Fakulta chemickej a potravinárskej technológie
Slovenskej technickej univerzity v Bratislave

Fakulta prírodných vied Univerzity Konštantína Filozofa v Nitre

Peter Švec – Miroslav Fikar – Jozef Dzivák – Libor Vozár

Základy práce v operačnom systéme Solaris

Bratislava, Nitra

2005

Základy práce v operačnom systéme Solaris

Mgr. Peter Švec, doc. Dr. Ing. Miroslav Fikar,
Ing. Jozef Dzivák, prof. RNDr. Libor Vozár, CSc.

Recenzenti:
Bc. Adrien Farkaš
Mgr. Ján Skalka, PhD.

Za odbornú stránku skrípt zodpovedajú autori.
Text neprešiel jazykovou úpravou.

Schválené vedením FPV UKF v Nitre dňa 19. 9. 2005

ISBN 80-8050-881-X

OBSAH

Úvod

Príručka *Základy práce v operačnom systéme Solaris* vznikla na základe potreby poskytnúť študentom začínajúcim pracovať v operačnom systéme Solaris základný študijný materiál v slovenskom jazyku.

Vysokoškolský učebný text je vytvorený na základe vzdelávacieho materiálu *Fundamentals of Solaris™ 8 Operating Environment for System Administrators (SA-118)*, ktorý je súčasťou vzdelávacích kurzov Sun Microsystems, Inc..

Učebný text je primárne určený na štúdium pod odborným vedením učiteľa, poslúži však aj ako učebnica pre samoukov. Učebný text je vydaný vo forme kurzu, členeného na jednotlivé moduly, ktorých zvládnutie je základom potrebných vedomostí pre prácu v systéme. Učebný text nemožno považovať za kompletnú referenčnú príručku pre prácu v systéme Solaris. Ambíciou autorov bolo vytvoriť kompaktný a stručný vstupný materiál vedúci čitateľa k zvládnutiu základov práce v prostredí UNIX. Následne sa študentom odporúča siahnuť po referenčných materiáloch, ktoré sú súčasťou inštalačného balíka príslušnej verzie operačného systému, prípadne sú k dispozícii prostredníctvom internetu.

Učebný text je zameraný na základný popis práce v operačnom systéme Solaris. Blízkosť operačných systémov UNIX však umožňuje študentovi po zvládnutí jeho obsahu aplikovať získané vedomosti aj pri práci v inom komerčnom alebo voľne šíriteľnom prostredí UNIX.

O kurze

Cieľ kurzu
Kurz „Základy práce v operačnom systéme Solaris“ Vás naučí ako používať základné príkazy operačného systému Solaris.

Prehľad kurzu
Tento kurz je určený pre nových používateľov operačného systému Solaris. Naučíte sa základné vlastnosti operačného systému Solaris na príkazovom riadku vrátane:

- navigácie v systéme súborov
- nastavenia prístupových práv pre súbory
- používania textového editora `vi`
- používania príkazových shellov
- základného používania siete

Plán kurzu

- *Začíname*
  - úvod do operačného systému Solaris
  - vstup do systému
- *Operácie so súbormi*
  - prístup k súborom a adresárom
  - súborové a adresárové príkazy
  - hľadanie textov a súborov
  - bezpečnosť súborov
- *Editovanie textu*
  - visual editor (`vi`)
- *Ukladanie súborov*
  - archivácia používateľských dát
- *Pripájanie k iným hostom*
  - vzdialené pripojenia
- *Narábanie s procesmi*
  - systémové procesy
- *Operácie so shellom*
  - Korn shell
  - úvod do sed a awk
  - čítanie skriptov shellu

Prehľad podľa modulov

- Modul 1 – „Prehľad operačného systému Solaris“
- Modul 2 – „Prístup do systému“
- Modul 3 – „Prístup k súborom a adresárom“
- Modul 4 – „Súborové a adresárové príkazy“
- Modul 5 – „Bezpečnosť súborov“
- Modul 6 – „Hľadanie súborov a textov“
- Modul 7 – „Visual (vi) Editor“
- Modul 8 – „Archivácia používateľských dát“
- Modul 9 – „Vzdialené pripojenia“
- Modul 10 – „Systémové procesy“
- Modul 11 – „Korn Shell“
- Modul 12 – „Úvod do sed a awk“
- Modul 13 – „Čítanie skriptov shellu“

Ciele kurzu

Po ukončení tohto kurzu by ste mali vedieť:

- prihlásiť a odhlásiť sa z operačného systému Solaris a všeobecného desktopového prostredia – systém CDE
- napísať príkaz v príkazovom riadku na vykonanie funkcií operačného systému Solaris
- orientovať sa v strome adresárov operačného systému Solaris
- vytvárať súbory a adresáre
- pracovať s textovými súbormi
- používať príkazy na hľadanie adresárov a súborov
- meniť práva súborov a adresárov
- používať textový editor `vi`
- zálohovať a obnovovať používateľské súbory a adresáre
- používať základné sieťové príkazy
- vypísať aktívne používateľské procesy a selektívne ich ukončiť
- používať vlastnosti shellu na zjednodušenie vykonávania príkazov
- identifikovať a upravovať inicializačné súbory a čítať jednoduché skripty shellu.

Zručnosti získané podľa modulov

	modul												
	1	2	3	4	5	6	7	8	9	10	11	12	13
prihlásiť a odhlásiť sa z operačného systému Solaris a všeobecného desktopového prostredia – systém CDE		■							▒				
napísať príkaz v príkazovom riadku na vykonanie funkcií operačného systému Solaris		■	■	■	■	■	▒	■	■	■	▒	■	
orientovať sa v strome adresárov operačného systému Solaris			■										
vytvárať súbory a adresáre			■										
pracovať s textovými súbormi				■									
meniť práva súborov a adresárov					■								
používať príkazy na hľadanie adresárov a súborov						■							
používať textový editor vi							■						
zálohovať a obnovovať používateľské súbory a adresáre								■					
používať základné sieťové príkazy									■				
vypísať aktívne používateľské procesy a selektívne ich ukončiť										■			
používať vlastnosti shellu na zjednodušenie vykonávania príkazov											■		
identifikovať a upravovať inicializačné súbory a čítať jednoduché skripty shellu													■

Typografické zvyklosti a symboly

- `Courier` je používaný pre príkazy, súbory a monitorový výstup.
- **`Courier bold`** je používaný pre znaky a čísla, ktoré píšete.
- *`Courier italic`* je používaný pre premenné a zástupné znaky príkazového riadku, ktoré sú nahradené buď menom alebo hodnotou.
- *Palatino italic* je používaný pre názvy kníh, nové slová alebo pojmy alebo zvýraznené slová.

Operačný systém UNIX

Operačný systém (OS) je skupina programov, ktoré ovládajú počítač. Zabezpečuje spoluprácu hardvéru (prvky, ktorých sa môžete dotknúť – klávesnica, obrazovka, pevný disk) a softvéru (aplikačné programy, ktoré môžete spúšťať, napr. textový editor). Vytvára medzi nimi určitú vrstvu abstrakcie, ktorá o. i. umožňuje prenositeľnosť softvéru medzi rôznymi OS.

Niektoré počítače majú jednopoužívateľský OS, čo znamená, že v rovnakom čase môže počítač používať len jedna osoba. Mnohé staršie OS, ako napr. DOS mohli mať v rovnakom čase spustenú iba jednu aplikáciu. Ale takmer všetky počítače sú schopné robiť viac, ak majú viacpoužívateľský, viacúlohový operačný systém, ako napr. UNIX. Silné OS umožňujú mnohým používateľom využívať počítač v tom istom čase, a umožňujú im spúšťať niekoľko aplikácií alebo programov súčasne.

UNIX vznikol na prelome 60-tych a 70-tych rokov 20. storočia. Bol určený pre vedcov a profesionálnych používateľov, ktorí chceli veľmi silný a flexibilný OS.

UNIX obsahuje obrovské množstvo silných aplikačných programov. Väčšina z nich je voľne dostupná na Internete. Voľne dostupné sú nielen aplikácie, ale aj niektoré verzie UNIXu (napr. Linux).

Mnohí poskytovatelia služieb Internetu používajú UNIX pretože je robustný, flexibilný a ekonomicky výhodný. Je ho možné nainštalovať takmer na každý hardvér.

Existuje veľké množstvo rôznych verzií UNIXu. Donedávna existovali dve hlavné línie. Prvá línia vznikla v laboratóriách AT&T (vetva System V, posledná verzia je System V Release 4). Druhá línia pochádza z Kalifornskej univerzity v Berkeley (vetva BSD). Medzi hlavné komerčné verzie patria SunOS, Solaris, SCO UNIX, AIX, HP/UX a ULTRIX. Voľne dostupné verzie sú OpenSolaris, Linux, FreeBSD. NetBSD, OpenBSD a iné.

UNIX môže byť používaný tak ako bol pôvodne navrhnutý, na prácu cez terminál a len cez klávesnicu. Dnes už väčšina UNIXových systémov umožňuje používateľom pracovať aj v grafickom prostredí.

Keď sa naučíte používať UNIX na jednom systéme, budete vedieť, ako ho používať na akomkoľvek inom systéme.

1 Úvod do operačného systému Solaris

Ciele

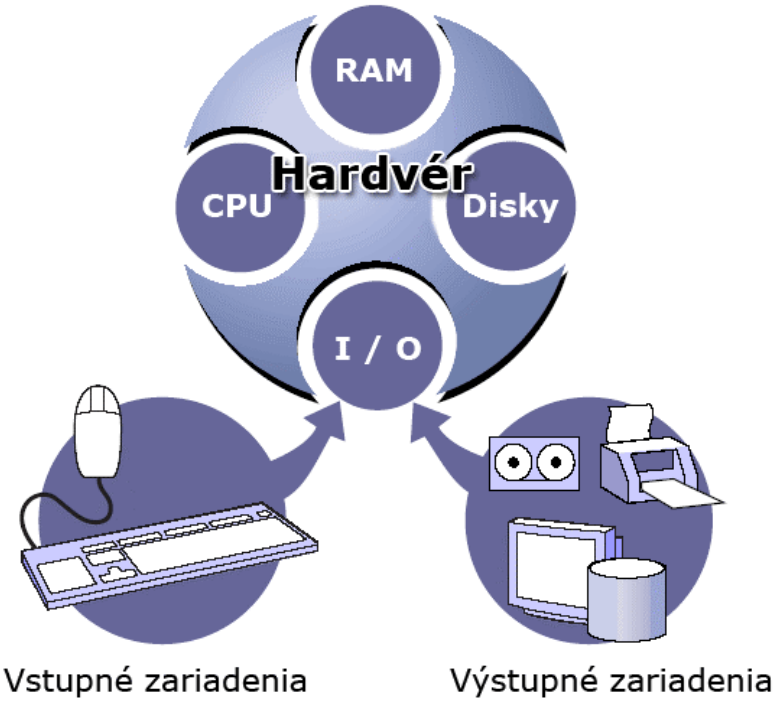
Po ukončení tohto modulu by ste mali vedieť:

- vymenovať 4 základné hardvérové komponenty počítača
- opísať 4 základné komponenty operačného systému Solaris
- vymenovať 3 základné komponenty operačného systému SunOS
- identifikovať základné shelly pre operačný systém Solaris

1.1 Hlavné komponenty počítača

Hlavnými komponentami počítača sú:

- operačná pamäť RAM (Random Access Memory)
- procesor (CPU – Central Processing Unit)
- vstupno / výstupné zariadenia (I/O – Input / Output Devices)
- pevný disk alebo iné zariadenie úschovy dát (mass storage devices)



Obr. 1 Hlavné komponenty počítača

1.2 Operačný systém Solaris

Operačný systém Solaris sa skladá z nasledovných 4 komponentov:

- operačný systém SunOS 5.x
- sieťové technológie na báze Open Network Computing (ONC+™) Technologies
- grafické používateľské rozhranie CDE (Common Desktop Enviroment)
- OpenWindows™ grafické prostredie

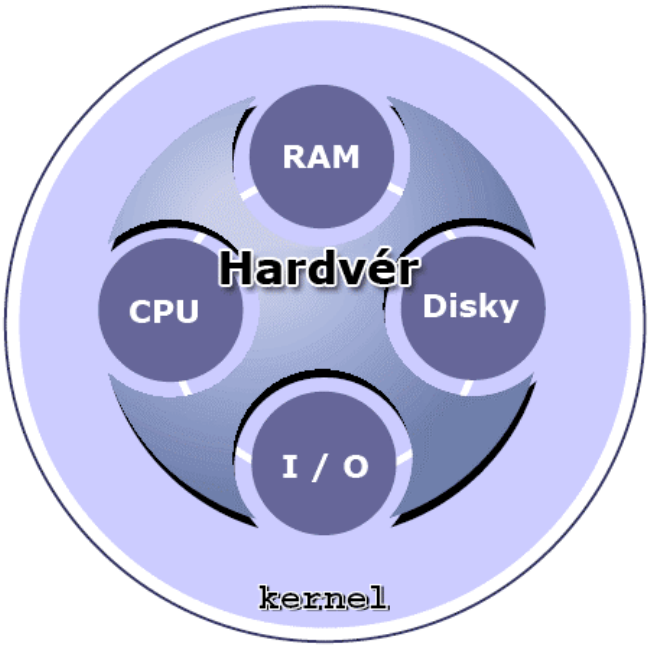
1.3 SunOS operačný systém

SunOS operačný systém sa skladá z 3 základných komponentov:

- kernel
- shell (príkazový procesor)
- správa súborového podsystému

Kernel tvorí jadro operačného systému SunOS a riadi všetky fyzické prostriedky počítača vrátane:

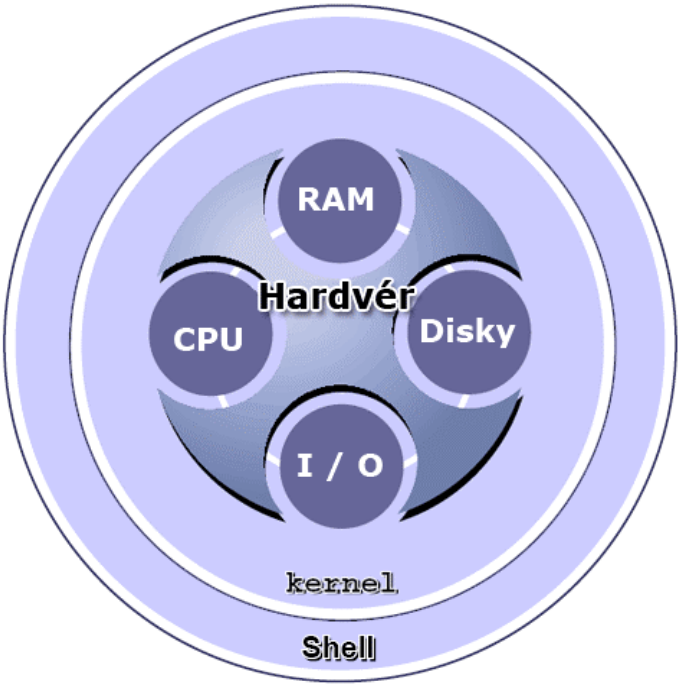
- systému súborov
- správy zariadení
- správy procesov
- správy pamäte



Obr. 2 Kernel

Shell je rozhranie, ktoré interpretuje príkazy používateľa kernelu. Solaris obsahuje 3 základné shelly:

- Bourne shell
- C shell
- Korn shell



Obr. 3 Shell

Operačný systém Solaris taktiež obsahuje ďalšie shelly:

- BASH – The GNU Bourne-Again shell
- Z shell
- TC shell – rozšírený C shell

Bourne Shell

- pôvodný shell UNIX®
- prednastavený shell používaný v operačnom systéme Solaris
- prednastavený prompt je znak doláru (`$`), pre super-používateľa znak mriežky (#)

C Shell

- vyvinutý Billom Joyom v Univerzite Berkeley v Kalifornii
- založený na programovacom jazyku C
- prednastavený prompt je meno počítača nasledované znakom percenta (*`hostname%`*)

Korn Shell

- nasledovník Bourne shellu
- odporučený ako štandardný shell pre klasických používateľov
- prednastavený prompt je znak doláru (`$`), pre super-používateľa znak mriežky (#)

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- vymenujte 4 základné komponenty počítača
- opíšte 4 základné komponenty operačného systému Solaris
- vymenujte 3 základné komponenty operačného systému SunOS
- identifikujte shelly dostupné v operačnom systéme Solaris

2 Prihlasovanie do systému

Ciele

Po ukončení tohto modulu by ste mali vedieť:

- vymenovať vlastnosti účinného hesla
- prihlásiť a odhlásiť sa zo systému z príkazového riadku
- prihlásiť a odhlásiť sa zo všeobecného desktopového prostredia (CDE)
- vykonať základné príkazy
- zmeniť svoje heslo
- určiť a opísať komponenty príkazového riadku
- používať kontrolné znaky na zmazanie príkazového riadku, zastaviť vykonanie príkazu a zastaviť a naštartovať výstup na monitor
- zobraziť on-line manuál
- hľadať v on-line manuáli podľa kľúčových slov
- identifikovať používateľov prihlásených do systému pomocou príkazov `who`, `whoami`, `id`
- vložiť viaceré príkazy na jeden príkazový riadok

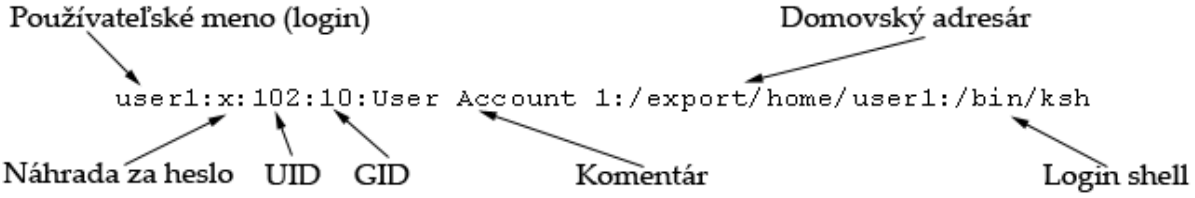
2.1 Používateľské kontá (účty)

V UNIXových systémoch rozdeľujeme používateľské kontá na dve skupiny:

- `root` konto
  - konto `root` je používateľské konto so špeciálnymi a neobmedzenými právami
  - konto `root` sa používa na špecifické administratívne zásahy v systéme
- používateľské kontá
  - sú všetky ostatné kontá v systéme, nie každé konto musí mať právo prihlásenia sa do systému
  - niektoré informácie o používateľských kontách sú uložené v `/etc/passwd` súbore

Záznam v `/etc/passwd`

V súbore `/etc/passwd` sa nachádzajú základné informácie o každom konte. Syntax každého riadku tohto súboru je nasledovná:



Obr. 4 `/etc/passwd` záznam

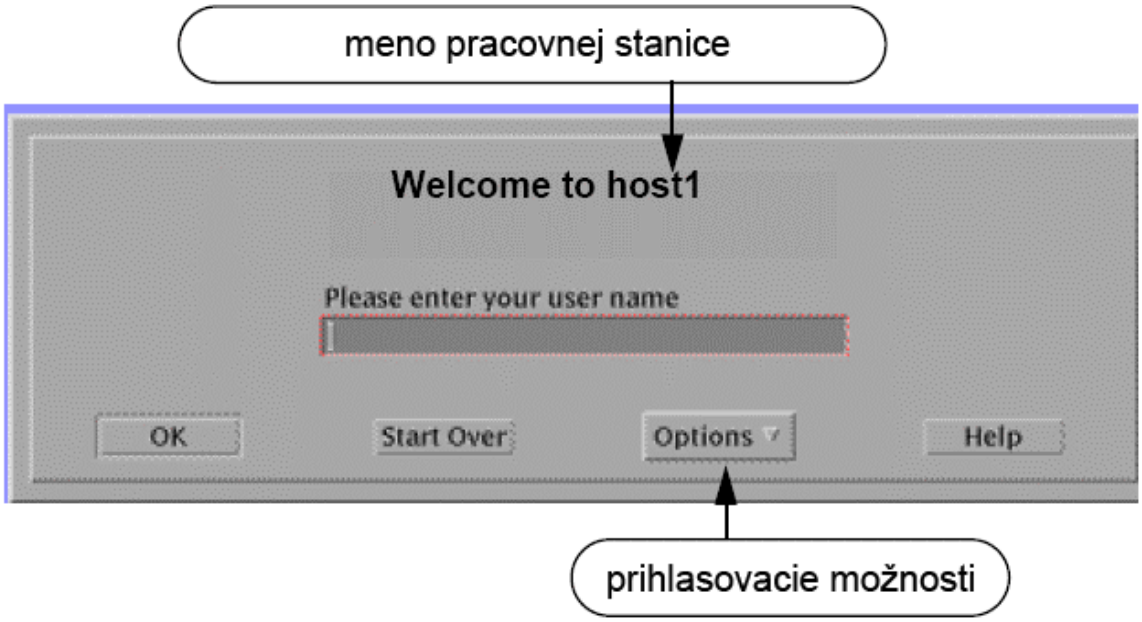
- pole *náhrada za heslo* (placeholder) určuje, že používateľovo zakódované heslo je uložené v súbore `/etc/shadow`
- UID je číslo – ID, jedinečné pre každého používateľa, je ekvivalentom pre login

- GID je číslo – ID, jedinečné pre každú skupinu v systéme, určuje primárnu skupinu používateľa

2.2 Prihlásenie do systému

Prihlasovací proces identifikuje systému používateľa. Do systému sa môžete prihlásiť cez

- prihlasovaciu obrazovku
- príkazový riadok



Obr. 5 Prihlasovacia obrazovka CDE

Prihlásenie cez prihlasovaciu obrazovku

1. do textového poľa zadajte svoje používateľské meno a stlačte ENTER alebo kliknite na tlačidlo OK
2. do textového poľa pre heslo zadajte svoje heslo a stlačte ENTER alebo kliknite na tlačidlo OK

Poznámka: prihlasovacie heslo sa pri zadávaní nezobrazuje, aby niekto blízko terminálu nemohol spočítať jeho počet znakov. Tiež je dobré si dávať pozor na numerickú časť klávesnice, ktorá býva štandardne často vypnutá, a teda čísla v hesle sú zadávané nesprávne.

Prihlásenie cez príkazový riadok

1. Zobrazte možnosti v ponuke OPTIONS a zvoľte prihlásenie cez príkazový riadok (Command Line Login). Prihlasovacia obrazovka zmizne a nahradí ju konzolový prompt.
2. Po stlačení ENTER sa zobrazí výzva pre zadanie používateľského mena.
3. Zadajte svoje používateľské meno (alebo prihlasovacie ID) a stlačte ENTER
4. Zadajte svoje heslo a stlačte ENTER.

2.3 Heslo

Požiadavky na heslo

Heslo chráni používateľské kontá pred nedovoleným prístupom. Pre heslo by malo platiť nasledovné:

- malo by mať 6 až 8 písmen.
- malo by obsahovať aspoň 2 znaky abecedy a musí obsahovať aspoň jeden číselný alebo zvláštny znak
- malo by byť rozdielne od prihlasovacieho mena
- malo by sa odlišovať od predchádzajúceho hesla aspoň troma znakmi
- môže obsahovať medzery
- pri zadávaní hesla sa zadávané znaky na monitore nezobrazujú, treba si tiež dať pozor na to, že numerická časť klávesnice býva často vypnutá

Zmena hesla

- časté zmeny používateľského hesla pomáhajú predchádzať nedovolenému prístupu do systému
- svoje heslo môžete zmeniť vo všeobecnom desktopovom prostredí CDE alebo z príkazového riadka
- na zmenu hesla cez príkazový riadok slúži príkaz `passwd`, ktorý si po spustení vypýta staré heslo, a ak je toto heslo správne, tak si vypýta nové heslo
- formát príkazu: `passwd`
- príklad:

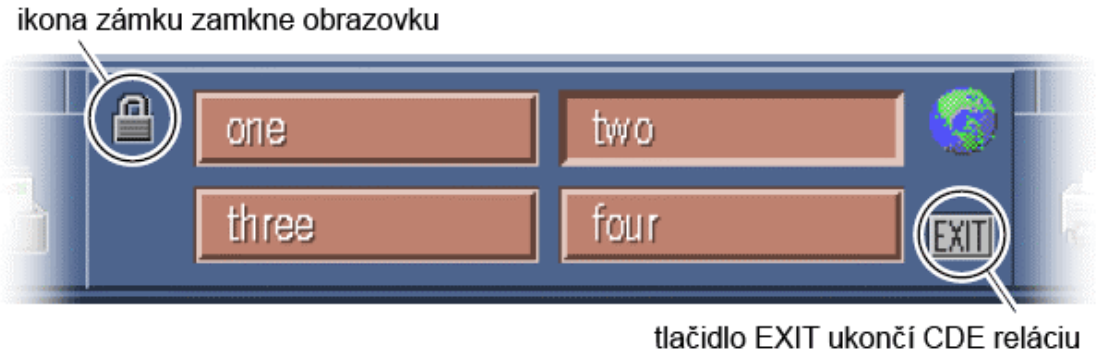
```
$ passwd
passwd: Changing password for angel
Enter existing login password:
New Password:
Re-enter new Password:
```

2.4 Zabezpečenie CDE relácie

Zabezpečenie CDE relácie zabraňuje neoprávneným používateľom pristupovať do systému.

Poznáme dva spôsoby zabezpečenia systému:

- zamknutie obrazovky
- ukončenie relácie
  - kliknutím na tlačidlo EXIT
  - použitím možnosti LOGOUT z ponuky pracovnej plochy (kliknutím pravým tlačidlom myši na prázdne časti obrazovky)



Obr. 6 Ovládacie prvky relácie

2.5 Základné príkazy v UNIXe

Medzi príklady základných príkazov v UNIXe patria:

- príkaz `uname`, ktorý zobrazuje informáciu o systéme
- príkaz `date`, ktorý zobrazuje aktuálny dátum a čas systému
- príkaz `cal`, ktorý zobrazuje kalendár pre aktuálny mesiac a rok
- príkaz `man`, ktorý zobrazuje on-line manuál

Syntax príkazového riadku

Základná syntax príkazu v Unixe obsahuje:

```
$ príkaz možnost(i) argument(y)
```

napríklad:

```
$ date                      (príkaz)
$ date mmddhhmmyy           (príkaz a argument)
$ cal 12 2000               (príkaz a 2 argumenty)
$ uname -a                  (príkaz a možnosť)
$ uname -rpns               (príkaz a niekoľko možností)
$ uname -r -p -n -s         (príkaz a niekoľko možností)
```

Kontrolné znaky

Počas vykonávania príkazu môžeme kedykoľvek použiť kontrolné znaky. Kontrolné znaky sa zadávajú stlačením klávesy CTRL.

Znak	Význam
Control-C	ukončuje momentálne bežiaci príkaz
Control-U	maže všetky znaky príkazového riadku
Control-S	zastavuje výstup na obrazovku
Control-Q	obnovuje výstup na obrazovku potom ako bolo stlačené Control-S
Control-D	určuje koniec súboru alebo ukončenie (exit)
Control-W	maže posledné slovo na príkazovom riadku

Prezeranie on-line dokumentácie

- on-line referenčné manuály pre Unix (*man pages*) poskytujú podrobné opisy príkazov a ich použitie
- formát príkazu: `$ man názov_príkazu`
- príkaz `man` je pravdepodobne najužitočnejší príkaz v systéme
  - umožňuje používateľovi preveriť správanie zriedkavo používaných príkazov alebo zriedkavo používaných možností
- pri otvorení manuálových stránok sú defaultne nastavené na ovládanie nasledovné klávesy: `medzera` (posun o stránku dole), `Enter` (posun o riadok dole), `b` (posun o stránku hore), `/ret` (vyhľadáva `ret` smerom dole), `n` (opäť vyhľadá reťazec dole), `q` (ukončí manuálové stránky)

Zisťovanie momentálne prihlásených používateľov

- Na zobrazenie zoznamu práve prihlásených používateľov slúži príkaz `who`. Vo výpise príkazu `who` sa nachádzajú nasledovné informácie:
  - prihlasovacie meno

- identifikátor portu prihlasovacieho terminálu (TTY)
- čas a dátum prihlásenia
- čas uplynutý od poslednej aktivity

- formát príkazu: `who`
- príklad:

```
$ who
domer            pts/1    Aug  3 09:48 (62.168.82.214)
angel            pts/2    Aug  1 10:55 (194.160.210.18)
bauer            pts/3    Aug  3 16:54 (213.129.252.186)
magic            pts/4    Aug  3 17:46 (217.144.30.108)
angel            pts/5    Aug  1 10:56 (194.160.210.18)
```

Identifikácia používateľa

- na identifikáciu vlastného prihlasovacieho mena slúži príkaz `whoami`.
- formát príkazu: `whoami`

Identifikácia detailov skupiny používateľa

- Na identifikáciu ID používateľa, používateľského mena, ID skupiny a mena skupiny používateľa systému slúži príkaz `id`.
- formát príkazu: `id` *`používateľské_meno`*
- príklad:

```
$ id
uid=105(angel) gid=10(staff)
```

Vkladanie viacerých príkazov na jeden príkazový riadok

- bodkočiarka (`;`) je špeciálny znak shellu a používa sa ako *oddeľovač príkazov.*
- bodkočiarka umožňuje vložiť niekoľko príkazov na jeden príkazový riadok. Shell vykoná každý príkaz zľava doprava po stlačení ENTER-u.

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- vymenovať charakteristiky účinného hesla
- prihlásiť a odhlásiť sa zo systému z príkazového riadku
- prihlásiť a odhlásiť sa zo všeobecného desktopového prostredia CDE
- vykonať základné príkazy
- zmeniť svoje heslo
- určiť a opísať komponenty príkazového riadku
- použiť kontrolné znaky na zmazanie príkazového riadku, zastaviť vykonávanie príkazu a zastaviť a spustiť výstup na obrazovku
- zobraziť stránky on-line manuálu
- hľadať stránky on-line manuálu podľa kľúčového slova
- identifikovať prihlásených používateľov v systéme pomocou príkazov `who`, `whoami`, a `id`
- vložiť viaceré príkazy na jeden príkazový riadok

3 Prístup k súborom a adresárom

Ciele

Po ukončení tohto modulu by ste mali vedieť:

- prezentovať rozdiel medzi názvami absolútnych a relatívnych ciest
- pristupovať k súborom a adresárom v adresárovom strome pomocou mien absolútnych a relatívnych ciest
- používať skratky mien ciest k prístupu k súborom a adresárom v adresárovom strome
- zobraziť obsah adresárov a určiť typ súborov
- identifikovať rozličné metaznaky shellu na skrátenie súborových mien a mien ciest

Názvy ciest

- jedinečne pomenovávajú určitý súbor alebo adresár špecifikovaním ich umiestnenia v adresárovom strome
- sú podobné cestným mapám, ktoré ukazujú ako sa dostať z jedného miesta v adresárovom strome na druhé
- lomky v mene cesty (`/`) oddeľujú názvy objektov
- prvá lomka v názve cesty vždy určuje koreňový (`root`) adresár

Názov absolútnej cesty

- *názov absolútnej cesty* špecifikuje súbor alebo adresár vo vzťahu k celému adresárovému stromu operačného systému Solaris
- názov absolútnej cesty vždy:
  - začína v adresári `root`(`/`) a potom zobrazuje každý adresár celou cestou až do cieľového objektu
  - používa lomku (`/`) na oddelenie viacerých názvov adresárov alebo súborov
- príklad:
  `/export/home/st2000/angel/preferences.pl`

Názov relatívnej cesty

- *názov relatívnej cesty* opisuje vzťah umiestnenia adresára alebo súboru k momentálnemu adresáru.
- názvy relatívnych ciest:
  - nikdy nezačínajú lomkou (`/`)
  - používajú lomku (`/`) v názve cesty ako oddeľovače medzi názvami objektov
- príklad:
  `st2000/angel/preferences.pl`

Zvyklosti pomenovávania súborov a adresárov

Pri tvorení názvov súborov alebo adresárov musíte dodržiavať nasledovné zásady:

- názvy adresárov a súborov môžu mať dĺžku max. 255 alfanumerických znakov
- v názvoch súborov a adresárov sú povolené aj nealfanumerické znaky
- nemali by sa používať zvláštne znaky
- nemali by sa používať medzery
- názvy môžu, ale nemusia obsahovať príponu
- názvy súborov a adresárov sú *case-sensitive* (rozlišujú sa veľké a malé písmená)

3.1 Práca s adresármi

Zmena adresára

- Po prihlásení sa do systému, je pracovným adresárom váš domovský adresár. Pracovný adresár zmeníte príkazom `cd`
- nadradený adresár je označený dvoma bodkami (`..`), aktuálny adresár jednou bodkou (`.`)
- Do domovského adresára sa dostaneme príkazom `cd` bez argumentov.
- formát príkazu: `cd` *`názov_adresára`*
- príklady:

```
$ cd ..
$ cd ./dokumenty/
$ cd /var/www/data/
```

Zobrazenie momentálneho adresára

- na identifikáciu momentálneho adresára slúži príkaz `pwd`
- príkaz `pwd` zobrazí názov absolútnej cesty momentálneho pracovného adresára
- formát príkazu: `pwd`
- `príklad:`

```
$ pwd
      /export/home/st2000/angel
```

Zobrazenie obsahu adresára

- na zobrazenie súborov a adresárov vnútri určeného adresára použite príkaz `ls`
- použitie príkazu `ls` bez argumentu zobrazí obsah momentálneho adresára
- formát príkazu: `ls [-`*`možnosti`*`] `*`názov_cesty`*`...`
- príklad:

```
$ ls -la
  total 20
  drwxr-x---   4 angel    staff        512 Aug  3 18:06 .
  dr-xr-xr-x  10 root     root         512 Jul 26 10:53 ..
  -rw-------   1 angel    staff        570 Aug  3 18:36 .bash_history
  -rw-r--r--   1 angel    staff        136 Jul 26 10:53 .cshrc
  -rw-r--r--   1 angel    staff        167 Jul 26 10:53 .login
  drwxr-xr-x   3 angel    staff        512 Aug  3 18:36 .mc
  -rw-r--r--   1 angel    staff        256 Aug  3 18:36 .profile
  drwx------   2 angel    staff        512 Aug  3 18:06 .ssh
```

Niektoré z možností príkazu `ls`

`-a:` zobrazí kompletný obsah, vrátane toho, čo začína bodkou, čo sa štandardne nezobrazuje

`-A:` ako "`-a`" ale bez „`.`" a „`..`"

`-d:` ak je argument adresár, potom nezobrazí obsah toho adresára

`-l:` zobrazenie v dlhom formáte

`-r:` triedenie pospiatky

`-R:` rekurzívne zobrazí obsah všetkých podadresárov na ceste

`-t:` triedi podľa dátumu zmeny

`-u:` použije triedenie podľa času (s možnosťami "`-t`" alebo "`-l`")

`-1:` výstup zobrazí len jeden údaj na riadok

Metaznaky shellu

- metaznaky shellu sú špecifické znaky, ktoré majú pre shell zvláštny význam
- niektoré príklady metaznakov shellu sú `~ - + * ? [ ].`
- niektoré metaznaky môžu byť v iných shelloch odlišné

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- ukázať rozdiel medzi názvom absolútnej a relatívnej cesty
- pristúpiť na súbor alebo adresár v rámci adresárového stromu za použitia názvov absolútnych a relatívnych ciest
- použiť skratky názvov ciest na prístup k súborom a adresárom v rámci adresárového stromu
- zobraziť obsah adresárov a určiť typy súborov
- identifikovať rozličné metaznaky shellu na skrátenie názvov súborov a ciest

4 Príkazy adresárov a súborov

Ciele

Po dokončení tohto modulu by ste mali vedieť:

- určiť typy súborov pomocou príkazu `file`
- zobraziť obsah textových súborov pomocou príkazov `cat`, `less`, `head`, a `tail`
- zistiť počty znakov, slov a riadkov pomocou príkazu `wc`
- vytvoriť prázdne súbory alebo vynoviť čas zmeny na existujúcich súboroch pomocou príkazu `touch`
- použiť príkaz `tee` na vytvorenie textu v súbore
- vytvoriť a odstrániť adresáre pomocou príkazov `mkdir` a `rmdir`
- spravovať súbory a adresáre pomocou príkazov `mv`, `cp`, a `rm`
- použiť príkazy na tlač súboru, zistiť stav tlačového frontu a zrušiť požiadavku na tlač
- formátovať a tlačiť obsah súborov pomocou príkazu `pr`

V OS UNIX sú všetky údajové štruktúry chápané ako súbory, pričom nie je dôležité, či ide o štandardné súbory, adresáre, zariadenia alebo sockety. Obrovskou výhodou tohto prístupu je, že so všetkými typmi sa pracuje rovnakým spôsobom (čítanie, zápis, nastavovanie práv).

Určovanie typu súboru

- na jednoduché zistenie typov niektorých súborov použite príkaz `file`
- formát príkazu: `file` *`názov_súboru(ov)`*
- výstup príkazu `file` sa často skladá z nasledovného (ale nemusí len z toho):
  - text
  - údaje
  - spustiteľný alebo binárny súbor
- súbor `/etc/magic` obsahuje údaje, podľa ktorých príkaz `file` určuje typ súboru
- príklady:

```
$ file .profile
      .profile:       ascii text

$ file /usr/bin/bash
      /usr/bin/bash:    ELF 32-bit MSB executable SPARC Version 1,
      dynamically linked, stripped
```

Zobrazenie obsahu súboru

- príkaz `cat` (concatenate) zobrazí obsah jedného alebo viacerých súborov na obrazovku
  - keď sú zadané viaceré názvy, tak sú súbory vypísané v príslušnom poradí
- príkaz `cat` môžete takisto použiť na vytvorenie krátkych textových súborov bez použitia editora. V tom príklade použijeme druhú možnosť a zapisovanie do súboru ukončíme klávesou Ctrl-D.
- formát príkazu:

```
cat [názov_súboru ...]
cat > názov_súboru
```

Prezeranie obsahu súboru

- na prezretie obsahu dlhého textového súboru slúži príkaz `less`
- formát príkazu: `less [`*`názov_súboru ...`*`]`

Zobrazenie niekoľkých prvých riadkov súboru

- príkaz `head` zobrazí prvých 10 riadkov súboru
- počet riadkov môžete zmeniť použitím možnosti `-n`
- formát príkazu: `head [-n]` *`meno_súboru...`*

Zobrazenie niekoľkých posledných riadkov súboru

- príkaz `tail` zobrazí posledných 10 riadkov jedného alebo viacerých súborov
  - použite možnosť -n na určenie počtu riadkov
  - použite možnosť +n na zobrazenie od riadku n po koniec súboru
- formát príkazu:

```
tail [-n] [názov_súboru]
tail [+n] [názov_súboru]
```

Použitie súčasne head a tail na zobrazenie rozsahu riadkov v súbore

Na zobrazenie riadkov 10 až 20 súboru nazvaného `yadda.txt` môžeme použiť nasledovné dve kombinácie príkazov:

```
$ head -20 yadda.txt | tail +11
```

alebo

```
$ tail +11 yadda.txt | head -11
```

Zobrazenie riadkov, slov a znakov v súbore

- príkaz `wc` zobrazí počet riadkov, slov a znakov obsiahnutých v súbore.
- formát príkazu : `wc [-c | -m | -C] [-lw] [`*`názov_súboru...`*`]`
- príklad:

```
$ wc .profile
      13      38     256 .profile
```

Vytvorenie prázdnych súborov

- na vytvorenie prázdneho súboru slúži príkaz `touch`
  - ak už súbor alebo adresár existuje, tak potom príkaz `touch` aktualizuje čas zmeny a prístupu súboru alebo adresáru
- formát príkazu: `touch` *`názov_súboru...`*

Vytváranie adresárov

- na vytvorenie adresára slúži príkaz `mkdir`. Adresáre môžete vytvoriť pomocou názvov absolútnych alebo relatívnych ciest.
- príkaz `mkdir` umožňuje určiť viac ako jeden adresár v jednom príkazovom riadku a naraz vytvorí viaceré adresáre
- formát príkazu :

```
mkdir názov adresára...
```

```
mkdir [-p] názov_adresára...
```

- možnosť `-p` vytvorí všetky potrebné rodičovské adresáre cieľového adresára ak neexistujú

Kopírovanie súborov

- na kopírovanie súborov a adresárov slúži príkaz `cp`
- príkaz `cp` kopíruje obsah jedného súboru do súboru druhého, alebo môže kopírovať viaceré súbory bez toho, aby prepísal už existujúce súbory
- formát príkazu:

```
cp -i zdrojovy_súbor cieľový_súbor
cp -i zdrojovy_súbor... cieľový_adresár
```

- možnosť `-i` si vypýta potvrdenie, či sa má prepísať existujúci cieľ.
  - odpoveď áno (`yes`) znamená, že sa prepis uskutoční
  - odpoveď nie (`no`) znemožní príkazu `cp` prepísať cieľ

Kopírovanie adresárov

- na kopírovanie adresára a jeho obsahu do iného adresára slúži príkaz `cp -r`
- formát príkazu:

```
cp -ir zdrojový_adresár(e) cieľový_adresár
```

- možnosť `-i` si vypýta potvrdenie, či sa má prepísať existujúci cieľ.
  - odpoveď áno (`yes`) znamená, že sa prepis uskutoční
  - odpoveď nie (`no`) znemožní príkazu `cp` prepísať cieľ

Premiestňovanie a premenovávanie súborov a adresárov

- na premiestnenie alebo premenovanie súboru alebo adresáru slúži príkaz `mv`
- formát príkazu:

```
mv -i zdroj cieľový_súbor
mv -i zdroj... cieľový_adresár
```

- možnosť `-i` si vypýta potvrdenie, či sa má prepísať existujúci cieľ.
  - odpoveď áno (`yes`) znamená, že sa prepis uskutoční
  - odpoveď nie (`no`) znemožní príkazu `mv` prepísať cieľ

Odstraňovanie súborov

- na odstránenie nepotrebných súborov a adresárov použite príkaz `rm`. Odstrániť môžete jeden alebo viac súborov naraz
- formát príkazu : `rm [ -i ] názov_súboru...`
- Príkaz `rm` nemôže byť vrátený späť. Keď si nie ste istí, použite možnosť `-i`. Táto možnosť spôsobí, že zmazanie každého súboru musíte potvrdiť.

Odstraňovanie adresárov

- odstráňte nepotrebné adresáre použitím príkazov `rmdir` a `rm`
  - príkaz `rmdir` odstraňuje iba prázdne adresáre
  - príkaz `rm -r` odstraňuje adresáre, ktoré obsahujú súbory.
- formát príkazu:

```
rmdir názov_adresára(ov)
rm -r[i] názov_adresára(ov)
```

Tlač z príkazového riadku

- príkaz `lp` zaraďuje textové súbory na tlač
- z príkazového riadku môžete vytlačiť:
  - ASCII text
  - PostScript™ súbory
- formát príkazu : `lp [možnosti]` *`názov_súboru(ov)`*
- príklady:

```
$ lp ~/feathers
      request id is printerA (1 file(s))
$ lp -d printerB ~/feathers
      request id is printerB (1 file(s))
```

Zobrazenie stavu tlačiarne a rady

- príkaz `lpstat` zobrazí stav frontu na tlač
- formát príkazu: `lpstat [-podtsa]`
- príklad:

```
$ lpstat -o
      printerA-7   user1   1391   Aug 10 16:30   on printerA
      printerB-1   user2   2551   Aug 10 16:45   filtered
```

Odstránenie požiadavky na tlač

- Na zrušenie požiadavky na tlač poslanú príkazom `lp` slúži príkaz `cancel`. Tento príkaz vyžaduje ako parameter *`request-ID`*. Ten zistíte príkazom `lpstat`.
- formát príkazu: `cancel` *`Request-ID...`*
- príklady:

```
$ cancel printerB-6
      printerB-6: cancelled
$ cancel -u user2
      printerB-5: cancelled
```

Úprava a tlač súboru

- na upravenie a tlač obsahu súboru podľa rozličných možností úpravy slúži príkaz `pr`
- príkaz `pr` automaticky zobrazuje obsah súboru na obrazovku
- formát príkazu: `pr [možnosti]` *`názov_súboru(ov)`*

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- zistiť typ súboru príkazom `file`
- zobraziť obsah textového súboru príkazmi `cat`, `more`, `pg`, `head`, a `tail`
- určiť počty znakov, slov a riadkov príkazom `wc`
- vytvoriť prázdne súbory alebo vynoviť časy zmien existujúcich súborov príkazom `touch`
- použiť príkaz `tee` na vytvorenie textu v súbore
- vytvoriť a odstrániť adresáre pomocou príkazov `mkdir` a `rmdir`
- ovládať súbory a adresáre pomocou príkazov `mv`, `cp` a `rm`
- používať príkazy na tlač súborov, skontrolovať stav tlačového frontu a zrušiť požiadavku na tlač
- upraviť a tlačiť obsah súborov pomocou príkazu `pr`

5 Bezpečnosť súborov

Ciele

Po dokončení tohto modulu by ste mali vedieť:

- zobraziť práva súborov a adresárov
- vymenovať štandardné typy práv (read/write/execute)
- použiť príkaz `chmod` na zmenu práv v symbolickom a oktálovom móde
- určiť prednastavené práva dané novovytvoreným súborom a adresárom pomocou príkazu `umask`

Bezpečnosť

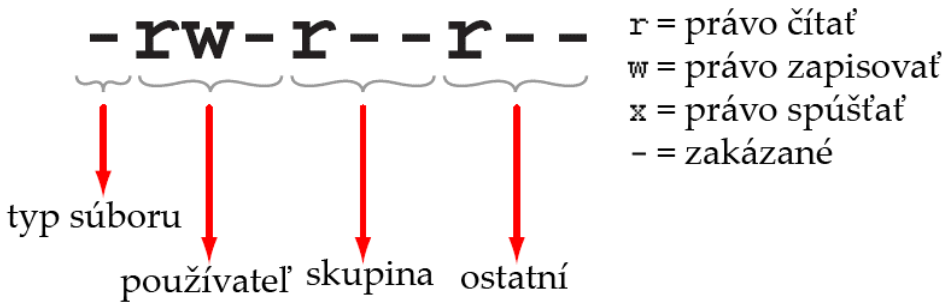
- operačný systém Solaris poskytuje dva základné spôsoby zabránenia nepovoleného prístupu do systému a ochrany údajov
  - autentifikáciou používateľovho prihlásenia do systému overením existencie prihlasovacieho mena a hesla v `/etc/passwd` a `/etc/shadow`
  - automatickou ochranou prístupu k súborom a adresárom nastavením štandardných prístupových práv súborov a adresárov už pri ich vytvorení

Prezeranie prístupových práv súborov a adresárov

Prístupové práva súborov a adresárov zobrazíme pomocou príkazu `ls -l`

`r` = právo čítať
`w` = právo zapisovať
`x` = právo spúšťať (vykonávať)
`-` = zakázané (právo je odobrané)

```
$ ls -l .profile
-rw-r--r-- 1 angel staff 560 Jun 11 11:23 .profile
```



Obr. 7 Práva súborov

Kategórie prístupových práv

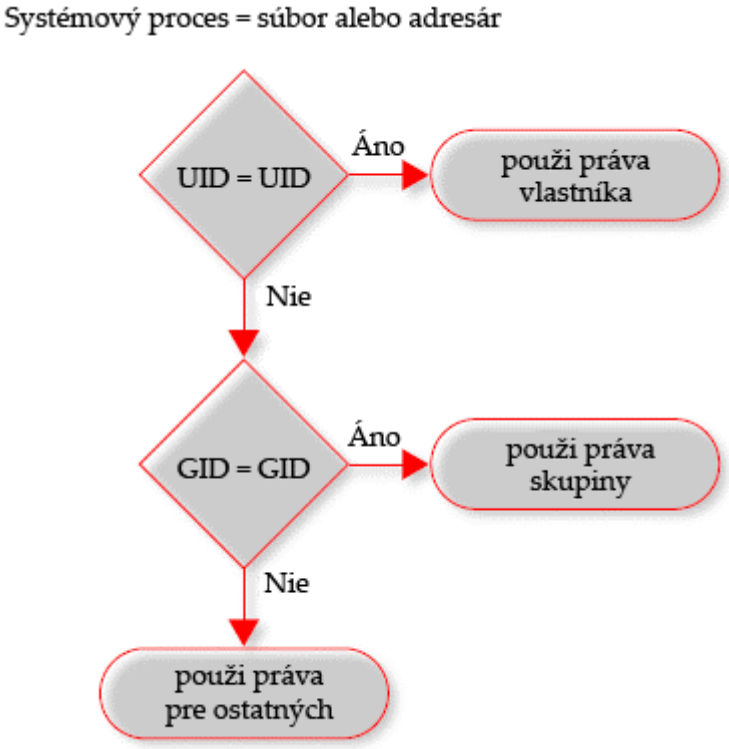
- práva používateľa (vlastníka) súboru alebo adresára
- práva skupiny používateľov, ktorá vlastní súbor alebo adresár
- práva ostatných, ktorí nie sú ani vlastníkom ani skupinou vlastníkov súboru alebo adresára

Určovanie prístupu k súborom a adresárom

- prístup k súboru alebo adresáru sa určuje na základe používateľovho identifikačného čísla UID a identifikačným číslom skupiny GID
  - UID – identifikuje používateľa, ktorý vytvoril adresár alebo súbor a zároveň určuje jeho vlastníctvo
  - GID – identifikuje skupinu používateľov, ktorí vlastnia adresár alebo súbor. Súbor alebo adresár môže súčasne patriť len jednej skupine.
- UID a GID čísla zistíte príkazom `ls -n`

```
$ ls -n .profile
-rw-r--r--   1 105      10           256 Aug  3 18:36 .profile
```

Postup určenia prístupových práv



Obr. 7 Postup povolenia prístupu k súboru na základe práv

Typy práv

- prístup k súborom a adresárom je chránený štandardnou množinou prednastavených práv, ktoré sú automaticky priradené operačným systémom Solaris, v okamihu vytvorenia súboru alebo adresára
- práva `rwx` kontrolujú, kto a čo sa môže robiť so súborom alebo adresárom.

Typy práv

Právo	znak	súbor	adresár
Čítať	`r`	súbor môže byť videný alebo kopírovaný	obsah môže byť zobrazený pomocou príkazu `ls`
Zapisovať	`w`	obsah súboru môže byť menený	keď má používateľ právo vykonať, tak súbory môžu byť pridávané alebo vymazávané

Vykonať	`x`	súbor môže byť vykonaný, spustený (iba skripty shellu alebo spúšťateľné súbory).	používateľ môže použiť príkaz `cd` na prístup do adresára a ak má zároveň právo na čítanie, môže použiť príkaz `ls`.

Zmena práv

- na novovytvorených súboroch alebo adresároch môžete zmeniť práva príkazom `chmod`
- tento príkaz môže použiť len vlastník súboru alebo adresára a superuser
- príkaz `chmod` môže meniť práva zapísané buď v *znakovom* alebo *oktálovom* móde.
  - znakový mód používa kombináciu písmen a znakov na pridanie, odobranie alebo nastavenie práv pre každú skupinu používateľov
  - oktálový mód používa čísla reprezentujúce jednotlivé práva; nazýva sa aj *absolútny* mód

Práva podľa oktálových hodnôt

Jednotlivé povolenia môžeme zapisovať aj ako čísla v osmičkovej sústave, tzv. oktálové hodnoty:

4 čítať
2 zapisovať
1 vykonávať

Hodnoty oktálových povolení sú

7 `r w x`
6 `r w -`
5 `r - x`
4 `r - -`
3 `- w x`
2 `- w -`
1 `- - x`
0 `- - -`

Kombinované práva v oktálovom móde

644 `rw-r--r--`(všetci môžu čítať a vlastník aj meniť – obyčajne platí pre súbor)
751 `rwxr-x--x`
775 `rwxrwxr-x` (vlastník a skupina môžu všetko, ostatní iba čítať a vykonávať)
777 `rwxrwxrwx`

Porovnanie znakového a absolútneho (oktálového) módu

- niektoré znakové výrazy nemajú porovnateľný výraz v absolútnom móde
  - napríklad. `chmod u+x,g+w` *`názov_súboru`* sa nedá napísať v absolútnom móde
- výrazy v absolútnom móde sú niekedy stručnejšie ako výrazy v symbolickom móde
- výrazy v absolútnom móde sú niekedy vhodnejšie na použitie v skriptoch shellu, do ktorých ide vstup v číselnej forme z iných programov

Prednastavené práva a filter `umask`

- filter `umask` dohliada na prednastavené práva pridelené novovytvoreným súborom a adresárom
- filter `umask` je 3-číselná oktálna hodnota, ktorá sa vzťahuje na práva čítanie/zápis/vykonávanie pre vlastníka, skupinu a ostatných
- počiatočná hodnota práv daná systémom pri vytvorení súboru je 666 (`rw-rw-rw-`).
- počiatočná hodnota práv daná systémom pri vytvorení adresára je 777 (`rwxrwxrwx`).

Výpočet ako sa maska tvorenia súborov prejaví

- napíšte prednastavené povolenia v rozšírenej forme
- napíšte masku tvoreného súboru pod prednastavené povolenia
- odpočítajte bity a napíšte výsledok

Dôležité upozornenie o vypočítavaní filtra umask

Prednastavené práva súboru	`r w - r w - r w -`	666
`umask` 123	`- - x - w - - w x`	123
výsledné práva súboru	`r w - r - - r - -`	644 **(nie 543!)**
prednastavené práva adresára	`r w x r w x r w x`	777
`umask` 123	`- - x - w - - w x`	123
výsledné práva adresára	`r w - r - x r - -`	654

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- zobraziť povolenia súborov a adresárov
- definovať štandardné typy povolení (čítanie/zápis/vykonávanie)
- použiť príkaz `chmod` na zmenu povolenia v znakovom alebo oktálovom móde
- zistiť prednastavené povolenia priradené novovytvoreným súborom a adresárom príkazom `umask`

6 Hľadanie súborov a textu

Ciele

Po dokončení tohto modulu by ste mali vedieť:

- použiť príkaz `find` na nájdenie súborov v strome adresárov operačného systému Solaris podľa zadaných kritérií hľadania
- použiť príkazy `cmp` a `diff` na porovnanie rozdielov v obsahu súborov
- triediť obsah textových súborov v abecednom a číselnom poradí pomocou príkazu `sort`
- hľadať regulárne výrazy v obsahu jedného alebo viacerých súborov pomocou príkazov `grep`, `egrep` a `fgrep`

Hľadanie súborov pomocou príkazu `find`

- na nájdenie súboru v adresárovom strome použite príkaz `find`
  - príkaz rekurzívne postupuje cestou v adresárovom strome podľa názvu cesty hľadajúc súbory, ktoré vyhovujú kritériám
  - keď príkaz `find` nájde súbory, ktoré vyhovujú kritériám, tak na monitore zobrazí cestu ku každému súboru
- formát príkazu: `find` *`názov_cesty(ciest) výraz(y) akcia(e)`*

Argumenty používané s príkazom `find`

Argument	Definícia
`názov_cesty`	Adresár, z ktorého sa začne hľadanie.
`výraz`	Kritériá hľadania spresnené jednou alebo viacerými možnosťami. Spresnenie viacerými možnosťami bude príkaz `find` brať ako požiadavku AND, takže všetky zadané výrazy musia byť vyhodnotené ako pravdivé.

Výrazy používané s príkazom find

Výraz	Definícia
`-name` *`meno_súboru`*	Nájde súbory podľa zadaného *`mena_súboru`*. Ak uzatvoríme *`meno_súboru`* do úvodzoviek, môžeme v ňom použiť aj metaznaky.
`-size [+|-]`*`n`*	Nájde súbory, ktoré sú väčšie ako +*n*, menšie ako −*n*, alebo presne *n*. Číslo *n* predstavuje 512 bytový blok.
`-atime [+|-]`*`n`*	Nájde súbory, na ktoré bolo pristupované viac ako +*n* dní, menej ako −*n* dní alebo presne *n* dní.
`-mtime [+|-]`*`n`*	Nájde súbory, ktoré boli pozmenené viac ako +*n* dní, menej ako −*n* dní alebo presne *n* dní.
`-user` *`loginID`*	Nájde všetky súbory patriace používateľovi s *`loginID`*.
`-type`	Nájde typ súboru, napríklad `f` (file, súbor) alebo `d` (directory, adresár).
`-perm`	Nájde súbory, ktoré majú určité prístupové práva.

Akcie používané s príkazom `find`

Akcia	Definícia
`-exec` *`príkaz`* `{} \;`	Vykoná zadaný *`príkaz`* na každom súbore, ktorý nájde. Zátvorky {} sa nahradia menami nájdených súborov. Medzera, opačná lomka a bodkočiarka (`\;`) určujú koniec príkazu. Pred opačnou lomkou musí byť medzera!
`-ok` *`príkaz`* `{} \;`	Určuje interaktívnu formu možnosti `-exec`. Požaduje nejaký vstup predtým ako príkaz `find` aplikuje *`príkaz`* na súbor, inak sa správa rovnako ako `-exec`.
`-print`	Príkaz `find` zobrazí názov aktuálnej cesty na obrazovku. Táto možnosť je prednastavená.
`-ls`	Zobrazí názov aktuálnej cesty spolu s príslušnými štatistikami (inode číslo, veľkosť v kB, práva, počet pevných odkazov, meno používateľa).

Príklady použitia príkazu `find`:

- $ **`find ~ -name dante`** (vyhľadať súbor `dante` a začať hľadať od domovského adresára)
- $ **`find ~ -name core –exec rm {} \;`** (vyhľadať a vymazať súbor `core` a začať hľadať od domovského adresára)
- znak ~ je označenie pre domovský adresár práve prihláseného používateľa
- za znakom ~ môže nasledovať meno používateľa, ktorého domovský adresár sa má použiť (`~angel`)

Hľadanie rozdielov pomocou príkazu `cmp`

- príkaz `cmp` slúži na porovnanie súborov
- príkaz pracuje s binárnymi aj ASCII súbormi
- formát príkazu: `cmp` *`názov_súboru1 názov_súboru2`*

Hľadanie rozdielov pomocou príkazu `diff`

- príkaz `diff` sa takisto používa na hľadanie rozdielov medzi súbormi
  - príkaz `diff` nezobrazí výstup pre netextové súbory
- zistené rozdiely sa zobrazujú na obrazovke riadok po riadku
- formát príkazu :

  `diff` *`–možnosť názov_súboru1 názov_súboru2`*

Triedenie dát pomocou príkazu `sort`

- príkaz `sort` triedi textové riadky v jednom alebo viacerých súboroch a výsledok zobrazuje na obrazovke
- je to rýchly a jednoduchý spôsob organizovania údajov podľa číselného alebo abecedného poradia
- formát príkazu: `sort (-|+)`*`možnosti názov súboru(ov)`*
- na alfanumerické zoradenie výstupu slúži možnosť `sort –d`, údaje zoraďuje ako v slovníku
- na numerické zoradenie výstupu slúži možnosť `sort –n`
- na reverzné zoradenie výstupu slúži možnosť `sort -r`

Hľadanie textu v súboroch pomocou príkazu `grep`

- príkaz `grep` hľadá regulárny výraz alebo znakový vzor v jednom alebo viacerých súboroch
- ak niečo nájde, tak zobrazí každý riadok obsahujúci hľadaný výraz na obrazovku
- príkaz nemení obsah súboru
- formát príkazu: `grep` *`možnosť(i) vzor meno_súboru(ov)`*

Metaznaky regulárnych výrazov

znak	význam	príklad	čo urobí
`^`	začiatok riadku	`'^vzor'`	nájde všetky riadky začínajúce slovom „vzor“
`$`	koniec riadku	`'vzor$'`	nájde všetky riadky končiace slovom „vzor“
`.`	ľubovoľný jeden znak	`'v.....r'`	nájde riadky obsahujúce slovo začínajúce na „v“ nasledované piatimi ľubovoľnými znakmi a končiace „r“
`[ ]`	množina znakov	`'[Vv]zor'`	nájde riadky obsahujúce slová „Vzor“ alebo „vzor“
`[^]`	množina znakov, ktorá sa nesmie vo vzore nachádzať	`'[^a-m]zor'`	nájde riadky, ktoré neobsahujú slová „azor“, „bzor“ až „mzor“
`*`	predchádzajúci znak sa musí vyskytovať nula až viackrát	`'[a-z][a-z]*'`	nájde riadky, ktoré obsahujú malé písmená

Príklady použitia príkazu `grep`:

- $ **`grep –n root/etc/group`** (vyhľadať riadky v súbore `/etc/group`, ktoré obsahujú text root a vypísať ich čísla)
- $ **`grep –v root/etc/group`** (vyhľadať všetky ostatné riadky)
- $ **`grep '^no' /etc/passwd`** (vyhľadať riadky v súbore `/etc/passwd`, ktoré začínajú na no)
- $ **`grep 'adm$' /etc/passwd`** (vyhľadať riadky v súbore `/etc/passwd`, ktoré končia na adm)

Hľadanie textu v súboroch pomocou príkazu `egrep`

- príkaz `egrep` hľadá v obsahu jedného alebo viacerých súborov vzor pomocou rozšírených metaznakov regulárneho výrazu
- príkaz `egrep` používa tie isté možnosti ako príkaz `grep`
- formát príkazu :

```
egrep [-možnosti] vzor názov_súboru ...
```

Rozšírené metaznaky regulárnych výrazov

znak	účel	príklad	čo robí
+	predchádzajúci znak sa musí vyskytovať aspoň raz	'[a-z]+ark'	nájde riadky, ktoré obsahujú slová, v ktorých je jeden alebo viacero malých písmen nasledovaných slovom „ark“ (napr.: „airpark“, „bark“, „dark“, „landmark“, „shark“, „sparkle“, „trademark“)
x|y	buď *x* alebo *y*	'jablko|pomaranc'	
(|)	skupinové znaky	'(1|2)+' 'search(es|ing)+'	Nájde riadky obsahujúce aspoň jeden výskyt zadaných vzorov (napríklad „1“ alebo „2“, „searches“, alebo „searching“)

Hľadanie textu v súboroch pomocou príkazu `fgrep`

- príkaz `fgrep` hľadá v súbore vzor vyjadrený ako pevný reťazec
- od `grep` a `egrep` sa odlišuje pretože považuje všetky znaky za také aké sú a nespracováva metaznaky regulárnych výrazov v príkazovom riadku.
- `fgrep` sa používa na vyhľadanie slova, ktorý obsahuje znaky, ktoré by `grep` považoval za metaznaky.
- formát príkazu :

```
fgrep možnosť(i) vzor názov_súboru(ov)
```

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- použiť príkaz `find` na nájdenie súborov v adresárovom strome operačného systému Solaris podľa zadaných kritérií
- použiť príkazy `cmp` a `diff` na zistenie rozdielov v súboroch
- roztriediť obsah súborov v abecednom a číselnom poradí pomocou príkazu `sort`
- hľadať regulárne výrazy v obsahu jedného alebo viacerých súborov pomocou `grep`, `egrep`, a `fgrep`

7 Visual Editor (vi)

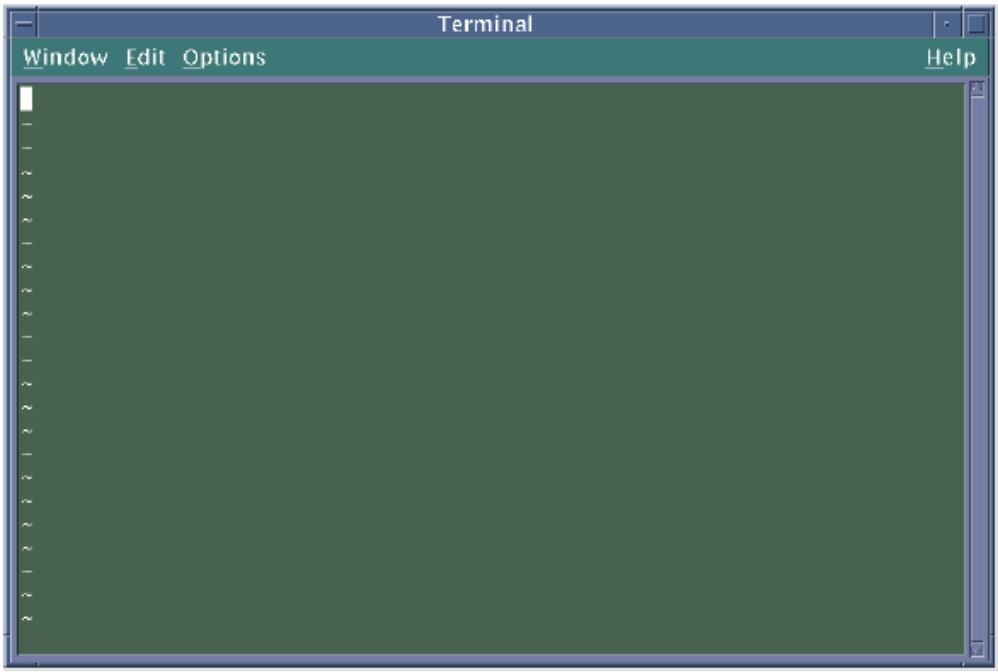
Ciele

Po dokončení tohto modulu by ste mali vedieť:

- definovať tri spôsoby práce editora vi
- spustiť editor vi
- nastaviť a pohybovať kurzorom v editore vi
- prepínať medzi módmi editoru vi
- vytvoriť a zmazať text
- kopírovať alebo premiestňovať text
- nastaviť možnosti editora vi
- použiť funkcie hľadania a nahrádzania v editore vi
- opustiť editor vi

Úvod do editora vi

- editor vi (visual display) je interaktívny editor na tvorbu alebo zmenu textových súborov
- všetky úpravy s vi sa uchovávaúj vo dočasnom súbore (vyrovnávacia pamäť). Zmeny môžu byť buď zapísané na disk alebo zrušené.
- editor vi používa presne určené množstvo systémových prostriedkov
- editor vi nezávisí od systému okien
- editor vi je prítomný vo všetkých systémoch Unix, kým niektoré iné textové editory nie sú



Obr. 8 Úvodná obrazovka editora vi

Režimy vi

- editor vi je editor štýlu príkazového riadku, ktorý má tri základné režimy práce
  - príkazový režim
  - editovací režim
  - režim spodného riadku
- aktuálny režim sa zobrazuje po zapnutí možnosti `:set showmode`

Príkazový režim

- je prednastavený režim pre editor vi
- môžete v ňom vkladať príkazy na vymazanie, zmenu, kopírovanie a prenášanie textu, nastaviť kurzor, hľadať text alebo opustiť editor vi
- príkazy sa zadávajú stlačením klávesy na klávesnici, niektoré príkazy sa zadávajú kombináciou viacerých kláves
- príkazy rozlišujú medzi veľkými a malými písmenami

Editovací režim

- v tomto móde môžte do súboru vkladať text.
- aby sa editor vi prepol do editovacieho režimu zadajte jeden z nasledujúcich príkazov
  - i (I) – vloží text na pozíciu kurzora (na začiatok riadku)
  - o (O) – vloží prázdny riadok pod (nad) pozíciu kurzora
  - a (A) – vloží text za pozíciu kurzora (na koniec riadku)
  - `:r súbor` – vloží súbor za pozíciu kurzora ale zostane v príkazovom režime
- do príkazového režimu sa vrátite stlačením klávesy ESC

Režim spodného riadku

- pokiaľ ste v príkazovom móde, tak po napísaní dvojbodky (:) môžete použiť rozšírené editovacie príkazy
- dvojbodka vás umiestni na spodok obrazovky

Spustenie editoru vi

- na vytvorenie nového súboru spustite vi editor s *názvom_súboru*.
- formát príkazu
  vi *možnosť(i) názov_súboru*
  view *názov_súboru* (otvorí súbor len na čítanie)

Ukončenie editora vi

- editor vi je možné ukončiť viacerými spôsobmi
  - v príkazovom režime, stlačením klávesov „ZZ“
  - v režime riadku napísaním príkazu `:wq` (ulož zmeny a ukonči)
  - v režime riadku napísaním príkazu `:q` (ukonči)
  - v režime riadku napísaním príkazu `:q!` (ukonči ale neukladaj zmeny)

Pohyb kurzora

- editor vi používa na základný pohyb kurzora klávesy „h“ „j“ „k“ „l“
  - k – hore
  - j – dole
  - h – doľava

  - l – doprava

- ďalšie klávesy, resp. kombinácie kláves na pohyb v rámci dokumentu sú
  - 0 – začiatok riadku
  - $ – koniec riadku
  - 1G – začiatok dokumentu
  - G – koniec dokumentu

Mazanie textu

- editor vi pozná viaceré spôsoby, ako zmazávať text, pričom posledný zmazaný prvok (znak, slovo alebo riadok) uloží do vyrovnávacej pamäte
  - ctrl-h, alebo backspace – zmaže znak naľavo od kurzora (v editovacom režime)
  - x – zmaže znak nad kurzorom (v príkazovom režime)
  - dw – zmaže slovo nad kurzorom (v príkazovom režime)
  - dd – zmaže riadok, v ktorom je kurzor (v príkazovom režime)
  - D – zmaže text od pozície kurzora po koniec riadku (v príkazovom režime)
  - dG – zmaže text od pozície kurzora po koniec dokumentu (v príkazovom režime)
  - 10dd – zmaže riadok, na ktorom sa nachádza kurzor a 10 ďalších riadkov

Kopírovanie textu

- editor vi nikde nesignalizuje, že riadok skopíroval do vyrovnávacej pamäte
  - yy – skopíruje aktuálny riadok
  - yw – skopíruje slovo na ktorom je kurzor
  - 10y – skopíruje riadok na ktorom je kurzor a 10 ďalších riadkov
  - p – vloží najvrchnejší prvok z vyrovnávacej pamäte (znak, slovo, riadok alebo viac riadkov) do textu
  - P – vloží najvrchnejší prvok z vyrovnávacej pamäte pred pozíciu kurzora

Vyhľadávanie a nahrádzanie textu

- pre vyhľadávanie stlačte lomku (/) a za ňu napíšte hľadaný reťazec
  - n – hľadá ďalší výskyt slova smerom ku koncu súboru
  - N alebo ? – hľadá ďalší výskyt slova smerom k začiatku súboru
- na nahrádzanie textu slúži príkaz s v príkazovom riadku
  - :%s/starý/nový/g – nahradí starý novým v celom dokumente
  - :.,$s/starý/nový/g – nahradí starý novým od pozície kurzora po koniec dokumentu
  - :^,.s/starý/nový/g – nahradí starý novým od začiatku dokumentu po pozíciu kurzora

Prispôsobenie relácie editora vi

- editor vi obsahuje možnosti prispôsobenia editovacieho režimu, napríklad:
  - zobrazenie čísla riadku (:set nu)
  - zobrazenie neviditeľných znakov ako napríklad tabulátor a znaky konca riadku (:set list)
- na ovládanie týchto možností použite v príkazovom režime príkaz set.
- ak chcete danú možnosť vypnúť pred jej názov dajte slovo no (:set nolist)

- ak si chcete zobraziť aktuálne nastavené možnosti použite príkaz `set` bez parametrov
- ak si chcete zobraziť všetky dostupné možnosti použite príkaz `:set all`

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- vymenovať tri režimy editora `vi`
- spustiť editor `vi`
- polohovať kurzor na žiadané miesto a pohybovať ním
- prepínať medzi módmi `vi`
- tvoriť a mazať text
- kopírovať alebo prenášať text
- nastaviť možnosti vo `vi`
- funkcie editora `vi` hľadať a nahradiť
- opustiť editor `vi`

8 Archivácia používateľských údajov

Ciele

Po ukončení tohto modulu by ste mali vedieť:

- určiť, ktoré príkazy sú vhodné pre uchovávanie, prezeranie alebo vyberanie rozličných typov súborov
- predviesť, ako zmenšiť veľkosť súborov a adresárov a uchovať ich na pásku pomocou príkazu `tar`
- opísať kroky potrebné pre dekomprimáciu alebo prezeranie komprimovaného súboru pomocou príkazov `uncompress` a `zcat`
- použiť príkazy `gzip` a `gunzip` na komprimáciu alebo dekomprimáciu súborov
- použiť príkaz `zip` na zbalenie a komprimovanie viacerých súborov a použiť príkaz `unzip` na dekomprimovanie súboru zip.
- komprimovať a kopírovať viaceré súbory do jedného archívneho súboru pomocou príkazu `jar`
- kopírovať súbory do a vyberať súbory z archívneho súboru alebo pásky pomocou príkazu `cpio`
- pochopiť ako používať správu dátovej úschovy na prístup k CD-ROM a disketám
- použiť príkaz `eject` na vybranie CD-ROMov a diskiet zo zariadení

Prehľad príkazov archivácie

Príkazy na uloženie, nájdenie a vybratie súborov na páskovom zariadení alebo archívnom súbore sú:

- `tar` – vytvorí a extrahuje súbory z páskového zariadenia alebo archívneho súboru
- `compress` alebo `uncompress` – komprimuje alebo dekomprimuje súbor
- `zcat` – dekomprimuje komprimovaný súbor a výstup pošle na obrazovku bez zmenenia komprimovaného súboru
- `gzip` alebo `gunzip` – komprimuje alebo dekomprimuje súbor
- `gzcat` – dekomprimuje súbor typu gzip a výstup pošle na obrazovku bez zmenenia komprimovaného súboru
- `zip` a `unzip` – balí a komprimuje súbory a dekomprimuje súbory
- `jar` – balí a komprimuje viaceré súbory do jedného archívneho súboru
- `cpio` – kopíruje a vyberá súbory z archívneho súboru alebo páskového zariadenia
- po komprimácií súboru je pôvodný súbor zmazaný
- pri dekomprimácii súboru je pôvodný komprimovaný súbor zmazaný

Archivácia súborov pomocou príkazu `tar`

- príkaz `tar` archivuje a vyberá súbory do a z jedného súboru nazývaného *súbor tar*. *Súbor tar* obvykle reprezentuje magnetickú pásku, no pri zadaní prepínača `f` môžeme manipulovať s ľubovoľným súborom.
- formát príkazu:

```
tar funkcia(e) názov_súboru(ov)
```

Funkcie príkazu tar

c	vytvorí nový súbor tar. Zápis nezačína na konci, ale na začiatku súboru
t	vypíše tabuľku obsahu súboru tar
x	vyberie určený súbor zo súboru tar
f *archívny_súbor*	Zadáva archívny súbor alebo páskové zariadenie. Prednastavené páskové zariadenie je /dev/rmt/0. Ak je archívny súbor "-","potom príkaz tar podľa potreby číta zo štandardného vstupu alebo píše do štandardného výstupu.
v	vykoná v oznamovacom (verbose) móde
r	vymení. Vymenované súbory sú zapísané na koniec páskového archívu
u	obnoví, ak zadané súbory v archíve neexistujú alebo ak boli medzičasom obnovené, potom sú zapísané na koniec archívu.
e	ak sa vyskytne neočakávaná chyba, okamžite skončí
h	nasleduje symbolické odkazy tak, ako keby to boli normálne súbory alebo adresáre
m	keď sa extrahuje z archívu, tak čas modifikácie súboru je časom extrahovania
o	keď sa vyberá zo súboru, tak sa vlastníkom súboru stáva momentálny používateľ
w	určuje výzvu na potvrdenie každého súboru, ktorý má byť vybraný z alebo daný do archívu

Príklady:

- vytvorenie tar archívu

```
$ tar cvf angel.tar angel_files
      a angel_files/ 0K
      a angel_files/súbor 1K
      a angel_files/súbor2 1K
      a angel_files/súbor3 1K
      a angel_files/súbor4 1K
```

- extrahovanie jedného súboru z tar archívu

```
$ tar xvf angel.tar angel_files/súbor2
      tar: blocksize = 11
      x angel_files/súbor2, 35 bytes, 1 tape blocks
```

- extrahovanie celého tar archívu

```
$ tar xvf angel.tar
      tar: blocksize = 11
      x angel_files, 0 bytes, 0 tape blocks
      x angel_files/súbor, 31 bytes, 1 tape blocks
      x angel_files/súbor2, 35 bytes, 1 tape blocks
      x angel_files/súbor3, 21 bytes, 1 tape blocks
      x angel_files/súbor4, 84 bytes, 1 tape blocks
```

Komprimácia súborov pomocou príkazu `compress`

- príkaz compress slúži na zmenšenie veľkosti súboru
- keď je súbor skomprimovaný, tak pôvodný súbor je nahradený novým súborom s príponou .Z
- formát príkazu: compress [-v] *názov_súboru*...

Dekomprimácia súborov pomocou príkazu `uncompress`

- príkaz `uncompress` vracia komprimovaný súbor do pôvodného stavu
- formát príkazu: `uncompress [-vc] *názov_súboru...*`

Pozeranie súborov pomocou príkazu `zcat`

- príkaz `zcat` umožňuje vidieť súbor, ktorý bol komprimovaný príkazom `compress`
- príkaz `zcat` analyzuje komprimované dáta a zobrazuje obsah súboru ako keby komprimovaný nebol, pôvodný komprimovaný súbor zostáva nezmenený
- formát príkazu: `zcat *názov_súboru ...*`

Komprimovanie súboru pomocou príkazu `gzip`

- keď `gzip` komprimuje súbor tak mu pridá príponu `.gz`
- formát príkazu: `gzip *názov_súboru názov_súboru názov_súboru*`
- príklad:

```
$ gzip súbor1 súbor2 súbor3 súbor4
$ ls -1
      súbor1.gz
      súbor2.gz
      súbor3.gz
      súbor4.gz
```

Obnova súboru gzip pomocou príkazu `gunzip`

- na obnovu súboru komprimovaného príkazom `gzip` slúži príkaz `gunzip`
- formát príkazu: `gunzip *názov_súboru*`
- príklad:

```
$ gunzip súbor1.gz
```

Komprimovanie viacerých súborov pomocou príkazu `zip`

- keď príkaz `zip` komprimuje súbor, tak mu pridá príponu `.zip`
- formát príkazu: `zip *cieľový_súbor zdrojový_súbor(y)*`
- príklad:

```
$ zip súbor.zip súbor1 súbor2 súbor3
$ ls
      súbor.zip
      súbor1
      súbor2
      súbor3
```

Komprimácia súborov pomocou príkazu `jar`

- príkaz `jar` je podobný príkazu `tar`, ale súbory komprimuje naraz
- `jar` archívy sú kompatibilné s `zip` archívmi
- tento príkaz kopíruje a komprimuje viaceré súbory do jedného archívneho súboru
- formát príkazu:
  `jar *možnosti výstupný_súbor meno_súboru(ov)/adresára(ov)*`

Používanie príkazu `cpio`

- príkaz `cpio` (copy in/copy out) archivuje alebo extrahuje súbor na pásku alebo do súboru
- niektoré výhody príkazu `cpio` sú:

  - na pásku balí súbory efektívnejšie ako `tar`
  - pri obnove súborov preskočí chybné miesta na páske
  - obsahuje možnosti, ktoré menia formát výstupu na zvýšenie prenositeľnosti medzi rozličnými páskovými systémami
  - tvorí viacnásobné dátové jednotky na páske
  - archivuje súbory bez zmeny času prístupu
- formát príkazu: `cpio` *`možnosti názov_súboru(ov)`*
- základné možnosti
  - `-i` extrahuje súbory zo štandardného vstupu
  - `-o` zo štandardného vstupu načíta názvy súborov, ktoré potom spolu s informáciou `cpio` archívu skopíruje na štandardný výstup
  - `-p` zo štandardného vstupu načíta názvy súborov a skopíruje ich do cieľového adresára
- príklady
  vytvorenie archívu s použitím štandardného vstupu
  `$ ls | cpio -oc > ../súbor`

  extrahovanie súborov do adresárov
  `$ cat súbor | cpio -icd`

  kopírovanie súborov do iného adresára
  `$ find . -depth -print | cpio -pdlmv adresár`

Detekcia prenosných dátových zariadení

- správa zväzkov poskytuje automatickú detekciu CD-ROM
- čo sa diskiet týka, tak správa zväzkov musí byť informovaná zakaždým, keď sa do mechaniky vloží disketa
- na informovanie správy zväzkov o vložení diskety slúži príkaz `volcheck`
- formát príkazu: `volcheck [-v]` *`zariadenie_prístupová cesta`*
- prenosné médium nie je možné vysunúť pokiaľ sa používa, teda jeho adresár sa zhoduje s momentálnym pracovným adresárom
- prenosné médium sa vysunie príkazov `eject`

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- určiť, ktoré príkazy sú vhodné na ukladanie, prezerania alebo vyberanie rozličných druhov súborov
- zmenšiť veľkosť súborov a adresárov a uložiť ich na pásku pomocou príkazov `compress` a `tar`
- opísať kroky na dekomprimovanie alebo prezeranie komprimovaného súboru pomocou príkazov `uncompress` a `zcat`
- použiť príkazy `gzip` a `gunzip` na komprimovanie a dekomprimovanie súborov
- použiť príkaz `zip` na zbalenie a komprimovanie viacerých súborov a použiť príkaz `unzip` na dekomprimáciu archívneho súboru typu zip.
- komprimovať a kopírovať viaceré súbory do jedného archívneho súboru v jednom kroku pomocou príkazu `jar`

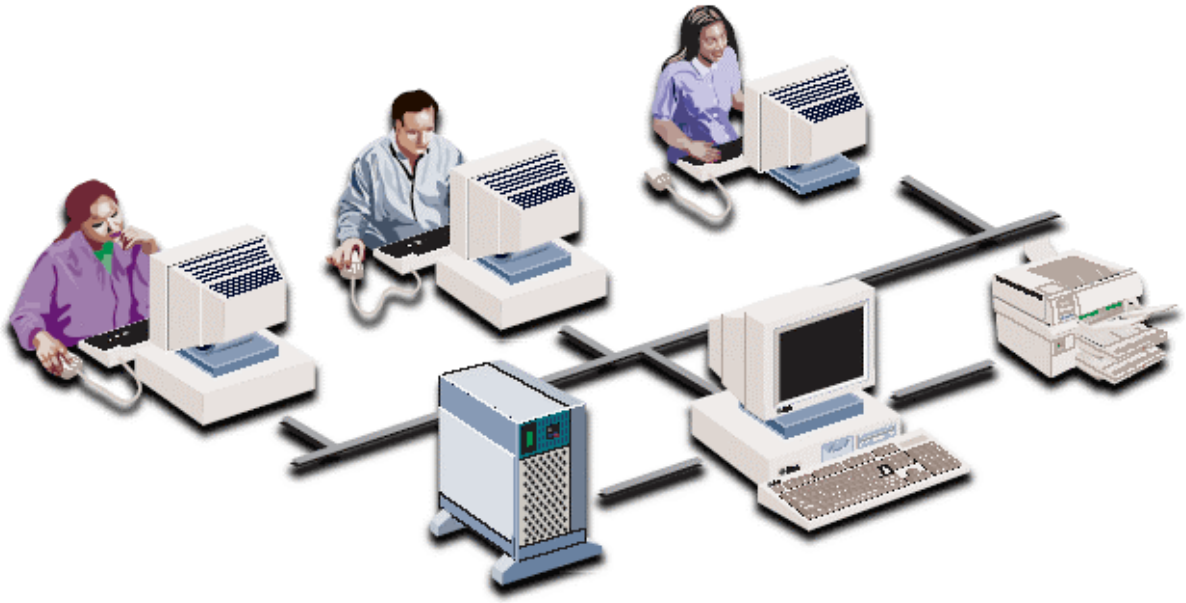
- kopírovať a vyberať súbory z archívneho súboru alebo páskového zariadenia pomocou príkazu `cpio`
- porozumieť ako používať riadenie dát k prístupu na CD-ROM a diskety
- použiť príkaz `eject` na vybratie CD-ROM alebo diskiet z mechaník

9 Vzdialené pripojenia

Ciele

Po dokončení tohto modulu by ste mali vedieť:

- otvoriť session na vzdialenom systéme pomocou príkazu `telnet, ssh`
- vzdialene sa prihlásiť na iný systém na sieti
- použiť príkaz `ftp` na prebratie súboru zo vzdialeného systému



Obr. 9 Príklad sieťového pripojenia

Sieť

- *sieť* je prepojenie, ktoré umožňuje výmenu informácií medzi systémami. Dva typy siete sú:
  - lokálna počítačová sieť (LAN) – sieť ktorá pokrýva malý priestor, obyčajne menej ako niekoľko tisíc metrov
  - rozľahlá počítačová sieť široká sieť (WAN) – sieť cez tisíce kilometrov

Host

- *host* je počítačový systém na sieti
  - *local host* je systém, na ktorom momentálne pracujete
  - *remote host* je systém, na ktorý sa pristupuje z iného systému. Z pohľadu používateľa toho systému je remote host jeho *local hostom.*

Bezpečnosť

- príkazy preberané v tomto module majú zásadný bezpečnostný problém, pretože posielajú heslá v nezašifrovanom tvare, a preto je ich použitie doporučované iba na lokálnej sieti bezpečných počítačov
- z hľadiska bezpečnosti je vhodnejšie používať iné programy, ktoré prenášajú dáta po sieti šifrovane. Jedná sa o programy `ssh(rsh)`, `scp(rcp)`, `slogin(rlogin)`, `sftp(ftp)`, kde v zátvorkách uvádzame nešifrované aplikácie vysvetľované nižšie.

Používanie príkazu `telnet`, `ssh`

- príkaz `telnet` je štandardnou aplikáciou, ktorú poskytuje skoro každá implemetácia TCP/IP
  - umožňuje prihlásiť sa na vzdialený systém a pracovať v jeho prostredí.
  - príkaz `telnet` pracuje aj medzi hostami s rozdielnymi operačnými systémami
  - `telnet` si pri nadväzovaní spojenia so serverom zisťuje procesom, ktorý sa nazýva negociácia, možnosti a funkcie, ktoré sú na každom so systémov podporované
- formát príkazu: `telnet` *`názov_hostu`*
- formát príkazu: `ssh` *`názov_hostu`*

Použitie príkazu `rlogin, slogin`

- príkaz `rlogin` slúži na vytvorenie relácie vzdialeného prihlásenia na druhú pracovnú stanicu
- príkaz `rlogin` bol vyvinutý na prácu medzi systémami Unix, ale je zapracovaný aj v iných operačných systémoch
- príkaz `rlogin` je podobný príkazu `telnet`, ale môže byť nastavený tak, aby na host prihlásil dôveryhodných používateľov aj bez hesla
  - `rlogin` môže byť preto bezpečnostným problémom a nemusí byť povolený na každom systéme
- formát príkazu: `rlogin` *`názov_hostu`*
- formát príkazu: `slogin` *`názov_hostu`*
  - príkaz `slogin` je totožný s príkazom `ssh`

Zadanie rozdielneho prihlasovacieho ID

- na zadanie rozdielneho prihlasovacieho ID (používateľského mena) pre vzdialenú reláciu slúži možnosť `-l`
- formát príkazu: `rlogin [-l` *`meno_používateľa`*`]` *`názov_hostu`*

Spustenie programu na vzdialenom systéme

- na spustenie programu na vzdialenom systéme slúži príkaz `rsh`
- formát príkazu:

```
rsh [-l meno používateľa] názov_hostu príkaz
rsh [-l meno používateľa] IP_adresa príkaz
```

Kopírovanie na a z druhého systému pomocou príkazov `rcp`, `scp`

- príkaz `rcp` umožňuje kopírovať súbory alebo adresáre na alebo z druhého systému
- formát príkazu:

```
rcp zdrojový_súbor názov_hostu:cieľový_súbor
rcp meno_hostu:zdrojový_súbor cieľový_súbor
```

- formát príkazu:
  `scp` *`zdrojový_súbor názov_hostu:cieľový_súbor`*
  `scp` *`meno_hostu:zdrojový_súbor cieľový_súbor`*

Používanie príkazu `ftp, sftp`

- príkaz `ftp` (File Transfer Protocol) je súčasť priemyselne štandardizovaného protokolu, a používa sa na prenos súborov medzi podobnými alebo rôznymi operačnými systémami
- formát príkazu: `ftp` *`názov_hostu`*
- formát príkazu: `sftp` *`názov_hostu`*
- pre prácu s ftp je nutné poznať niektoré príkazy
  - `ls` – zobrazí obsah aktuálneho adresára ftp servera
  - `get` *`názov_súboru`* – skopíruje súbor z ftp servera na klienta
  - `put` *`názov_súboru`* – skopíruje súbor z klienta na ftp server
  - `cd` *`názov_adresára`* – zmení adresár na ftp serveri
  - `lcd` *`názov_adresára`* – zmení adresár na klientovi
  - `help` – zobrazí zoznam príkazov ftp servera

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- otvoriť session na vzdialenom systéme pomocou príkazu `telnet` alebo `ssh`
- vzdialene sa prihlásiť na druhý systém na sieti
- použiť príkaz `ftp` a stiahnuť súbor zo vzdialeného systému

10 Systémové procesy

Ciele

Po dokončení tohto modulu by ste mali vedieť:

- opísať ako vznikajú procesy
- zobraziť aktívne procesy na systéme pomocou príkazu `ps`
- nájsť určený proces pomocou príkazu `pgrep`
- prebrať význam signálov na kontrolovanie aktivít procesov
- ukončiť proces pomocou príkazov `kill` a `pkill`
- používať príkazy na ovládanie jobov bežiacich v shelli

Prehľad procesov

- proces je časť pamäte a sústava dátových štruktúr vnútri kernelu

Kde vznikajú procesy?

- nové procesy sú tvorené ďalšími procesmi:
  - všetky rodičovské procesy majú odkazy na všetky svoje odvodené procesy
  - všetky odvodené procesy majú odkaz k svojmu rodičovskému procesu

UID a GID procesu

- aby kernel vedel, čo smie proces vykonať, musí si uložiť informáciu o tom, kto proces vlastní
- kernel kvôli tomuto ukladá ID číslo používateľov (UID) a identifikáciu skupín (GID)
- čísla procesov UID a GID sú rovnaké ako čísla UID a GID používateľa, ktorý ich spustil

Prezeranie procesov a PID

- príkaz `ps` (process status) vypíše procesy momentálne bežiace na systéme
- pre každý proces príkaz `ps` zobrazí ID procesu (PID), terminálový identifikátor (TTY), celkový čas spustenia (TIME) a taktiež zobrazí daný príkaz (CMD)
- formát príkazu: `ps` *`možnosti`*
- samotný príkaz `ps` vypíše iba procesy bežiace v aktuálnom termináli
- príklad:

```
$ ps -ef | more
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0   Aug 01 ?        0:01 sched
    root     1     0  0   Aug 01 ?        0:00 /etc/init -
    root     2     0  0   Aug 01 ?        0:00 pageout
    root     3     0  0   Aug 01 ?        1:22 fsflush
    root   270     1  0   Aug 01 ?        0:00 /usr/sbin/inetd -s
    angel 2410  1726  0 21:40:56 pts/2    0:00 more
--More--
```

- niektoré iné unixové distribúcie používajú namiesto príkazu `ps -ef` príkaz `ps aux`.

Príkaz `pgrep`

- príkaz `pgrep` poskytuje efektívny spôsob nájdenia procesu podľa jeho názvu
- príkaz `pgrep` je prednastavený tak, aby na príkazovom riadku zobrazil PID každého procesu, ktorý spĺňa zadané kritériá
- formát príkazu: `pgrep` *`možnosti vzor`*

Posielanie signálov procesom

- signál je odkaz poslaný procesu (softvérové prerušenie)
- niektoré procesy obsahujú rutinu na spracovanie niektorých signálov
- keď proces rutinu na spracovanie signálu neobsahuje, tak signál spracúva kernel
- niektoré procesy niektoré signály ignorujú alebo blokujú
- jedine signál `SIGKILL` proces ignorovať nemôže

Mená a čísla signálov

Číslo signálu	Meno signálu	čo robí	vysvetlenie
1	`SIGHUP`	zloženie	odpojí telefónnu linku alebo terminálové pripojenie k procesu, najčastejšie však ide o znovunačítanie konfigurácie
2	`SIGINT`	prerušenie	preruší signál z klávesnice, väčšinou Control-C
9	`SIGKILL`	zrušenie	zruší proces bez toho, aby mu umožnil po sebe upratať
15	`SIGTERM`	ukončenie	programové ukončenie signálu, regulárne ukončenie

Príkaz `kill`

- príkaz `kill` sa používa na zaslanie signálu jednému alebo viacerým bežiacim procesom. Tento príkaz sa na ukončenie procesu používaný veľmi často.
- formát príkazu: `kill [-`*`signál`*`] PID...`
  - prednastavený signál je SIGTERM, 15.
- keď musíte ukončiť proces, tak sa odporúča vydať príkazy v nasledovnom poradí:
  - 15 : `SIGTERM`
  - 2 : `SIGINT`
  - 1 : `SIGHUP`
  - 9 : `SIGKILL`

Príkaz `pkill`

- príkaz `pkill` slúži na ukončenie procesu
- prednastavený je signál 15
- tento príkaz môže ukončiť proces podľa jeho mena
- formát príkazu: `pkill [-`*`možnosti`*`] vzor`

Riadenie jobov

- job je proces odvodený od terminálového procesu
- joby sú kontrolované terminálovými procesmi
- Bourne shell nepodporuje riadenie jobov.
  - existuje variant Bourne shellu, ktorý riadenie procesov podporuje (bash)

Príkazy riadenia jobov

Príkaz	Hodnota
`jobs`	zobrazí momentálne bežiace joby
`bg %n`	umiestni daný job do pozadia (`n` je ID jobu)
`fg %n`	umiestni daný job do popredia (`n` je ID jobu)
`^Z`	zastaví job v popredí
`stop %n`	zastaví job v pozadí (`n` je ID jobu)

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- opísať ako vznikajú procesy
- vypísať aktívne procesy systému pomocou príkazu `ps`
- nájsť určitý proces pomocou príkazu `pgrep`
- prediskutovať účel signálov na ovládanie aktivity procesov
- ukončiť proces pomocou príkazov `kill` a `pkill`
- použiť príkazy ovládania jobov na riadenie jobov bežiacich v shelli

11 Korn Shell

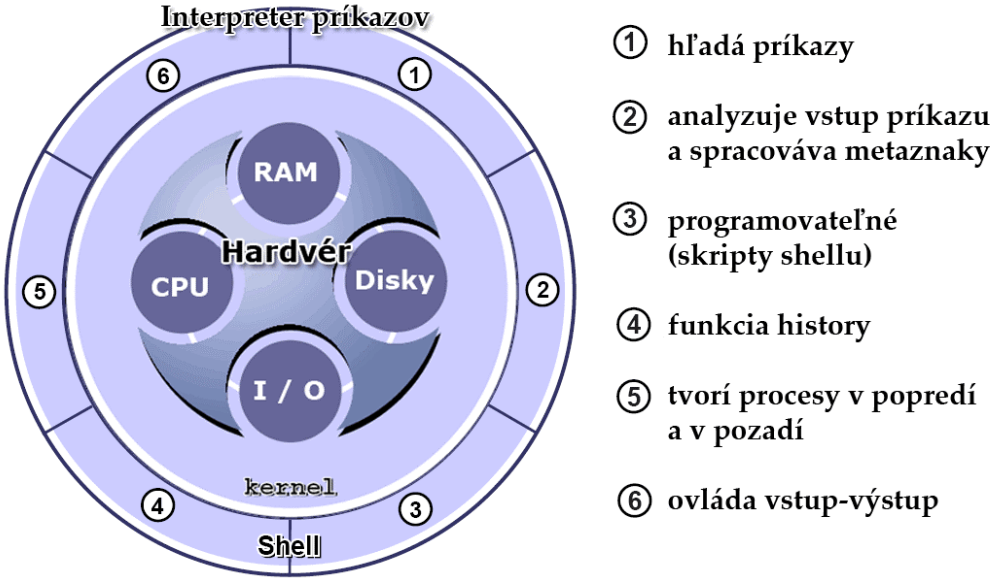
Ciele

Po ukončení tohto modulu by ste mali vedieť:

- opísať funkcie Korn shellu ako interpretera príkazov
- predviesť použitie úvodzoviek na maskovanie zvláštneho významu metaznakov Korn shellom
- definovať výraz štandardný vstup, štandardný výstup a štandardná chyba
- použiť metaznaky na presmerovanie štandardného vstupu, štandardného výstupu a štandardnej chyby
- spojiť dva alebo viac príkazov pomocou rúry
- do Korn shellu implementovať mechanizmus dokončovania názvov súborov
- použiť príkazy na zobrazenie, nastavenie a zrušenie nastavenia premenných shellu
- vyvolať mechanizmus `history` na opakovanie alebo úpravy predtým vykonaných príkazov
- použiť utilitu `alias` na prispôsobenie a skrátenie príkazov Unixu
- vytvoriť funkcie Korn shellu na tvorbu osobných príkazov
- definovať inicializačné súbory Korn shellu používané na prispôsobenie si používateľského prostredia

Shell ako interpreter príkazov

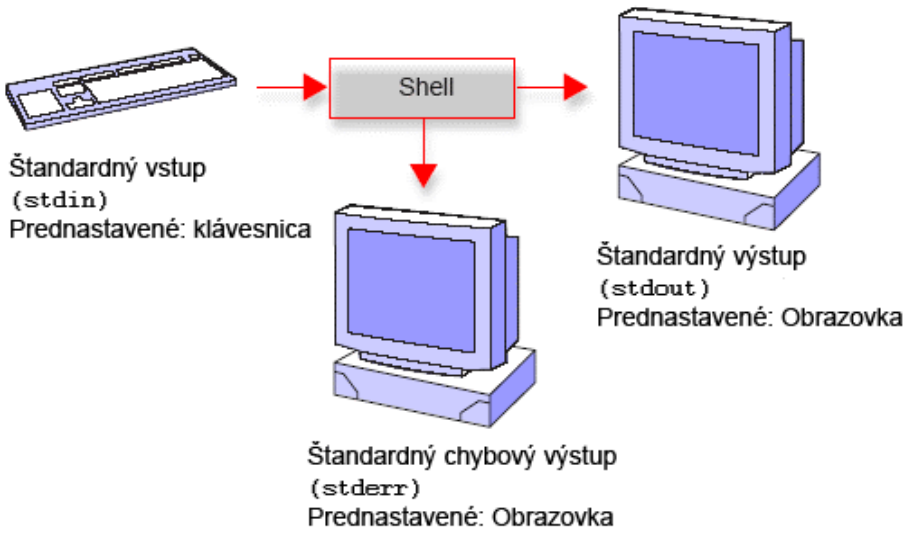
- shell vám umožňuje komunikovať s kernelom pomocou interpretácie príkazov, ktoré sú buď napísané na príkazovom riadku alebo uložené v skripte shellu
- shell prijme, analyzuje a spracúva vstup od používateľov alebo skriptov shellu a taktiež tvorí súvisiace hlásenia o chybe
- shelly sú zameniteľné, tak si môžete vybrať, ktorý príkazový interpreter kedy použiť – a ľahko prepnúť medzi shellom Bourne, C, Korn alebo ľubovoľným iným, ktorý je k dispozícii



Obr. 10 Interpreter príkazov

Presmerovanie vstupu/výstupu

- shell obyčajne prijíma príkazy z klávesnice a výstup zobrazuje na obrazovku
- použitím znakov presmerovania (< >) mu ale môžeme prikázať, aby presmeroval príkazový vstup a príkazový výstup
- presmerovávanie I/O je často používané na presmerovanie vstupu alebo výstupu do súborov na príkazovom riadku alebo v rámci skriptov shellu.



Obr. 11 Možnosti vstupu a výstupu príkazov

Opisovače súborov

0	stdin	štandardný vstup štandard
1	stdout	štandardný výstup
2	stderr	štandardná chyba

- presmerovanie sa môže vzťahovať okrem súborov aj na opisky (deskriptory) súborov
- `n>` presmeruje z deskriptoru `n`
- `>&n` presmeruje na deskriptor `n`

Presmerovanie `stdin`

`stdin` robí nasledovné:

- *`príkaz < názov_súboru`*
- *`príkaz 0< názov_súboru`*

Príkaz číta vstup zo súboru nazvaného *`názov_súboru`* namiesto čítania vstupu z klávesnice. Napríklad:

```
$ mailx user1 < ~/dante
```

Presmerovanie `stdout`

Nasledujúci príklad ukazuje ako presmerovať `stdout`:

- *`príkaz > názov_súboru`*
- *`príkaz 1> názov_súboru`*

Výstup príkazu je presmerovaný do súboru nazvaného *názov_súboru*. Keď *názov_súboru* neexistuje, tak súbor bude vytvorený. Keď názov_súboru existuje, tak presmerovanie prepíše obsah súboru. Napríklad:

```
$ ps -ef > zoznam_procesov
```

Presmerovanie `stdout` pomocou módu pripojenia (append)

Nasledovný príklad ukazuje presmerovanie stdout pomocou znakov pripojenia

- príkaz >> *názov_súboru*

Výstup príkazu je smerovaný do súboru s menom *názov_súboru* a je pripojený na koniec existujúceho súboru. Ak súbor neexistuje, tak je vytvorený.

```
$cat /etc/passwd > my_file;cat my_file
$ echo "Toto je môj súbor hesla" >> my_file;cat my_file
      Toto je môj súbor hesla
```

Presmerovanie `stderr`

Nasledujúci príklad ukazuje presmerovanie stderr:

- *príkaz* 2> /dev/null

Každý oznam chyby príkazu bude presmerovaný do súboru /dev/null. Tento súbor slúži na zahodenie nežiadúcich výpisov.

```
$ find /etc -type f -exec grep PASSREQ {} \; -print 2> /dev/null
      # PASSREQ určuje či prihlásenie potrebuje heslo
      PASSREQ=YES
      /etc/default/login
```

Presmerovanie `stderr` do `stdout`

Nasledujúci príkaz ukazuje presmerovanie stdout a stderr:

príkaz 1> *názov_súboru* 2>&1

syntax 2>&1 hovorí shellu, aby presmeroval stderr (2) na rovnaký súbor, ako je nasmerovaný stdout (1) na zápis, napríklad:

```
$ ls /var /no 1> dat 2>&1
$ more dat
      /no: No such file or directory
      /var:
      adm
      audit
      cron
```

Rúry

- shell umožňuje odovzdať výstup z jedného príkazu ako vstup ďalšiemu príkazu. Toto prepojenie sa nazýva rúra (pipe) (|).
- rúra je označená znakom | a je umiestnená medzi dva príkazy.
- formát príkazu: *príkaz | príkaz*
- príklady:

```
$ head -10 dante | tail -3 | lp
      request id is printerA-177 (standard input)
```

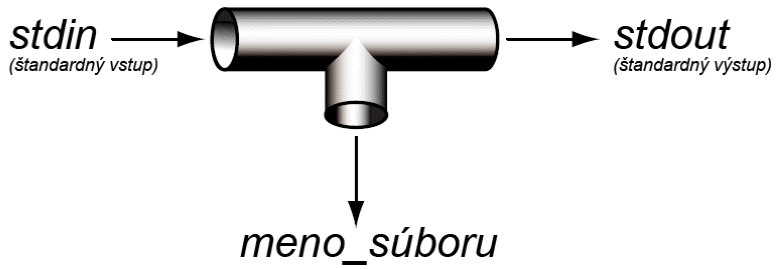
```
$ ps -ef | tail +2 | wc -l
      74
```

Vytváranie a pripájanie súborov pomocou príkazu tee

- príkaz `tee` skopíruje štandardný vstup do štandardného výstupu vytvorením duplikátu popisovaču nula až niekoľkých súborov
- pre prijímanie štandardného vstupu a zapisovanie na štandardný výstup a zároveň všetkých zadaných súborov použite príkaz `tee` spolu s rúrou (pipe) | . Viac o štandardnom vstupe, výstupe a rúre je spomenuté v module 11 Korn shell.

Zachytenie a zobrazenie výstupu pomocou príkazu `tee`

príkaz | tee *názov_súboru* | *príkaz*



Obr. 13 Princíp príkazu **`tee`**

Nastavenia možností v Korn shelli

- možnosti sú nastavenia, ktoré kontrolujú správanie shellu Korn
- na zapnutie nejakej možnosti napíšte:
  `$ set -o názov_možnosti`
- na vypnutie nejakej možnosti napíšte:
  `$ set +o názov_možnosti`
- na zobrazenie všetkých nastavených možností napíšte:
  `$ set –o`

Premenné Korn shellu

- premenné obsahujú informácie pre prispôsobenie shellu a informácie potrebné pre iné procesy, aby správne pracovali.
  - premenná je označená svojim zadaným menom, pričom jej hodnota je uložená v pamäti procesu
- programovanie v shelle Korn používa dva druhy premenných: premenné, ktoré sú exportované podprocesom, a tie ktoré exportované nie sú.

Prispôsobovanie si premenných shellu Korn

- premenná promptu `PS1`
  - reťazec promptu shellu je uložený v premennej shellu `PS1` a môžete si ho prispôsobiť podľa svojich požiadaviek.
- premenná `PATH`
  - premenná `PATH` obsahuje zoznam názvov adresárových ciest oddelených dvojbodkami

  - ak z príkazového riadku vyvoláte príkaz, tak shell hľadá v týchto adresároch zľava doprava, aby našiel príkaz, ktorý treba vykonať.
- príklady:

```
$ PS1='$USER@$HOSTNAME $'
$ export PS1
      angel@sun $

$ PATH=/usr/bin:/usr/ucb:/etc:.
$ export PATH
$ echo $PATH
/usr/bin:/opt/sfw/bin/:/usr/ucb:/etc:.
```

- z dôvodu bezpečnosti sa systémovým administrátorom nedoporučuje mať v premennej `PATH` aktuálny adresár (.)

Metaznaky Korn shellu

- opačná lomka `\` zabráni shellu vykonať funkciu nasledujúceho znaku (ak je špeciálny).
- pár jednoduchých úvodzoviek `' '`, ochráni pred vykonaním všetko, čo je medzi nimi.
- pár dvojitých úvodzoviek `" "`, ochráni pred vykonaním skoro všetko, čo je medzi úvodzovkami je
  - opačná lomka (`\`), dolár (`$`), a invertované úvodzovky (`‘`) nie sú dvojitými úvodzovkami chránené

Substitúcia príkazov

- príkazy medzi invertovanými úvodzovkami `‘` sú vykonané tak, že ich výstup je písaný do štandardného výstupu.
  - týmto spôsobom môže byť výstup niektorých príkazov upravený pre potrebu iných príkazov
- príklad:

```
$ filename="archive.‘date +20%y%m%d‘.tar"
$ echo $filename
      archive.20000419.tar
```

História

- Korn Shell vedie zoznam histórie posledne použitých príkazov
- fungovanie histórie umožňuje používateľom prezrieť, opakovať alebo upraviť predchádzajúce príkazy
- `history` podľa prednastavenia zobrazuje posledných 16 príkazov do štandartného výstupu.
- formát príkazu: `history`

Utilita Alias v shelle Korn

- alias je skratkové upozornenie v shelle Kornu na prispôsobenie a skrátenie príkazov UNIXu.
- alias je zadaný príkazom `alias`
- formát príkazu: `alias` *`názov=príkazový_reťazec`*
- príklad:

```
$ alias mc='mc -xc'
$ alias la='ls -la'
```

Funkcie shellu Korn

- funkcie sú účinným nástrojom programovania shellu a sú používané na vytváranie osobných príkazov
- funkcia je skupina príkazov UNIXu organizovaná ako osobitné rutiny
- použitie funkcie sa skladá z dvoch krokov:
  - definícia funkcie
  - vyvolanie funkcie

Definovanie funkcie

- funkcie sú účinným nástrojom programovania shellu a sú používané na tvorbu osobných príkazov
- funkcia je definovaná použitím všeobecného formátu:
  - názov funkcie { príkaz; . . . príkaz; }
- príklad:

```
$ function num { who | wc -l; }
$ num
      9
```

Súbor `~/.profile`

- súbor `.profile` je používateľom nastaviteľný inicializačný súbor, ktorý sa vykoná jednorázovo pri prihlásení do shellu a nachádza sa vo vašom domovskom adresári
- poskytuje vám možnosť si prispôsobiť vaše pracovné prostredie

Súbor `~/.kshrc`

- ak mate premennú prostredia s názvom `ENV` tak, že hodnota `ENV` je názov súboru, potom každý spustený shell Kornu vykoná inštrukcie nachádzajúce sa v `$ENV`.
- podľa zvyklosti sa tento súbor `ENV` väčšinou volá `.kshrc`. Obsahuje premenné a aliasy shellu Korn.
- súbor `ENV` je vykonaný zakaždým, keď sa naštartuje podshell `ksh`.

Súbor `~/.dtprofile`

- v prostredí CDE sa nachádza aj súbor nazvaný `.dtprofile.`
- je umiestnený vo vašom domovskom adresári a určuje všeobecné a prispôsobené nastavenia CDE.
- súbor `.dtprofile` značí, že vaše hodnoty premenných môžu prepísať prednastavené hodnoty

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- opísať funkcie Korn shellu ako príkazového interpreteru
- predviesť použitie úvodzoviek na maskovanie zvláštnych významu metaznakov shellu Korn
- definovať pojmy ako štandardný vstup, štandardný výstup a štandardná chyba
- použiť metaznaky na presmerovanie štandardného vstupu, štandardného výstupu a štandardnej chyby
- spojiť dva alebo viac príkazov použitím rúry
- použiť mechanizmus dokončovania názvov v Korn shelli

- použiť príkazy na prezeranie, nastavenie a zvrátenie nastavenia premenných shellu.
- vyvolať mechanizmus `history` na opakovanie alebo úpravu predtým vykonaných príkazov
- použiť `alias` na prispôsobenie a skrátenie príkazov UNIXu
- vytvoriť funkcie Korn shellu na tvorbu osobných príkazov
- definovať inicializačné súbory Korn shellu používané na prispôsobenie si používateľského prostredia

12 Úvod do sed a awk

Ciele

Po ukončení tohto modulu by ste mali vedieť:

- použiť streamový editor (`sed`) na úpravu obsahu textového súboru z príkazového riadku a odoslať výsledok do štandardného výstupu
- použiť príkazy `sed` na vymazanie riadkov, tlač riadkov obsahujúcich vzor, pridanie textu na riadky alebo zmenu znakov za použitia metaznakov regulárnych výrazov
- použiť `awk` na prehľadanie textových súborov alebo štandardného vstupu za účelom zobrazenia požadovaných dát, zmeny formátu dátumu alebo pridania textu k už existujúcim dátam

12.1 Editor streamu sed

Program `sed` sa používa na úpravu údajov v súboroch bez toho, aby ste ich otvorili v interaktívnom editore ako napríklad `vi`.

- umožňuje vám zadať príkazy na úpravy alebo zmeny súboru z príkazového riadku a podľa prednastavenia poslať výstup na obrazovku
- najčastejšie je využitie pri rýchlej realizácii rovnakých zmien vo viacerých súboroch. Takisto slúži na čítanie skriptov, ktoré potrebujú administrátori.
- formát príkazu:

```
sed [možnosti] [adresa] príkaz súbor... [>nový_súbor]
```

Metaznaky regulárnych výrazov používané sed-om

Znak	význam	vzor	čo robí
^	začiatok riadku	’^vzor’	označí všetky riadky začínajúce s “vzor”
$	koniec riadku	’vzor$’	označí všetky riadky končiace s “vzor”
.	označí jeden znak	’p.....n’	označí riadky obsahujúce p, nasledované 5 znakmi a nasledované n
[ ]	označí jeden znak vo vzore	’[Vv]zor’	označí riadok obsahujúci “Vzor” alebo “vzor”
*	označí predchádzajúci predmet 0 alebo viackrát	’[a-z]*’	onačí malé alfanumerické znaky
[^]	označí jeden znak nie vo vzore	’[^a-m]zor’	označí riadky neobsahujúce od “a” po “m” nasledované “zor”

Substitúcia slov

- na substitúciu (nahradenie) slov slúži príkaz `s`
- formát príkazu:
- `[adresa1[,adresa2]]`
  `s/regularny_vyraz/nahradzany_text/[priznaky]`
- príznaky môžu byť
  - `n` – nahradí n-tý výskyt výrazu
  - `g` – nahradí všetky výskyty výrazu
  - `p` – vypíše úspešné nahradenie
- príklad:

```
$ cat súbor
      Mam troch psov a dvoch kocurov
$ sed 's/psov/kocurov/pg'; 's/kocurov/slonov/pg' súbor
      Mam troch kocurov a dvoch kocurov
      Mam troch slonov a dvoch slonov
```

- nahradenia sa vykonávajú postupne
- adresy predstavujú čísla riadkov, medzi ktorými sa vykonáva substitúcia; ak je zadaná len jedna adresa, substitúcia je vykonaná v len tomto riadku

Mazanie slov

- na mazanie slúži príkaz `d`
- formát príkazu:

`[adresa1[,adresa2]]d`

- príklad:

```
$ cat súbor2
      riadok jedna
      riadok dva
      riadok tri
$ sed '1,2d' súbor2
      riadok tri
```

- nahradenia sa vykonávajú postupne
- adresy predstavujú čísla riadkov, ktoré sa majú odstrániť. Ak je zadaná jediná adresa, substitúcia je vykonaná len v tomto riadku

12.2 Príkaz awk

Textové procedúry používania príkazu awk

- príkaz `awk` je flexibilný textový procesor
- prechádza súbor riadok po riadku, vyhľadáva riadky zhodné so zadaným vzorom a na ne aplikuje zvolené činnosti
- medzi základné použitia `awk` patria zmeny formátu dát, preskupovanie stĺpcov a pridávanie k už existujúcemu textu
- formát príkazu: `awk '{`*cinnost*`}' [`*názov_súboru*`]`
- `awk` je silný a veľký nástroj, preto spomenieme len jeho základnú syntax

Základná syntax awk programu:

```
BEGIN
 vyhladavaci_reťazec1 {cinnost}
 vyhladavaci_reťazec2 {cinnost}
```

```
  ...
END
```

vyhľadávacie reťazce

/reťazec/ Vyhľadá slovo *reťazec*
/^reťazec/ Vyhľadá riadky, ktoré začínajú slovom *reťazec*
/reťazec$/ Vyhľadá riadky, ktoré končia slovom *reťazec*

- Medzi reťazce je možné vložiť logickú spojku alebo (|)
  /(reťazec1)|(reťazec2)/
- Vyhľadávanie sa môže v rozsahu medzi dvoma reťazcami
  /reťazec1/,/reťazec2/
- Podmienky pre vyhľadávanie môžu byť rôzne (napr. číslo riadku) a môžu sa používať porovnávacie operátory
  == != < > <= >=
- Medzi podmienkami môžeme používať logické operácie ALEBO (||), A (&&)
  [*zoznam_pismen* ALEBO *rozsah*]
  Zodpovedá akémukoľvek písmenu v zozname alebo v rozsahu

  [^*zoznam_pismen* ALEBO *rozsah*]
  Zodpovedá akémukoľvek písmenu, ktoré nie je v zozname alebo je v rozsahu

  . Zodpovedá jednému ľubovoľnému znaku
  * Zodpovedá 0 alebo viac výskytom predchádzajúceho reťazca
  ? Zodpovedá 0 alebo1 výskytu predchádzajúceho reťazca
  + Zodpovedá aspoň jednému výskytu predchádzajúceho reťazca
  Ak je niektorý z týchto znakov súčasťou reťazca, zadávame predeň znak „\“

Premenné

- $0; $1,$2,$3,... čísla polí (že $0 je celý záznam, ostatné sú jednotlivé položky)
- NR Počet záznamov (riadkov)
- NF Počet polí
- FILENAME Názov vstupného súboru
- FS Oddeľovač polí (štandardne: " ")
- RS Oddeľovač záznamov (štandardne: "\n")
- OFS Oddeľovač výstupných polí (štandardne: " ")
- ORS Oddeľovač výstupných záznamov (štandardne: "\n")
- OFMT Výstupný formát: (štandardne: "%.6g")

- Vyhľadávanie dokážeme obmedziť na zvolené polia
  $<pole> ~ /<reťazec>/
  Hľadá reťazec v definovanom poli
  $<pole>!~ /<reťazec>/
  Hľadá reťazec vo všetkých poliach okrem definovaného

Aritmetické operácie:

- + - * / % ++ --
- sqrt() log() exp() int()

Funkcie pre prácu s reťazcami

- `length()`
  - dĺžka reťazca
- `substr(`*`reťazec`*`, `*`podreťazec`*`, `*`max_dlzka:podretazca`*`)`
  - vráti časť reťazca
- `split(`*`reťazec`*`, `*`pole`*`, [`*`oddelovac_poli`*`])`
  - rozdelí reťazec na základe oddeľovača a jednotlivé slová uloží do poľa s počiatočným indexom 1
- `index(`*`reťazec`*`, `*`hladany_reťazec`*
  - nájde pozíciu hľadaného reťazca a reťazci
- `sprintf()`
  - uloží formátovaný výstup do reťazca

Riadiace štruktúry

- `awk` používa pre riadiace štruktúry štandardnú céčkovu syntax

Výstup

- `print <i1>, <i2>,...`
  - vypíše prvky oddelené OFS a výpis ukončí novým riadkom
- `print <i1> <i2> ...`
  - vypíše prvky ako jeden celok a výpis ukončí novým riadkom
- `printf(`*`reťazec_s_formatovacimi_znakmi`*`,[`*`parametre`*`])`
  - formátovaný výpis na základne formátovacích znakov v tvare *`%kod`*
- formátovacie kódy sú:
  - `d` vypíše číslo v desiatkovej sústave
  - `o` vypíše číslo v osmičkovej sústave
  - `x` vypíše číslo v šestnástkovej sústave
  - `c` vypíše znak zadaný jeho numerickým kódom
  - `s` vypíše reťazec
  - `e` vypíše číslo v exponenciálnom tvare
  - `f` vypíše číslo vo formáte plávajucej desatinnej čiarky g vypíše číslo buď v exponenciálnom tvare ale vo formáte plávajúcej desatinnej čiarky
- Pri použití funkcií `print` a `printf` je možne použiť presmerovanie výstupu (>, >>) alebo rúry (`|`)

Príklady

- **`$ cat súbor3`**

```
názov    ks  cena
procesor 25  1000
cooler    8  350
HDD      20  2500
```

vypíšte hodnoty v 3. a prvom stĺpci:

```
$ awk '{print $3, $1}' súbor3
cena názov
1000 procesor
350 cooler
2500 HDD
```

- príklad: pre ten istý súbor zobrazte riadky, kde je hodnota v 2. stĺpci nižšia ako 10

```
$ awk '$2<10' súbor3
     cooler 8 350
```

- Vypíšte zoznam používateľov v systéme
  `$ awk '{FS=":"} { print $1 }' /etc/passwd`
- Vypíšte zoznam používateľov spolu a veľkosť pošty, ktorú majú uloženú na serveri, zoznam zoraďte od najväčšieho po najmenší
  `ls -la /var/mail/ | awk '{ print $5, $3 }' | sort -g -r`

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- použiť z príkazového riadku streamový editor (`sed`) na úpravu obsahu textového súboru a výsledok poslať do štandardného výstupu
- použiť príkazy `sed` na mazanie riadkov, tlač riadkov obsahujúcich vzor, pridať riadkom text alebo zmeniť znaky za použitia metaznakov regulárnych výrazov
- použiť `awk` na prehľadanie textových súborov alebo štandardného vstupu za účelom zobrazenia požadovaných dát, zmeny formátu dát a pridania textu k existujúcim dátam

13 Čítanie skriptov shellu

Ciele

Po skončení tohto modulu by ste mali vedieť:

- určiť, ktorý program shellu interpretuje riadky skriptu shellu
- vysvetliť, ako sú argumenty príkazového riadku odovzdané shellovému skriptu so špeciálnymi premennými nazývanými pozičné parametre
- použiť dva podmienečné príkazy: `if, case, for`
- vysvetliť obsah jednoduchého administrátorského skriptu Bourne shellu

Základy skriptov shellu

- skript shellu je ASCII súbor, ktorý obsahuje postupnosť príkazov a komentárov
- komentáre sú texty, slúžiace na dokumentáciu práce skriptu, opisujú význam jednotlivých riadkov skriptu
- komentár začína znakom mriežka (`#`) a končí ukončením riadku

Určovanie typu shellového skriptu

- vrchný riadok určuje program shellu, ktorý vykoná riadky skriptu
- napríklad, v skripte Bourne, prvý riadok je: `#!/bin/sh.`
- znaky `#!` na začiatku súboru používa kernel na identifikáciu programu, ktorý interpretuje riadky skriptu

Tvorenie základného skriptu shellu

1. editorom `vi` vytvorte súbor nazvaný `myfile1` a každý vložený príkaz nasledujte tabulátorom a znakom # ktorý ho definuje ako komentár
2. zabezpečte, aby bol súbor spustiteľný a spustite nový script shellu tak, že na príkazovom riadku napíšete jeho meno.

```
$ vi myfile1
      who # Kto je prihlaseny v systeme
      date # Vypisanie aktualneho datumu a casu
      ls -l # Vypis súborov v aktualnom adresári
$ chmod 755 myfile1
$ myfile1
```

Skripty Bourne shellu

- aby ste vedeli prečítať a pochopiť obsah základného skriptu shellu, tak musíte:
  - porozumieť, ako sú skriptu odovzdávané argumenty pomocou špeciálnych zabudovaných premenných nazvaných pozičné parametre
  - určiť a analyzovať jednoduché podmienečné konštrukcie a kontrolu toku

Pozičné parametre

- špeciálne vstavané premenné shellu, nazývané *pozičné parametre* odovzdávajú argumenty z príkazového riadku do skriptu shellu
- na príkazovom riadku je každé slovo oddelené medzerou, ktorá nasleduje po názve skriptu, nazvané argumentom. Na tieto argumenty sú v shellovom skripte odkazy pomocou pozičných parametrov

- formát príkazu:
```
názov_skriptu argument1 argument2 argument3 ...
```

Podmienené príkazy a kontrola toku

- podmienečné príkazy umožňujú vykonať niektoré úkony na základe toho, či je podmienka splnená alebo nie
- najjednoduchšou formou podmienečného príkazu je príkaz `if`, ktorý umožňuje otestovať podmienku a potom zmeniť tok vykonávania shellového skriptu na základe výsledkov testu
- formát príkazu:

```
if výraz
then
     príkaz
     príkaz
else
     príkaz
fi
```

Príkaz `case`

príkaz `case` sa používa, keď existuje veľa podmienok na testovanie
formát príkazu:

```
case premenná in
hodnota1 ) príkaz1
           príkaz2
          ;;
hodnota2 ) príkaz3
           príkaz4
          ;;

* )        príkaz5
          ;;
esac
```

Príkaz `for`

- príkaz `for` sa používa na vytvorenie cyklu s konečným počtom opakovaní, vykoná sa pre každý prvok zoznamu
- formát príkazu:

```
for { premenná } in { zoznam }
do
     príkaz1
     príkaz2
done
```

Príkaz `exit`

- vstavaný príkaz shellu `exit` poskytuje možnosť okamžite zrušiť vykonávanie shellového skriptu
- príkaz `exit` je často používaný ako spôsob ukončenia alebo prerušenia skriptu a návratu na príkazový riadok
- formát príkazu: `exit n`

Overte si vedomosti

Skontrolujte, či viete vykonať alebo odpovedať na nasledovné:

- určiť, ktorý program shellu interpretuje riadky skriptu shellu
- vysvetliť, ako sú argumenty príkazového riadku podávané skriptu shellu pomocou špeciálnych premenných nazvaných pozičné parametre
- ukázať použitie dvoch podmienečných príkazov: `if` a `case` a taktiež príkazu `test`
- vysvetliť obsah jednoduchého skriptu Bourne shellu

APPENDIX
Zoznam použitých príkazov Unixu

who	zobrazí zoznam práve prihlásených používateľov
alias	vytvorí *alias* príkazu alebo skupiny príkazov
awk	textový procesor
bg	umiestni daný job do pozadia
cal	kalendár pre aktuálny mesiac a rok
cancel	zruší požiadavku na tlač
cat	zobrazí obsah jedného alebo viacerých súborov na obrazovku
cd	umožní zmeniť pracovného adresára
cmp	porovná binárne alebo textové súbory
compress	komprimuje súbor
cp	skopíruje súbor alebo adresár
cpio	archivuje alebo extrahuje súbor na pásku alebo do súboru
date	zobrazí aktuálny dátum a čas
diff	vyhľadá rozdiely v textových súboroch
egrep	hľadá v obsahu súboru vzor pomocou rozšírených metaznakov regulárneho výrazu
eject	vysunie prenosné médium
fg	umiestni daný job do popredia
fgrep	hľadá v súbore vzor vyjadrený ako pevný reťazec
file	zistí typ súboru
find	nájde súbor
ftp	dovolí prenos súborov medzi lokálnym a vzdialeným systémom
grep	hľadá regulárny výraz alebo znakový vzor v súbore
gunzip	dekomprimuje súbor
gzcat	dekomprimuje súbor typu gzip a výstup pošle na obrazovku bez zmeny (dátum, čas) komprimovaného súboru
gzip	komprimuje súbor
head	zobrazí prvých 10 riadkov súboru
history	zobrazí posledné zadané príkazy
chmod	zmení prístupové práva súboru alebo adresára
id	zobrazí detaily skupiny používateľa
jar	archivuje súbory
jobs	zobrazí momentálne bežiace joby
kill	zašle signál procesu
lp	zaradí textový súbor na tlač
lpstat	zobrazí stav tlačového frontu
ls	zobrazí súbory a adresáre určeného adresára
mailx	odošle e-mail
man	zobrazí manuálové stránky
mkdir	vytvorí adresár
more	umožní prezrieť obsah dlhého textového súboru
mv	premiestni alebo premenuje súbor
passwd	umožní zmena hesla cez príkazový riadok
pg	umožní prezrieť súbor, ktorý je dlhší ako vojde na obrazovku
pgrep	zobrazí PID
pkill	ukončí proces

pr	nastaví parametre tlače a vytlačí súbor
ps	vypíše procesy momentálne bežiace na systéme
pwd	zobrazí cestu aktuálneho pracovného adresára
rcp	umožní kopírovanie súborov na vzdialený systém
rlogin	vytvorí reláciu vzdialeného pripojenia
rm	odstráni súbor
rmdir	odstráni adresár
rsh	spustí príkaz na vzdialenom systéme
scp	umožní bezpečné kopírovanie súborov na vzdialený systém
sed	umožní úpravu údajov v súboroch
set	zobrazí premenné prostredia
sftp	zabezpečí bezpečný prenos súborov medzi lokálnym a vzdialeným systémom
slogin	vytvorí bezpečnú reláciu vzdialeného pripojenia
sort	triedi textové riadky v súbore a výsledok zobrazuje na obrazovke
ssh	umožní bezpečné pripojenie k vzdialenému systému
stop	zastaví job v pozadí
tail	zobrazí posledných 10 riadkov súboru
tar	archivuje a vyberá súbory do a z jedného súboru
tee	skopíruje štandardný vstup do štandardného výstupu
telnet	pripojí k vzdialenému systému
touch	vytvorí prázdny súbor
umask	určí prednastavené práva pridelené novovytvoreným súborom a adresárom
uname	zobrazí informácie o systéme
uncompress	dekomprimuje súbor
unzip	dekomprimuje súbor
vi	interaktívny editor na tvorbu alebo zmenu textových súborov
volcheck	informuje správu zväzkov o vložení prenosného média (diskety)
wc	zobrazí počet riadkov, slov a znakov obsiahnutých v súbore
whoami	zobrazí údaje o vlastnom prihlasovacom mene
zcat	dekomprimuje súbor a výstup pošle na obrazovku so zachovaním pôvodného komprimovaného súboru
zip	komprimuje súbor

Literatúra

1. ČADA, O.: Operační systémy, Grada, 1994.
2. DOUGHERTY. D. – ROBBINS, A.: Sed and Awk, 2nd edition, O'Reilly and Associates, 1997.
3. FRIEDL, J.: Mastering Regular Expressions, O'Reilly and Associates, 2002.
4. POWERS, S. - PEEK, J. - O'REILLY, T. – LOUKIDES, M. A KOL.: UNIX Power Tools, 3. vydanie, O'Reilly and Associates, 2002.
5. ROSENBLATT, B.: Learning the Korn Shell, O'Reilly and Associates, 1993.
6. SHEER, P.: LINUX: Rute User's Tutorial and Exposition, 1. vydanie, 2002.
7. SOVA, M.: Unix V, úvod do operačného systému, Grada, 1991.
8. TAYLOR, D.: Solaris™ 9 For Dummies®, Wiley Publishing, Inc., 2003

internetové zdroje:

1. http://www.vectorsite.net/tsawk.html (AWK)
2. http://pegasus.rutgers.edu/~elflord/unix/sed.html (SED)
3. http://math.la.asu.edu/vi_tutorial/vicontents.html (VI)
4. http://www.unixguide.net/
5. http://docs.sun.com/app/docs/prod/solaris.10
6. http://www.sun.com/bigadmin/
7. http://www.adminschoice.com/

Mgr. Peter Švec, doc. Dr. Ing. Miroslav Fikar,
Ing. Jozef Dzivák, prof. RNDr. Libor Vozár, CSc.

Základy práce v operačnom systéme Solaris

Za odbornú stránku skrípt zodpovedajú autori.
Text korigovali autori.

Vydavateľ: Fakulta prírodných vied Univerzity Konštantína Filozofa v Nitre
Katedra informatizácie a riadenia procesov FCHPT STU v Bratislave
Tlač: Michal Vaško, Námestie Kráľovnej pokoja 3, 080 01 Prešov
Vydanie: prvé
Náklad: 400 kusov
Strán: 66
Formát: A4

Schválené vedením FPV UKF v Nitre dňa 19. 9. 2005

ISBN 80-8050-881-X