

OS Project#5
Due date: 11/07/2018

In this project you are asked to implement a very basic operating systems. The most critical components for an operating systems is the scheduler and the shell. The shell gives us the ability to launch, execute and stop applications at any time. In other words, a shell can be considered as the interface for the user.

On Moodle you can find the code for a simple shell (shell.c) and the code for the programs that this shell can execute (p-shell.c). The commands that are supported by this shell are:

- ver: Shows details about the shell version
- help: Shows the help
- ps: Shows the living process with the given pid
- kill(pid): Ends the process with the given pid
- exec p1(n1,qt1) p2(n2,qt2) ...: Executes the programs p1, p2 ... Each program types a message for n times and it is given a time quantum of qt msec. If parameter (&) is given the program will be executed in the background
- exit: Exits the program

Below you can find some sample output of our basic OS:

```
$ gcc p-shell.c -o p
$ gcc shell.c -o shell
$ ./shell
```

```
New Shell. Works properly!!
=>exec p(5,100)
```

```
This is program p and it prints for the 1 time of 5...
This is program p and it prints for the 2 time of 5...
This is program p and it prints for the 3 time of 5...
This is program p and it prints for the 4 time of 5...
This is program p and it prints for the 5 time of 5...
The child 9734 is dead
```

```
=>exec p(5,10) p(9,100)
```

```
This is program p and it prints for the 1 time of 5...
```

This is program p and it prints for the 2 time of 5...
This is program p and it prints for the 3 time of 5...
This is program p and it prints for the 4 time of 5...
This is program p and it prints for the 5 time of 5...
The child 9737 is dead

This is program p and it prints for the 1 time of 9...
This is program p and it prints for the 2 time of 9...
This is program p and it prints for the 3 time of 9...
This is program p and it prints for the 4 time of 9...
This is program p and it prints for the 5 time of 9...
This is program p and it prints for the 6 time of 9...
This is program p and it prints for the 7 time of 9...
This is program p and it prints for the 8 time of 9...
This is program p and it prints for the 9 time of 9...
The child 9738 is dead

=>exec p(10,50000,&)

=>This is program p and it prints for the 1 time of 10...
ps

NEW SHELL presents the following living processes

PID NAME

7073 NEW SHELL

9740 p

=>kill 9740

You have just killed process 9740

=>The child 9740 is dead

exit

```
$ gcc p-shell.c -o p
$ gcc shell.c -o shell
$ ./shell
```

New Shell. Works properly!!

=>exec p(5,100) p(12,300) , p(50,100000,&)

```
This is program p and it prints for the 1 time of 5...
This is program p and it prints for the 2 time of 5...
This is program p and it prints for the 3 time of 5...
This is program p and it prints for the 4 time of 5...
This is program p and it prints for the 5 time of 5...
The child 9815 is dead
```

```
This is program p and it prints for the 1 time of 12...
This is program p and it prints for the 2 time of 12...
This is program p and it prints for the 3 time of 12...
This is program p and it prints for the 4 time of 12...
This is program p and it prints for the 5 time of 12...
This is program p and it prints for the 6 time of 12...
This is program p and it prints for the 7 time of 12...
```

```
This is program p and it prints for the 8 time of 12...
This is program p and it prints for the 9 time of 12...
This is program p and it prints for the 10 time of 12...
This is program p and it prints for the 11 time of 12...
This is program p and it prints for the 12 time of 12...
The child 9816 is dead
```

```
=>This is program p and it prints for the 1 time of 50...
ps
```

NEW SHELL presents the following living processes

```
PID NAME
```

```
7073  NEW SHELL
```

```
9817  p
```

```
=>exit
```

```
There are still living processes Do you want to kill them? (y/n): y
```

```
You have just killed process 9817
```

```
The child 9817 is dead
```

As it can be seen, this shell/small operating system follows the FCFS scheduling policy. All programs wait for their turn in order to be executed. In this exercise you are to asked to:

- Become familiar with the code and understand all the sections and the commands.
- Experiment with different commands and arguments to the shell.
- Change the FCFS scheduling policy to Round Robin (The time quantum will be given as an argument when the shell starts). Make any necessary changes to shell.c and p-shell.c.
- Change the FCFS scheduling policy to MFQ with two Round Robin queues (The time quantum will be given as an argument when the shell starts). Make any necessary changes to shell.c and p-shell.c.
- Change the FCFS scheduling policy to preemptive Priority scheduling (The priority will be given as an additional argument to the programs). Make any necessary changes to shell.c and p-shell.c (TIP: You need to implement a non-blocking read).