve časti hľadajú.

V súčasnosti sa predčíslie siete v IP adrese určuje tzv. **sieťovou maskou**. Základnou ideou je, že predčíslie siete bude vyjadrené istým počtom *horných bitov* IP adresy. Zostávajúce bity IP adresy budú vyjadrovať číslo uzla. Napríklad môžeme povedať, že v IP adrese 87.197.31.42 je predčíslím siete prvých 29 bitov adresy, a len zostávajúce 3 bity sú číslom uzla v tejto sieti. Takáto sieť obsahuje $2^3 = 8$ adries. Pre názornosť uvádzame túto IP adresu zapísanú v dvojkovej sústave a horných 29 bitov, ktoré podľa dohody tvoria predčíslie, sme označili červenou farbou. Zostávajúce tri bity môžeme pri zachovaní predčíslia ľubovoľne meniť, čím získame skutočne $2^3 = 8$ adries v tej istej sieti (keďže hodnota bitov v predčíslí sa nezmení).

```
01010111.11000101.00011111.00101010 ( = 87.197.31.42 )
```

Sieťová maska je pomocné 32-bitové číslo, ktoré vyjadruje, ktoré bity v IP adrese patria do predčíslia siete a ktoré do čísla uzla. Ak je i-ty bit v maske nastavený na 1, potom bit na tej istej i-tej pozícii v IP adrese patrí do predčíslia siete. Naopak, ak je i-ty bit v maske nastavený na 0, potom i-ty bit v IP adrese patrí do čísla uzla. V binárnom vyjadrení je sieťová maska zľava doprava rad bitov nastavený na 1 (ich počet je zhodný s dĺžkou predčíslia siete v bitoch) a potom rad bitov nastavený na 0 do celkovej dĺžky masky 32 bitov. Ak by sme chceli pomocou masky vyjadriť, že predčíslie siete má mať dĺžku 29 bitov, vytvorili by sme teda 32-bitové číslo, v ktorom horných 29 bitov by bolo nastavených na hodnotu 1 a zvyšné 3 bity na hodnotu 0, čo sa vo formáte IP adresy dá zapísať ako číslo 255.255.255.248, a v binárnom tvare:

```
01010111.11000101.00011111.00101010 ( = 87.197.31.42 )
11111111.11111111.11111111.11111000 ( = 255.255.255.248 )
```

V tomto binárnom zápise je viditeľné, ako maska svojimi 29 hornými bitmi nastavenými na hodnotu 1 vyčleňuje z pôvodnej IP adresy 29-bitové predčíslie.

IP adresa, ktorú získame odpísaním predčíslia siete z danej IP adresy a doplnením nulových bitov do časti s číslom uzla, sa nazýva **IP adresa siete**. IP adresa siete je vlastne najnižšia možná IP adresa s predčíslím danej siete a identifikuje sieť ako celok. Adresu siete nemožno priradiť nijakému konkrétnemu uzlu v danej sieti – je vyhradená len pre sieť. V každej sieti je vyhradené ešte jedno číslo, a síce tzv. **obežníková adresa** (broadcast address), ktorú možno získať odpísaním predčíslia siete a doplnením bitov s hodnotou 1 do časti s číslom uzla. Obežníková adresa je teda najvyššia možná IP adresa s predčíslím danej siete. Na obežníkovej adrese sú povinné počúvať všetky uzly danej siete. Paket odoslaný na obežníkovú adresu siete musia prevziať a spracovať všetky uzly tejto siete. Obežníkovú adresu takisto nemožno prideliť nijakému konkrétnemu uzlu. Počet použiteľných IP adries v sieti je preto vždy o 2 menší než celkový počet adries, pretože najnižšia a najvyššia adresa sú vyhradené (prvá je adresa siete samotnej, druhá je obežníková adresa).

Sieťová maska má popri svojej základnej funkcii – vyjadriť dĺžku predčíslia v IP adrese – ďalšiu dôležitú aplikáciu. Pomocou sieťovej masky sa vypočítava IP adresa siete a obežníková adresa. Zapísané pomocou operátorov jazyka C:

- Adresa siete = (IP adresa) & (maska)
- Obežníková adresa = (IP adresa) | ~(maska)

Adresa siete sa teda získa ako bitový súčin (bitwise AND) medzi IP adresou a maskou. Táto operácia prenesie do výsledku hodnoty všetkých bitov z predčíslia siete, zatiaľ čo bity v čísle uzla vynuluje. Adresa obežníka sa zasa získa ako bitový súčet (bitwise OR) IP adresy a bitového doplnku masky. Tento súčet ponechá bity v predčíslí siete nezmenené, no bity v čísle uzla nastaví na hodnotu 1. Pre príklad, IP adresa 87.197.31.42 s dĺžkou predčíslia 29 bitov patrí do siete 87.197.31.40, pretože 87.197.31.42 & 255.255.255.248 = 87.197.31.40:

```
        01010111.11000101.00011111.00101010 ( = 87.197.31.42 )
AND     11111111.11111111.11111111.11111000 ( = 255.255.255.248 )
        -----------------------------------
        01010111.11000101.00011111.00101010 ( = 87.197.31.40 )
```

a obežníková adresa je 87.197.31.47, pretože
87.197.31.42 | ~(255.255.255.248) = 87.197.31.42 | 0.0.0.7 = 87.197.31.47:

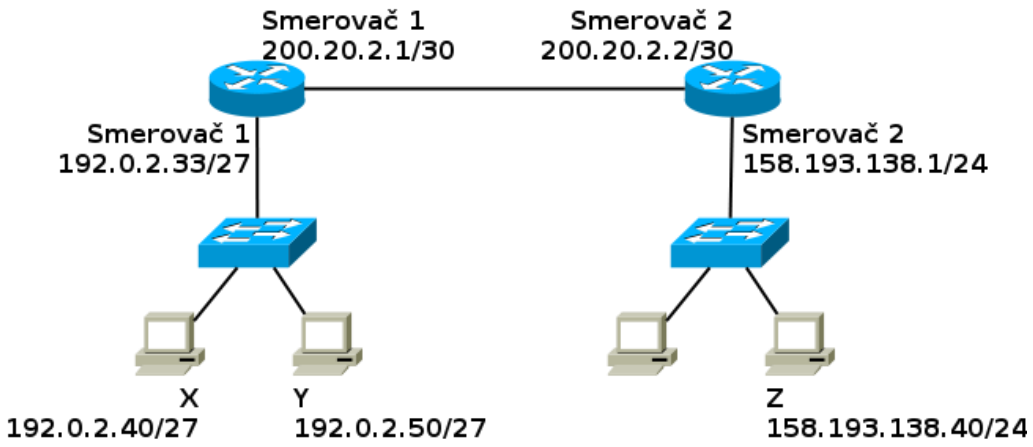
```
~       11111111.11111111.11111111.11111000 ( = 255.255.255.248 )
        -----------------------------------
        00000000.00000000.00000000.00000111 ( = 0.0.0.7)

        01010111.11000101.00011111.00101010 ( = 87.197.31.42 )
OR      00000000.00000000.00000000.00000111 ( = 0.0.0.7)
        -----------------------------------
        01010111.11000101.00011111.00101111 ( = 87.197.31.47 )
```

IP komunikácia uzlov v spoločnej sieti

Predpokladajme, že máme dva počítače X a Y prepojené spoločným médiom (na nasledujúcom obrázku funkcie spoločného média zabezpečuje ethernetový prepínač). Počítač X má IP adresu 192.0.2.40/27 a počítač Y 192.0.2.50/27 (číslo za lomkou vyjadruje dĺžku predčíslia siete). Ďalej predpokladajme, že počítač X chce poslať počítaču Y paket. Čo všetko sa musí udiať?

Zápis masky pomocou lomky a čísla sa niekedy nazýva CIDR formát (Classless Inter-Domain Routing). Tento zápis je pohodlnejší ako zápis celej sieťovej masky po jednotlivých bajtoch.



Obrázok 5: Vzorová sieťová topológia

Predovšetkým, počítač X musí zistiť, či sa Y nachádza *v rovnakej sieti*, a teda na spoločnom médiu. Prečo? Pretože ak sú počítače X a Y susedné, potom môžu komunikovať priamo – jeden môže druhému ihneď poslať rámec linkovej vrstvy a v ňom zabalený paket. Ak by však počítače X a Y boli v rôznych sieťach, potom nie sú na spoločnom médiu a nemôžu komunikovať priamo, a v takom prípade sa o doručenie paketu musia postarať smerovače.

Ako počítač X zistí, či je s počítačom Y na spoločnej sieti? X pozná svoju IP adresu a vie, kde sa v nej nachádza predčíslie siete a akú má hodnotu. Ak je Y v rovnakej sieti ako X, hodnota predčíslia v adrese Y musí byť rovnaká, a teda aj IP adresy sietí vypočítané z adries počítačov X a Y musia byť identické. Podľa tejto úvahy X najprv vypočíta, v akej sieti sa nachádza sám – zoberie svoju IP adresu, vyberie z nej horných 27 bitov, ktoré tvoria predčíslie, a zostávajúce vynuluje – to všetko prostredníctvom binárneho AND medzi vlastnou adresou a maskou:

192.0.2.40 & 255.255.255.224 = 192.0.2.32

X sa teda nachádza v sieti 192.0.2.32/27. Teraz zoberie adresu počítača Y, takisto z nej vyberie horných 27 bitov, ktoré z pohľadu počítača X tvoria predčíslie,

Uzol X v IP adresách hľadá predčíslie dlhé vždy 27 bitov, pretože je tak nastavený. Ak aj cieľový uzol leží v sieti s inou dĺžkou predčíslia, uzol X to nemá ako vedieť, avšak na jeho rozhodnutie to nemá vplyv. To, čo je pre uzol X podstatné, je, či príjemca má alebo nemá rovnaké predčíslie.

a zostávajúce vynuluje, takisto pomocou binárneho AND medzi adresou Y a maskou:

192.0.2.50 & 255.255.255.224 = 192.0.2.32

Y sa teda nachádza v tej istej sieti ako X. Teraz X vie, že môže vytvoriť IP paket s prenášanými dátami, vložiť ho do linkového rámca a odoslať ho priamo počítaču Y.

Je tu však ešte jeden drobný problém – IP pakety budú prenášané vo vnútri rámcov použitej linkovej technológie, lenže tieto rámce takisto musia v hlavičke niesť príslušnú linkovú adresu odosielateľa a adresáta. Ak sú uzly X a Y prepojené sieťou Ethernet, potom X pri odoslaní IP paketu počítaču Y musí poznať aj jeho MAC adresu. Ako ju však zistiť? Treba zdôrazniť, že IP a MAC adresy medzi sebou nijako nesúvisia a jedna sa z druhej nijako nedá výpočtovo odvodiť. Riešenie je prosté – X pošle do svojej siete linkový obežník (t.j. rámec s MAC adresou príjemcu FF:FF:FF:FF:FF:FF), ktorý bude obsahovať správu: „Hľadá sa MAC adresa uzla, ktorý má IP adresu 192.0.2.50“. Tento obežník spracujú všetky uzly v sieti a ak ho príjme aj počítač Y, ktorý je vlastníkom hľadanej adresy, odpovie počítaču X ďalšou správou: „Som to ja, moja MAC adresa je ...“. Ostatné uzly na obežník neodpovedia. Na tomto princípe pracuje pomocný protokol **Address Resolution Protocol** (ARP), ktorý má za úlohu k *známej* IP adrese vyhľadať *neznámu* MAC adresu, pokiaľ sa cieľová stanica nachádza v rovnakej sieti ako odosielateľ. ARP je súčasť rodiny protokolov TCP/IP a používa ho každý uzol pripojený k sieti Ethernet.

Výsledok vyhľadania MAC adresy k známej IP adrese pomocou ARP sa ukladá do tzv. ARP tabuľky, ktorú si vedie každý uzol. Ak je v ARP tabuľke už zaznamenané mapovanie medzi IP a MAC adresou, stanica ho pozná a nemusí posielať ARP otázku.

```
⊞ Frame 4 (42 bytes on wire, 42 bytes captured)
⊞ Ethernet II, Src: 04:4b:80:80:80:03 (04:4b:80:80:80:03), Dst: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    [Is gratuitous: False]
    Sender MAC address: 04:4b:80:80:80:03 (04:4b:80:80:80:03)
    Sender IP address: 158.193.152.140 (158.193.152.140)
    Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
    Target IP address: 158.193.152.44 (158.193.152.44)
```

Obrázok 6: ARP otázka – hľadá sa MAC adresa stanice 158.193.152.44

```
⊞ Frame 5 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: 00:0e:7f:e9:2f:c2 (00:0e:7f:e9:2f:c2), Dst: 04:4b:80:80:80:03 (04:4b:80:80:80:03)
⊟ Address Resolution Protocol (reply)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (0x0002)
    [Is gratuitous: False]
    Sender MAC address: 00:0e:7f:e9:2f:c2 (00:0e:7f:e9:2f:c2)
    Sender IP address: 158.193.152.44 (158.193.152.44)
    Target MAC address: 04:4b:80:80:80:03 (04:4b:80:80:80:03)
    Target IP address: 158.193.152.140 (158.193.152.140)
```

Obrázok 7: ARP odpoveď – stanica 158.193.152.44 má MAC adresu 00:0E:7F:E9:2F:C2

Príkaz ping slúži na odoslanie testovacieho paketu inému počítaču. Tento počítač by mal paket obratom vrátiť nazad. Pomocou príkazu ping je možné otestovať schopnosť obojsmernej komunikácie s iným počítačom.

Zadanie 5

Zobrazte si zistené mapovania IP adries na MAC adresy pomocou príkazu **arp -a** pod OS Windows alebo **/usr/sbin/arp -n** pod OS GNU/Linux.

Spustite v programe Wireshark zachytávanie sieťovej prevádzky. Potom zmažte mapovania príkazom **arp -d** pod OS Windows alebo príkazom **ip neigh flush all** pod OS GNU/Linux (musíte pracovať ako root). Príkazom **ping** overte konektivitu s IP adresou vášho kolegu. Potom v programe Wireshark zastavte zachytávanie a nájdite komunikáciu medzi vašou a kolegovou stanicou (môžete využiť výraz **arp** v riadku Filter pre ARP pakety a výraz **icmp** pre pakety programu ping).

IP komunikácia uzlov v rôznych sieťach, IP smerovanie

Predpokladajme ďalej, že počítač Z (Obrázok 5: Vzorová sieťová topológia) sa nachádza v inej sieti a jeho IP adresa je 158.193.138.40/24. Počítač X má stále IP adresu 192.0.2.40/27 a chce počítaču Z poslať paket.

Postup X na začiatku bude rovnaký ako v predošlom prípade – musí zistiť, či je s počítačom Z v rovnakej sieti. X sa nachádza v sieti s IP adresou

192.0.2.40 & 255.255.255.224 = 192.0.2.32

Rovnakým postupom X vypočíta, v akej sieti sa z jeho pohľadu nachádza Z:

158.193.138.40 & 255.255.255.224 = 158.193.138.32

Uzol X použije svoju dĺžku predčíslia 27 bitov aj pri výpočte siete uzla Z, i keď tá má skutočné predčíslie dlhé 24 bitov. Cieľom uzla X nie je vypočítať skutočnú adresu uzla Z, iba porovnať vlastné predčíslie dlhé 27 bitov s hornými 27 bitmi adresy príjemcu.

Keďže tieto dva výsledky nie sú zhodné, X vie, že príjemca Z sa nenachádza v jeho sieti. Ako mu však paket doručiť? V popise sieťovej vrstvy RM OSI sme uviedli, že viaceré siete sú prepojené osobitným zariadením – smerovačom, ktorého úlohou je preposlať každý prijatý paket správnou cestou do jeho cieľovej siete. Do doručenia paketu príjemcovi Z je teda potrebné zainteresovať smerovač.

Predpokladajme teda, že v sieti počítača X sa nachádza smerovač s IP adresou 192.0.2.33/27. X už zistil, že adresát Z nie je v jeho sieti. X teda uvažuje – ak adresát Z nie je v rovnakej sieti, potom sa o doručenie paketu musí postarať smerovač. Do hlavičky IP paketu však X nemôže ako príjemcu vložiť IP adresu smerovača, pretože by to znamenalo, že paket je určený priamo pre smerovač a nie pre počítač Z. Ako teda postupovať?

Nemá zmysel, aby sa X snažil protokolom ARP zistiť MAC adresu počítača Z, pretože Z nie je v sieti počítača X a žiaden uzol by na otázku neodpovedal.

To, čo X musí v skutočnosti urobiť, je odoslať paket príjemcovi Z, no zariadiť, aby ethernetový rámec, do ktorého bude paket vložený, bol doručený na smerovač. Pretože X a smerovač sú v rovnakej sieti a X pozná jeho IP adresu, nie je preň žiaden problém pomocou ARP protokolu zistiť, akú MAC adresu má smerovač. Teraz už X môže vytvoriť a odoslať paket zabalený do rámca. Jednotlivé adresové údaje budú:

- IP paket: **odosielateľ** = **IP** adresa **X**, **príjemca** = **IP** adresa **Z**
- Rámec: **odosielateľ** = **MAC** adresa **X**, **príjemca** = **MAC** adresa **smerovača**

Odoslaním rámca s paketom podľa týchto adresových informácií sa pre X celá záležitosť s doručovaním paketu skončila.

Poďme sa teraz pozrieť, čo sa udeje na smerovači. Sieťová karta smerovača prevezme rámec, ktorý jej podľa MAC adresy príjemcu patrí, z dátovej časti vyberie vložený paket a odovzdá ho operačnému systému smerovača na spracovanie. Operačný systém smerovača zistí, že paket nie je určený pre smerovač, ale pre uzol Z s adresou 158.193.138.40. Smerovač teda musí paket odoslať ďalej vhodným rozhraním a vhodnému ďalšiemu smerovaču na ceste k príjemcovi Z.

Ako smerovač vie, cez ktoré rozhranie a susedný smerovač je potrebné paket preposlať? Táto znalosť je v každom smerovači uložená v tzv. **smerovacej tabuľke**. Smerovacia tabuľka obsahuje zoznam sietí, o existencii ktorých smerovač vie, ich sieťové masky a informáciu, cez aké výstupné rozhranie, prípadne cez ktorý najbližší smerovač vedú cesty do týchto sietí. Úlohou smerovača pri preposielaní paketu je nájsť v smerovacej tabuľke sieť, do ktorej patrí príjemca paketu, a podľa vyhľadanej informácie paket zodpovedajúcim spôsobom preposlať. Každý smerovač v sieťovej infraštruktúre musí poznať *všetky* cieľové siete.

Oba smerovače v našom príklade musia poznať tri siete – sieť 192.0.2.32/27, v ktorej sa nachádzajú počítače X a Y, ďalej sieť 158.193.138.0/24, v ktorej sa nachádza počítač Z, a sieť 200.20.2.0/30, ktorá prepája oba smerovače navzájom.

Smerovacia tabuľka smerovača 1 bude obsahovať tieto údaje:

Cieľová sieť	Maska siete	Sused alebo rozhranie
200.20.2.0	255.255.255.252	Priamo pripojené
192.0.2.32	255.255.255.224	Priamo pripojené
158.193.138.0	255.255.255.0	200.20.2.2 (smerovač 2)

Smerovacia tabuľka smerovača 2 bude zasa obsahovať tieto údaje:

Cieľová sieť	Maska siete	Sused alebo rozhranie
200.20.2.0	255.255.255.252	Priamo pripojené
192.0.2.32	255.255.255.224	200.20.2.1 (smerovač 1)
158.193.138.0	255.255.255.0	Priamo pripojené

Smerovač operáciou AND medzi IP adresou príjemcu a maskou siete z daného riadku tabuľky vypočíta IP adresu cieľovej siete, pričom dĺžka predčíslia je daná maskou siete. Smerovač teda vyberá z IP adresy príjemcu istý počet horných bitov a porovnáva ich s hodnotou toho istého počtu bitov v adrese siete v danom riadku tabuľky.

Keď smerovač 1 príjme paket adresovaný príjemcovi 158.193.138.40, musí podľa svojej smerovacej tabuľky zistiť, v akej sieti sa príjemca nachádza. Postupne teda prechádza riadkami smerovacej tabuľky, realizuje operáciu binárneho AND medzi IP adresou príjemcu a sieťovou maskou v danom riadku tabuľky a výsledok porovnáva s adresou siete v tom istom riadku tabuľky:

- 158.193.138.40 & 255.255.255.252 = 158.193.138.40
  Nie je zhoda, lebo v 1. riadku je adresa siete 200.20.2.0
- 158.193.138.40 & 255.255.255.224 = 158.193.138.32
  Nie je zhoda, lebo v 2. riadku je adresa siete 192.0.2.32
- 158.193.138.40 & 255.255.255.0 = 158.193.138.0
  Zhoda – toto je cieľová sieť

Týmto spôsobom smerovač 1 zistil, že z jeho pohľadu paket pre príjemcu 158.193.138.40 patrí do siete 158.193.138.0/24. Smerovacia tabuľka v tomto riadku uvádza, že paket je potrebné preposlať na susedný (priamo pripojený) smerovač 2 identifikovaný IP adresou 200.20.2.2. Smerovač 1 teda prostredníctvom ARP protokolu zistí MAC adresu smerovača 2 – bude hľadať MAC adresu toho uzla, ktorého IP adresa je 200.20.2.2. Po jej zistení vytvorí nový rámec, ktorého MAC adresa odosielateľa bude sieťové rozhranie smerovača 1 prepojené so smerovačom 2, MAC adresa príjemcu bude smerovač 2 a v tele rámca sa bude nachádzať pôvodný IP paket s nezmenenými adresami.

Na smerovači 2 sa situácia zopakuje. Smerovač 2 príjme rámec, ktorý mu podľa MAC adresy príjemcu patrí, a z rámca vyberie IP paket. IP adresa príjemcu je adresa počítača Z – 158.193.138.40. Smerovač 2 musí teraz zistiť, do akej cieľovej siete tento paket patrí a komu ho treba preposlať. Identickým spôsobom začne prehľadávať svoju vlastnú smerovaciu tabuľku a zistí:

- 158.193.138.40 & 255.255.255.252 = 158.193.138.40
  Nie je zhoda, lebo v 1. riadku je adresa siete 200.20.2.0
- 158.193.138.40 & 255.255.255.224 = 158.193.138.32
  Nie je zhoda, lebo v 2. riadku je adresa siete 192.0.2.32
- 158.193.138.40 & 255.255.255.0 = 158.193.138.0
  Zhoda – toto je cieľová sieť

Podľa smerovača 2 paket patrí do cieľovej siete 158.193.138.0/24, o ktorej smerovacia tabuľka hovorí, že je priamo pripojená. Na priamo pripojenej sieti sa teda nachádza koncový príjemca – počítač Z. Smerovač 2 sa preto môže priamo opýtať na jeho MAC adresu pomocou protokolu ARP a po jej zistení vytvorí nový rámec, do ktorého vloží pôvodný IP paket. V tomto rámci bude MAC adresa odosielateľa nastavená na MAC adresu sieťovej karty, ktorou je smerovač 2 pripojený do cieľovej siete, a MAC adresa príjemcu bude nastavená na MAC adresu počítača Z. Takto adresovaný rámec bude doručený k počítaču Z, ktorý ho prevezme a spracuje.

Tento opis ilustruje niekoľko dôležitých vlastností IP smerovania:

1. Smerovač má v smerovacej tabuľke *iba zoznam cieľových sietí*, nie zoznam všetkých individuálnych uzlov vo všetkých cieľových sieťach. Tým sa významne znižuje počet záznamov v smerovacej tabuľke bez toho, aby utrpela funkčnosť smerovania.
2. Pri smerovaní paketu sa *IP adresa zdroja a cieľa nemenia*, pretože označujú koncového odosielateľa a adresáta paketu. *Menia sa však linkové adresy*, pretože paket si vždy odovzdáva dvojica priamo prepojených zariadení, pričom paket putuje medzi nimi vždy zabalený v linkovom rámci.
3. Smerovač *nepozná* úplnú cestu do cieľovej siete, má vedomosť iba o *najbližšom ďalšom kroku* na ceste do cieľa.
4. Vyhľadávanie cieľovej siete príjemcu sa opakuje na každom smerovači, ktorým daný paket prechádza. Každý smerovač sa pri smerovaní rozhoduje len sám za seba.
5. Smerovacie tabuľky smerovačov sú usporiadané v poradí od najdlhších predčíslí po najkratšie (čiže od najväčších sieťových masiek po najmenšie). Prehľadávaním tabuľky odhora nadol smerovač hľadá zhodu *v čo najdlhšom predčíslí* siete – tzv. **longest prefix match**.
6. Ak v smerovacej tabuľke nie je možné pre nejaký paket cieľovú sieť nájsť, potom podľa smerovača táto sieť neexistuje a paket bude zahodený.
7. Aby smerovacie tabuľky nemuseli obsahovať všetky existujúce siete, ktorých je v súčasnosti na internete okolo 300 000, vkladá sa do tabuliek zástupný záznam s anglickým názvom default route – sieť 0.0.0.0 s maskou 0.0.0.0, ktorý ukazuje na nejaký nadradený smerovač (spravidla smerovač poskytovateľa internetových služieb).

Tento záznam pri prehľadávaní poskytne zhodu s akoukoľvek IP adresou, pretože pre ľubovoľnú IP adresu platí W.X.Y.Z & 0.0.0.0 = 0.0.0.0. Skutočne teda dokáže zastúpiť všetky siete.

Problematika, ako smerovacie tabuľky napĺňať, je nad rámec tohto materiálu. Je ale vhodné povedať, že obsah smerovacích tabuliek môže stanoviť buď administrátor smerovačov ručne, alebo si smerovače automaticky navzájom vymenia obsahy svojich smerovacích tabuliek a do všetkých objavených sietí vyhľadajú najkratšie cesty. Algoritmy a údajové štruktúry, ktorými sa toto automatické napĺňanie smerovacích tabuliek realizuje, sa nazývajú *smerovacie protokoly*.

Zadanie 6

Príkazom **ipconfig** (Windows) alebo **route -n** (GNU/Linux) zistite, aká je IP adresa brány, ktorú používa váš počítač.

Spustite v programe Wireshark zachytávanie sieťovej prevádzky. Potom zmažte IP/MAC mapovania príkazom **arp -d** pod OS Windows alebo príkazom **ip neigh flush all** pod OS GNU/Linux (musíte pracovať ako root). Príkazom **ping** overte konektivitu s IP adresou **158.193.138.40**. Potom v programe Wireshark zastavte zachytávanie a nájdite komunikáciu medzi vašou stanicou a počítačom 158.193.138.40. V ARP paketoch má vaša stanica hľadať MAC adresu brány. Odchádzajúce pakety typu ICMP PING budú odosielané na adresu 158.193.138.40, no cieľová MAC adresa má byť adresa brány získaná pomocou ARP.

Pomocou príkazu **tracert** (Windows) alebo **traceroute** (GNU/Linux) si vypíšte, cez aké smerovače putujú pakety z vašej stanice na cieľový server **158.193.138.40**

Príkaz tracert resp. traceroute slúži na zistenie, cez ktoré smerovače prechádzajú pakety adresované jednému konkrétnemu príjemcovi.

Čo sme sa naučili

V IP protokole majú uzly (presnejšie, ich rozhrania) pridelené IP adresy v tvare štvorbajtového čísla. Toto číslo sa skladá z dvoch časti – z predčíslia siete, v ktorej sa uzol nachádza, a z čísla uzla. Hranicu medzi predčíslím siete a číslom uzla v IP adrese určuje sieťová maska. Medzi IP adresou a maskou sa využívajú binárne operácie AND a OR na výpočet IP adresy siete, v ktorej sa uzol nachádza, a výpočet obežníkovej adresy v tejto sieti. Vyhľadanie MAC adresy stanice, ktorej IP adresu poznáme a ktorá je v tej istej sieti ako my, zabezpečuje protokol ARP. Komunikácia dvoch staníc v tej istej sieti prebieha bezprostredne. Komunikácia staníc v dvoch sieťach prebieha prostredníctvom smerovačov. Smerovače si udržiavajú v smerovacích tabuľkách zoznamy všetkých existujúcich sietí a každý paket preposielajú na základe informácie vyhľadanej v smerovacej tabuľke.

Budúcnosť IP – protokol IPv6

Protokol IPv4 má už 30 rokov a napriek jeho životaschopnosti už na ňom poznať aj niektoré jeho historicky podmienené obmedzenia. Pripravuje sa preto jeho postupné nahradenie novou verziou IPv6. Niekoľko podrobností o tejto novej generácii protokolu IP nájdete v e-learningovej podpore.

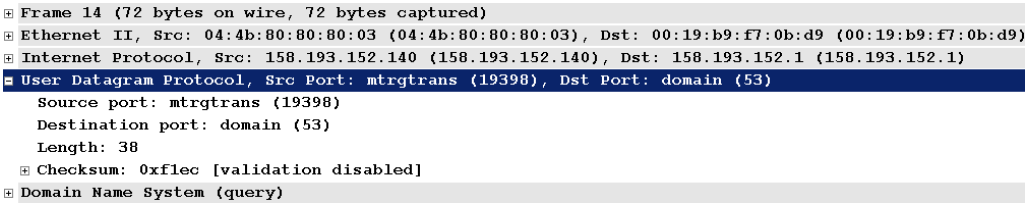
Protokoly TCP a UDP

Ako sme už naznačili v popise vrstiev RM OSI, IP ako protokol sieťovej vrstvy doručí odoslané dáta na cieľový uzol, avšak neponúka žiadne garancie – pakety môžu prísť v nesprávnom poradí, môže dôjsť k ich strate alebo poškodeniu. Naviac, adresa paketu síce označuje odosielateľa a príjemcu, avšak identifikuje len uzly (presnejšie, ich sieťové rozhrania) a nevyjadruje, ktorá aplikácia paket odoslala a pre akú aplikáciu u príjemcu je určený. O doplnenie týchto vlastností sa stará dvojica protokolov transportnej vrstvy: **Transmission Control Protocol** (TCP) a **User Datagram Protocol** (UDP).

Spoločnou vlastnosťou protokolov TCP a UDP je identifikácia odosielajúcej a prijímajúcej aplikácie. Na tento účel je v TCP a UDP zavedený pojem **portu**. Port je číslo v rozsahu od 0 do 65535, ktoré operačný systém uzla v reálnom čase pridelí aplikácii, keď ho požiada o možnosť sieťovo komunikovať. Dvojica portov u odosielateľa a príjemcu identifikuje komunikujúci pár aplikácií. Hlavičky TCP a UDP preto obsahujú informáciu o zdrojovom a cieľovom porte, a tak identifikujú aplikáciu, ktorá daný segment odoslala a pre akú aplikáciu je určený. Aplikácie, ktoré poskytujú služby, t.j. *servery*, majú pridelené všeobecne známe čísla portov. Najbežnejšie sieťové služby a ich porty sú uvedené v nasledujúcej tabuľke:

Sieťová služba	Protokol a port
Odosielanie a doručovanie pošty (SMTP)	TCP/25
World Wide Web (WWW)	TCP/80, TCP/443 (šifrovaný)
Prenos doručenej pošty na lokálny počítač	TCP/110 (POP3), TCP/143 (IMAP)
Domain Name Service (DNS)	TCP/53, UDP/53

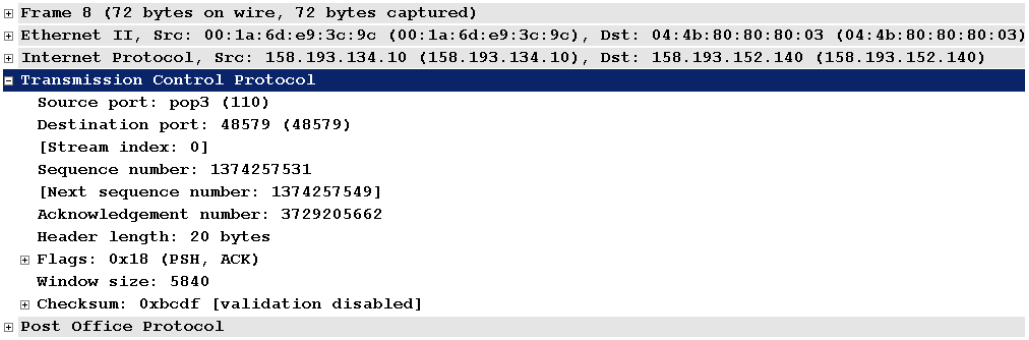
Spoločným konceptom portov sa však podobnosť medzi protokolmi TCP a UDP končí. Protokol UDP (Obrázok 8: Segment protokolu UDP) neposkytuje nijaké ďalšie prídavné vlastnosti – nezabezpečuje ani spoľahlivosť, ani spojovanosť, ani riadenie toku dát. Jeho jedinou funkciou je identifikovať odosielajúcu a prijímajúcu aplikáciu. Napriek týmto zdanlivým nedostatkom je UDP mimoriadne dôležitý protokol, pretože s jeho využitím je možné realizovať broadcasting (odosielanie dát všetkým uzlom v sieti naraz), multicasting (odosielanie dát vybraným uzlom v sieti naraz) a prenosy dát v reálnom čase. Pre žiadnu z týchto aplikácií sa protokol TCP nehodí.



Obrázok 8: Segment protokolu UDP

Protokol TCP (Obrázok 9) sa využíva vo všetkých aplikáciách, ktoré vyžadujú, aby prenos dát bol spoľahlivý, spojovo orientovaný a s riadením toku. Spoľahlivosť v TCP protokole je zabezpečená tým, že všetky TCP segmenty odoslané v priebehu jedného spojenia sú číslované a príjemca segmentov odosiela potvrdenia o ich prijatí. Ak odosielateľ TCP segmentov neprijme do istého času potvrdenie o ich úspešnom doručení, odošle ich znovu, a tento proces opakuje, kým nebude ich správne doručenie potvrdené.

Riadenie toku dát je schopnosť TCP odosielať dáta rýchlejšie alebo pomalšie podľa schopností siete a príjemcu spracovávať ich. Cieľom riadenia toku dát je zamedziť stratám segmentov kvôli zahlteniu siete a pritom dosiahnuť čo najvyššiu priepustnosť. Hoci má už protokol TCP 30 rokov, v oblasti TCP algoritmov na riadenie toku dát prebieha v súčasnosti veľký výskum.



Obrázok 9: Segment protokolu TCP

TCP poskytuje spojovo orientovanú službu prenosu dát. Pred vzájomnou komunikáciou nad TCP protokolom najprv musí klient (softvér využívajúci sieťovú službu) požiadať server (softvér poskytujúci sieťovú službu) o spojenie. Až po odsúhlasení žiadosti o spojenie je možné komunikovať. Spojenie je v prípade TCP dané vzájomnou náväznosťou TCP stavových informácií na oboch komunikujúcich uzloch (poradové čísla paketov, potvrdzovacie čísla). Kedykoľvek počas spojenia môže jedna z komunikujúcich strán odoslať TCP segment druhej strane a má istotu, že tento segment je očakávaný a bude spracovaný. Uzatvorenie spojenia si takisto musia komunikujúce aplikácie navzájom oznámiť.

Čo sme sa naučili

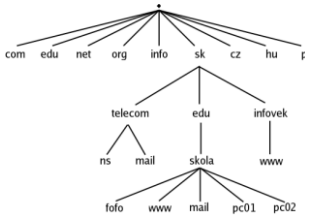
Protokoly TCP a UDP zabezpečujú funkcie transportnej a čiastočne relačnej vrstvy podľa RM OSI. Umožňujú identifikovať komunikujúce aplikácie a jednotlivé ich dialógy pomocou tzv. portov. TCP naviac zabezpečuje funkcie pre spojovo orientovaný prenos, spoľahlivosť a spojovanosť.

Zadanie 7

Spustite v programe Wireshark zachytávanie sieťovej prevádzky. Navštívte niekoľko webových stránok a ukončite zachytávanie. Nechajte si vypísať zachytené TCP (výraz **tcp**) a UDP (výraz **udp**) segmenty. Porovnajte veľkosť TCP/UDP hlavičiek a ich obsah.

Dôležité aplikačné protokoly v TCP/IP

Ako používateľom sieťových služieb sú nám najbližšie protokoly aplikačnej vrstvy. Do tohto modulu sme vybrali tie najpoužívanejšie, ktoré používa prakticky každý z nás. Vzhľadom na potrebný priestor nebolo možné vložiť ich popis do tejto publikácie, avšak bližšie informácie o každom z nich vrátane ukážok ich činnosti nájdete v e-learningovej podpore k tomuto modulu.



Obrázok 10: Strom domén podľa [1]

Domain Name Service

Počítače sa navzájom identifikujú a adresujú pomocou IP adries, no pre používateľov počítačových sietí je pohodlnejšie používať slovné názvy namiesto číselných adries. V roku 1983 vznikla preto služba DNS (Domain Name Service), ktorá plní funkciu „telefónneho zoznamu internetu“ – umožňuje k slovnému menu počítača vyhľadať jeho IP adresu. Činnosť služby DNS je pre používateľov transparentná – operačné systémy ju využívajú na pozadí bez nášho vedomia.

Obrázok 11: Symbol Webu, zdroj: http://www.worldtime.com

Hyper Text Transfer Protocol

Protokol HTTP (Hyper Text Transfer Protocol) sa používa na komunikáciu s webovým serverom. Patrí medzi najbežnejšie textové protokoly, t.j. všetky jeho príkazy sú tvorené textovými riadkami, ktorým je možné rozumieť i bez špecializovaného sieťového analyzátora.

Poštový protokol SMTP

Protokol SMTP (Simple Mail Transport Protocol) sa používa na prenos elektronickej pošty. Pomocou protokolu SMTP poštový program (Outlook, Thunderbird a i.) odovzdá hotovú správu elektronickej pošty svojmu prednastavenému serveru odchádzajúcej pošty. Server odchádzajúcej pošty takisto pomocou SMTP odovzdáva túto správu ďalej cieľovému poštovému serveru, na ktorom sa nachádza poštový priečinok adresáta správy.

Obrázok 12: Symbol elektronickej pošty, zdroj: http://www.library.illinois.edu/doc/images/email.gif

Poštový protokol POP3

Pomocou protokolu POP3 (Post Office Protocol, verzia 3) si klientský poštový program môže vyzdvihnúť doručenú poštu z priečinka používateľa na serveri a preniesť ju na lokálny počítač.

Vytvárame si vlastnú sieť

V tejto kapitole by sme sa radi podelili o praktické skúsenosti a rady pri oživovaní a nastavovaní vlastnej počítačovej siete. O stavbe a konfigurácii sietí sú popísané hrubé knihy a nie je našou ambíciou nahrádzať ich. Napriek tomu by sme radi poukázali na niekoľko pravidiel, ktoré je vhodné pri oživovaní vlastnej siete dodržať, aby bola sieť čím skôr funkčná a spoľahlivá.

Kvôli obmedzenému priestoru v tejto publikácii je text tejto kapitoly k dispozícii v e-learningovej podpore k tomuto modulu.

Čo sme sa naučili v tomto module

Zhrnutie

V tomto module sme si zopakovali a prehĺbili vedomosti o operačných systémoch a počítačových sieťach.

Poznáme základné pojmy teórie operačných systémov a dokážeme naprogramovať jednoduché multithreadové programy ako aj programy narábajúce so štruktúrou adresárov.

Dokážeme používať príkazový riadok OS GNU/Linux aj jeho vizuálne programy na riadenie práva používateľov, monitorovanie bežiacich procesov, prácu so súbormi a adresármi.

Poznáme základné pojmy z oblasti počítačových sietí tak ako ich definuje OSI odel, poznáme princípy fungovania základných protokolov lokálnych sietí a internetu.

Preverenie výstupných vedomostí

Vypracujte jedno z týchto zadaní, prípade zadanie podľa :

Záverečné zadanie 1	Upravte program Adresare z kapitoly 1 tak, aby pre každý adresár napísal do riadku za jeho meno aj počet súborov, ktoré sa nachádzajú priamo v tom adresári a súčet ich veľkosti v bajtoch.
Záverečné zadanie 2	Naprogramujte multithreadovú aplikáciu pre iný náročný výpočtový problém, podobne ako je v 1. Kapitole naprogramovaná aplikácia pre Mandelbrotovu množinu.

Literatúra a použité zdroje

[1] Jakab, F., SIVÝ, I., Bučko, M., Kulich, V., Palúch, P., László, R., Farkaš, M., Feciľak, P., Michálek, J., Genči, J., Janočko, M., Plžík, M., Troppa, Ľ., Krutý, P., Zeman, M.: Príručka správcov počítačových učební. Košice : Elfa, 2004, ISBN 80-89066-89-5

[2] Kabelová, A., Dostálek, L.: Velký průvodce protokoly TCP/IP a systémem DNS; Computer Press; 5. aktualizované vydanie; ISBN 978-80-251-2236-5

[3] Kroah-Hartman G, Corbet J, McPherson A., Linux Kernel Development . Publikované online: http://www.linuxfoundation.org/publications/whowriteslinux.pdf

[4] Silberschatz, A., Galvin, P. B., Gagne, G.: Operating System Concepts. John Wiley & Sons., 2004, ISBN:0471694665

[5] Tanenbaum A. S.: *Structured Computer Organization (5th Edition)*, Pearson Education; 5th edition 2006, ISBN 0-13-148521-0

[6] Tanenbaum, A. S.: Modern Operating Systems (2nd Edition), Prentice Hall; 2 edition (February 28, 2001), ISBN 0130313580

[7] Tomcsányi, P.: Synchronising processes in Imagine Logo: Why and How, In:Proceedings of Eurologo 2007, Bratislava
Dostupné online: http://www.di.unito.it/~barbara/MicRobot/AttiEuroLogo2007/proceedings/P-Tomcsanyi.pdf

[8] http://www.ibm.com/developerworks/linux/library/l-linux-kernel/

[9] http://stuleja.org/vscience/materialy/mandelbrot/F.htm

Tento študijný materiál vznikol ako súčasť národného projektu Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika v rámci Aktivity „Ďalšie vzdelávanie kvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ“.

Autori © RNDr. Peter Tomcsányi
prof. Ing. Miloš Šrámek, PhD.
Ing. Peter Palúch, PhD.

Názov Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Podnázov Operačné systémy a počítačové siete

Študijný materiál prešiel recenzným pokračovaním.

Recenzenti doc. Ing. Fabián Peter, CSc
doc. Ing. Ľudovít Trajteľ, PhD.

Počet strán 36

Náklad 400 ks

Prvé vydanie, Bratislava 2010

Vydal Štátny pedagogický ústav, Pluhová 8, 830 00 Bratislava, v súčinnosti s Univerzitou Pavla Jozefa Šafárika v Košiciach, Univerzitou Komenského v Bratislave, Univerzitou Konštantína Filozofa v Nitre, Univerzitou Mateja Bela v Banskej Bystrici a Žilinskou univerzitou v Žiline

Vytlačil BRATIA SABOVCI, s r.o., Zvolen

ISBN 978-80-8118-031-6