SYSC 4001ASSIGNMENT 2

Part 3 Report

STUDENT 1 – DEARELL TOBENNA EZEOKE (101245819)

STUDENT 2 - MOFIOPEFOLUWA OLATUNJI (101237522)

In this part of the assignment, we were tasked with implementing fork and exec to the where we simulated an interrupt system, where the *fork* system call (page 472 of the Linux book), create two independent processes, which run indefinitely. Process 1 will run forever, will initialize a counter at 0, and will increment it in each cycle of an infinite loop. Process 2 will run forever, will initialize a counter at 0, and will increment it in each cycle of an infinite loop. Use delay functions to slow the display speed. To finish the program, use the kill command (man pages), find the PID of both processes (ps) and kill them. The exec function is a function that replaces the child process' memory image and replaces it with a new program, making the child able to run its own program independently while still retaining the same PID as the parent. In this case the parent process would be process 1, which after the fork function occurs, a copy of process 1 which is now process 2 is created. Exec() then allows process 2 to run a similar program that decrements the value of the counter while program 1 increments.

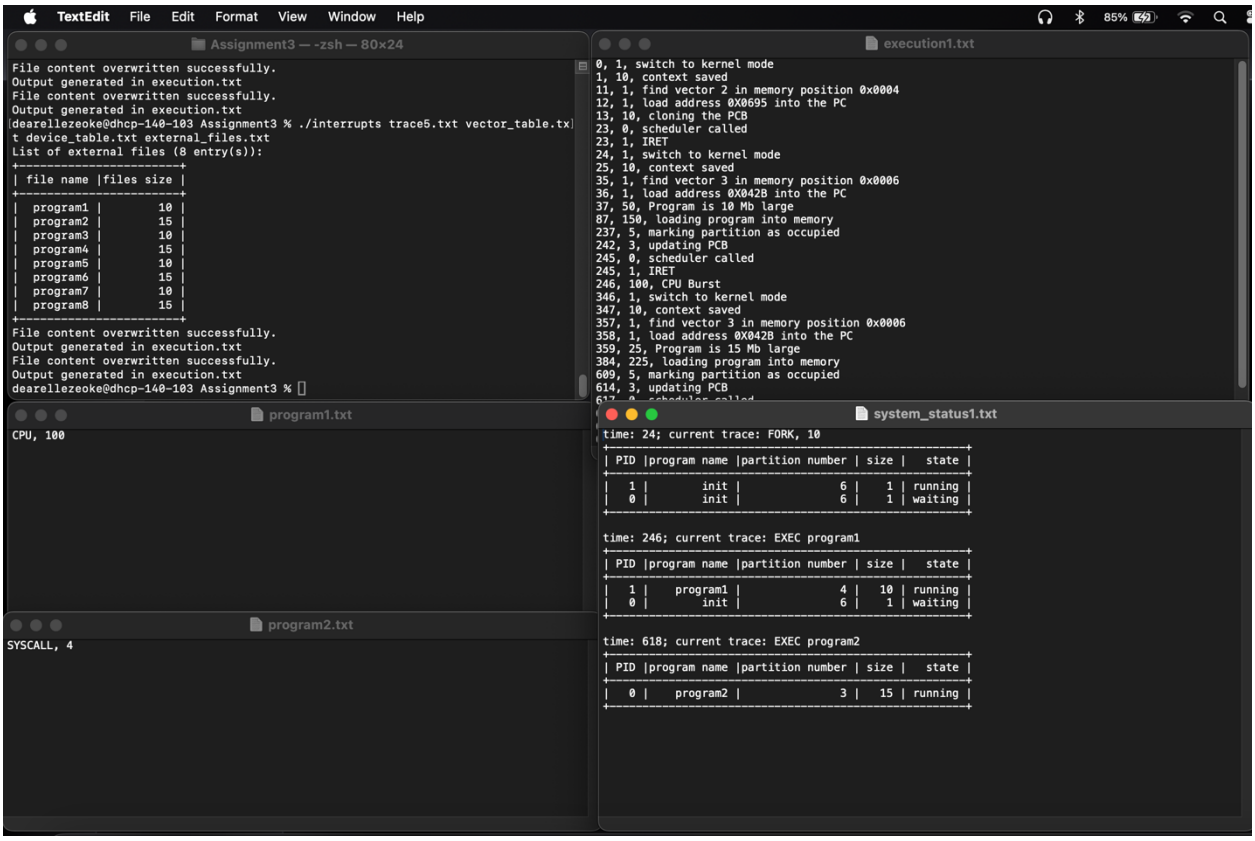
A system call is the privileged instruction which is allowed to access the OS in kernel mode. In our simulation, whenever the SYSCALL line is parsed, the system has to first switch the mode from user mode to kernel mode. After the switch has been made, the context is saved and the CPU looks for through the vector for the address. It then loads the address into the PC. It then loads the address. Fork then clones the program's PCB and calls the scheduler and Exec runs the child's until the program is done. This pauses the parents process. When the child is done, the parent then completes its program and then the mode is switched back to the user mode.

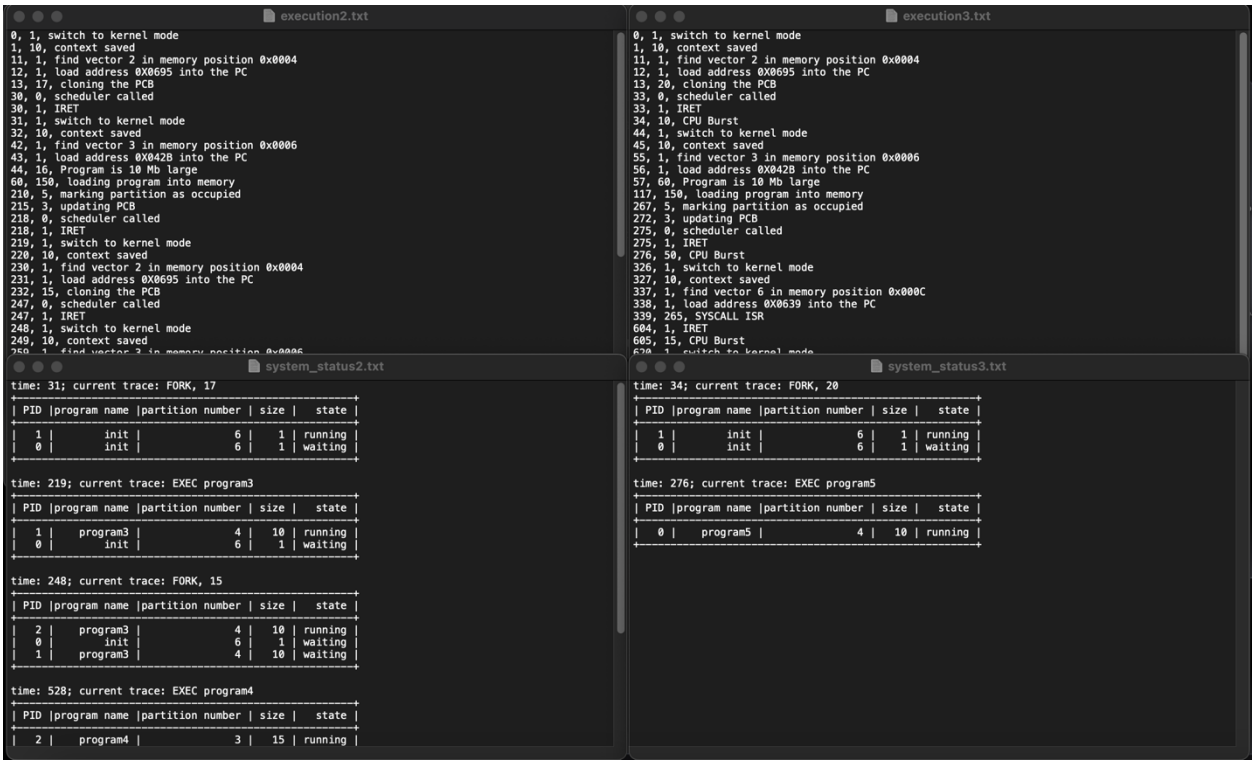
```
FORK, 15
IF_CHILD, 0
IF_PARENT, 0
ENDIF, 0 //notice how nothing is happening in the conditionals
EXEC program2, 33 //which process executes this? Why?
```

This process is executed by the parent because both processes are empty meaning that the that the child does nothing and runs immediately while the parent continues the process as normal.

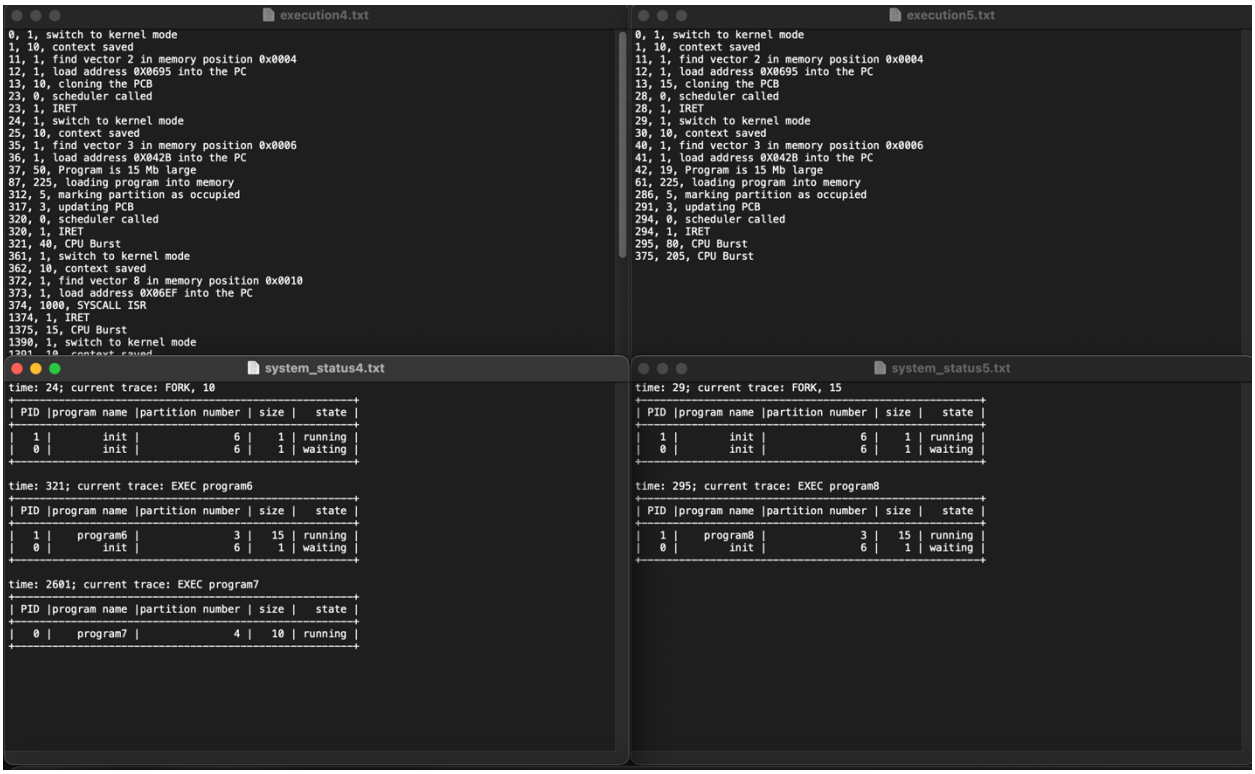
```
break  //Why is this important? (answer in report)
```

This is important because it is what allows the new process to continue running after the old process is being paused.





These are the execution and system status files from the assignment test cases.



These are our execution and system status files from the test cases we created.