SYSC 4001

Assignment 2 Report

Sydney Kuhn - 101302923
(Student 1)

Keon Foster - 101303908
(Student 2)

https://github.com/Jeveloper25/SYSC4001_A2_P2

https://github.com/Jeveloper25/SYSC4001_A2_P3

For each of the simulation tests above, you should analyze the results obtained, and based on the simulation logs, explain what the system is doing. Go through the execution steps of your simulation and explain how the actual system call works and how this is done in your simulation, using the log results to discuss each of the steps in detail.

TEST 1

The test case begins by forking the initial program. Forking the program clones it into a child process, which is an identical PCB with a new PID and memory partition. The simulation gives the child precedence, allowing it to run to completion before returning to the parent. Every line between IF_CHILD and IF_PARENT or ENDIF is executed only by the child, and every line between IF_PARENT and ENDIF is executed only by the parent. All lines after END_IF are executed by both. In test 1, the child process will call EXEC program1, and the parent will call EXEC program2. EXEC replaces the current process with the given program. The PID remains the same, but the memory is reallocated. The new program is then run, resulting in a CPU burst of 100ms. The parent process also performs an EXEC after the child has completed, but for a different program, resulting in a SYSCALL. The simulation then finishes once the parent completes the SYSCALL.

TEST 2

The test case begins by forking the initial program. The child process has a unique instruction to EXEC program 1. Program1 begins by forking itself, creating a child identical to itself. Neither the child nor the parent have unique instructions, both simply perform an EXEC for program2. Prorgam2 only contains a CPU burst. After the child and parent complete, the original parent proceeds and performs the CPU burst at the end of the initial program.

TEST 3

The test case begins by forking the initial program. The child has no unique instructions, only performing a CPU burst before completing. The parent has the unique instruction to EXEC program1. Program1 performs CPU and IO operations before completing, marking the end of the parent process.

TEST 4

The test case begins by forking the initial program. The child process has no unique instructions, simply performing a CPU burst before completing. The parent has a unique instruction to perform an EXEC of program1. Program1 is a simple program performing CPU bursts and IO operations. It also performs another EXEC at the end, replacing the PCB once

again, but with the information of program2. Program2 then executes CPU and I/O operations until completion. Upon completion, the parent does not perform the CPU burst at the end of the initial program because the initial EXEC call swaps the program's PCB.

TEST 5

The test case begins by forking the initial program. The child process has a unique instruction to EXEC program1. Program1 begins by forking itself. The child and parent have no unique instructions, they both only EXEC program2. Program2 performs an initial CPU burst before forking. The child process performs a burst, and both the child and the parent perform CPU and I/O operations before completing. Upon completion, the initial child has finished execution, allowing the initial parent to continue execution and perform a CPU burst before completing.