CSSE 332 – Operating Systems
Rose-Hulman Institute of Technology
Computer Science and Software Engineering Department

Exam 1 – Paper Part

Name:_____ Section:_____ CM:_____

This exam consists of two parts. The first part is to be done on paper without using your computer. The second part of the exam is to be done on your computer. You have the full lab period (a total of three periods) to complete the entire exam.

Instructions: The paper part of the exam is closed book, open notes limited to one double-sided sheet of hand written notes, but **no computer or electronic devices**. Write all of your answers in the spaces provided.

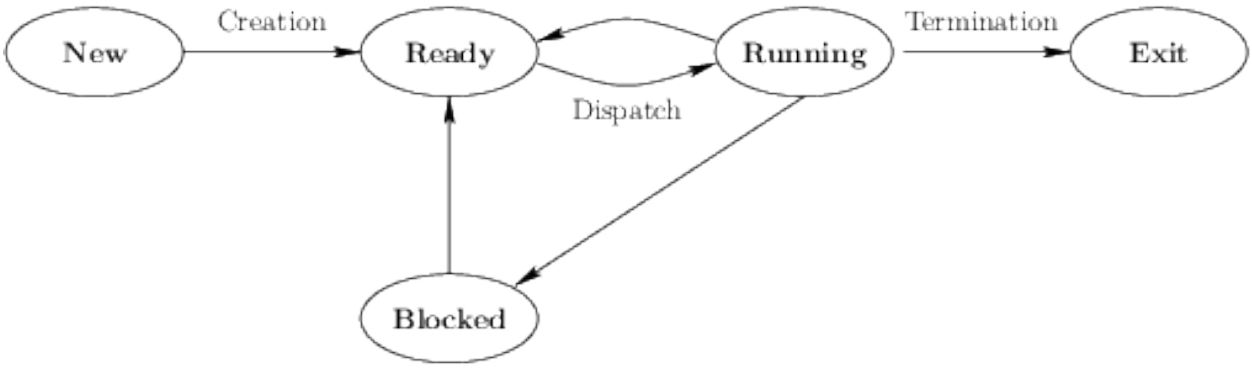
When you complete the paper part of the exam, turn it in to an exam proctor. You may then begin using your computer. **Use of your computer before turning in the paper part of the exam will be considered academic dishonesty.**

To allow sufficient time to work on the computer part of the exam, we suggest using no more than 50 minutes for the paper part.

Please begin by putting your name on the first page and your initials on every page of the exam. We encourage you to skim the entire exam before answering any questions and show all your work to receive partial credit.

Problem	Points available	Your Points
1	10	
2	10	
3	8	
4	7	
C	65	
Total	100	

Problem 1 (10 points) Assume that a particular operating system uses the five-state process model discussed in class.



(a) (2 points) There's a state transition from the Running state to the Ready state. What is the most common event that leads to this state transition?

Answer: The OS preempts the current running process.

What type of interrupt is responsible for triggering such an event?

Answer: CPU scheduler timer interruption.

Consider the sequence of events and timeline given below.

Events:

i. Process P1 is created, admitted, and dispatched.

ii. Process P2 is created and admitted.

iii. The running process issues an I/O request.

iv. The process at the head of the ready queue is dispatched.

v. Process P3 is created and admitted.

. . . . . .

vi. Process P4 is created and admitted.

vii. The running process times out.

viii. The process at the head of the ready queue is dispatched.

. . . . . .

ix. The I/O operation has completed.

Timeline:

t_0 t_1 t_2 t_n

i ii iii iv v vi vii $viii$ ix

(b) (4 points) Draw a figure that depicts the state of each process in the system at time t_1.

Running: P2
Ready Queue: P3
I/O Queue: P1

(c) (4 points) Draw a figure that depicts the state of each process in the system at time t_2.

Running: P3
Ready Queue: P4 $\rightarrow$ P2
I/O Queue: P1

Problem 2 (10 points) Consider the following set of processes with their lengths of CPU bursts and arrival times given in milliseconds. Assume they are scheduled on a uniprocessor system.

Fill in the Gantt charts using the given scheduling policies and the table below. If a new process is admitted at the same time that an existing process is preempted, the new process enters the ready queue first.

Note: processes are always enqueued at the tail of a ready queue and dequeued from the head of the queue.

RR2: Round Robin scheduling algorithm with a time quantum of 2 milliseconds.

MLFQ: Multi-level feedback queue scheduling algorithm with 3 ready queues indexed from 0 to 2. Processes in each of ready queues 0 and 1 are scheduled in FCFS order for a quantum of 3 milliseconds each. All new processes are admitted in queue 0. If a process from queue 0 has exhausted its quantum and needs more CPU time, then it is demoted to queue 1. From queue 1, if a process exhausts its quantum, it is demoted to queue 2. Processes in queue 2 are scheduled using the Round Robin scheduling algorithm with a quantum of 3 milliseconds.

Process	Arrival Time	Service Time
A	0	8
B	1	2
C	2	6
D	4	4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
RR2	A	A	B	B	C	C	A	A	D	D	C	C	A	A	D	D	C	C	A	A
MLFQ	A	A	A	B	B	C	C	C	D	D	D	A	A	A	C	C	C	D	A	A

Problem 3 (8 points) Your answers to these questions should be as specific and brief as possible. You should not offer explanations if they are not requested.

(a) (4 points) In class we discussed several limitations of user level threads. List 2 critical issues or disadvantages associated with them.

i. They can't have a distinct memory space from other threads in the same process all user-space threads share the same memory

ii. They can't run simultaneously on multiple CPUs if my system has multiple CPUs

iii. If one userspace thread does something that causes a wait (e.g. requests something from memory, disk, the network) the OS will de-schedule all the userspace threads

iv. They have to manually yield they can't be preempted (moved off the CPU involuntary)

(b) (4 points) List 2 advantages of user level threads.

i. Advantage 1:
Answer: No operating system support required: user level threads can be implemented on any system.

ii. Advantage 2:
Answer: Low management cost: user level threads are fairly cheaper to maintain and run compared to kernel level threads.

Problem 4 (7 points) Your answers to these questions should be brief and to the point.

(a) (3 points) Most of our processes that use signals have a specific function or block of code included to handle specific signals. What would happen to one of our processes that receives the signal triggered by the Ctrl + C keyboard key combinations if that process lacked a signal handler?

Answer: The signal handler registered by OS will run to kill the process.

(b) (2 points) If we wanted to ignore a specific signal while something else is happening in the code, what would we create and use in the code to accomplish this?

Answer: Use signal mask to block that particular signal.

(c) (2 points) If a SIGALARM signal is triggered and the signal handler for this signal is running, what would typically happen to a subsequent SIGALARM that is received while the signal handler is still running?

Answer: The signal will be automatically blocked inside the signal handler.