**CSE 321 Operating Systems HW #2 Read Me**
**Gökçe Nur Erer**
**171044079**

This read me will include 3 parts. First part will be explaining how the process table and interrupt handling is implemented. The second part will be explaining which system calls are implemented and how they work. The third part will be explaining the microkernels.

## Part 1 : Process Table and Interrupt Handling

Process table is implemented as an vector in syscall.cpp file. This vector is a collection of process table entry structs defined in the same file. A process table entry consists:
1) Process pid
2) Process name
3) Process state : This state is defined with an enum. A process can be ready, running or finished.
4) Process memory items: These items are collected and defined in a struct called mem_backup which is also implemented in syscall.cpp file. These items are the memory items that SPIM requires to run a specific process.

Process table entries use 2 functions to update or backup their memory structure inside. One of the functions is called “set_memory_backup” which reads the current memory items into process' memory structure. Second function is called “get_memory_backup” which reads the memory of the process and loads it to SPIM so the process can run.

**Interrupt Handling:**
When a timer interrupt occurs program first prints the current situation of the process table before switching. After printing, the current running process' memory is saved to process table entry it belongs and it is state is changed with ready if it is not finished. The first ready process index is found in the process table vector. The newly found process' memory contents are loaded to SPIM and the new process' state has been changed to running.

PS: Finished process' stay in the process table yet they are not scheduled to run again.

## Part 2 : System Calls

System calls are implemented in the syscall.cpp file. The following syscalls are implemented for the purpose of this homework:

**1) Fork :** This syscall is used to create new processes. A process table entry is created for the new process with the same name of the process that created it. The current process' memory is copied to the new process and new process' program counter is increased by one instruction. This syscall returns 0 to the new process created. And the pid of the new process is returned to the process that created the new process. Then the new process is added to the process table as an entry.

**2) Execve :** This syscall takes a program name as a parameter. The current memory is resetted and emptied first. Then by using initialize_world and initialize_run_stack functions of the SPIM the SPIM enviroment is resetted. After that the corresponding assembly file to the name parameter is loaded to spim by using read_assembly_file function of SPIM. The program counter is set to the default run

location of SPIM which is the __start label which is added when initialize_world function runs and adds exception file of SPIM. After all this steps the current loaded memory items are saved to the process table entry corresponding to the process that called this syscall.

**3) Wait_pid :** This syscall gets a parameter which is the pid of the process which the caller will wait. This syscall pulls program counter by one instruction so the program stalls until the requirement is satisfied.

4**) Process exit :** This syscall sets the caller process' state “finished”. And it pulls program counter by one instruction so the program stalls and waits for the scheduler to switch it. After the switch the finished process never runs again.

**5) Create init process :** This syscall is called only once in an assembly file which registers the file as the init process, who has the pid of 0. Creates a new process table entry and adds it to the process table.

**6) Rand_Int :** This syscall generates a random number between 0 to 2.

**7) Set_Rand_Int_Seed :** This syscall sets the seed to the rand function which is used in rand_int syscall. This syscall should be called once in an assembly file.

**PS:** The reason of this syscall is when seed is defined in the rand_int syscall, the situation where 2 different random number has to be created is not satisfied. The rand function is called within milliseconds so the seed cannot produce different numbers.

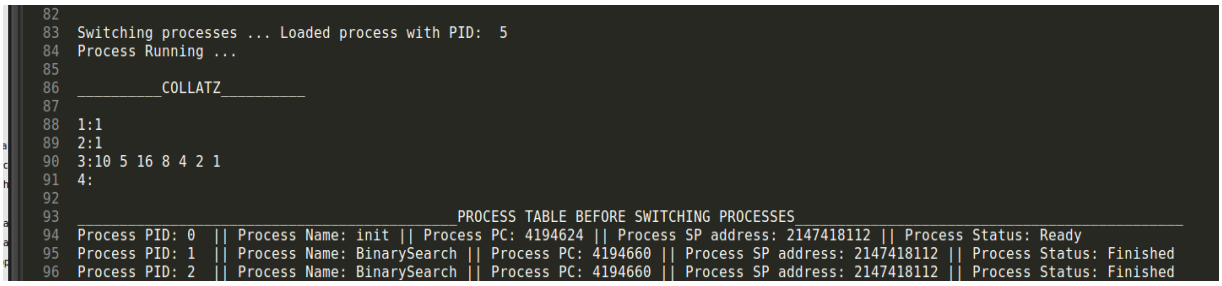
## Part 3 : Microkernels

**SPIMOS_GTU_1:** First, microkernel registers itself as the init process, then 3 fork syscall is called. Each child process created by these fork syscalls calls an execve with a different program name. First one calls LinearSearch, second one calls BinarySearch, third one calls Collatz. And the init process who is the parent to all these child processes wait for their pids until all the child processes end, then exits.

**SPIMOS_GTU_2:** First, microkernel registers itself as the init process then a random number is generated. Depending on the random number the program name is selected and 10 fork syscalls are made to create 10 child processes. Each child process' pid is saved into an array. After that 10 child processes use execve to change their images to the selected program. The init process who is the parent to all these child processes wait for the 10 different process pid's saved in to an array, then exits when they all finish.

**SPIMOS_GTU_3:** First, microkernel registers itself as the init process then a random number is generated. Depending on the random number the program name is selected and 3 fork syscalls are made to create 3 child processes. Each child process' pid is saved into an array. With the selected name execve calls are made by the 3 child process so they can change their image to the program that is selected. After that another random number is generated. And all these steps are repeated again for the newly selected program name. Then the init process who is the parent to all these child process waits for these 6 processes whose pid's are saved in to an array and then exits when they all finish.

Notes:
1)Timer interrupt can happen when Collatz is running so the outputs like the following is the cause of that.

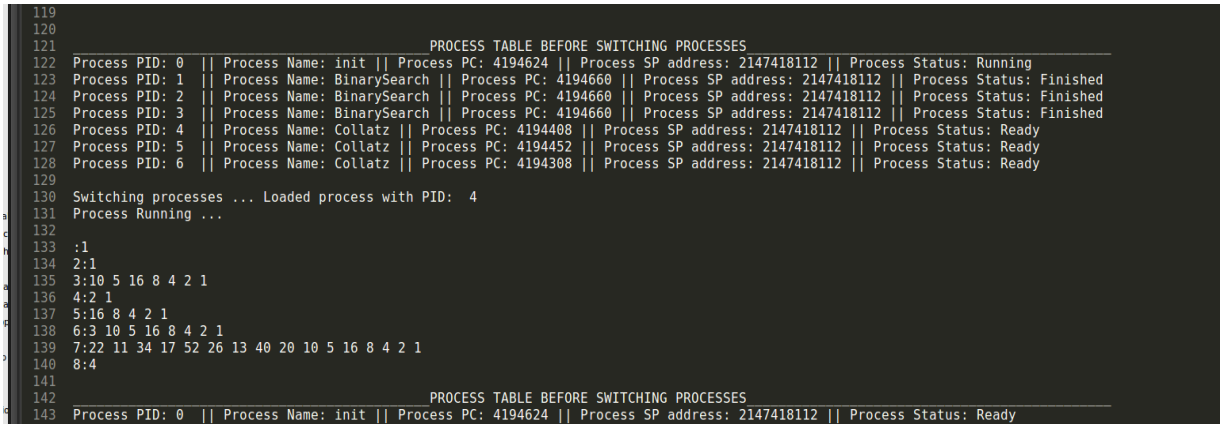


```
Switching processes ... Loaded process with PID:  5
Process Running ...

_________COLLATZ_________

1:1
2:1
3:10 5 16 8 4 2 1
4:

_____________________________________________PROCESS TABLE BEFORE SWITCHING PROCESSES_____________________________________________
Process PID: 0  || Process Name: init || Process PC: 4194624 || Process SP address: 2147418112 || Process Status: Ready
Process PID: 1  || Process Name: BinarySearch || Process PC: 4194660 || Process SP address: 2147418112 || Process Status: Finished
Process PID: 2  || Process Name: BinarySearch || Process PC: 4194660 || Process SP address: 2147418112 || Process Status: Finished
```



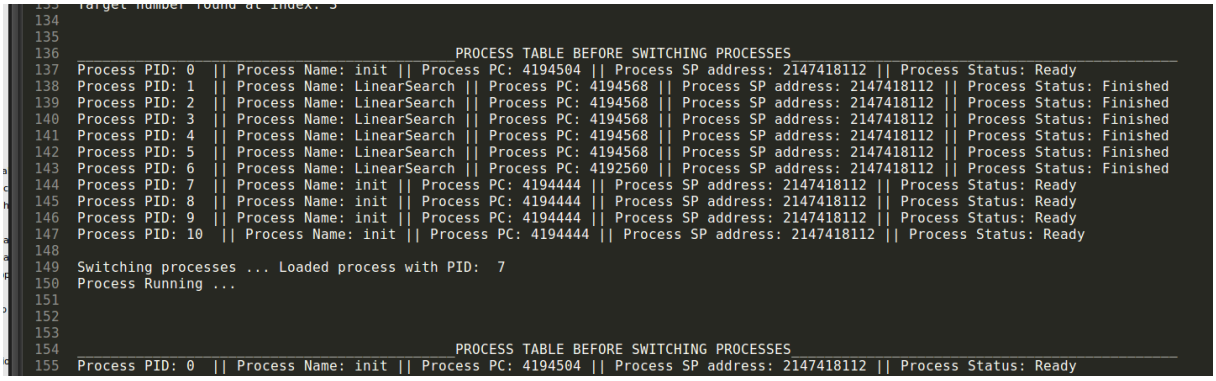
```
_____________________________________________PROCESS TABLE BEFORE SWITCHING PROCESSES_____________________________________________
Process PID: 0  || Process Name: init || Process PC: 4194624 || Process SP address: 2147418112 || Process Status: Running
Process PID: 1  || Process Name: BinarySearch || Process PC: 4194660 || Process SP address: 2147418112 || Process Status: Finished
Process PID: 2  || Process Name: BinarySearch || Process PC: 4194660 || Process SP address: 2147418112 || Process Status: Finished
Process PID: 3  || Process Name: BinarySearch || Process PC: 4194660 || Process SP address: 2147418112 || Process Status: Finished
Process PID: 4  || Process Name: Collatz || Process PC: 4194408 || Process SP address: 2147418112 || Process Status: Ready
Process PID: 5  || Process Name: Collatz || Process PC: 4194452 || Process SP address: 2147418112 || Process Status: Ready
Process PID: 6  || Process Name: Collatz || Process PC: 4194308 || Process SP address: 2147418112 || Process Status: Ready

Switching processes ... Loaded process with PID:  4
Process Running ...

: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

_____________________________________________PROCESS TABLE BEFORE SWITCHING PROCESSES_____________________________________________
Process PID: 0  || Process Name: init || Process PC: 4194624 || Process SP address: 2147418112 || Process Status: Ready
```

2) Timer interrupt can happen before a process finish execve syscall but couldn't run itself. So no output is shown. It is not a problem.



```
Target number found at index: 3


_____________________________________________PROCESS TABLE BEFORE SWITCHING PROCESSES_____________________________________________
Process PID: 0  || Process Name: init || Process PC: 4194504 || Process SP address: 2147418112 || Process Status: Ready
Process PID: 1  || Process Name: LinearSearch || Process PC: 4194568 || Process SP address: 2147418112 || Process Status: Finished
Process PID: 2  || Process Name: LinearSearch || Process PC: 4194568 || Process SP address: 2147418112 || Process Status: Finished
Process PID: 3  || Process Name: LinearSearch || Process PC: 4194568 || Process SP address: 2147418112 || Process Status: Finished
Process PID: 4  || Process Name: LinearSearch || Process PC: 4194568 || Process SP address: 2147418112 || Process Status: Finished
Process PID: 5  || Process Name: LinearSearch || Process PC: 4194568 || Process SP address: 2147418112 || Process Status: Finished
Process PID: 6  || Process Name: LinearSearch || Process PC: 4192560 || Process SP address: 2147418112 || Process Status: Finished
Process PID: 7  || Process Name: init || Process PC: 4194444 || Process SP address: 2147418112 || Process Status: Ready
Process PID: 8  || Process Name: init || Process PC: 4194444 || Process SP address: 2147418112 || Process Status: Ready
Process PID: 9  || Process Name: init || Process PC: 4194444 || Process SP address: 2147418112 || Process Status: Ready
Process PID: 10  || Process Name: init || Process PC: 4194444 || Process SP address: 2147418112 || Process Status: Ready

Switching processes ... Loaded process with PID:  7
Process Running ...



_____________________________________________PROCESS TABLE BEFORE SWITCHING PROCESSES_____________________________________________
Process PID: 0  || Process Name: init || Process PC: 4194504 || Process SP address: 2147418112 || Process Status: Ready
```