Programming Assignment-7 Report
CS3523: Operating Systems-2
Priority Scheduling

Name: Vikhyath Sai Kothamasu
Roll Number: CS20BTECH11056

Part-1: System call to assign/change the priority of any process

Implementation:

The changes made are:

1. In the file `syscall.c`, added the new system call to the array of system calls (`syscalls[]`). Also added the statement `extern int sys_setpriority(void);` which extends to the function `int sys_setpriority(void)` in the file `sysproc.c` . The extern keyword extends the visibility to the whole program.
2. In the file `sysproc.c`, added definition to the function `int sys_setpriority(void)` which obtains the command line argument that specifies the priority to be assigned to the current process. It first runs a basic check if the specified priority is in the accepted range and if yes, it is assigned as the priority of the current process.
3. In the file `syscall.h`, added the line `#define SYS_setpriority 22` which assigns the system call number 22 to `SYS_pgtprint`.
4. In the file `user.h`, added the function definition `int setpriority(int priority);`
5. In the file `usys.s`, added `SYSCALL(setpriority)` for the user to call the function.
6. Created a file named `setPriority.c` which is an empty file.
7. For the new file to be compiled and included, added `_setPriority\` in the makefile.
8. In the file `defs.h`, added the function definition `int setpriority(int priority)` so that it can be accessible across other files which include `defs.h`
9. In the file `proc.h`, add another parameter named priority under the process structure so that each process will have a designated priority value assigned to it.

10. In the file `proc.c`, added the line `p->priority = 5;` which assigns the value of 5 as default priority to every new process created.

Observations:

1. One problem with including a system call to change the priority of a given process is the misuse of the priority. The user can randomly assign high/low priority to some process which may hinder the execution of other processes. Priority should be assigned based on the task it does, waiting time, etc.
2. One method to mitigate the aforementioned problem is to give limited power to the user as in he/she can assign priority of his/her own wish only for a set of processes that does not affect other processes.

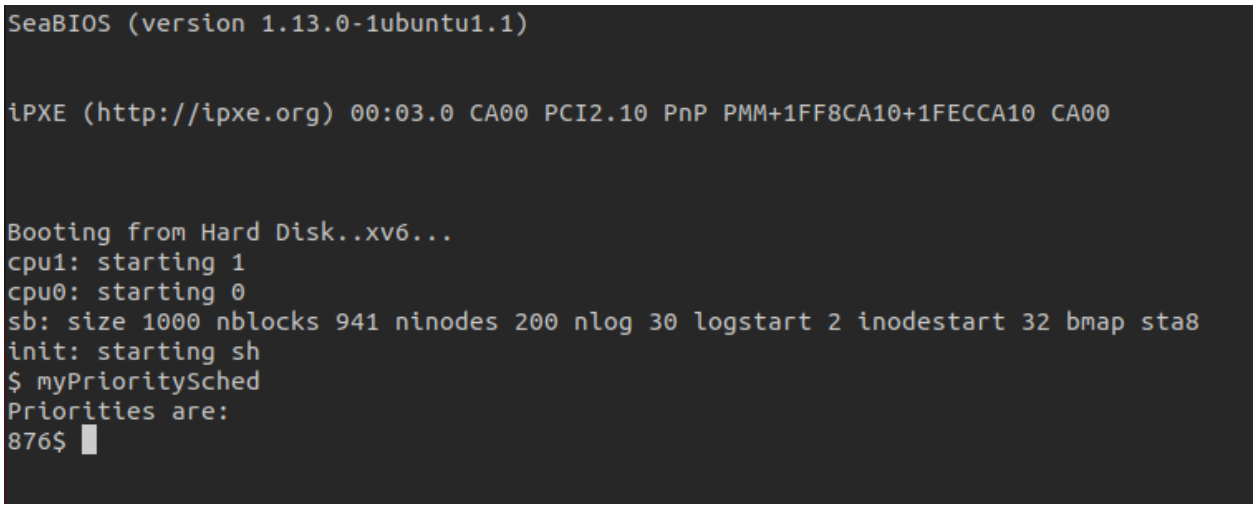
Part-2: Scheduling policy based on the priority of processes

Implementation:

The changes made are:

1. In the file `proc.c`, modified the `void scheduler(void)` function so to implement priority-based scheduling. I first ran a for loop to identify the process which is runnable and has the highest priority. Once we have the highest priority, we can run the process with the found highest priority.
2. Created a new file named `myPrioritySched.c` which acts as a user program for testing the implementation of scheduling and priorities. It basically creates n number of child processes and assigns them some priority calculated using the addition and modulo functions. We then wait for the child processes to finish and merge with the main thread. We use sleep to act as if each process is doing some time-consuming task.
3. For the new file to be compiled and included, added `_myPrioritySched\` in the makefile.

Output:



Observations:

1. The default implementation of the scheduling policy is that of round-robin.
2. One problem with the priority-based scheduling policy is that processes with low priority can starve because processes with higher priority can take loads of CPU time as well as new processes queue in, there might be a new process with higher priority and this can continue for a long period of time. Thus, a process with lower priority will not be able to utilize the CPU.
3. One method to mitigate this problem is to increase the priority of a process as it waits for longer periods of time. This way, it does not starve as eventually, it will reach the highest priority and thus will be able to utilize the CPU.

Learning:

Through this assignment, I have learnt how to add another parameter to the process state structure and about the scheduling algorithm xv6 uses by default. From part 1, how to use a new parameter added to use it in part 2 and how a user program with multiple child processes utilizes.