Chapter 3
UNIX Utilities for Power Users

YUAN LONG

CSC 3320 SYSTEM LEVEL PROGRAMMING

FALL 2016

Updated based on original notes from Raj Sunderraman and Michael Weeks

What will be covered?

Section	Utilities
Filtering files	**egrep, fgrep, grep**
Programmable text processing	**awk, sed**
Sorting files	**sort**
Archiving files	**tar**
Searching for files	**find**
Switching users	**su**
Scheduling commands	**crontab**

Filtering files

- grep, egrep, and fgrep
  - Filter out all lines that do not contain a **specified pattern** (i.e. output all lines containing a specified pattern)
  - grep -hilnvw '**pattern**' {fileName}*
  - egrep -hilnvw '**pattern**' {fileName}*
  - fgrep -hilnvw '**pattern**' {fileName}*
    - -h : do not list file names if many files are specified
    - **-i : ignore case**
    - -l : displays list of files containing pattern
    - **-n : display line numbers**
    - -v : displays lines that do not match the pattern
    - **-w : matches only whole words only**

Differences

- **grep** : pattern must be basic regular expression
- **fgrep** : pattern must be fixed string **FAST**
- **egrep** : pattern can be extended regular expression
  - -x option in **fgrep**: displays only lines that are exactly equal to string

Extended regular expressions:

+ matches one or more of the single preceding character
? matches zero or one of the single preceding character
| either or (ex. a* | b*)
() *, +, ? operate on entire subexpression not just on preceding character; ex. (ab | ba)*

grep Examples

$ cat grepfile

```
Well you know it's your bedtime,
So turn off the light,
Say all your prayers and then,
Oh you sleepy young heads dream of wonderful things,
Beautiful mermaids will swim through the sea,
And you will be swimming there too.
```

$ grep --color -n 'sw.*ng' grepfile

```
6:And you will be swimming there too.
```

$ grep --color -n 'a.' grepfile

```
3:Say all your prayers and then,
4:Oh you sleepy young heads dream of wonderful things,
5:Beautiful mermaids will swim through the sea,
```

grep Examples

grep pattern	Lines that match
`.nd`	Say all your prayers **and** then, Oh you sleepy young heads dream of w**ond**erful things, **And** you will be swimming there too.
`^.nd`	**And** you will be swimming there too.
`sw.*ng`	And you will be **swimming** there too.
`[A-D]`	**B**eautiful mermaids will swim through the sea, **A**nd you will be swimming there too.
`\.`	And you will be swimming there too**.**
`a.$`	Beautiful mermaids will swim through the se**a,**
`[a-m]nd`	Say all your prayers **and** then,
`[^a-m]nd`	Oh you sleepy young heads dream of w**ond**erful things, **And** you will be swimming there too.

egrep Examples

$ cat grepfile

```
Well you know it's your bedtime,
So turn off the light,
Say all your prayers and then,
Oh you sleepy young heads dream of wonderful things,
Beautiful mermaids will swim through the sea,
And you will be swimming there too.
```

$ egrep --color -n 'sw.*ng' grepfile

```
6:And you will be swimming there too.
```

$ egrep --color -n 's.+w' grepfile

```
4:Oh you sleepy young heads dream of wonderful things,
5:Beautiful mermaids will swim through the sea,
```

egrep Examples

grep pattern	Lines that match
`s.*w`	Oh you **sleepy young heads dream of w**onderful things, Beautiful mermaid**s will sw**im through the sea, And you will be **sw**imming there too.
`s.+w`	Oh you **sleepy young heads dream of w**onderful things, Beautiful mermaid**s will sw**im through the sea,
`Off|will`	So turn **off** the light, Beautiful mermaids **will** swim through the sea, And you **will** be swimming there too.
`im*ing`	And you will be sw**imming** there too.
`im?ing`	<No Matches>

Programmable Text Processing

- sed, awk
  - Scans one or more files
  - Performs an **action on all lines** that match a particular **condition**
  - Actions and conditions may be **stored in a file**
    - **E.g. sed -f sedfile test.txt**
  - Or may be specified at command line in **single quotes**
    - **E.g. sed '1,100 s/A/a/' test.txt**
  - Does not modify the input file
  - Writes modified file to standard file

Stream Editor (sed)

- Command
  - begins with an address or an **addressRange** or a **Regular expression**
    - E.g. 1,4 10,$ /Expr/
- Action : things you can do with *sed*
  - **d** delete lines d
  - **P** print line
  - **s/oldExpr/newStr/f** substitution
    - f=g, replace all occurrences
    - f=p, print
  - **i** insert following text before next output until one not ending in \
  - **c** change lines to following text until one not ending in \

sed ‘1,100 s/A/a/’ test.txt
Command: **1,100**
Action: **s/A/a**

Substituting Text

- `$ sed 's/^/  /' file > file.new`
  - indents each line in the file by 2 spaces
- `$ sed 's/^ *//' file > file.new`
  - removes all leading spaces from each line of the file
- `$ sed '/a/d' file > file.new`
  - deletes all lines containing 'a'
- `$ sed '200,300 s/A/a/' f1 f2 f3 >new`
  - combine file f1, f2 and f3 together
  - replace 'A' with 'a' from line 200 to 300 in the new combined content
  - store the output of sed command to file new
- `$ cat f1 f2 f3 | sed '200,300 s/A/a/' > new`

Inserting Text

- Add two lines at the beginning of file

$ cat dummy

```
one
two
three
four
five
six
```

$ cat sed1

```
1i\
Copyright 2016 by Yuan\
All rights reserved
```

$ sed –f sed1 dummy

```
Copyright 2016 by Yuan
All rights reserved
one
two
three
four
five
six
```

Replacing Text

- Replace lines 1-3 by "Lines 1-3 are censored"

$ cat dummy

```
one
two
three
four
five
six
```

$ cat sed2

```
1,3c\
Lines 1-3 are censored
```

$ sed -f sed2 dummy

```
Lines 1-3 are censored
four
five
six
```

Deleting Text

- Delete only those lines that contain ‘o’

$ cat dummy

```
one
two
three
four
five
six
```

$ cat sed3

```
/.*o/d
```

$ sed -f sed2 dummy

```
three
five
six
```

$ sed '/.*o/d' dummy

```
three
five
six
```

awk Command

- awk [condition] [\{action\}]
- Condition
  - special tokens **BEGIN** or **END**
  - an expression involving logical operators, relational operators, and/or regular expressions
- Action: one of the following kinds of C-like statements
  - if-else; while; for; break; continue
  - assignment statement: var=expression
  - print; printf;
  - next (skip remaining patterns on current line)
  - exit (skips the rest of the current line)
  - list of statements

awk Command

- *awk* reads a line
  - breaks it into fields separated by tabs/spaces
  - or other separators specified by –F option
- Accessing individual fields
  - $1,…,$n refer to fields 1 through n
  - $0 refers to entire line
- Example: Print the number of fields and first field in the /etc/passwd file.

$ awk -F: '{ print NF, $1 }' /etc/passwd

-F:	Use colon ':' as the field separator
NF	Built-in variable, means number of fields
$1	Refers to filed 1

awk Command

- Special tokens in awk

BEGIN	Triggered before first line read
END	Triggered after last line read
FILENAME	Name of file being processed
NR	Current line #
NF	Number of fields

awk Example

$cat /etc/passwd

```
nobody:*:-2:-2:Unprivileged User:/:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
...
lp:*:26:26:Printing Services:/var/spool/cups:/usr/bin/false
```

$cat p1.awk

```
#Before processing first line, print out "Start of file"
BEGIN { print "Start of file: "}
#Print out the first, sixth, and seventh fileds in the remained lines
{ print $1 "   " $6 "   " $7 }
#After processing all lines, Print out "End of file" with Filename
END { print "End of file", FILENAME }
```

$awk -F: -f p1.awk /etc/passwd

```
Start of file:
nobody  /  /usr/bin/false
root  /var/root  /bin/sh
...
lp  /var/spool/cups  /usr/bin/false
End of file /etc/passwd
```

awk Example

$cat p2.awk

```
NR > 1 && NR < 4 { print NR, $1, $6, NF }
#For the second and third lines, print out line number, the first field, the sixth fields and number of fields
```

$ awk -F: -f p2.awk /etc/passwd

```
2 root /var/root /bin/sh 7
3 daemon /var/root /usr/bin/false 7
```

awk Variables

$cat p3.awk

```
BEGIN {print "Scanning file"}
{  printf "line %d: %s\n", NR, $0;
   lineCount++;
   wordCount += NF;
}
END {printf "lines =  %d, words = %d\n", lineCount, wordCount }
```

$awk -f p3.awk /etc/passwd

```
Scanning file
line 1: nobody:*:-2:-2:Unprivileged User:/:/usr/bin/false
line 2: root:*:0:0:System Administrator:/var/root:/bin/sh
...
line 37: lp:*:26:26:Printing
Services:/var/spool/cups:/usr/bin/false
lines = 37, words = 141
```

awk Condition Ranges

- Ranges by line numbers
- Ranges by patterns
  - From the first line that matches first expression
  - Until that matches second condition

```
$ cat /etc/passwd
nobody:*:-2:-2:Unprivileged User:/:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
...
lp:*:26:26:Printing Services:/var/spool/cups:/usr/bin/false


$ awk -F: '/nobody/,/root/ {print $0}' /etc/passwd
nobody:*:-2:-2:Unprivileged User:/:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
```

Sorting Files (sort)

- Sorts a file in ascending or descending order based on one or more fields.
- Individual fields are ordered lexicographically, which means that corresponding characters are compared based on their ASCII value.

Sorting Files (sort)

- sort -tc -r [+POS1 [-POS2]] {sortField -bfMn}* {fileName}*
  - **-tc** separator is c instead of blank e.g. -t:
  - **-r** descending instead of ascending
  - **+POS1 [-POS2]** key positions start [up to end]
  - -b ignore leading blanks
  - -f ignore case
  - **-M** month sort (3 letter month abbreviation)
  - **-n** numeric sort

Sort Examples

$ cat sort.dat

```
John Smith 1222 20 Apr 1956
Tony Jones 1012 20 Mar 1950
John Duncan 1111 20 Jan 1966
Larry Jones 1223 20 Dec 1946
```

$ sort +4 -5 -M sort.dat

```
John Duncan 1111 20 Jan 1966
Tony Jones 1012 20 Mar 1950
John Smith 1222 20 Apr 1956
Larry Jones 1223 20 Dec 1946
```

$ sort +0 -2 sort.dat

```
John Duncan 1111 20 Jan 1966
John Smith 1222 20 Apr 1956
Larry Jones 1223 20 Dec 1946
Tony Jones 1012 20 Mar 1950
```

$ sort +4 -5 sort.dat

```
John Smith 1222 20 Apr 1956
Larry Jones 1223 20 Dec 1946
John Duncan 1111 20 Jan 1966
Tony Jones 1012 20 Mar 1950
```

Note: the position of field for *sort* starts from 0

Archiving (tar)

- Create a "tap archive" format file from the file list
  - tar –**c**vf tarFileName fileList
- Extract files from a "tap archive" format file to current directory
  - tar -**x**vf tarFileName
- Show the content of a "tap archive" format file
  - tar –**t**vf tarFileName

-f enables you to give a tar file name
Default name is /dev/rmt0
-v verbose

Create a tar file


```
$ tar -cvf ch6.tar ch6
  ch6/
  ch6/menu.csh
  ch6/junk/
  ch6/junk/junk.csh
  ch6/junk.csh
  ch6/menu2.csh
  ch6/multi.csh
  ch6/expr1.csh
  ch6/expr3.csh
  ch6/expr4.csh
  ch6/if.csh
  ch6/menu3.csh

$ ls -l ch6.tar
  -rw-rw-r--   1 raj     raj         20480 Jun 26 20:08 ch6.tar
```

Show Contents in a Tar File


```
$ tar -tvf ch6.tar
 drwxr-xr-x raj/raj       0 2007-06-03 09:57 ch6/
 -rwxr-xr-x raj/raj     403 2007-06-02 14:50 ch6/menu.csh
 drwxr-xr-x raj/raj       0 2007-06-03 09:57 ch6/junk/
 -rwxr-xr-x raj/raj    1475 2007-06-03 09:57 ch6/junk/junk.csh
 -rwxr-xr-x raj/raj    1475 2007-06-03 09:56 ch6/junk.csh
 -rw-r--r-- raj/raj     744 2007-06-02 15:59 ch6/menu2.csh
 -rwxr-xr-x raj/raj     445 2007-06-02 15:26 ch6/multi.csh
 -rwxr-xr-x raj/raj     279 2007-06-02 15:18 ch6/expr1.csh
 -rwxr-xr-x raj/raj      98 2007-06-02 15:20 ch6/expr3.csh
 -rwxr-xr-x raj/raj     262 2007-06-02 15:21 ch6/expr4.csh
 -rwxr-xr-x raj/raj     204 2007-06-02 15:22 ch6/if.csh
 -rw-r--r-- raj/raj     744 2007-06-02 16:01 ch6/menu3.csh
 -rw-rw-r-- raj/raj      29 2007-06-21 11:06 date.txt
```

Extract a Tar File


```
$ rm -fr ch6

$ tar -xvf ch6.tar
  ch6/
  ch6/menu.csh
  ch6/junk/
  ch6/junk/junk.csh
  ch6/junk.csh
  ch6/menu2.csh
  ch6/multi.csh
  ch6/expr1.csh
  ch6/expr3.csh
  ch6/expr4.csh
  ch6/if.csh
  ch6/menu3.csh
  date.txt
```

Searching files (find)

- *find* <staringDirectory> <matching criteria and actions>
  - Searching the files matching given expression starting from pathName
- Expression
  - -name pattern
    - true if the file name matches pattern
  - -perm oct
    - true if the octal description of file's permission equals oct
  - -type ch
    - true if the type of the file is ch (b=block, c=char ..)
  - -user userId
    - true if the owner of the file is userId
  - -group groupId
    - true if the group of the file is groupId

Find Examples

- $ find / -name *.java
  - searches for all Java file in the entire file system
- $ find . -name 'sed*'
  - Searches for all files with names starting "sed"

Substituing User

- % su userName
  - If userName is not specified, root is assumed
  - Need access privileges for this
  - Requires password
- % sudo command
  - User can execute command as superuser
  - Requires password

Review

- Pattern matching (grep)
- Pattern matching and processing (awk,sed)
- Sort
- Archiving(tar)
- Searching files(find)