CS220 - Computer System II

Lab 11

Due: 11/16/2017, 11:59pm

1 Introduction

In this lab, you will write a programs to investigate the behavior of `fork() and exec()` system calls. Create a folder **Lab11**. You will implement your code in this folder. The operating system associates a unique unsigned integer id with each process. This id can be retrieved using the function `getpid()`. Further, the OS also maintains the information about the parent process that created a given process. Parent process' id can be retrieved using `getppid()`. You will use `getpid()` and `getppid()` along with `fork()` and/or `exec()` to examine how `fork()` and `exec()` system calls behave.

2 Single Fork

Create **fork.c** with the following contents. Also, create **lab11.txt** to record your findings.

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

int main() {
    int x = 1;
    pid_t pid;
    pid = fork();

    if (pid ==0) {
        x++;
        printf("In child\n");
    } else {
        x--;
        printf("In parent\n");
    }
    return (0);
}
```

Compile and run the program.

1. Why is the order of print statements in the output inconsistent across multiple runs?

2. Introduce a 1 second delay by calling the `sleep()` function before the print statement in parent. Recompile and run. What is the the difference in ordering of statements

2

now and the before?

3. Add a line to the parent code that prints the process id (pid) of the child. Also, add a line to the child that calls `getpid()` and then prints the process id that is returned. Compile and run your modified program. What are the two values?

4. Now, print the parent's process id in both child and parent. You can obtain it by calling function `getppid()`. What are the two values?

5. Add a line to print the value of x in the child after it has been incremented. Add a line to print the value of x in the parent after it has been decremented. Recompile the program and rerun it. Why isn't the last value printed 1?

3   Multiple Forks

Create **multifork.c** with the following contents:

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

int main() {
    fork();
    printf("Line 1\n");
    fork();
    printf("Line 2\n");
    if (fork() == 0)
        printf("Line 3\n");
    else
        printf("Line 4\n");
    return 0;
}
```

Compile and run multifork.

1. What is the total number of times each of the lines is printed? Why?

2. Modify the program to print the pids and parent pids. (e.g., `printf("%d: %d: Line 1\n", getpid(), getppid());)`). Capture the output in **lab11.txt**. Why is the parent pid 1 for some processes?

3

4 Exec

Implement **runner.c** that simply obtains the pid using getpid() and prints it (e.g., `printf("My pid is %d, parent pid is %d.\n", getpid(), getppid());`). Compile it to generate the program runner.

Next, create **exec.c** with the following code:

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

int main()
{
  char *prog = "./runner";
  /* TODO: Initialize args here */
  if (fork() == 0) {
    printf("Child pid = %d, parent pid = %d\n", getpid(), getppid());
    execvp(prog, args);
    fprintf(stderr, "exec: %s\n", strerror(errno));
  } else {
    wait(NULL);
  }
  return 0;
}
```

Compile **exec.c** to generate program **exec**, and run it. Because **exec** replaces the current process image with the program being loaded, you will find that the process id is retained. Record the output in **Lab11.txt**.

5 Submission

Compress all .c source files and your Lab11.txt document in Lab11.tar.gz and upload to MyCourses.