UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA

# Course Name : Operating System Laboratory

UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA

## Week 4 Assignment

Lab Assignments:

a) Find out the PID of your login shell.

a) Remove the header line from the ps output.

## ps command in Linux:

A process is an executing instance of a program and carry out different tasks within the operating system.

Linux provides us a utility called ps for viewing information related with the processes on a system which stands as abbreviation for **“Process Status”.**

ps command is used to list the currently running processes and their PIDs along with some other information depends on different options.

It reads the process information from the virtual files in /proc file-system. /proc contains virtual files, this is the reason it’s referred as a virtual file system.

ps provides numerous options for manipulating the output according to our need

## kill command in Linux with Examples

*kill* command in Linux (located in /bin/kill), is a built-in command which is used to terminate processes manually. *kill* command sends a signal to a process which terminates the process. If the user doesn't specify any signal which is to be sent along with kill command then default *TERM* signal is sent that terminates the process.

$kill -l

**kill -l :**To display all the available signals you can use below command option:

*Signals can be specified in three ways:*

**By number (e.g. -5)**

**With SIG prefix (e.g. -SIGkill) Without SIG prefix (e.g. -kill)**

**Note:**
Negative PID values are used to indicate the process group ID. If you pass a process group ID then all the process within that group will receive the signal.

A PID of -1 is very special as it indicates all the processes except kill and init, which is the parent process of all processes on the system.

To display a list of running processes use the command *ps* and this will show you running processes with their PID number. To specify which process should receive the kill signal we need to provide the PID.

**kill -s :** To show how to send signal to processes.
**Syntax:**
kill {-signal | -s signal} pid

## grep command in Unix/Linux

The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and print out).

**Syntax:**

**grep [options] pattern [files]**

**$cat > geekfile.txt**

unix is great os. unix is opensource. unix is free os. learn operating system. Unix linux which one you choose. uNix is easy to learn.unix is a multiuser os.Learn unix
.unix is a powerful.

**1. Case insensitive search :** The -i option enables to search for a string case insensitively in the given file. It matches the words like “UNIX”, “Unix”, “unix”.

**$grep -i "UNix" geekfile.txtOutput:**

## grep command in Unix/Linux

**Options Description**

**-c** : This prints only a count of the lines that match a pattern
**-h :** Display the matched lines, but do not display the filenames.
**-i :** Ignores, case for matching
**-l :** Displays list of a filenames only.
**-n :** Display the matched lines and their line numbers.
**-v :** This prints out all the lines that do not matches the pattern
**-e exp :** Specifies expression with this option. Can use multiple times.
**-f file :** Takes patterns from file, one per line.
**-E :** Treats pattern as an extended regular expression (ERE)
**-w :** Match whole word
**-o :** Print only the matched parts of a matching line, with each such part on a separate output line.
**-A n :** Prints searched line and nlines after the result.
**-B n :** Prints searched line and n line before the result.
**-C n :** Prints searched line and n lines after before the result.

**Displaying the count of number of matches :** We can find the number of lines that matches the given string/pattern

**$grep -c "unix" geekfile.txtOutput:**

**Display the file names that matches the pattern :** We can just display the files that contains the given string/pattern.

**$grep -l "unix"* or $grep -l "unix" f1.txt f2.txt f3.xt f4.txtOutput:**

geekfile.txt.

**4. Checking for the whole words in a file :** By default, grep matches the given string/pattern even if it is found as a substring in a file. The -w option to grep makes it match only the whole words.

**$ grep -w "unix" geekfile.txtOutput:**

**Displaying the count of number of matches :** We can find the number of lines that matches the given string/pattern

**$grep -c "unix" geekfile.txtOutput:**

**Display the file names that matches the pattern :** We can just display the files that contains the given string/pattern.

**$grep -l "unix"* or $grep -l "unix" f1.txt f2.txt f3.xt f4.txtOutput:**

geekfile.txt.

**4. Checking for the whole words in a file :** By default, grep matches the given string/pattern even if it is found as a substring in a file. The -w option to grep makes it match only the whole words.

**$ grep -w "unix" geekfile.txtOutput:**

**egrep** is a pattern searching command which belongs to the family of grep functions. It works the same way as ***grep -E*** does.

It treats the pattern as an extended regular expression and prints out the lines that match the pattern.

If there are several files with the matching pattern, it also displays the file names for each line.

**Note:** The *egrep* command used mainly due to the fact that it is faster than the grep command. The egrep command treats the meta-characters as they are and do not require to be escaped as is the case with grep. This allows reducing the overhead of replacing these characters while pattern matching making egrep faster than *grep* or *fgrep*.

**Options:** Most of the options for this command are same as *grep*.

**-c:** Used to counts and prints the number of lines that matched the pattern and not the lines.

## fgrep command in Linux with examples

The *fgrep* filter is used to search for the fixed-character strings in a file. There can be multiple files also to be searched. This command is useful when you need to search for strings which contain lots of regular expression metacharacters, such as “^”, “$”, etc.

**Syntax:**

fgrep [options] [ -e pattern_list] [pattern] [file]

**Options with Description:**

**-c :** It is used to print only a count of the lines which contain the pattern.
**-h :** Used to display the matched lines.
**-i :** During comparisons, it will ignore upper/lower case distinction.
**-l :** Used to print the names of files with matching lines once, separated by new-lines. It will not repeat the names of files when the pattern is found more than once.
**-n :** It is used precede each line by its line number in the file (first line is 1).
**-s :** It will only display the error messages.
**-v :** Print all lines except those contain the pattern.

**-x :** Print only lines matched entirely.
**-e pattern_list :** Search for a string in pattern-list (useful when the string begins with a “-“).
**-f pattern-file :** Take the list of patterns from pattern-file.
**pattern :** Specify a pattern to be used during the search for input.

**1) Anchor Characters:** '^' and '$' at the beginning and end of the pattern are used to anchor the pattern to the start of the line, and to the end of the line respectively.
**Example:** "^Name" matches all lines that start with the string "Name". The strings "\<" and "\>" are used to anchor the pattern to the start and end of a word respectively.

**2) Wildcard Character:** '.' Is used to match any character.
**Example:**"^.$" will match all lines with any single character.

**3) Escaped Characters:** Any of the special characters can be matched as a regular character by escaping them with a '\'.
**Example:** "\$*" will match the lines that contain the string "$*"

**4) Character Range:** A set of characters enclosed in a '[' and ']' pair specify a range of characters to be matched.

**5) Repetition Modifier:** A '*' after a character or group of characters is used to allow matching zero or more instances of the preceding pattern.
**The grep command supports a number of options for additional controls on the matching:**
-i: performs a case-insensitive search.
-n: displays the lines containing the pattern along with the line numbers.
-v: displays the lines not containing the specified pattern.
-c: displays the count of the matching patterns.

#Match all lines that start with ‘hello’. **E.g:** “hello there”

$ grep “^hello” file1

Match all lines that end with ‘done’. **E.g:** “well done”

$ grep “done$” file1

Match all lines that contain any of the letters ‘a’, ‘b’, ‘c’, ‘d’ or ‘e’.

$ grep “[a-e]” file1

Match all lines that do not contain a vowel

$ grep “[^aeiou]” file1

Match all lines that start with a digit following zero or more spaces. **E.g:** “ 1.” or “2.”
$ grep “ *[0-9]” file1
Match all lines that contain the word hello in upper-case or lower-case
$ grep -i “hello”

## fgrep command in Linux with examples

Consider below file as input. Here it is create using cat command and ***"name of the file is  para"***.

**-c option:** Displaying the count of number of matches. We can find the number of lines that  match the given string.

**Example:**
$fgrep -c "usin.g" para**Output:**
1
**-h option:** To display the matched lines.
**Example:**
fgrep -h "usin.g" para**Output:**

.

**-i option:** Used in case insensitive search. It ignore upper/lower case distinction during comparisons.

c) List all processes that you are currently running on your machine, sorted by the command name in alphabetical order (i.e. a process running acroread should be listed before a process running zwgc). The output should consist only of the processes you are running and nothing else (i.e. if you are running 6 processes, the output should only have 6 lines).

c) Display the files in the current directory that contain the string CSEUEMK in either upper- or lowercase.

**d)**
```
$  grep  -il       'CSEUEMK'       *
```

c) Store in a variable the number of lines containing the word CSE in the files cse1, cse2 and cse3.

**e)**
```
$ var=`grep CSE cse[1-3] | wc  -l`
         $ echo      $var
```

.

f) If you did not have the wc command, how would you use grep to count the number of users currently using the system?

**f) who | grep -c “.*”**

f) Remove blank lines from a file using grep.

```
$ grep  -v            “^$”      aa1
```

h) List the ordinary files in your current directory that are not writable by the owner.

```
$  ls       -l  | grep  -v         ‘^..w’
```

i) Locate lines ending and beginning with a dot and containing anything between them.

```
  $ grep  ‘^\..*\.$’   mca4
```

j) Locate lines that are less than 100 characters in length

```
$  grep   ‘^.\{0,99\}$’         * file1
```

# Thank You