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 黄彦臻) Date Feb. 14, 2023

Name Huang Yanzhen SID DC126732

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.

(a) PCB serves as the repository for all the data needed to start / restart a process.

(b) Process State, Program Counter, CPU registers.

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);
```

Lines of code: 1 → 2 → 2 → 3 → 4 → 4 → 5 → after 5.
Process State: New → Waiting → Ready → Running → Waiting → Ready → Running → Terminated

Question 3

Consider the following code:

```
1    int main() {
2        int pid;
3        pid = fork();
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) No. After execution of "execlp" function, execlp wouldn't return if no error occurs. Hence "printf" won't be reached.

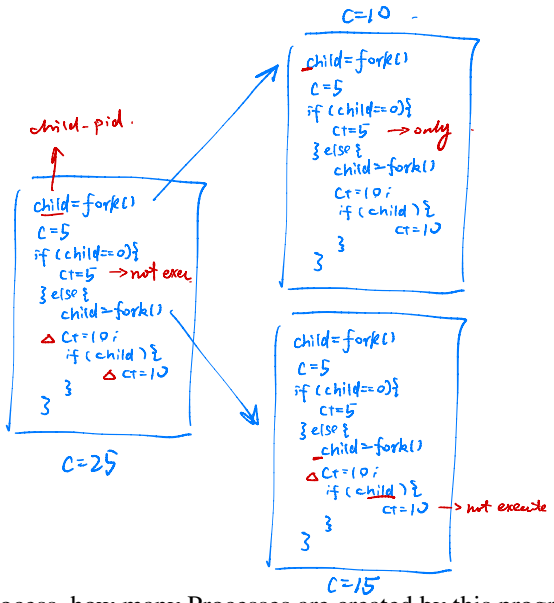
(b) Output:
ppid is: 2006
Because its parent pid is 2005, and getppid() gets the pid of its parent.

child pid.

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   }
```



(a) Including the initial parent process, how many Processes are created by this program?

(b) In each of process, what is the final value of variable c?

(c) Suppose a program calls $fork()$ n times. Proof: Including the initial parent process, there will be 2^n processes in total.

(a) 3.

(b) 25, 10, 15

(c) Proof:

Let x be the order of fork from the lowest to highest.

$f(x)$ be the total num. of child process the x-th fork can create.

The i-th fork() will create a process with executable fork starting from #1 ~ #i-1, which can generate $f(i-1)+f(i-2)+\cdots+f(1)$ processes. So, we have:

$$f(x)=f(x-1)+f(x-2)+\cdots+f(1)+1$$

$$\Rightarrow f(x+1)=f(x)+f(x-1)+\cdots+f(1)+1$$

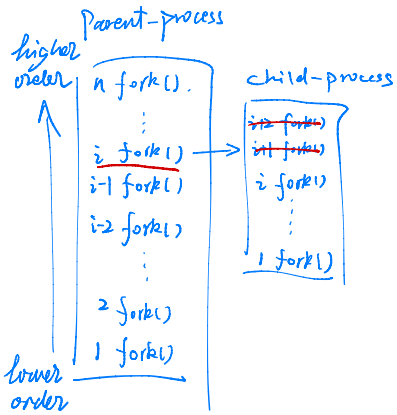
$$\Rightarrow f(x+1)=2f(x),\quad \text{where } f(1)=1,\ f(2)=f(1)+1=2,\ f(3)=4,\ \ldots$$

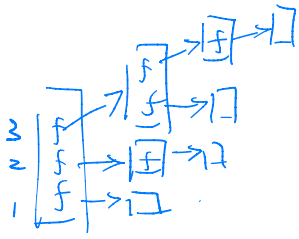
Hence, $f(1), f(2), \ldots, f(n)$ is a geometric sequence with $f(1)=1$, and $q=2$.

i.e. $f(x)=1\cdot 2^{x-1}=2^{x-1}$

In total, n fork() would generate: $2^0+2^1+\cdots+2^{n-1}=\frac{2(1-2^n)}{1-2}=2^n-1$ processes.

Including the original parent process, we have: $2^n-1+1=2^n$ processes.





Question 5

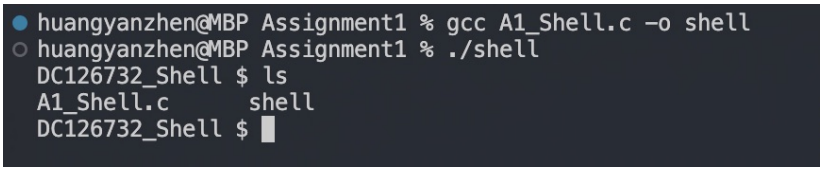
Shell is an environment in which users can run their commands, programs, and shell scripts. It gathers input from users and executes programs based on that input. When a program finishes executing, it displays that program's output.

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.



Partly Followed the tutorial: https://brennan.io/2015/01/16/write-a-shell-in-c/