UNIVERSITI PUTRA MALAYSIA

FAKULTI SAINS KOMPUTER DAN TEKNOLOGI MAKLUMAT

Name:	YUE CHENGHAO HUA JIE WANG KAILUN	Matric:	227154 226758 227046

LAB 3

ASSIGNMENT TITLE : Process Management and System Utilities

ASSIGNMENT TYPE : Individual

OBJECTIVE:
The main objective of this lab is to introduce the students to the Linux operating system's process management and communications using Bash (Bourne Again Shell) commands. Students also will be introduced to some Linux's utilities such as text editor which helps system users to execute batch or complex computing tasks in Linux.

DURATION:
1 week / 3 hours

DUE DATE OF SUBMISSION: WEEK 5
Submit your assignment to your demonstrator. Make sure you sign the submission form after submitting.

PROGRAMME OBJECTIVE (PO):
PO2: Use the available software package with appropriate computer components.

RUBRIC:

CRITERIA	LEVEL 1 Very Poor	LEVEL 2 Poor	LEVEL 3 Good	LEVEL 4 Very Good	LEVEL 5 Excellent
The ability to perform operating system (OS) tasks.	Unable to perform OS tasks or able to perform OS tasks with very limited success (< 40%).	Able to perform OS tasks with limited success (≥ 40%,< 60%).	Able to perform OS tasks with some success (≥ 60%, < 80%).	Able to perform OS tasks with considerable success (≥ 80%, < 100%).	Able to perform OS tasks with outstanding success (100%).

INSTRUCTIONS:

1. Redirecting I/O Streams

Task: Using Output Redirection '>' in Linux
The use of the '>' is to send output to file instead of screen (the standard output). Type the following command, which is to send the output to file named 'myoutput.txt'

```
~$ ls > myoutput.txt
```

Task: Using Output Redirection '>>' in Linux
The use of the '>>' is to send output to file by adding the new content with the existing content. Type the following command, which adding more content to file named 'myoutput.txt'

```
~$ ls >> myoutput.txt
```

Task: Using Input Redirection '<' in Linux
The use of the '<' is to set a specified file as the source of input data for a command. Type the following command, which read the file specified file and display its content.

```
~$ ls < myoutput.txt
```

(Note: The command will be the same as **`ls myoutput.txt`** (without the '<'), since the **ls** command already expecting the first parameter to be the input file. The following example demonstrates a situation where '<' is mandatory

```
~$ ls > outputFile.txt < inputFile.txt
```

Here '<' becomes mandatory since we changed the order of parameters.)

Questions:

1. Describe the examples of tasks that might require the use of the redirection operators (>, >>, <)?

Answer:

> is used when we want to save the output of a command into a file, for example **ls > files.txt** to store the directory listing instead of just printing it on the screen.

>> is used when we want to append new output to an existing file without overwriting the old content, for example appending log messages: **echo "new log" >> app.log.**

< is used when a command should read its input from a file instead of from the keyboard, for example **sort < numbers.txt** to let sort read the numbers from a file.

2. Process Communication Using Pipe

Task: Using Pipe '|' in Linux

The use of pipe '|' is to allow programs to share data (e.g. to allow a program to share its output to become input for another program/command). Type the following command which connect **ls** command with **head** command,

```
~$ ls | head -3
```

The number of pipes can be more than one. Type the following command, which runs the command **ls** and send its output as the input for **head** command, and then send its output to be the input for **tail** command.

```
~$ ls | head -3 | tail -1
```

Questions:

1. Use **man** to learn more about the use of ls, head, tail and then describe the purpose of using the switches (-3 for head, and -1 for tail).

Answer:

For ***head -3,*** the option ***-3*** means that ***head*** will show only the first 3 lines of its input.

For ***tail -1***, the option ***-1*** means that ***tail*** will show only the last 1 line of its input.

2. Give an example of other instruction that can be used with pipes. Show how you use it, and describe the objective/purpose.

Answer:

Example of another command using pipes:

ls | grep ".txt"

In this command, **ls** lists all files in the current directory, and the **pipe |** sends this output to **grep**. The **grep ".txt"** command filters the list and shows only the files whose names contain ".txt". The purpose is to quickly search for specific files in the directory using pattern matching.

3. Process Management

Task: Creating Process
Create a process by running any program or user application.

Task: Viewing Process List
View the currently running process(es) by using typing the following command,

```
~$ ps
```

Task: Terminating Process
We may terminate any running processes by using the **kill** command.

Run a Vim program. Then press Ctrl+Z to make it as background process.

Use the **ps** command to list the currently running processes. Then use the kill command to terminate a process by supplying the process' id. Assuming there is a Vim process with ID (PID) 2390, signal -9 will caused the program to be terminated instantly. Type the following command,

```
~$ kill -9 2390
```

We may also use the **pidof** command to determine a known process. For example, we know a Vim program is running, use the following command to determine the PID of the program.

```
~$ pidof vim
```

Questions:

1. Use **man** to learn more about the use of kill and then describe the three more switches (other than -9) that can be used with **kill** command.

Answer:

-15 (or -TERM): This sends the TERM signal, which politely asks the process to terminate. It allows the program to clean up and save data before exiting.

-2 (or -INT): This sends the INT (interrupt) signal, similar to pressing Ctrl+C in the terminal, and is often used to stop a process interactively.

-STOP: This signal stops (pauses) a process without terminating it, so it can later be resumed with the -CONT signal.

4. Create and Editing Text File Using Text Editor

Task: Open the Vim editor by typing

```
~$ vim
```

Press Ctrl+Z to make it as background process.

Use **vimtutor** to learn more about Vim (commands for editing file, and also how you can save and open an existing file).

```
~$ vimtutor
```

Learn from the vimtutor. Then recall the Vim with fg command and practice creating editing and saving text file.

Questions:

1. Specify three Vim commands that you can use in editing

Answer:

i : enter insert mode at the current cursor position to start typing text.
dd : delete (cut) the entire current line.
yy : yank (copy) the current line into the Vim clipboard so that it can be pasted with p.

2. Specify Vim commands that you can use to save a file

Answer:

:w — write (save) the current file.
:w filename — save the current buffer to a new file with the given name.
:wq or ZZ — save the file and quit Vim.

3. Specify Vim commands that you can use to open a file

Answer:

From the shell: **vim filename** — starts Vim and opens the specified file.

Inside Vim: **:e filename** — edit/open another file by name."

Note: Exploring Shell Scripting

Shell scripting is a powerful facility provided by Bash (and many other shells in other operating systems). With the experience from the exercise, students are encouraged to explore features like using variables, branching and looping command provided in Bash's shell scripting to produce more advanced process management tasks such as batch commands, scheduled processing or automatic software installation/setup.