

Architektura OS Linux

Martin Opletal

Bakalářská práce
2006



Univerzita Tomáše Bati ve Zlíně
Fakulta Aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav aplikované informatiky

akademický rok: 2005/2006

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Martin OPLETAL**

Studijní program: **B 3902 Inženýrská informatika**

Studijní obor: **Informační technologie**

Téma práce: **Architektura operačního systému Linux**

Zásady pro vypracování:

Vypracujte studijní materiály na zadané téma.

Popište následující součásti operačního systému

Správa procesů

Správa paměti

Správa souborového systému

Zabezpečení operačního systému Linux

Vytvořte zjednodušenou HTML verzi vhodnou pro publikování na internetu.

Použijte dodanou šablonu pro jednotný vzhled.

Navrhněte soubor zkouškových otázek.

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

Deitel, H. M.: Operating Systems, Prentice Hall, 2004

Klímeš, C.: Operační systémy 1. Ostravská univerzita v Ostravě, Katedra informatiky a počítačů.

Tanenbaum, A. S.: Modern operating systems, Prentice Hall, 2002

Linux – Dokumentační projekt, Computer Press, 2003

Vedoucí bakalářské práce:

Ing. Martin Sysel, Ph.D.

Ústav aplikované informatiky

Datum zadání bakalářské práce:

14. února 2006

Termín odevzdání bakalářské práce:

16. června 2006

Ve Zlíně dne 14. února 2006


prof. Ing. Vladimír Vašek, CSc.
pověřený děkan




doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

Prohlašuji, že jsem na celé diplomové práci pracoval/a samostatně a použitou literaturu jsem citoval/a.

Ve Zlíně, 03. 09. 2006

.....

jméno diplomanta

Abstrakt

Cílem této bakalářské práce je popsat architekturu operačního systému Linux. Text je rozdělen do šesti kapitol. V první kapitole popisují historii Unixu a Linuxu. Ve druhé kapitole popisují základní práci v Linuxu (přihlášení, odhlášení, atd.). V další kapitole popisují práci se soubory a adresáři. Ve čtvrté a páté kapitole se zabývám správou procesů a správou paměti. Poslední kapitola se zabývá bezpečností.

Klíčová slova: Linux, Unix, historie, soubor, adresář, správa procesů, správa paměti, bezpečnost

Abstract

The aim of this bachelor thesis is to describe the architecture of the operating system Linux. Text is divided into six sections. In the first section I describe history of Unix and Linux. In the second section I describe basic work in Linux (login, logout, etc.). In the next section I describe work with files and directories. In the fourth and the fifth sections I deal with processes management and memory management. The last section deals with security.

Keywords: Linux, Unix, history, file, directory, processes management, memory management, security

OBSAH

ÚVOD	6
1 HISTORIE SYSTÉMU	7
1.1 HISTORIE OS UNIX	7
1.2 HISTORIE OS LINUX	7
2 ZÁKLADY PRÁCE V SYSTÉMU	8
2.1 PŘIHLÁŠENÍ UŽIVATELE	8
2.1.1 Lokální přihlášení	8
2.1.2 Vzdálené přihlášení	9
2.2 PRVNÍ KROKY	9
2.3 SYSTÉMOVÁ DOKUMENTACE	10
2.4 ODHLÁŠENÍ UŽIVATELE	11
3 SOUBOROVÝ SYSTÉM	12
3.1 OBYČEJNÉ SOUBORY A ADRESÁŘE	12
3.2 SPECIÁLNÍ ADRESÁŘE	12
3.3 PRÁCE SE SOUBORY A ADRESÁŘI	13
3.4 SPECIÁLNÍ SOUBORY	13
3.5 ODKAZY	14
3.6 PŘÍSTUPOVÁ PRÁVA	15
3.6.1 Změna přístupových práv	16
3.6.2 Uživatelská maska	18
3.7 ZÁKLADNÍ ADRESÁŘOVÁ STRUKTURA SYSTÉMU	18
4 SPRÁVA PAMĚTI	19
4.1 VIRTUÁLNÍ PAMĚŤ	20
4.1.1 Funkce virtuální paměti	20
4.1.2 Tabulky stránek Linuxu	21
4.2 ALOKACE A DEALOKACE STRÁNEK	22
4.3 VYROVNÁVACÍ PAMĚŤ	23
5 SYSTÉM PROCESŮ	26
5.1 VZNIK PROCESU	26
5.2 ŽIVOT PROCESU	27
5.2.1 Běh procesů na popředí a na pozadí	29
5.2.2 Změna priority procesů	30
5.3 UKONČENÍ ŽIVOTA PROCESU	31
6 BEZPEČNOST SYSTÉMU	33
6.1 LOKÁLNÍ BEZPEČNOST	33
6.2 SÍŤOVÁ BEZPEČNOST	35
ZÁVĚR	38
SEZNAM POUŽITÝCH ZDROJŮ	40
SEZNAM OBRÁZKŮ	41
SEZNAM TABULEK	42
SEZNAM PŘÍLOH	43

ÚVOD

Operační systém

Operační systém je základní programové vybavení počítače, které se stará o správu systémových zdrojů. Za systémové zdroje jsou obvykle považovány výpočetní čas procesoru, dostupná paměť, diskový prostor, přístup k síťovým rozhraním a další. Jednotlivé zdroje jsou uživatelským úlohám přidělovány buď automaticky nebo na základě jejich požadavků. Všem úlohám poskytuje operační systém jednotné softwarové rozhraní, přes které mohou žádat o přidělení systémových zdrojů. Dále softwarové rozhraní umožňuje uživatelským úlohám komunikaci s periferním hardwarem. Operační systém komunikuje s hardwarem pomocí ovladačů. Ty jsou zahrnuty buď přímo v operačním systému nebo jsou dodávány formou dodatečných modulů.

Základní části operačních systémů

Základem všech operačních systémů je jádro, které poskytuje uživatelským programům jisté množství služeb. Z hlediska konstrukce jádra a množství poskytovaných služeb se jádra dělí na několik kategorií.

Operační systémy s monolitickým jádrem – na jejich bázi je sestavena většina současných operačních systémů. Jádro zahrnuje služby, které se týkají správy paměti, správy úloh a periferií, grafického rozhraní.

Mikrojádrové systémy – zpravidla obstarávají pouze správu úloh a jejich vzájemnou komunikaci. Ostatní služby obstarávají dodatečné moduly.

S jádrem systému nelze komunikovat přímo. Komunikaci s uživatelem zajišťuje uživatelské rozhraní. To může nebo nemusí být integrováno přímo do jádra systému (ve většině vyspělých operačních systémech není jeho částí).

1 HISTORIE SYSTÉMU

1.1 Historie OS Unix

V roce 1965 společnosti Bell Telephone Laboratories a General Electric začaly spolupracovat na vytvoření nového operačního systému MULTICS. Později se společnost Bell Telephone Laboratories rozhodla od spolupráce odstoupit. Skupina programátorů pod vedením pánů K. Thompsona, D. Ritchieho a B. Kernighana navrhla vlastní multiuživatelský a multiprocesorový operační systém, který byl pojmenován – UNIX. Tento systém byl v počátku vytvořen v assembleru na počítači PDP-7. V roce 1973 byl kompletně přepsán do programovacího jazyka C a díky tomu se stal přenositelným i na jiné počítače, což se poprvé stalo v roce 1977. Koncem sedmdesátých let se UNIX dostal na některé univerzity, později se začal prosazovat i v komerční sféře. Další vývoj probíhal jak v Bellových laboratořích, tak i na Kalifornské univerzitě v Berkeley (UCB). Dnes existují dvě základní varianty systému: Systém V z Bellových laboratoří a BSD (Berkeley System Distribution) z univerzity v Berkeley. Mimo tyto dvě základní verze existuje spousta dalších verzí operačního systému UNIX, které jsou z nich odvozeny.

1.2 Historie OS Linux

Linux je volně šířitelný operační systém, který vychází z koncepce UNIX. Lze si jej zdarma pořídit např. ze sítě Internet, popřípadě si zakoupit některou z komerčních distribucí. Autorem Linuxu je pan Linus Torvald. Původní verze byla v průběhu let zdokonalována mnoha lidmi na celém světě. Operační systém Linux podporuje většinu programového vybavení, které bylo napsáno pro systém UNIX.

2 ZÁKLADY PRÁCE V SYSTÉMU

2.1 Přihlášení uživatele

Než zahájíme práci v systému, je třeba se nejprve identifikovat, nebo-li „přihlásit se do systému“, pomocí uživatelského jména a hesla. Systémy Unix rozlišují dva uživatelské režimy.

Jednouživatelský režim (singleuser mode)

Běžný uživatel se s ním nesetká. Je používán v průběhu startu systému nebo se do něj systém může přepnout při kritických chybách. Tím je znemožněna práce běžným uživatelům a další kroky jsou již na administrátorovi systému.

Víceuživatelský režim (multiuser mode)

V tomto režimu mohou pracovat i běžní uživatelé. Každý uživatel má přidělen svůj účet (**account**), jehož součástí jsou základní informace o uživateli. Ne všichni uživatelé jsou si v systému rovni. Můžeme je rozdělit do tří skupin:

Běžní uživatelé – jsou skutečnými fyzickými uživateli systému (studenti, zaměstnanci apod.).

Systémoví uživatelé – nerepresentují fyzické osoby. V systému mají speciální význam a zpravidla jsou nezbytní pro správný a bezpečný běh systémových služeb. Mezi typické systémové uživatele patří např. daemon, mail, news, nobody a další.

Superuživatel root – má neomezená práva. Slouží k administraci systému, instalaci softwaru, správě uživatelských účtů apod.

Při přihlašování do systému, mohou nastat dvě situace. Uživatel je fyzicky přítomen u počítače (terminálu) a s počítačem bude pracovat přímo. Takové přihlášení nazýváme lokální přihlášení (**local login**). Další možností je pracovat s počítačem na dálku prostřednictvím počítačové sítě a jiného počítače (terminálu), u kterého je uživatel fyzicky přítomen. Takovému přihlášení říkáme vzdálené přihlášení (**remote login**).

2.1.1 Lokální přihlášení

U lokálního přihlašování je vždy třeba provést tzv. **autentizaci** uživatele. Děje se tak pomocí uživatelského jména a hesla. Nejprve systém vyzve uživatele aby vložil své uživatelské jméno (**Login:**). Po vložení a stisknutí klávesy **ENTER** je třeba zadat heslo

(**Password:**). Heslo se při vkládání z bezpečnostních důvodů nevypisuje. Je také třeba dát pozor na to, že systém rozlišuje malá a velká písmena. Jestliže uživatel udělal chybu při vkládání uživatelského jména nebo hesla, systém jej na to upozorní chybovým hlášením - **Login incorrect**.

Heslo je možné změnit příkazem **passwd**.

```
Př.: $ passwd

Old password:

New password:

Retype new password:

Password changed.
```

Po zadání příkazu je třeba nejprve zadat staré heslo. Po jeho správném zadání systém žádá vložení nového hesla. To se pro kontrolu zadává dvakrát. Stejně jako při přihlašování, ani v tomto případě se hesla nevypisují na obrazovku. Na závěr systém vypíše hlášení, že heslo bylo změněno.

2.1.2 Vzdálené přihlášení

Princip vzdáleného přihlášení je obdobný lokálnímu přihlášení, ovšem s tím rozdílem, že uživatele od počítače dělí počítačová síť (Internet, lokální počítačová síť). Aby bylo přihlášení uživatele úspěšné, je třeba splnit dvě podmínky. Uživatel musí mít na vzdáleném počítači platný uživatelský účet. Zároveň je třeba, aby bylo funkční síťové spojení a oboustranná podpora některé ze služeb vzdáleného přihlašování. Na počítači, ke kterému se chce uživatel připojit, musí být spuštěn program zajišťující vzdálené připojení - **remote login daemon**, který komunikuje s počítačem uživatele pomocí speciálního klientského programu. Ten nejprve požádá o spojení, které může být vzdáleným počítačem odmítnuto. Je-li spojení úspěšné a uživatel má přístup povolen, je další práce se vzdáleným systémem stejná jako při lokálním přihlášení.

2.2 První kroky

Po úspěšném vstupu do systému se uživateli zobrazí informace o jeho posledním přihlášení, o verzi systému a další systémové informace. Může se objevit tzv. zpráva dne, jejíž znění zadává správce systému a je společná všem uživatelům. Po těchto informacích systém informuje uživatele o přijaté poště. Dále se zobrazí **výzva příkazového řádku**.

Tato výzva se nazývá - **prompt**. Je to posloupnost symbolů, kterou vypisuje příkazový interpret. Tvar promptu si může uživatel upravit. Je ukončen znakem \$ - běžný uživatel nebo znakem # - superuživatel root. Prompt ohlašuje že uživatel může zadat posloupnost příkazů. Systém rozlišuje malá a velká písmena!

Příkazový interpret

Příkazový interpret (**shell**) přijímá a realizuje příkazy. Mezi nejpoužívanější příkazy shellu patří příkaz pro spuštění programu. Příkaz se skládá ze slov oddělených mezerami. První slovo příkazu se musí shodovat se jménem vestavěného příkazu interpretů nebo se jménem spustitelného souboru, jinak by shell vypsál chybové hlášení. Je-li příkaz složen z více slov, označujeme tato slova jako **argumenty**. V příkazu se také může vyskytnout **přepínač**, který začíná znakem minus.

Operační systém Linux se distribuuje s příkazovým interpretem **bash**.

Tab.1. Základní klávesové zkratky

klávesa	význam
Ctrl a	skoč na začátek řádku
Ctrl e	skoč na konec řádku
Ctrl d	smaž znak na pozici kurzoru
Ctrl u	smaž znaky od začátku řádku do pozice kurzoru
Ctrl k	smaž znaky od pozice kurzoru do konce řádku
Ctrl t	zaměň poslední dva znaky a posuň se vpravo
Ctrl r	hledej v historii příkazů
Ctrl _	krok zpět
Ctrl y	vlož smazaný text na pozici kurzoru
Ctrl l	vyčisti obrazovku
Levý Alt <	najdi poslední příkaz
Levý Alt >	najdi první příkaz
Levý Alt l	převed' slovo na malá písmena
Levý Alt u	převed' slovo na velká písmena

2.3 Systémová dokumentace

V systému jsou všechny dostupné nástroje systematicky dokumentovány. Uživatel má možnost používat hned několik prohlížečů dokumentace. Základní informace o obslužných programech, funkcích programovacích jazyků nebo o voláních jádra se dají získat pomocí manuálových stránek. Manuálové stránky jsou rozděleny do sekcí, lze je prohlížet pomocí programů **apropos**, **whatis** a **man**.

2.4 Odhlášení uživatele

Pro ukončení práce s počítačem nestačí jej pouze vypnout, protože by mohlo dojít ke ztrátě dat nebo k poškození důležitých souborů. Je to způsobeno tím, že Linux používá pro diskové operace **paměť cache**. Před ukončením práce s počítačem je proto třeba nejdříve se odhlásit. K tomuto účelu slouží příkaz **logout**. Tím je práce ukončena a může se přihlásit další uživatel.

Pracuje-li s počítačem jediný uživatel, může jej vypnout. Je třeba se přihlásit jako uživatel se speciálním jménem účtu root. Běh Linuxu se ukončí příkazem **shutdown**. Jestliže uživatel chce, aby se systém vypnul se zpožděním, je třeba jej zadat ve tvaru:

shutdown -h +číslo

Ukončení systému pak bude zpožděno o zadané číslo v minutách. [4]

3 SOUBOROVÝ SYSTÉM

3.1 Obyčejné soubory a adresáře

Každý soubor se skládá ze **jména souboru**, **administrativních informací** a **obsahu**. Administrativní informace jsou uloženy v tzv. **i-uzlu (i-node)**, který je jednoznačným identifikátorem souboru. V i-uzlu jsou uloženy délka souboru, informace o vlastníkovi souboru, čas vzniku a modifikace souboru, přístupová práva a další. V i-uzlu není uloženo jméno souboru. [6]

V Linuxu je zavedena hierarchická struktura souborů, kterou nazýváme **strom**. Díky tomu lze soubory shlukovat do logických celků a lze je jednoznačně identifikovat a vyhledávat. Na vrcholu stromu stojí kořenový adresář **root**, značí se „/“. Každý soubor ve stromu je jednoznačně určen absolutní cestou, což je posloupnost jmen souborů a adresářů. Jsou mezi sebou odděleny znakem „/“. Posloupnost začíná kořenovým adresářem a končí daným souborem.

Jméno souboru se může skládat z libovolných znaků. Některé znaky ovšem mohou způsobovat problémy. Jméno souboru není rozděleno na jméno a příponu jako u jiných operačních systémů. Nemělo by začínat znakem minus. Je-li na začátku jména souboru tečka, znamená to, že jde o **neviditelný soubor**. Jsou to soubory, které není třeba mít neustále „na očích“. Nejedná se o ochranu před nepovolanými osobami.

3.2 Speciální adresáře

Čím více je hierarchie adresářů hlubší, tím je delší absolutní cesta. Při práci by pak nebylo praktické odkazovat se na každý soubor pomocí absolutní cesty. Proto je dobré si v danou chvíli označit jeden adresář jako pracovní a všechny soubory uvažovat relativně k němu. Pracovní adresář pak plní funkci pracovního kořenu. Tento adresář se nastavuje příkazem **cd**. Na soubory v pracovním adresáři se lze odkazovat uvedením jména bez cesty. Lze se v něm odkazovat i na soubory (adresáře) v jeho podadresářích, uvedením cesty pouze z pracovního adresáře bez lomítka na začátku. Pro odkaz na soubory (adresáře) v nadřazeném adresáři se používá speciální jméno „..“. Toto speciální jméno se uvádí na začátku cesty. Pro odkaz na pracovní adresář se používá „.“.

3.3 Práce se soubory a adresáři

Adresáře

Pro výpis obsahu adresáře slouží příkaz **ls**. Zadáme-li jej bez parametrů, vypíše jména všech souborů a podadresářů, které se nacházejí v aktuálním adresáři, a jejichž jméno nezačíná tečkou. Chceme-li vypsát jména všech souborů a adresářů, je třeba použít přepínač **-a**. Použitím přepínače **-l** se navíc zobrazí spolu se jmény souborů také informace o nich. Při použití přepínače **-F** budou adresáře ukončeny „/“.

Pro vytvoření adresáře slouží příkaz **mkdir**, pro smazání příkaz **rmdir**. Smazat lze jen prázdný adresář. Pro zrušení adresáře s obsahem je nutno použít přepínač **-r**.

Soubory

Pro kopírování a přesun souborů slouží příkaz **cp** a příkaz **mv**.

cp <soubor> <kam> - zkopíruje vyjmenované soubory do zadaného adresáře

mv <soubor> <kam> - přemístí vyjmenované soubory do zadaného adresáře. Tento příkaz lze použít i pro přejmenování souboru.

Pro smazání souborů se používá příkaz **rm**, který smaže zadaný soubor. V Linuxu není možnost smazaný soubor obnovit.

Všechny předchozí uvedené příkazy mají množství různých přepínačů. Více se o nich lze dozvědět z manuálových stránek (**man <příkaz>**).

3.4 Speciální soubory

Speciální soubory reprezentují hardwarová zařízení. Neobsahují data, ale jsou spjaty s nějakou rutinou v jádru systému. Ta je volána, jakmile je k souboru přistupováno. Podle stylu práce s periferiemi rozlišujeme dva typy speciálních souborů. Prvním typem jsou soubory pro **bloková zařízení**, druhým typem jsou soubory pro **znaková zařízení**.

Bloková zařízení

Umožňují náhodný přístup k datům, jádro systému si pro ně udržuje vyrovnávací paměti. Operačnímu systému se bloková zařízení jeví jako posloupnost bloků dané délky. Bloková zařízení mohou být nosičem souborového systému, mezi jejich typické představitele patří disk nebo CD-ROM.

Znaková zařízení

Neumožňují uživateli náhodný přístup. Slouží zpravidla k manipulaci s periferiemi jako jsou polohovací zařízení, tiskárny apod.

Operační systém přiřadí ke každému speciálnímu souboru dvě čísla. Hlavní číslo – **major number** a vedlejší číslo – **minor number**. Jádro pak identifikuje obslužnou rutinu na základě těchto dvou čísel. Při identifikaci se neřídí jmény souborů, ty slouží pouze jako odkazy na zařízení pro obslužné programy. Speciální soubory bývají umístěny v adresáři **/dev**. Vytvářet je může pouze superuživatel root pomocí speciálního programu **mknod**. Běžný uživatel k nim má silně omezený přístup. Jednou z výjimek jsou soubory **/dev/zero**, **/dev/null**, **/dev/random** a **/dev/full**.

/dev/zero – speciální soubor, který na požádání vrací samé nuly.

/dev/random – chová se obdobně jako **/dev/zero** s tím rozdílem, že na požádání vrací náhodně vygenerované znaky.

/dev/null – cokoli do něj zkopírujeme, data nenávratně zmizí.

/dev/full – chová se jako plný disk. Při pokusu na něj cokoli zkopírovat dojde k chybě a systém hlásí **No space left on device**. Lze jej využít k simulaci přeplněného disku. [10]

3.5 Odkazy

Dalším typem souboru je odkaz – **link**. Vedle obyčejných souborů a adresářů jde o nejčastěji používané typy souborů. Odkaz je soubor, který zavádí nové jméno pro již existující soubory. Při jeho zavedení nedochází k fyzické kopii dat, pouze se vytvoří nové jméno ukazující na původní soubor. Je vhodné jej použít tam, kde uživatel požaduje přímou dostupnost jednoho souboru z více adresářů. Pomocí odkazů lze zjednodušit procházení hierarchické struktury adresářů a souborů. Odkaz má však vedlejší efekt, struktura souborů a adresářů obecně přestává být hierarchická, absolutní cesta k souboru již nemusí být jednoznačná.

Pevný odkaz – hard link

Lze jej vytvořit pouze na existující soubor a jen v rámci jednoho reálného souborového systému. Ve většině systémů nelze vytvořit pevný odkaz na adresář.

Symbolický odkaz – symbolic link

V podstatě jde o obyčejný soubor, který obsahuje jako data cestu k cílovému souboru. Lze jej zavést i na neexistující soubory nebo soubory ležící na jiném reálném souborovém systému a rovněž i na adresáře. Dojde-li k přesunu nebo k smazání zdrojového souboru, bude symbolický odkaz ukazovat na neexistující soubor a nazývá se **narušený – broken link**.

Oba druhy odkazů se vytvářejí pomocí programu **ln**.

ln <přepínač> <soubor(adresář)> <název odkazu>

Příkaz má dva přepínače. Pro vytvoření symbolického odkazu slouží přepínač **-s** pro vytvoření pevného odkazu slouží přepínač **-f**. Odkaz lze smazat programem **rm**.

3.6 Přístupová práva

V Linuxu má každý soubor svého vlastníka. Tím obvykle bývá ten, kdo soubor vytvořil. Kromě vlastníka má každý soubor přiřazen ještě skupinu. Superuživatel root může měnit vlastníky, skupiny a přístupová práva ke všem souborům v systému.

Uživatele můžeme rozdělit do tří kategorií:

owner - vlastník souboru

group - člen skupiny přiřazené souboru

others - ostatní

Každá z předchozích tří skupin může mít přiřazenou libovolnou kombinaci tří základních přístupových práv:

právo na čtení – **read (r)**

právo na zápis – **write (w)**

právo spouštět soubor – **execute (x)**

Práva u souborů

Read (r) – umožňuje uživateli čtení souboru (jeho zobrazení, kopírování, tisk, apod.).

Write (w) – umožňuje uživateli zápis do souboru, měnit jeho obsah.

Execute (x) – umožňuje uživateli spuštění souboru.

Práva u adresářů

Práva adresářů neovlivňují možnost čtení, zápisu a spouštění souborů. Ovlivňují možnost výpisu obsahu adresáře, vytváření, rušení a přejmenování souborů a podadresářů.

Read (r) – umožňuje uživateli číst obsah adresáře.

Write (w) – umožňuje uživateli vytvářet, rušit a přejmenovávat soubory a podadresáře, používá se s právem execute (x).

Execute (x) – umožňuje uživateli vstoupit do adresáře. Je podmínkou pro všechny operace s obsahem adresáře (kromě vypsání jmen).

Každý soubor může mít ještě přidělena tři speciální práva:

SUID – set user ID – pro uživatele

SGID – set group ID - pro skupinu

Sticky bit – pro ostatní

Je-li spuštěn soubor, který má nastaveno právo SUID (SGID), nastaví se efektivní uživatel na vlastníka souboru. Program je pak vykonáván, jako by jej spustil jeho vlastník

Je-li nastaven Sticky bit, obsah spustitelného souboru je ponechán ve vyrovnávací paměti. Tím se urychlí příští start programu. V současnosti je používán zřídka.

Má-li adresář nastaveno právo SGID, budou mít soubory a podadresáře v něm vytvářené nastaveny stejnou skupinu, jako tento adresář.

Má-li adresář nastaven Sticky bit, může v něm uživatel mazat jen soubory, které vlastní nebo má na ně právo write.

3.6.1 Změna přístupových práv

Pro změnu přístupových práv slouží příkaz **chmod**

chmod <číslo> <soubor>

Číslo se zadává v osmičkové soustavě, zpravidla je třímístné. První číslice udává práva vlastníka, druhá práva skupiny, třetí práva ostatních. Jsou-li zadávána i speciální práva pak je toto číslo čtyřmístné a první číslice patří těmto právům. Hodnota 4 je pak právo SUID, hodnota 2 právo SGID a hodnota 1 je Sticky bit.

Tab. 2. Práva a jejich hodnoty

Právo	číslo binárně	číslo oktalově
Execute	001	1
Write	010	2
Read	100	4

Pro kombinaci jednotlivých práv stačí sčítat jejich hodnoty (např. právo pro čtení a zápis má hodnotu $4+2=6$).

Symbolický zápis přístupových práv

Tento způsob je o něco složitější než předchozí. Práva k souborům nejsou definována absolutně, ale relativně k již existujícím právům. K nastavení práv symbolickým zápisem se opět používá program

chmod <kategorie><operace><právo> <soubor>

Které ze tří skupin se změna týká, určuje <kategorie> (viz. Tabulka III). Výsledná práva jsou vypočtena pomocí symbolu <operace> ze stávajících práv a z nastavovaných práv. Tento symbol může nabývat z hodnot:

„+“ – znamená, že právo bude přidáno

„-“ – znamená, že právo bude odebráno

„=“ – znamená, že právo bude nastaveno absolutně. Jestliže <operace> má tuto hodnotu, lze vynechat argument <právo>. Dané kategorii pak budou odebrána všechna přístupová práva.

Tab. 3. Kategorie a jejich označení

symbol	kategorie	poznámky
u	owner	vlastník
g	group	skupina
o	others	ostatní
a	all	vlastník, skupina i ostatní

Tab. 4. Práva

Právo	symbol
Read	r
Write	w
Execute	x
SUID/SGID	s
Sticky	t

3.6.2 Uživatelská maska

Uživatelská maska slouží k přidělování práv u nově vytvářených souborů. Tato maska se nastavuje příkazem **umask**. Bez parametru vypíše aktuální nastavení masky. S parametrem slouží ke změně nastavení. Zadáváme jej jako osmičkové číslo.

Je-li nový soubor vytvořen např. kopií souboru, pak jeho přístupová práva budou nastavena jako logický rozdíl práv výchozího souboru a uživatelské masky.

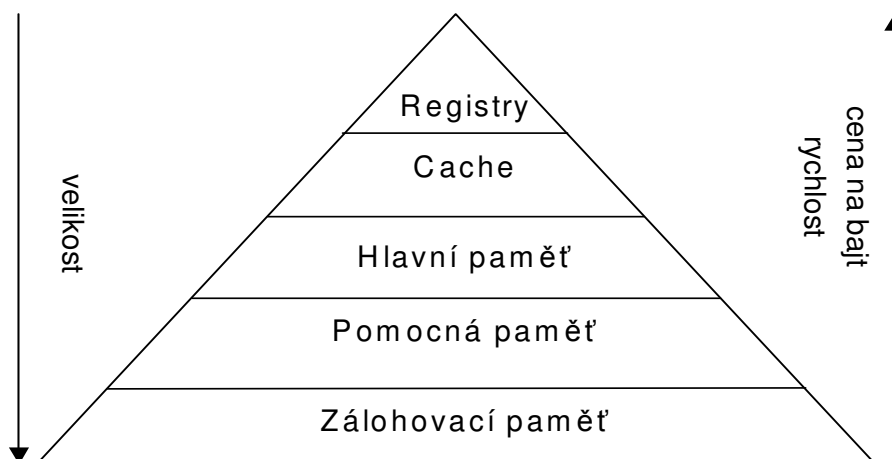
Není-li při vytváření nového souboru použit žádný zdrojový soubor, přístupová práva jsou vypočtena jako logický rozdíl 666 a uživatelské masky (u adresářů je to logický rozdíl 777 a uživatelské masky).

3.7 Základní adresářová struktura systému

Operační systém Linux má strukturu adresářů odvozenou od standardní struktury, která je použita v Unixu. Jednotlivé distribuce se však mohou v některých detailech lišit. Některé distribuce se drží standardů Filesystem Hierarchy Standard (FHS) a Linux Standard Base (LSB). Stručný přehled adresářové struktury je uveden v příloze PI. Podrobnější přehled lze nalézt v [4] a [10].

4 SPRÁVA PAMĚTI

Jedním ze základních prostředků, které potřebuje proces (viz. 5. Správa procesů) ke svému běhu, je paměť. V počítači najdeme několik druhů paměti, liší se podle rychlosti přístupu a svojí velikostí:



Obr. 1. Hierarchie paměti

Hlavní funkce správy paměti

Velký adresový prostor – systém se chová jako by měl k dispozici více paměti, než ve skutečnosti má. Velikost této virtuální paměti může být několikanásobně větší, než skutečná fyzická paměť.

Ochrana – každý proces má vlastní virtuální adresový prostor. Tyto prostory jsou vzájemně odděleny a díky tomu nemůže běžící proces jedné aplikace ovlivňovat jiné procesy.

Mapování paměti – slouží k mapování obrazů programu a datových souborů do adresového prostoru procesu.

Alokace fyzické paměti – každému spuštěnému procesu je umožněno alokovat určitý díl fyzické paměti systému.

Sdílení virtuální paměti – mohou se vyskytnout situace, kdy je vhodné, aby více procesů mohlo sdílet paměť. Linux podporuje komunikaci mezi procesy sdílením paměti mechanismem Unix Systém V IPC. [4]

4.1 Virtuální paměť

Operační systém Linux podporuje virtuální paměť. To znamená, že jako rozšíření paměti RAM využívá část disku. Paměťové bloky, které nejsou právě využívány zapisuje jádro systému na disk a paměť se tím pádem může využívat i pro jiné účely. Když pak přijde požadavek na její původní obsah, bloky se načtou z disku zpět do paměti. To vše uživatel nevnímá. Programy, které pracují pod Linuxem vidí, že mají k dispozici hodně paměti a nestarají se o to, že část této paměti je na disku. Ta část disku, která je využita jako virtuální paměť se nazývá **odkládací prostor (swap space)**.

4.1.1 Funkce virtuální paměti

Při použití virtuální paměti jsou všechny adresy virtuální. Na fyzické adresy je procesor překládá pomocí informací v tabulkách, které jsou udržovány operačním systémem. Virtuální i fyzická paměť jsou rozděleny na části o stejné velikosti. Tyto části se nazývají **stránky**. Na systémech s procesory Intel x86 používá Linux stránky o velikosti 4kB. Každá stránka má své jedinečné číslo – **číslo rámce stránky (PFN)**.

Virtuální adresa je složena ze dvou částí. Tyto části se nazývají **offset** a **číslo rámce virtuální stránky**. Procesor musí přeložit číslo rámce virtuální stránky na fyzickou a poté ve fyzické stránce přistoupit na místo se správným offsetem. Procesor k tomuto využívá tabulky stránek.

Při překladač virtuální adresy na fyzickou musí procesor nejdříve zjistit číslo rámce virtuální stránky a poté offset v tabulce virtuálních stránek. Číslo rámce je použito jako index do tabulky stránek. Tím se získá položka tabulky pro danou stránku. Je-li tato položka s daným offsetem platná, převezme z ní procesor číslo rámce fyzické paměti. Není-li tato položka platná, pokouší se procesor o přístup k neexistující oblasti virtuální paměti. Procesor v tomto případě nemůže přeložit adresu a musí předat řízení operačnímu systému, který problém vyřeší. Této události se říká **výpadek stránky**.

Pokud se proces odkazuje na platnou položku tabulky stránek, vezme procesor číslo fyzického rámce stránky a vynásobí jej velikostí stránky. Tím se získá bazová adresa stránky ve fyzické paměti. K té se připočítá offset instrukce nebo dat, která se požadují.

Vynucené stránkování

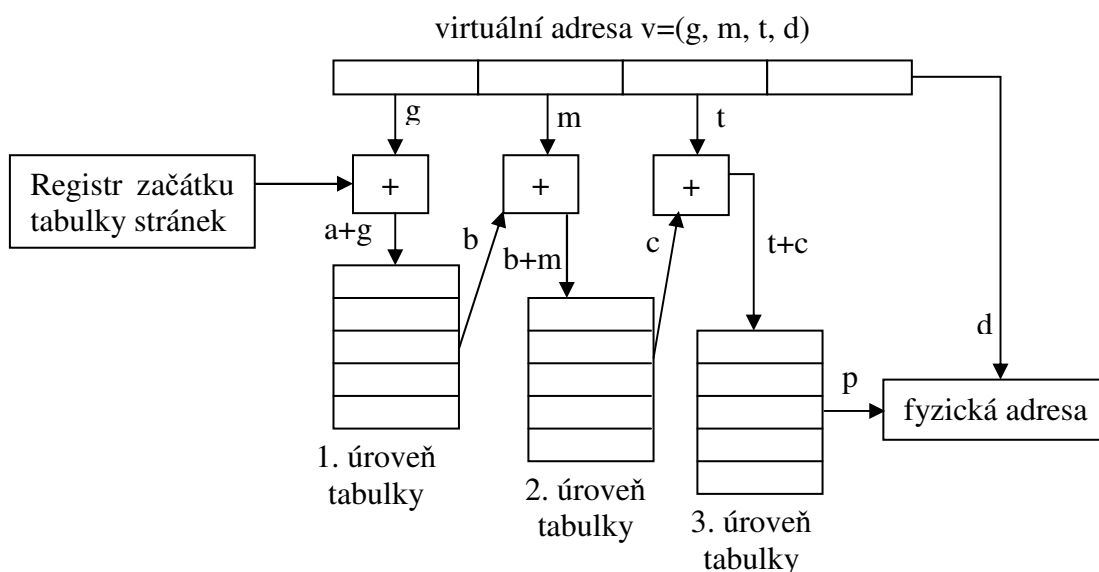
Protože fyzické paměti je v systému méně než virtuální, je třeba, aby operační systém fyzickou paměť příliš neplýtvál. Jednou z možností je nahrávat pouze ty virtuální stránky, které spuštěný program opravdu potřebuje. Té metodě nahrávání pouze potřebných stránek se říká **vynucené stránkování**.

Odkládání na disk

Když proces potřebuje přenést do fyzické paměti nějakou stránku a není v ní už místo, musí si operační systém místo pro tuto stránku vytvořit. Místo si udělá tak, že zruší nějakou jinou stránku. Jestliže stránka, kterou chce operační systém zrušit, byla modifikována, musí se její obsah zachovat tak, aby bylo možné k ní později opět přistupovat. Takovým stránkám se říká **modifikované (dirty)**. Tyto stránky se při odstraňování z paměti ukládají do speciálního souboru, kterému se říká **odkládací soubor**. Existuje několik algoritmů, podle kterých se odsunují stránky z paměti. Linux využívá algoritmus zvaný **Least Recently Used (LRU)** [6]. Podle tohoto algoritmu má každá stránka svůj věk, který se mění s přístupem ke stránce. Nejčastěji navštěvovaná stránka je nejmladší. Stránky, ke kterým se přistupuje nejméně, jsou nejstarší. Nejstarší stránky jsou odkládány jako první.

4.1.2 Tabulky stránek Linuxu

Linux předpokládá tři úrovně tabulky stránek. Každá obsahuje číslo rámce stránky tabulky stránek další úrovně.



Obr. 2. Organizace tabulek stránek

Vektor **free_area** slouží kódu alokace stránek k nalezení volných stránek. Každý jeho prvek obsahuje informace o bloku volných stránek. První prvek popisuje bloky o jedné stránce, druhý prvek popisuje bloky o dvou stránkách, třetí prvek popisuje bloky o čtyřech stránkách. Každý další prvek popisuje bloky stránek s krokem po mocninách dvou.

Alokace stránek

K alokaci a dealokaci bloků stránek používá Linux **Buddyho algoritmus**. Kód pro alokování stránek se pokouší alokovat blok jedné nebo více fyzických stránek. Stránky jsou alokovány v blocích o velikosti mocniny dvou. Dokud je v systému dostatek volných stránek (**nr_free_pages > min_free_pages**), bude alokační kód prohledávat strukturu **free_area** a hledat blok o požadovaném počtu stránek.

Alokační algoritmus nejprve hledá bloky stránek, které mají požadovanou velikost. Není-li nalezen žádný volný blok žádané velikosti, hledá se volný blok dvojnásobné velikosti. To se opakuje do doby než je nalezen nějaký volný blok nebo než je prohledána celá struktura **free_area**. Není-li nalezen žádný volný blok, alokace selže. Pokud je nalezený blok větší než je požadovaná velikost, je třeba jej rozdělit na blok požadované velikosti. Tento blok je přidělen. Jeho druhá polovina je přidělena k blokům o stejné velikosti. [6]

Dealokace stránek

Alokování bloků vede k fragmentaci paměti, kdy se větší bloky rozbíjejí na menší. Dealokační kód má za úkol rekombinovat stránky zpět na větší bloky vždy, když je to možné.

Jestliže dojde k uvolnění bloku stránek, zkontroluje se, jestli sousední blok volných stránek nemá stejnou velikost. Má-li stejnou velikost, pak se oba volné bloky spojí na nový volný blok stránek o dvojnásobné velikosti. Pokaždé, když se spojí dva volné bloky do většího bloku, zkusí dealokační kód spojit tento blok do ještě většího bloku. Díky tomu mají bloky volných stránek takovou velikost, jakou využití paměti dovolí.

4.3 Vyrovnávací paměť

Čtení z disku je ve srovnání s přístupem k fyzické paměti velmi pomalé. V běžném provozu se navíc z disku několikrát načítají stejná data během relativně krátkých časových intervalů. Jediným načtením informací z disku do paměti a jejich uložením v paměti do doby, než je bude potřeba, lze zrychlit všechny operace čtení z disku, vyjma prvního čtení. Paměť vyhrazená pro tyto účely se nazývá **vyrovnávací paměť**.

Vyrovnávací paměť obvykle nebývá příliš velká. Když se celá naplní, je třeba ji uvolnit. Data, která nebyla nejdéle používána se zruší a uvolněný prostor je použit k uložení nových dat.

Vyrovnávací paměť funguje i při zápisu na disk. Data, která se zapisují na disk, se velmi často brzy opakovaně načítají. Je proto výhodné uložit zapisovaná data i do vyrovnávací paměti. Další výhodou je, že uložením dat do vyrovnávací paměti se zlepší odezva programu, který data zapisuje.

Podle toho jak vyrovnávací paměti fungují je dělíme na **write-through** a na **write-back**.

Write-through – data se zapisují na disk ihned a zároveň zůstávají uložena ve vyrovnávací paměti.

Write-back – zápis dat na disk je odložen na pozdější dobu. Tento systém zápisu je efektivnější než výše popsany, ovšem je také více náchylnější k chybám. Dojde-li totiž k havárii systému nebo k výpadku proudu v nevhodný okamžik, data která čekají na zápis ve vyrovnávací paměti se obvykle ztratí.

Pro vyprázdnění vyrovnávací paměti (všechna neuložená data se uloží na disk) slouží příkaz **sync**. Tento příkaz se používá vždy, když chceme obsah vyrovnávací paměti bezpečně uložit na disk. V tradičních systémech Unix existuje program **update**. Ten běží na pozadí a každých 30 sekund spouští příkaz **sync**. V Linuxu běží daemon **bdflush**, který dělá něco podobného. Spouští se příkazem **update**. V případě, že je daemon **bdflush** z jakéhokoliv důvodu neočekávaně ukončen, podá o tom jádro systému varovné hlášení.

Do vyrovnávací paměti se ve skutečnosti neukládají celé soubory, ale **bloky**.

Blok – nejmenší jednotka při vstupně-výstupních diskových operacích.

Velikost vyrovnávací paměti

O tom jak je vyrovnávací paměť efektivní, rozhoduje její velikost. Je-li vyrovnávací paměť malá, bude v ní uloženo tak málo dat, že se vyprázdní dřív, než dojde k opakovanému použití dat. Příliš velká vyrovnávací paměť může zmenšit dostupnou systémovou paměť a zapříčinit časté odkládání, které je také pomalé.

Aby byla reálná paměť využívána co nejefektivněji, využívá Linux veškerou volnou paměť RAM jako vyrovnávací paměť. Potřebují-li běžící programy více fyzické paměti, je vyrovnávací paměť automaticky zmenšována.

V systému Linux je vyrovnávací paměť využívána automaticky. Uživatel proto nemusí dělat nic zvláštního. Při správě paměti využívá Linux řadu vyrovnávacích pamětí.

Vyrovnávací paměť bufferů – obsahuje datové buffery, které využívají ovladače blokových zařízení. Buffery mají pevnou délku. Obsahují bloky informací, které už byly z blokového zařízení načteny nebo se mají na něj zapsat. Buffery jsou indexovány identifikátorem zařízení a číslem požadovaného bloku a slouží k rychlému nalezení bloku dat. Jestliže jsou data v bufferu, nemusejí se načítat z fyzického blokového zařízení. Tím se přístup výrazně zvyšuje.

Vyrovnávací paměť stránek – používá se pro zrychlení přístupu k obrazům programů a k datům na disku. Je v ní uložen logický obsah souborů stránek. Přístup k ní se provádí prostřednictvím identifikace souboru a offsetu. Stránky, které jsou načítány z disku, jsou uchovávány v této paměti.

Odkládací paměť – v odkládacím souboru jsou uloženy pouze modifikované stránky. Nedojde-li k modifikaci stránky od posledního zápisu do odkládacího souboru, není nutné stránku do souboru znovu zapisovat, jestliže je potřeba stránku odložit. Stačí stránku pouze zrušit. Tím se ušetří zbytečné a náročné diskové operace.

Hardwarové vyrovnávací paměti – přímo v procesoru je vyrovnávací paměť položek tabulek stránek. Procesor nemusí přímo načítat položku tabulky stránek, ale využívá podle potřeby informace, které má uloženy ve své vyrovnávací paměti. Tato paměť je označena jako **překládový buffer – Translation Look-aside Buffer (TLB)**. Jejím obsahem jsou kopie položek tabulek stránek jednoho či více procesů v systému. [4]

5 SYSTÉM PROCESŮ

Program – zápis algoritmu v nějakém programovacím jazyce. Je uložen na disku, je statický, neměnný.

Proces – běžící program. Je tvořen neměnným kódem programu, konstantami a proměnnými daty (stav procesu, data na zásobníku, globální proměnné, soubory atd.).

Každý proces je během svého života charakterizován celkovým stavem – **kontextem**. Jeho uchování je důležité pro přepínání běhu jednotlivých procesů. Plánování běhu procesů a přidělování výpočetního času má na starost speciální část jádra systému, které se říká **plánovač – scheduler**. Naplánovanému procesu je přidělen čas. Proces je v tomto čase vykonáván. Po uplynutí vymezeného času je jeho vykonávání pozastaveno a jádro si uloží aktuální kontext procesu. Potom je naplánován další proces. Jádro obnoví jeho uložený kontext a po stanovenou dobu je tento proces vykonáván. Dále se celá procedura opakuje.

5.1 Vznik procesu

Každý proces prochází během svého životního cyklu třemi stádii. Nejdříve vznikne, pak je nějakou dobu v činnosti a nakonec zanikne. Životní cyklus procesu je spjat s jádrem systému. Jádro zasahuje do života procesu při jeho vzniku a ukončení. Stará se také o cyklické přepínání kontextu procesů a o jejich vzájemnou bezpečnost. Každý proces vlastní privátní část paměti a ostatní procesy do ní nemohou přímo zasahovat.

Protože je Linux víceúlohový systém, je třeba jednotlivé procesy od sebe jednoznačně odlišit. Jádro si o běžících procesech udržuje přehled, každý proces je jednoznačně identifikován pomocí čísla PID – **process identifikation**. Procesy tvoří v systému hierarchickou strukturu. Proces je vytvářen jiným procesem a ten se pak stává jeho rodičem. Každý proces, kromě prvního, má svého rodiče – **rodičovský proces**.

Systém Linux poskytuje volání jádra **fork()**, které vytváří nový duplicitní proces - **potomka**. Ten se od svého rodiče liší pouze ve svém PID. Rodiči i potomkovi jsou předány různé návratové hodnoty. Rodiči je vráceno PID jeho potomka, potomkovi je předána nula. Jiné návratové hodnoty umožňují další větvení procesů.

5.2 Život procesu

Během práce v systému se lze setkat se třemi druhy procesů:

Uživatelské procesy – userspace processes – jsou spouštěny jednotlivými uživateli. Obrazem procesu je obvykle program uložený na některém souborovém systému. Má-li program nastaven SUID bit, stává se vlastníkem procesu vlastník tohoto programu. V opačném případě je vlastníkem procesu uživatel, který jej spustil. Vlastníkovi procesu se také říká – **efektivní uživatel**.

Systémové procesy – daemons – jsou spouštěny během startu systému. Zajišťují různé služby. Systémový proces běží zpravidla v nekonečné smyčce a plní vymezené úkoly (např. periodická obsluha klientů). Vlastníkem systémového procesu je superuživatel root nebo některý ze systémových uživatelů.

Vlákná jádra – kernel threads – v podstatě jsou to speciální části jádra operačního systému, které běží jako speciální procesy.

Stavy procesu

Proces může během svého života procházet různými stavy. Jednotlivé procesy se mohou aktivně nacházet v paměti nebo čekat v odkládacím prostoru. Proces může běžet v uživatelském režimu nebo v režimu jádra. Do režimu jádra se proces může dostat pomocí volání jádra. [6]

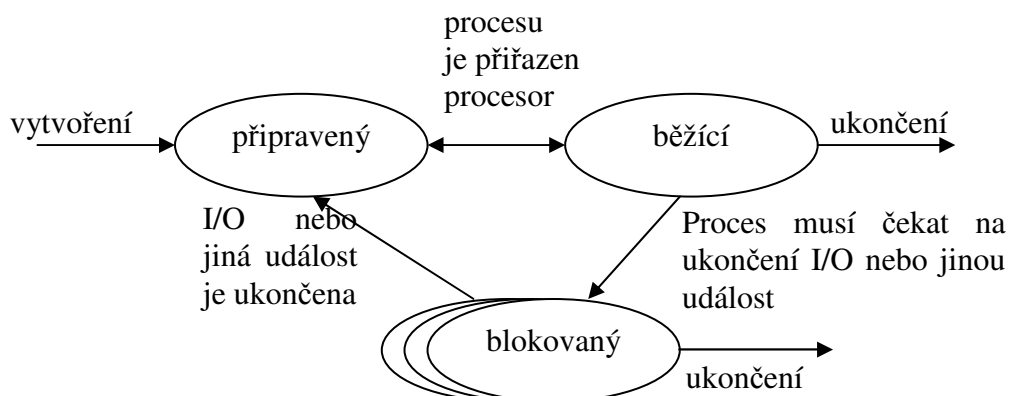
Procesy se mohou nacházet v následujících stavech:

Běžící – proces je ve stavu běžící, pokud je mu přidělen procesor.

Připravený – proces je ve stavu připravený, jestliže je připraven k běhu a čeká na přidělení procesoru.

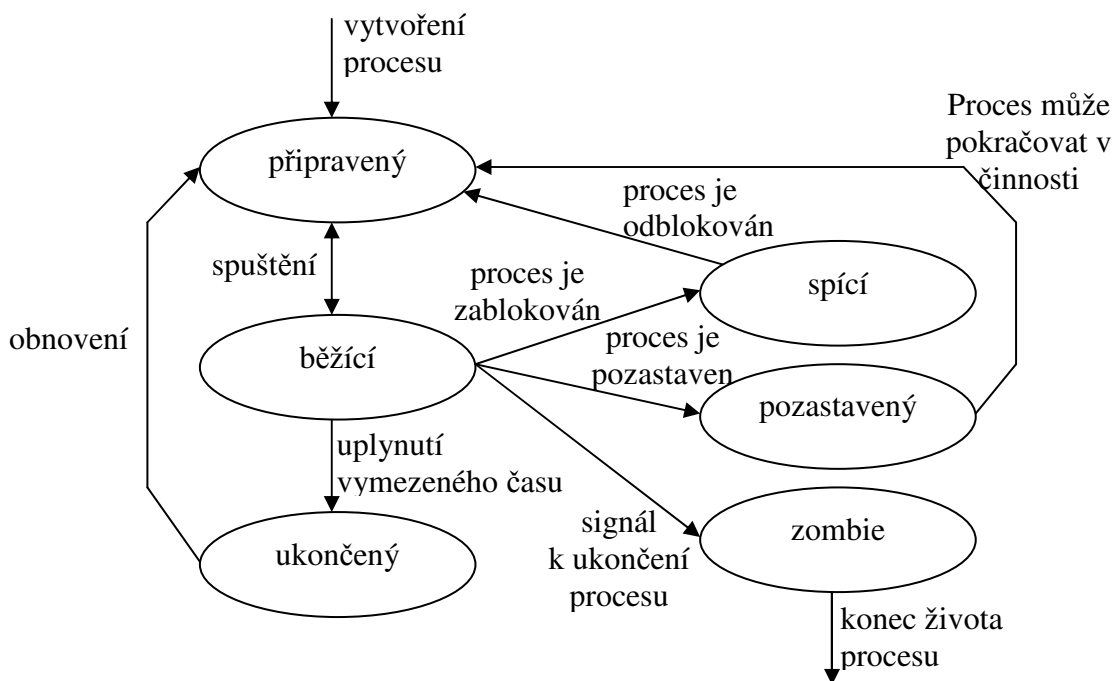
Čekající, spící nebo blokový – proces čeká na určitou událost (např. dokončení I/O operace).

Podrobnější informace o stavech procesů se nacházejí v [3] nebo anglicky v [9].



Obr. 4. Základní stavy procesu

V systému Linux mohou procesy ve skutečnosti procházet i více stavy. [2]



Obr. 5. Stavy procesu v Linuxu

Seznam procesů v systému lze získat příkazem **ps**. Tento příkaz má mnoho argumentů, z nichž nejdůležitější jsou **a**, **u** a **x**.

Argument **a** – zobrazení seznamu všech aktivních procesů včetně těch, které spustili i jiní uživatelé.

Argument **x** – zobrazení seznamu těch procesů, které nejsou nijak svázány s terminálem.

Argument **u** – zobrazení dalších užitečných informací o běžících nebo pozastavených procesech.

Je-li příkaz **ps** zadán bez argumentu, zobrazí se seznam aktivních procesů spojených s aktuálně používaným terminálem.

Pro zobrazení obnovujících se informací o procesech, které v systému běží slouží příkaz **top**. Po jeho zadání je zobrazena plná obrazovka informací o procesech běžících v systému (PID, jméno uživatele, priorita, využití CPU a paměti procesem, doba běhu procesu, jméno procesu). Také se vypíše obecné informace o systému samotném (průměrná zátěž, počet procesů, stav CPU, informace o volné paměti). Na vrcholu výpisu jsou zobrazeny procesy, které nejvíce zatěžují systém.

5.2.1 Běh procesů na popředí a na pozadí

Protože Linux je víceúlohový operační systém, vzniká přirozený uživatelský požadavek spouštět některé procesy na **pozadí – background**. Od procesu běžícího na pozadí se očekává, že nebude omezovat uživatele při práci v shellu. Dalším požadavkem je samostatnost procesu běžícího na pozadí a také by v žádném případě neměl být interaktivní.

Proces lze spustit na pozadí hned při zadávání příkazu pro spuštění. K tomu slouží znak „&“, který se zadává na konci tohoto příkazu. Proces je spuštěn na pozadí. Bezprostředně po jeho spuštění na pozadí zobrazí shell uživateli informační řádek, na kterém jsou dvě čísla. První číslo (v hranatých závorkách) je **identifikátor procesu v rámci aktuálního shellu**. Druhé číslo je PID procesu. V rámci každého shellu jsou procesy číslovány od jedné. Na proces se lze při další práci odkazovat oběma čísly.

Na pozadí lze přesunout i proces, který byl původně spuštěn na popředí. Uživatel nejprve stiskne CTRL Z. Tím se chod procesu pozastaví. K přesunu na pozadí takto pozastaveného procesu slouží příkaz **bg**. Argumentem tohoto příkazu je identifikátor procesu v rámci shellu.

Opakem příkazu **bg** je příkaz **fg**, který slouží k přesunu procesu na popředí. Na popředí lze přesunout jak proces běžící na pozadí, tak i proces pozastavený. [10]

5.2.2 Změna priority procesů

Priorita procesů určuje, jak často jim plánovač procesů bude přidělovat procesorový čas. Priorita se značí číselně. Čím větší číslo tím, tím nižší priorita. Změnu priority procesů může provádět superuživatel root i běžní uživatelé. Ti mohou pouze snižovat prioritu jen u svých vlastních procesů.

Pro manipulaci s prioritou jsou k dispozici příkazy **nice** a **renice**. Prvním z nich lze spustit proces s pozměněnou prioritou, pomocí druhého lze změnit prioritu již běžícímu procesu.

nice -n <priorita> <příkaz>

renice <priorita> -p <PID>

<priorita> může nabývat hodnot z intervalu $-20 \dots +19$. Je-li hodnota kladná, priorita procesu bude adekvátně snížena, v opačném případě bude priorita zvýšena. Zápornou hodnotu může použít pouze superuživatel root.

5.3 Ukončení života procesu

Signály

Signál je mechanismus jádra, s jehož pomocí se mohou procesy vzájemně informovat o různých událostech. Signály jsou jednoznačně určeny svým číslem a kromě nich mají i své mnemonické značky. Procesy mohou na příchod signálu reagovat různými způsoby:

1. Přenechat signál k vyřízení jádru systému – ze strany jádra to většinou znamená ukončení běhu procesu.
2. Ignorovat signál.
3. Zachytit signál a reagovat na něj nějakou akcí.

Při ukončení svého běhu navrácí proces celočíselnou hodnotu. Rodičovský proces může čekat na dokončení svého potomka a potom převzme jeho návratovou hodnotu. K tomuto v systému slouží volání jádra **wait** (). Při jeho použití je proces pozastaven a jádro jej znovu rozeběhne až po dokončení některého z jeho potomků. Je-li potomek ukončen, ale rodičovský proces ještě nepřevzal informace o jeho skončení pomocí **wait** (), jádro si informace o potomkovi neustále udržuje. Takový potomek se pak nazývá **zombie**. Jádro informuje rodiče o dokončení potomka signálem SIGCHLD.

Tab. 5. Nejčastěji se vyskytující signály

signál	číslo	Význam
SIGHUP	1	odpojení terminálu
SIGINT	2	přerušení z klávesnice
SIGQUIT	3	konec s uložením obrazu paměti
SIGILL	4	použita instrukce neznámá pro procesor
SIGTRAP	5	ladicí přerušení
SIGABRT	6	ukončení z důvodu I/O operace
SIGFPE	8	chyba aritmetiky s pohyblivou desetinou čárkou
SIGKILL	9	okamžité ukončení procesu
SIGSEGV	11	selhání segmentace
SIGALRM	14	vypršení časového intervalu
SIGTERM	15	ukončení z důvodu I/O operace
SIGUSR1	16	uživatelský signál 1
SIGUSR2	17	uživatelský signál 2
SIGCHLD	18	změna stavu potomka
SIGSTOP	23	pozastavení procesu
SIGCONT	24	pokračování v činnosti

Zasílání signálů

Uživatelé mohou procesům zasílat signály pomocí příkazů **kill** a **killall**. Běžný uživatel může zasílat signály pouze vlastním procesům, superuživatel root může zasílat signály libovolným procesům. Procesy mohou signály ignorovat, existují však dvě výjimky. Signály SIGKILL a SIGSTOP jsou nemaskovatelné a nezachytitelné. Po zaslání signálu SIGKILL bude proces okamžitě ukončen. Signálem SIGSTOP může být proces pozastaven.

kill -<signál> <proces>

<signál> může být číslo nebo mnemonická zkratka signálu. Je-li argument <signál> použit, musí před ním být uvedená pomlčka. V případě, že není <signál> použit, bude procesu zaslán implicitní signál SIGTERM. Ten je na rozdíl od signálu SIGKILL zachytitelný a proces na něj může reagovat tím, že uloží rozpracovaná data na disk a až poté se sám ukončí.

<proces> je buď PID procesu nebo číslo ve tvaru %<číslo>, kde <číslo> je identifikátor v rámci aktuálního shellu.

Příkaz **killall** rovněž zasílá signály. Procesy však nejsou identifikovány svým PID, ale svým jménem. Tento příkaz poskytuje uživateli větší komfort, ale jeho neopatrné použití může vést k nechtěnému ukončení více procesů se stejným jménem.

Signály SIGSTOP a SIGCONT je možné pozastavovat procesy a přesouvat je na pozadí obdobným způsobem jako použitím CTRL Z a příkazu **bg**. [10]

6 BEZPEČNOST SYSTÉMU

Druhy útočníků

Průzkumník – nesnaží se něco poškodit nebo zničit. Do systému většinou proniká z dobrého pocitu, ze samotného průniku. Často se jedná o studenty, kteří se tímto způsobem baví. Je jim jedno kam se dostanou. Jediné, čím se orientují, je dostupnost cílového systému.

Cracker – své oběti si vybírají cíleně. Jejich cílem je zisk. Někteří z nich používají cizí stroje jako odrazové můstky pro další činnost nebo jako skladiště kradených nebo jiných ilegálních dat. Jiní pracují jako nájemní vyhledávači informací, mohou také sloužit jako likvidátoři konkurence.

Podrobnější dělení útočníků je v [8].

6.1 Lokální bezpečnost

Důvody útoků na systém:

- 1.) Útočník pouze zkoumá, co se na strojích nachází a hledá něco zajímavého. Zavádí si zadní vrátka pro pozdější použití.
- 2.) Útočník se potřebuje dostat do vnitřní sítě. Napadá firewall.
- 3.) Útočník se snaží získat stroj, který chce dále používat k jinému útoku.
- 4.) Útočník hledá konkrétní data
- 5.) Útočník napadá web server, aby změnil jeho obsah a poté na stránkách vystavil vlastní informace.
- 6.) Útočník chce využít diskovou kapacitu pro uchování ilegálních dat, která získal.
- 7.) Útočník chce využít výpočetní kapacitu stroje (například k prolomení hesel hrubou silou).

Útočník se většinou nejdříve snaží získat přístup do systému jako lokální uživatel. Když se mu to podaří, může již snadněji získat práva superuživatele. Tím získá takřka neomezenou moc a může si v systému dělat co se mu zlíbí. Bude-li systém dobře lokálně zabezpečen, útočnickova pozice se ztíží.

V některých případech může lokální uživatel napáchat škodu, přestože není útočníkem. Je proto dobré nevytvářet nové účty neznámým lidem.

Při vytváření nových účtů je vhodné, nastavit uživatelům minimální potřebná oprávnění, která potřebují ke své práci.

Superuživatel by měl sledovat kdo a odkud se přihlásil. Veškeré již nepoužívané účty je třeba odstranit, protože takové si většinou útočník vybírá jako svůj cíl a z nich se pak snaží dostat hlouběji do systému.

Hesla

Jsou jedním z nejvýznamnějších prvků k zajištění bezpečnosti. V Unixových systémech jsou hesla šifrována jednosměrně. Je vybírán postup, u něhož není možné z utajené podoby hesla získat zpět originál. Tím pádem nezná otevřenou podobu hesla ani systém, ani správce, pouze samotný uživatel. Při pokusu o přihlášení se heslo, které uživatel zadá, zašifruje a porovná se s uloženou podobou správného hesla. Pokud heslo souhlasí, je uživatel vpuštěn do systému. Aby se nestalo, že v případě shodných hesel více uživatelů budou mít tato hesla stejný tvar i zašifrována, připojuje se k heslu obvykle i uživatelské jméno účtu, ke kterému heslo náleží.

Hesla by měla být volena tak, aby nebylo snadné je uhodnout. Měla by být co nejméně běžná a měla by obsahovat neobvyklé znaky. Jsou programy, které dokáží bezpečné heslo náhodně vygenerovat. Správce systému má několik možností jak odhalit snadno uhodnutelná hesla (např. programy John the Ripper nebo Crack). Narazí-li správce na nějaké snadno uhodnutelné heslo, měl by kontaktovat konkrétního uživatele a požádat ho o změnu hesla.

Šifrování

Symetrické šifrování – používá se stejný klíč pro šifrování a dešifrování.

Asymetrické šifrování – používají se dva klíče. Veřejný klíč k šifrování dat, tajný klíč k dešifrování dat.

PGP (Pretty Good Privacy) – program zajišťující služby šifrování a elektronického podpisu.

GPG (GNU Privacy Guard) – snaží se nahradit původní **PGP**. Umí šifrovat a zpětně dešifrovat data a elektronicky je podepisovat, ověřovat elektronický podpis druhých stran, spravovat elektronické klíče.

GPG používá následující šifrovací algoritmy:

- **ElGamal**
- **DSA**
- **3DES**
- **Blowfish**
- **Twofish**
- **CAST5**
- **MD5** – viz. dále
- **SHA-1** - viz. dále
- **RIPE-MD-160**
- **TIGER**

Pro šifrování hesel se v Linuxu používají algoritmy **DES** a **MD5**. Jsou to efektivní šifrovací algoritmy, které je téměř nemožné v rozumném čase prolomit.

DES (Data Encryption Standard) – implementuje jej funkce **crypt**, je standardní součástí UNIX. Byl vyvíjen za účasti vlády USA a IBM, nebyl exportovaný za hranice USA.

MD5 – hashovací algoritmus, proti DES má řadu vylepšení (neomezená délka hesel, rozsáhlejší prostor klíčů).

6.2 Síťová bezpečnost

Počítačová síť představuje obecně velkou hrozbu útoku na počítače, které jsou k ní připojeny.

Druhy síťových útoků

Útok na web server – nejčastějším vstupem do systému se stávají PHP skripty. Takový útok sám o sobě není příliš nebezpečný, ale s PHP skriptami se dají dělat i jiné věci, které mohou server přímo ohrozit. Platí pravidlo, že server by měl obecně běžet s co nejomezenějšími právy.

Odposlech komunikace – druhým nejběžnějším útokem je odposlechnutí důležitých dat, která procházejí nešifrovaným kanálem.

Denial of service (DoS) – druh útoku, který nemá za úkol přímo odcizit data, ale útočník jej může použít k podpoře svých plánů. Princip je v zahlcení serveru daty. Ten pak pod tíhou dat nevydrží a zkolabuje.

Podvržené IP adresy – mnoho strojů používá pro ověření totožnosti jiného serveru pouze IP adresu. Tu však lze velmi jednoduše podvrhnout.

Odcizení relace – používány jsou podobné postupy jako při falšování IP adres.

Man-in-the-middle (muž uprostřed) – tento druh útoku lze jednoduše využít při použití ověřování identifikace pomocí IP nebo DNS záznamu. Útočník přesměruje spojení na svůj stroj a vydává jej za originální server. [5]

Nebezpečné služby

Útočník má přes tyto služby snadnější přístup. Měly by se proto používat co nejméně nebo ve spojení s nějakou ochranou.

Finger – služba, která slouží k získávání informací o uživateli na lokálních i vzdálených systémech.

Telnet – slouží pro zpřístupnění vzdáleného terminálu přes TCP/IP spojení. Telnet přenáší veškerá data nešifrovaně a může je tedy kdokoliv vidět.

Rlogin – je velmi podobný telnetu, plní stejnou službu. Má však některé další funkce pro pohodlí uživatelů.

SMTP – protokol pro jednoduché předávání pošty mezi různými stroji. Je už z principu poměrně nebezpečný. Díky své jednoduchosti umožňuje útočníkovi provádět s poštou mnohé nepodloženosti.

FTP – protokol pro přenos souborů mezi dvěma počítači.

CGI skripty – malé programy, které slouží tvůrcům webů k rozšíření webových funkcí a možností.

NTP – protokol určený k velmi chytře a propracované synchronizaci času. Je sice dobře navržen, je však velmi neodolný vůči některým útokům.

DNS – způsob překladu doménových jmen na IP adresy. [5]

Druhy zabezpečení na sítích

SSH (*Secure Shell*)

Jedná se o bezpečnou alternativu k programům pro použití vzdáleného terminálu (telnet, rsh, rlogin). Umožňuje ověřování identity hostitele, šifrování přenosu dat, vytvoření šifrovaného tunelu na úrovni aplikační vrstvy.

Firewall

Je prostředkem k řízení, které informace mohou směřovat vně a dovnitř počítačové sítě. Je připojen jednak k internetu a jednak k lokální síti. Veškerá komunikace pak probíhá přes něj. Většinou funguje jako první linie obrany proti jakýmkoliv útokům. Jeho pokořením získá útočník zpravidla volný přístup do celé sítě. Je proto nezbytné zamezit co možná největšímu počtu útoků.

Firewall obvykle není náročný na strojový výkon, proto jako hardware postačí starší počítač, který je doplněn o síťové karty.

ZÁVĚR

Operační systém Linux si získává stále více příznivců. Ve světě operačních systémů si vybudoval své pevné místo. S Linuxem se lze setkat od kapesních počítačů, přes pracovní stanice a servery, až k počítačům třídy mainframe. Pro Linux existuje velké množství aplikací, které v současnosti pokrývají snad všechny nejdůležitější oblasti nasazení výpočetní techniky.

Mnozí uživatelé mají Linux na svém počítači nainstalován vedle Windows. Někteří nedají na Linux dopustit, jiní jej zase zatracují. Ve Windows běží Microsoft Office, spousta her. Jdou poměrně snadno nainstalovat a nastavit. Jsou vyhlášené nestabilitou a často havarují. V Linuxu běží OpenOffice, spousta technických programů, málo her. Instalace a nastavení může být obtížnější. Je mimořádně stabilní a má vysoký výkon. K haváriím dochází výjimečně. Také jeho cena může být pro někoho rozhodující, dá se legálně pořídit téměř nebo úplně zdarma. Na jeho vývoji pracují tisíce uživatelů po celém světě. Linux přináší spoustu možností, uživateli ovšem bude chvilku trvat než se s ním naučí zacházet. Pro Linux je napsána spousta dokumentace a také knížek. Většina z nich je ovšem v angličtině. Rovněž na internetu se dá při troše trpělivosti nalézt spousta informací, které se zabývají tematikou Linuxu.

Cílem této bakalářské práce bylo vytvořit studijní materiál, ve kterém je popsána architektura operačního systému Linux. Text měl obsahovat následující části: správa procesů, správa paměti, správa souborového systému a zabezpečení operačního systému Linux.

Celý text jsem rozdělil do šesti hlavních částí. V úvodu je definován pojem operační systém. Následuje první kapitola, která nás seznámí s historií vzniku operačního systému Linux. Ve druhé kapitole je popsáno přihlášení a odhlášení uživatelů. Je zde vysvětlen pojem příkazový interpret a v tabulce jsou uvedeny základní klávesové zkratky pro práci v něm.

Následuje kapitola nazvaná souborový systém. V jejím úvodu se lze dočíst, ze kterých částí se v Linuxu skládá soubor a obecná pravidla pro jeho pojmenování. V další části jsou popsány základní příkazy pro práci se soubory, adresáři a odkazy, je zde vysvětlen princip přístupových práv.

V kapitole nazvané správa paměti se nejprve dozvíme, jaké jsou hlavní úkoly správy paměti. Následuje popis funkce virtuální paměti, princip alokace a dealokace stránek, rozdělení a funkce vyrovnávací paměti.

Pátá část je věnována systému procesů. Je složena ze tří částí: vznik procesu, život procesu a jeho zánik. Dozvíme se jak spouštět procesy na popředí, jak je přesunout na pozadí, jak jim změnit prioritu. Dočteme se co jsou to signály, jak je lze zasílat a jakým způsobem na signály reaguje proces.

Poslední kapitola se zabývá bezpečností. Je rozdělena na část věnovanou lokální bezpečnosti a část zabývající se síťovou bezpečností. Protože téma zabezpečení operačního systému Linux by bylo možné zpracovat jako samostatnou bakalářskou práci, zabývá se tato kapitola bezpečností pouze okrajově.

Dalším mým úkolem bylo navrhnout soubor zkouškových otázek. Ty jsou uvedeny v příloze PII. Odpovědi na všechny otázky lze najít v této bakalářské práci.

SEZNAM POUŽITÝCH ZDROJŮ

- [1] Bovet, D. P.: Cesati, M.: Understanding the Linux Kernel, 1. vydání, O'Reilly, 2000, ISBN: 0-596-00002-2
- [2] Deitel, H. M.: Operating systems, 3. vydání, Prentice Hall, 2004
- [3] Klimeš, C.: Operační systémy 1, Ostravská univerzita v Ostravě, Katedra informatiky a počítačů
- [4] Kolektiv autorů: Linux Dokumentační projekt, 3. vydání, Computer Press, 2003, ISBN:80-7226-761-2
- [5] Krčmář, P.: Linux tipy a triky pro bezpečnost, 1. vydání, Grada Publishing, 2004, ISBN: 80-247-0812-4
- [6] Patočka, M.: Diplomová práce – Srovnávací studie jádra Linuxu a FreeBSD, Univerzita Karlova v Praze, Matematicko – fyzikální fakulta
- [7] Písek, S.: HTML tvorba jednoduchých internetových stránek, 1. vydání, Grada Publishing, 2001, ISBN:80-2470094-8
- [8] Schmotzer, M.: Operačné systémy, Univerzita Pavla Jozefa Šafárika v Košiciach, Ústav informatiky
- [9] Tanenbaum, A. S.: Modern operating systems, 2. vydání, Prentice Hall, 2002, ISBN: 7-111-09156-6
- [10] Vychodil, V.: Operační systém Linux - Příručka českého uživatele, 1. vydání, Computer Press, 2003, IBN: 80-7226-33-1

SEZNAM OBRÁZKŮ

Obr. 1 Hierarchie paměti.....	19
Obr. 2 Organizace tabulek stránek.....	21
Obr. 3 Vektor free_area	22
Obr. 4 Základní stavy procesu.....	28
Obr. 5 Stavy procesu v Linuxu.....	28

SEZNAM TABULEK

Tab. 1 Základní klávesové zkratky.....	10
Tab. 2 Práva a jejich hodnoty.....	17
Tab. 3 Kategorie a jejich označení.....	17
Tab. 4 Práva a jejich symboly.....	17
Tab. 5 Nejčastěji se vyskytující signály.....	31

SEZNAM PŘÍLOH

Příloha PI: Základní adresářová struktura systému

Příloha PII: Soubor zkouškových otázek

PŘÍLOHA PI: ZÁKLADNÍ ADRESÁŘOVÁ STRUKTURA SYSTÉMU

/bin – programy potřebné pro úspěšné zavedení systému, základní programy pro práci uživatelů v systému.

/boot – obsahuje obraz jádra operačního systému a obrazy zavaděčů jádra.

/dev – obsahuje speciální soubory pro bloková a znaková zařízení.

/etc – systémové a aplikační konfigurační soubory.

/home – domovské adresáře uživatelů.

/lib – sdílené knihovny nutné pro start systému. Podadresář **/lib/modules** obsahuje moduly jádra (speciální části, které lze za běhu nahrávat do paměti).

/mnt – adresář pro dočasná připojení souborových systémů.

/opt – obvykle v něm bývá instalován komerční software nebo nestandardní balíčky.

/proc – adresář se speciálním souborovým systémem ProcFS obsahuje soubory s informacemi o systému.

/root – domovský adresář superuživatele root.

/sbin – programy potřebné pro zavedení a administraci systému (programy jsou vesměs používány jen superuživatelem root).

/tmp – adresář pro dočasné soubory.

/usr – programy instalované s distribuce systému a další software.

/var – adresář obsahuje data, která se během života systému dynamicky mění (soubory s uživatelskou poštou, www stránky, archivy Network News, fronta pro tiskárnu apod.).

PŘÍLOHA PII: SOUBOR ZKOUŠKOVÝCH OTÁZEK

- 1.) Vysvětlete pojem operační systém
- 2.) Popište základní příkazy pro práci s adresáři
- 3.) Popište základní příkazy pro práci se soubory
- 4.) Co jsou to speciální soubory, jejich rozdělení
- 5.) Co jsou to odkazy, jejich vytváření a mazání, jejich rozdělení
- 6.) K čemu slouží přístupová práva, rozdělení uživatelů do tří skupin
- 7.) Popište základní přístupová práva
- 10.) Popište speciální práva
- 11.) Změna přístupových práv – jak se provádí
- 12.) Popište symbolický zápis přístupových práv
- 13.) Co je a k čemu slouží uživatelská maska
- 14.) Co je a jak pracuje virtuální paměť
- 15.) Co je vynucené stránkování, odkládání na disk
- 16.) Popište alokaci stránek
- 17.) Popište dealokaci stránek
- 18.) Popište vektor `free_area`
- 19.) K čemu slouží Buddyho algoritmus
- 20.) Popište funkci vyrovnávací paměti
- 21.) Dělení vyrovnávacích pamětí
- 22.) Popište vznik a zánik procesů
- 23.) Rozdělení procesů
- 24.) Popište stavy, kterými proces prochází
- 25.) Popište jak se spouští procesy na popředí a na pozadí
- 26.) Co jsou to signály, popište jejich funkci
- 27.) Popište rozdíl mezi symetrickým a asymetrickým šifrováním