Otázka 27

Zadání

Architektura OS Unix
Interpret příkazů Bourne Again Shell
Souborový systém OS Unix

Předmět: **YD38UOS**

Základní pojmy

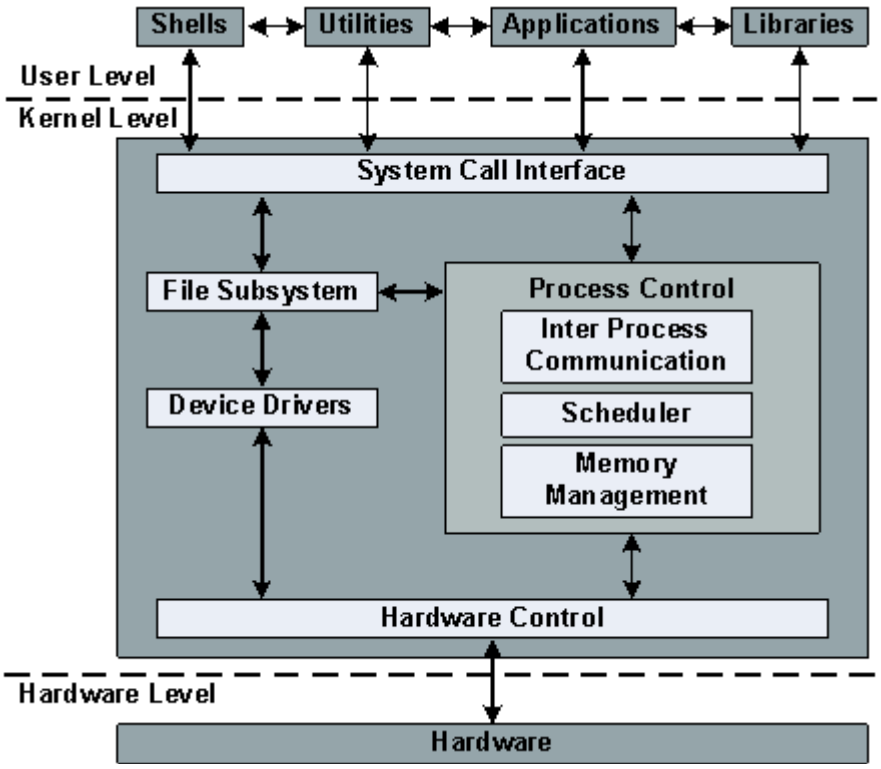
Firmware
Je zcela základní programové vybavení, které bývá integrální součástí elektronického zařízení a podporuje jeho běh a funkci. V kontextu moderních počítačů není jeho úkolem umožňovat interakci uživatele s hardwarem, nýbrž pouze poskytovat interface a nízkoúrovňové služby operačnímu systému a tím mu umožnit přístup k hardwaru. Firmware jsou většinou naprogramovány (vypáleny) do stálých (non-volatile) pamětí (ROM, EPROM, EEPROM, FLASH atd.) přímo výrobcem vyrábějícím dané zařízení.
Jako firmware se dá počít také obsah programovatelných hradlových polí pokud jsou v daném zařízení použita. V závislosti na způsobu uložení, tedy typu paměti, v nichž se nachází jej lze v některých případech modifikovat buď výměnou (tak zvané přečipováni) nebo přepsáním jejich obsahu (takzvané „flashování“). Firmware je typicky úzce vázaný na hardware pro který je určen, protože bývá psán buď přímo ve strojovém kódu nebo assembleru daného procesoru.
BIOS
Basic Input Output System – firmware v počítačích typu IBM PC
OS – operační systém
Je software, který obsluhuje přístup k hardwarovým zařízením počítače, jako jsou pevné disky, grafický subsystém, klávesnice, myši a další I/O zařízení a poskytuje rozhraní pro přístup k nim uživateli a plánuje úlohy, tj. přiděluje procesorový čas, paměť a úložné místo softwaru vyšší úrovně (tedy dalším pomocným systémovým rutinám, službám a aplikačním programům).
Systémový software
Software, který uživatel „nevidí“, který běží skrytě a podporuje běh počítače jako celku, ale uživatel s ním přímo nepřichází do styku (ovladače zařízení, NTP klient – utilita zajišťující synchronizaci systémového času počítače, loggery...)
Aplikační software
Software, se kterým úživatel přímo pracuje (správce souborů, office, www proghlíeč, hry...)
API
Application Programming Interface – kolekce funkcí, kterou disponuje platforma určitého programovacího jazyka či přímo nějaký software a tím umožňuje programátorovi jeho efektivní použití.
FS
File System – Systém souborů – abstraktní datová struktura, která na jedné straně umožňuje uživateli organizaci dat ve formě souborů a adresářů, tak aby k nim bylo možno snadno přistupovat a na straně druhé zajišťující jejich fyzické umístění na paměťovém médiu (pevný disk HDD, CD, DVD...)
I/O
Input/Output**–** česky V/V **-** Vstupně/Výstupní zařízení
kooperativní multitasking - proces běží až do chvíle, kdy dojde k předem definované události (např. žádost o V/V operaci) - pak je procesor přidělen dalšímu procesu
preemptivní multitasking - každému procesu je přidělen procesor na krátké, pevně stanovené kvantum času

Architektura OS Unix

V prvé řadě je nutné si uvědomit, že v dnešní době neexistuje nějaký operační systém s konkrétním jménem Unix. Původní UNIX je dnes již historický operační systém, který byl napsán v Bellových laboratořích v roce 1969 Dennisem Ritchiem a Kenem Thompsonem v roce 1969. To, o čem dnes mluvíme, jsou tzv. unixové operační systémy (nebo OS unixového typu) – a ty jsou na filozofii a architektuře původního UNIXu založeny. Sdílejí s ním souborový systém založený na stejné filozofii a stejně nebo podobně se ovládají, původního zdrojového kódu s ním však sdílejí jen velice málo nebo vůbec žádný.
Během 80 let se velice rozšířily dvě verze Unixových systémů a ty dnešní z nich přímo nebo nepřímo velkou měrou

vycházejí:
Unix System V - produkt firmy AT&T, jenž je přímým pokračovatelem systému původního Unixu
BSD (Berkeley Software Distribution, též Berkeley Unix) - odvozenina Unixu distribuovaná Kalifornskou univerzitou v Berkeley
Mezi aktuální moderní unixové systémy patří např.: GNU/Linux, FreeBSD, OpenBSD, Solaris, AIX

Operační systém typu Unix se skládá ze dvou hlavních částí: jádra a systémových programů.
Jádro (Kernel) se stará o přidělování systémových prostředků (paměti, místa na discích a cyklů CPU, tedy procesorového času) všem ostatním programům, které jsou na počítači spuštěny (na obrázku Memory Management a Scheduler) a zajišťuje synchronizaci procesů a komunikaci mezi nimi (Inter Process Communication).
Systémové programy provádí nejrůznější interní úlohy na vyšších úrovních, často vystupují v roli serverů obsluhujících různé klienty. Unixové systémy disponují také velmi dobře promyšlenou sadou pomocných programů, mezi které patří i některé utility pro správu systému. Na obrázku spadají do User Level vrstvy.



Z obrázku můžeme vidět, že operační systém běží 'nad' hardwarem. Jeho nejnižší vrstvou je subsystém řízení hardwaru (Hardware Control), který se stará o správu fyzických výpočetních prostředků (tedy již zmíněného procesoru, paměti, disků a periferií), nad nímž běží procesy ovladačů jednotlivých hw zařízení a nad těmi souborový subsystém, jejichž prostřednictvím systém vůči uživateli abstrahuje od složitosti hardwaru (vše v Unixu je soubor – viz. níže Vlastnosti OS Unix). Dokonce ne jen na jednotlivá zařízení, ale i na každý proces se lze dívat jako na soubor.

Jakožto své API poskytuje každý unixový OS tzv. systémová volání (System Call Interface). Ta se dají přirovnat například k WIN32API systémů Windows či voláním služeb OS DOS. Mohou je užívat jednotlivé aplikační i systémové programy a jejich prostřednictvím komunikovat s jádrem a využívat jeho služeb.

Další vlastnosti OS Unix

Jednoduchý (keep it simple and stupid) – byl navržen tak, aby bylo snadné jej úspešně implementovat; základní filozofií je, že na vše v Unixu se lze dívat jako na soubor – ne jen na datové soubory uložené na záznamových prostředcích, ale i na každé zařízení, periferii či programový proces.
Důsledek je, že při programování na vyšší úrovni není třeba řešit, jak které zařízení funguje – stejný výstup/vstup můžu poslat jak na pevný disk, tak na CD/DVD či na tiskárnu nebo do sítě (tedy zařízení představující připojení k síti) a operační systém se sám postará o příslušnou konverzi na jimi uživaný interní formát.
Víceúlohový (multitasking, time-sharing)
- umožňuje současný běh více úloh současně a sdílení procesorového času (tzv. realtime běh) pomocí plánování

procesů
- unixové operační systémy implementují preemptivní multitasking
- poskytuje jednotlivým procesům ochranu paměti (jiný proces nemůže zasahovat do oblasti paměti, kde běží jiný proces a tím narušit jeho běh)

Vícevláknový (multithreading) – nemusí platit pro všechny verze OS
- proces se může větvit a následně skládat z několika současně běžících vláken

Víceuživatelský (multi-user)
- na jednom počítači může současně být přihlášeno a pracovat více uživatelů; ti spolu mohou navzájem komunikovat, pokud je to povoleno, ale jeden uživatel nemůže přistupovat k souborům a procesům, které patří jinému, pokud to tento nedovolí (identifikace a vzájemná ochrana uživatelů)

Multiprocesorový (multiprocessing) – neplatí pro všechny verze OS
- jednotlivé procesy a jejich vlákna mohou běžet na více prosesorech současně (u více procesorových čí vícejádrových počítačů)

Přenositelný (portable)
- jelikož je téměř celý napsán v jazyce C, což je obecný programovací jazyk, pro který existují kompilátory pro nesčetné množství procesorových architektur, je možné ten samý systém provozovat na různých typech procesorů – a tím i počítačů (x86, PowerPC, a dalších 32 i 64 bitových mikroprosesorech)

Interaktivní přístup s možností vytváření dávek příkazů
- součástí je shell (tedy textové uživatelské rozhraní – CLI, Command Line Interface) jako rozhraní uživatele a interpret skriptovacího jazyka
- pomocí shellu lze též řídit úlohy (pozastavovat, ukončovat, poslat na pozadí, vyvolat na popředí)

Důraz na vztahy a komunikaci mezi programy
- umožňuje přesměrování a řetězení vstupů a výstupů příkazů

Bezpečný hierarchický systém souborů
- systém poskytuje přehledný strom adresářů s jedním kořenem, do nich lze organizovat soubory; automaticky pro ně alokuje pottřebné místo; jednotlivým souborům a adresářům lze přiřazovat různá práva na bázi identity uživatelů

Podpora práce v síti
- historicky nejprve komunikace mezi dvěma počítači – protokol **UUCP** (**Unix-to-Unix Copy**) – sada protokolů a utilit, které zajišťují spuštění vzdálených programů, přenosy souborů, e-mailů a diskuzních skupin Usenetu
- později podpora protokolů TCP/IP, NFS a dalších moderních široce používaných protokolů

Grafické prostředí
- na Unixu byl implementován grafický server X Window System, který umožňuje vytvořit grafické uživatelské prostředí na bázi oken ve spolupráci s tzv. správcem oken (Windows Managerem), pracuje na modelu klient-server

Virtuální grafický terminál
- shell běžící v okně grafického uživatelského prostředí; oken může běžet v rámci GUI teoreticky nekonečně mnoho
- nad X byla vystavena různá grafická uživatelská prostředí (CDE, GNOME, KDE,…)

Typy kernelu (jádra operačního systému)

Monolitické jádro

Monolitické jádro je druh jádra operačního systému, jehož veškerý kód běží v jednom (jaderném) paměťovém prostoru, který se anglicky označuje jako kernel space. V monolitickém jádru tedy všechny služby operačního systému běží spolu s hlavním vláknem jádra ve stejné oblasti paměti. To umožňuje neomezený a efektivní přístup k hardware a velkou efektivitu. Hlavní nevýhodou je závislost mezi systémovými komponentami - chyba v libovolném subsystému (např. ovladači zařízení) může shodit celý systém - a fakt, že velká jádra mohou být při vývoji těžko udržovatelná.
Jádro Linux je typickým příkladem monolitického jádra.

Mikrojádro

Mikrojádro je typ jádra operačního systému, které je velmi malé, protože obsahuje jen nejzákladnější funkce (typicky správu paměti a podporu pro plánování procesů a meziprocesovou komunikaci), čímž se minimalizuje objem běžícího kódu v privilegovaném režimu. Ostatní potřebné části jádra jsou řešeny v uživatelském prostoru jako běžné procesy (u mikrojader se označují servery) - například správa souborového systému, ovladače zařízení, podpora protokolů pro počítačové sítě a další. Mezi zástupce systémů s mikrojádrem patří MINIX, GNU Mach (což je také unixové jádro), PikeOS, QNX a Symbian OS.
Výhodou návrhu je snadnější programování systému pomocí rozdělení na více samostatných logických celků a vyšší přehlednost kódu. Jednoduchost též umožňuje snadnější zajištění bezpečnosti a vysoké spolehlivosti.
Nevýhodou bývá menší efektivita práce jádra a tím nižší výkon.

Hybridní jádro
Hybridní jádro operačního systému se snaží kombinovat vlastnosti monolitického jádra a mikrojádra za účelem získání výhod obou vyhraněných řešení. Používá ho např. systémy Windows řady NT (2000, 2003, XP, Vista, 7).

Unixová jádra jsou typicky spojována s monolitickým typem jádra, ale není to pravidlem.

Novější generace monolitických jáder přidávají podporu zavádění tzv. modulů za běhu, a proto není nutné při přidání dalšího hardware restartovat celý systém (např. USB flash disku). Takto lze ale zavádět dynamicky i jiné funkce (například podpora pro další síťové protokoly, souborové systémy). Stačí za běhu nahrát moduly, které se zavedou do adresního prostoru jádra a propojí se s jeho funkcemi. Modulární jádro ale i s moduly zůstává monolitickým jádrem, protože vše běží v jediném paměťovém prostoru.

Další pojmy

Zavedení jádra OS
Jádro se zavádí jako první část operačního systému při jeho startu. K jeho zavedení slouží tzv. bootloader, jenž je volán rutinou BIOSu, který je zaveden bezprostředně po zapnutí počítače. Moderními zavaděči jsou LILO, GRUB, FreeBSD bootloader.

Správa paměti
Jádro OS má úplný přístup k systémové paměti a musí umožnit procesům bezpečně přistupovat k této paměti, pokud to tyto procesy vyžadují. Obvykle se používá virtuální adresování realizované stránkováním, segmentací nebo kombinací obou dvou. Díky virtuálnímu adresování máme zaručenu ochranu paměťového prostoru každého procesu a navíc získáme možnost adresovat více operační paměti, než má systém fyzicky k dispozici.

Správa V/V zařízení
Jádro OS musí udržovat seznam dostupných V/V zařízení. Tento seznam může být znám dopředu (např. u embedded systémů, kde je kompletní HW znám předem), určený uživatelem (typické pro starší PC) nebo jsou jednotlivá zařízení detekována a přidávána do seznamu operačním systémem za běhu (tzv. Plug And Play).

Program, proces
Program je soubor na disku, který si můžete spustit a on bude vykonávat nějakou činnost.
Když ho spustíte, nahraje se do paměti a tím se z něj stane proces. Jeden program (např. textový editor vi) tedy může být spuštěn více uživateli. Každý z nich má ale svůj vlastní proces (instanci běhu programu).

Démon
Je označení programu, který je spuštěn dlouhodobě v paměti (tzv. rezidentní program) a není v přímém kontaktu s uživatelem (na rozdíl od běžných aplikací). Démon je součástí multitaskingových operačních systémů a obvykle se spouští při startu. Jeho úkolem je vyčkávat v nečinnosti na nějakou událost, tu posléze obsloužit a zajišťovat tak různé úkoly a funkce bez nutnosti interakce s uživatelem (obsluha počítačové sítě, tiskové fronty a podobně).
Démon je typickým programem v unixových systémech, kde zajišťuje různé systémové funkce. Jeho název obvykle končí na písmeno „d“, čímž se naznačuje jeho status démona (httpd, sshd, syslogd, klogd, ntpd, …). Nejdůležitějším démonem je zde démon init (u novějších systémů upstart nebo systemd), který má číslo procesu PID 1. Je spuštěn při startu jako první (s výjimkou kernel threadů) a je rodičem všech ostatních procesů v systému a řídí přechod do nastaveného runlevelu, což je režim běhu systému (jednouživatelský, víceuživatelský, v textovém či grafickém režimu). Je zodpovědný za spuštění skriptů při startu systému, adoptuje procesy, které ztratí svého rodiče (stane se jejich novým rodičem) a plní i další úkoly.
Je analogií služeb (services) ve Windows.

Knihovny
Anglicky library, jsou též klíčovou součástí systému. Díky tomu, že je GNU/Linux open source, velké množství kódu je právě v knihovnách. Jedná se především o věci standardní, jako jsou např. operace se soubory .jpeg, či matematické funkce jako sin, ale i existují například i rozsáhlé knihovny GTK a Qt, sloužící pro vykreslování tlačítek a dalšího prvků grafického uživatelského rozhraní. Pomocí knihoven lze snadno psát nové programy bez vymýšlení již jednou vymyšlených funkcí a navrženého kódu.

Interpret příkazů

Nebo také shell je proces interpretující příkazy zadávané uživatelem v textovém prostředí CLI.
Je to prostředek komunikace uživatele s operačním systémem a jako takový tvoří textové rozhraní mezi uživatelem a operačním systémem. Historicky prvním a dodnes nejdůležitějším unixovým je Bourne shell, pojmenovaný po svém autorovi A. S. Bourne, který jej navrhl a naprogramoval na přelomu 60. a 70. let. Principy Bourne shell přejaly všechny následující příkazové interprety, obvykle jde pouze o jiný uživatelský přístup.

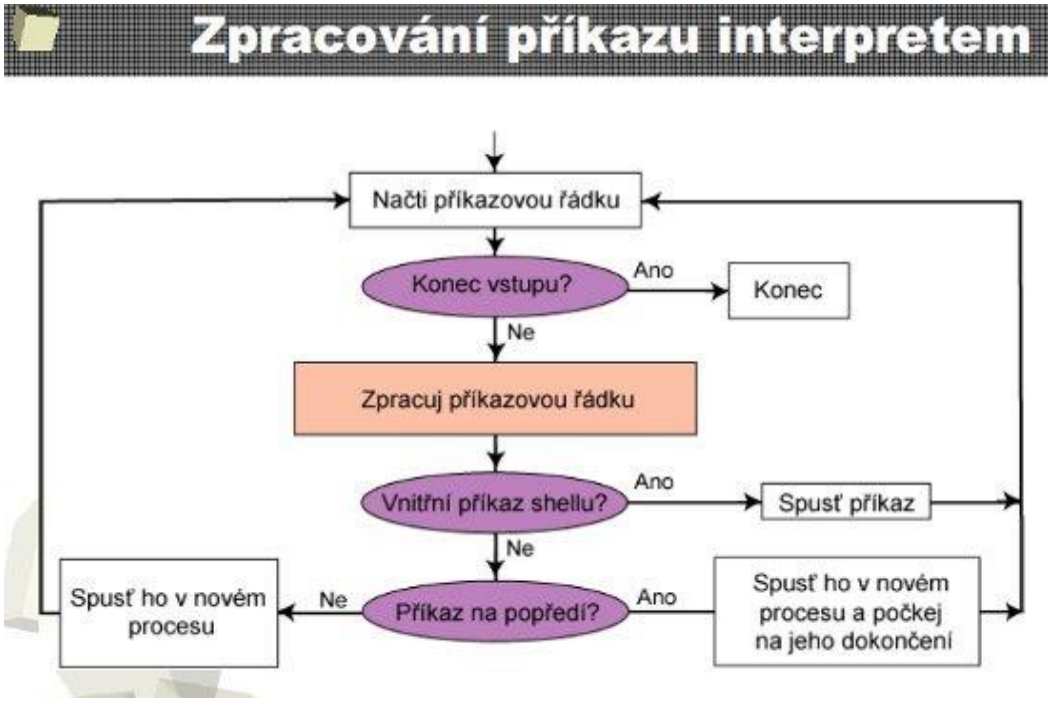
Shell slouží také jako interpretovaný (překládaný za běhu) programovací jazyk.
Můžeme v něm definovat proměnné, které řídí chování našeho unixového sezení či je užívat pro vytvářené skripty a programy. Interpret může rovnou přebírat jednotlivé příkazy od uživatele a ihned je vykonávat (interaktivní režim) nebo dokáže provádět příkazy uložené ve skriptu v dávce (dávkový režim). Skript = přikazy Unixu + řídicí struktury, dávkový režim.

Interpret příkazů Bourne Again Shell - Bash

Bourne Again Shell je dnes standardním interpretem příkazů v Linuxu založený na dříve nejpoužívanějším unixovém shellu - Bourne shellu. Je jeho značně inovovanou a zmodernizovanou verzí.
Jelikož je součástí projektu GNU, což je projekt pro vývoj otevřeného operačního systému a softwaru, nebylo problémem ho portovat i na ostatní unixové systémy, takže se jeho znalost uplatní i mimo samotný GNU/Linux.
Jeho funkcionalitu můžeme rozdělit na 3 základní části:
V interaktivním režimu čeká na zadání příkazu od uživatele. Příkazy mohou být buď přímo zabudované v shellu nebo samostatné programy napsané téměř v libovolném programovacím jazyku. Pro práci v CLI, ovládání shellu a řízení procesů umí také interpretovat škálu klávesových zkratek.

Pomocí systémových proměnných umožňuje přizpůsobení pracovního prostředí. Některé z těchto proměnných jsou přednastaveny systémem, ostatní nastavuje uživatel např. v inicializačních souborech při spuštění shellu.

A v neposlední řadě, jak již bylo zmíněno výše, je to také velice mocný programovací nástroj. Když nám chybí nějaký program nemusíme ho hned psát v kompilovaném jazyku (C, C++, Ada, Java), ale je možné problém vyřešit vytvořením skriptu.



Program spouštějící shell se vetšinou nachází v adresáři **/bin/bash**

Speciální soubory shellu
/etc/shells - použitelné přihlašovací shelly
/etc/adduser.conf - výchozí hodnoty pro adduser
/etc/profile - načítaný při přihlášení
$HOME/.bash_profile - načítaný při přihlášení
$HOME/.bashrc - načítaný při startu interpretu
$HOME/.bash_logout - načítaný při odhlášení
$HOME/.bash_history - evidence naposledy prováděných příkazů

Vypsání hodnot všech proměnných známých aktuálnímu interpretu příkazů:
```
$ set
```

Proměnná $PS1 definuje tvar primárního promptu (zobrazuje se, když shell čeká na zadání příkazu). V definici tvaru proměnných $PSn (n = 1, 2, 3, 4) můžeme použít sekvence se **speciálním významem.**

Seznam implicitního namapování klávesových zkratek
TAB : Automatické doplnění od pozice kurzoru.
`Ctrl + a` : Přesun kurzoru na začátek řádku (stejně jako Home).
`Ctrl + e` : Přesun kurzoru na konec řádku (stejně jako End).
`Ctrl + p` : Napíše předchozí příkaz (stejně jako Up).
`Ctrl + n` : Napíše následující příkaz (stejně jako Down).
`Ctrl + r` : Napíše dříve použitý příkaz obsahující zadaný výraz. Opakované stisknutí vyhledá další příkaz obsahující požadovaný výraz.
`Ctrl + s` : Napíše nejnovější příkaz odpovidající předchozímu vyhlednávání (vyvarujte se pouzití v terminálu, jelikož tato zkratka také spouští jehe XOFF). Pokud změníte nastavení XOFF, použijte Ctrl + q pro návrat.
`Ctrl + o` : Vykoná nalezený příkaz.
`Ctrl + l` : Vymaže obsah obrazovky (stejně jako příkaz clear).
`Ctrl + u` : Vymaže obsah řádku před kurzorem a zkopíruje jej do schránky.
`Ctrl + k` : Vymaže obsah řádku za kurzorem a zkopíruje jej do schránky.
`Ctrl + w` : Vymaže slovo před kurzorem a zkopíruje jej do schránky.
`Ctrl + y` : Vloží obsah schránky na pozici kurzoru.
`Ctrl + d` : Odešle příznak EOF (End Of File), který (pokud není tato funkce vypnuta v nastavení) zavře právě aktivní shell (stejně jako příkaz exit). (Pouze pokud je na řádku s kurzorem nějaký text.)
`Ctrl + c` : Odešle signál SIGINT právě probíhající úloze, který způsobí její přerušení a zavření.
`Ctrl + z` : Odešle signál SIGTSTP právě probíhající úloze, který úlohu pozastaví. Pro pokračování úlohy zadejte příkaz fg ['jméno nebo id procesu'].
`Ctrl + x Ctrl + x` : Přemístí kurzor na jeho předchozí pozici. Lze tak efektivně přeskakovat mezi dvěma kusy kódu.
`Ctrl + x Ctrl + e` : Zapne editaci zvoleného řádku pomocí editoru definovaného v $EDITOR nebo v vi editoru, pokud není nastaven.
`Alt + f` : Posune kurzor o jedno slovo dopředu.
`Alt + b` : Posune kurzor o jedno slovo dozadu.
`Alt + Del` : Vymaže slovo před kurzorem.
`Alt + d` : Vymaže slovo za kurzorem.
`Alt + u` : Přepíše velkými písmeny všechny znaky od pozice kurzoru do konce slova.
`Alt + l` : Přepíše malými písmeny všechny znaky od pozice kurzoru do konce slova.
`Alt + c` : Přepíše znak pod kurzorem velkým písmenem a posune ho na konec daného slova.
`Alt + a` : Zruší poslední editaci řádku.

Přehled některých základních příkazů

Dostupných příkazů a utilit je celá řada. Přehled a podrobnosti lze dohledat v manuálových stránkach.
Přehled nejduležitějších příkazů:

man – zobrazí systémový manuál (manuálové stránky)
info – zobrazí ineraktivní hypertextovou nápovědu pro utility GNU
hostname – zobrazí jméno systému
uname – zobrazí verzi operačního systému
echo – vypíše svůj argument na standardní výstup (obrazovku)

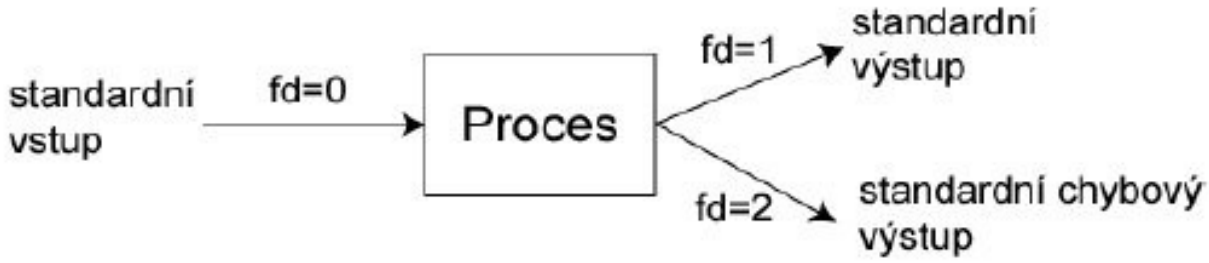
cp - kopíruje soubor
rm - ruší soubor
mkdir - vytvoří adresář
rmdir - ruší prázdný adresář
ln - vytvoří odkaz na soubory (tvrdý nebo symbolický)
chown – změní vlastníka souboru
chmod - mění přístupová práva k souborům
ls, dir, vdir - vypíše obsah adresáře
find - vyhledávání souborů
which - zobrazí absolutní cestu k programu
df - vypisuje informace o volném místu na disku
ps - informace o spuštěných procesech
cat - výpis souboru na obrazovku
more – stránkovaný výpis souboru na obrazovku
less – sofistikovaný stránkovaný výpis souboru na obrazovku
grep - tiskne řádky, které odpovídají zadanému vzoru
wc - vypíše počet písmen, slov a řádků souboru
sort - setřídí řádky vstupního souboru
uniq – odstraní duplicitní řádky ze vstupního souboru
xargs - spustí zadaný příkaz a zbylé argumenty čte ze standardního vstupu

Zpracování příkazové řádky
Každý příkaz vrací návratový kód. **0** = úspěšné provedení, **1,..,255** = chyba. Jednotlivé příkazy můžeme řetězit způsobem uvedeným níže.
příkaz & asynchronní provádění příkazu shell nečeká na jeho dokončení, příkaz se provádí na pozadí
příkaz1 ; příkaz2 sekvenční provádění příkazů, nejdříve se provedou příkazy před a pak příkazy za středníkem
(příkaz1 ; příkaz2) podshell, příkazy jsou spuštěny v nové instanci shellu
příkaz1 | příkaz2 roura, příkazy se startují zleva a běží paralelně, standardní výstup předchozího je standarním vstupem následujícího příkazu, návratový kód roury je návratovým kódem posledního příkazu
příkaz1 && příkaz2 sekvenční provádění příkazů, příkaz **za** se provede pouze tehdy, vrátí-li příkaz **před** nulový návratový kód (skončí bez chyby)
příkaz1 || příkaz2 sekvenční provádění příkazů, příkaz **za** se provede pouze tehdy, vrátí-li příkaz **před** nenulový návratový kód (skončí s chybou)
`** **příkaz** **` příkaz mezi opačnými apostrofy je proveden a nahrazen (včetně těchto apostrofů) svým std. výstupem

Pří práci se soubory ať už při jakékoliv manipulaci se soubory(find, rm , ..) můžeme využit náhrady jmen souborů kdy při zápisu nemusíme uvádět celé jméno souborů.
* odpovídá libovolnému řetězci kromě tečky na začátku a / kdekoliv
? odpovídá jednomu libovolnému znaku kromě tečky na začátku a / kdekoliv
[abc] [a-z] odpovídá jednomu znaku z uvedených znaků resp. z uvedeného intervalu
[!abc] [!a-z] odpovídá jednomu znaku mimo uvedených znaků
~ odpovídá domovskému adresáři (kromě sh)
~uživatel odpovídá domovskému adresáři daného uživatele (kromě sh)

Přesměrování vstupu/výstupu
Procesy přistupují k souborům pomocí tzv. deskriptorů souborů (0,1,2,…).
Každý proces má při spuštění standardně otevřeny tyto deskriptory:
0 – standardní vstup
1 – standardní výstup
2 – standardní chybový výstup
Nový proces standardně dědí deskriptory souborů od svého rodiče.
Pomocí speciálních znaků lze v shellu předefinovat jednotlivé deskriptory.
Přesměrování vstupu/výstupu
příkaz < soubor soubor bude otevřen a nastaven jako std. vstup příkazu
příkaz > soubor soubor bude otevřen jako std. výstup z příkazu. Pokud soubor

neexistuje, bude otevřen. Pokud existuje, bude přepsán (lze potlačit nastavením parametru noclobber v shellu – mimo sh)
příkaz » soubor soubor bude otevřen jako std. výstup z příkazu. Pokud soubor neexistuje, bude otevřen. Pokud existuje, bude výstup připojen na konec
příkaz « řetězec shell čte vstup až do řádky začínající daným řetězcem (tzv. here-document) načtený text se stane std. vstupem příkazu
příkaz 2> soubor soubor bude otevřen jako std. chybový výstup z příkazu, pokud soubor neexistuje, bude otevřen, pokud existuje, bude přepsán
příkaz >&n std. výstup bude zapsán do souboru určeného deskriptorem n
příkaz m>&n deskriptor n se přiřadí do deskriptoru m. Výstup příkazu do souboru určeného deskriptorem m se přesměruje do souboru určeného deskriptorem n
příkaz > soubor 2>&1 Std. výstup i std. chybový výstup bude zapsán do souboru.



Příklady

```
$ JMENO=Honza
$ echo $JMENO
Honza
$ jmeno=„Petr Jitka Tomas“
$ echo $jmeno
$ ls
$ ls –lia
$ ls -lia /etc
$ ls -i *
$ rm -i *
$ echo *
f1 f2 f3 f31 f32 f33 f34 f35
$ echo \*
*$
echo '*' „*“
* *
$ echo „Adresar $PWD obsahuje `ls|wc -l` souboru“
Adresar /home/k336/trdlicka obsahuje 8 souboru
$ date > v.txt
$ finger » v.txt
$ cat v.txt
Sat Sep 30 17:23:11 MEST 2006
Login Name TTY Idle When Where
skvorj Jiri Skvor pts/2 3:30 Sat 13:46 skvor.felk.cvut.cz
trdlicka Jan Trdlicka pts/19 Sat 16:56 r5bx111.net.upc.cz
Petr Jitka Tomas
$ find / 1> v.txt 2> /dev/null &
$ find / 1> v.txt 2>&1 &
$ mail honza < zprava
$ cat > s.txt « KONEC
```

```
> Dobry den.
> Jak se mate?
> Ja dobre. KONEC
> KONEC
$ cat s.txt
Dobry den.
Jak se mate?
Ja dobre. KONEC
```

Souborový systém OS Unix

Soubor je posloupnost bytů identifikovaná jménem.

Typy souborů

- obyčejné soubory (textové soubory, binární soubory)
- adresářové soubory (čili adresáře)
- speciální soubory (reprezentují rozhraní mezi zařízením a OS)
- symbolické linky (soubory, které obsahují odkaz na jiný soubor)
- pojmenované roury (named pipes - mechanismus spojování procesů do řetězce pomocí
- vzájemného propojení standardních proudů tak, že výstup stdout procesu je nasměrován do vstupu stdin následujícího procesu)
- sockety (prostředek meziprocesní komunikace, kdy komunikující procesy nemusí být na stejném počítači)
- adresář (přesněji adresářový soubor) je zvláštní typ souboru, který poskytuje vazbu mezi jmény souborů a lokací vlastních souborů v souborovém úložišti - adresář je tedy jakousi tabulkou adresářových položek
- každá adresářová položka obsahuje jméno souboru a číslo jemu odpovídajícího i-uzlu (i-node)
- i-node obsahuje všechny další informace nutné pro práci se souborem, jako je:
    - vlastník souboru
    - přístupová práva,
    - počty linků...
- každému souboru v systému souborů (obyčejnému souboru, adresářovému souboru, speciálnímu souboru) odpovídá právě jeden i-uzel
- na jeden a tentýž i-uzel může být více odkazů z různých adresářů: dvojice jméno souboru+i-uzel se nazývá link
- do adresáře (jakožto do souboru) nelze přímo zapisovat, obsah adresáře lze měnit

Stromová adresářová struktura

Operační systémy typu UNIX používají systém souborů uspořádaných do stromové struktury.
Počátek struktury se nazývá **kořenový adresář** (root directory, zápis jeho jména je /).
Pod tímto adresářem jsou umístěny další adresáře (obsahující opět adresáře a soubory atd.) a soubory.
Adresáře představují větvení stromu, soubory představují listy.
Každý adresář obsahuje dva speciální (pod)adresáře:
adresář tečka (.), který označuje adresář samotný
adresář dvě tečky (..) označuje (odkazuje) na rodičovský adresář

Typy adresářů

aktuální (pracovní) adresář (working directory), je adresář, ve kterém se uživatel momentálně nachází (označuje se .)
rodičovský (nadřízený) adresář (parent directory), je adresář o jednu úroveň výše směrem ke kořenovému adresáři (označuje se ..)
domovský adresář (home directory), je adresář nastavený uživatelům po přihlášení do systému, jméno určeno správcem, pro každého uživatele jeden
domovský adresář superuživatele (root) je adresář /root
Absolutní cesta je spojení kořenového adresáře (/) s adresářem, ve kterém se nachází soubor (odtud úplné – absolutní - jméno souboru)
Relativní cesta je spojení od pracovního adresáře k adresáři, ve kterém se nachází soubor (odtud relativní jméno, relativní odkaz - vyjadřuje cestu k danému souboru relativně vzhledem k pracovnímu adresáři)

Jména souborů a adresářů

- maximální délka závisí na implementaci, obvykle 255 znaků
- nepovolené znaky: lomítko /

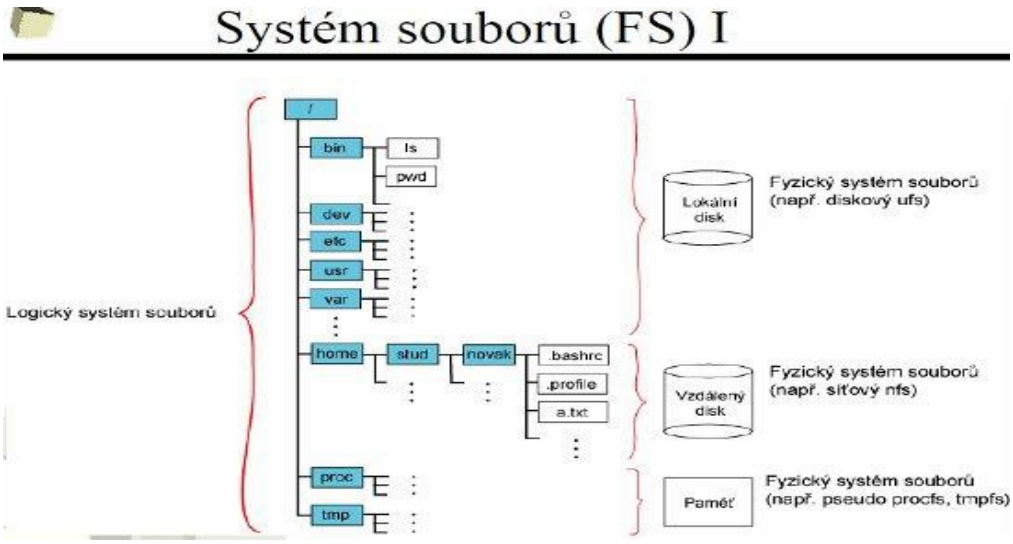
- nedoporučené znaky: * ? [ ] ( ) \ “ “ ‘ ‘ ` ` + ;
- doporučené znaky: alfanumerické znaky, tečky, pomlčka (ne na začátku), podtržítko
- rozlišování velkých a malých písmen
- jména začínající tečkou jsou skryté soubory/adresáře
- rezervovaná jména: tečka (.) aktuální adresář a dvě tečky (..) nadřazený adresář

Systém souborů
Slouží pro organizaci a uložení dat na permanentních paměťových médiích (např. HDD, FDD, CD-ROM, DVD-ROM).
Je to logické schéma realizované na fyzickém paměťovém médiu, transformaci logických adres na fyzické adresy zajišťuje ovladač příslušného zařízení (např. ovladač HDD).
Systém souborů lze definovat jako množinu logických diskových bloků, která má přesně definované uspořádání.
Velikost logických diskových bloků závisí na typu systému souborů (512, 1024, 2048, 4096 b ..).
Každé paměťové médium má určitou fyzickou organizaci, např. HDD se skládá z povrchů, na kterých jsou stopy. Stopy jsou soustředné kružnice na diskovém povrchu, které jsou rozdělené do sektorů (úseků stejné velikosti). Válec je tvořen stopami nad sebou přes všechny povrchy. Základní paměťový fyzický diskový blok je sektor (512 byte).
Z hlediska logické organizace bývá fyzický HDD rozdělen na několik oddílů (partitions) (min 1). Ke každé partition lze přiřadit systém souborů, existují však partitions, ke kterým není systém souborů přiřazen (swap, odkládací prostor).
Základní charakteristiky systému souborů
- hierarchická struktura
- konzistentní přístup k datům souborů
- schopnost vytvářet a rušit soubory
- možnost dynamických změn velikosti souborů (zvětšování)
- zajištění bezpečnosti dat v souborech (mechanismus přístupových práv)
- přístup k periferním zařízením přes speciální soubory
- uspořádání dat do bloků
- mechanismus připojování a odpojování systémů souborů (mount/umount)

Logický systém souborů
- pro běžného uživatele se jeví jako jediná homogenní struktura(strom adresářů)
- přistupujeme k němu např. pomocí příkazů cd, pwd, ls, cp, rm,
- ve skutečnosti je tvořen jedním nebo více fyzickými systémy souborů (viz. příkazy mount, df)

Fyzický systém souborů
podstrom adresářů, které je celý uložený na jednom fyzickém mediu (lokální disk, vzdálený disk, paměť,…)
typy souborových systémů
- diskové (ext2, ext3, ext4, ufs, vxfs, fat, hpfs, ntfs, reiserfs, zfs …)
- síťové (NFS, SMB…)
- pseudo (profs, tmpfs, fdfs,…)

Výhody této koncepce
- za pokrytí logického FS fyzickými FS je odpovědný administrátorpokrytí lze měnit bez vlivu na logický FS
- možnostzvětšování kapacity logického FS bez změny jeho struktury
- izolace chyb



Implementace FS

Rozloženi dat na fyzickém disku

Disk label
definuje rozdělení disku na menší oblasti (každá oblast může obsahovat jeden fyzický systém souborů)
Zavaděč OS
nahraje jádro OS do paměti a předá mu řízení
Super blok
obsahuje klíčové informace o celém systému souborů
Informace o volných datových strukturách
Tabulka i-uzlů
obsahuje atributy souborů a adresy dat. bloků, kde je uložen obsah souborů
i-node
Je datová struktura uchovávající metadata o souborech a adresářích používaná v unixových souborových systémech, které vycházejí z tradičního UFS (například linuxová řada ext2, ext3, ext4). Z důvodu zachování zpětné kompatibility ostatní souborové systémy i-uzly emulují (např. NFS). I-uzel obsahuje metadata pro každý libovolně velký soubor i adresář, například čas poslední změny, přístupová práva, seznam datových bloků a podobně. V adresářích jsou pak dvojice název souboru a k němu příslušné číslo i-uzlu, které pomocí metadat popisuje vlastní uložená datovou část souboru nebo adresáře. Vzhledem k tomu, že (téměř) vše je v unixových systémech soubor (vlastně i adresář je speciální soubor), je i-uzel univerzální datovou strukturou pro metadata. Počet i-uzlů je u klasických souborových systémů (ext2, ext3) určen při formátování systému souborů a později již nemůže být změněn. Jejich množství určuje maximální počet adresářů a souborů, které lze v souborovém systému vytvořit. I když může být na disku volné místo pro data (tj. volné datové bloky), nemusí být možné z důvodu nedostatku volných i-uzlů vytvořit další soubory a adresáře.

Soubor = jméno + atributy + data
Přístup k souboru
pomocí systémových volání: open(), close(), seek(), read(), write(), stat() ,…
příkazy OS: more , less, cp, rm, mv, ln,…

Základní příkazy práce se soubory a s adresáři
cp s1 s2 s2 nexistuje: zkopíruje soubor s1 na soubor s2. s2 existuje: přepíše s2 souborem s1
cp s1 s2 adr soubory s1 s2 zkopíruje adresáře adr
mv s1 s2 přesune/přejmenuje s1 na s2
rm <soubor> smaže soubor
file soubor vypíše informaci o typu souboru
cat soubor zobrazí obsah text. souboru
more soubor zobrazí obsah text. souboru po stránkách
less soubor zobrazí obsah text. souboru po stránkách
od –c soubor | more zobrazí obsah binárního souboru
strings soubor zobrazí tisknutelné znaky z binárního souboru
pwd vypíše jméno pracovního adresáře
cd adresář změní pracovní adresář
ls [-ladL] vypíše obsah adresáře adresář
mkdir [-p] vytvoří adresář/adresáře adresář
rmdir adresář smaže prázdný adresář
rm –r adresář smaže adresář (nemusí být prázdný)
cp –r adr1 adr2 adr2 neexistuje: vytvoří kopii adr1 pojmenovanou adr2.adr2 existuje: v adr2 vytvoří kopii adr1 (adr2/adr1)
mv adr1 adr2 adr2 neexistuje: přejmenuje adr1 na adr2. adr2 existuje: přesune **adr1 do adr2 (adr2/adr1)**

Důležité adresáře logického systému souborů dle standardu FHS (Filesystem Hierarchy Standard)
/ - Root, kořenový adresář, přítomný na všech souborových strukturách Unixu, je předkem všech souboru ve FS
/boot – statické soubory zavaděče systému
Obsahuje většinu souborů potřebných pro zavedení systému.
/bin – nepostradatelné binární soubory příkazů
Zde jsou uloženy ty nejzákladnější uživatelské programy. Ty představují nejminimálnější sadu programů nutných k tomu, aby uživatel mohl systém používat. Jsou tu uloženy takové věci, jako je shell, příkazy souborového systému (ls, cp) a podobně. Adresář /bin se po nainstalování systému už zpravidla nemění. Pokud ano, tak obvykle jen z důvodu upgradu softwarových balíčků.
/sbin – nepostradatelné binární soubory systému
Zde jsou programy, které může spouštět jen root a nebo které jsou spouštěny v průběhu startu systému. Normální uživatelé nemohou spouštět programy z tohoto adresáře.
/dev – soubory zařízení
Se všemi věcmi v Unixu se zachází jako se soubory. Takže i s hardwarovými
zařízeními jako např. sériovými porty, harddisky a scanery. Abychom mohli k těmto
zařízením přistupovat, musí být přítomen speciální soubor nazývaný „device node“.
Všechy „device nodes“ jsou uloženy v tomto adresáři.
/etc – systémová konfigurace lokálního počítače
V tomto adresáři jsou uloženy systémové konfigurační soubory. Všechno od konfigruačních souborů pro X Window, přes uživatelské databáze až po systémové
Také se zde nachází startovací skripty. Systémový administrátor se časem s tímto adresářem hodně sžije.
/home – domovské adresáře uživatelů
Unix je víceuživatelský operační systém. Každému uživateli je v systému
zřízen účet a jedinečný adresář pro jeho osobní soubory. Tento adresář je nazýván
jako uživatelův domovský (home). Adresář /home je používán jako defaultní lokace
pro umístění uživatelských domovských adresářů.
/lib – oduly jádra a knohovny, jež je možno zavést
Zde jsou uloženy systémové knihovny, které jsou vyžadovány pro základní operace. Najdeme tu například knihovnu jazyka C, dynamický zavaděč, knihovnu ncurses, moduly jádra, atd.
/tmp – dočasné soubory
Místo pro umisťování dočasně existujících souborů. Všichni uživatelé mohou z
tohoto adresáře číst i zapisovat do něj.
/opt – přídavné softwarové balíčky, volitelný software
Aplikace, které nejsou součásti systému
/usr – druhá hlavní hiearchie
Toto je největší adresář v linuxovém systému. Všechno ostatní musí jít hezky sem: Programy, dokumentace, zdrojový kód kernelu a X Window systém. Většinu programů budete instalovat do tohoto adresáře.
/usr/bin – vetšina uživatelských příkazů
Obsahuje standardní pomocné programy Linuxu, tedy binární soubory, které nejsou nutné v jednouživatelském režimu.
/var – proměnlivá data
Systémové logovací soubory, cache data a programové zámky jsou ukládány sem. Toto je adresář pro často se měnící data.
/root – domovský adresář uživatele root
Správce systému - administrátor - je systému znám pod jménem root. Rootův domovský adresář je v /root místo v /home/root. Důvod k tomu je jednoduchý. Co kdyby /home byl na jiném diskovém oddílu než / a nemohl by být připojen? Root by se měl přirozeně přihlásit do systému a opravit tento problém. Kdyby byl jeho domovský adresář na poškozeném filesystému, asi by mu to přihlašování hodně zkomplikovalo.
/proc – virtuální souborový systém pro ukládání informací o jádru a procesech
Toto je velmi zvláštní adresář. Ve skutečnosti není částí souborového systému,
ale je virtuálním souborovým systémem, který poskytuje přístup k informacím jádra. Různé informace, které vám chce kernel dávat na vědomí, jsou vám dávány prostřednictvím „souborů“ v adresáři /proc. Rověž vy můžete zasílat informace do kernelu prostřednictvím některých těchto „souborů“. Vyzkoušejte si třeba cat /proc/cpuinfo.
/mnt – bod připojení pro dočasné připojování souborových systémů

Zdroje

Slajdy předmětu Y38UOS
http://service.felk.cvut.cz/courses/X36ADU/ [http://service.felk.cvut.cz/courses/X36ADU/]
http://cs.wikipedia.org/wiki/Kernel [http://cs.wikipedia.org/wiki/Kernel]
http://og2.aspweb.cz/Statnice/sw.aspx?Category=25 [http://og2.aspweb.cz/Statnice/sw.aspx?Category=25]
http://www.abclinuxu.cz/clanky/navody/bash-i [http://www.abclinuxu.cz/clanky/navody/bash-i]
http://www.kiv.zcu.cz/~txkoutny/common/ups/skoleni/skoleni-unix.html#toc1l [http://www.kiv.zcu.cz/~txkoutny/common/ups/skoleni/skoleni-unix.html#toc1l]
http://vse.ugic.net/doc/4IT425_Prednaska_I-2.pdf [http://vse.ugic.net/doc/4IT425_Prednaska_I-2.pdf]
http://vse.ugic.net/doc/4IT425_Prednaska_I-3.pdf [http://vse.ugic.net/doc/4IT425_Prednaska_I-3.pdf]
Mistrovství v Linuxu – Příkazový řádek, shell, programování