Some exercises on Processes

1. Write a C Ubuntu program in which the child process executes a command to create a new directory OperatingSystems

2. Write a C program in which the child process changes the name of the directory OperatingSystems into OS

3. Write a C program in which the child process copy the directory OS into OS1

4. Write a C program in which the child process delete the directory OS

5. Write a C program that generates a chain of processes of length 5 (not including the source process) in which the PID and PPID of each process is displayed

6. Write a C program that generates a fan of processes of length 5 (not including the source process) in which the PID and PPID of each process is displayed

7. Describe what happens in the kernel when the following command is executed in a Ubuntu command line interpreter (a shell): ls -l

8. Explain what happens before and when you type a command in a Ubuntu command line interpreter (a shell) if you don't put & after the command (that is the command is not running in background)

9. A shell command that is not follow by & is said be running in foreground. What can you do to send a SIGINT signal to such a process running in foreground?

10. Start a process that will last some time (by executing the sleep() command at the end of process code). Take the PID of this process and follow this process up the process hierarchy via PPIDs until the top-most process is reached. Make sure you understand what each process is in your process tree.

11. * When exec() is called, which part of the following PCB fields is modified? the PCB is completely re-written; the PID is changed; the parent pointer is modified; the program counter is updated;

12. * There are 4 sections in the address space of a process: text, stack, heap, and data. Which of these sections are replaced when exec() is called?

13. Consider the following program:

```
int i;
for (i=0; i<3; i++)
   fork();
printf("xin chao");
```

    How many times this program print "xin chao"?

14. Consider the following program:

```
int i;
for (i = 0; fork(); i++) {
  if (i == 4) { break; }
  printf{("PID: %d, i = %d \n", getpid(), i);
}
```

(a) Excluding the calling process, how many new processes are created?

(b) Assume that initially the process ID executing this code is 3376, and the PID always increases sequentially by 1. For example, the first time this process calls fork() the PID of the child is 3377. You can also assume no other processes in the system are calling fork() after this program starts to execute. What will be printed by this program?

15. How many processes are created by the following code

```
int main() {
    int i;
    for (i = 0; i < 10; i++)
      if (i % 2 == 0) fork();
    exit(0);}
```

16. A programmer has written a command line interpreter as follow:

```
bash(...){
 ...
    exec(cmd, args);
    fork();
 ...
}
```

Which statement describes the most accurately this code ?

- Will not work as the NULL parameter is missing in exec();
- It will work, a shell program must fork and replace itself with the user command;
- Will not work because the shell program is replaced by cmd, the shell will terminate once cmd exits;
- Will not work, the shell should create a thread and then call exec().

17. Including the initial parent process, how many processes are created by the program below?

```
int main()
int i;
pid_t pid;
for (i=0;i<5;i++)
  pid = fork();
return 0;
```

18. In the program below a new process is created and the program waits for the process termination. After that, a new process is created and the whole procedure is repeated. Modify this program to create 2 processes that run concurrently and the parent process waits for the termination of both of them.

```
#include <sys/types.h>
#include <stdio.h>

int main(int argc, char *argv[] )
{
       pid_t pid;
       int status;

       pid = fork();
       if (pid != 0)
       {
           while (pid != wait(&status));
       }
       else
       {
           sleep(5);
           exit(5);
       }

       pid = fork();
       if (pid != 0)
       {
           while (pid != wait(&status));
       }
       else
       {
           sleep(1);
           exit(1);
       }
}
```

19. * The following recurrence takes a positive integer n and eventually reach 1:

$$n = \begin{cases} n/2 & \text{if } n \text{ is even} \\ 3 \times n + 1 & \text{if } n \text{ is odd} \end{cases}$$

For example, if $n = 35$, the sequence is

$$35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1$$

Write a C or C++ program using the fork() system call that generates this sequence in the child process. The starting number will be provided from the standard input. Have the parent invoke the wait() call to wait for the child process to complete before exiting the program.

20. Does the process state model with 2 states, ready and running make sense? Explain your answer.

21. Enter the names of the states in the diagram of the process state model with 5 states.

22. Give three conditions that cause switching from user mode to kernel mode.

23. What is a system call? How does a system call differ from a call to a method or a function in a software library? What kinds of services can be obtained through system calls?

24. Using the execl() command, change the program below such that the child processes will execute the command "date". How many processes will execute the fprintf() command, which one execute it and why?

```
#include <stdio.h> #include <stdlib.h> #include <unistd.h> #include <sys/wait.h>
/*create a fan of processes (a tree of level 1). */
int main (int argc, char *argv[]) {
pid_t childpid = 0;
int i, n;
n = atoi(argv[1]);
for (i = 1; i < n; i++)
  if ((childpid = fork()) <= 0) break;
fprintf(stderr, "i:%d process ID:%ld child ID:%ld, parent ID:%ld\n",
i, (long)getpid(), (long)childpid,(long)getppid());
 wait(NULL);
return 0;
}
```

25. A process that has been running is now waiting for the user input. What is among the 5 possible states, the state in which the process is when waiting for the user input?