

# CISC2005 Principles of Operating Systems

## Assignment 1

Release date: Feb 14, 2023

Due date: Feb 26, 2023 23:59

***No late assignment will be accepted***

**Every Student MUST include the following statement, together with his/her signature in the submitted homework.**

*I declare that the assignment submitted is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations.*

Signed(Student ) Date 24/2/23

Name WONG KWAI WAN SID 00026157

### **General homework policies:**

A student may discuss the problems with others. However, the work a student turns in must be created COMPLETELY by oneself ALONE. A student may not share ANY written work or pictures, nor may one copy answers from any source other than one's own brain.

Each student **MUST LIST** on the homework paper the **name of every person he/she has discussed or worked with**. If the answer includes content from any other source, the student **MUST STATE THE SOURCE**. Failure to do so is cheating and will result in sanctions. Copying answers from someone else is cheating even if one lists their name(s) on the homework.

If there is information you need to solve a problem but the information is not stated in the problem, try to find the data somewhere. If you cannot find it, state what data you need, make a reasonable estimate of its value, and justify any assumptions you make. You will be graded not only on whether your answer is correct, but also on whether you have done an intelligent analysis.

## Question 1

What is the usage of Process Control Block (PCB)? List at least three data stored in the PCB.

PCB is the representation of process in OS. It contains all information of process.

- ③ 1) process state      3) PID  
2) hardware state      4) memory management

## Question 2

Write down the list of process state transitions that occur during the following program. You may assume that this is the only process that the CPU is executing.

```
1  int i = 1;  
2  while (i < 100) { i++; }  
3  printf("%d ", i);  
4  while (i > 0) { i--; }  
5  printf("%d ", i);
```

new → ready → running → ready → running → ready → running  
(wait for CPU) (first loop) (finish loop) (printf) (finish printf) (2nd loop)  
→ ready → running → terminate.  
(finish loop) (printf) (finish).

## Question 3

Consider the following code:

```
1  int main() {  
2      int pid;  
3      pid = fork(); child process  
4      if (pid == 0) { // child process  
5          sleep(5); // wait for 5 seconds  
6          int ppid;  
7          ppid = getppid(); // get parent's pid  
8          printf("ppid is: %d\n", ppid);  
9          execlp("/bin/ls", "ls", NULL);  
10         printf("Child Process Done!");  
11     }  
12     return 0;  
13 }
```

(a) Will the program print "Child Process Done!"? Why?

(b) Suppose that the pid of the parent process is 2005, what does the child process print in line 8? Why?

a) Nope, it won't print that, because for child process, as it runs 'execlp' successfully, what is under it will not be executed. unless there is error on 'execlp' line.

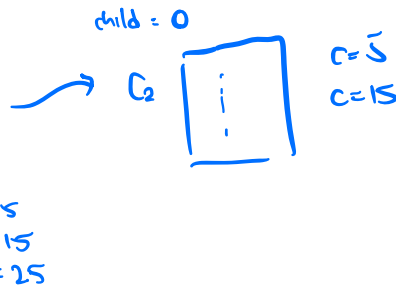
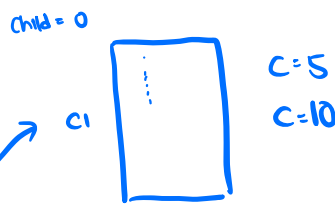
b) will print 2005 since it's printing parent's PID and because it's store in variable ppid using function getppid().

## Question 4

Consider the following code:

```

1  int child = fork();
2  int c = 5;
3  if (child == 0) {
4      c += 5;
5  } else {
6      child = fork();
7      c += 10;
8      if (child) {
9          c += 10;
10     }
11 }
```



- Including the initial parent process, how many Processes are created by this program?
- In each of process, what is the final value of variable  $c$ ?
- Suppose a program calls  $fork()$   $n$  times. Proof: Including the initial parent process, there will be  $2^n$  processes in total.

a) 3 processes including initial parent

b) ambiguity, have 2 solution

① If child means child process, then

C1, C = 10

C2, C = 25

P1, C = 15

② If child means the variable child, and  $child = 0$  is false, otherwise is true, then

C1, C = 10

C2, C = 15

P1, C = 25

c) math induction.

↳ base case : call 1 time, there will be  $2^1 = 2$ , true, (1 parent, 1 child)

↳ assume it's true for  $k$ , call  $fork()$   $k$  time, there will be  $2^k$  processes

∴ proof for  $k+1$ ,

$\therefore$  call `fork()`  $k+1$  times,

for example:

`fork()`  
`fork()`  
 $\vdots$   
`fork()`  
`fork()`

}  $k$  times, as above, have  $2^k$  process

$\therefore$  for the last `fork()` function, it will create 2 process (1 parent, 1 child)

for each of the  $2^k$  process, they will run the final `fork()`

$$\begin{aligned}\therefore 2^k \cdot 2 &= 2^k \cdot 2^1 \\ &= 2^{k+1} \quad (\text{proved}) \quad \#.\end{aligned}$$

$\therefore$  prove for all  $n$  cases.

## Question 5

Shell is an environment in which users can run their commands, programs, and shell scripts. It gathers input from users and executes programs based on that input. When a program finishes executing, it displays that program's output.

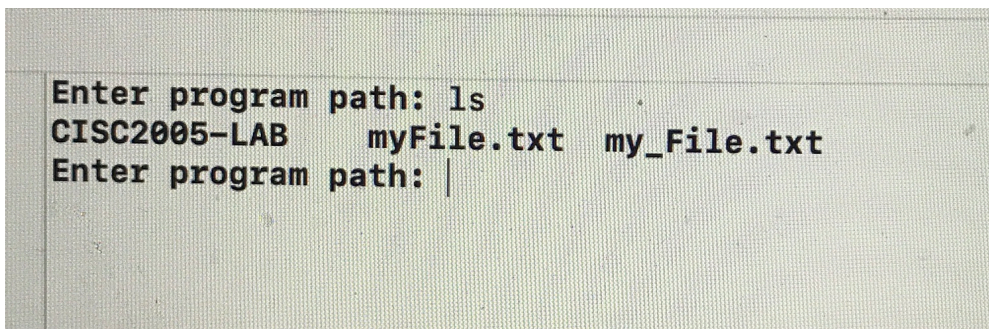
Here we are trying to build a simple shell with the following running loop:

1. Waiting for input of the user.
2. Read input (program path) from the user.
3. Execute the program based on the program path.
4. Display the output of the program.
5. Return to the first step.

The program should be written in C.

Hints:

1. Make use of *execlp()*, *wait()*, *fork()* and *scanf()*.
2. Test your program under MacOS/WSL/Linux environment, input "ls" and check the output.



```
Enter program path: ls
CISC2005-LAB myFile.txt my_File.txt
Enter program path: |
```