IT 694 - CN

Date: 08/02/2022

Assignment : 2

List of Topics: Inter-Process Communication(PIPE), Fork()

Introduction to Fork():

The fork system call is used to create a new process called the child process, which runs concurrently with the process that makes the fork() call (parent process). After creating a new child process, both processes will execute the next instruction following the fork() system call. A child process uses the same pc(program counter), CPU registers, and open files used in the parent process.

Library: #include<unistd.h>

Below are different values returned by fork().

- Negative Value: the creation of a child process was unsuccessful.
- Zero: Returned to the newly created child process.
- Positive value: Returned to parent or caller. The value contains the process ID of the newly created child process.

How to get the process id of any processes?

- pid_t getpid() : Process id of current process
- pid_t getppid(): Process id of the parent process

Understanding fork using examples:

A call to fork() might fail with a return value of -1. For each problem below, assume that fork() succeeds at every call.

1. **Enumerate all possible outputs of the following program.**

```
int main() {
int x = 3;
if (fork() != 0)
printf("x=%d\n", ++x);
printf("x=%d\n", --x);
exit(0);
}
```

2. **How many “hello” output lines does this program print?**

```
void doit() {
if (fork() == 0) {
fork();
printf("hello\n");
exit(0);
}
return;
}

int main() {
doit();
printf("hello\n");
exit(0);
}
```

Basic code for the fork

Snippet 1:

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

```
int main()
{
     if(fork()==0)
     {
         printf("\nI am ChildProcess, My PID: %d",getpid());
         printf("\nI am ChildProcess, My Parent PID:
%d",getppid());
     }

     else
         printf("\nI am ParentProcess, My PID: %d",getpid());
     return 0;
}
```

Snippet 2:

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

void forkexample(int n)
{
    // creates 2^n - 1 child processes
     for (int i = 0; i < n; i++)
         fork();

}
int main()
{
    printf("Hello World\n");
     forkexample(4);
     return 0;
}
```

Pipe:

A child and a parent process (or any two related processes) can communicate with each other using a pipe() system call.

Header file: unistd.h

```
int pipe(int fd[2])
int fd[2];
int status
status=pipe(fd)
```

status=0 for success
        -1 for error

fd[0] is open for reading
fd[1] is open for writing.

Basic script to do communication between 2 process

```
#include <stdio.h>
#include <unistd.h>
#define MSGSIZE 16
char* msg1 = "hello, world #1";

int main()
{
char inbuf[MSGSIZE];
int p[2], i;
// To check creation of pipe
if (pipe(p) < 0)
exit(1);
/* write pipe */
write(p[1], msg1, MSGSIZE);
/* read pipe */
read(p[0], inbuf, MSGSIZE);
printf("% s\n", inbuf);
return 0;
}
```

Exercise:

1. Using the fork, you need to create 4 children of a parent. And by use of the IPC mechanism, you need to set up communication between them.
   Show your communication set up by the following scenario by sending echo packets:
   a. Parent to it's all child
   b. Any one child to its parent
   c. Any one child to it's all siblings

Submission:

1. Submit assignment with the report consists of an input file and output file with proper explanation of each output of all the exercises in pdf format.
2. Add all the outputs and a brief description of the commands used in the given demo scripts in the report.
3. Submitted code in a report should be well commented.
4. Submit a zip file with your student id, which will consist of the folder for each script & respective outputs. one common report in a parent directory.

   Eg:

   group-id-2021***.zip

   - report.pdf
   - PR1
     - Script
     - Its respective output
   - PR2
     - Script
     - Its respective output

5. Submission Deadline: 12/02/2022(Saturday), 23:59:00

6. Penalties will be imposed for the late submission.