Nov 7 2025
SYSC 4001

SIMULATION REPORT

Group 34
Esteban Heidrich - 101267959
RIley Foxton - 101304022

The objective of this report is to discuss what the simulator components do and how it is implemented. Before this can be discussed fork() and exec() must be explained as a basic level of understanding.

fork() is a system call that takes the process that calls it and duplicates it exactly, except for the new processes (the child's) PID, which must be unique to the processes running at that time. Both the code and the PCB are copied by the loader from the calling process (called the parent process). If there is not enough space for the child then the fork() fails. In the simulator it is very similar, where whatever the current process is (represented by a structure of attributes like name and size) is duplicated and tries to find space in the memory struct and if there is no space it does not get allocated. If it does then the parent process stops running and waits for the child.

exec() is a system call that takes the process that calls it and deletes its memory image and its PCB entry, in its place the program specified by the exec() is loaded by the loader and tries to allocate memory for its PCB. If there is not enough space then it will not run. In the simulation the effect is the same but recursion is used to run the exec command so logic does not have to be duplicated, then in the recursed function recursion is used again to load the new program trace.

The structure of the simulator is a large loop that interprets the given trace file one line at a time, if the action is “CPU”, “SYSCALL” or “END_IO” then the output is given to the master string to be printed later and the time it takes to execute the actions is given to the master time counter, but no memory or process is represented during this. If the action is “FORK” then the current process is duplicated in a different PCB struct and it is checked to see if it can be allocated, if it can then it becomes the current process, otherwise the fork fails. Still inside the fork section of the code the lines are read to look for indications of what to do if the process is a parent, child or whether execution should end. Once the system finds an “EXEC” command it breaks free and uses recursion to go into the simulate_trace and runs the exec there, once it is done the process gets deleted. If the command is “EXEC” then the program adds some output to the master string and the times associated with it to the master time, then uses recursion to run the smaller programs. Once the exec is done the trace returns back to the trace it executed from. Also during exec the programs are represented as created and allocated in the memory tables.

Trace4.txt, t4_program1.txt, t4_execution.txt and t4_system_status.txt are part of a custom test that is designed to test only the EXEC call. If run successfully there should be 1 table in t4_system_status showing that program1 was allocated in partition 1. The t4_execution output should show outputs relating to the EXEC switch and a CPU call that takes 1000 time units to execute. This is a good test case because we can verify that the switching of programs is done correctly without any interference from others sources of errors.

Trace5.txt, t5_program1.txt, t5_program2.txt, t5_execution.txt and t5_system_status.txt are part of a test designed to test whether the simulator can handle an allocation failure. Program1 is 45 mb, larger than any of the partitions. The simulator attempts to call an exec with it. Afterwards, program2, which can fit, is called an exec with it to confirm that the simulator still runs normally. This is a good test because it verifies that the simulator will continue to run in the case of bad input or that too many programs are being run at once.

The break command at the end of the trace simulator prevents files that have been overridden by exec() to continue after the system call.

Repository Links:
P2 repository - https://github.com/rileyfoxton/SYSC4001_A2_P2
P3 repository - https://github.com/rileyfoxton/SYSC4001_A2_P3

File naming conventions in the part 3 repository are t(trace number)_(file name).txt
Ex: For trace 2 the
execution would be t2_execution.txt
program1 would be t2_program1.txt