

## SYSC 4001 ASSIGNMENT 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.

TextEdit File Edit Format View Window Help

Assignment3 -- zsh -- 80x24

```

File content overwritten successfully.
Output generated in execution.txt
File content overwritten successfully.
Output generated in execution.txt
|dearellezeo@dhcp-146-103 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
|dearellezeo@dhcp-146-103 Assignment3 %

```

program1.txt

CPU, 100

```

time: 24; current trace: FORK, 10
+-----+
| PID |program name |partition number | size | state |
+-----+
| 1 |      init |             6 |   1 | running |
| 0 |      init |             6 |   1 | waiting |
+-----+

```

program2.txt

SYSCALL, 4

```

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 |
+-----+

```

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 0x0695 into the PC  
13, 17, cloning the PCB  
30, 0, scheduler called  
30, 1, IRET  
31, 10, switch to kernel mode  
32, 10, context saved  
42, 1, find vector 3 in memory position 0x0006  
43, 1, load address 0x042B into the PC  
44, 16, Program is 10 Mb large  
60, 150, loading program into memory  
210, 5, marking partition as occupied  
235, 3, updating PCB  
238, 0, scheduler called  
238, 1, IRET  
219, 1, switch to kernel mode  
220, 10, context saved  
230, 1, find vector 2 in memory position 0x0004  
231, 1, load address 0x0695 into the PC  
232, 15, cloning the PCB  
240, 0, scheduler called  
247, 1, IRET  
248, 1, switch to kernel mode  
249, 10, context saved  
250, 1, find vector 3 in memory position 0x0006

system\_status1.txt

0, 1, switch to kernel mode  
1, 10, context saved  
11, 1, find vector 3 in memory position 0x0006  
35, 1, load address 0x042B into the PC  
36, 1, Program is 10 Mb large  
37, 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  
385, 5, marking partition as occupied  
314, 3, updating PCB

program3.txt

time: 24; current trace: FORK, 10

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

time: 246; current trace: EXEC program1

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

time: 618; current trace: EXEC program2

PID	program name	partition number	size	state
0	program2		3	15

execution2.txt

0, 1, switch to kernel mode  
1, 10, context saved  
11, 1, find vector 2 in memory position 0x0004  
12, 1, load address 0x0695 into the PC  
13, 20, cloning the PCB  
33, 0, scheduler called  
33, 1, IRET  
34, 100, CPU Burst  
44, 1, switch to kernel mode  
45, 10, context saved  
55, 1, find vector 3 in memory position 0x0006  
56, 1, load address 0x042B into the PC  
57, 60, Program is 10 Mb large  
117, 150, loading program into memory  
267, 5, marking partition as occupied  
272, 3, updating PCB  
275, 0, scheduler called  
275, 1, IRET  
276, 50, CPU Burst  
326, 1, switch to kernel mode  
327, 10, context saved  
337, 1, find vector 6 in memory position 0x000C  
338, 1, load address 0x0639 into the PC  
339, 265, SYSCALL ISR  
604, 1, IRET  
605, 15, CPU Burst  
620, 1, switch to kernel mode

system\_status2.txt

time: 31; current trace: FORK, 17

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

time: 219; current trace: EXEC program3

PID	program name	partition number	size	state
1	program3		4	10
0	init		6	1

time: 248; current trace: FORK, 15

PID	program name	partition number	size	state
2	program3		4	10
0	init		6	1
1	program3		4	10

time: 528; current trace: EXEC program4

PID	program name	partition number	size	state
2	program4		3	15

execution3.txt

time: 34; current trace: FORK, 20

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

time: 276; current trace: EXEC program5

PID	program name	partition number	size	state
0	program5		4	10

system\_status3.txt

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

The screenshot shows four terminal windows side-by-side. The first two windows are labeled `execution4.txt` and `execution5.txt`, each containing a series of numerical log entries. The last two windows are labeled `system_status4.txt` and `system_status5.txt`, each displaying a table of system processes.

**Execution Log (execution4.txt):**

```

8, 1, switch to kernel mode
1, 10, context saved
11, 1, find vector 2 in memory position 0x0004
12, 1, load address 0XB695 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 0XA42B into the PC
37, 50, Program is 15 Mb large
87, 225, loading program into memory
312, 5, marking partition as occupied
317, 0, updating PCB
320, 0, scheduler called
320, 1, IRET
321, 40, CPU Burst
361, 1, switch to kernel mode
362, 10, context saved
372, 1, find vector 8 in memory position 0x0010
373, 1, load address 0X06EF into the PC
374, 1000, INDCALL ISR
1374, 1, IRET
1375, 15, CPU Burst
1390, 1, switch to kernel mode
1201, 10, context saved

```

**Execution Log (execution5.txt):**

```

8, 1, switch to kernel mode
1, 10, context saved
11, 1, find vector 2 in memory position 0x0004
12, 1, load address 0XB695 into the PC
13, 15, cloning the PCB
28, 0, scheduler called
28, 1, IRET
29, 1, switch to kernel mode
30, 10, context saved
40, 1, find vector 3 in memory position 0x0006
41, 1, load address 0XA42B into the PC
42, 19, Program is 15 Mb large
61, 225, loading program into memory
286, 5, marking partition as occupied
291, 0, updating PCB
294, 0, scheduler called
294, 1, IRET
295, 80, CPU Burst
375, 205, CPU Burst

```

**System Status (system\_status4.txt):**

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

**System Status (system\_status5.txt):**

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

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