Ryan “Bob” Dean
CS 460 - Operation Systems
Paper Assignment 1

1.) *Why do we need user space and kernel space? Describe how a process in user space can invoke functionality of kernel space.*

By giving the user access to the kernel space, you give them access to the hardware, which you want to limit their access to for security reasons. However, users have access to system calls that allow them to do certain functions. For example, by calling malloc, the user can make a system call to allocate a certain amount of memory for use by the user.

2.) Which of the following components of program state are shared across threads in a multithreaded process? Register values, heap memory, program stack, global/static variables.

Heap memory and variables.

3.) *Why is context switching for user-level threads much more efficient than context switching for kernel-level threads?*

For kernel level threads, creating and switching between threads requires a system call which is much slower.

4.) *For the following jobs, show the schedules (Gantt charts) illustrating the execution of the jobs. Then report (a) average wait time, (b) average turnaround time, and (c) job throughput using FCFS, Round Robin with T = 5 and Shortest Time Remaining First. Assume context switch times are negligible.*

Job	P1	P2	P3
Burst length	30	25	50
Arrival time	60	10	0

Note: I have no idea what “throughput” means if context switches are free. Aren’t they all 3/105?

FCFS

P3 (50)	P2 (25)	P1 (30)

Wait Time: (0 + 40 + 15) / 3 = 18.33
Turnaround Time: (50 + 65 + 45) / 3 = 53.33
Throughput: 3/105

Round Robin

P3 (10)	P2 (5)	P3 (5)	P2 (5)	P3 (5)	P2 (5)	P3 (5)	P2 (5)	P3 (5)	P2 (5)	P3 (5)	P1 (5)	P3 (5)	P1 (5)	P3 (5)	P1 (5)	P3 (5)	P1 (15)

Wait Time: (40 + 20 + 15) / 3 = 25
Turnaround Time: (90 + 45 + 45) / 3 = 60
Throughput: 3/105

STRF

P3 (10)	P2 (25)	P3 (40)	P1 (30)

Wait Time: (0 + 0 + 25 + 15) / 4 = 10
Turnaround Time: (75 + 25 + 45) / 3 = 48.33
Throughput: 3/105

5. For the following jobs, show the resulting schedule (Gantt chart) and the state of the ready queues if we use a 3-level preemptive MLFQ: HighPriority (RR, T = 10), MedPriority (RR, T = 20), and LowPriority (FCFS). A preemptive MLFQ means that, if a job Pm was in the medium queue, is currently executing, then if a job Phis introduced in the highest queue, it will preempt Pm. Pm is placed back in the medium queue, but its age is now 1.

An aging function is implemented: If a process in level n is preempted by the scheduler 3 times, then it is promoted into the tail of the queue in level n - 1.

If a process blocks for I/O, it is promoted one level, and placed at the end of the next - higher queue.

The durations of context-switches are negligible.

Show all promotion/demotion activities among the ready queues. You can stop as soon as P2 goes for its third I/O burst.

Job	P1	P2	P3
Burst length	30	25	50
Arrival time	60	10	0
I/O burst length	10	5	1
Period between I/O bursts	15	8	16

Activity and priority (running time in parentheses)

HighPri	P3 (10) (pri -)	P2 (8) (burst)		P2 (8) (burst)		N/ A (1)	P3 (10) (pri -)	P2 (8) (burst)
MedPri			P3 (5)		P3 (1) (burst) (pri +)			
LowPri								

0 10 18 23 31 32 33 43 51