**CS2106 Introduction to Operating Systems**

**Lab 2 - Shell Scripting and Process Programming**

**Answer Book**

Please read the instructions in the main lab sheet before completing this document. Submission deadline is **Sunday 20 February 2022, 11.59 pm**. The folder will stay open slightly after this, but once the folder closes, **absolutely no submissions will be allowed.**

**Submission checklist:** A ZIP file called AxxxxxxY.zip, where AxxxxxxY is the student ID of the student submitting. The ZIP file should contain:

* This file, appropriately renamed to the submitter’s student ID.
* grade.sh
* lab2p2f.c

|  |  |
| --- | --- |
| **Student 1** | |
| Name: | Wu Xiao Yun |
| Student ID (AxxxxxxY): | A0221772J |
| Group (Bxx): | B06 |
| **Student 2** | |
| Name: | Ang Koon Hwee |
| Student ID (AxxxxxxY): | A0221986W |
| Group (Bxx): | B06 |

**Part 1 – Bash Scripting**

**Question 1.1 (1 mark)**

This line is called shebang, it specifies the shell to be used for the execution of the file. In this case, the terminal uses bash to execute the file.

**Question 1.2 (1 mark)**

Change z=$x-$y to let z=x-y

**Question 1.3 (1 mark)**

Cut and paste code here.

#!/bin/bash

echo “Hello $(whoami), today is $(date +%A), $(date +"%d %B %Y"), and the time is $(date +%T).”

**Question 1.4 (1 mark)**

* $# is the number of positional parameters passed into the script, shell or shell function. In this example, it refers to the arguments supplied to func, which is 3, so $# is 3.
* $1 is the first argument passed into func, in this case $1 is “hello”.
* $2 is the second argument passed into func, in this case $2 is “world”.
* $@ expands to the list of positional parameters starting with $1, in this case $@ is “hello world 13.5”
* $? holds the exit status of the last executed command. 0 means success (true) and non-zero means failure (false). In this case, $? is 0.

**Question 1.5 (1 mark)**

echo $? prints 11. echo $? prints the value of argument i supplied to exit(i).

**Question 1.6 (1 mark)**

./slow 5 ; ./slow 10 executes sequentially. The shell waits for each command to terminate before executing the next command. In this case, the first command (./slow 5) is executed till completion before starting to execute the next command (./slow 10).

5

6

7

8

9

10

Final value of i is 11

10

11

12

13

14

15

Final value of i is 16

./slow 5 & ./slow 10 executes the preceding command (./slow 5) in the background in a subshell. The shell does not wait for the preceding command to terminate before starting to execute the next one, resulting in the interweaving print lines from both programs in this case. It also prints the process ID that is running in the background in the first line.

[1] 69561

5

10

6

11

7

12

8

13

9

14

10

15

Final value of i is 16

Final value of i is 11

[1] + exit 11 ./slow 5

; operator waits for the preceding command to terminate before executing the next one while & operator does not wait.

(For grader only) Part 1 total: \_\_\_\_\_\_\_\_\_\_\_ / 6

**Part 2 – Playing with POSIX Calls**

**Question 2.1 (1 mark)**

**Question 2.2 (1 mark)**

**Question 2.3 (1 mark)**

**Question 2.4 (1 mark)**

Cut and paste new code here and explain

**Question 2.5 (1 mark)**

**Question 2.6 (1 mark)**

**Question 2.7 (1 mark)**

(For grader only)

Part 2 total: \_\_\_\_\_\_\_\_\_\_\_ / 6

**REPORT TOTAL: \_\_\_\_\_\_\_\_\_\_\_\_ / 13**