WEEK –8

Filesystem links/Hard and symbolic links/Process management/Aliases /Command History

| **Command** | **Useful Options** | **Purpose** |
| --- | --- | --- |
| ln | -s, -f | Creates links between files (hard or symbolic) |
| ps | -e, -f, -u | Displays a list of currently running processes |
| top | -d, -u, -p | Provides a dynamic real-time view of system processes and their resource usage |
| fg | %job\_id, %process\_id | Brings a suspended or backgrounded process back to the foreground |
| bg | %job\_id, %process\_id | Puts a suspended or stopped process in the background, allowing it to continue running |
| jobs | -l, -n | Lists all current jobs, including their job ID, status, and command |
| kill | -9, -15, -SIGKILL, -SIGTERM | Sends a signal to terminate a process or job |
| alias | -p | Creates a shortcut (alias) for a command or a sequence of commands |
| unalias | -a | Removes an alias previously defined with the **alias** command |
| history | -c, -d, -a | Displays a list of previously executed commands, including their line numbers, and allows for various manipulations of that list |

The & symbol at the end of the command is used to run a process in the background.

|  |  |
| --- | --- |
| **ctrl-c** | **Terminates** a process running in the foreground |
| **ctrl-z** | Sends a process running in the foreground into the **background**. |

top To display running processes in "real-time"

WEEK –9

**Complex Regular Expressions**

*Complex Regular Expressions* use symbols to help match text for more precise (complex) patterns.  
The most common complex regular expression symbols are displayed below:

**Anchors: ^** , **$**  
Match lines the begin (^) or end ($) with a pattern.

**Single Character:**   **.**  
Represents a single character that can be any type of character.

**Character Class:**  **[ ]** , **[^ ]**  
Represents a single character but with restrictions.

**Zero or More Occurrence:**  **\***  
Zero or more occurrences of previous character.

**Extended Regular Expressions**

*Extended Regular Expressions* consist of additional special characters to “extend”  
the capability of regular expressions( **egrep** or **grep -E)**

**Repetition:** **{min,max}**  
Allows for more precise repetitions. Using braces, you can specify  
the **minimum** and/or **maximum** number of repetitions.

**Groups:** **( )**  
Allows you to search for repetition for a **group of characters**, a **word**, or a **phase**.  
You enclose them within brackets **( )** to specify a **group**.

**or Condition:**  **|**  
Can be used with **groups** to match a variety of character(s), words or phases.  
The | symbol is used to separate the variety of character(s) within a *group*.

"-w": match only whole words

"-i": ignore case sensitivity

.\* means match any character zero or more times

+ shortcut for 1 or more {1, } (only for egrep)

? shortcut for zero or one {0,1} (only for egrep)

WEEK -10

SED/AWK

Sed

Graphical user interface, text, application, email

Description automatically generated

sed -n '3,6 p' ~murray.saul/uli101/stuff.txt

sed '4 q' ~murray.saul/uli101/stuff.txt

sed '/the/ d' ~murray.saul/uli101/stuff.txt

sed 's/line/NUMBER/g' ~murray.saul/uli101/stuff.txt

**Syntax: sed [-n] 'address instruction' filename**

**–n** option is used to suppress default display (only 1 line)

\sed -n '3,6 p' ~murray.saul/uli101/stuff.txt

// print line 3 to 6 only once from stuff.txt file

**sed -n '3,6 p' means 3 to 6**

**Awk**

**awk [-F] 'selection-criteria {action}’ file-name**

**–F** option can be used to specify the default **field delimiter** (separator) character

**awk '$5 ~ /^[0-9]/ {print $1,$4}' ~/cars**

**// if field 5 begins with any number, then print field 1 and field four from the file cars.**

$0 (printing the entire input line)

NR is a built-in variable (number of line). >> NR==1 >> line 1

WEEK –11

Bash Shell Scripting

**she-bang line >> #! /bin/bash**

**Create a file using nano or vi (bash.sh)**

**Add #! /bin/bash at the beginning**

**Add Permission chmod +x bash.sh**

**Run ./bash.sh**

**Environment Variables**

**$HOME: The user's home directory.**

**$PATH: A colon-separated list of directories in which the shell looks for commands.**

**$USER: The username of the current user.**

$HOSTNAME: The name of the current host.

the full pathname of the shell. (which bash)

Difference between ‘’ "

Text

Description automatically generated

Asking input from user

A picture containing text

Description automatically generated

#>> to write comment

$# >>the number of arguments passed to a script or function.

$\* >>display all the arguments as a string

$? >>exit code of last command

Text, letter

Description automatically generated

$0 >>the name of shell

$1>> Argument…

-f >>if file exists or not

-d >>if directory exists or not

Math Operation

*num1=5;num2=10*

*echo “$(( $num1 + $num2))”*

*CLI Result: 15*

*on Purpose*

*-eq Equal to*

*-ne Not equal to*

*-lt , -le Less than, Less than or equal to*

*-gt, -ge Greater than, greater than or equal to*

WEEK- 12

Control Flow Statement If/Else

Text

Description automatically generated

Contorl Flow (LOOP)

Text

Description automatically generated

Contorl Flow (If /elif)

Text

Description automatically generated

Control Flow (While)

Text

Description automatically generated

Exit >>to terminate the shell

Break>>to terminate the loop