

SYSC 4001ASSIGNMENT 2

Part 3 Report

STUDENT 1 – DEARELL TOBENNA
EZEOKI (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.

TextEditFileEditFormatViewWindowHelp

Assignment3 - zsh - 80x24

File content overwritten successfully.
Output generated in execution.txt
File content overwritten successfully.
Output generated in execution.txt
dearellezeoke@dhcp-140-103 Assignment3 % ./interrupts trace5.txt vector_table.tx
t device_table.txt external_files.txt
List of external files (8 entry(s)):

file name	files size
program1	10
program2	15
program3	10
program4	15
program5	10
program6	15
program7	10
program8	15

File content overwritten successfully.
Output generated in execution.txt
File content overwritten successfully.
Output generated in execution.txt
dearellezeoke@dhcp-140-103 Assignment3 %

program1.txt

CPU, 100

program2.txt

SYSCALL, 4

execution1.txt

0, 1, switch to kernel mode
1, 10, context saved
11, 1, find vector 2 in memory position 0x0004
12, 1, load address 0x0095 into the PC
13, 10, cloning the PCB
23, 0, scheduler called
23, 1, IRET
24, 1, switch to kernel mode
25, 10, context saved
35, 1, find vector 3 in memory position 0x0006
36, 1, load address 0x042B into the PC
37, 50, Program is 10 Mb large
87, 150, loading program into memory
237, 5, marking partition as occupied
242, 3, updating PCB
245, 0, scheduler called
245, 1, IRET
246, 100, CPU Burst
346, 1, switch to kernel mode
347, 10, context saved
357, 1, find vector 3 in memory position 0x0006
358, 1, load address 0x042B into the PC
359, 25, Program is 15 Mb large
384, 225, loading program into memory
609, 5, marking partition as occupied
614, 3, updating PCB
617, 0, scheduler called

system_status1.txt

time: 24; current trace: FORK, 10

PID	program name	partition number	size	state
1	init	6	1	running
0	init	6	1	waiting

time: 246; current trace: EXEC program1

PID	program name	partition number	size	state
1	program1	4	10	running
0	init	6	1	waiting

time: 618; current trace: EXEC program2

PID	program name	partition number	size	state
0	program2	3	15	running