<u>Workload and graph description</u>

<u>IO intense</u>

We have created 4 IO intense workloads in order to show how the MLFQ scheduler schedules IO processes. The workloads are as follows:

1. Graph_loop_p0.c
   This workload is an example of how IO processes can game the MLFQ scheduler to keep the process at priority 0 for a long time. In this workload we have created a for loop than iteratively calls sleep in such a way that it yields prior to the time interrupt occurring. As it yields prior to timer interrupt occurring, the number of time ticks for the process no not get incremented and it remains at priority 0 even though it ran and used the CPU.
2. Graph_loop_p1.c
   This workload is similar to the above workload. But the for loop runs in such a way that the process does not yield before the timer interrupt occurs. Due to this the process priority is reduced from 0 to 1.
3. Graph_loop_p3.c
   This workload is similar to the above workload. But the for loop runs in such a way that the process does not yield before the timer interrupt occurs. Due to this the process priority is reduced from 1 to 2.
4. Graph_loop_p4.c
   This workload is similar to the above workload. But the for loop runs in such a way that the process does not yield before the timer interrupt occurs. Due to this the process priority is reduced from 2 to 3.

Graph2.pdf : the y axis of this graph plots the number of timer ticks taken by each process at each priority level. The x axis plots the processes.

<u>Batch Workloads</u>

We have created the following workloads to display the way in which non IO intense workloads operate .

graph_dir -> This workload creates directories
graph_exec -> This workload creates a child process . The child process execs the "ls" program
graph_mem -> This workload allocates memory and frees memory using inbuilt system calls.

Graph1.pdf: the y axis of this graph plots the number of timer ticks taken by each process at each priority level. The x axis plots the processes.