Homework 2
Part 3

ajavade@ncsu.edu
200153995

Modified files:

1.) clkhandler.c: To reduce TC (time to completion) variable for each running process.
For ctr1000++. To change the values of counters and timers for all MLFQ objects.
To change priority and allocated time for particular queue for each running process.
2.) clock.h
3.) process.h : For adding variables to procent.
3.) initialize.c : nullproc is given usr_proc==0 and other small additions.
4.) create.c : All system processes are given usr_proc=0.
5.) resched.c : To handle complete rescheduling of the system. Choose the right process to run according to priority. Choose system processes over user processes always.
6.) kill.c :Terminate processes and print termination time.
7.) ready.c : For inserting in ready (system procs) list and user process list accordingly.
8.) prototypes.h : For adding new functions like boost(), create_user_proc(),burst_execution,etc.

Created files:

create_user_proc.c: For creating user processes similar to system processes. usr_proc==1.
Implemented burst_execution() and boost() function here.

Implementation Methodology:

Using a ready (system) list and a user process list. Always give priority to system processes on ready list over user processes.
At every resched() call if currently executing process is a system process let it run as it would normally run.
If currently executing process is a user process then traverse the readylist to:
Find a system process with highest priority on ready list that will be context switched.
OR
Among all user processes in user list process with higher priority than the currently executing one to context swtich.
If both of these processes do not exist then continue executing the currently running user process.

Made user list which works similarly to the ready list.
Priorities 10, 8 ,6, 4 for 4 queues.
TA = TA/8 , TA/4, TA/2, TA respectively.

After every TA (Time allocated) (as per queue) change the priority and TA to get the process to the back of the user list.

Use Case:

```
TIME_ALLOCATED 100
PRIORITY_BOOST_PERIOD 300
main{
        recvclr();

        int sl_time=5000;
        pid1 = resume(create_user_proc(burst_execution, 1024, 100,"fun_A",3,2,200,100));
        sleep(TIME_ALLOCATED);
        pid2 = resume(create_user_proc(burst_execution, 1024, 100,"fun_B",3,2,100,0));

        sleepms(sl_time);


}
```

1. MAIN resume creates process 1 and as MAIN sleeps it runs for 100ms
2. 1 runs through this time losing priority as per TA in every queue.
3. MAIN wakes up and resume creates process 2.
4. 2 starts running as it is in high priority queue.
5. 1 and 2 run in round robin.
6. 1 sleeps.
7. 1 wakes up and 1 and 2 run in round robin and then experience a boost.
8. This boost does not help as the still keep running in round robin only at a higher priority queue.
9. 1 terminates
10. 2 fails to terminate for a long time until main is called (I cannot explain this behaviour).