219223 Computer System Laboratory

Homework 5: Process API

This is a homework that will prepare you for a more extensive lab on processes. You will learn about the use of important process APIs such as fork(), wait(), and exec(). Page 1 to 5 of the attached cpu-api.pdf contains the reading materials that will be useful when doing this homework.

Download the file cpu-api.zip, unzip it, and enter the cpu-api directory. Then:

```
make
```

to get all the executables for p1, p2, and p3.

Answer the following questions in a file named StudentID_Firstname_hw5_ans.pdf

1. Study p1.c source code and run p1

```
./p1
```

- Try running p1 multiple times and see what remains the same and what differs from one run to another.
- Compare your result to your friend's result and see if there are discrepancies.
- Do research and explain the relationships between parent process and child process. In addition, describe what getpid() is for and what exactly is pid.
- When we use fork() to create a new process, how can we distinguish a parent process from a child process.

2. Study p2.c source code and run p2

```
./p2
```

- How does p2.c differ from p1.c?
- What exactly does the wait() API do that makes the run in p2 differ from that in p1?
- Try running p2 multiple times and explain how the wait() API make the result for p2 more predictable than that of p1.

3. Study p3.c source code and run p3

```
./p3
```

- Explain how this program functions.
- What is the difference between exec() (the version of exec() used here is execvp()) and fork()?

If you are comfortable with fork() and other process APIs you have learned, can you predict and explain the outcome of the following executions?

4. Can you predict and explain the result when you run the following code?

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

5. Can you predict and explain the result when you run the following code?

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

Submission:
• Submit the StudentID_Firstname_hw5_ans.pdf file to the course's Google Classroom before the due date

1. Study p1.c source code and run p1

```
./p1
```

- Try running p1 multiple times and see what remains the same and what differs from one run to another.
- Compare your result to your friend's result and see if there are discrepancies.
- Do research and explain the relationships between parent process and child process. In addition, describe what getpid() is for and what exactly is pid.
- When we use fork() to create a new process, how can we distinguish a parent process from a child process.

- Number of parent and child are same but changing everytimes that we run. Pid is also different from child
- Different from my friend because sometimes child is not showing.
- Parent process and Child process are the same
  getpid() : is returns the process id of parent and child.
  pid : only return process id of parent
- By using pid

2. Study p2.c source code and run p2

```
./p2
```

- How does p2.c differ from p1.c?
- What exactly does the wait() API do that makes the run in p2 differ from that in p1?
- Try running p2 multiple times and explain how the wait() API make the result for p2 more predictable than that of p1.

- Child come first and then parent come second.
- parent process waiting for child process to finish first so the zombie process it won't happen
- child process is always finish first and return back to parent process

3. Study p3.c source code and run p3

```
./p3
```

- Explain how this program functions.
- What is the difference between exec() (the version of exec() used here is execvp()) and fork()?

If you are comfortable with fork() and other process APIs you have learned, can you predict and explain the outcome of the following executions?

- The child process is showing inside process because parent process wait until child process finish first.
- Fork is the starter of new process but it copy from the main process and exec is replace the current process with the new process.

4. Can you predict and explain the result when you run the following code?

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

Predict: 2 hello

Run code: 2 hello is showing because the zombie process is happen during the process

5. Can you predict and explain the result when you run the following code?

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

Predict: 8 hello

Run code: Only showing 2 hello. There is no wait in the parent process This make zombie process happen.