SYSC4001
Assignment 2

Report

Eric Cui – 101237617

Aydan Eng – 101295820

Part 2 GitHub

Part 3 GitHub

Objective

This assignment is an extension of the work from assignment 1. The objective is to create an extension by implementing fork/exec functions in the simulator, as well as taking account of partition memory limits. The output aim to capture the partition change throughout the execution as well as the CPU usage from the previous assignment.

Analysis

6 Simulations were run with variations in programs instruction, trace file, and external files. From the analyzing the execution and system status files, we found that the partitions with lower sizes are more frequently occupied. This matches with our design goal to use the best fit for every processes. This also showed its usefulness in scenarios where later processes require more partition size than prior processes. If the system uses other algorithms such as first-fit, the prior processes may occupy a larger partition, which may prevent the later processes from getting the partition they need. The execution file is updated as well. With the addition of the partition, the CPU requires additional task of assigning each processes with a partition, leading to a bit more overhead compared to assignment 1.

Discussion

//rest of the trace doesn't really matter (why?)

When a program run an exec() function, the program runs the respected process while immediately ending the current process. In the example code where this question is asked, the process is forked, where each instance of the fork is accepted by one of the if statements. Both of these instances resulting in an exec function, which replace itself with another processes, therefore it is impossible for the PCB to reach the line where the comment is on.

//which process executes this? Why?

The content of program 1 have two if statements before the EXEC function. However, none of the if statements have any exec function to interrupt the process. Therefore, both the parent and the child will execute program 2.

Conclusion

The simulator simulates the operation of partition allocation in a real operating system with the expected change between idle, running, and waiting. It also simulate the interaction between processes and partition allocations.