HOMEWORK 3 REPORT

FATIH SELIM YAKAR

CSE312 OPERATING SYSTEMS

161044054

Process Table in Kernel (Assembly source file)

I created a process table on assembly, similar to the process table I did in the previous assignment. According to the content of my previous process table, I allocated space aligned to the assembly as much as I measured by measuring how many bytes of process control block occupies. (I measured using the sizeof (process_control_block) function)

As a result of this measurement, I found that a process control block occupies 200 bytes. Then, for each kernel, how many process control blocks are required, I allocated that much space as follows.

```
#Process table includes 5 process control block
pcb0: .align 2
      .space 200
pcb1: .align 2
      .space 200
pcb2: .align 2
      .space 200
pcb3: .align 2
      .space 200
pcb4: .align 2
      .space 200
```

Figure 1-Process table for kernel1

Global Variables and Flags in Kernel (Assembly source file)

To achieve the optimization, I also used nine variables to provide the link between .cpp and .asm / .s. These variables and their intended use are:

running_process: Holds the current running process's index in the process table.
running_process_count: Holds the total number of running process in current state.
all_process_count: Holds the total number of created and deleted process in current states.
handler_flag: It prevents context switching in unsuitable situations.
waitpid_flag: If this flag is 1, the context switch will be between the same processes and so on until it ends.
waitpid_process: Shows the index of the process that waitpid will be applied.
waitpid_end_addres: Indicates the end point of the process in text segment to be performed by the waitpid syscall.
process_table_size: Holds the total number of process control block. So size of process table.

critical_region_flag: When this flag is active, no interrupt is entered, that is, the context switch does not occur.

```
running_process: .word 4
running_process_count: .word 4
all_process_count: .word 4
handler_flag: .word 4
waitpid_flag: .word 4
waitpid_process: .word 4
waitpid_end_addres: .word 4
process_table_size: .word 4
critical_region_flag: .word 4
```

Figure 2-flag and global variables in kernels

Syscalls in the syscall.cpp

- **Init** : Initialize the process table (in the assembly file) with "0" pid and main process.
- **Fork**: Copies the current process and then creates the new process in process table (in the assembly file) with these.
- **Execve** : Loads a new executable file and then changes the created process informations to new loaded process.
- **Waitpid** : Waits for the given process to finish.
- process_exit : Controls the process if there is no process in process table, then system finishes.
- **Delete_process** : Deletes the current running process and if there are any process in process table, then changes the process else system finishes.
- **Rng** : Creates the new integer random number bounded by given integer number.
- **Save_the_state** : Saves the machines current state to ProcessTable[running_process].
- **Round_robin_scheduling** : Finds the another not empty place in ProcessTable for change.
- **Change_the_process** : Updates the current registers and machine informations by ProcessTable[$a0] process

Functions in syscall.cpp

- **void write_string_addres(char string[],int start_address):** Writes string in the given data memory address usind set_mem_byte().
- **void print_pcb_infos(int pcb_num):** Prints out the given pcb_num's process informations.
- **void register_to_memory(int pcb_num):** Saves the current registers to process table in the data memory.

- **void memory_to_register(int pcb_num):** Saves the process table in the memory to current registers.
- **bool is_finish():** Controls is finished according to running process count.
- **void delete_process():** Deletes the current running process.
- **void exchange_func():** Changes the current process to new process.

Functions in Kernels (Assembly source files)

- **context_switch:** Saves the current state in process table and does round robin scheduling, at the end changes the process and updates data.
- **set_struct:** Main process control block data setter function in the kernel.
- **get_struct:** Main process control block data getter function in the kernel.
- For each process control block data, there are 12 other functions that use the above set and get main functions (such as set_process_state, get_program_counter).

General working mechanism of kernels

Firstly, pid 0 is created with init syscall and this is the main process (after Init syscall, I make waitpid (0) that the main process runs first, create new processes and finish, then others run in parallel). Then, if you want to create a new process, a copy of the init process is created using fork syscall and it becomes process number 1 as pid. If execve is used after the fork and the path of the new program is given as a parameter, the data of the last process created with the fork is deleted and the data of the new incoming process is saved to the process control block number pid 1. In other words, after init syscall, fork + execve is performed and a new process is created. In this way, if several processes occur and interrupt occurs until init ends, the context switch passes from the init process to the init process, that is, it does not change the process. After the Init process is finished, it switches to other processes in the process table using the round robin scheduling, and deletes itself from the process table by calling the delete_process syscall whenever the process ends, thereby eliminating the zombie process. When all processes are finished, process_exit finishes the entire program by running syscall. All the operations here perform their operations by making operations on the global variable, flag and process table in the assembly source code.

In case of interrupt, the process state sets READY and in case of I/O syscalls, the process state sets BLOCKED. Other than these the process states sets RUNNING

Commands and Notes

- There can be some warnings in the execution because of the program's timing. Although I minimized ,by using flags, the parts that are not optimized, it warns that the places where the interrupt will enter cannot be predicted and enters in critical places. If it warns, try running it again.

- Init can sometimes exit its own addres space immediately after the end of the process, but it does not affect the work.

- Sometimes processes can run more or less than desired.
- Kernel 3 can print errors at the end. In such cases, try running it again.
- While the palindrome code is working exactly outside the kernel, it can sometimes pass without getting the input from the user in the kernel section or finish all the words without printing (kernel2,3)

Sample Running Screenshots

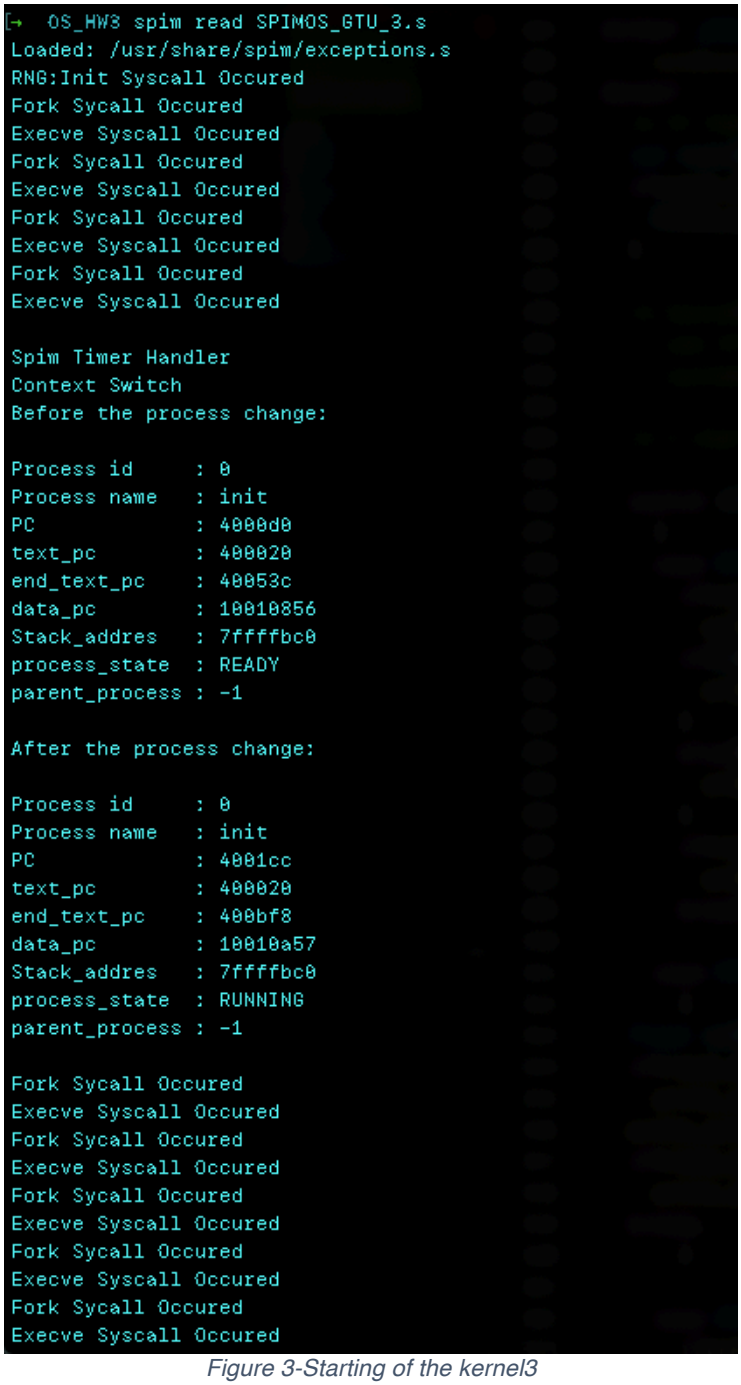

Figure 3-Starting of the kernel3

```
Binary searching array is:10 20 22 50 51 60
Spim Timer Handler
Context Switch
Before the process change:

Process id     : 6
Process name   : BinarySearch.asm
PC             : 400da4
text_pc        : 400da4
end_text_pc    : 400f50
data_pc        : 10010b57
Stack_addres   : 7ffffbc0
process_state  : READY
parent_process : 0

After the process change:

Process id     : 7
Process name   : Collatz.asm
PC             : 400f50
text_pc        : 400f50
end_text_pc    : 401030
data_pc        : 10010b76
Stack_addres   : 7ffffbc0
process_state  : RUNNING
parent_process : 0

1:4 2 1
2:1
3:10 5 16 8 4 2 1
4:2 1
5:16 8 4 2 1
6:3 10 5 16 8 4 2 1
7:22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
8:4 2 1
9:28 14 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
10:5 16 8 4 2 1
11:34 17 52 26 13 40 20 10 5 16 8
```

Figure 4-Context switching BinarySearch to Collatz in kernel3

```
40 20 10 5 16 8 4 2 1
23:70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
24:12 6 3 10 5 16 8 4 2 1
25:76 38 19 58 29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Collatz finished.
Process 9 deleted. After the delete process count: 1
Delete Process Syscall Occured

Spim Timer Handler
Context Switch
Before the process change:

Process id     : 9
Process name   : Collatz.asm
PC             : 0
text_pc        : 401110
end_text_pc    : 4011f0
data_pc        : 10010bb6
Stack_addres   : 7ffffbc0
process_state  : READY
parent_process : 0

After the process change:

Process id     : 8
Process name   : Collatz.asm
PC             : 4010b8
text_pc        : 401030
end_text_pc    : 401110
data_pc        : 10010b96
Stack_addres   : 7ffffbc0
process_state  : RUNNING
parent_process : 0

20 10 5 16 8 4 2 1

Collatz finished.
Process 8 deleted. After the delete process count: 0
Delete Process Syscall Occured
Program Has Been Finished
➜  OS_HW3
```

Figure 5- End of the kernel3

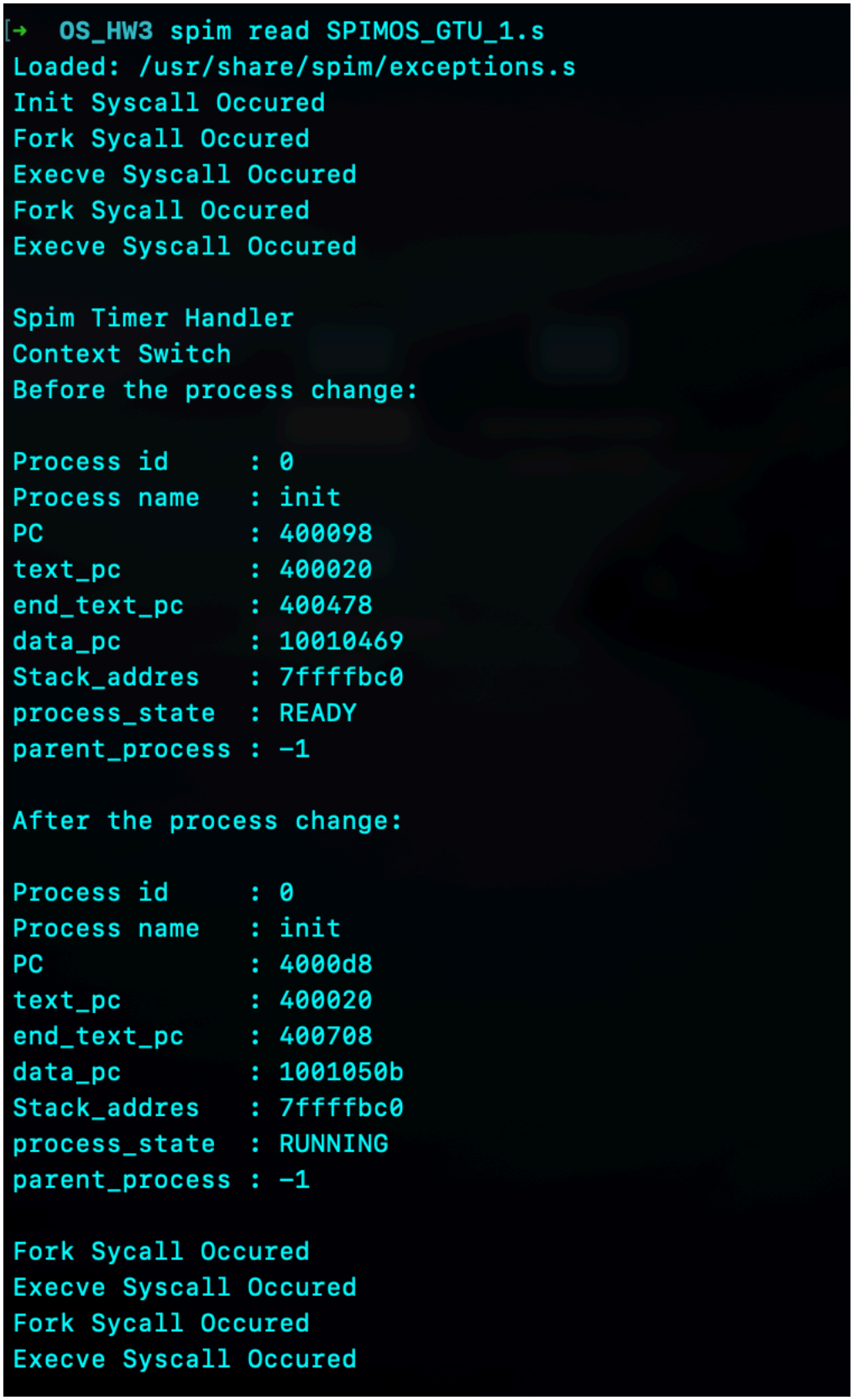

Figure 6-Starting of the kernel1

```
8 4 2 1
11:34 17 52 26 13 40 20 10 5 16 8 4 2 1
12:6 3 10 5 16 8 4 2 1
13:40 20 10 5 16 8 4 2 1
14:7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
15:46 23 70 35 106 53 160 80 40 20 10 5
Spim Timer Handler
Context Switch
Before the process change:

Process id     : 1
Process name   : Collatz.asm
PC             : 400500
text_pc        : 400478
end_text_pc    : 400558
data_pc        : 1001048a
Stack_addres   : 7ffffbc0
process_state  : READY
parent_process : 0

After the process change:

Process id     : 4
Process name   : palindrome.asm
PC             : 4009e4
text_pc        : 4008b4
end_text_pc    : 400af0
data_pc        : 10010943
Stack_addres   : 7ffffba0
process_state  : RUNNING
parent_process : 0

Not Palindrome
2: just: Not Palindrome
3: able: Not Palindrome
4: about: Not Palindrome
5: above: Not Palindrome
6: accept: Not Palindrome
7: according: Not Palindrome
8: account: Not Palindrome
```

Figure 7-Context switch in kernel1

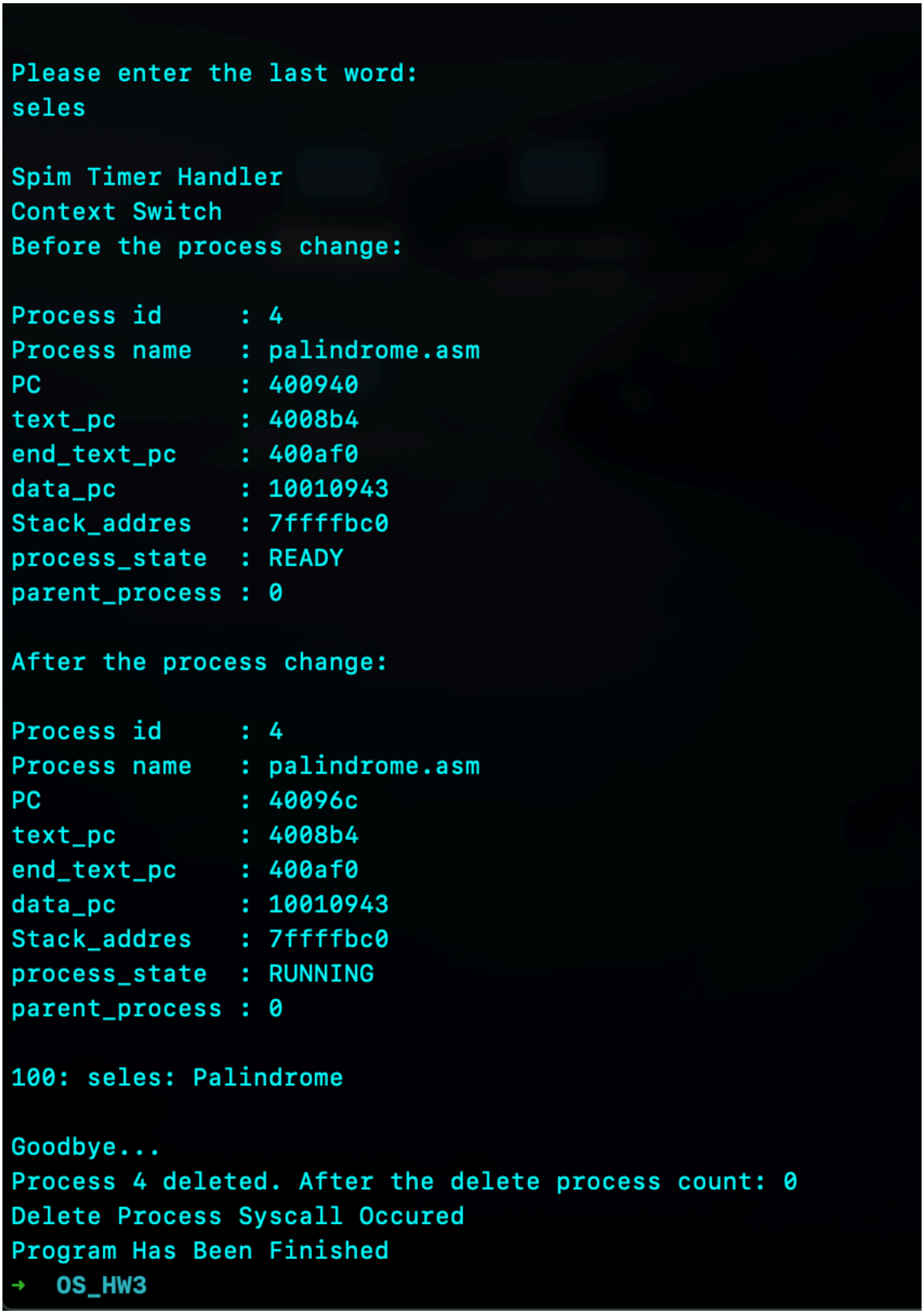

Figure 8-Finish of the kernel1

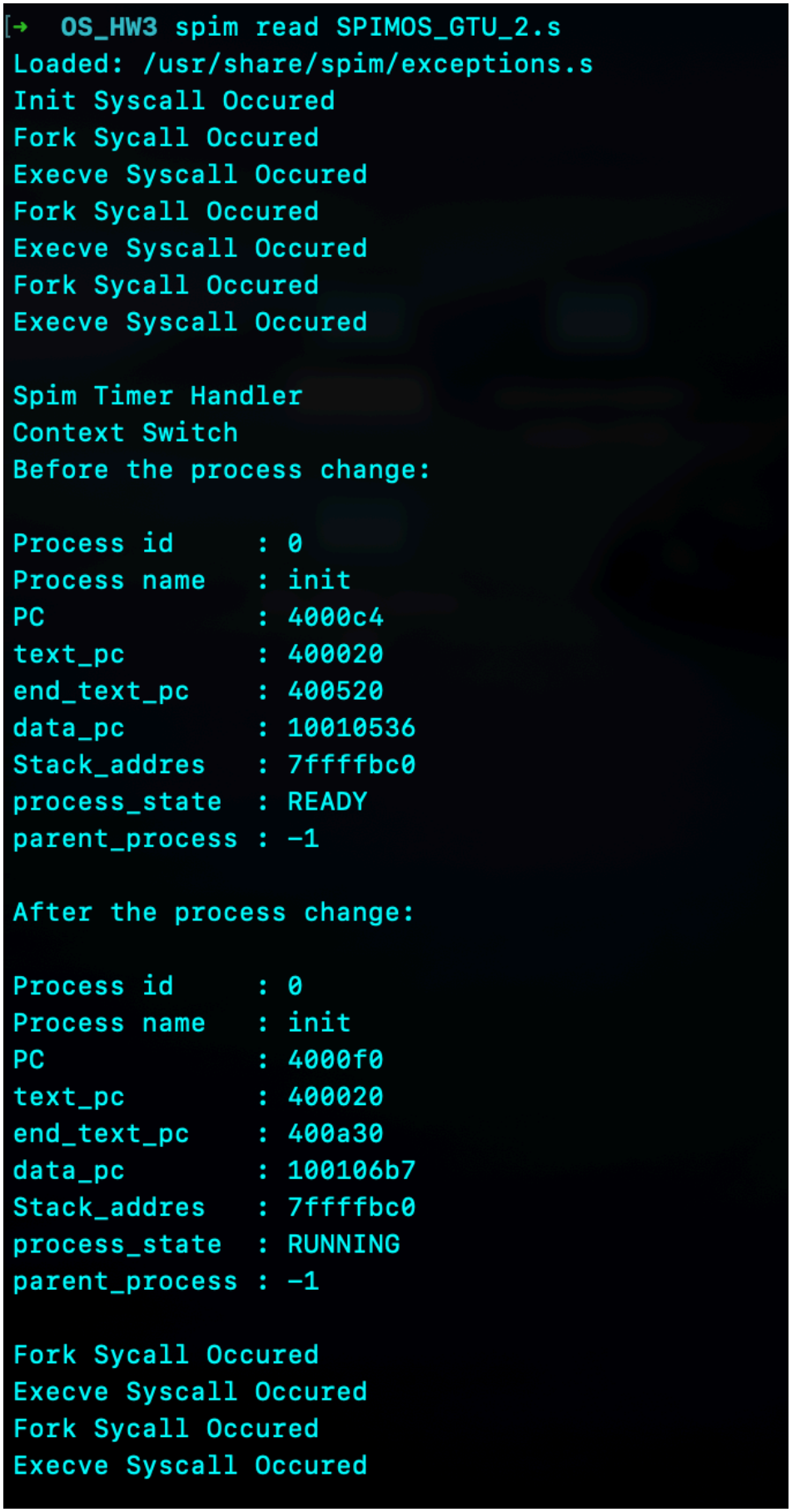

Figure 9-Starting of the kernel2

```
Linear searching array is:60 20 22 75 50 51 10
Searching value is:0
Output:-1
Linear search finished.
Process 2 deleted. After the delete process count: 2
Delete Process Syscall Occured

Spim Timer Handler
Context Switch
Before the process change:

Process id     : 2
Process name   : LinearSearch.asm
PC             : 0
text_pc        : 4006d0
end_text_pc    : 400880
data_pc        : 10010637
Stack_addres   : 7ffffbc0
process_state  : READY
parent_process : 0

After the process change:

Process id     : 3
Process name   : LinearSearch.asm
PC             : 400880
text_pc        : 400880
end_text_pc    : 400a30
data_pc        : 100106b7
Stack_addres   : 7ffffbc0
process_state  : RUNNING
parent_process : 0


Linear searching array is:60 20 22 75 50 51 10
Searching value is:0
Output:-1
Linear search finished.
Process 3 deleted. After the delete process count: 1
Delete Process Syscall Occured
```

Figure 10-Context switch in kernel2

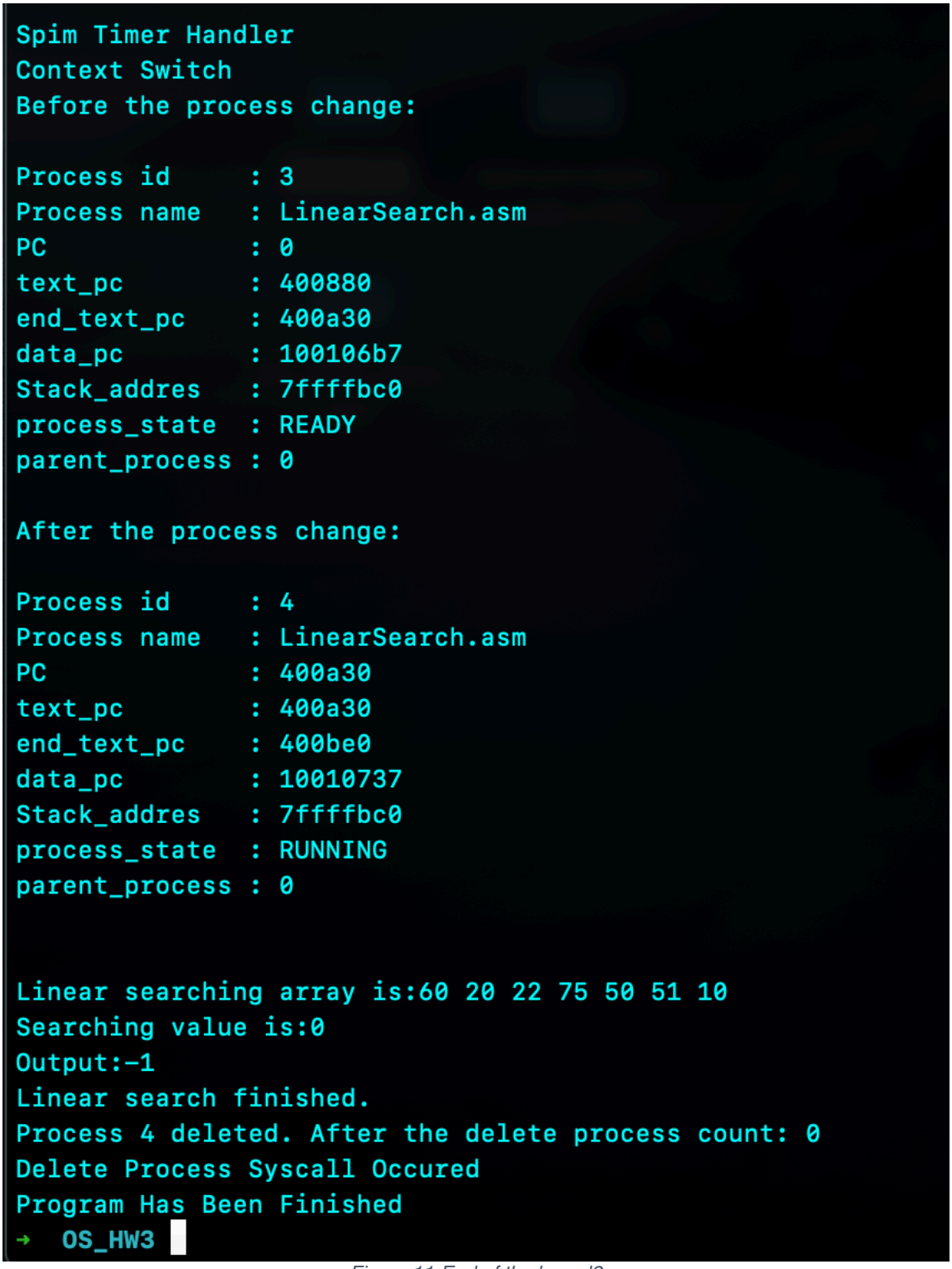

Figure 11-End of the kernel2