Assignment5 Report

Name: Huang Yingyi

Student ID: 119010114

File Tree

```
```

1. Running Environment

Version of OS: Ubuntu 16.04.5 LTS

OS Kernel Version: 4.10.14 (**Delete the static** keyword of do_wait function definition and **export** it by EXPORT_SYMBOL() before compile the kernel code)

My Test environment is the same as the above:

2. Execution Steps

Preparations

- Unzip the file and come to the directory of source file.

Execution

3. Program Design

Basic Task

-

Bonus Task

-
-

Details

-

4. Execution Results Demonstration

All the tests follow the execution steps above, and the test results are screenshots of output.

Basic Task

Bonus Task

5. Conclusion

According to the tests results, we can conclude that the design and code implementation of these three programs are successful.

In these tasks, I learnt:

- Parent process should reap the child process and analyse its exit status to free the remained occupied memory resources of un-reaped child process. (program1)
- When the program invokes system call, the corresponding kernel functions will be invoked to perform the tasks, as indicated in the program2, where do_wait() kernel function did the wait-like jobs.
- Draw a clear process fork chart is beneficial to write multiple-process programs. And we should choose appropriate inter-process communication methods to perform the message passing between processes. For example, PIPE/FIFO is useful when there is simple and high-frequency communication between two local processes who have parent-child relationship. Shared memory is useful when the size of memory is large and there are many processes to access the shared memory. However, designs such as mutex-lock should be implemented to avoid race between multiple writers. Socket is useful to make communication between

two processes (two ports) of different hosts, since the other methods cannot pass information through local access network. (program3)