==========

Tutorial 02 – Process Management

Operating Systems Principles and Programming [18ECSC202]

Semester: IV Batch: D Slot: Week 02

==========

Note:

- This is an individual assignment.
- Submit your answers in the Google form link before you complete the session: **tinyurl.com/ospp-tute-02-2022**
- This tutorial assignment is for practice session and to get familiar with fork, wait and exec APIs.

PART I: Demonstration

1. Understand the program given below:

```
#include <stdio.h>
#include <unistd.h>

void PC() {
    if(fork() == 0)
        printf(“child\n”);
    else
        printf(“parent\n”);
}
int main() {
    PC();
    return 0;
}
```

The output can be in any order based on the CPU execution of child or parent first and next. Note that after fork child is returned 0 and its pid is sent to parent.

2. What is the output of the below program?

```
#include<stdio.h>
#include<unistd.h>

int main() {
    pid_t pid, ppid;
    fork();
    printf("%d=process id, %d=parent id\n", getpid(), getppid());
    return 0;
}
```

3. Usage of exec and wait

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <stdlib.h>

int main() {
    pid_t pid;
    pid = fork();

    if (pid < 0) {
        printf("fork error");
        exit(-1);
    }
    else if (pid == 0) {
        execlp(“/bin/ls”,“ls”, NULL);
    }
    else {
        wait(NULL);
        printf(Execution of child is complete\n);
     }
    return 0;
}
```

PART II: Exercise

4. Consider the program given below:

```
#include <stdio.h>
#include <unistd.h>
int globvar = 10;

int main() {
    int var = 100;
    pid_t pid;

    printf("Before fork\n");
    printf("PID = %d, Global Variable = %d, Variable = %d\n",
                                        getpid(), globvar, var);


    pid = fork();
```

```
    if (pid < 0) {
        printf("fork error");
    }
    else if (pid == 0) {
        globvar++;
        var++;
    }
    else {
        sleep(2);
     }

    printf("PID = %d, Global Variable = %d, Variable = %d\n",
                                        getpid(), globvar, var);
    return 0;
}
```

What will be the output of the program? Explain your understanding. 5 Points

5. For the program given in question 4, make the following change:

- Call vfork() instead of fork(

What will be the program output? Why? 5 Points

6. Consider the program given below:

```
#include <stdio.h>
#include <unistd.h>

int main()
{
    if (fork() || fork())
        fork();
    printf("1 ");
    return 0;
}
```

What will be the output? Why? 5 Points

7. For the given program, what will be the output? 5 Points

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

int main()
{
   if(fork()==0)
   {
      printf("2");

      if(fork()==0)
      {
          printf("4");
      }
      else
      {
          printf("3");
      }
   }
   else
   {
      printf("1");
   }

   return 0;
}
```

8. For the program given below, what will be the output? 5 Points

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

int main()
{
    int i;
    for (i = 0; i < 2; i++ ) {
        fork();
        printf("-");
    }
    return 0;
}
```

PART III: Theory

9. Should a terminating process also terminate all of its children?
   - Give an example where this is a good idea and
   - Another example where it is a bad idea.

5 Points

10. Give two differences between fork() and vfork(). 5 Points

11. The process table contains all the information the operating system keeps about each process. The process table is normally kept in the operating system's own memory and is not accessible by user processes. Why don't the user processes need access to the process table?

Could we put the process table information for each process in the address space of each process? What problems would that cause? 5 Points

"Break a Leg"
- Prakash Hegade