CSE-2431 Homework 1

Yifan Yao.740

Question 1

Two processes are created when the program runs, a parent process and a child process (forked from parent). The output from two processes are intersect. Parent process will print values from 1 to 8. Child process will print values from 9 to 16.

1. Results are expected as mine, since there are many other processes were running on the OS, it is not possiable to predict the behavior of the CPU scheduler to schedule two process run on the CPU.
2. Everytime when I run the program, the sequence of values are different.
3. Since there is only one process can make I/O system call, therefore, the another process have to wait unitl the pervious I/O system call has been completed which may cause the difference of the sequence of values during execution. Also, the CPU scheduler is also unpredictable because two process are independent.

Question 2

Two processes are created when the program runs, a parent process and a child process (forked from parent). The output parent process will output the result first, then the child process will output the result. Parent process will print values from 8 to 14. Child process will print values from 1 to 7.

1. Outputs were printed by both processes, and when the output is done, the parent process will print "I am Process2, pid xxxx: num = 10", child process will print "I am Process1, pid xxxx: num = 15"
2. Since the `fork()` was called before the num value changed, therefore, the value of num will only changed when the process is a child process.
3. When `fork()` was called, the PID of the child process will be returned in parent process. Also, when the process is a child process, return value will become 0. The value of num will only change when the return value from `fork()` is 0, which means only in

child process, the value of num will get changed.

Question 3

Two processes are created when the program runs, a parent process and a child process (forked from parent). The output parent process will output the result first, then the child process will output the result. Parent process will print values from 7 to 12. Child process will print values from 1 to 6.

1. Outputs were printed by both processes, and when the output is done, the parent process will print "I am Process1, pid xxxx: num = 8", child process will print "I am Process2 pid xxxx: num = 18"
2. Since the `fork()` was called before the num value changed, therefore, the value of num will only changed when the process is a child process.
3. When `fork()` was called, the PID of the child process will be returned in parent process. Also, when the process is a child process, return value will become 0. The value of num will only change when the return value from `fork()` is 0, which means only in child process, the value of num will get changed.

Question 4

Four processes are created when the program runs, a parent process and a child process (forked from parent in `main()`), two grandchilren process (forked from child process twice in `Code1()`).

1. See below

```
                                        / grandchild 1 (process 3)
parent (process 1) -- child (process 2)
                                        \ grandchild 2 (process 4)
```

Question 5

In fact, there are two programs were running on the OS and the logic for both parent and child process can be total different and they have own memory address which means they are running independently; the result can be significantly different. Also the parent process and child process have their own memory spaces and they can only access their own parts.

Question 6

Process Identifiers (PID) is used to identify each process in the OS. It is useful because when the process is uncontrollable, we can use PID to instruct system to kill the specific process. Also, there are two pids associated with the child process because child process has it own PID, additionally, the child process is associated with a parent process which has a parent PID (PPID).

Question 7

> ps - report a snapshot of the current processes.
>
> -f Do full-format listing. This option can be combined with many other UNIX-style options to add additional columns.
>
> -a Select all processes except both session leaders (see getsid(2)) and processes not associated with a terminal.
>
> `ps` : PID TTY TIME CMD
>
> `ps -f` : UID PID PPID C STIME TTY TIME CMD
>
> `ps -af` : UID PID PPID C STIME TTY TIME CMD

`ps` will display the process information associated with current terminal. `-f` option will provides more details about processes running on the OS like user id (UID), PPID, process start time (STIME). `-af` option will provide details about child process's partent processes and processes not associated with a terminal.

Question 8

> kill - terminate a process

We can use `ps` to locate/identify a process and use `kill` to send a signal to terminate that process, there are many signals we can do (see `kill -l`), like we used to use option SIGKILL (`-9`) to force kill a process. By default, `kill` command will send a SIGTERM (`-15`) to terminate a process gracefully.

Question 9

Duplicate an exactly same process of current process. The `fork()` will return 0 in child process, return PID of the child process in parent process. The child process may behavie differently than parent process, depends on progarm logic.

Question 10

Return the Process Identifiers (PID) of the program who was calling it.

Question 11

Return the Parent Process Identifiers (PPID) of the program who was calling it.

Question 12

`wait()` can be used when the program need to be suspended until designated process terminated. If PID is provided, `wait()` will wait the specific process terminated, then continue execution, otherwise, `wait()` will wait all active child processes and return 0. If the given PID or job is not exist, `wait()` will return 127.