Пренасочување, филтрирање и регуларни изрази

Оперативни системи

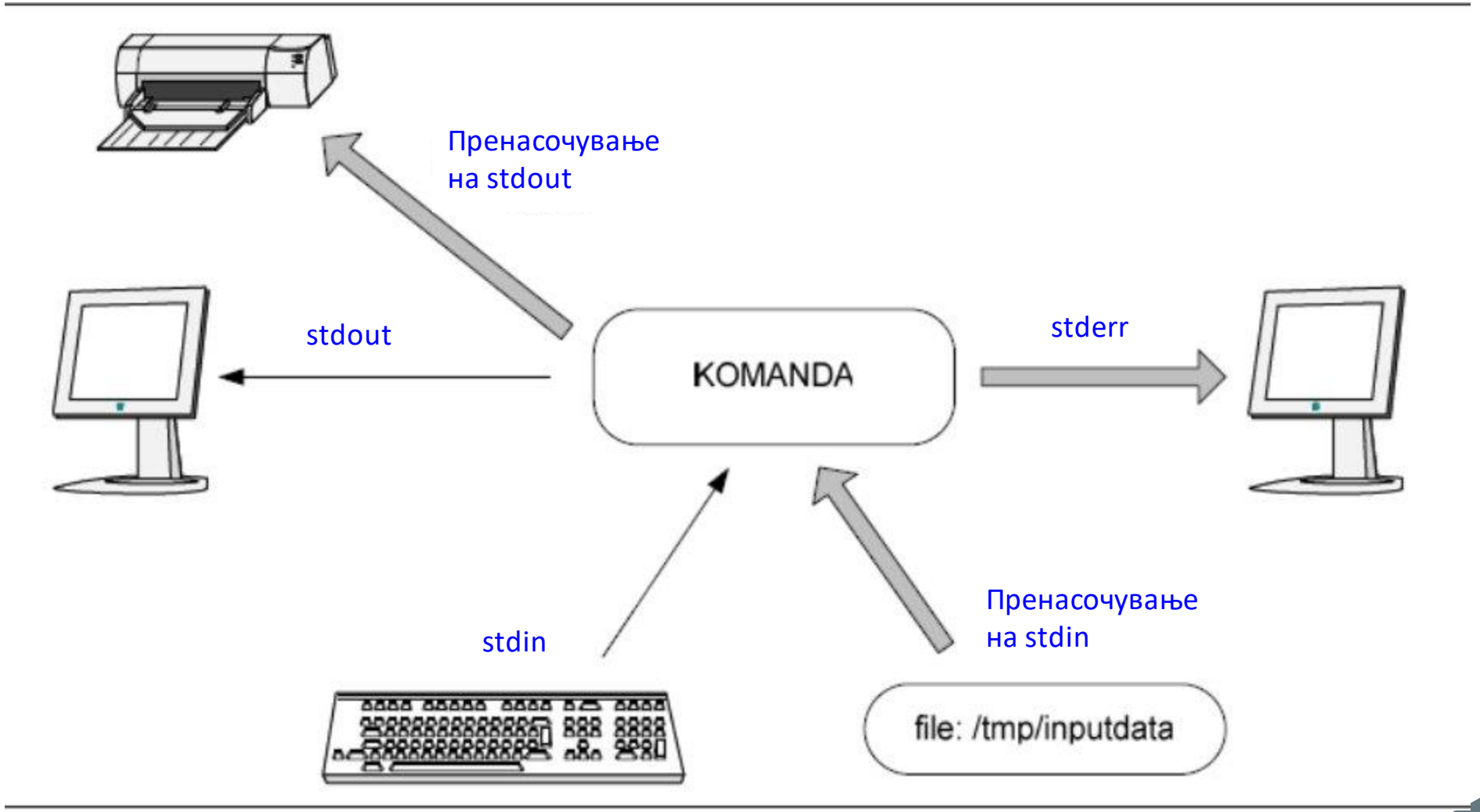
Аудиториска вежба 2

Пренасочување и концепт на цевки (pipe)

Двата основни концепти во работата со командниот интерпретер во UNIX се:

- пренасочувањето на стандардниот влез и стандардниот излез во кориснички дефинирани датотеки (redirection).
- проследувањето на излезот од една команда како влез на друга (концепт на цевки, т.е. pipes).

Пренасочување на стандардниот влез и излез



Пренасочување на излез (>, >>)

- Пренасочувањето на излезот со > овозможува креирање на нова датотека (доколку таква не постои) или пребришување на содржината (доколку датотека со такво име веќе постои). Во двата случаи резултатот од претходната команда наместо на стандарден излез (екран) се праќа во датотека.

- Пренасочувањето на излезот со >> овозможува додавање на излезот од командата на крајот во датотеката, по претходната содржина (доколку таква датотека постои) или пак креира нова датотека (доколку таква не постои). Во секој случај, не се пребришува ништо.

Пренасочување на влез (<)

- Пренасочувањето на влезот со < овозможува влезот за командата да не се чита од стандардниот влез, туку од кориснички дефинирана датотека.

Пример за пренасочување

- **ls -l > lista.txt**
  - Ако не постои lista.txt, ќе биде креирана
  - Ако постои lista.txt, ќе биде пребришана

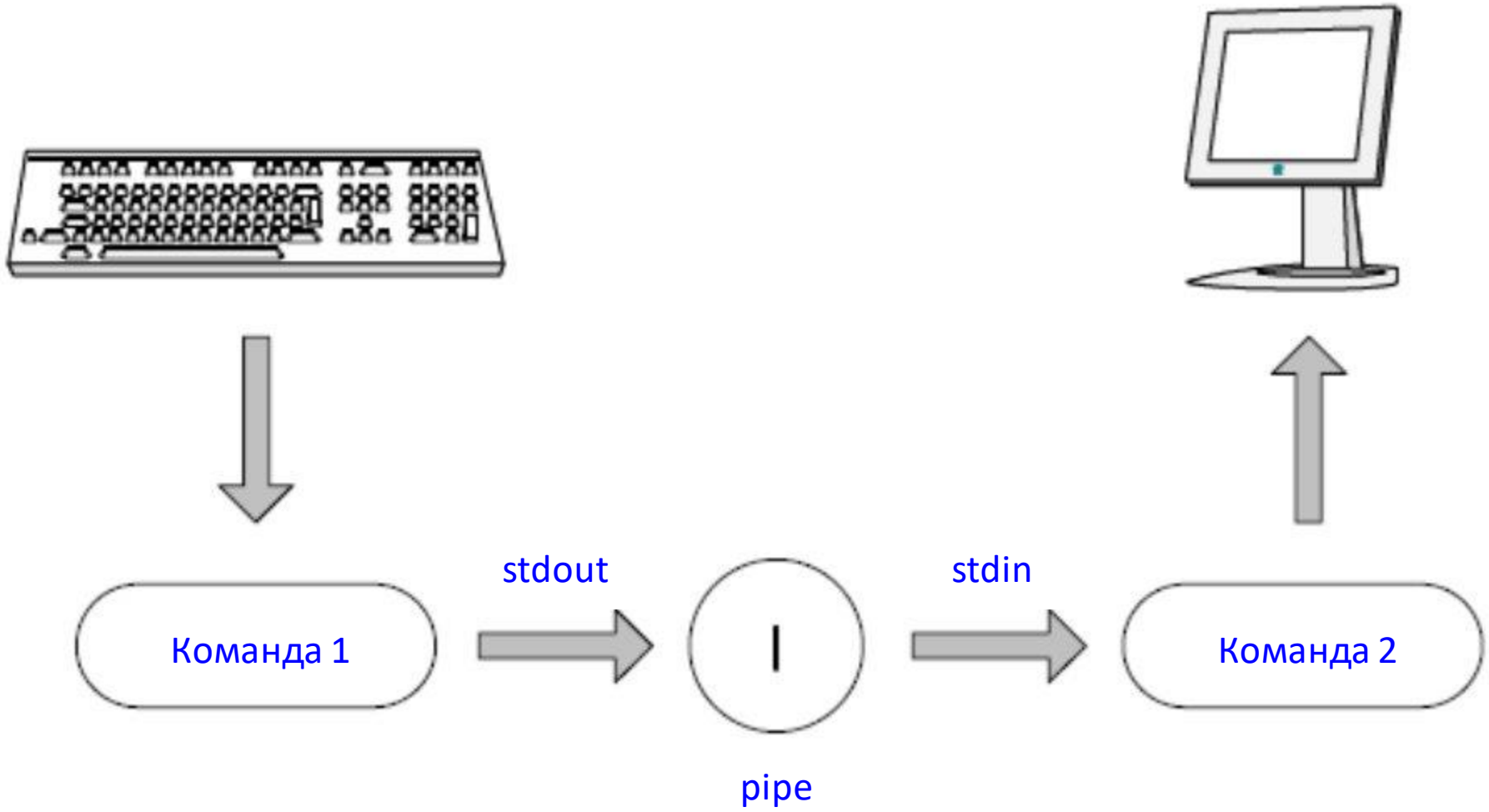
- **wc -l < lista.txt**

- **ls >> postoecka.txt**
  - Ако не постои postoecka.txt, ќе биде креирана
  - Ако постои postoecka.txt, ќе се продолжи со запишување на крајот од датотеката

- **sort < lista.txt > sortirana.txt**

- **sort -r < lista.txt > sortirana.txt**
  - сортирана листа во обратен редослед

Концепт на цевки



Концепт на цевки

- Концептот на цевки овозможува излезот од претходната команда да се третира како влез за командата што следи, без притоа тој да се испишува на стандарден излез (екран).
  - **ls -l | wc -l**
  - **ls -l | tee lista.txt | wc -l**
  - **man ls | more**

Џокер знаци (wildcards) во Shell

- Употребата на т.н. џокер знаци овозможува копирање, прелистување, преместување, бришење, итн., на повеќе датотеки со слични имиња одеднаш (со единствена команда).
- Симболот **?** заменува кој било знак во името на датотеката на неговото место.
- Симболот ***** заменува 0 или повеќе произволни знаци на неговото место.
- Знаците наведени во средни загради **[ ]** заменуваат кој било од нив на даденото место. Дозволено е дефинирање на опсег.

Примери за џокер знаци

- **$ cp ?ab2 nov/**
  - во именикот nov ќе ги ископира сите датотеки од тековниот именик чии имиња имаат 4 знаци и завршуваат на ab2;
- **$ mv ab* nov/**
  - во именикот nov ќе ги премести сите датотеки од тековниот именик чии имиња почнуваат на ab (вклучувајќи ја и датотеката со име ab);
- **$ mv a*b nov/**
  - во именикот нов ќе ги премести сите датотеки од тековниот именик чии имиња почнуваат на a и завршуваат на b (вклучувајќи ја и датотеката со име ab);
- **$ rm s[aqz]**
  - ќе ги избрише датотеките sa, sq и sz;
- **$ rm s[2-4]x**
  - ќе ги избрише датотеките s2x, s3x и s4x;

Примери за џокер знаци

- **ls *.cc**
  - ги листа сите датотеки кои имаат наставка .cc
- **ls –l [abcxyz]* | more**
  - ќе се излистаат сите именици/датотеки чии имиња почнуваат на буквите 'a', 'b', 'c', 'x', 'y' или 'z'
- **rm *[^0-9]**
  - ќе се избришат сите именици/датотеки чие име не завршува на број
  - ^ во средни загради [] означува негација, т.е. сите карактери освен наведените
- **mv {ii,[0-9]}*.{txt,doc,jpg} /home/student/lab4/**
  - ќе се преместат сите датотеки чие име започнува на ii или некоја цифра, па следува низа од карактери и завршува со .txt или .doc или .jpg во именикот student/lab4

Команди за филтрирање на текст

Постојат повеќе команди за филтрирање на текстуална содржина, од кои најзначајни се:

- **grep**: основна команда за филтрирање на текст
- **sed**: понапредна команда
- **awk**: команда со програмски јазик за напредно филтрирање

Повремено ќе ги користиме и командите:

- **tr**: превод /замена на карактери или низи од карактери во текст
- **sort**: сортирање на текстуална содржина
- **uniq**: детекција и/или отстранување на линии кои се повторуваат

Командата grep

- Синтакса: **grep [-options] pattern [file]**
- Командата grep е една од најкористените алатки за филтрирање во UNIX.
- grep пребарува линија по линија од датотеката (или влезот), за да ја пронајде бараната низа (pattern) и на екран ја испишува секоја линија што ја содржи низата.
- Доколку не се зададе датотека, grep чита од стандардниот влез.
- За формирање на низите што се бараат со grep може да се користат следните знаци:
  - ^ означува почеток на ред
  - $ означува крај на ред
  - . означува кој било знак
  - * означува 0 или колку било појавувања на претходниот знак
  - [a-b] означува кој било знак кој лексички се наоѓа меѓу a и b.

Командата grep

- **grep 'jon' /etc/passwd**
- **grep '^jon' /etc/passwd**
- **ls -l /tmp | grep 'root'**
- **grep foo ***
  - го бара foo во содржината на сите датотеки во тековниот именик
- **grep -v foo ***
  - select non-matching lines
- **grep -l foo ***
  - print only names of FILEs containing matches
- **grep -L foo ***
  - print only names of FILEs containing no match
- **grep ´^once.*forever *$´**
- **grep -i unix ch16.doc**
  - ignore case distinctions

Командата sed

- Синтакса: **sed [-option] ‘pattern’ [file]**
- Pattern делот може да биде во формат:
  - **‘/pattern/action’**
    - ја презема action акцијата (d за бришење, p за печатење) над линијата која го содржи регуларниот израз pattern;
  - **‘s/pattern1/pattern2’**
    - го заменува првото појавување на pattern1 во секоја од редиците, со pattern2;
  - **‘s/pattern1/pattern2/gi’**
    - опцијата g означува global – сите појавувања на pattern1 ќе се заменат, не само првото;
    - опцијата i означува ignore case – ќе се детектираат сите верзии на pattern1, без разлика на големината на буквите;

Командата sed

- $ cat fruit_prices.txt

Ovosje	Cena	
Banana	0.89	
Praska	0.79	
Kivi		1.50
Ananas	1.29	
Jabolka	0.99	
Mango	2.20	

Командата sed

- $ sed '/0\.[0-9][0-9]$/p' fruit_prices.txt

Ovosje	Cena	
Banana	0.89	
Banana	0.89	
Praska		0.79
Praska		0.79
Kivi	1.50	
Ananas	1.29	
Jabolka	0.99	
Jabolka	0.99	
Mango	2.20	

Командата sed

- $ sed -n '/0\.[0-9][0-9]$/p' fruit_prices.txt

```
Banana          0.89
Praska                  0.79
Jabolka         0.99
```

Командата sed

- $ sed '/^[Mm]ango/d' fruit_prices.txt

Ovosje	Cena	
Banana	0.89	
Praska		0.79
Kivi	1.50	
Ananas	1.29	
Jabolka	0.99	

Командата sed

- $ sed 's/Paska/Praska/' fruit_prices.txt
- $ sed 's/eqal/equal/g' nash.txt

Командата sed

- Синтакса за повеќе sed команди во една наредба:
  - sed -e 'command1' -e 'command2' ... -e 'commandN' [file]
- Пример:
  - $ sed -e 's/Paska/Praska/'
    -e 's/[0-9][0-9]*\.[0-9][0-9]$/\$&/' fruit_prices.txt

Ovosje	Cena
Banana	$0.89
Praska	$0.79
Kivi	$1.50
Ananas	$1.29
Jabolka	$0.99

Командата awk

- Програмски јазик кој овозможува пребарување низ датотеки и манипулирање со нивната содржина.
- Синтакса:
  - awk [-options] 'script' [file]
- Пример:
  - awk '{ print ; }' fruit_prices.txt

Fruit	Price/lbs	Quantity
Banana	$0.89	100
Peach	$0.79	65
Kiwi	$1.50	22
Pineapple	$1.29	35
Apple	$0.99	78

Командата awk

- $ awk '{ print $1 $3 ; }' fruit_prices.txt

  Fruit Quantity
  Banana 100
  Peach 65
  Kiwi 22
  Pineapple 35
  Apple 78

Командата awk

- $ awk '{ print $1, $3 ; }' fruit_prices.txt
  Fruit Quantity
  Banana 100
  Peach 65
  Kiwi 22
  Pineapple 35
  Apple 78

- $ awk '{ printf "%-15s %s\n" , $1 , $3 ; }' fruit_prices.txt
  Fruit Quantity
  Banana 100
  Peach 65
  Kiwi 22
  Pineapple 35
  Apple 78

Командата awk

- $ awk '
  / *\$[1-9][0-9]*\.[0-9][0-9] */
  { print $0, "*"; }
  / *\$0\.[0-9][0-9] */
  { print ; }' fruit_prices.txt

```
Banana          $0.89           100
Peach                   $0.79           65
Kiwi                    $1.50           22 *
Pineapple  $1.29                35 *
Apple                   $0.99           78
```

Оператори за споредба кај awk

- Слично како кај C и кај оваа команда се користат оператори за споредба на броеви и стрингови.
- expression { actions; }
  - <, >, <=, >=, ==, !=, ~,!~
- $ awk '
  $3 <= 75
  { printf "%s\t%s\n", $0, "REORDER" ;}
  $3 > 75
  { print $0 ; } ' fruit_prices.txt

```
Banana          $0.89           100
Peach           $0.79   65 REORDER
Kiwi                    $1.50   22 REORDER
Pineapple       $1.29           35 REORDER
Apple                   $0.99           78
```

Сложени изрази кај awk

- Може да се комбинираат повеќе проверки во логички изрази:
  - (expr1) && (expr2)
  - (expr1) || (expr2)
- $ awk '
  ($2 ~ /^\$[1-9][0-9]*\.[0-9][0-9]$/)
  && ($3 < 75)
  { printf "%s\t%s\t%s\n",$0,"*","REORDER“; }' fruit_prices.txt

Kiwi $1.50 22 * REORDER
Pineapple $1.29 35 * REORDER

Користење на stdin како влез кај awk

- $ ls –l

```
total 32
-rw-r--r-- 1 asistent users   62 Apr  3 17:23 dat1.txt
-rw-r--r-- 1 asistent users   64 Apr  3 17:26 dat2.txt
-rw-r--r-- 1 asistent users   64 Apr  3 17:30 dat.txt
-rw-r--r-- 1 asistent users  100 Apr  8 21:50 fruit_prices_awk.txt
-rw-r--r-- 1 asistent users   82 Apr  8 21:31 fruit_prices.txt
drwxr-xr-x 2 asistent users 4096 Apr  3 17:29 imenik
-rw-r--r-- 1 asistent users    6 Apr  8 08:15 izlez.txt
-rw-r--r-- 1 root     root     0 Jan 21 08:54 krajna.txt
drwxr-xr-x 2 root     root  4096 Apr  8 21:12 nov
```

- $ ls -l | awk '$1 !~ /total/ { printf "%-32s %s\n",$9,$5 ; }‘

```
dat1.txt                         62
dat2.txt                         64
dat.txt                          64
fruit_prices_awk.txt             100
fruit_prices.txt                 82
imenik                           4096
izlez.txt                        6
krajna.txt                       0
nov                              4096
```

Командата awk

- Листање на имиња на датотеки / именици (со вклучени празни места) и големина на датотеката / именикот.
- awk скрипта (awk.sh):

```
$1 !~ /total/
{
    x=9
    while(x<=NF) {
        printf "%s ", $x;
        x++;
    }
    printf "\t%s\n", $5;
}
```

- Извршување:
  - $ ls -l | awk -f awk.sh

Користење оператори кај awk

- num1 operator num2
- оператори: +, -, *, /, %, ^, +=, -=, *=, /=, %=, ^=
-
- $ awk '
  BEGIN { x=0; }
  /^ *$/ { x+=1 ; }
  END { print x ; }' datoteka.txt

Командата awk

- Кај awk постои опција за работа и со различен field separator.
  - За таа цел се користи опцијата –F, по која веднаш се наведува новиот field separator;
  - Пример
    - awk –F, ‘script’ file
    - awk –F\; ‘script’ file

Командата awk

- $ awk '{ print $1 ; }' fruit_prices.csv

  Fruit,Price/lbs,Quantity
  Banana,$0.89,100
  Peach,$0.79,65
  Kiwi,$1.50,22
  Pine,$1.29,35
  Apple,$0.99,78

- $ awk –F, '{ print $1 ; }' fruit_prices.csv

  Fruit
  Banana
  Peach
  Kiwi
  Pine
  Apple

Останати команди за работа со текст

- tr 'set1' 'set2'
  - tr 'A-Z' 'a-z'
  - echo "Softver" | tr 'S' 's'
  - echo "poozdrav" | tr -s 'o'
- sort
  - sort –rn врши нумеричко сортирање по опаѓачки редослед
- uniq
  - uniq fruits.txt

Команди за следење на процеси

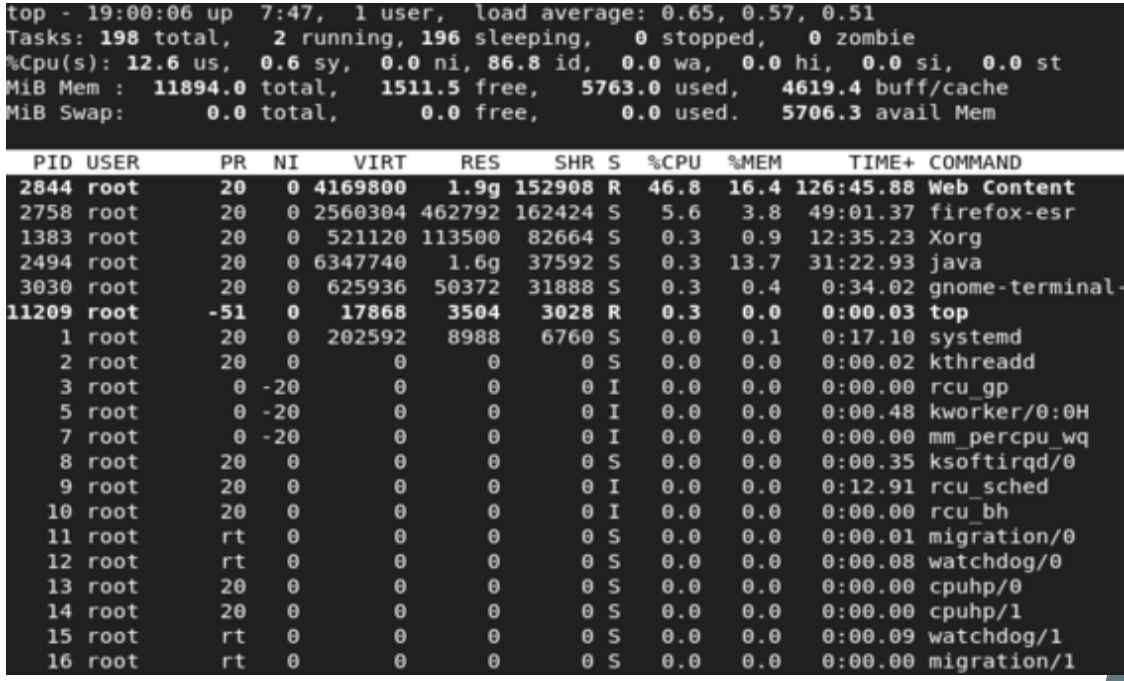
top = table of processes

- Прикажување на сите активни процеси во околината со информации за нивна искористеност на CPU и меморија
- Подредена листа на активни процеси која повремено се апдејтира и ги прикажува процесите сортирани почнувајќи од најголемата искористеност на CPU

- Опции:
  - -m = подредување на активните процеси според искористеноста на меморијата
  - -u UID
    = прикажување на активните процеси само за конкретен корисник според неговиот UID

top

- **PID** = process ID
- **USER** = username на корисникот кој го активирал процесот
- **PR** = приоритет на процесот
- **NI** = (nice value) негативна вредност означува повисок приоритет
- **VIRT** = вкупна искористеност на виртуелна меморија од страна на процесот
- **RES** = (resident) вкупна искористеност на физичка меморија во kb
- **SHR** = количина на искористена споделена меморија во kb
- **S** = статус на процесот (R – running, S – sleeping, Z – zombie,…)
- **%CPU** = процент на CPU искористеност од процесот по последниот апдејт на top
- **%MEM** = процент на искористеност на физичката меморија
- **TIME+** = вкупно CPU време искористено од почеток на процесот до стотинки од секунда
- **COMMAND** = командна линија или име на програмата која го започнала процесот



netstat = приказ на сите мрежни конекции на системот

- Приказ на сите TCP, UDP и UNIX socket конекции
- Ги прикажува и оние sockets кои слушаат односно чекаат конекции

- Доколку не се наведат аргументи, netstat прикажува листа на open sockets

  - -r прикажува рутирачки табели на јадрото
  - -g прикажува групи за IPv4 и IPv6
  - -I прикажува мрежни интерфејси
  - -s прикажува вкупна статистика за секој протокол

netstat

- -c = (continuous) продолжување на прикажување на информациите на една секунда
- -e = (extend) прикажува детални информации
- -p = (program) прикажува PID и името на програмата за секој socket
- -l = (listening) прикажување на listening sockets
- -a = (all) прикажување на сите sockets (listening & non listening)

```
sagar@LHB:~$ netstat -a | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:6463          0.0.0.0:*               LISTEN
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN
tcp        0      0 LHB.bbrouter:33596      server-13-227-138:https ESTABLISHED
tcp        0      0 LHB.bbrouter:54328      server-108-159-61:https ESTABLISHED
tcp        0      0 LHB.bbrouter:51556      server-52-222-144:https ESTABLISHED
tcp        0      0 LHB.bbrouter:41232      unn-169-150-247-3:https ESTABLISHED
tcp        0      0 LHB.bbrouter:34936      bom12s15-in-f10.1:https ESTABLISHED
tcp        0      0 LHB.bbrouter:41222      unn-169-150-247-3:https ESTABLISHED
tcp        0      0 LHB.bbrouter:49608      bom07s20-in-f4.1e:https ESTABLISHED
tcp        0      0 LHB.bbrouter:52314      bom07s29-in-f19.1:https ESTABLISHED
tcp        0      0 LHB.bbrouter:40234      bom12s15-in-f3.1e:https ESTABLISHED
tcp        0      0 LHB.bbrouter:49614      bom07s20-in-f4.1e:https ESTABLISHED
tcp        0      0 LHB.bbrouter:41054      ec2-52-9-108-19.u:https ESTABLISHED
tcp        0      0 LHB.bbrouter:43968      ec2-35-83-22-170.:https ESTABLISHED
tcp        0      0 LHB.bbrouter:49626      ec2-3-7-209-31.ap:https ESTABLISHED
tcp        0      0 LHB.bbrouter:56420      ec2-65-1-124-112.:https ESTABLISHED
```

ifconfig = (Interface Configuration)

- Доделување адреси на мрежни интерфејси и конфигурирање/приказ на детали за мрежната конфигурација
- Синтакса

```
ifconfig [option] [interface]
```

- Опции
  - -a = приказ на сите интерфејси од системот
  - -d = приказ на оние интерфејси кои се исклучени односно down
  - -l = листа на сите достапни интерфејси без дополнителни информации
  - -u = приказ на оние интерфејси кои се активни односно up

ifconfig

```
eth0      Link encap:Ethernet  HWaddr 08:00:27:6B:21:1C
          inet addr:10.0.2.15  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe6b:211c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
          RX packets:19157 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8596 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22458220 (21.4 MiB)  TX bytes:1262046 (1.2 MiB)

eth0:1    Link encap:Ethernet  HWaddr 08:00:27:6B:21:1C
          inet addr:192.168.1.101  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

ПРАШАЊА?