ابزارها

بررسی

Grep, sed and awk

در

پوسته

Shell Programming

محمد عزیزی

azizikam@yahoo.com

مرداد ۱۳۸۳

training@farsilinux.org

Shell programming

ابزارها
GREP

- Get Regular ExPressions یا
- Global Regular Expression Print

- توسط نوشته شد.
- جست و جو به دنبال یک کلمه یا رشته در فایل(ها).
  - اگر تطبیقی پیدا شد خروجی را در خروجی استاندارد نمایش و یا با توجه به گزینه خاص عمل متناسب را انجام می دهد.
- چون newline برای جدا کردن خطوط در فایل استفاده می شود، راهی برای جست و جو به دنبال newline وجود ندارد.

ابزارها
GREP(cont)

$ grep [options] regexp file(s)

-c تعداد خطوط تطبیق شده را نمایش می دهد

-i از حالت حساس به نوع حرف چشم پوشی می کند

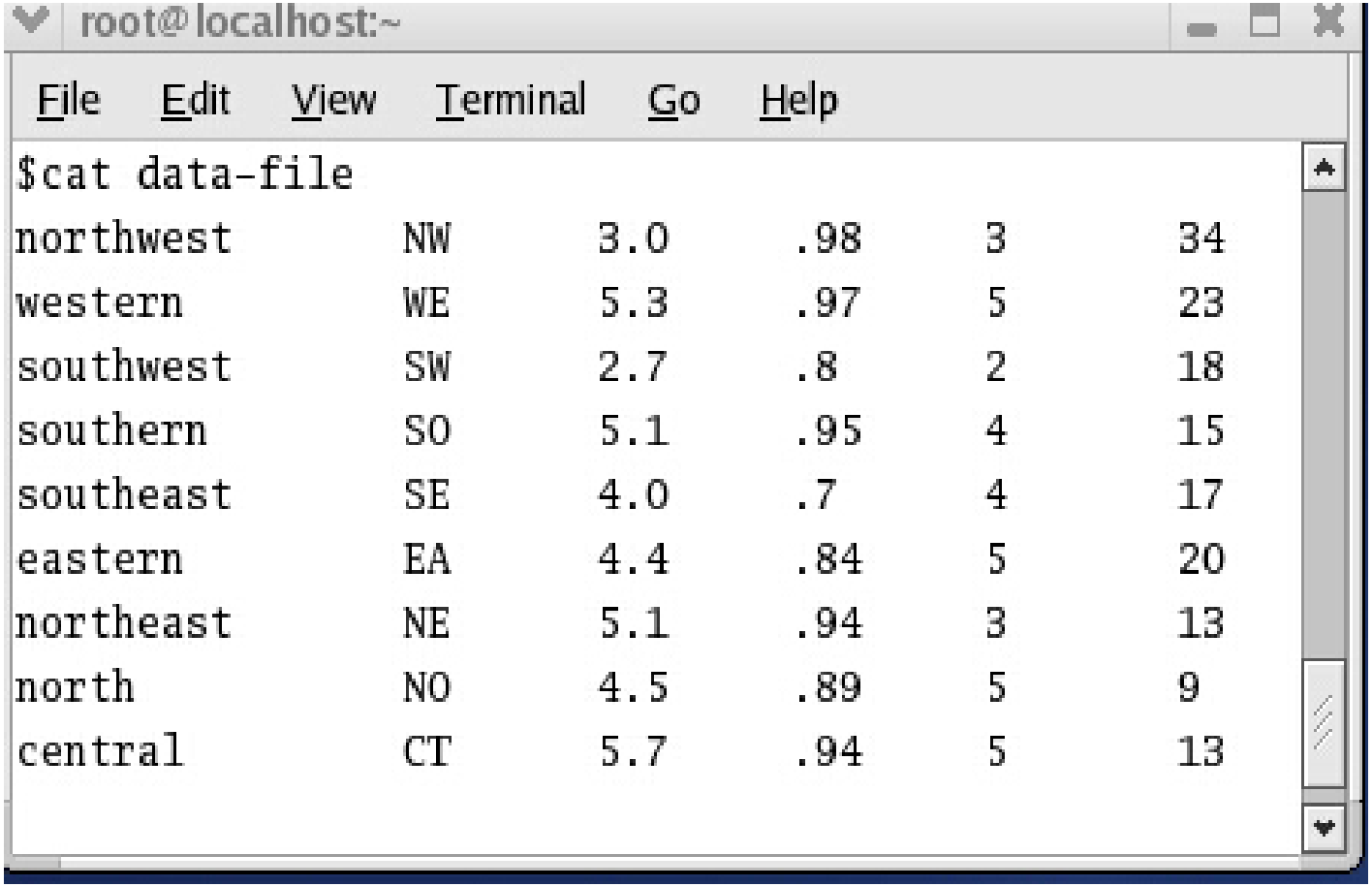
-l نمایش لیستی از نام فایلها به همراه تطبیق

-n نمایش خطوطی که تطبیقی در آنها پیدا نشده است

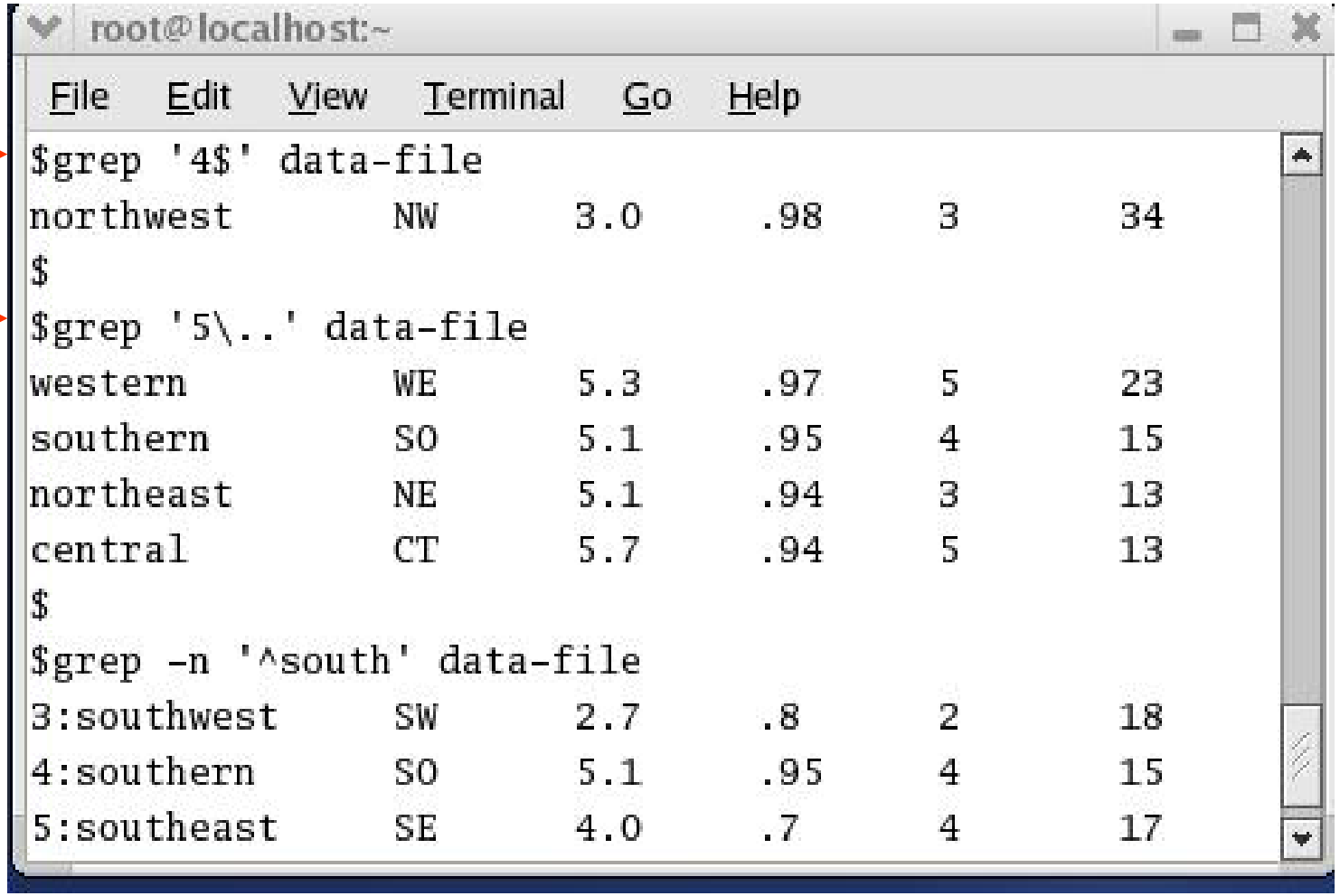
-s شماره خط تطبیق شده را نیز چاپ می کند.

فقط در صورت وجود خطا، پیام خطا را چاپ می کند

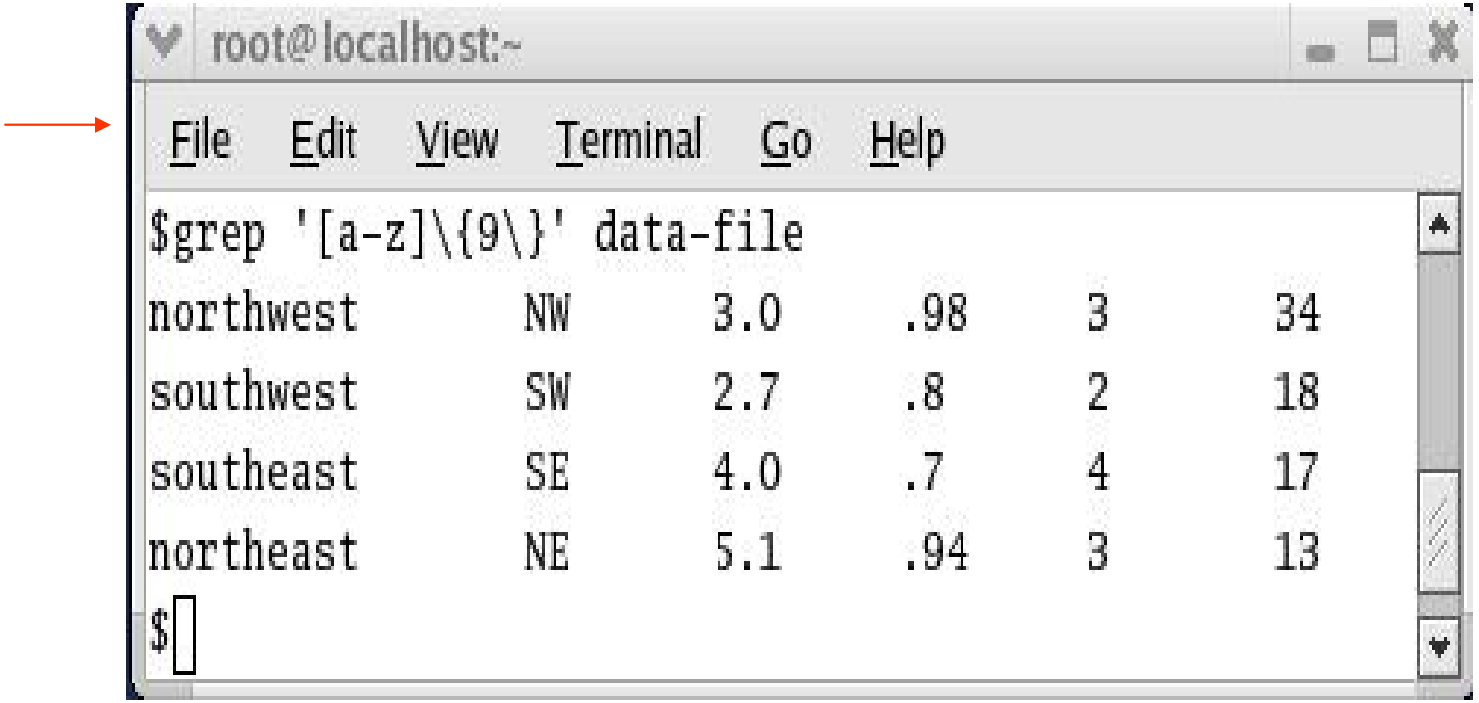
ابزارها
GREP(cont)



ابزارها
GREP(cont)



ابزارها
GREP(cont)



ابزارها
GREP(cont)

- یک اسکریپت بنویسید که مشخص کند آیا شخص خاصی در سیستم وجود دارد یا خیر؟


```
$ cat user-logon
#! /bin/bash
If  who | grep  –s  “$۱”  > /dev/null  ; then
      echo “$۱ is logged in”
else
   echo “$۱ is not logged in”
fi
```

ابزارها
GREP(cont)

- یک اسکریپت بنویسید که منتظر ورود شخص خاصی به سیستم باشد.

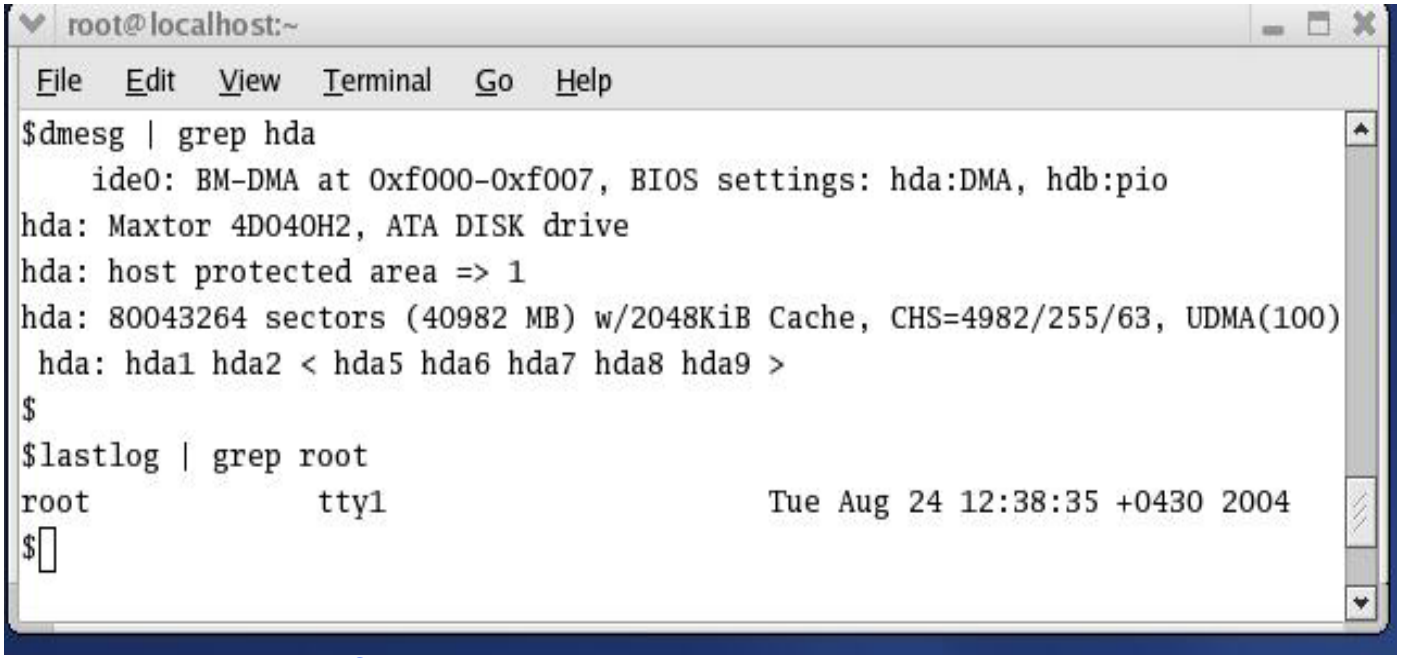
```
$ cat wath-for-in
#! /bin/bash
case $# in
    ١)  ;;
    ٢)  echo “Usage: watch-for-in username”
        exit ١
        ;;
esac
```

ابزارها
GREP(cont)

```
until  who | grep  –s “$۱ ”  > /dev/null
do
    sleep ۶۰
done
echo “$۱ has logged on”
exit ۰
```

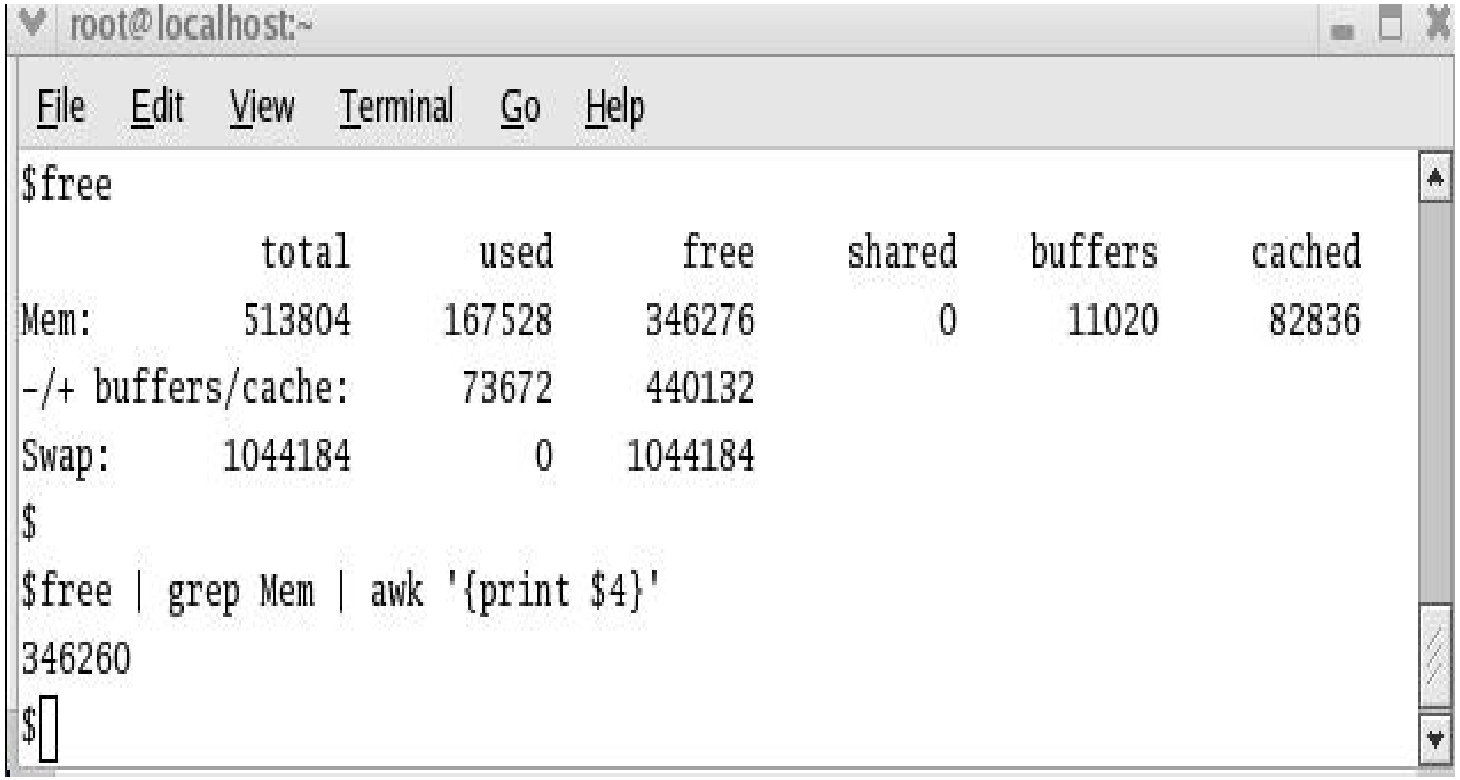
ابزارها
GREP(cont)

- dmesg تمام پیامهای سیستم در هنگام بوت را در خروجی استاندارد نمایش می دهد.
- lastlog آخرین زمان login همه کاربران را نمایش می دهد.



ابزارها
GREP(cont)

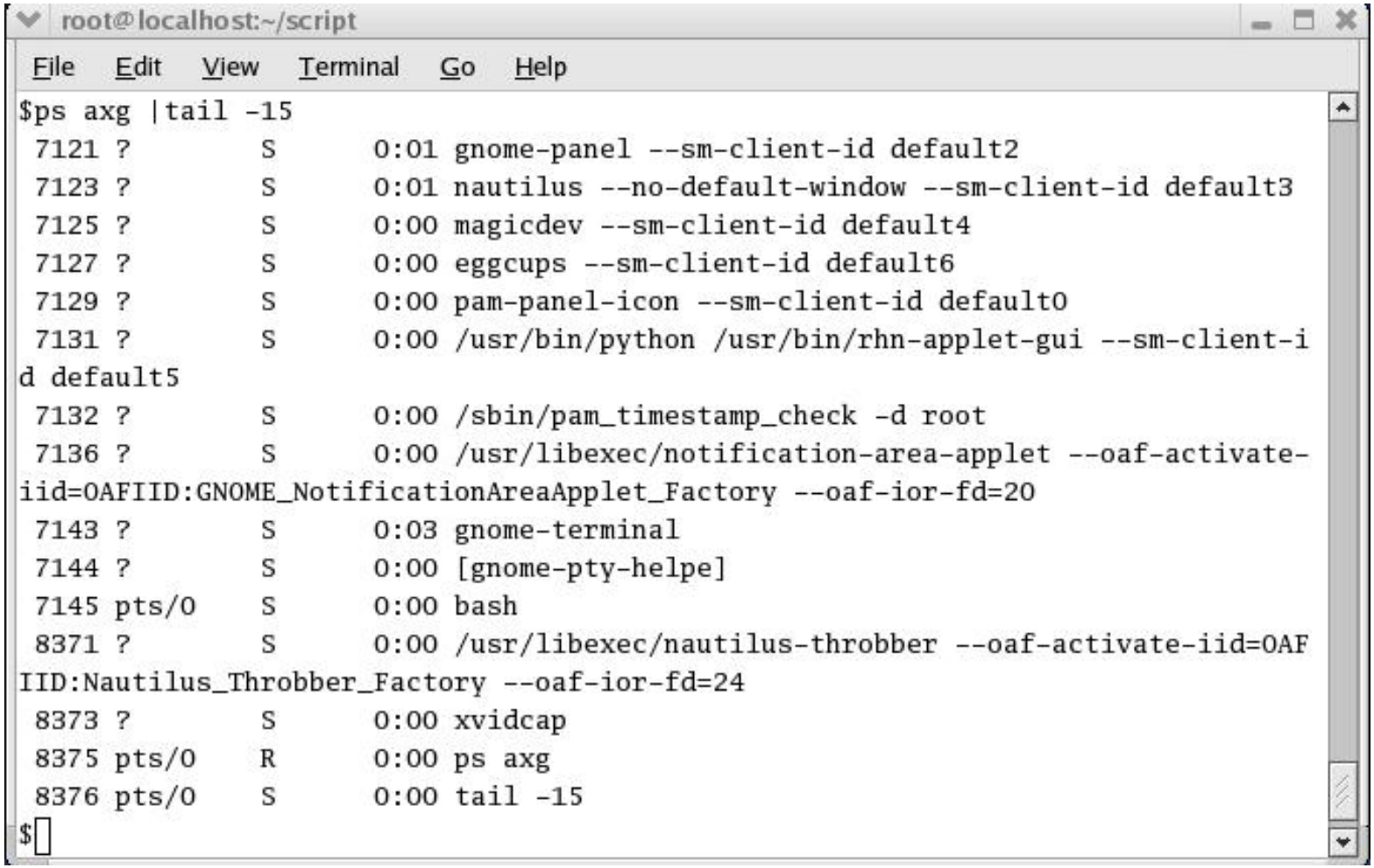
- فرمان free حافظه و cache مصرفی را به صورت جدول نمایش می دهد.



ابزارها
GREP(cont)

- با توجه به شکل زیر، اسکریپتی بنویسید که نام یک پردازه و email شخصی را گرفته و در صورت اجرا شدن آن پردازه به شخص خبر دهد، ضمناً مانع اجرا شدن آن پردازه شود.

ابزارها
GREP(cont)



ابزارها
GREP(cont)

root@localhost:~/script

File Edit View Terminal Go Help

```
$cat watcher
#! /bin/bash
echo -n "Enter a process name to watch for: "
read processname
echo -n "Enter a email address to warn when the process runs: "
read email
echo "Press CTRL C to end"
while true
do
  until  ps axg | grep "$processname" | grep -s -v "grep" |
        grep -s -v "watcher" > /dev/null
  do
    sleep 5
  done
  dayandtime=`date`
 echo "$processname was running on $dayandtime" >> \
       /var/log/watchprocess
 ps axgu | grep "$processname" | grep -v "grep" | /bin/mail -s "run!" "$email"
 killall -9 $processname
done
$
```

ابزارها
AWK

- در سال ۱۹۹۷ توسط

Alfred V. Aho
Peter J. Weinberger and
Brian W. Kernigan

طراحی و پیاده سازی شد.

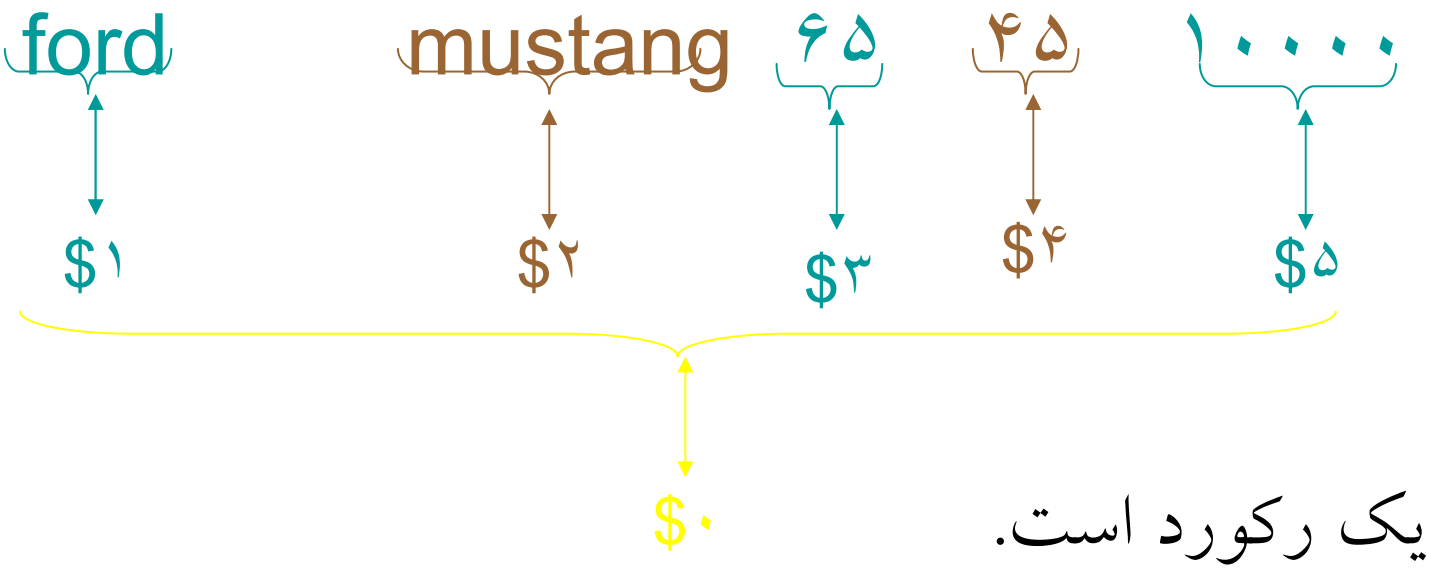
- یک زبان برنامه نویسی پردازش داده و تولید گزارش می باشد.
- معمولاً برای انجام عملیات ساده از آن در خط فرمان استفاده می شود.
- همچنین می توان آن را برای کاربردهای بزرگتر در قالب یک برنامه یا اسکریپت نوشت.

ابزارها
AWK(cont.)

- با awk آدرس دهی در سطح فیلد خواهیم داشت.
- ورودی awk می تواند
  - از فایل(ها) بیاید.
  - از pipe یا redirection بیاید.
  - مستقیماً از ورودی استاندارد بیاید.
- awk ورودی خود را خط به خط برای یافتن یک pattern جست و جو کرده و در صورت مشاهده تطبیق متناظر را بر آن خط اعمال می کند.
  - اگر pattern مشخص نشده باشد، action متناظر بر روی تمام خطوط اعمال خواهد شد.

ابزارها
AWK

```
$ cat cars | head -١
```

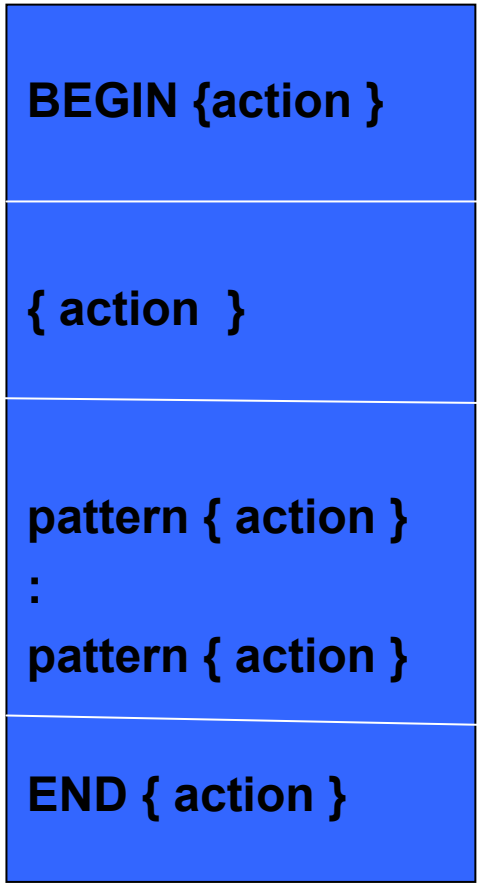


- هر خط یک رکورد است.
- هر رکورد از یک فیلد یا بیشتر تشکیل شده است.

ابزارها
AWK

- جدا کننده فیلدها را می توان هر چیزی تعیین کرد حتی یک عبارت منظم.
- جدا کننده پیش فرض فیلدها، <tab> می باشد.
- به فیلدها می توان به ترتیب با $۱، $۲ و ... رجوع نمود.
- $۰ به کل رکورد رجوع می کند.

ابزارها
AWK



- اختیاری
- اختیاری

ابزارها
AWK

- چندین روش برای اجرای یک برنامه awk وجود دارد:

➢ awk ‘program’ input_file(s)

➢ program و input_file(s) هر دو به عنوان آرگومان خط فرمان می باشند.

➢ awk ‘program’

➢ program به عنوان آرگومان خط فرمان

➢ ورودی از

- بنابراین می تواند در pipe به همراه فیلترهای دیگر استفاده شود

ابزارها
AWK

- awk –f program_file_name input_file(s)
  - برنامه از یک فایل خوانده می شود.

ابزارها
AWK

- awk مجموعه ای از خطوط ورودی را به ترتیب پویش کرده و اگر تطبیقی با pattern پیدا شد action متناظر را روی آن اجرا می کند.
- pattern می تواند:

  - /regular expressions/
  - relational expressions
  - pattern, pattern

ابزارها
AWK

- awk می تواند هر فیلد را با یک عدد یا عبارت منظم مقایسه کند.

➢Variable operator pattern

~	⟶	دو رشته مساوی
!~	⟶	دو رشته نامساوی
&&	⟶	Logical AND
||	⟶	Logical OR
!	⟶	Logical NOT

ابزارها
AWK

- Variable operator pattern

<	⟵	کوچکتر
<=	⟵	کوچکترمساوی
>	⟵	بزرگتر
>=	⟵	بزرگترمساوی
=	⟵	تساوی عددی
!=	⟵	نامساوی عددی

ابزارها
AWK

- محدوده ای از خطوط را انتخاب می کند که:
  - اولین خط آن با pattern۱ تطبیق شود.
  - آخرین خط آن با pattern۲ تطبیق شود.
- ممکن است چندین گروه از خطوط را برگرداند.

ابزارها
AWK

- action شامل یک یا چند فرمان، تابع یا مقداردهی متغیر احاطه شده در {} است که با newline یا semicolon از هم جدا شده اند.
- فرمانها به چهار دسته تقسیم می شوند:
  - مقدار دهی متغیر
  - فرمانهای نمایش
  - توابع توکار
  - دستورهای کنترل جریان

ابزارها
AWK

متغیر	معنی
NR	شماره خط جاری
FS	جدا کننده فیلدها(پیش فرض newline و tab)
OFS	جدا کننده فیلدهای خروجی(پیش فرض blank)
NF	تعداد فیلدها
RS	جداکننده رکورد(پیش فرض newline)
FILENAME	نام فایل ورودی

ابزارها

awk

- فرمان print ابزار خروجی ساده awk است.
- می توانید خروجی این فرمان را با استفاده از متاکاراکترهای > و >> به یک فایل هدایت کرد.

```
$ls –l | awk ‘{print $۲}’
```

ابزارها

awk

- این فرمان نمایش یک خروجی قالب بندی شده را ممکن می سازد.

- printf(“format string”, var۱, var۲, …)
- Foramt specifiers
  - %c -
  - %d -
  - %f -
  - %s -
  - \n -
  - \t -

ابزارها

awk

- Format modifiers
  - -
  - n
  - .n

ابزارها

awk

- $ echo “Linux” | awk ‘{printf(“%-۱۵s\n”,$۱)}’

۱۰ کاراکتر

باعث می شود اعلان پیش فرض
پوسته در خط بعد نمایش داده شود.

- $ echo “Linux” | awk ‘{printf(“٪ ۱۵s\n”,$۱)}’

۱۰ کاراکتر

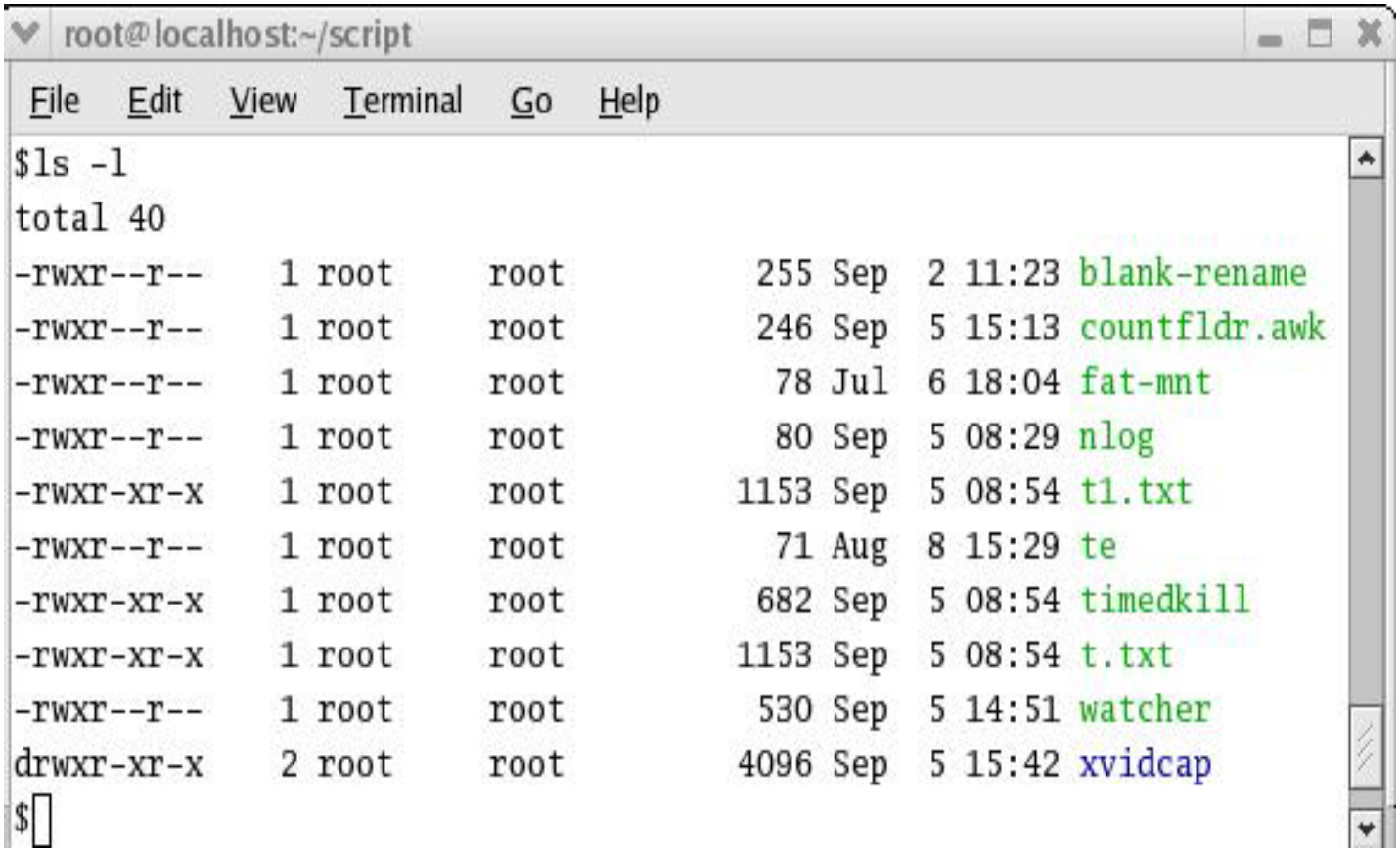
- $ echo ۲۷,۸۷۶ | awk ‘{printf(“٪ ۱۵,۲f\n”,$۱)}’

۱۰ کاراکتر

ابزارها

awk

- با توجه به شکل به سؤالات زیر پاسخ دهید.



ابزارها

awk

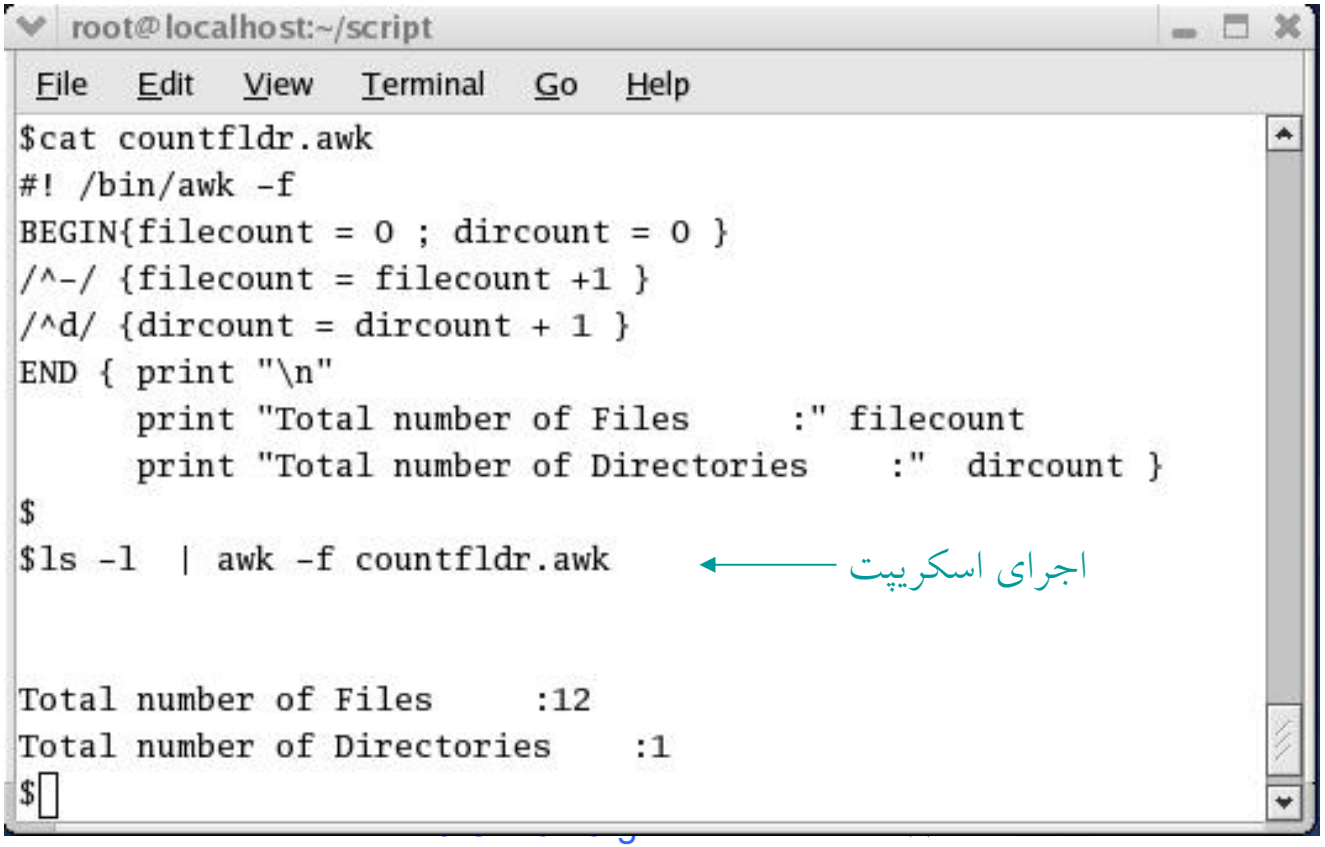
- $ ls –l | awk ‘/^d/’ { print “rm –r “$۹ }’ | bash

- $ ls –l | grep –v ‘^d’ |
  > awk ‘{ print “rm –f”$۹}’ | bash

ابزارها

awk

- اسکریپتی بنویسید که تعداد فایلها و دایرکتوریهای شاخه جاری را نمایش دهد:



ابزارها

awk

- length([argument])
  - این تابع طول رشته argument را می دهد.
  - اگر argument نیاید، این تابع طول $۰ را می دهد.
- Index()
  - تابع index(string,target) موقعیت اولین رخداد target در string را می دهد.
- substr()
  - تابع substr(string,start[,length]) قسمتی از string که با start شروع می شود به طول length کاراکتر را بر می گرداند.

ابزارها

awk

ابزارها
awk

- Syntax
- if (condition is true or non zero)

  statement۱

else

  statement۲

ابزارها
awk

A is less than B
A is less than or equal B
A equals B
A is greater than B
A is greater than or equal B
A is not equal B
A contains the regular expression RE

ابزارها
awk

- Syntax
- while (condition is true or non zero)
  statement

ابزارها
awk

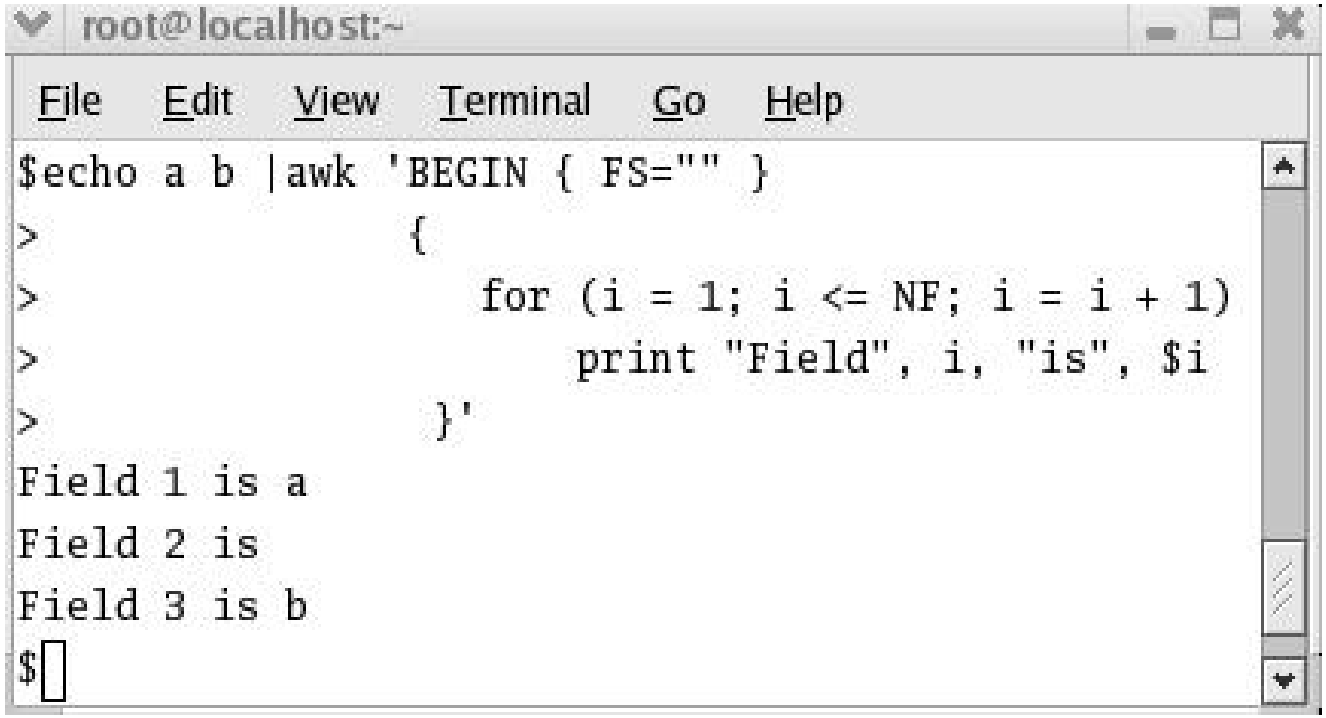
- for (expression۱;expression۲;expression۳)
  statement

- عملکرد حلقه بالا شبیه حلقه while باشد:

```
expression۱
while (expression۲){
  statement(s)
  expression۳
}
```

ابزارها
awk

- دستورات زیر باعث می شوند هر کاراکتر فیلد جداگانه ای محسوب شود:



ابزارها
awk

- do{
  statement(s)
  } while(condition is false or non zero)
  - در حلقه while do ابتدا condition تست می شود و بعد بنده حلقه اجرا می گردد.
  - در حلقه do while ابتدا بدنه حلقه اجرا و بعد condition تست می شود.

ابزارها
awk

- باعث خروج از بدنه حلقه می شود.
  - و باعث خروج از اسکریپت awk نمی شود.
- کنترل به خط بعد از بدنه حلقه منتقل می شود.

- باعث می شود که awk به ازاء مقدار کنونی از اجرای باقیمانده بدنه حلقه خودداری کرده و تکرار بعدی آغاز شود.

- باعث می شود اسکریپت awk از اول اجرا شود.

ابزارها

awk: example

plym	fury	۷۷	۷۳	۲۵۰۰
chevy	nova	۷۹	۶۰	۳۰۰۰
ford	mustang	۶۵	۴۵	۱۰۰۰۰
volvo	gl	۷۸	۱۰۲	۹۸۵۰
ford	ltd	۸۳	۱۵	۱۰۵۰۰
chevy	nova	۸۰	۵۰	۳۵۰۰
fiat	۶۰۰	۶۵	۱۱۵	۴۵۰
honda	accord	۸۱	۳۰	۶۰۰۰
ford	thundbd	۸۴	۱۰	۱۷۰۰۰
toyota	tercel	۸۲	۱۸۰	۷۵۰

ابزارها

awk: example(cont.)

```
chevy     nova ۷۹   ۶۰   ۳۰۰۰
chevy     nova ۸۰   ۵۰   ۳۵۰۰
```

```
chevy     nova ۷۹   ۶۰   ۳۰۰۰
chevy     nova ۸۰   ۵۰   ۳۵۰۰
```

ابزارها

awk: example(cont.)

۷۷	plym
۷۹	chevy
۶۵	ford
۷۸	volvo
۸۳	ford
۸۰	chevy
:	:

awk example(cont)

۷۷plym

۷۹chevy

۶۵ford

۷۸volvo

۸۳ford

۸۰chevy

: :

ابزارها

awk: example(cont.)

```
۷۷    plym
۷۹    chevy
۶۵    ford
۷۸    volvo
۸۳    ford
۸۰    chevy
:     :
۷۳۰۵۰
```

ابزارها

awk: example(cont.)

volvo	gl	۷۸	۱۰۲	۹۸۵۰
toyota	tercel	۸۲	۱۸۰	۷۵۰
plym	fury	۷۷	۷۳	۲۵۰۰
chevy	nova	۷۹	۶۰	۳۰۰۰
chevy	nova	۸۰	۵۰	۳۵۰۰
honda	accord	۸۱	۳۰	۶۰۰۰

ابزارها

awk: example(cont.)

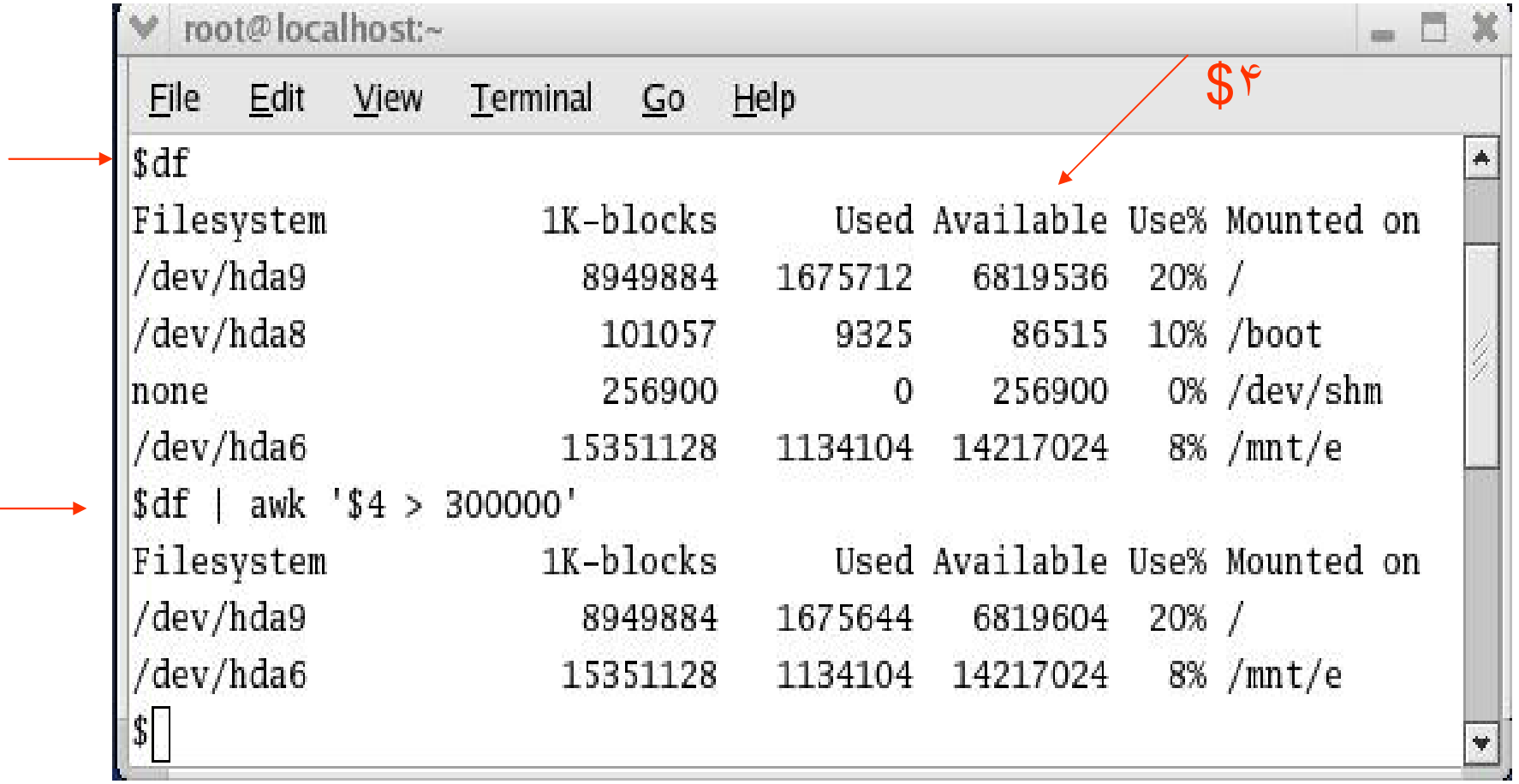
honda	accord	۸۱	۳۰	۶۰۰۰
chevy	nova	۷۹	۶۰	۳۰۰۰
ford	mustang	۶۵	۴۵	۱۰۰۰۰
chevy	nova	۸۰	۵۰	۳۵۰۰
fiat	۶۰۰	۶۵	۱۱۵	۴۵۰
honda	accord	۸۱	۳۰	۶۰۰۰
ford	thundbd	۸۴	۱۰	۱۷۰۰۰

گروه ۱

گروه ۲

ابزارها

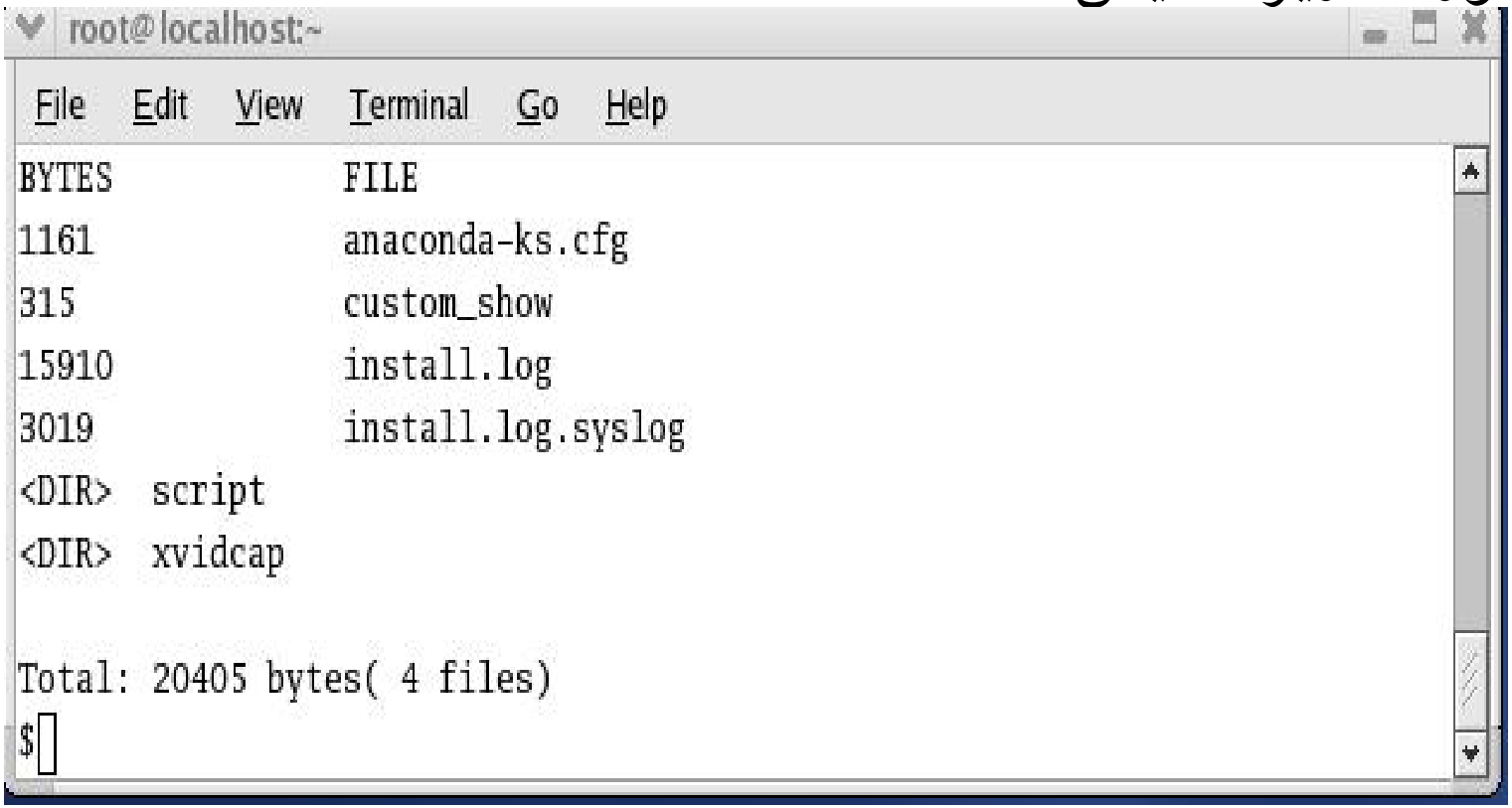
awk: example(cont.)



ابزارها

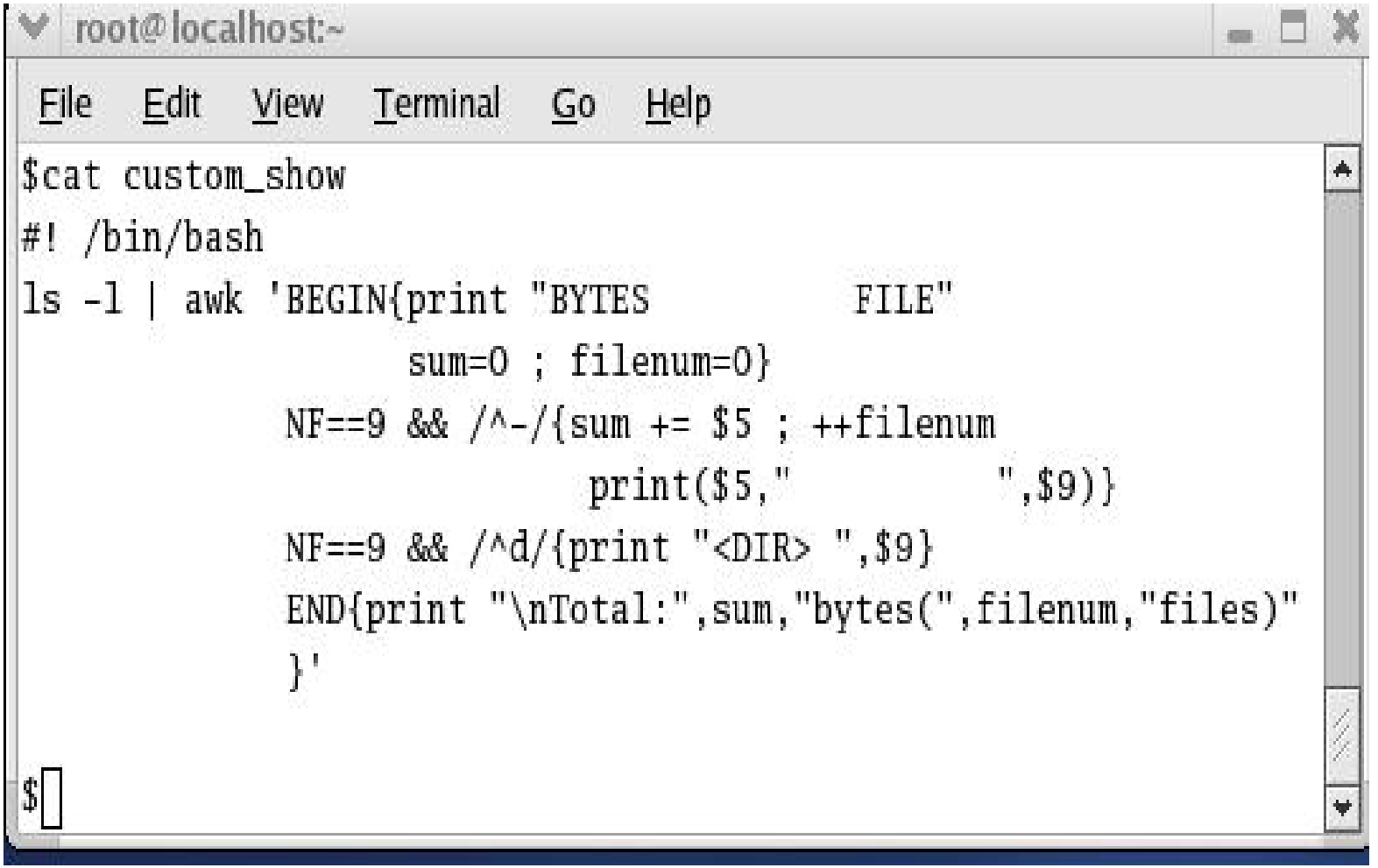
awk: example(cont.)

- یک برنامه awk بنویسید که محتوای دایرکتوری جاری را به صورت زیر نمایش دهد.



ابزارها

awk: example(cont.)



ابزارها

awk: example(cont.)

- یک اسکریپت بنویسید که مسیری را از کاربر بعنوان پارامتر گرفته و بزرگترین فایل در آن مسیر را نمایش دهد.

```
$ cat find-big-file
#! /bin/bash
#usage: find-big-file  path
echo –e “Please enter your path:\c”
read path
if [ ! -d “$path” ]
 then
```

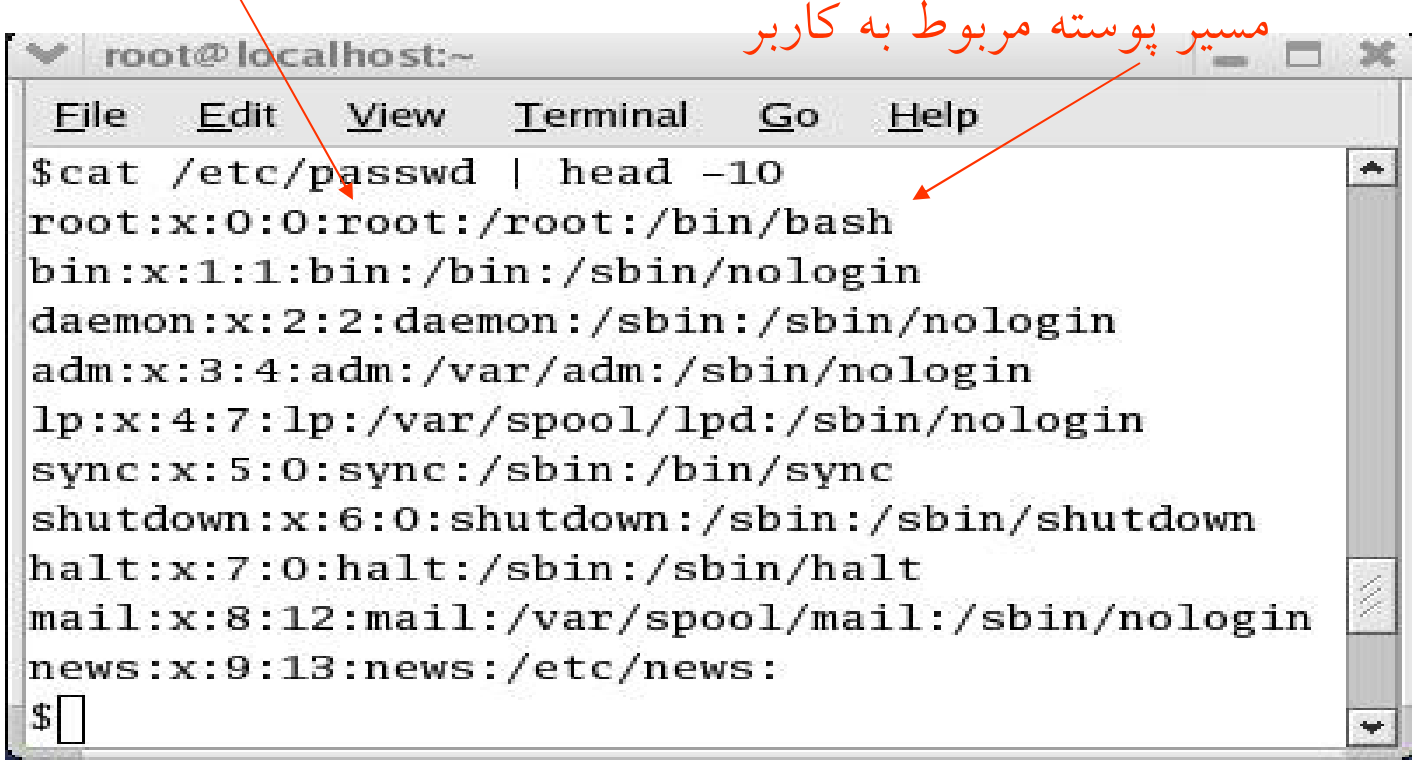
awk: example(cont.)

```
    echo “Your path is not valid.”
    exit ١
fi
echo “The biggest file in $path is”
big=`ls -l | awk '/^-/ {print $٥}' | \
sort –nr | head -١`
ls -l | grep $big | awk '/^-/ { print $٩ }‘
exit ٠
```

ابزارها

awk: example(cont.)

- با توجه به فایل /etc/passwd که ده خط اول آن در زیر آمده است، به سؤالات زیر پاسخ دهید:



ابزارها

awk: example(cont.)

- نام واقعی تمام کاربرانی که به A شروع می شوند:

```
$ awk      ‘$۵ ~ /^A/ { print }’ /etc/passwd
```

- نمایش تعداد کاربران از هر نوع پوسته sh، bash، csh و ksh:

```
$cat count_shell_users.awk
BEGIN {FS=“:”}
```

ابزارها

awk: example(cont.)

```
/\/bash$/ {bash++}
/\/sh$  / { sh++}
/\/csh$/ { csh++}
/\/ksh$/ { ksh++}
END { print bash,"Users of bash."
      print sh,"Users of sh"
      print csh,"Users of csh."
      print ksh,"Users of ksh."}
```

ابزارها

awk: example(cont.)

```
$awk -f count_shell_users.awk  /etc/passwd
```

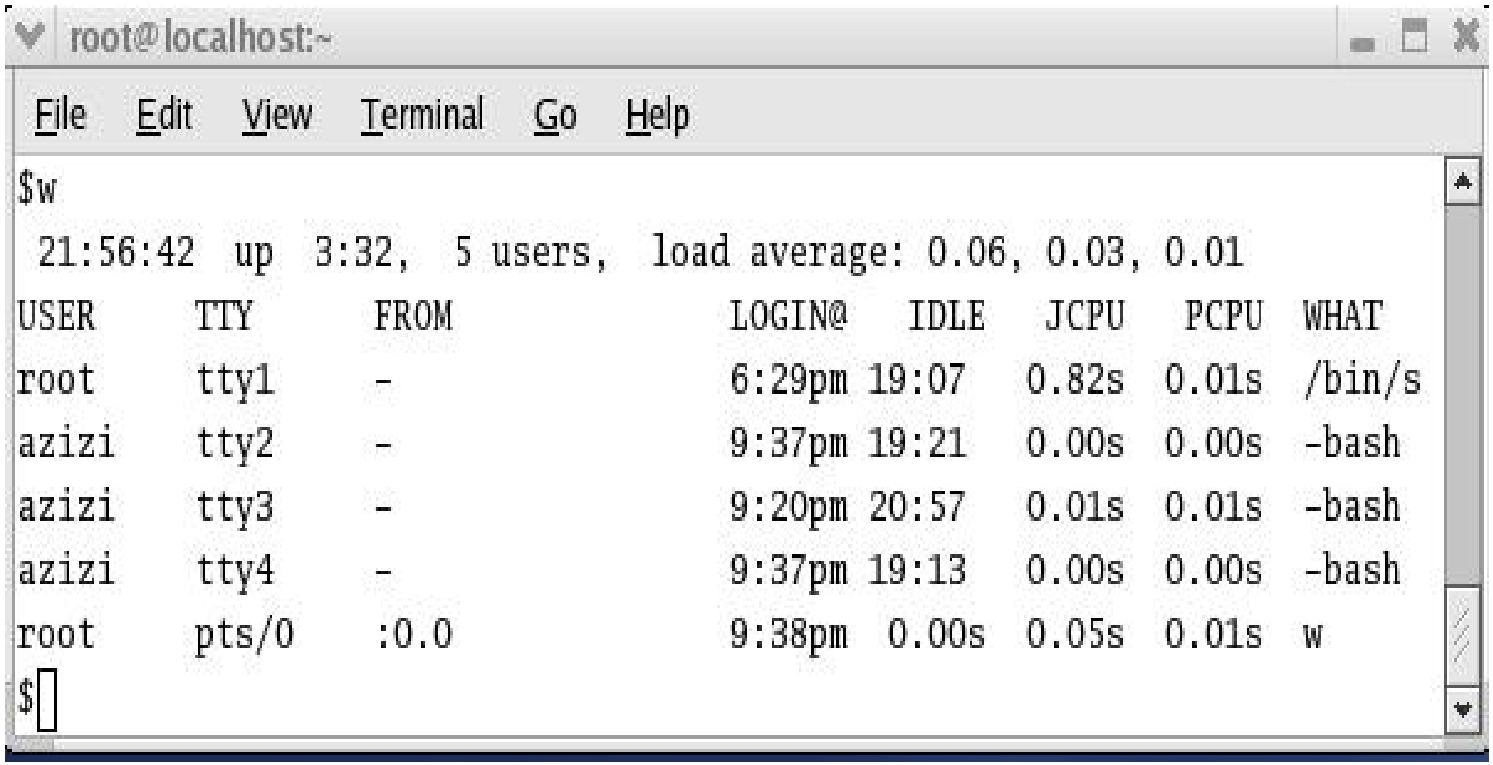
نمایش هر قسمت از متغیر PATH در خطی جداگانه:

```
$ echo $PATH | awk 'BEGIN { RS=:}
                    { print $0 }'
```

ابزارها

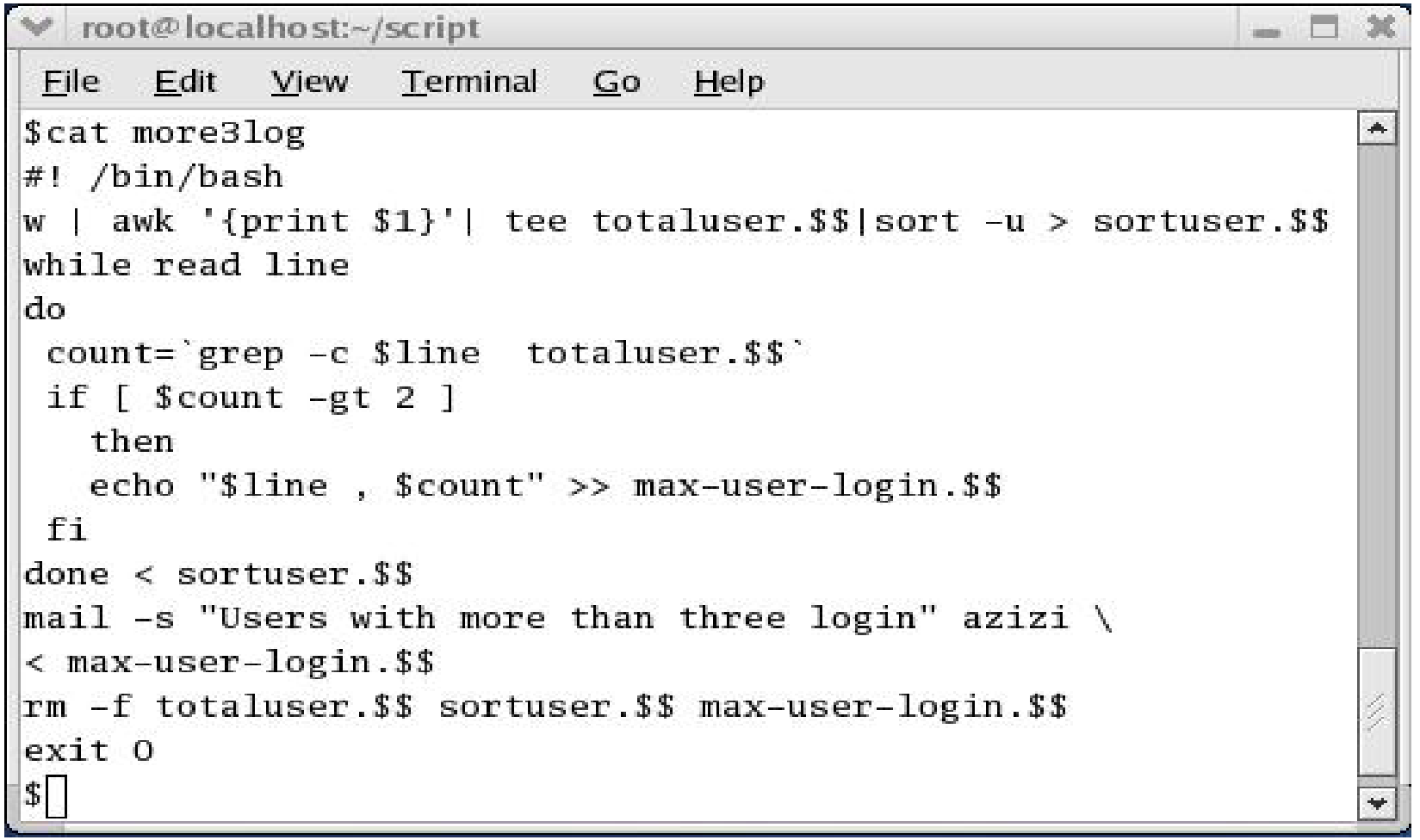
awk: example(cont.)

- اسکریپتی بنویسید که اگر کاربری بیشتر از سه login داشت، نام آن کاربر را به مسئول سایت email بزند.



ابزارها

awk: example(cont.)



ابزارها

sed, The Stream EDitor

- ویراستاری خاص منظوره است که فرمانهای خود را فقط از خط فرمان و یا از یک اسکریپت می تواند بگیرد و به صورت محاوره ای نمی تواند استفاده شود.
- ورودی ها همگی از ورودی استاندارد و خروجی نیز در خروجی استاندارد نوشته می شود.
- بنابراین فایل ویرایش شده هیچ تغییری نمی کند، اما فایل ورودی به همراه تغییرات در خروجی استاندارد نوشته می شود.

ابزارها
sed Syntax

- $ sed [-n] [-e] [‘command’] [file…]
- $ sed [-n] [-f scriptfile] [file …]

فقط خطوطی که فرمان p یا s بر روی آنها اجرا می شود را نمایش می دهد

آرگومان بعدی یک فرمان ویرایش است و نام فایل نمی باشد.

آرگومان بعدی نام فایلی است که حاوی فرمانهای ویرایش است.

ابزارها
sed Syntax(cont.)

- اگر لازم است تغییرات حاصل از sed را ذخیره کنید باید خروجی استاندارد را به یک فایل redirect نمایید:

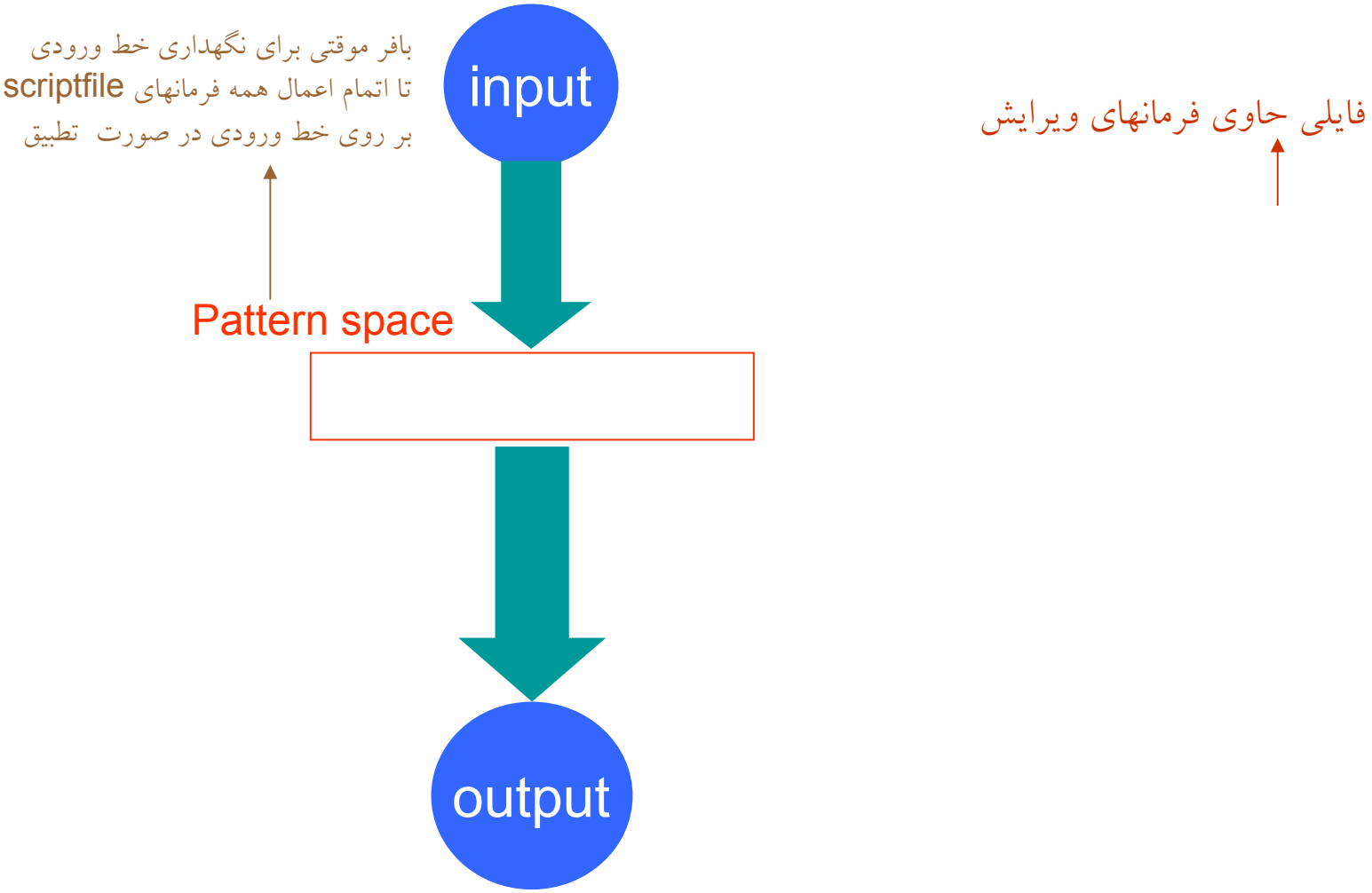
```
$ sed  –f  scriptfile editfile > outputfile
```

- شکلهای مختلف فراخوانی sed همگی مشابه هستند:

```
$sed   [options]  script  file_argument(s)
```

ابزارها

نحوه برخورد sed با فایلها



ابزارها

Sed script

- اسکریپت یک فایل حاوی فرمانهای ویرایش است.
- هر فرمان شامل یک آدرس و یک action می باشد که آدرس pattern(عبارت منظم) نیز می تواند باشد.
- هر وقت یک خط از فایل ورودی خوانده شد sed اولین فرمان از اسکریپت را خوانده و آدرس یا pattern را با آن مقایسه می کند:
  - اگر تطبیقی رخ داد فرمان اجرا می شود.
  - اگر تطبیقی رخ نداد از فرمان صرفنظر می شود.
  - sed سپس همین عمل را برای فرمانهای بعدی در اسکریپت تکرار می کند.

ابزارها

Sed script(cont.)

- وقتی به انتهای اسکریپت رسید آنگاه sed خط جاری را در خروجی نمایش می دهد، مگر اینکه گزینه n– فعال باشد.
- sed سپس خط ورودی بعدی را می خواند و دوباره از ابتدای فایل اسکریپت شروع به خواندن فرمانها می کند.
- فایل ورودی اولیه بدون تغییر می ماند و هر بار sed پردازش خود را به ترتیب بر روی یک کپی از خطوط آن انجام و نتیجه را در خروجی استاندارد نمایش می دهد.

ابزارها

Sed command(cont.)

- شکل کلی فرمانهای sed به صورت زیر است:
  - [address[, address]] [!] command [arguments]
- sed هر خط ورودی را در pattern space کپی می کند:
  - اگر آدرس فرمان با خط موجود در pattern space تطبیق شد، فرمان بر آن خط اعمال می شود.
  - اگر فرمان آدرسی نداشت، بر هر خط که در pattern space بیاید اعمال می شود.
  - اگر فرمانی خط موجود در pattern space را تغییر داد، فرمان بعدی بر خط تصحیح شده اعمال خواهد شد.
- وقتی همه فرمانها خوانده شدند، خط موجود در pattern space در خروجی نوشته شده و خط جدید به داخل pattern space خوانده می شود.

ابزارها
Sed addressing

- آدرس می تواند شماره خط و یا یک pattern که در اسلش(/pattern/) قرار گرفته است باشد.
- عبارت منظم pattern را توصیف می کند.
- بیشتر فرمانها دو آدرس را قبول می کنند
  - اگر فقط یک آدرس مشخص شده باشد فرمان فقط بر آن خط اعمال می شود.
  - اگر دو آدرس که با کاما متمایز شده اند مشخص شده باشد فرمان بر روی محدوده ای از خطوط بین اولین و دومین آدرس اعمال می شود.
- عملگر ! برای منفی کردن محدوده آدرس استفاده می شود.
  - مثلاً address ! command باعث می شود فرمان بر تمام خطوطی که آدرس آنها address اعمال شود.

ابزارها

Sed addressing(cont.)

- برای اعمال چندین فرمان بر یک آدرس از براکت { } استفاده می شود.

```
[/pattern/[,/pattern/]]{
        command۱
        command۲
        command۳
        }
```

ابزارها

Sed addressing(cont.)

- d — خط جاری را حذف می کند
- ۶d — شش خط را حذف می کند
- /^$/d — همه خطوط blank را حذف می کند
- ۱،۱۰d — خطوط یک تا ۱۰ را حذف می کند
- ۱,/^$/d — از خط یک تا اولین خط blank را حذف می کند
- /^$/,۱۰d — از اولین خط blank تا خط ۱۰ را حذف می کند
- /^$/,$d — از اولین خط blank تا خط آخر را حذف می کند

ابزارها

معرفی چند فرمان sed

- اگر چه sed فرمانهای ویرایش زیادی دارد اما ما زیر مجموعه کوچکی از آنها را بررسی می کنیم:
  - s - substitute
  - a - append
  - i - insert
  - c - change
  - d - delete
  - p - print
  - ! – Don't
  - r - read
  - w – write
  - t – transform

ابزارها

sed commands: substitute

- Syntax:

[flags]

- n عددی از یک تا ۵۱۲ که نشان می دهد کدامین رویداد pattern باید جایگزین شود.
- p محتوای pattern space را نمایش می دهد.
- g تمام رویدادهای pattern را جایگزین می کند.
- w file محتوای pattern space را در فایل file می نویسد.

sed commands: substitute(cont)

$ sed ‘s/is/IS/g’ sedin

th a test
for the
next sixty
seconds
th station will
conducting a test

this is a test
for the
next sixty
seconds
this station will
be conducting a test

ابزارها

sed commands:

- شکل این فرمانها نسبتاً عجیب است زیرا آنها را باید در چندین خط مشخص نمود.

- Append [address]a\
  text
- Insert [address]i\
  text
- Change [address(es)]c\
  text

ابزارها

sed commands: Delete

- Delete صفر، یک یا دو آدرس می تواند قبول کند و
  - Pattern space جاری را حذف کند.
  - Pattern space که با اولین آدرس تطبیق شد را حذف می کند.
  - خطوط در محدوده دو آدرس را حذف می کند.
- وقتی Delete اجرا شد فرمان دیگری به pattern space اعمال نمی شود. در عوض خط جدید به داخل pattern space خوانده شده و اسکریپت از بالا شروع به اجرا خواهد شد.
- Delete تمام خط را حذف می کند. برای حذف قسمتی از خط آن را با blank جایگزین کنید.

sed commands: Delete(cont.)

```
$ sed  ‘/^next/ d’ sedin
this is a test
for the
seconds
this station will
be conducting a test
$ sed /^next/,/^be/ d’ sedin
this is a test
for the
```

```
this is a test
for the
next sixty
seconds
this station will
be conducting a test
```

sed commands: Print

$ sed ‘/^for/,/^seconds/ p’ sedin

this is a test
for the

next sixty

seconds

this station will
be conducting a test

this is a test
for the
next sixty
seconds
this station will
be conducting a test

ابزارها

sed commands: Don’t

- اگر بعد از آدرس علامت تعجب بیاید، فرمان به تمامی خطوط به غیر از آن آدرس یا محدوده آدرس اعمال می شود.

```
$ sed  ‘۲،۳ !d’ sedin
this is a test
this station will
be conducting a test
```

```
this is a test
for the
next sixty
seconds
this station will
be conducting a test
```

ابزارها

sed commands: Read and Write

- با استفاده از این دو فرمان می توان مستقیماً با فایلها کار کرد.

➢ Read: [address]r filename

باید حداقل یک فاصله باشد.

➢ Write: [address,[address]]w filename

- آرگومان هر دو فرمان فقط نام فایل می باشد.
- فرمان read بعد از خواندن خط به آدرس address، شروع به خواندن فایل filename به داخل pattern space می کند.

ابزارها

sed commands: Read and Write(cont.)

- فرمان write آدرس یک خط(یا محدوده ای از خطوط) را گرفته و محتوای pattern space را به داخل فایل filenam می نویسد.
- در فرمان read اگر فایل filename موجود نباشد، sed پیغام خطایی نخواهد داد.
- در فرمان write اگر فایل filename موجود نباشد، ایجاد و در غیر اینصورت بازنویسی خواهد شد.

sed commands: Read and Write(cont.)

```
$ sed '۲ r test' sedin
```

this is a test
for the
insert text۱
insert text۲
insert text۳
next sixty
seconds

this is a test
for the
next sixty
seconds

Insert text۱
Insert text۲
Inset text۳

sed commands: Read and Write(cont.)

```
$ sed  '۲،۳ w test' sedin
this is a test
for the
next sixty
seconds
$ cat test
for the
next sixty
```

```
this is a test
for the
next sixty
seconds
```

ابزارها

sed commands: Transform

- این فرمان شبیه **tr** بوده و یک جایگزینی یک به یک(کاراکتر به کاراکتر) انجام می دهد.
- این فرمان صفر، یک یا دو آدرس می تواند قبول کند.

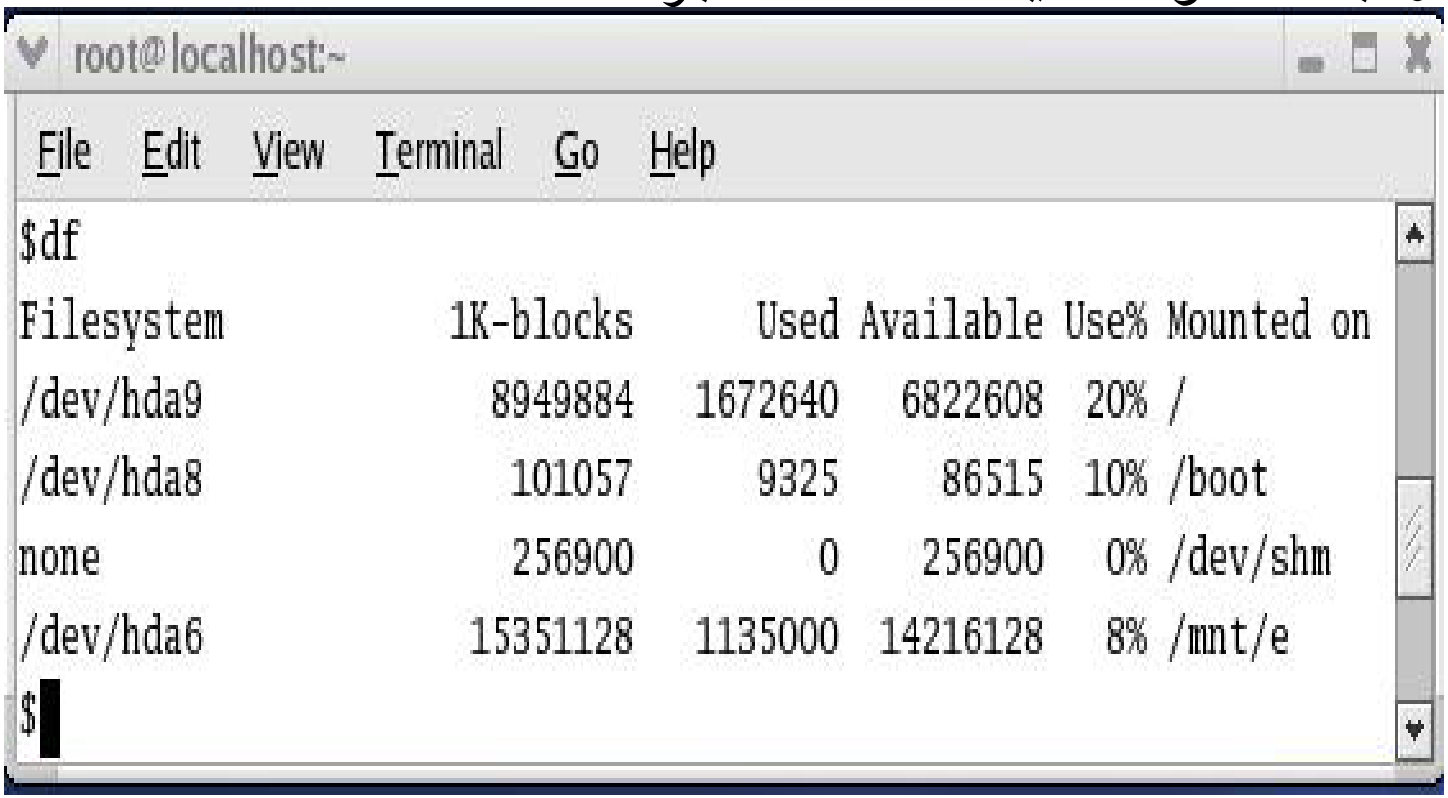
```
$ sed ‘y/th s/si  t/’ sedin
siht ht a sets
for sie
nexs thxsy
```

```
this is a test
for the
next sixty
```

ابزارها

sed examples

- اسکریپتی بنویسید که اگر پارتیشنی بیشتر از ۹۰٪ آن پر شد، نام آن را به مسئول سایت email بزند.



ابزارها

sed examples(cont.)

$cat fullpartition

```
#! /bin/bash
MAX=۹۰
df | sed ‘s/%/ /g’ |
   awk ‘{ if ( $۵ > $MAX) ; print $۶ }’ |
   mail –s “Full Partition Names” root
exit ۰
```