SYSC 4001 Assignment 2 — Part III

Nate Babyak — 101310590
Ozan Kaya — 101322055

November 7, 2025

1 Introduction

The objective of this part of the assignment is to build a small API simulator. This simulator has one CPU, fixed partitions in memory, and is primarily focused at providing functionality for `fork()` and `exec()`. This final part of the assignment builds on the final part of the previous assignment, where we simulated an interrupt system.

2 Tests

2.1 Test 1

2.1.1 Analysis of `execution.txt`

Init forks a child process. The child executes `program1` using exec, which is loaded into a 10 Mb partition, and its PCB is updated. Upon child completion, the parent will execute `program2` in a separate 15 Mb partition. Each system call causes an ISR to be triggered, which will be followed by memory partition allocation and a scheduler call. The execution log contains timestamps for context switches, entry into kernel mode, and CPU bursts to demonstrate that fork/exec and priority handling are being sequenced correctly.

2.1.2 Analysis of `system_status.txt`

At time 24, the system executes `FORK, 10` creating a child process. The child (PID 1) inherits the parent's PCB and is made to run, while the parent (PID 0) is made to wait because children have a higher priority. Around time 256, the child executes `EXEC program1, 50`. Using best-fit allocation, `program1` (10 Mb) is allocated partition number 4 (10 Mb). Around time 633, after the child completes execution, the parent executes `EXEC program2, 25`, which allocates partition number 3 (15 Mb) to `program2` (15 Mb).

2.2 Test 2

2.2.1 Analysis of `execution.txt`

Init forks a child process. The child executes `program1` using `EXEC`. This program forks a child process, does nothing with it, then excutes `program2`. After this program executes, we return to the

2.2.2 Analysis of `system_status.txt`

Around time 31, the system executes `FORK`, creating a child process. The child (PID 1) has the same PCB as its parent (PID 0) and is set to running. Around time 229, the child executes `program1`, which takes 10 Mb, and is thus allocated partition number 4 (10 Mb). Around time 258, `program1` executes `FORK`, which creates another `program1` with PID 2 and at partition number 3. Around time 543, the child executes `program2`.

2.3 Test 3

2.3.1 Analysis of execution.txt

In this test, FORK is called and program1 is loaded into the parent, while child is left untouched. The child then runs until completion before the parent executes.

2.3.2 Analysis of system_status.txt

At around time 34, init executes FORK, which creates a child (PID 1) with the same PCB as the parent (PID 0). The child is set to running and, using best-fit allocation, is allocated partition number 5. At around time 286, the child has completed and the parent executes program1, which is 10 Mb, and is thus allocated parition number 4 (10 Mb).

2.4 Test 4

2.4.1 Analysis of execution.txt

This test starts with a FORK, which clone the current PCB into the process. Then, the memory image of program1 is copied into the child. This program also invokes FORK and this child executes program2. The programs run child-first, meaning program2 first, then program1, and finally program3 is executed by init.

2.4.2 Analysis of system_status.txt

At around time 26, FORK is invoked, resulting in a child process (PID 1). At around time 228, the child process executes program1, which occupies parition number 4. At around time 252, program1 executes FORK itself, creating another program1 with PID 2. This child then executes program2, which occupies paritition number 5 as 4 is already occupied. Finally, at around time 1791, after the two children have completed, init executes program3 (10 Mb), which is allocated parititon number 3 (15 Mb).

2.5 Test 5

2.5.1 Analysis of execution.txt

First, FORK is used to create a child (PID 1), which starts running. This child executes program1, which then runs to completion. The parent (PID 1) then executes program2, which also runs to completion.

2.5.2 Analysis of system_status.txt

At around time 32, init invokes FORK, which creates a child (PID 1) and allocates it partition number 5. At around time 274, the child process exeuctes program1, which is 12 Mb and is thus allocated parition number 3 (15 Mb). After program1 completes, the parent process executes program2, which is allocated parition number 2 (25 Mb) to fit its 18 Mb size. Finally program2 runs until completion.

3 Conclusion

In conclusion,

Appendix

- SYSC 4001 Assignment 2 — Part II
- SYSC 4001 Assignment 2 — Part III