

ADVANCED UNIX SHELL COMMANDS

OS LAB 2

**Refer lab manual in which each command is
explained in detail**

Note

- Only few commands related to Lab 2 are discussed here
- Students can learn these Advanced Unix Shell Commands and practice
- This will help you in understanding few commands now, all Advanced Shell commands can be focused during regular labs.

OS Lab 2 Commands

1. Commands used to extract, sort, filter and process data

- a. grep
- b. sort
- c. wc (word count)
- d. cut
- e. sed (stream editor)
- f. tr (translate)

OS Lab 2 Commands (Contd...)

2. Process management commands

- a. ps
- b. kill

3. File permission commands: chmod (Change mode)

4. Other useful commands :

- a. echo
- b. bc (Basic Calculator)
- c. The vi editor

1a. grep

➤ **g/re/p (globally search a regular expression and print)**, which has the same effect: doing a global search with the regular expression and printing all matching lines.

➤ *grep* <someText> <fileName>

#search, case sensitive, for <someText> in <file-name>, use -i for case insensitive search

1a. grep

```
$ cat sample.txt
```

1. A Unix shell is a command-line interpreter or shell that provides a command line user interface for Unix-like operating systems.
2. The shell is both an interactive command language and a scripting language, and is used by the operating system to control the execution of the system using shell scripts.
3. Users typically interact with a Unix shell using a terminal emulator; however, direct operation via serial hardware connections or Secure Shell are common for server systems.
4. All Unix shells provide filename wildcarding, piping, here documents, command substitution, variables and control structures for condition-testing and iteration.~\$

1a. grep

\$ grep shell sample.txt **#case insensitive search**

1. A Unix **shell** is a command-line interpreter or **shell** that provides a command line user interface for Unix-like operating systems.
2. The **shell** is both an interactive command language and a scripting language, and is used by the operating system to control the execution of the system using **shell** scripts.
3. Users typically interact with a Unix **shell** using a terminal emulator; however, direct operation via serial hardware connections or Secure Shell are common for server systems.
4. All Unix **shells** provide filename wildcarding, piping, here documents, command substitution, variables and control structures for condition-testing and iteration.

1a. grep

\$ grep an sample.txt

1. A Unix shell is a comm~~and~~-line interpreter or shell that provides a comm~~and~~ line user interface for Unix-like operating systems.
2. The shell is both ~~an~~ interactive comm~~and~~ ~~lan~~guage ~~and~~ a scripting ~~lan~~guage, ~~and~~ is used by the operating system to control the execution of the system using shell scripts.
4. All Unix shells provide filename wildcarding, piping, here documents, comm~~and~~ substitution, variables ~~and~~ control structures for condition-testing ~~and~~ iteration.

1a. grep

Expression	Meaning
.	(Dot) match any single character
[<i>charset</i>]	Match any member of the set <i>charset</i>
<i>limited_expression</i> *	Match any number of repetitions of <i>limited_expression</i> including zero.
<i>limited_expression</i> \{ <i>M</i> \}	Match exactly <i>M</i> repetitions of <i>limited_expression</i>
<i>limited_expression</i> \{, <i>N</i> \}	Match zero to <i>N</i> repetitions of <i>limited_expression</i>
<i>limited_expression</i> \{ <i>M,N</i> \}	Match <i>M</i> to <i>N</i> repetitions of <i>limited_expression</i>
<i>expr0expr1</i>	(Concatenation) match <i>expr0</i> then <i>expr1</i>
^ <i>expression</i>	Match <i>expression</i> only at beginning of line
<i>expression</i> \$	Match <i>expression</i> only at end of line

1a. grep

- 'c...s' Matches lines containing an c, followed by three characters, followed by an s
- ' *class' Matches lines with zero or more spaces, of the preceding characters followed by the pattern class [here, preceding character is space]
- 'o\{5\},, Matches if line has 5 o's
- 'o\{5,\},, at least 5 o's
- 'o\{5,10\},, between 5 and 10 o's
- \$ grep '5\..' datafile

Prints a line containing the number 5, followed by a literal period and any single character

1a. grep

Examples:

\$cat fruitlist.txt

apple
apples
pineapple
fruit-apple
banana
pear
peach
orange

\$grep apple fruitlist.txt

apple
apples
pineapple
fruit-apple

\$grep e\$ fruitlist.txt

apple
pineapple
fruit-apple
orange

\$grep ^p fruitlist.txt

pineapple
pear
peach

Examples:

\$cat fruitlist.txt

apple
apples
pineapple
fruit-apple
banana
pear
peach
orange

\$grep -x apple fruitlist.txt # match whole line
apple

\$grep -v apple fruitlist.txt #print unmatched lines
banana
pear
peach
orange

1b sort

sort is a program that prints the lines of its input or **concatenation of all files listed in its argument** list in sorted order.

\$cat fruitlist1.txt

apple
apples
pineapple
fruit-apple
banana
pear
peach
orange

\$cat fruitlist2.txt

figs
watermelon
dates
guava
breadfruit
apricots

\$cat fruitlist1.txt fruitlist2.txt

apple
apples
pineapple
fruit-apple
banana
pear
peach
orange
figs
watermelon
dates
guava
breadfruit
apricots

1b sort

\$sort fruitlist1.txt

apple
apples
banana
fruit-apple
orange
peach
pear
pineapple

\$sort fruitlist1.txt fruitlist2.txt

apple
apples
apricots
banana
breadfruit
dates
figs
fruit-apple
guava
orange
peach
pear
pineapple
watermelon

\$cat fruitlist1.txt fruitlist2.txt | sort

apple
apples
apricots
banana
breadfruit
dates
figs
fruit-apple
guava
orange
peach
pear
pineapple
watermelon

1b sort

- Note: Sort doesn't modify the input file content.
- *sort* <any number of filenames> #sort the content of file(s)
- *sort* <fileName> #sort alphabetically
- *sort -o <outputFile> <file>* #write result to a file

1b sort

sort -r <fileName> #sort in reverse order

```
$sort -r fruitlist1.txt
```

pineapple

pear

peach

orange

fruit-apple

banana

apples

apple

1b sort

sort -n <fileName> #sort numbers

\$cat numbers

352
3242
63
67
687
242
47656
857
367
245
24
235

\$sort numbers

#non-numeric sorting

235
24
242
245
3242
352
367
47656
63
67
687
857

\$sort -n numbers

#numeric sorting

24
63
67
235
242
245
352
367
687
857
3242
47656

1c wc

wc (word count)

➤ This shell command is used to print the number of lines words in the input file/s.

\$wc <fileName> #Number of lines, number of words, byte size of <fileName>.

```
$cat fruitlist.txt
```

```
apple  
apples  
pineapple  
fruit-apple  
banana  
pear  
peach  
orange
```

```
$wc fruitlist.txt
```

```
8 8 60 f1
```

1c wc

Other arguments includes: -l (lines), -w (words), -c (byte size), -m

```
$wc -l fruitlist.txt
```

```
8 f1
```

```
$wc -w fruitlist.txt
```

```
8 f1
```

```
$wc -c fruitlist.txt
```

```
60 f1
```

1c wc

\$ wc * : counts for all files in the current directory.

```
osboxes@osboxes:~$ wc *
  3      20      111 a
 14      14      108 f
  8       8       60 f1
  6       6       48 f12
 13      12       49 numbers
wc: sample: Is a directory
  0       0       0 sample
  5      38      214 sort
 49      98      590 total
```

1d. cut

- cut is a data filter: it extracts columns from tabular data.
- Columns can be character positions or—relevant in this example—fields that are separated by TAB characters (default delimiter) or other delimiters.

\$cat record.txt #tab as field delimiter

Name	Roll number	Regno	Semester	Section	Department
ABC	1	111	4	A	IT
DEF	2	222	4	A	IT
GHI	3	333	4	A	CCE
JKL	4	444	4	B	CCE
MNO	5	555	4	B	CCE
PQR	6	666	4	A	CSE
STU	7	777	4	B	CSE
VWX	8	888	4	A	ECE

\$cut -c 1-3 record.txt # -c specifies characters to be extracted from each record

Name
ABC
DEF
GHI
JKL
MNO
PQR
STU
VWX

cut -c1-3,8 record.txt
#characters 1 thru 3, and 8

ABC4

DEF4

GHI4

JKL4

MNO4

PQR4

STU4

VWX4

cut -f 1,4,7 record.txt

#tab-separated fields 1, 4, and 6

#-f specifies fields to be extracted.

Name	Semester	Department
ABC	4	IT
DEF	4	IT
GHI	4	CCE
JKL	4	CCE
MNO	4	CCE
PQR	4	CSE
STU	4	CSE
VWX	4	ECE

f1: Name, f2: Roll number,, f3: Regno,
f4: Semester, f5: Section, f6: Department

\$cat record_comma.txt #comma (,) as field delimiter

Name, Roll number, Regno, Semester, Section, Department

ABC, 1, 111, 4, A, IT

DEF, 2, 222, 4, A, IT

GHI, 3, 333, 4, A, CCE

JKL, 4, 444, 4, B, CCE

MNO, 5, 555, 4, B, CCE

PQR, 6, 666, 4, A, CSE

STU, 7, 777, 4, B, CSE

VWX, 8, 888, 4, A, ECE

cut -d “,” -f 1,4,7 record_comma.txt

#comma separated fields 1, 4, and 6 #-f specifies fields to be extracted.

#option -d specifies field delimiter (Note: Default delimiter is TAB)

Name	Semester	Department
ABC	4	IT
DEF	4	IT
GHI	4	CCE
JKL	4	CCE
MNO	4	CCE
PQR	4	CSE
STU	4	CSE
VWX	4	ECE

f1: Name, f2: Roll number,, f3: Regno,
f4: Semester, f5: Section, f6: Department

f. tr (translate)

The tr filter is used to translate one set of characters from the standard inputs to an-other.

Examples:

\$tr “[a-z]” “[A-Z]” < filename #maps all lowercase characters in filename to up-percase.
Content of the file is not changed.

\$cat fruitlist.txt

apple
apples
pineapple
fruit-apple
banana
pear
peach
orange

\$tr a-z A-Z < fruitlist.txt

APPLE
APPLES
PINEAPPLE
FRUIT-APPLE
BANANA
PEAR
PEACH
ORANGE

f. tr (translate)

tr 'abcd' 'jkmn' #maps all characters a to j, b to k, c to m, and d to n.

The character set may be abbreviated by using character ranges.

The previous example could be written: tr 'a-d' 'jkmn'

\$cat fruitlist.txt

apple
apples
pineapple
fruit-apple
banana
pear
peach
orange

\$tr 'abcd' 'jkmn' < fruitlist.txt

jpple
jpples
pinejpple
fruit-jpple
kijnjn
pejr
pejmh
orjnge

f. tr (translate)

The s flag (suppress) causes tr to compress sequences of identical adjacent characters in its output to a single token.

tr -s '\n' #replaces sequences of one or more newline characters with a single newline.

\$cat fruitlist.txt

apple
apples
pineapple
fruit-apple
banana
pear
peach
orange

\$tr -s 'p' < fruitlist.txt

aple
aples
pineapple
fruit-ape
banana
pear
peach
orange

f. tr (translate)

The d flag causes tr to delete all tokens of the specified set of characters from its input. The tr -d '\r' statement removes carriage return characters.

```
$cat fruitlist.txt
```

```
apple  
apples  
pineapple  
fruit-apple  
banana  
pear  
peach  
orange
```

```
$tr -d 'a' < fruitlist.txt
```

```
pple  
pples  
pinepple  
fruit-pple  
bnn  
per  
pech  
ornge
```

f. tr (translate)

The `c` flag indicates the complement of the first set of characters. The invocation `tr -cd '[:alnum:]'` therefore removes all non-alphanumeric characters.

```
$cat fruitlist.txt
```

```
apple  
apples  
pineapple  
fruit-apple  
banana  
pear  
peach  
orange
```

```
$tr -c a j < fruitlist.txt
```

```
#except a everything else is replaced by letter j
```

```
#even new line character is replaced by letter j
```

```
ajjjjjajjjjjjjjjjjajjjjjjjjjjjajjjjjajajajjjjjajjjjjjjjjjj
```


2. Process management commands

a. ps

- The ps command (short for "process status") displays the currently-running processes.
- ps command displays process id (PID), TTY (Terminal associated with the process), time The amount of CPU time used by the process and command name (CMD).

2. Process management commands

b. kill

- kill is a command that is used in several popular operating systems to send signals to running processes in order to request the termination of the process.
- The signals in which users are generally most interested are **SIGTERM** and **SIGKILL**.
- The **SIGTERM** signal is sent to a process to request its termination.
- Unlike the **SIGKILL** signal, it can be caught and interpreted or ignored by the process.
- This allows the process to perform nice termination releasing resources and saving state if appropriate.
- The **SIGKILL** signal is sent to a process to cause it to terminate immediately (kill). This signal cannot be caught or ignored, and the receiving process cannot perform any clean-up upon receiving this signal.

3. File permission commands

a. chmod (Change mode)

- The chmod numerical format accepts up to four octal digits. The three rightmost digits refer to permissions for the file owner, the group, and other users.

Numerical permissions #	Permission	rwX
7	read, write and execute	rwX
6	read and write	rw-
5	read and execute	r-X
4	read only	r--
3	write and execute	-wX
2	write only	-w-
1	execute only	--X
0	none	---