Základy operačného systému Linux

(študijný materiál)

Verzia: 0.5

2021

Obsah

Obsah

1. Súborový systém

1.1. Adresárová štruktúra

Adresár /bin obsahuje **najdôležitejšie binárne súbory operačného systému.** V systéme sa nachádza viac adresárov s názvom bin. Je to kvôli prehľadnosti. Jednotlivé bin adresáre obsahujú spustiteľné súbory k rôznym typom programov. Adresár /bin obsahuje základné spustiteľné súbory, môžete tu nájsť napr. programy ls, cat, chown, chmod atď.

Adresár /boot obsahuje **súbory používané pri bootovaní systému**. Najdôležitejší (aj keď nie jediný nevyhnutný pre správne nabootovanie systému) je súbor vmlinuz, ktorý obsahuje jadro systému (kernel). Samotný kernel je srdcom celého systému; je to niečo ako hlavná časť Linuxu. Je nutný pre samotný beh systému, najmä správu pamäte, riadenie procesov; môže však už obsahovať aj ovládače pre zariadenia. Bez kernelu by žiaden program nemohol byť ani spustený. Okrem jadra je dôležité poznať programy grub a lilo. Tieto programy majú za úlohu pri bootovaní počítača nahrať do pamäte kernel. Hlavnou funkcionalitou sú si veľmi podobné; buď je použitý jeden, alebo druhý. V CentOS Linux sa štandardne používa grub.

Adresár /dev (device - zariadenie) obsahuje prepojenia na **hardvérové zariadenia systému**. V operačnom systéme Linux okrem už dobre známej reprezentácie dát (texty, obrázky, programy, knižnice, …) existujú ešte špeciálne druhy súborov. Práve jedným typom špeciálneho súboru je súbor, ktorý reprezentuje určité (napr. hardvérové) zariadenie. V adresári dev teda môžete nájsť súbory, ktoré reprezentujú napr. myš (/dev/mouse), IDE harddisk (/dev/hda, /dev/hdb), SCSI zariadenie (/dev/sda, /dev/sdb), zvukovú kartu (/dev/dsp), atď. Spomenutý systém súborov reprezentujúcich zariadenia operačného systému umožňuje jednotnú prácu takmer so všetkým, čo sa v systéme nachádza. Programátor si stále vystačí s niekoľkými štandardnými nástrojmi (príkazmi) na otvorenie súboru, čítanie, zapisovanie, zatvorenie súboru a ešte určitými špeciálnymi nastaveniami.

Adresár /etc je **povinný adresár** s **najdôležitejšími konfiguračnými súbormi** k jednotlivým nástrojom. Napríklad obsahuje súbor **/etc/passwd** (súbor obsahujúci informácie o používateľoch), súbor **/etc/shadow** (súbor obsahujúci informácie o heslách používateľov) a súbor **/etc/group** (súbor obsahujúci informácie o skupinách).

Adresár /home **obsahuje domovské adresáre všetkých používateľov**, až na super používateľa root (ktorý má domovský adresár /root).

Adresár /lib obsahuje **hlavné knižnice systému**. Obzvlášť dôležitý je adresár /lib/modules. Čím má mať operačný systém viac rôznych funkcií a podporovať viac hardvéru, tým musí obsahovať viac rôznych programov s danou funkcionalitou. Takýchto programov je ale obrovské množstvo – od základnej sieťovej podpory až po špeciálny hardwér na veľmi špecifické použitie. Je zrejmé, že bežnému používateľovi budú ovládače na takýto typ hardvéru zbytočné. Aby nebolo jadro systému (vyššie spomínaný kernel) prepchaté možno práve zbytočnými programami, je možné ho nejakým spôsobom poskladať práve len z tých častí, ktoré sú pre nás potrebné. Práve tieto súčiastky, ktoré sa dajú podľa potreby obmieňať, sa

nazývajú moduly (modules). Ako príklad modulu môžem uviesť ovládač na sieťovú kartu; takisto to však môže byť podpora určitého sieťového protokolu.

Adresár /media slúži **na pripojenie prenosných médií** (napr. **cdrom**, floppy) v operačnom systéme Linux. Veľa systémov má len adresár /mnt.

Adresár /mnt je používané v spolupráci so zariadeniami z adresára /dev. Na rozdiel od adresára /dev adresár /mnt neobsahuje žiadne špeciálne súbory. Čo sa týka špeciálnych zariadení adresára /dev z pohľadu používateľa, je dôležité poznať príkaz mount, ktorý "primontuje" určité zariadenie do našej adresárovej štruktúry. Obyčajne sa **zariadenia pripájajú (mountujú) práve do podadresára /mnt (alebo /media).** Nie je to vôbec podmienka. Skôr ide o nepísané pravidlo, ktoré sprehľadňuje systém. Pripojenie (primountovanie) zariadenia do systému („napojenie" jeho obsahu do adresára) sa vykoná nástrojom (príkazom) mount.

Adresár /proc je špeciálny adresár. Obsah tohto adresára sa nenachádza na disku ani na inom podobnom médiu, ale je to len určitý obraz pamäte. Inými slovami **ide o virtuálny súborový systém**. Ako bolo spomenuté pri adresári/mnt a pripájaní (mountovaní), do adresára je možné pripojiť (namountovať) takmer hocičo, len je potrebný nástroj, ktorý to zobrazí vo forme súborov. Obsah adresára /proc je generovaný jadrom operačného systému (kernelom) a **obsahuje rôzne údaje o práve bežiacom systéme**. Je to robené v reálnom čase (t.j. nie len napr. pri boote systému). Pre **každý proces v systéme obsahuje práve 1 adresár so štruktúrou.**

Adresár /root je **domovský adresár superpoužívateľa (roota)** operačného systému.

Adresár /sbin obsahuje **systémové spustiteľné súbory** (napr. program init).

Adresár /srv obvykle obsahuje **podadresáre ftp (**domovský adresár pre ftp) a **www** (hlavný adresár apache).

Adresár /tmp slúži na **ukladanie dočasných súborov rôznych programov (nástrojov)**.

Adresár /usr obsahuje súbory, ktoré už **nie sú až také základné pre samotný systém**. Na prvý pohľad je zrejmá podobnosť podadresárov s adresármi v roote – sú tu podadresáre bin, etc, lib, sbin … Samotné podadresáre majú totožný význam ako podadresáre v koreňovom adresári – sem sa ale **inštalujú už konkrétne špecifické programy**, napr. rôzne editory, komprimačné programy, a pod.

Adresár /var je adresár, ktorý obsahuje dôležité **informácie o záznamoch (logoch)** (/var/log), **poštu** (/var/spool/mail) a pod. O záznamoch (logoch) platí to, čo aj o konfiguračných súboroch – je potrebné hľadať súbor/adresár totožný/podobný s názvom programu.

1.2. Cesta

Absolútna cesta začína znakom "/" (označenie koreňového adresára) a musí obsahovať všetky adresáre vedúce k výslednému adresáru, do ktorého chceme vstúpiť, resp. zobraziť jeho

obsah alebo spustiť nejaký program. Na druhej strane, **relatívna cesta** sa vzťahuje k aktuálnemu adresáru. Príklady relatívnych ciest:

- ../home/user/ahoj.txt
- ../../home/user/ahoj.txt
- ./test/ahoj.txt
- ~/test/ahoj.txt (~ - zástupný znak pre domovský adresár)

Ak máme súbor **cron** nachádzajúci sa v adresári **/etc/default**:

```
root@Linux:/etc/default# ls /etc/default/
apache2          cron       ifupdown  rcS        ssh      wide-
dhcpv6-client
bind9            dbus       klogd     rrdcached  syslogd  xinetd
bind9.dpkg-dist  devpts     locale    rsync      sysstat
bootlogd         fetchmail  portmap   samba      tmpfs
collectd         halt       quota     saslauthd  useradd
```

1.3. Špeciálne adresáre

V operačnom systéme Linux, podobne ako v operačnom systéme DOS, existuje niekoľko špecifických adresárov:

- adresár "." špecifikuje **aktuálny adresár** a
- adresár ".." špecifikuje **nadradený adresár**.
- adresár "~" špecifikuje **domovský adresár**.

```
root@Linux:/etc/default# ls -la /home
total 24
drwxr-xr-x  6 root   root   4096 Aug 17 17:09 .
drwxr-xr-x 20 root   root   4096 Aug 10 19:14 ..
drwxr-xr-x  3 maria  maria   4096 Sep 14  2012 maria
drwxr-xr-x  2 martin martin 4096 Jul 30 13:15 martin
drwxr-xr-x  4 user   user   4096 Aug 18 20:07 user
drwxr-xr-x 11 stevo  stevo  4096 Aug 12 16:20 stevo
```

Tieto zástupné znaky existujú v každom adresári, nie vždy však majú tento význam. Napríklad hlavný adresár má ako nadradený adresár sám seba.

Ďalším špeciálnym adresárom je **domovský adresár**. Označuje sa zástupným znakom "~". Ak máme ako domovský adresár ***/home/user*** namiesto:

```
/home/user/mojedata/notes.txt
```

môžeme napísať (ak domovský adresár používateľa je `/home/user/`):

```
~/mojedata/notes.txt
```

Vyššie uvedený zápis môžeme považovať za **relatívnu cestu.**

1.4. Nástroje (príkazy) na prácu so súbormi a adresármi

1.4.1. touch

Príkaz (nástroj) **touch** má niekoľko spôsobov použitia, ale v praxi využijeme najviac možnosť vytvárať nové prázdne súbory. Pôvodný zámer tohto príkazu bolo **aktualizovanie časovej pečiatky súborov**. Avšak, vzhľadom k programovej chybe, príkaz touch vytvoril **prázdny súbor**, ak daný súbor neexistuje. V čase, keď bola táto chyba objavená, ľudia používali príkaz touch na vytvorenie prázdnych súborov.

Syntax:

touch názov_súboru

Príklad:

touch /home/user/ahoj

```
root@Linux:/home/user# ls /etc/default/
apache2          cron       ifupdown  rcS        ssh      wide-
dhcpv6-client
bind9            dbus       klogd     rrdcached  syslogd  xinetd
bind9.dpkg-dist  devpts     locale    rsync      sysstat
bootlogd         fetchmail  portmap   samba      tmpfs
collectd         halt       quota     saslauthd  useradd

root@Linux:/home/user# touch /etc/default/ahoj.txt
root@Linux:/home/user# ls /etc/default/
ahoj.txt         cron       klogd      rsync      tmpfs
apache2          dbus       locale     samba      useradd
```

```
bind9            devpts     portmap    saslauthd  wide-dhcpv6-client
bootlogd         halt       rcS        syslogd
collectd         ifupdown   rrdcached  sysstat
ODBCDataSources  inittab    postfix
```

Príkaz **touch** vie vytvoriť viac ako 1 súbor v danom čase.

```
[user@Linux]$ touch file-{one,two}-{a,b}
[user@Linux]$ ls -l
-rw-rw-r-- 1 user  user  0 Apr 8 10:33 file-one-a
-rw-rw-r-- 1 user  user  0 Apr 8 10:33 file-one-b
-rw-rw-r-- 1 user  user  0 Apr 8 10:33 file-two-a
-rw-rw-r-- 1 user  user  0 Apr 8 10:33 file-two-b
-rw-rw-r-- 1 user  user  0 Apr 8 10:29 newfile
```

1.4.2. ls (list)

Prvým z nástrojov (príkazov) potrebných pre prácu so súbormi a adresármi je nástroj nástroj **ls**. Účelom tohto nástroja je **zobrazenie obsahu adresára**.

Syntax:

ls [prepínače] názov_adresára

Príklad:

ls /etc

```
root@Linux:/etc/default# ls /etc
ODBCDataSources          inittab            postfix
X11                      inputrc            ppp
adduser.conf             insserv            profile
adjtime                  insserv.conf       profile.d
ajaxplorer               insserv.conf.d     protocols
aliases                  iproute2           python
aliases.db               issue              python2.6
alternatives             issue.net          python3.1
apache2                  javascript-common  quotagrpadmins
apparmor.d               kernel             quotatab
bash_completion.d        ld.so.conf.d       rc1.d
```

Prepínače:

- **„-a“** – zobrazí aj skryté súbory (skrytý súbor sa začína na „.“)

```
root@Linux:/etc/default# ls -a /etc
.                       init.d              phpmyadmin
..                      initramfs-tools     postfix
.pwd.lock               inittab             ppp
ODBCDataSources         inputrc             profile
X11                     insserv             profile.d
adduser.conf            insserv.conf        protocols
adjtime                 insserv.conf.d      python
ajaxplorer              iproute2            python2.6
aliases                 issue               python3.1
aliases.db              issue.net           quotagrpadmins
alternatives            javascript-common   quotatab
apache2                 kernel              rc.local
apparmor.d              ld.so.cache         rc0.d
apt                     ld.so.conf          rc1.d
```

- **„-l“ – zobrazí podrobnosti o súboroch a adresároch („dlhý“ long zoznam)**

```
ls -l /etc
total 872
drwxr-xr-x 2 root  root    4096 Mar 25  2010 ODBCDataSources
drwxr-xr-x 5 root  root    4096 Jul 16 14:07 X11
-rw-r--r-- 1 root  root    2981 Aug 21  2012 adduser.conf
-rw-r--r-- 1 root  root      10 Jul 13 23:19 adjtime
drwxr-xr-x 2 root  root    4096 Jul 31 00:56 ajaxplorer
-rw-r--r-- 1 root  root     233 Aug 21  2012 aliases
-rw-r--r-- 1 root  root   12288 Aug  4 01:59 aliases.db
drwxr-xr-x 2 root  root    4096 Aug 12 12:02 alternatives
drwxr-xr-x 7 root  root    4096 Jul 19 04:50 apache2
drwxr-xr-x 2 root  root    4096 Jul 13 23:27 apparmor.d
drwxr-xr-x 6 root  root    4096 Jul 31 00:55 apt
```

- **„-h“ – používa sa spoločne s prepínačom –l a zobrazuje veľkosti súborov/adresárov v čitateľnej podobe**

```
root@Linux:/etc/default# ls -lh /etc

total 872K
drwxr-xr-x 2 root  root   4.0K Mar 25  2010 ODBCDataSources
drwxr-xr-x 5 root  root   4.0K Jul 16 14:07 X11
-rw-r--r-- 1 root  root   3.0K Aug 21  2012 adduser.conf
-rw-r--r-- 1 root  root     10 Jul 13 23:19 adjtime
drwxr-xr-x 2 root  root   4.0K Jul 31 00:56 ajaxplorer
-rw-r--r-- 1 root  root    233 Aug 21  2012 aliases
-rw-r--r-- 1 root  root    12K Aug  4 01:59 aliases.db
```

```
drwxr-xr-x 2 root  root   4.0K Aug 12 12:02 alternatives
drwxr-xr-x 7 root  root   4.0K Jul 19 04:50 apache2
drwxr-xr-x 2 root  root   4.0K Jul 13 23:27 apparmor.d
drwxr-xr-x 6 root  root   4.0K Jul 31 00:55 apt
```

Vo výstupe ***ls***, jednotlivé stĺpce majú nasledujúci význam:

- typ súboru - („-„ súbor, „d“ – adresár, „l“ – symbolická linka),
- oprávnenia k súboru - r – read, w – write, x –executable (bližšie sa oprávneniam venujeme v 4. kapitole),
- počet odkazov k súboru,
- vlastník súboru (adresára, symbolickej linky),
- skupina viažúca sa k súboru (adresára, symbolickej linky),
- veľkosť (v bytoch),
- dátum a čas poslednej zmeny,
- názov súboru (adresára, symbolickej linky).

1.4.3. pwd (print working directory)

V operačnom systéme Linux je zavedený pojem **aktuálny adresár**. Pokiaľ chcete zistiť, aký je váš aktuálny adresár, použijete tento príkaz.

Syntax:

pwd

Príklad:

pwd

```
root@Linux:/etc/default# pwd
/etc/default
```

1.4.4. cd (changing directory)

Na **zmenu adresára** používame príkaz cd.

Syntax:

cd názov_adresára

Príklad:

cd /etc

```
root@Linux:/etc/default# pwd
/etc/default

root@Linux:/etc/default# cd /etc/
root@Linux:/etc# pwd
/etc
```

Ak sa chceme vrátiť o adresár vyššie, použijeme nasledujúci príkaz:

cd ..

1.4.5. cp (copy)

Na **kopírovanie súborov** používame príkaz cp. Tento nástroj má 2 režimy prevádzky:

1. Skopírovanie jedného alebo viac súborov do adresára

Syntax:

cp názov_súboru1 názov_súboru2 názov_súboru3 názov_cieľového_adresára

Príklad: Chceme skopírovať súbory **„jedalen1 jedalen2 jedalen3“** do adresára **„budova“** v adresári **„/etc“**

```
cp /home/user/jedalen1  /home/user/jedalen2  /home/user/jedalen3
/etc/budova
```

2. Skopírovanie obsahu súboru do nového súboru s novým názvom

Syntax:

cp názov_zdrojového_súboru názov_cieľového_súboru

Príklad: Chceme skopírovať súbor **„jedalen“** v adresári /home/user na **„budova“** v adresári /etc

```
cp /home/user/jedalen /etc/budova
```

Ak existujú viac ako dva parametre zadané nástrojom príkazom cp, potom **posledný parameter je vždy cieľový adresár.**

1.4.6. mv (move)

Na presunutie alebo premenovanie súboru/adresára používame príkaz mv.

Syntax:

```
mv názov_zdrojové_súboru cieľový_adresár
```

Príklad: Chceme premiestniť súbor subor.txt v adresári /etc do adresára /home

```
mv /etc/subor.txt /home
```

Ak presúvame ten istý súbor do toho istého adresára len s iným názvom docielime jeho premenovanie:

```
mv /etc/skuska.txt /etc/test.txt
```

1.4.7. mkdir (make directory)

Na vytvorenie adresára používame príkaz mkdir.

Syntax:

```
mkdir názov_adresára
```

Príklad: Chceme vytvoriť nový adresár s názvom test v adresári /home

```
mkdir /home/test
```

Príklad: Chceme vytvoriť niekoľko nových adresárov s názvami test2, test3 a test4 v adresári /home

```
mkdir /home/test2 /home/test3 /home/test4
```

V niektorých prípadoch je potrebné vytvoriť aj nadradené adresáre. Napríklad chceme vytvoriť adresár teta v adresári /home/test/rodina. Adresár rodina ale vytvorený nemáme. Podľa vyššie uvedeného by sme použili:

```
mkdir /home/test/rodina/teta
```

Vyššie uvedený príkaz vyhodí chybu, keďže neexistujú všetky adresáre v ceste.

```
root@Linux:/etc# mkdir /home/test/rodina/teta
mkdir: cannot create directory `/home/test/rodina/teta': No such file
or directory
```

Na takéto vytvorenie vnorených adresárov je potrebné použiť prepínač **–p** (parent-rodič). V nižšie uvedenom prípade sa vytvorí adresár teta a rodina.

```
mkdir -p /home/test/rodina/teta
```

1.4.8. rm (remove)

Na **vymazanie súborov** používame príkaz rm.

Syntax:

```
rm názov_suboru
```

Príklad: Chceme zmazať súbor skuska.txt v adresári /etc:

```
rm /etc/skuska.txt
```

Prepínače:

- **„-f“** – tento prepínač sa používa v prípade, ak nechceme odpovedať na otázky pri mazaní adresára

```
rm -f /etc/skuska.txt
```

1.4.9. rmdir (remove directory)

Na **mazanie adresára** používame príkaz rmdir.

Syntax:

```
rmdir názov_adresára
```

Príklad: Chceme zmazať adresár s názvom test v adresári /home

```
rmdir /home/test
```

Príklad: Chceme zmazať niekoľko adresárov v adresári /home s názvami test2, test3 a test4

```
rmdir /home/test2 /home/test3 /home/test4
```

Samotný príkaz rmdir odmietne odstrániť adresár, ktorý neexistuje, alebo ktorý nie je prázdny. Odstránenie neprázdneho adresára je možné prostredníctvom **nástroja rm** s použitím prepínača **–r** (recursive – rekurzívne):

```
rm -r /home/test
```

Pri mazaní podadresárov sa operačný systém Linux opýta, či chceme zmazať podadresáre, ktoré nie sú prázdne. Aby sme sa vyhli týmto otázkam, zadáme prepínač –f.

```
rm -rf /home/test
```

1.4.10. du

V prípade, ak chceme vedieť veľkosť niektorého súboru alebo celého adresára na disku, použijeme príkaz du.

Syntax:

```
du [prepínače] názov_súboru/názov_adresára
```

Príklad:

du /etc

```
root@Linux:/var/log# du /etc/
8       /etc/calendar
16      /etc/resolvconf/update-libc.d
20      /etc/resolvconf
12      /etc/apt/apt.conf.d
4       /etc/apt/sources.list.d
4       /etc/apt/trusted.gpg.d
4       /etc/apt/preferences.d
76      /etc/apt
12      /etc/ld.so.conf.d
```

Vyššie uvedený príklad zobrazí veľkosť jednotlivých adresárov v adresári /etc.

Prepínače:

- **„-s“** – zobrazí sumárnu veľkosť

```
root@Linux:/var/log# du -s /etc/
5172    /etc/
```

Vyššie uvedený príkaz s prepínačom –s zobrazí veľkosť adresára /etc

- **„-h“** – zobrazí veľkosť v čitateľnej podobe

```
root@Linux:/var/log# du -sh /etc/
5.1M    /etc/
```

Vyššie uvedený príkaz s prepínačmi –s a -h zobrazí veľkosť adresára /etc v čitateľnej podobe.

1.4.11. file

Príkaz **file** používame na zistenie typu súboru bez jeho samotného otvorenia.

Syntax:

file [prepínače] názov_súboru

Príklad:

file /etc/passwd

```
root@Linux:/home/user# file /etc/passwd
/etc/passwd: ASCII text
```

Príklad:

```
[root@Linux]# file *
```

Dostaneme opis všetkých súborov a podadresárov v danom adresári. Na tomto mieste je vhodné primenúť, že adresár je tiež určitá forma súboru. Z tohto dôvodu väčšina nástrojov (príkazov) pre súbory platí aj pre adresáre (okrem výnimiek ako mkdir, rmdir a pod.).

1.4.12. find

Na **vyhľadávanie súborov** v operačnom systéme Linuxe používame veľmi mocný príkaz find. Má mnoho rôznych volieb a kritérií, podľa ktorých vyhľadáva súbory a adresáre.

Syntax:

find [prepínače] výraz

Príklad: Chceme nájsť v celom koreňovom adresári súbory a adresáre, ktoré obsahujú slovo test

find test

```
root@Linux:/home/user# find test
test
test/Xsession.options
test/Xsession
test/Xwrapper.config
```

Prepínače:

Prepínač	Význam	Príklad
-name	vyhľadávanie na základe názvu súboru	find /etc -name 'ch*‘
-iname	case insensitívne vyhľadávanie názvu súboru	find /etc -iname '*pass*‘
-ctime	vyhľadávanie na základe času, kedy bol súbor vytvorený	find /etc -ctime +365
-mtime	vyhľadávanie na základe času, kedy bol súbor posledný krát modifikovaný	find /etc -mtime -l
-atime	vyhľadávanie na základe času, kedy bolo k súboru naposledy pristúpené	find /etc -atime +30
-group	vyhľadávanie na základe skupiny	find /etc -group root
-user	vyhľadávanie na základe používateľa	find /etc -user root
-perm	vyhľadávanie na základe oprávnení	find /etc -perm -777
-size	vyhľadávanie súborov na základe určitej veľkosti	find /etc -size +5000k
-type	vyhľadávanie súborov, adresárov, rúr, linkov	find /etc -type d

„-name“ – vyhľadávanie podľa **mena súboru**

Príklad: Predstavme si, že chceme vyhľadať súbor podľa mena zaloha.tgz, ale nevieme, kde sa môže nachádzať, takže radšej začneme hneď od koreňa. Výsledok chceme vypísať na obrazovku. Zadáme teda príkaz:

```
[root@Linux]#  find / -name zaloha.tgz -print
```

Keď find nájde viac súborov s tým istým menom, vypíše ich všetky. Výpis spôsobí práve parameter –print.

„-size“ – vyhľadávanie podľa **veľkosti súboru**

```
root@Linux:/home/user# find /var/log/ -size +4M
```

Vo vyššie uvedom príklade sme chceli vyhľadať v adresári /var/log súbory, ktorých veľkosť presahuje 4MB.

```
root@Linux:/home/user# find /var/log/ -size -4M
```

Vo vyššie uvedom príklade sme chceli vyhľadať v adresári /var/log súbory, ktorých veľkosť nepresahuje 4MB.

„-exec“ – vyhľadanie a **vykonanie akcie**

Keď príkaz find nájde súbor, predvolená akcia zobrazí meno súboru. Alternatívna akcia, ktorá je veľmi užitočná, je ***find –exec***

- príkaz find spustí zadaný program zasielaním názvu súboru na neho.
- príkaz find nahradí názov nástroja (príkazu) na mieste **špeciálnych znakov *{}***.

Príklad: Chceme nájsť v adresári /etc súbory (-type f)a následne ich zobraziť na terminál.

```
$ find /etc -type f -exec cat {} \;
```

Príklad: Chceme nájsť v adresári /bin s oprávneniami (775) a následne zozbaziť informácie o nich (ls –la)

```
Linux:~ $ find /bin -perm -755 -exec ls -l {} \;

-rwxr-xr-x 1 root root 35302 Jun 23 2002 /bin/ping
-rwxr-xr-x 1 root root 81996 Aug 30 2002 /bin/mount
-rwxr-xr-x 1 root root 40700 Aug 30 2002 /bin/umount
-rwxr-xr-x 1 root root 19132 Aug 29 2002 /bin/su
```

1.4.13. locate

Druhou možnosťou, ako nájsť súbor, je použitie nástroja (príkazu) **locate**. Ten však pracuje na inom princípe ako príkaz find. Locate nevyhľadáva súbor skutočne na disku, ale vo svojej databáze, kde má zapísané umiestnenie všetkých súborov. Keďže nepotrebuje prehľadávať všetky adresáre, ale stačí mu len pozrieť do databázy, dosahuje rýchlejšie výsledky ako find. Nevýhodou je, že tá databáza nemusí byť vždy aktuálna.

Najprv, ako použijete tento príkaz, musíte zaktualizovať databázu:

```
updatedb
```

Syntax:

```
locate [prepínače] reťazec
```

Príklad: Chceme nájsť v celom operačnom systéme všetky súbory, ktoré majú príponu png.

```
root@Linux:/home/user# locate "*.png"
/home/user/car.png
/home/user/flag.png
/home/user/flag_france.png
/home/test/admin/refresh.png
/home/test/admin/admin/refresh.png
```

Príklad: Chceme nájsť v celom operačnom systéme všetky súbory, ktoré obsahujú slovo test.

```
root@Linux:/home/user# locate "*test*"
/home/test
/home/test2
/home/user/test
/home/user/test/Xsession
/home/user/test/Xsession.options
/home/user/test/Xwrapper.config
/home/test/.bash_history
/home/test/.bash_logout
/home/test/.bashrc
/home/test/.profile
/home/test2/.bash_logout
/home/test2/.bashrc
/home/test2/.profile
```

Hľadaný výraz je v úvodzovkách. Hviezdičky (*) nahrádzajú rôzny počet znakov. Napr. *test* = test, test123, ahojtest, 12test20, testujem, podtest

Prepínače:

- **„-c“** – zobrazí počet výsledkov vyhľadávania

```
root@Linux:/home/user# locate -c "*.png"
7734
```

Vyššie uvedený príkaz locate s prepínačom –c zobrazí počet všetkých súborov s príponou png.

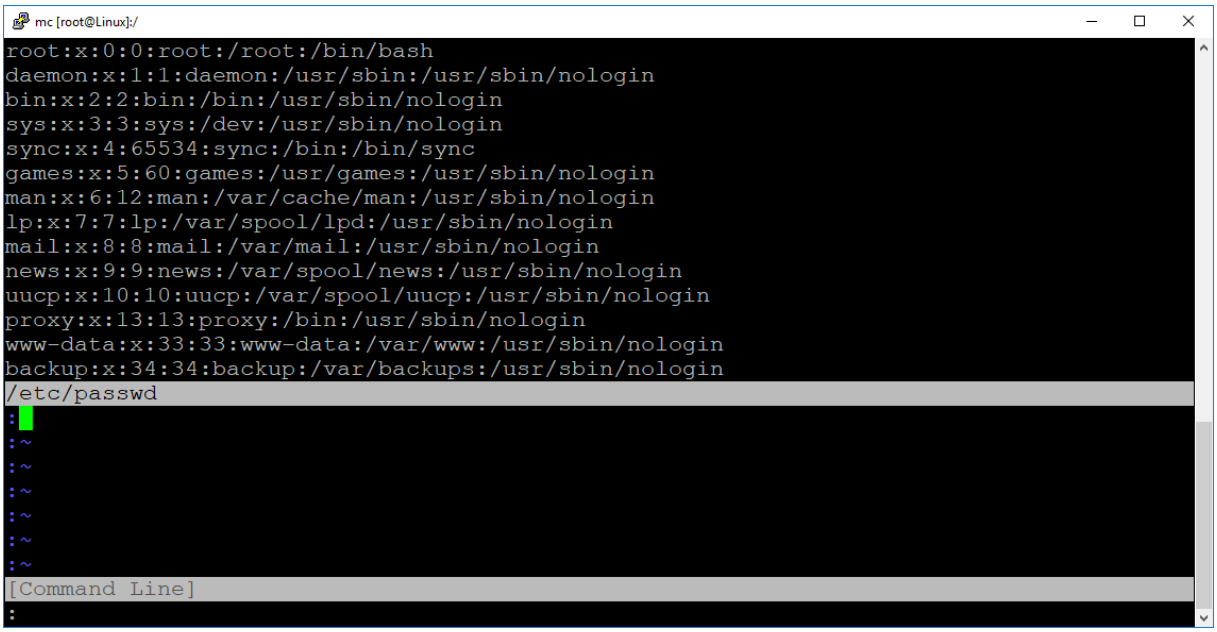
1.5. Úlohy k súborovému systému

- **Namiesto domovsky_adresar doplňte svoj domovský adresár**
- **Ku každej úlohe napíšte úplny príkaz vrátane parametrov a prepínačov**

2. Textové editory a práca s textom

2.1. Editor vi

Editor vi - je obrazovkový textový editor napísaný Billom Joyom v roku 1976 pre operačný systém BSD. Z pohľadu spôsobu používania, pre začínajúcich používateľov odporúčame editor nano.



Obr. 5 Ukážka editora vi.

2.2. Editor nano

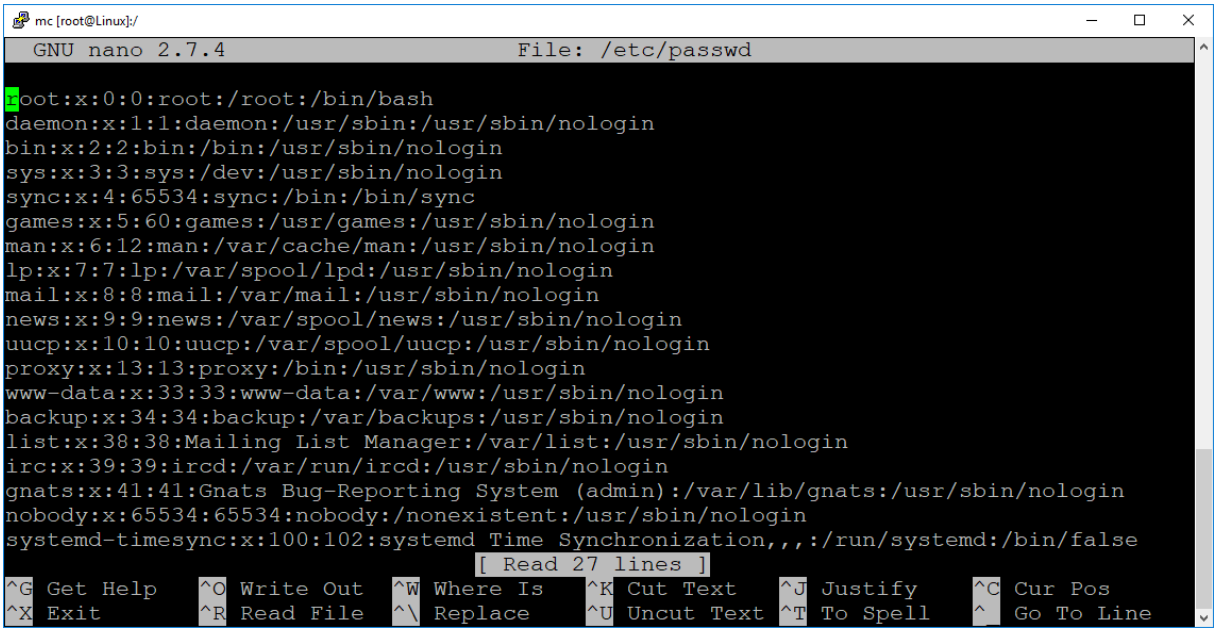
Inštalácia:

- **Debian, Ubuntu:**

```
apt-get install nano
```

- **CentOS:**

```
yum install nano
```



Obr. 6 Ukážka editora nano.

Klávesová skratka	Význam
CTRL+G	Pomoc
CTRL+O	Uloženie údajov
CTRL+X	Ukončenie nano / ukončenie nápovedy
CTRL+W	Vyhľadávanie reťazca (ukončenie vyhľadávania CTRL+c)
CTRL+K	Vystrihnuie 1 riadku
CTRL+U	Vloženie vystrihnuté riadku

2.3. Nástroje (príkazy) na prácu so súbormi

2.3.1. cat

V praxi veľmi často potrebujeme **vypísať obsah súboru** (na obrazovku). Na to je vhodný príkaz cat.

Syntax:

```
cat [prepínače] názov_súboru
```

Príklad:

```
root@Linux:/home/user# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
```

Prepínače:

- **„-n“** – zobrazí aj poradie riadku

```
root@Linux:/home/user# cat -n /etc/passwd
     1  root:x:0:0:root:/root:/bin/bash
     2  daemon:x:1:1:daemon:/usr/sbin:/bin/sh
     3  bin:x:2:2:bin:/bin:/bin/sh
     4  sys:x:3:3:sys:/dev:/bin/sh
     5  sync:x:4:65534:sync:/bin:/bin/sync
     6  games:x:5:60:games:/usr/games:/bin/sh
```

2.3.2. tac

Ak chceme **vypísať súbor v obrátenom poradí**, teda od konca súboru až na začiatok, použijeme nástrok (príkaz) tac.

Syntax:

tac názov_súboru

Príklad:

```
root@Linux:/home/user# tac /etc/passwd
postfix:x:108:112::/var/spool/postfix:/bin/false
messagebus:x:107:111::/var/run/dbus:/bin/false
mysql:x:106:110:MySQL Server,,,:/var/lib/mysql:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
fetchmail:x:104:65534::/var/lib/fetchmail:/bin/false
```

2.3.3. echo

Ak chceme **vypísať text na terminál**, použijeme príkaz echo.

Syntax:

echo text

Príklad:

```
root@Linux:/var/log# echo "Moj prvy text v Linuxe"
Moj prvy text v Linuxe
```

Príkaz echo môžeme použiť aj na zapísanie určitého textu do súboru (použitím presmerovania >):

```
root@Linux:/# echo "Ahoj" > /home/user/adresar.txt
root@Linux:/# cat /home/user/adresar.txt

Ahoj
```

2.3.4. more

Vo väčšine prípadov je zobrazený súbor dlhší, ako je počet riadkov terminála (obrazovky). Takto sa začiatok súboru na určitý čas zobrazí na obrazovke, a používateľ si môže prezerať iba jeho posledné riadky. Aby sa zabránilo scrolovaniu obrazovky, je možné použiť príkaz more. Jeho úlohou je zobraziť súbor po jednotlivých obrazovkách – akoby stránkach. Prečítanú časť posunieme ďalej stlačením medzerníka. Obdobná funkcia funguje aj vo webových prehliadačoch.

Syntax:

more názov_súboru

Príklad:

```
root@Linux:/home/user# more /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
--More--(72%)
```

2.3.5. less

Less neznamená v tomto prípade menej, ale je to zlepšený variant nástroj (príkazu) more. Less zobryzuje obsah po 1 riadku, čo umožňuje posúvanie sa v texte smerom na začiatok, resp. koniec súboru.

Syntax:

less názov_súboru

Príklad:

```
root@Linux:/home/user# less /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
```

Tento príkaz vypíše obsah na terminál (obrazovku), ale má komfortnejšie ovládanie. Môžeme nielen posúvať výpis dopredu, ale aj naspäť. Stačí, ak použijeme klávesy PageUp, PageDown, kurzorové šípky, Enter a samozrejme medzerník.

2.3.6. head

Príkaz head **vypíše prvé časti súboru.** Bez uvedenia prepínača zobrazí prvých 10 riadkov súbora.

Syntax:

head [prepínače] názov_súboru

Príklad:

```
root@Linux:/home/user# head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
```

```
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
```

Prepínače:

- **„-n“** – počet prvých riadkov

```
root@Linux:/home/user# head -n 3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
```

2.3.7. tail

Príkaz tail vypíše **posledné časti súboru**. Bez uvedenia prepínača zobrazí posledných 10 riadkov súbora.

Syntax:

tail [prepínače] názov_súboru

Príklad:

```
root@Linux:/home/user# tail /etc/passwd
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
smmta:x:101:103:Mail Transfer Agent,,,:/var/lib/sendmail:/bin/false
smmsp:x:102:104:Mail Submission
Program,,,:/var/lib/sendmail:/bin/false
bind:x:103:107::/var/cache/bind:/bin/false
fetchmail:x:104:65534::/var/lib/fetchmail:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
mysql:x:106:110:MySQL Server,,,:/var/lib/mysql:/bin/false
messagebus:x:107:111::/var/run/dbus:/bin/false
postfix:x:108:112::/var/spool/postfix:/bin/false
```

Prepínače:

- **„-n“** – počet posledných riadkov

```
root@Linux:/home/user# tail -n 4 /etc/passwd
mysql:x:106:110:MySQL Server,,,:/var/lib/mysql:/bin/false
messagebus:x:107:111::/var/run/dbus:/bin/false
postfix:x:108:112::/var/spool/postfix:/bin/false
root:x:0:0:root:/root:/bin/bash
```

2.3.8. cut

Ak potrebujeme oddeliť niektoré „stĺpce“ z textového súboru, použijeme príkaz cut.

Syntax:

cut [prepínače] názov_súboru

Príklad: Chceme zobraziť prvý a tretí stĺpec súboru /etc/passwd. Na tomto mieste je potrebné si uvedomiť, že jednotlivé údaje v súbore sú oddelené „:“.

```
root@Linux:/var/log# cut -d ":" -f "1,3" /etc/passwd
root:0
daemon:1
bin:2
sys:3
sync:4
games:5
man:6
lp:7
mail:8
news:9
```

Prepínače:

- **„-d“** – definovanie oddeľovača v súbore. Najčastejšie je to medzera, čiarka, bodkočiarka alebo dvojbodka (napr. v /etc/passwd)
- **„-f“** – určenie, ktoré stĺpce sa majú vypísať

2.3.9. grep

Príkaz grep (Global Regular Expression Printer) **vyhľadáva zadaný reťazec znakov** v príslušnom textovom súbore. Ak tento reťazec nájde, vypíše na terminál (obrazovku) riadok, kde sa reťazec v súbore nachádza.

Syntax:

grep [prepínače] vzor názov_súboru

Príklad: V súbore /etc/passwd chceme nájsť riadky, ktorý obsahujú reťazec „bin“.

```
root@Linux:/home/user# grep bin /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
```

Prepínače:

- **„-v“** –použijeme v prípade, ak chceme vybrať všetky riadky, ktoré **neobsahujú hľadaný výraz**

```
root@Linux:/home/user# grep -v home /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
```

Vyššie uvedený príklad príkazu nám zobrazí tie riadky súboru /etc/passwd, ktoré neobsahujú slovo home.

2.3.10. wc

Pomocou príkazu wc vieme **vypísať počet riadkov, slov, bytov** pre každý súbor.

Syntax:

wc [prepínače] názov_súboru

Príklad: Cheme zobraziť informácie o súbore /etc/passwd (počet riadkov, slov a bytov)

```
root@Linux:/home/user# wc /etc/passwd
  31   41 1375 /etc/passwd
```

Tento príkaz nám zobrazí 31 riadkov, 41 slov a 1375 bytov.

Prepínače:

- **„-c“** – zobrazí počet bytov v súbore

```
root@Linux:/home/user# wc -c /etc/passwd
     1375 /etc/passwd
```

- **„-m“** – zobrazí počet znakov v súbore

```
root@Linux:/home/user# wc -m /etc/passwd
     1375 /etc/passwd
```

- **„-l“** – zobrazí počet riadkov v súbore

```
root@Linux:/home/user# wc -l /etc/passwd
     31 /etc/passwd
```

- **„-w“** – zobrazí počet slov v súbore

```
root@Linux:/home/user# wc -w /etc/passwd
     41 /etc/passwd
```

2.3.11. sort

Na **zotriedenie riadkov v textovom súbore** používame príkaz sort.

Syntax:

sort [prepínače] názov_súboru

Príklad: Chceme zotriediť riadky v súbore /etc/passwd abecedne podľa prvého znaku v riadku.

```
root@Linux:/home/user# sort /etc/passwd
backup:x:34:34:backup:/var/backups:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
bind:x:103:107::/var/cache/bind:/bin/false
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
fetchmail:x:104:65534::/var/lib/fetchmail:/bin/false
games:x:5:60:games:/usr/games:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
```

Prepínače:

- **„-n“** – zotriedi podľa numerických hodnôt

```
root@Linux:/home/user# cat /etc/passwd | cut -d ":" -f 3 | sort -n
0
1
2
3
4
5
6
7
8
9
10
```

Vyššie uvedený príklad použitia príkazu je možné rozdeliť do 3 samostatných príkazov cat, cut a sort. Príkaz cat /etc/passwd vypíše obsah súboru, ale keďže tu je rúra „|“ tak namiesto vypísania na terminál (obrazovku) pošle obsah na vstup ďalšieho príkazu. Ďalší príkaz cut zobrazí vybrané stĺpce. Tento príkaz má na vstupe obsah súboru /etc/passwd. Pomocou prepínačov –d „:“ a –f 3 vyberie 3.sStĺpec. Keďže aj tu nasleduje rúra „|“, tak výstup sa pošle na vstup ďalšieho príkazu, ktorým je sort. Tento príkaz zotriedi vstup (prepínač –n poredpokladá na vstupe všetky čísla).

- **„-r“** – otočí výsledky triedenia

```
root@Linux:/home/user# sort -r /etc/passwd
www-data:x:33:33:www-data:/var/www:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
```

```
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
smmta:x:101:103:Mail Transfer Agent,,,:/var/lib/sendmail:/bin/false
smmsp:x:102:104:Mail Submission
Program,,,:/var/lib/sendmail:/bin/false
root:x:0:0:root:/root:/bin/bash
```

2.3.12. tr

Pomocou príkazu **tr** vieme **preložiť (transformovať) alebo zmazať znaky.**

Syntax:

tr [prepínače] „množina1“ „množina2“

Príklad: Chceme zmeniť všetky písmená v súbore text.txt na veľké písmená alebo zmeniť písmeno a na znak *.

```
root@Linux:/home/user# cat text.txt
Mam rad Linux. Ale najviac lubim najlepsiu osobku na svete :)
root@Linux:/home/user# cat text.txt | tr "a-z" "A-Z"
MAM RAD Linux. ALE NAJVIAC LUBIM NAJLEPSIU OSOBKU NA SVETE :)
root@Linux:/home/user# cat text.txt | tr "a" "*"
M*m r*d Linux. Ale n*jvi*c lubim n*jlepsiu osobku n* svete :)
```

3. Správa používateľov a oprávnení

3.1. Typy používateľov

Administrátor

- Označuje sa ako **superadministrátor, root**
- je používateľské meno alebo účet, ktorý v predvolenom nastavení **má prístup ku všetkým príkazom a súborom** na Linuxe.
- jeho **UID je 0**
- dokonca **vie odstrániť koreňový adresár „/“**

Užívatelia (users):

- každý iný používateľ
- ich **UID je rôzne od 0**
- obmedzené oprávnenia
- nutnosť používať sudo

Označenie používateľa v príkazovom interpréterovi:

- **#** - superpoužívateľ (root)
- **$** - obyčajný používateľ

3.2. Základné súbory používateľov

3.2.1. /etc/passwd

V súbore **/etc/passwd** je uložená **databáza používateľov.**

Príklad súboru /etc/passwd:

```
root:x:0:0::/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
```

```
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:
ftp:x:14:50::/home/ftp:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
nobody:x:99:99:nobody:/:
janko:x:1000:100:Janko Hrasko,,,:/home/janko:/bin/bash
```

Jednotlivé údaje o používateľoch (stĺpce) sú oddelené dvojbodkou.

Záznam (stĺpec)	Obsah
1.	Názov používateľa
2.	Heslo používateľa – x znamená, že heslo je uložené v /etc/shadow
3.	Identifikácia používateľa
4.	Identifikácia skupiny
5.	Informácie o používateľovi
6.	Domovský adresár
7.	Príkazový interpréter (Shell)

3.2.2. /etc/shadow

V súbore **/etc/shadow** sú uložené **heslá používateľov.**

Príklad súboru /etc/shadow:

```
root:nhC.YP4s8lF1Y:11783:0:::::
bin:*:9797:0:::::
daemon:*:9797:0:::::
adm:*:9797:0:::::
lp:*:9797:0:::::
sync:*:9797:0:::::
shutdown:*:9797:0:::::
halt:*:9797:0:::::
mail:*:9797:0:::::
news:*:9797:0:::::
uucp:*:9797:0:::::
operator:*:9797:0:::::
games:*:9797:0:::::
ftp:*:9797:0:::::
```

```
mysql:*:9797:0:::::
gdm:*:9797:0:::::
nobody:*:9797:0:::::
janko:in9.jjl2XgsXQ:11783:0:99999:7:::
```

Jednotlivé údaje o heslách používateľov (stĺpce) sú oddelené dvojbodkou.

Záznam (stĺpec)	Obsah
1.	Názov používateľa
2.	Heslo v zašifrovanej podobe
3.	Posledná zmena hesla počítaných od 1.1.1970
4.	Minimálny počet dní vyžadovaných pri zmenách hesla
5.	Maximálny počet dní, kedy je heslo platné
6.	Počet dní do exspirácie hesla
7.	Počet dní po exspirácii hesla
8.	Počet dní expirácie hesla počítaných od 1.1.1970

3.2.3. /etc/groups

V súbore **/etc/groups** je uložená **databáza skupín.**

Príklad súboru /etc/groups:

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
mem::8:
kmem::9:
wheel::10:root
floppy::11:root,jerry
mail::12:mail
news::13:news
uucp::14:uucp,jerry
man::15:
games::20:
slocate:x:21:
mysql::27:
gdm::42:
ftp::50:
nobody::98:nobody
```

Jednotlivé údaje o skupine (stĺpce) sú oddelené dvojbodkou.

Záznam (stĺpec)	Obsah
1.	Názov skupiny
2.	Heslo skupiny (štandardne nie je nastavené)
3.	Identifikačné číslo skupiny
4.	Zoznam používateľov v danej skupine

3.3. Príkazy na prácu s používateľmi

3.3.1. adduser

Pomocou príkazu adduser **pridávame používateľa.**

Syntax:

adduser meno_používateľa

Príklad:

```
root@Linux:/# adduser test
Adding user `test' ...
Adding new group `test' (1004) ...
Adding new user `test' (1004) with group `test' ...
Creating home directory `/home/test' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

3.3.2. useradd

Pomocou príkazu useradd **pridávame používateľa.** Rozdiel medzi príkazmi adduser a useradd spočíva v:

- adduser vytvára aj heslo pre používateľa,
- adduser pri pridávaní používateľa umožňuje vyplniť údaje
- adduser ako shell pridáva bash
- useradd dáva ako shell sh

Syntax:

useradd meno_používateľa

3.3.3. deluser

Pomocou príkazu deluser **zmažeme používateľa.**

Syntax:

deluser meno_používateľa

Príklad:

```
root@Linux:/# deluser test
Removing user `test' ...
Warning: group `test' has no more members.
Done.
```

3.3.4. userdel

Pomocou príkazu deluser **zmažeme používateľa a obsah jeho domovského adresára.**

Syntax:

deluser meno_používateľa

3.3.5. usermod

Na zmenu údajov používateľa používame príkaz usermod.

Syntax:

usermod [prepínače] meno_používateľa

Prepínače:

- **„-c“** – zmena komentára

```
root@Linux:/home/user# cat /etc/passwd | grep test
test:x:1004:1004::/home/test:/bin/bash
root@Linux:/home/user# usermod -c "Moje konto" test
root@Linux:/home/user# cat /etc/passwd | grep test
test:x:1004:1004:Moje konto:/home/test:/bin/bash
```

- **„-g“** – nastavenie skupiny pre používateľa

```
root@Linux:/home/user# cat /etc/passwd | grep test
test:x:1004:1004:Moje konto:/home/test:/bin/bash
root@Linux:/home/user# usermod -g www-data test
root@Linux:/home/user# cat /etc/passwd | grep test
test:x:1004:33:Moje konto:/home/test:/bin/bash
```

- **„-l“** – zmena prihlasovacieho mena používateľa

```
root@Linux:/home/user# cat /etc/passwd | grep test
test:x:1004:33:Moje konto:/home/test:/bin/bash
root@Linux:/home/user# usermod -l novy_login test
root@Linux:/home/user# cat /etc/passwd | grep test
novy_login:x:1004:33:Moje konto:/home/test:/bin/bash
```

- **„-s“** – zmena shellu

```
root@Linux:/home/user# cat /etc/passwd | grep test
test:x:1004:33:Moje konto:/home/test:/bin/bash
root@Linux:/home/user# usermod -s bin/false test
roo@Linux:/home/user# cat /etc/passwd | grep test
test:x:1004:33:Moje konto:/home/test:bin/false
```

- **„-d“** – zmena domovského adresára používateľa

```
root@Linux:/home/user# cat /etc/passwd | grep test
test:x:1004:33:Moje konto:/home/test:/bin/bash
root@Linux:/home/user# usermod -d /home/pes test
root@Linux:/home/user# cat /etc/passwd | grep test
test:x:1004:33:Moje konto:/home/pes:bin/bash
```

3.3.6. passwd

Pomocou príkazu passwd **zmeníme používateľovi heslo.**

Syntax:

passwd [prepínače] meno_používateľa

Príklad: Chceme zmeniť heslo používateľovi test

```
root@Linux:/# passwd test
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Prepínače:

- **„-d“** – zmaže heslo

```
root@Linux:/home/user# passwd -d test
passwd: password expiry information changed.
```

- **„-S“** – status hesla

```
root@Linux:/home/user# passwd -S test
test P 08/24/2013 0 99999 7 -1
```

3.4. Prístupové práva

Každý súbor v operačnom systéme Linux má nastavené tzv. **prístupové práva.** Prístupové práva pre súbory môžeme deliť z pohľadu oprávnených činností k súboru a z pohľadu používateľa, ktorý danú činnosť vykonáva. Podľa oprávnených činností rozoznávame nasledujúce oprávnenia:

- **právo čítať (read)** – v prípade súborov znamená, že máme oprávnenie zobraziť obsah súboru (napr. cat /etc/passwd). V prípade adresárov ide o oprávnenie zobraziť obsah adresára – prezrieť názvy súborov v danom adresári (napr. ls /etc)
- **právo zapisovať (write)** – v prípade súborov znamená, že máme oprávnenie modifikovať obsah súboru. V prípade adresárov ide o oprávnenie vytvoriť, premenovať, resp. zmazať súbor v rámci adresára.
- **právo na spustenie (execute)** – v prípade súborov znamená oprávnenie spustiť program (napr. skript napísaný v programovacom jazyku Python). V prípade adresárov ide o oprávnenie pristúpiť k súborom v danom adresári a pozrieť ich metaúdaje (napr. čas vytvorenia, veľkosť, oprávnenia a pod.)

Z pohľadu používateľa, ktorý môže vyššie uvedenú činnosť vykonávať, rozlišujeme:

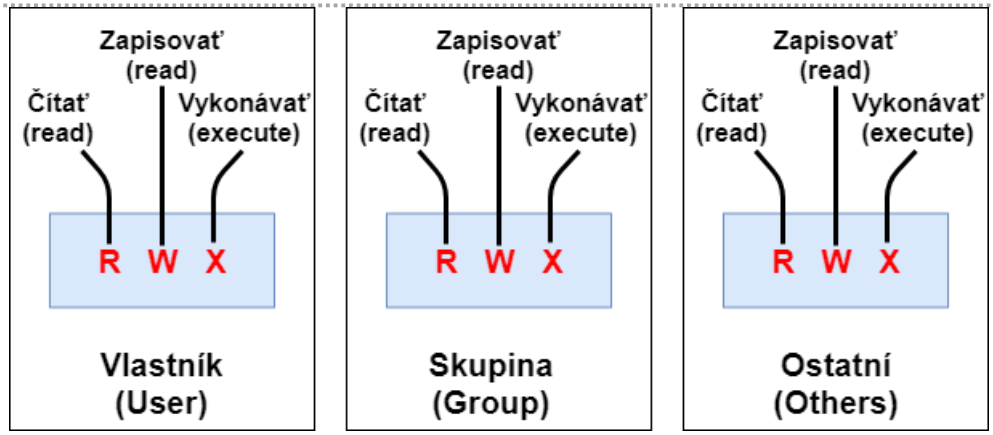
- **vlastníka (owner) -** používateľ, ktorý daný súbor/adresár vytvoril,
- **skupinu (group)** – skupina používateľov okrem vlastníka,
- **ostatných (other)** – ostatní používatelia operačného systému.

Na obrázku (Obr. 4.1) je zobrazený výstup nástroja **ls** (s prepínačom l, ktorý znamená dlhý výpis). V rámci tohto výstupu je možné vidieť oprávnenia k súborom, resp. adresárom. Prvý údaj nás informuje o type súboru. Najčastejšie pôjde o súbor (označenie „-„), adresár (označenie „d“) alebo symbolickú linku (označenie „l“). Symbolické linky vo väčšine prípadov predstavujú odkaz na iný súbor (podobne ako odkazy v operačnom systéme Windows na rôzne aplikácie). Ďalších 9 údajov označuje oprávnenia k súboru vzhľadom na používateľov.

```
root@Linux:/home/soki# cd /
root@Linux:/# ls -l
total 76
drwxr-xr-x   2 root root  4096 Jun 28 09:33 bin
drwxr-xr-x   3 root root  4096 Nov  3  2017 boot
drwxr-xr-x  19 root root  3060 Aug 26 20:08 dev
drwxr-xr-x  89 root root  4096 Dec 25 10:51 etc
drwxr-xr-x   5 root root  4096 Dec 25 01:35 home
lrwxrwxrwx   1 root root    29 Nov  3  2017 initrd.img -> boot/initrd.img-4.9.0-
4-amd64
lrwxrwxrwx   1 root root    29 Sep 22  2017 initrd.img.old -> boot/initrd.img-4.
9.0-3-amd64
drwxr-xr-x  15 root root  4096 Nov  3  2017 lib
drwxr-xr-x   2 root root  4096 Sep 22  2017 lib64
drwx------   2 root root 16384 Sep 22  2017 lost+found
drwxr-xr-x   3 root root  4096 Sep 22  2017 media
drwxr-xr-x   2 root root  4096 Sep 22  2017 mnt
drwxr-xr-x   2 root root  4096 Sep 22  2017 opt
dr-xr-xr-x 114 root root     0 Aug 26 20:08 proc
drwx------  10 root root  4096 Dec 25 07:05 root
drwxr-xr-x  17 root root   540 Dec 25 11:39 run
drwxr-xr-x   2 root root  4096 Jun 28 09:33 sbin
drwxr-xr-x   2 root root  4096 Sep 22  2017 srv
dr-xr-xr-x  13 root root     0 Dec 25 04:12 sys
```

Obrázok 4.1
Príklad výstupu nástroja ls.

Tieto údaje sú rozdelené do troch trojíc (Obr. 4.2). Každá trojica prislúcha postupne vlastníkovi, skupine a ostatným používateľom. Na základe týchto údajov, vieme presne určiť, ktorý používateľ v rámci systému má aké oprávnenia k danému súboru, resp. adresáru. Ak dané oprávnenie existuje, vo výpise vidíme jeho zástupný znak (r, w, x). Ak takéto oprávnenie nie je, vo výpise bude pomlčka („-“). Ak používateľ má oprávnenie čítať obsah súboru a súbor spustiť, ale nemá oprávnenie meniť jeho obsah, tieto oprávnenia budú charakterizované trojicou: r – x. Pomlčka v strede znamená, že oprávnenie na zápis (x) sa v danom prípade neuplatňuje.



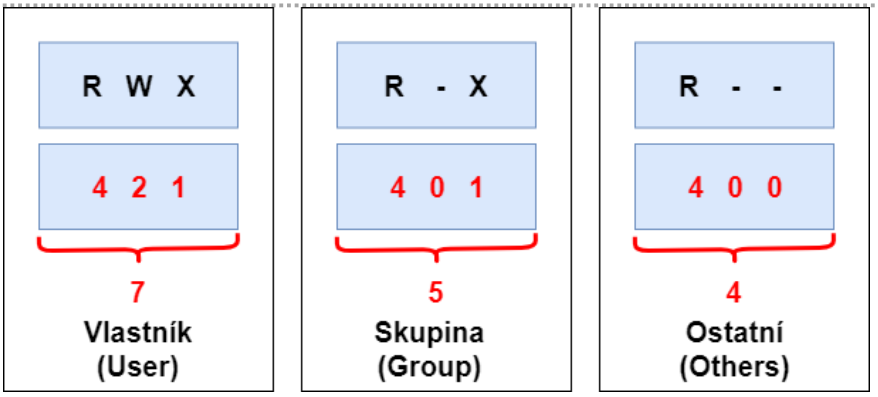
Obrázok 4.2
Schematické znázornenie práv k súborom a adresárom v operačnom systéme Linux.

Okrem vyššie uvedeného zápisu oprávnení sa v operačnom systéme Linux používa aj tzv. **oktálový zápis**. Ide o zápis oprávnení v osmičkovej sústave (0-7). Oktálový zápis obsahuje trojicu čísel. Prvé číslo prislúcha vlastníkovi, druhé skupine a napokon tretie ostatným používateľom. Jednotlivé oprávnenia majú pridelené konkrétne hodnoty čísel:

- „-„ –> 0 (bez oprávnenia)
- x –> $2^0 = 1$
- w –> $2^1 = 2$
- r –> $2^2 = 4$

Oprávnenia súboru, ktoré majú označenie **- rwx r-x r--** môžeme zapísať 754 v oktálovom zápise (Obr. 4.3) a interpretovať nasledujúcim spôsobom:

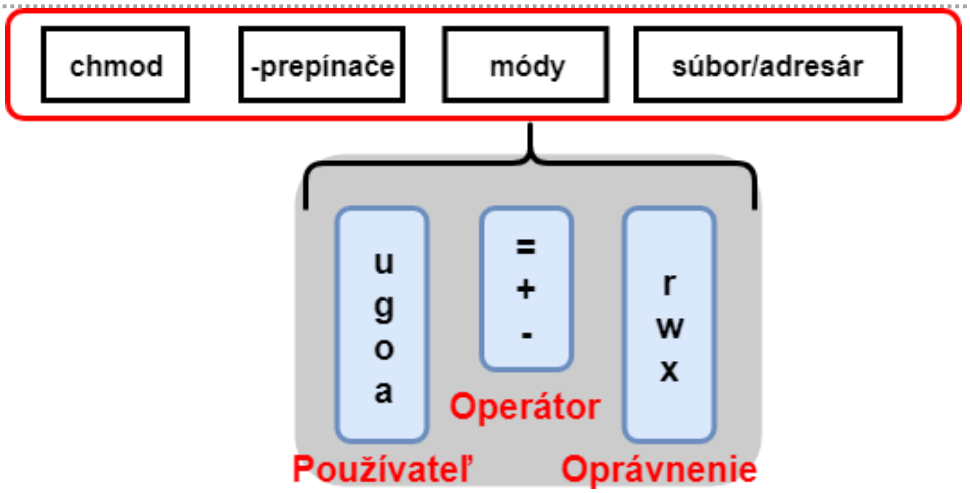
- ide o súbor (-)
- rwx - vlastník má právo čítať (r), zapisovať (w) a spúšťať daný súbor (x). Súčet oktálových hodnôt je 4 + 2 + 1 = 7.
- r-x - členovia skupiny majú právo čítať (r) a spúšťať (x) daný súbor. Súčet oktálových hodnôt je 4 + 1 = 5.
- r-x – ostatní používatelia majú právo len čítať (r) daný súbor. Súčet oktálových hodnôt je 4.



Obrázok 4.3
Príklad prevodu schematického znázornenia oprávnení na oktálový tvar.

3.4.1. chmod (change mode)

Na zmenu oprávnení k súborom a adresárom používame nástroj **chmod** (change mode). Tento nástroj dokáže pracovať so symbolickým označením oprávnení (r w x) alebo s oktálovým zápisom (Obr. 4.3).



Obrázok 4.4
Schematické znázornenie nástroja chmod.

Nástroj chmod vyžaduje pre správne fungovanie okrem zadania samotného súboru (súborov, adresáru alebo adresárov) aj zadanie konkrétnych oprávnení, resp. oprávnení, ktoré sa majú pridať alebo odobrať.

Na Obr. 4.4 je znázornené použitie tohto nástroja pri využití schematického označenia oprávnení. Ako prvé uvádzame, ktorých používateľov sa daná zmena dotýka (u – user, g – group, o –other, a – all). Následne uvedieme operátor (= presné práva, + pridanie práv, - odobranie práv). Ako posledné uvedieme označenie oprávnení (r,w,x).

Ak napríklad chceme zmeniť oprávnenia k súboru ita v adresári /home, tak, aby vlastník mal všetky oprávnenia, členovia skupiny mohli len čítať súbor a ostatní používatelia systému nemali žiadne oprávnenia, tak to môžeme vykonať nasledujúcimi príkazmi:

```
chmod 740 /home/ita
chmod u=rwx,g=r /home/ita
```

Pri použití schematického označenia by sme mali vedieť, aké oprávnenia daný súbor, resp. adresár má. Štandardne sa v operačnom systéme Linux vytvárajú súbory s oprávneniami zodpovedajúcimi 644 a adresáre s oprávneniami zodpovedajúcimi 755.

Nástroj chmod, obdobne ako väčšina nástroj v operačnom systéme Linux, umožňuje použitie prepínačov (options) na rozšírenie svojej funkcionality. Najčastejšie používaným prepínačom je **„-R“**, ktorý umožňuje vykonávať zmenu oprávnení rekurzívne. Ak chceme zmeniť oprávnenia adresára a jeho všetkých podadresárov a súborov, zadáme tento prepínač (napr. chmod 660 /home -R)

3.4.2. chown (change owner) a chgrp (change group)

Okrem vyššie uvedeného nástroje, vieme použiť ešte 2 užitočné nástroje. Prvý je nástroj **chown**, pomocou ktorého meníme vlastníka súboru, resp. adresára (napr. chown user /home). Druhým nástrojom je **chgrp**, ktorým dokážeme zmeniť skupinu používateľov, pre daný súbor, resp. adresár (napr. chgrp users /home). Obdobne ako pri nástroji chmod, aj pri týchto je možné používať prepínače (options), najmä však prepínač –R pre rekurzívnu zmenu vlastníka, resp. skupiny.

Na **zmenu vlastníka a skupiny k súborom a adresárom** používame príkaz **chown.**

Syntax:

chown [prepínače] [vlastník] súbor/adresár
chown [prepínače] [vlastník:skupina] súbor/adresár

Príklad: Chceme zmeniť vlastníka na test k súboru ahoj v adresári /home

```
chown test /home/ahoj
```

Príklad: Chceme zmeniť vlastníka na test a skupinu na talent k súboru ahoj v adresári /home

```
chown test:talent /home/ahoj
```

Prepínače:

- **„-R“** – umožňuje zmenu urobiť rekurzívne
  Chceme zmeniť vlastníka na test a skupinu na talent v celom adresári /home (aj podadresáre a súbory)

```
chown test:talent /home/ahoj -R
```

Na **zmenu skupiny k súborom a adresárom** používame príkaz **chgrp.**

Syntax:

chgrp [prepínače] [skupina] súbor/adresár

Príklad: Chceme zmeniť skupinu na talent k súboru ahoj v adresári /home

```
chgrp test /home/ahoj
```

Prepínače:

- **„-R“** – umožňuje zmenu urobiť rekurzívne
  Chceme zmeniť skupinu na talent v celom adresári /home (aj podadresáre a súbory)

```
chgrp talent /home/ahoj -R
```

3.5. Nástroje sudo a su

3.5.1. sudo

Príkaz **sudo** umožňuje oprávnenému používateľovi **vykonať príkaz ako superpoužívateľ** alebo **ako iný používateľ bez toho, aby musel zmeniť svoju identitu.** Inými slovami, nie je potrebné zmeniť používateľa operačného systému, aby sme vykonali činnosti pod týmto používateľom.

Syntax:

sudo príkaz

Príklad: Chceme spustiť príkaz ip ako superpoužívateľ (root) bez toho, aby sme sa museli prepnúť do účtu superpoužívateľa.

```
user@Linux:~$ sudo ip a
[sudo] password for user:
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 52:54:00:48:28:8a brd ff:ff:ff:ff:ff:ff
    inet 158.197.28.241/24 brd 158.197.28.255 scope global ens3
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe48:288a/64 scope link
       valid_lft forever preferred_lft forever
```

Prepínače:

- **„-u“** – prepínač pre určenie používateľa (bez prepínača sa vykonajú príkazy ako superpoužívateľ)

```
user@Linux:~$ sudo -u test ip a
[sudo] password for user:
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 52:54:00:48:28:8a brd ff:ff:ff:ff:ff:ff
    inet 158.197.28.241/24 brd 158.197.28.255 scope global ens3
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe48:288a/64 scope link
       valid_lft forever preferred_lft forever
```

3.5.2. su

Príkaz su sa používa na **zmenu používateľa počas relácie** (prihlásenia).

Syntax:

su [prepínače] meno_používateľa

Príklad: Chceme sa prepnúť na používateľa test:

```
root@Linux:/home/user# su test
[sudo] password for user:
```

```
test@Linux:/home/user$
```

Príklad: Chceme sa prepnúť na superpoužívateľa root:

```
test@Linux:/home/user$ su
root@Linux:/home/user#
```

4. Správa procesov

4.1. Úvod do správy procesov v Linuxe

Program, ktorý v rámci operačného systému spustíme, nazývame **proces** . Operačný systém procesu pridelí systémové prostriedky (najmä pamäť a procesor). Každému procesu v rámci systému je pridelený jedinečný **identifikátor** (*PID* – process ID). Operačný systém Windows, ako aj Linux, sú **viacúlohové *a* viacužívateľské** operačné systémy. To znamená, že v rámci operačného systému existuje niekoľko procesov bežiacich v tom istom čase. Údaje o procesoch sa ukladajú do **tabuľky procesov.** V tejto tabuľke je každý proces identifikovaný práve podľa svojho PID.

4.2. Zobrazenie procesov a ich stavov

V operačnom systéme Linux máme tabuľku procesov, ktorá sa podobá dátovej štruktúre. Uchováva v sebe všetky procesy a informácie o nich. Operačný systém Linux identifikuje procesy na základe ich PID.

4.2.1. Stavy procesov

Proces sa môže nachádzať v niekoľkých stavoch:

D	proces je v neprerušiteľnom spánku (v tomto stave sú zväčša procesy s I/O operáciami)
R	proces je práve spracovaný procesorom
S	proces zaspal
T	proces bol pozastavený
X	proces je mŕtvy
Z	zombie - proces, ktorého rodičovský proces ukončil svoju činnosť
<	proces s vysokou prioritou
N	proces s nízkou prioritou
L	proces ma uzamknuté stránky v pamäti – platí to zvyčajne pre procesy pracujúce v reálnom čase
s	je vedúci relácie
l	je multi-threading
+	proces je v skupine procesov, ktoré sú na popredí

4.2.2. ps (process status)

Na zobrazenie informácii o procesoch používame príkaz **ps.**

Syntax:

ps [prepínače]

Príklad:

```
root@Linux:/home/user# ps
  PID TTY          TIME CMD
29339 pts/0    00:00:00 sudo
29340 pts/0    00:00:00 su
29341 pts/0    00:00:00 bash
29470 pts/0    00:00:00 ps
```

Prepínače:

- **„-e“** – zobrazí všetky procesy

```
root@Linux:/etc/default# ps -e
```

- **„-a“** – zobrazí všetky procesy konkrétneho terminálu

```
root@Linux:/etc/default# ps -a
```

- **„-f“** – usporiada zobrazovanie podprocesov do stromu (ak je to možné)

```
root@Linux:/etc/default# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root     30023 30020  0 00:48 pts/0    00:00:00 sudo su
root     30024 30023  0 00:49 pts/0    00:00:00 su
root     30025 30024  0 00:49 pts/0    00:00:00 bash
root     30744 30025  0 08:56 pts/0    00:00:00 ps -af
```

- **„-o“** – zadefinovanie výstupného formátu

```
root@Linux:/etc/default# ps -o „pid ppid user"
  PID  PPID USER
29339 29336 root
29340 29339 root
29341 29340 root
```

```
29499 29341 root
```

Vyššie uvedený príkaz ps s prepínačom -o zobrazí procesy vo formáte pid, ppid a user.

Výstup z ps:

```
root@Linux:/home/user# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY STAT START   TIME COMMAND
root         1  0.0  0.0   8356   832 ?   Ss   Aug10   0:05 init [2]
root       338  0.0  0.0  54564  1000 ?   Ss   Aug10   0:00
/usr/sbin/sasla
root       339  0.0  0.0  54564   544 ?   S    Aug10   0:00
/usr/sbin/sasla
root       360  0.0  0.0   5984   732 ?   Ss   Aug10   0:01
/sbin/syslogd
root       408  0.0  0.0  20908   992 ?   Ss   Aug10   0:01
/usr/sbin/cron
107        413  0.0  0.0  23360   940 ?   Ss   Aug10   0:00
/usr/bin/dbus-d
root       439  0.0  0.0   9140  1392 ?   S    Aug10   0:00 /bin/sh
/usr/bi
mysql      551  0.0  1.3 241752 42092 ?   Sl   Aug10   5:42
/usr/sbin/mysql
root       552  0.0  0.0   3856   664 ?   S    Aug10   0:00 logger -t
mysql
root       691  0.0  0.0  49176  1140 ?   Ss   Aug10   0:00
/usr/sbin/sshd
user     29335  0.0  0.0  78976  1768 ?   S    15:47   0:00 sshd:
user@pts/
user     29336  0.0  0.0  17700  1888 pts/0 Ss   15:47   0:00 -bash
```

USER ()	informácia o používateľovi, ktorý proces spustil
PID (process ID)	identifikačné číslo procesu , **PID 1 má proces INIT.**
%CPU ()	informácia o využití procesoru pre proces v percentách
%MEM ()	informácia o využití pamäte pre proces v percentách
VSZ ()	veľkosť virtuálnej pamäte procesu v kB
RSS ()	časť pamäte obsadená procesom, ktorý je uchovávaný v hlavnej pamäti (RAM)

TTY ()	informácia o tom, z ktorého terminálu bol proces spustený. ? znamená, že proces nie je zviazaný so žiadnym terminálom
STAT ()	informácia, v akom stave je proces
START ()	dátum/čas, kedy bol proces spustený
TIME ()	informácia o procesorovom čase, ktorý bol danému procesu už pridelený
COMMAND ()	informácia o príkaze a jeho prepínačoch

4.2.3. pstree

Ďalším zaujímavým príkazom, ktorý prináša prehľadný pohľad na strom procesov, je príkaz **pstree.**

Syntax:

pstree [prepínače]

Príklad:

```
root@Linux:/home/user# pstree
init-+-apache2---5*[apache2]
     |-collectd---11*[{collectd}]
     |-cron
     |-dbus-daemon
     |-master-+-pickup
     |        `-qmgr
     |-mysqld_safe-+-logger
     |             `-mysqld---15*[{mysqld}]
     |-saslauthd---saslauthd
     |-sshd---sshd---sshd---bash---sudo---su---bash---pstree
     `-syslogd
```

- **„-p“** – zobrazenie stromu procesov spolu s ich identifikátorom (PID)

```
root@Linux:/etc/default# pstree -l
init(1)-+-apache2(2007)-+-apache2(28744)
        |               |-apache2(28745)
        |               |-apache2(29424)
        |               |-apache2(29425)
        |               `-apache2(29450)
        |-cron(408)
        |-dbus-daemon(413)
        |-master(683)-+-pickup(29458)
        |             `-qmgr(689)
        |-mysqld_safe(439)-+-logger(552)
```

```
        |                   `-mysqld(551)-+-{mysqld}(555)
        |                                 |-{mysqld}(556)
        |-saslauthd(338)---saslauthd(339)
        |-sshd(691)---sshd(29333)---sshd(29335)---bash(29336)---
sudo(29339)---s+
        `-syslogd(360)
```

- **pstree -a** - zobrazenie argumentov, s ktorými boli procesy spustené.
- **pstree -c** - zabrání zhlukovaniu rovnakých procesov, t.j. vypíšu sa aj tie, ktoré sa niekoľkokrát za sebou opakujú.
- **pstree -n** - zoradí procesy podľa PID (číselne) a nie podľa mena.
- **pstree -p** - za každým procesom zobrazí v zátvorke jeho PID.

4.2.4. top

Príkaz (nástroj) **top** poskytuje pohľad na vyťaženie procesora v reálnom čase. Tento príkaz zobrazuje zoznam procesov, ktoré najviac vyťažujú (používajú) procesor. Súčasne môže poskytnúť interaktívne rozhranie pre manipuláciu s nimi. Môže procesy triediť podľa využitia procesora, využitia pamäte a času behu procesu.

Syntax:

top [prepínače]

Príklad:

```
root@Linux:/home/user# top
top - 19:16:20 up 15 days, 1 min,  1 user,  load average: 0.00, 0.00,
0.00
Tasks:  27 total,   1 running,  26 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,
0.0%si,  0.0%st
Mem:   3145728k total,   676740k used,  2468988k free,        0k
buffers
Swap:        0k total,        0k used,        0k free,        0k
cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5588 root      20   0  632m 318m 3224 S    1 10.4  78:04.14 collectd
    1 root      20   0  8356  832  704 S    0  0.0   0:05.03 init
  338 root      20   0 54564 1000  500 S    0  0.0   0:00.00
saslauthd
  339 root      20   0 54564  544   44 S    0  0.0   0:00.00
saslauthd
  360 root      20   0  5984  732  564 S    0  0.0   0:01.50 syslogd
  408 root      20   0 20908  992  756 S    0  0.0   0:01.60 cron
```

```
  413 messageb  20   0 23360  940  624 S    0  0.0   0:00.04 dbus-
daemon
  439 root      20   0  9140 1392 1128 S    0  0.0   0:00.00
mysqld_safe
```

4.3. Práca s procesmi

4.3.1. &

Použitie 1 konzoly, resp. terminálu pre každý spustený príkaz by v praxi znamenalo, že pre spustenie ďalšieho príkazu by ste museli použiť ďalšiu konzolu, či terminál. Takýto prístup je neefektívny. Našťastie operačný systém Linux tento problém vyriešil a umožňuje **presunúť spustený program na pozadie**. K tomuto presunutiu slúži parameter **&**, ktorý sa zapisuje za daný príkaz.

Syntax:

príkaz &

Príklad: Chceme poslať 20 ICMP dopytov na server www.upjs.sk na zistenie dostupnosti tohto servera. Toto chceme spraviť tak, aby proces bežal na pozadí.

```
ping -c 20 www.upjs.sk &
```

4.3.2. jobs

Na **vypísanie bežiacich úloh v aktuálnom shelli** slúži príkaz **job.**

Syntax:

jobs [prepínače]

Príklad: Chceme vypísať bežiace úlohy spolu s ID skupiny procesu a adresára.

jobs -p

4.3.3. sleep

Na **uspanie procesu** slúži príkaz **sleep.**

Syntax:

sleep [čas]

- **s** - sekundy (defaultne)
- **m** - minúty
- **h** - hodiny
- **d** - dni

Príklad: Chceme proces uspať na 10 sekúnd.

```
#!/bin/bash
                                echo "Hi, I'm sleeping for 10 seconds..."
sleep 10
echo "all Done."
```

4.3.4. bg

Na **presunutie procesu na pozadie** používame príkaz bg:

Syntax:

bg príkaz

Príklad:

```
bg ping -c 20 www.upjs.sk
```

Na pozadie **možno presunúť aj už bežiaci proces**. Najprv musíme tento proces pozastaviť (na to použijeme klávesovú skratku **CTRL+z**), aby sme sa dostali k shellu a následne presunieme proces na pozadie.

4.3.5. fg

Na **presunutie procesu do popredia** používame príkaz fg:

Syntax:

fg príkaz

Príklad:

```
fg ping -c 20 www.upjs.sk
```

4.4. Signály a ich funkcia pri riadení procesov

Signál:

- je systémom generovaná udalosť, ktorá sa používa na základe reakcie na podmienku a dáva takto procesu šancu uskutočniť odozvu.
- začínajú prefixom SIG.

Popis základných signálov:

SIGTERM	žiadosť o ukončenie procesu
SIGSEGV	porušenie segmentácie pamäti
SIGABORT	prerušenie procesu
SIGHUP	odrezanie od terminálu (hangup)
SIGKILL	"vražda"
SIGQUIT	ukončenie terminálovej relácie procesu
SIGGINT	prerušenie terminálu **(Ctrl+c)**
SIGGILL	neplatná inštrukcia
SIGCONT	navrátenie procesu zo stavu pozastavenia
SIGSTOP	pozastavenie procesu (opak SIGCONT)
SIGINT	ukončenie procesu na popredí

4.4.1. Nástroj Kill

Na **zaslanie signálu k procesu** používame príkaz **kill.**

Syntax:

kill [signál] pid

Príklad: Chceme vypísať zoznam všetkých signálov.

```
root@Linux:/home/user# kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5)
SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10)
SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15)
SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20)
SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25)
SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30)
SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37)
SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42)
SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47)
SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52)
SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57)
SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62)
SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Príklad: Chceme poslať predvolený signál (SIGTERM) pre procesy s PID: 123,188,2000.

```
kill 123 188 2000
```

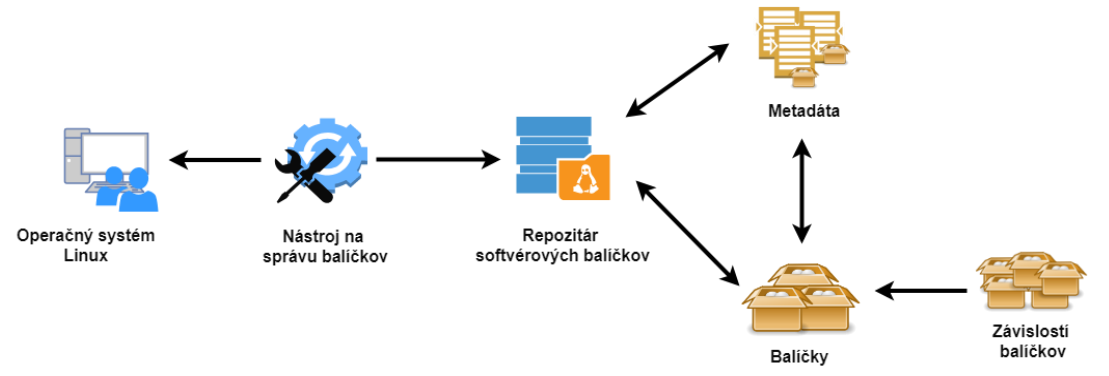
Príklad: Chceme zaslať signál (SIGKILL / 9) procesu s PID 200.

```
kill -9 200
```

5. Správa programov

5.1. Balíčkovacie systémy

V rámci operačného systému Linux sa programy spravujú v rámci softvérových balíčkov. *Softvérové balíčky* obsahujú všetky súbory súvisiace s programom, ich inštaláciou, resp. odstránením z operačného systému. Toto zahŕňa spustiteľné súbory, inštalačné skripty, údaje, konfiguračné súbory, dokumentáciu, softvérové prerekvizity a pod. Tieto balíčky sú uložené v rámci repozitárov softvérových balíčkov. Tieto repozitáre uchovávajú informácie o dostupných balíčkoch a ich metadátach (napr. dostupné verzie, potrebné prerekvizity).



Obrázok 6.1
Správa balíčkov v operačnom systéme Linux.

Na Obr. 6.1 je zobrazená schéma použitia repozitárov softvérových balíčkov a samotných balíčkov. Používateľ v rámci operačného systému používa nástroj na správu balíčkov v systéme. Pomocou tohto nástroja sa pripája k repozitáru balíčkov. Ako sme už vyššie uviedli, repozitár si uchováva informácie o balíčkoch, ich verziách, prerekvizitách, ale aj iné metadáta.

Jednotlivé distribúcie operačného systému Linux sa medzi sebou odlišujú aj použitím *nástrojov na správu balíčkov*. Najčastejšie sa používajú nasledujúce nástroje:

- *dpkg* – inštaluje balíčky z lokálneho úložiska (napr. disku). Tento nástroj použijeme u balíčkov, ktoré nenájdeme v repozitároch, ale stiahneme z Internetu. Používajú sa pre distribúciu Debianu a Ubuntu (balíčky majú príponu deb),
- *apt* – predstavuje nadstavbu nad dpkg, prehľadáva repozitáre, vie nainštalovať, resp. odoberať softvérové balíčky. Pri inštalácii automaticky zistí a stiahne všetko potrebné.
- *aptitude* – predstavuje nadstavbu nad apt. Obsahuje jednoduché grafické a konzolové rozhranie , vylepšuje riešenie problémov so závislosťami.
- *yum* – pre distribúcie Fedora a Red Hat Enterprise Linux (balíčky majú príponu rpm).
- *YaST* – pre distribúciu SUSE Linux.

5.2. Balíčky

Vyššie sme uviedli, že *softvérové balíčky (packages)* obsahujú všetky súbory súvisiace s programom, ich inštaláciou, resp. odstránením z operačného systému. Softvérový balíček je riadne začleniť do distribúcie, pre ktorý bol zostavený s ohľadom na inštalačné cesty, závislostí, integráciu plochy, vlastné spúšťacie skripty pre servery apod. Z týchto dôvodov by sa mali vždy inštalovať balíčky, ktoré boli zostavené pre konkrétnu distribúciu.

Balíček obsahuje tiež ďalšie informácie, bežne označované ako **metadáta,** ako je napríklad:

- zhrnutie,
- opis,
- zoznam súborov obsiahnutých v balíčku,
- verzia softvéru, rovnako ako číslo verzie balíka,
- kedy, kde a kto zostavil balíček,
- pre akú architektúru je vhodný
- kontrolné súčty súborov obsiahnutých v balíčku,
- licencie softvéru, ktorý obsahuje,
- ktoré ďalšie balíčky vyžaduje to, aby fungoval správne,
- atď.

Schéma pomenovania balíčkov pre Debian:

```
package_version-build_architecture.deb
```

- **balíček (package):** názov inštalovanej aplikácie
- **verzia (version):** číslo verzie aplikácie
- **zostavenie (build):** číslo zostavenia balíčka. Zakaždým, keď je balíček prepracovaný, toto číslo sa zvýši.
- **architektúra (architecture):** platforma, pre ktorú bol balíček zostavený

Dôležitým aspektom softvérových balíčkov sú vzťahy, ktoré obsahujú. Balíčky sú vo vzťahu k iným balíčkom, napríklad balíčky potrebujú prostredie pre spúšťanie (iné nástroje, knižnice, atď.) Je to tiež veľmi účinný spôsob, ako udržať stabilný a bezpečný systém. Keď sa objaví bezpečnostná diera v knižnici používanej jedným alebo mnohoými programami, upgrade jedného balíčku opraví všetky z nich.

5.3. Nástroje

5.3.1. Nástroj DPKG

Nástroj DPKG (Debian package management tools) [7] sa používa najmä pri inštalácii lokálnych balíčkov s príponou deb. Na inštaláciu balíčka sa použije príkaz *dpkg -i názov_balíčka*. Nástroj DPKG inštaluje balíčky z vybraného adresára na disku a kontroluje, či sú nainštalované všetky súvisiace balíčky v operačnom systéme. Ak nie, zobrazí sa upozornenie na chybné balíky a inštalácia sa nevykoná. Pomocou nástroja DPKG môžeme tiež zistiť zoznam nainštalovaných balíčkov, použitím *dpkg -l*. Pre odinštaláciu existuje príkaz *dpkg -r názov_balíčka*. Nástroj DPKG daný balíček odinštaluje, ale nekontroluje súvisiace balíčky. Z tohto dôvodu je dobré použiť nástroj APT. Nižšie si bližšie ukážeme základné príkazy nástroja DPKG:

1) **Inštalovanie balíčka:**

```
dpkg --install súbor_balíčka.deb
dpkg-i názov_balíčka.deb
```

2) **Odinštalovanie balíčka:**

```
dpkg --remove názov_balíčka.deb
dpkg-r názov_balíčka.deb
```

3) **Informácie o balíčku:**

```
dpkg --print-avail názov_balíčka.deb
dpkg-p názov_balíčka.deb
```

4) **Hľadanie balíčkov:**

```
dpkg --list <pattern>
dpkg -l <pattern>
```

5.3.2. Nástroj APT

Druhou možnosťou pre správu programov v distribúcii Debian, je použiť nástroj *APT* (Advanced Package Tool) *[8]*. Ukážku inštalácie balíčka mc (midnight commander), pomocou tohto nástroja (*apt install mc*), môžeme vidieť na Obr. 6.2. Medzi bežné používané príkazy môžeme zaradiť:

- *apt update* - uskutoční aktualizáciu zdrojov balíčkov,
- *apt upgrade* - stiahne a nainštaluje aktualizované balíčky,
- *apt autoremove* – odstráni všetky už nepotrebné balíčky,
- *apt install názov_balíčka* - nainštaluje balíček,
- *apt remove názov_balíčka* - odinštaluje balíček,
- *apt dist-upgrade* - používa sa pre upgrade celej distribúcie,
- *apt list* – zobrazí zoznam nainštalovaných balíčkov,
- *apt-cache* - je nástroj pre prácu s informáciami o balíčkoch,
- *apt-cache search slovo* - vyhľadá balíčky, ktoré obsahujú text „slovo“ a
- *apt-cache show názov_balíčka* - zobrazí informácie o balíčku.

```
root@Linux:~# apt install mc
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  arj catdvi | texlive-binaries dbview djvulibre-bin genisoimage gv imagemagick libaspell-dev
  links | w3m | lynx odt2txt poppler-utils python-boto python-tz xpdf | pdf-viewer
The following NEW packages will be installed:
  mc
0 upgraded, 1 newly installed, 0 to remove and 41 not upgraded.
Need to get 513 kB of archives.
After this operation, 1,465 kB of additional disk space will be used.
Get:1 http://mgt.science.upjs.sk:3142/debian stretch/main amd64 mc amd64 3:4.8.18-1 [513 kB]
Fetched 513 kB in 0s (22.0 MB/s)
Selecting previously unselected package mc.
(Reading database ... 67012 files and directories currently installed.)
Preparing to unpack .../mc_3%3a4.8.18-1_amd64.deb ...
Unpacking mc (3:4.8.18-1) ...
Processing triggers for mime-support (3.60) ...
Setting up mc (3:4.8.18-1) ...
update-alternatives: using /usr/bin/mcview to provide /usr/bin/view (view) in auto mode
root@Linux:~#
```

Obrázok 6.2
Ukážka inštalácie nástroja mc (midnight commander) pomocou nástroja apt v operačnom systéme Debian.

Nižšie uvádzame podrobnejší popis niektorých príkazov nástroja **APT:**

1) **aktualizácia zoznamu balíčkov** - stiahne zoznam balíčkov z repozitárov a aktualizuje informácie o nových verziách balíkov a závislostí

apt update

```
Hit http://sk.archive.ubuntu.com trusty/multiverse Sources
Hit http://sk.archive.ubuntu.com trusty/main i386 Packages
```

```
Hit http://sk.archive.ubuntu.com trusty/restricted i386 Packages
Hit http://sk.archive.ubuntu.com trusty/universe i386 Packages
```

2) aktualizácia softvérového vybavenia

apt upgrade

```
Linux 3.13.0-74-gene
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages have been kept back:
openjdk-6-jre openjdk-6-jre-headless openjdk-6-jre-lib python-
reportbug
reportbug
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
```

3) inštalácia balíka

apt install názov_balíčka

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer
required:
python3.1 python3.1-minimal python3-minimal
Suggested packages:
arj xpdf-reader pdf-viewer dbview odt2txt gv catdvi djvulibre-bin
imagemagick python-boto python-tz
The following NEW packages will be installed:
mc
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 2,173 kB of archives.
After this operation, 6,603 kB of additional disk space will be used.
```

4) Zobrazenie zoznamu balíčkov

apt list

- zobrazenie nainštalovaných balíčkov:

apt list --installed

- zobrazenie balíčkov, ktoré je možné upgradovať:

apt list --upgradable

- zobrazenie zoznamu balíčkov všetkých verzií:

apt list --all-versions

5) vyhľadanie balíčka

apt-cache search hľadaný_výraz

```
root@Linux:/# apt-cache search bash
abs-guide - The Advanced Bash-Scripting Guide
acr - autoconf like tool
android-androresolvd - Daemon to transfer Android DNS property to
resolv.conf
apparix - console-based bookmark tool for fast file system navigation
awesome-extra - additional modules for awesome
backup-manager - command-line backup tool
bash - GNU Bourne Again SHell
bash-builtins - Bash loadable builtins - headers & examples
bash-doc - Documentation and examples for the GNU Bourne Again SHell
bash-static - GNU Bourne Again SHell (static version)
bash-completion - programmable completion for the bash shell
```

6) zobrazenie popisu balíka

apt show názov_balíčka

```
root@Linux:/# apt show mc
Package: mc
Status: install ok installed
Priority: optional
Section: utils
Installed-Size: 1420
Maintainer: Ubuntu Developers <ubuntu-devel-
```

```
discuss@lists.ubuntu.com>;
Architecture: i386
Version: 3:4.8.11-1
Provides: mcedit
Depends: e2fslibs (>= 1.37), libc6 (>= 2.15), libglib2.0-0 (>=
2.35.9),
libgpm2 (>= 1.20.4), libslang2 (>= 2.2.4), libssh2-1 (>= 1.2.5), mc-
data (= 3:4.8.11-1)
Recommends: mime-support, perl, unzip
Suggests: arj, bzip2, catdvi | texlive-binaries, dbview, djvulibre-
bin,
file, genisoimage, gv, imagemagick, links | w3m | lynx, odt2txt,
poppler-utils,
python, python-boto, python-tz, xpdf | pdf-viewer, zip
Conffiles:
/etc/mc/mc.emacs.keymap 441b449b490d5766f407fd59a69a7db2
```

6. Plánovanie úloh

6.1. Plánovanie opakovaných úloh

Cron je démon, ktorý je možné použiť na plánovanie vykonávania opakovaných úloh podľa zadaného časového údaju (deň v mesiaci, mesiac, deň v týždni a týždeň). Cron predpokladá, že operačný systém beží a je schopný vykonávať naplánované úlohy. Ak operačný systém nie je zapnutý v čase, keď je naplánovaná úloha, tá sa nevykonaná. Hlavným konfiguračným súborom pre Cron je súbor **/etc/crontab:**

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root HOME=/
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Na editáciu tabuľky Cron používame:

```
crontab -e
```

Každý riadok v súbore /etc/crontab reprezentuje úlohu a má nasledujúci formát:

```
minute   hour   day   month   dayofweek   command
```

```
* * * * * príkaz_korý_sa_má_vykonať
- - - - -
|  |   |  |  |
|  |   |  |   ----- Day of week (0 - 7) (Sunday=0 or 7)
|  |   |   ------- Month (1 - 12)
|  |   --------- Day of month (1 - 31)
|  ----------- Hour (0 - 23)
------------- Minute (0 - 59)
```

- **minúta (minute)** — prirodzené číslo od 0 do 59

- **hodina (hour)** — číslo od 0 do 23
- **deň (day)** — číslo od 1 do 31
- **mesiac (month)** — číslo od 1 do 12
- **deň v týždni (dayofweek)** — číslo od 0 do 7
- **príkaz (command)** — príkaz na vykonanie

Ako je môžné vidieť v **/etc/crontab**, existujú osobitné adresaře, ktorých obsah (súbory) sa spustí v preddefinovanom čase (hodina, deň, týždeň, mesiac):

- **/etc/cron.hourly/** - každú hodinu,
- **/etc/cron.daily/** - každý deň,
- **/etc/cron.weekly/** - každý týždeň,
- **/etc/cron.monthly/** - každý mesiac.

Nižšie je možné vidieť ukážku adresára **/etc/cron.daily/**

```
root@Linux:/# ls -la /etc/cron.daily/

drwxr-xr-x  2 root root 4096 Dec 11  2017 .
drwxr-xr-x 89 root root 4096 Jan  3 20:10 ..
-rwxr-xr-x  1 root root  539 Sep 19  2017 apache2
-rwxr-xr-x  1 root root 1474 Jul 13  2017 apt-compat
-rwxr-xr-x  1 root root  355 Oct 25  2016 bsdmainutils
-rwxr-xr-x  1 root root 1597 Feb 22  2017 dpkg
-rwxr-xr-x  1 root root   89 May  6  2015 logrotate
-rwxr-xr-x  1 root root 1065 Dec 13  2016 man-db
-rwxr-xr-x  1 root root  249 May 17  2017 passwd
-rw-r--r--  1 root root  102 May  3  2015 .placeholder
```

Príklad: Chceme, aby sa skript na zálohovanie backup.sh spúšťal o 4:10 1. deň v mesiaci.

crontab -e

```
Pridáme nasledujúci riadok:
10 4 1 * * /root/scripts/backup.sh
```

Ako môžme vidieť, pre údaje, pri ktorých chceme, aby platili pre celý rozsah, dávame „*“. V danom prípade sme chceli, aby sa skript vykonal každý mesiac a nebol obmedzený dňom v týždni.

Príklad: Chceme, aby sa skript na zálohovanie backup.sh spúšťal 5 minút po polnoci každý deň (00:05).

crontab -e

```
Pridáme nasledujúci riadok:
5 0 * * * /root/scripts/backup.sh
```

Príklad: Chceme, aby sa skript na zálohovanie backup.sh spúšťal 14:15 prvý deň každý mesiac.

crontab -e

```
Pridáme nasledujúci riadok:
15 14 1 * * /root/scripts/backup.sh
```

Príklad: Chceme, aby sa skript na zálohovanie backup.sh spúšťal každých 15 minút.

crontab -e

```
Pridáme nasledujúci riadok:
0,15,30,45 * * * * /root/scripts/backup.sh
```

Alternatívne vieme použiť zástupný znak /časová_jednotka. Napr. každých 5 minút - /5, každých 15 minút /15.

```
Pridáme nasledujúci riadok:
0/15 * * * * /root/scripts/backup.sh
```

Rozsah údajov vieme zadávať cez pomlčku (napr. 10-20. deň v mesiaci – 10-20), alebo vymenovaním možností cez čiarku (napr.máj,septeber, december – 5,9,12).

Príklad: Chceme, aby sa skript na zálohovanie backup.sh spúšťal 12-18 minútu v júni a auguste.

```
crontab -e
```

```
Pridáme nasledujúci riadok:
12-18 * * 6,8 * /root/scripts/backup.sh
```

6.2. Plánovanie jednorázových úloh

Príkaz **at** - vykoná príkaz (úlohu) v špecifickom čase. Úlohy môžeme naplánovať dvoma rôznymi spôsobmi:

- naplánovanie úlohy majú byť vykonané v určitom čase (napr. 21.11.2014 o 14:05),
- naplánovanie úlohy majú byť vykonané v relatívnom čase od danej chvíle (napr. 5 minút od tejto chvíle).

Syntax:

```
$ at čas dátum
```

Príklad:

```
$ at 8am Oct 21
warning: commands will be executed using /bin/sh
at> echo "Ahoj"
at> <EOT>
job 1 at Wed Oct 21 08:00:00 2015
```

EOT znamená End-of-Transmission character (ukončovací znak). V príkaze EOT znamená, že už nebudú pokračovať žiadne znaky. Zadáva sa klávesovou skratkou CTRL+D.

Atq (at -l) - príkaz zobrazí všetky naplánované úlohy

Syntax:

```
$ atq
```

Príklad: Nasledujúci príkaz atq vypíše všetky čakajúce úlohy. Prvé číslo je číslo úlohy, nasleduje čas, kedy má byť uloha vykonaná a používateľské meno.

```
$ atq
2       Wed Oct 21 09:00:00 2015 a user1
1       Wed Oct 21 08:00:00 2015 a user2
```

Atrm (at -d) - používa sa na odstránenie určitých úloh.
Syntax:

```
$ atrm číslo_úlohy
```

Príklad: Nasledujúce príkaz odstráni úlohu č. 2.

```
$ atrm 2
```