Operating System Midterm exam

1) For each of the following process state transition, say whether the transition is legal and how the transition occurs or why it cannot.
    a. Change from thread state WAIT to thread state RUNNING

This legal transition occurs when a child process of a process has terminated, resulting in the wait() call on the parent process terminating or otherwise.

    b. Change from thread state RUNNING to thread state WAIT

This legal transition occurs when a running process yields for a child process, meaning that wait() is called until its child process is finished.

    c. Change from thread state READY to thread state WAIT

This illegal transition cannot occur, for a process to wait it has to be running first.

2) Write a program that opens a file (with the open () system call) and then calls fork () to create a new process. Can both the child and parent access the file descriptor returned by open ()? What happens when they are writing to the file concurrently, i.e., at the same time?

Both the child and parent processes can access the file descriptor returned by open. Whatever happens when both processes write to the file is determined by the OS's race rules regarding processes, as there are no guards in the code otherwise. This means whatever is written can be perfect or completely garbled.

3) Write another program using a fork (). The child process should print "hello"; the parent process should print "goodbye". You should try to ensure that the child process always prints first; can you do this without calling wait () in the parent?

You can do this without calling wait(), by using usleep() to always delay the parent process by just a small margin.

4) Write a program that creates a child process, and then in the child closes standard output (STDOUT FILENO). What happens if the child calls printf () to print some output after closing the descriptor?

Calling printf() after closing standard output will cause printf() to not output anything, as the channel is already closed.

5) Consider the following piece of C code:

```
void main ( ) {
        fork ( );
        fork ( );
         exit ( );
}
```

How many child processes are created upon execution of this program?

Three processes. The first fork call creates one child process, and the second fork creates two child processes, one by the original parent process and the other by the child process created by the first fork call.

6) An interactive shell program such as bash shell (terminal in Linux and Mac) or PowerShell or CMD prompt in Windows takes command line input from the user and then execute the command/program specified by the user. In this exercise, you will implement closh (Clone Shell), a simple shell-like program designed to run multiple copies of a program at once.
   Like any other shell, closh takes as input the name of the program to run (e.g., hello world). However, closh also takes two additional inputs:
   1) The number of copies (processes) of the program to run. This is an integer from 1 to 9.
   2) Whether the processes should be executed concurrently or sequentially, in sequential execution, the shell should wait for every time a program is executed. In case concurrent execution, the shell does not need to wait for the program to complete execution.

   Closh executes the given program, the specified number of times, then returns to the prompt once all processes have either completed. Here is a simple example of using closh

   (italic is user input and hello.exe is the hello world as in previous question):
   mint@mint:~$ ./closh
   closh> ./hello.exe
   count> 3
   [p]arallel or [s]equential> p
   hello world
   hello world
   hello world
   closh>

   To write such a shell in C++, refer to the process lecture. As noted in the slide, you can use “execvp()” to create a new child process and have it execute a command. Use also the “waitpid” in the parent process to wait for the child process to finish in case
   sequential execution is selected.
   For simplicity, assume that the user specifies the full path name for any command/executable that they wish to execute. Thus, you do not need to deal

with path name completion issues. You can test your shell on programs you write in C++ (after compiling to machine language).

Create a GitHub repository and upload this document with answers to question 1 and 6 with all your .cpp or .c programs. Copy the repository link and paste it in Canvas assignment.