Exam 1 – Operating Systems in Practice

Name: Aryant Katuwal

ID: C7297933

|  |
| --- |
| Duration:  1. Complete the info above: your name and ID  2. Phones off, please.  3. Leave your ID on the table.  4. Neither Internet nor VLE are not allowed.  5. Answers of the labsheets are not allowed.  6. You are more than welcome to use Linux man.  7. You can use your notes.  8. If you don’t know one item (question), go to the next item (question) solve and return to the ones you left later.  9. Mandatory:  a. Go to File > Info > Protect Document (Left) > Encrypt with Password.  b. Clear the password in the Password box, and then click OK.  10. Mandatory: do file, save as with your name. During the exam, keep saving it.  11. By the end, submit your exam file.  Wish you the best of luck! |

|  |
| --- |
| Instructions for transferring the files:  -Login on the bottom computer (Windows).  Download the following files:  Exam1.doc, question1.c, and question3.c.  Take notes of where you have saved these files (drive letter)  -Create three terminals in Windows:  type cmd.exe <enter> at the search bar 3 times  -In one of the terminals, change the drive to the drive letter you took notes of, for example, typing:  p: <enter>  Then do type:  cd Downloads <enter>  Then transfer the following files using scp:  scp question1.c [pi@192.168.0.227](mailto:pi@192.168.0.227): <enter>  -type the password “raspberry” <enter>  And same for question3.c:  scp question3.c [pi@192.168.0.227](mailto:pi@192.168.0.227): <enter>  -type the password “raspberry” <enter>  On each of the three terminals, do:  ssh [pi@192.168.0.227](mailto:pi@192.168.0.227) <enter> |

**Question 1 (50 marks)**

Open question1.c in any editor (pico/nano), and make sure you read and understand it. Note that this program has two infinite for loops. It is also relisted below to facilitate.

**//Operating Systems in Practice -**

**#include <sys/types.h>**

**#include <unistd.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**int main()**

**{**

**pid\_t r;**

**int i = 0;**

**/\* now create new process \*/**

**r = fork();**

**if (r > 0)**

**{**

**for (; ;)**

**{**

**i+=2;**

**}**

**}**

**else**

**{**

**if (r == 0)**

**{**

**for (; ;)**

**{**

**i+=2;**

**}**

**}**

**else**

**{**

**printf("error"); /\* display error message \*/**

**exit(0);**

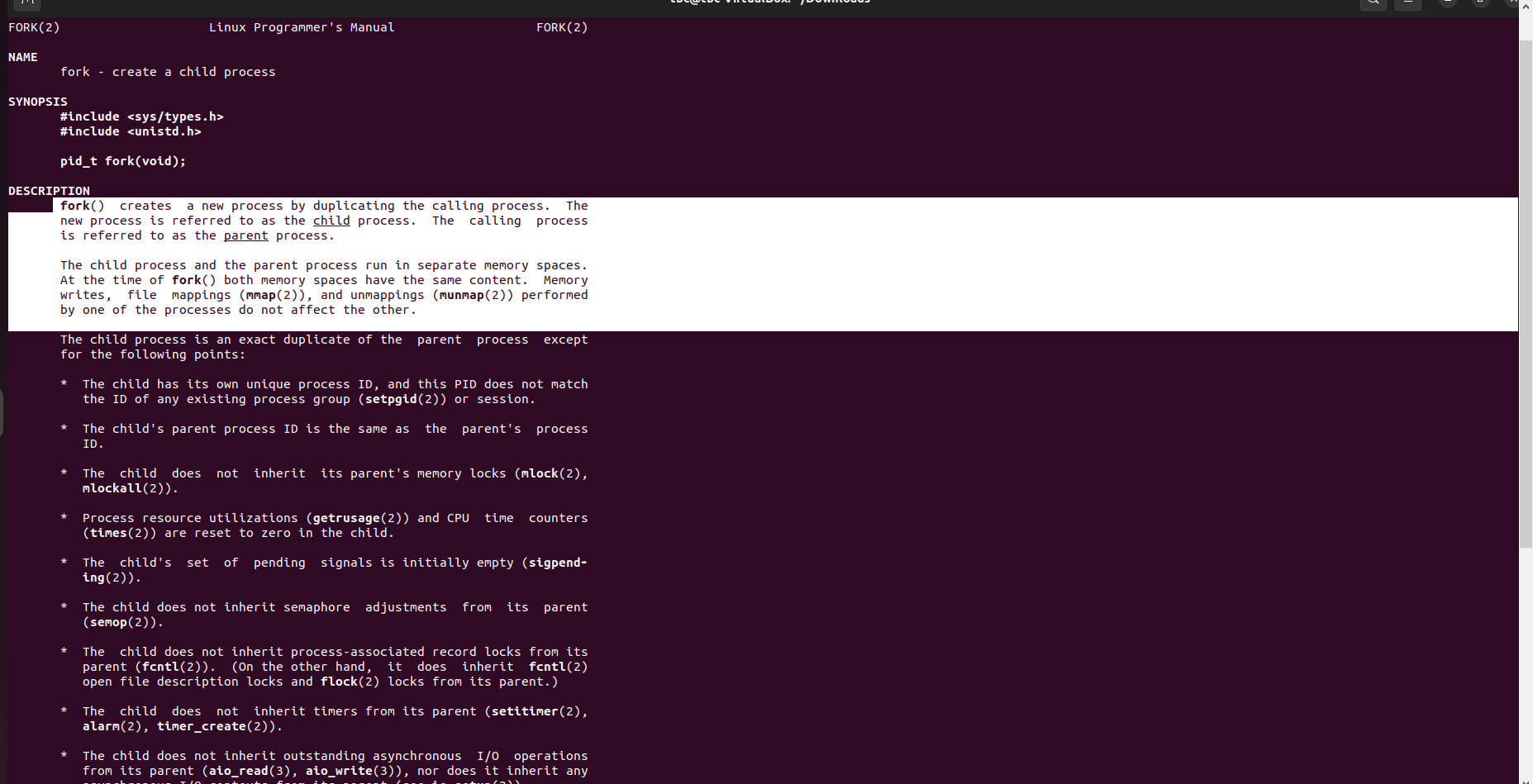
**}**

**}**

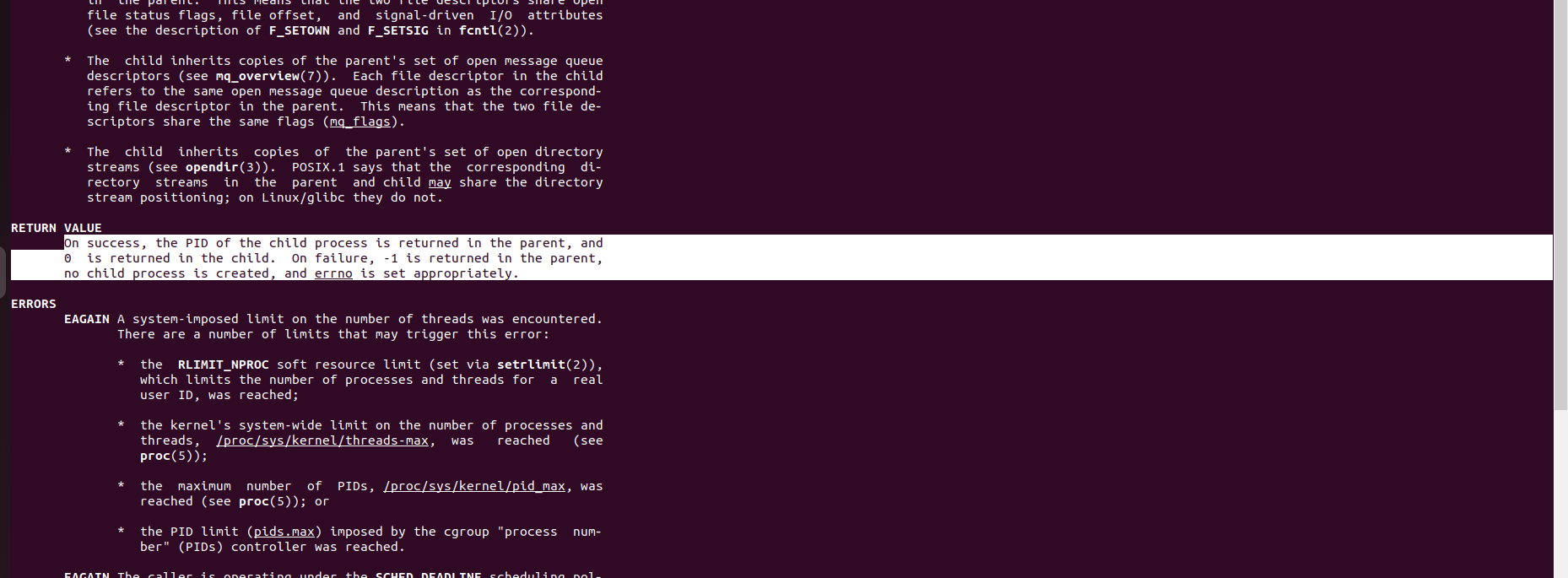
**}**

1a. You can see on the program above a C command named fork. Is fork a system call? Explain, and include a screenshot of fork manual to justify your statements. (5 marks)

Answers: Yes,a fork is a system call. Forking is a concept in operating systems that involves creating a new process by duplicating an existing process.



1b. What does fork return? Explain it and use a screenshot of the fork manual to justify your statements. (5 marks)

Answers: When a fork succeeds, it returns the PID of the child process to the child. Failure results in a -1 return to the parent process, and no child processes are generated.

1c. From the program above, extract the code executed by the parent, and list it below. Explain why the parent executes such code. (5 marks)

Answers: **r = fork();**

**if (r > 0)**

**{**

**for (; ;)**

**{**

**i+=2;**

**}**

**}**

= Given that the variable with the fork has a value greater than 1, the parent has performed this code.

The PID of the kid is returned when the child has been successfully created and is saved in the r. As a result, parent code is being run.

1d. From the program above, extract the code executed by the child and list it below. Explain why the child executes such code. (5 marks)

Answers: **if (r == 0)**

**{**

**for (; ;)**

**{**

**i+=2;**

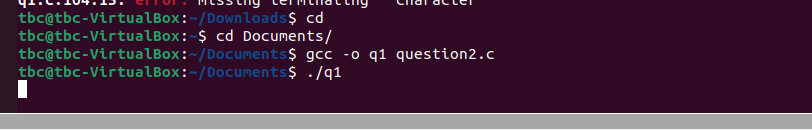
**}**

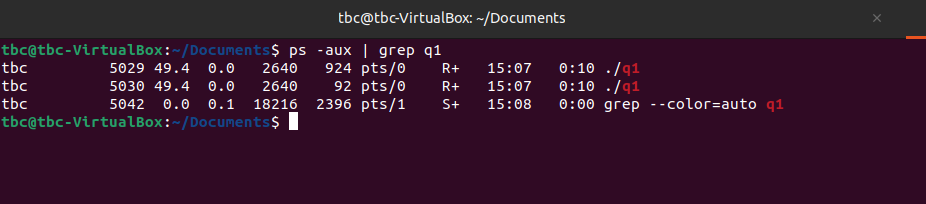
**}**

= This is code executed by the child as the variable r is equal to 0.

As a result, if the fork succeeds in creating a child process, it returns 0. We may infer that it is child code as it only executes specific code when r returns 0.

1e. Get one (combined) or two-three screenshots of the following: compile question1.c and generate the binary (hint: use gcc to compile and generate the binary named question1). Then, run question1. Next, use ps to get the PID, CPU usage of question1 processes. (5 marks)

Answers: 



For parent

PID : 5029

%CPU : 49.4

For Child

PID : 5030

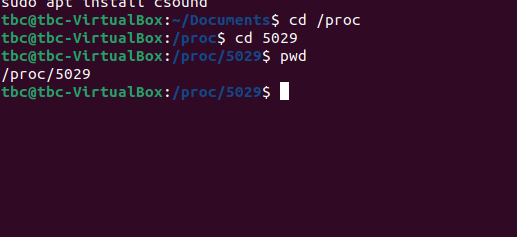
%CPU : 49.4

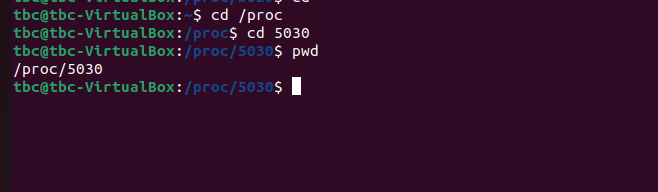
1f. Based on ps output of item 1e, explain which one is the parent process. Based on the source code (you have analysed in item 1c) and the output of item1e, explain the CPU usage of the parent. (5 marks)

Answers: Here, the process with PID 5029 is the child's parent since the child's PID is recorded in the variable r and is 1 greater than the parent's. Additionally, the process is still operating, using 49.4 percent of the parent's CPU.

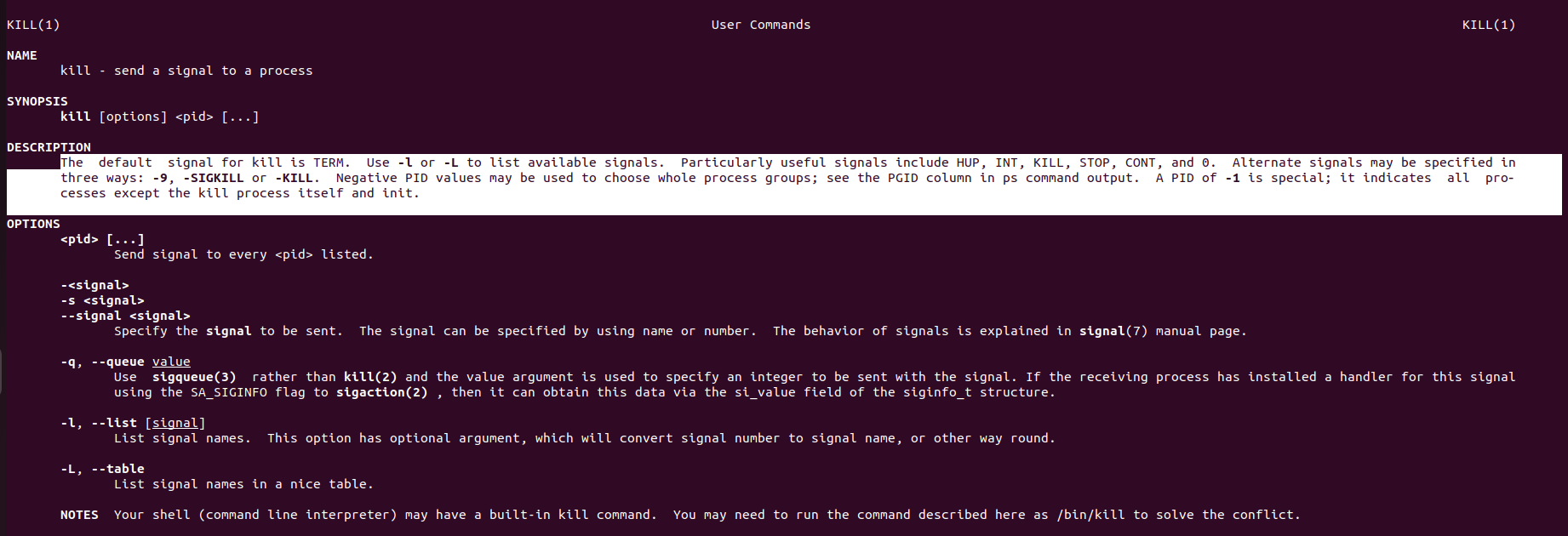
1g. Use the proc directory and the PID number obtained in item 1e to confirm the values of PID and state obtained in item 1e. Get a screenshot. (5 marks)

Answers:



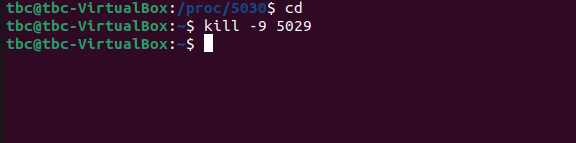
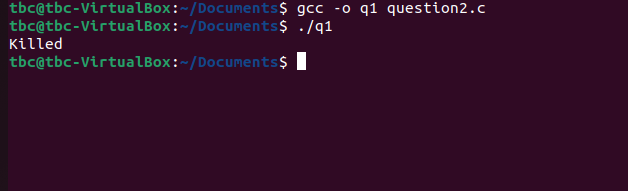
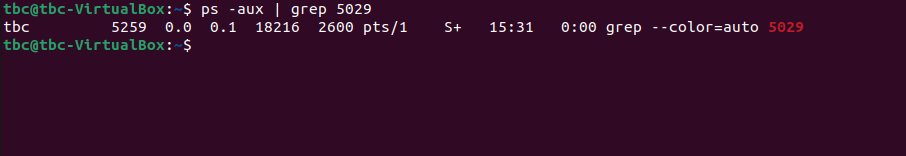


1h. We are about to use kill system call: to prove that kill is a system call, find it in syscalls manual (highlight it) and get a screenshot. (5 marks)

Answers: 

1i. Use kill to stop the parent. After the kill, similar to item 1e, get another ps screenshot of the parent. (total 5 marks).

Answers:

1j. In item 1e you have taken a screenshot before, and in item 1i another one, after the requested kill. Explain in detail the theory behind the differences observed between these two screenshots. (5 marks)

Answers: In 1e, the program is in the operating stage when the system assigns a distinct PID to each process and those processes require a little amount of CPU to ensure correct operation.

We used the kill command in 1i to end the process. Since the application is no longer executing, it will have no PID and a CPU percentage of 0.

*Note: Before proceeding to question 2, at the terminal where you are running question1, make sure to type <CRTL+C> to stop it or use kill to stop all processes of question1.*

**Question 2 (30 marks)**

2a. Assuming you have only two processes running in a computer managed by an operating system (OS), and that these processes are CPU-bound (only use CPU). Explain in detail how the process control block (PCB) is used by the OS to manage the execution of these processes. Make sure to include the “time slice“ term in your explanation. (total 15 marks)

Answers: The process control block oversees the execution of the processes by assigning a unique process number (PID) to each process so that it can be identified. The process state, such as Running or Sleeping, is then determined. Additionally, it verifies the files and registers utilized by each process. To obtain input or output, it verifies I/O status information. Time slices were utilized to regulate process time, giving each process time to execute as it should and according to priority.

2b. Briefly explain about the memory table used in the PCB (5 marks).

Answer: The memory allocations for each process are contained in the memory table used in PCB. It determines if the OS has adequate memory to support the process.

2c. In terms of operating systems (OS), task\_struct variable represents the code of the PCB in Linux. A student has found task\_struct and extracted some of its source code as listed below:

**struct task\_struct {**

**volatile long state;**

**void \*stack;**

**unsigned int flags;**

**int prio, static\_prio;**

**struct list\_head tasks;**

**struct mm\_struct \*mm, \*active\_mm;**

**...**

**pid\_t pid;**

**pid\_t tgid;**

**struct task\_struct \*real\_parent;**

**char comm[TASK\_COMM\_LEN];**

**struct thread\_struct thread;**

**struct files\_struct \*files;**

**...**

**}**

In Linux, you have used top and/or ps to monitor the computer. Explain how ps or top obtain data that come from the Linux kernel. (5 marks) In your explanation, make sure to use two variables of the above code to exemplify the flow of the data transferred (5 marks). (total of 10 marks)

Answers:

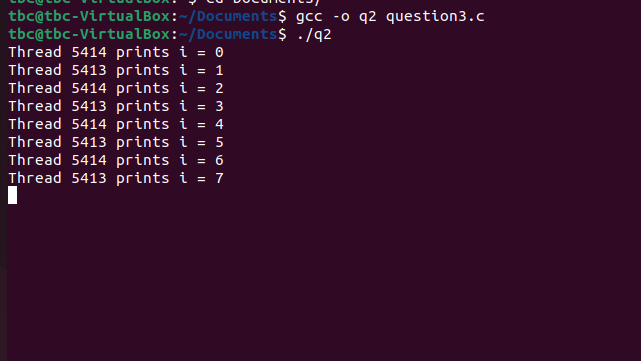
**Question 3 (20 marks)**

3a. Why are threads of lower overhead when compared to processes? (5 marks)

Answer: Threads have less overhead than processes because they only take up one memory space compared to the several memory spaces used by processes.

3b. Open question3.c in any editor (pico/nano) and make sure you read and understand it. Get a screenshot of the following: compile question3.c (hint: use gcc to obtain question3 binary, but note that this code has threads). Then, run it. (5 marks)

Answers:



3c. In the source code of question3.c you will notice that there is a variable that is shared by both threads: which variable is that? Explain briefly about the mechanism used to be able to share this variable. (5 marks)

Answers: Semaphore mechanism is used., sem\_wait, sem\_int

3d. Briefly explain how both threads are able to change the referred variable (shared) in item 3c (5 marks).

Answer: