Bronson Edralin
EE 468 Midterm Exam Fall 2014
Due November 26, 2013 WED. Hardcopy due at the beginning of class.

Instructions: This is a take home exam. You may use your book, notes, homeworks and projects and their solutions. You may look up information on the Internet, except searching for the answers of this exam. You may use calculators and Microsoft word and excel. You may use wiliki for programming. **You are not allowed to discuss this with anyone, use message boards, or any type of communication, i.e., you are to do your own work**. Problem 1 is a programming assignment which requires source code prob1.c. Source code prob1.c is attached to this exam on laulima. Your program should be able to compile and execute on wiliki. If I can't compile and run it on wiliki then you won't get credit.

To submit your program for Problem 1, put them in a tarred gzipped directory named EE468<First six letters of your last name and First initial>.tar.gz, e.g., if your name is John Doe then EE468DoeJ.tar.gz or if your name is Jane Washington then EE468WashinJ.tar.gz. Your directory should also have a README file with your name and any instructions for compiling and running your programs. Upload into laulima. You have until 11:30 PM on the due date November 26, 2014 Wednesday.

Turn in the written portion (problems without programs) in class as a hard copy on the due date November 26, 2014 Wednesday.

Problem 1 (15 pts). Find the program prob1.c. The program will create 9 child processes, numbered 0 through 8. A child process does the following

1. Outputs "Child process k is created"
2. Sleeps for one second
3. Outputs "Child process k is terminated"

Rewrite the program so that the parent process will

- Create three child processes
- Then wait for the three to terminate
- Then create another three child processes
- Then wait for the three to terminate.
- Then finally create another three processes
- Then wait for these to terminate

Use the wait( ) system call or variations of it, e.g., waitpid or waitid. Your program should not modify the child, but it can modify anything else. The child process is indicated by comments within the code.

Problem 2. Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time (ms)	Priority
P1	3	2
P2	2	1
P3	4	3
P4	1	4

The processes are assumed to have arrived in the order P1, P2, P3, P4, all at time 0.

(a) (10 pts) Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, non-preemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).

First Come First Serve (FCFS)

FCFS	**Initial**	**Rd 1**	**Rd 2**	**Rd 3**	**Rd 4**	**Rd 5**	**Rd 6**
Time:	0	0->2	2->3	3->5	5->7	7->9	9->10
Quant:		2	1	2	2	2	1
P1:	3	1	0	0	0	0	0
P2:	2	2	2	0	0	0	0
P3:	4	4	4	4	2	0	0
P4:	1	1	1	1	1	1	0

In FCFS, P1 takes 3ms to finish; P2 takes 5ms to finish; P3 takes 9ms to finish; P4 takes 10ms to finish.

Shortest Job First (SJF)

SJF	**Initial**	**Rd 1**	**Rd 2**	**Rd 3**	**Rd 4**	**Rd 5**	**Rd 6**
Time:	0	0->1	1->3	3->5	5->6	6->8	8->10
Quant:		1	2	2	1	2	2
P1:	3	3	3	1	0	0	0
P2:	2	2	0	0	0	0	0
P3:	4	4	4	4	4	2	0
P4:	1	0	0	0	0	0	0

In SJF, P4 takes 1ms to finish; P2 takes 3ms to finish; P1 takes 6ms to finish; P3 takes 10ms to finish.

Non-Preemptive Priority (NP Priority)

NP	**Initial**	**Rd 1**	**Rd 2**	**Rd 3**	**Rd 4**	**Rd 5**	**Rd 6**
Time:	0	0->1	1->3	3->5	5->7	7->8	8->10
Quant:		1	2	2	2	1	2
P1:	3	3	3	3	1	0	0
P2:	2	2	2	2	2	2	0
P3:	4	4	2	0	0	0	0
P4:	1	0	0	0	0	0	0

In Non-Preemptive Priority, P4 takes 1ms to finish; P3 takes 5ms to finish; P1 takes 8ms to finish; P2 takes 10ms to finish.

Round Robin (RR)

RR	Initial	Rd 1	Rd 2	Rd 3	Rd 4	Rd 5	Rd 6
Time:	0	0->2	2->4	4->6	6->7	7->8	9->10
Quant:		2	2	2	1	1	2
P1:	3	1	1	1	1	0	0
P2:	2	2	0	0	0	0	0
P3:	4	4	4	2	2	2	0
P4:	1	1	1	1	0	0	0

In Round Robin, P1 finishes 8ms; P2 finishes 4ms; P3 finishes 10ms; P4 finishes 7ms.

(b) (5 pts) What is the turnaround time of each process for each of the scheduling algorithms in part a?

Turnaround time (ms)

First Come First Serve (FCFS)			
P1=3	P2=5	P3=9	P4=10
Shortest Job First (SJF)			
P1=6	P2=3	P3=10	P4=1
Nonpreemptive priority			
P1=8	P2=10	P3=5	P4=1
Round Robin (RR)			
P1=8	P2=4	P3=10	P4=7

(c) (5 pts) What is the wait time of each process for each of the scheduling algorithms in part a?

Waittime (ms)

First Come First Serve (FCFS)			
P1=0	P2=3	P3=5	P4=9
Shortest Job First (SJF)			
P1=3	P2=1	P3=6	P4=0
Nonpreemptive priority			
P1=5	P2=8	P3=1	P4=0
Round Robin (RR)			
P1=5	P2=2	P3=6	P4=6

Problem 3. (10 pts) Consider the following two processes. Assume that both processes start at the same time.

Process	Period p	Processing time t
P1	6	3
P2	8	3

(a) Can the two processes be scheduled under rate-monotonic scheduling? Illustrate your answer using a Gantt chart such as the ones in Figure 6.16-Figure 6.19 in the textbook.

Feasible schedule that always meet deadlines exists if CPU utilization is below a specific bound (depending on number of tasks):

- C_i=Processing time t
- T_i=The release period p
- n=number of processes to be scheduled
- $U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n\left(2^{\frac{1}{n}} - 1\right)$

The utilization will be:

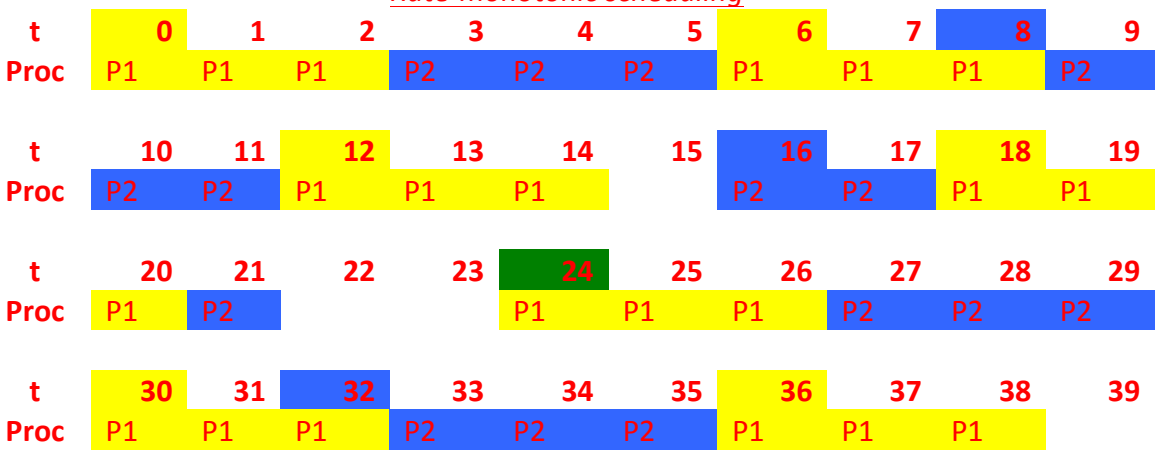
- $\frac{3}{6} + \frac{3}{8} = 0.875$
- 87.5%

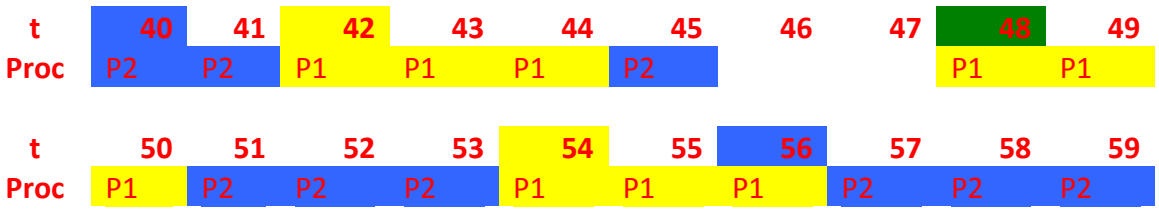
Sufficient condition for 2 processes, under which we can conclude the system is schedulable:

- $U = 2\left(2^{\frac{1}{2}} - 1\right) = 0.8284$
- 82.84%
- Since 0.875 > 0.8284, it is questionable.

Priority is raised for processes with lower period. From the Gant Chart, you will see that P1 starts first and takes priority. P1 will start every time 0, 6, 12, 18, 24, 30, 36….etc while P2 wants to start every 0, 8, 16, 24, 32, 40 but P1 will have priority over P2. After doing some iterations, it is obvious that all the processes will finish in time before the next time. Green represents the time when both periods overlap and this scheduling algorithm passed.

Rate-monotonic scheduling





(b) Illustrate the scheduling of these two processes using earliest deadline first (EDF) scheduling in a Gantt chart.

Feasible schedule that always meet deadlines exists if CPU utilization is below a specific bound (depending on number of tasks):

- C_i=Processing time t
- T_i=The release period p
- n=number of processes to be scheduled
- $U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$

The utilization will be:

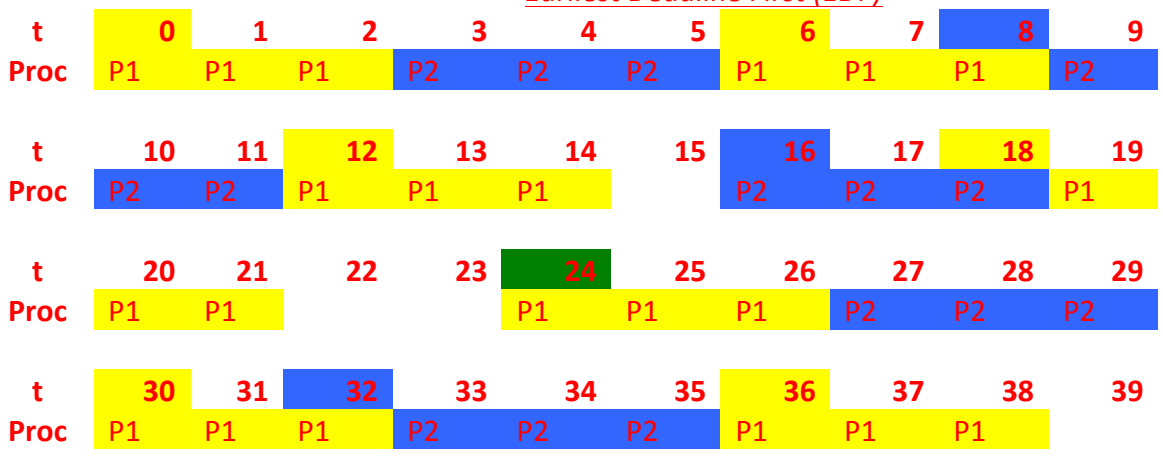
- $\frac{3}{6} + \frac{3}{8} = 0.875$
- 87.5%

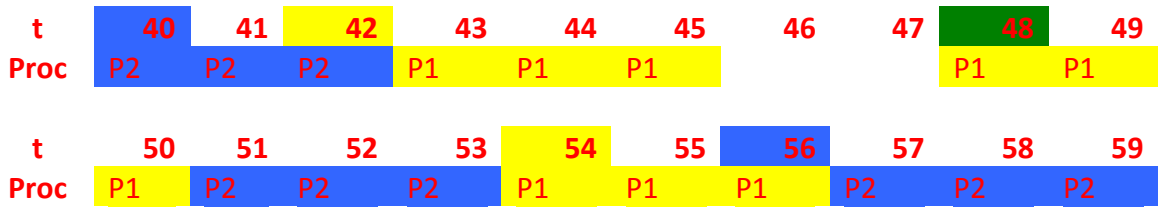
Sufficient condition for 2 processes, under which we can conclude the system is schedulable:

- $U = 87.5\% \leq 1$ (TRUE)
- Therefore, this system should be schedulable but lets prove it anyways

EDF is called the Earliest Deadline First. It is assumed that the highest priority will be given to the process with the least time period to finish. Therefore, process P1 will start first. Each process is preempted after every time period. P1 starts every time period 6 while P2 starts every time period 8. Whenever it is time for P1 or P2 to start at its regular time period, highest priority will be given to the least time period to finish. From the Gant Chart below, every process was allowed to finish before a new process started. This scheduling algorithm is definitely stable and passes.

Earliest Deadline First (EDF)





Problem 4 (10 pts). Consider the following snapshot of a system:

	Allocation					Max			
Process	A	B	C	D		A	B	C	D
P0	3	1	0	4		5	1	1	7
P1	0	1	1	0		3	2	1	1
P2	1	1	2	4		2	5	2	6
P3	1	1	3	1		2	3	3	3

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the processes may complete. Otherwise, illustrate why the state is unsafe, and identify the processes that causes the system to be unsafe.

Need = Max – Allocation

	Need
	A B C D
P_0	2 0 1 3
P_1	3 1 0 1
P_2	1 4 0 2
P_3	1 2 0 2

(a) Available = (2, 0, 1, 3)

RESULT

	Need	***Allocated***	***Res Avail.***
	A B C D	A B C D	A B C D
P_0	2 0 1 3	3 1 0 4	2 0 1 3
P_1	3 1 0 1	0 1 1 0	5 1 1 7
P_3	1 2 0 2	1 1 3 1	5 2 2 7
			6 3 5 8

This is not safe. Processes P0, P1 and P3 are able to finish, but P2 will not be able to finish because there are not enough available resources.

(b) Available = (3, 1, 0, 3)

RESULT

	Need				***Allocated***				***Res Avail.***			
	A	B	C	D	A	B	C	D	A	B	C	D
P_1	3	1	0	1	0	1	1	0	3	1	0	3
P_0	2	0	1	3	3	1	0	4	3	2	1	3
P_3	1	2	0	2	1	1	3	1	6	3	1	7
P_2	1	4	0	2	1	1	2	4	7	4	4	8
									8	5	6	12

This is safe. The processes finish in this order: P1, P0, P3, and P2.

Problem 5 (10 pts) Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 1500, and the previous request was at cylinder 1000. The queue of pending requests, in FIFO order, is:

1250, 4000, 3000, 1500, 500

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

a) FCFS

The First Come First Serve (FCFS) schedule is 1500, 1250, 4000, 3000, 1500, and then 500.

Calculations:
- (1500-1250)+(4000-1250)+(4000-3000)+(3000-1500)+(1500-500)=6500
- **Total seek distance is 6500 cylinders**

b) SSTF

The Shortest Seek Time First (SSTF) schedule is 1500, 1500, 1250, 500, 3000, and then 4000.

Calculations
- (1500-1500)+(1500-1250)+(1250-500)+(3000-500)+(4000-3000)=4500
- **Total seek distance is 4500 cylinders**
- Note: This is assuming it won't go back to 1500 since it started there already.

c) SCAN

SCAN schedule is 1500, 1500, 1250, 500, 0, 3000, and 4000

Calculations
- (1500-0)+(4000-0) = 5500
- **Total seek distance is 5500 cylinders**
- Note: This is assuming it won't go back to 1500 since it started there already.

d) LOOK

LOOK schedule is 1500, 1500, 1250, 500, 3000, and 4000.

Calculations
- (1500-500)+(4000-500)=4500
- **Total seek distance is 4500 cylinders**
- Note: This is assuming it won't go back to 1500 since it started there already.

e) C-SCAN

Circular-SCAN (C-SCAN) schedule is 1500, 1500, 1250, 500, 0, 4999, 4000, and 3000.

Calculations
- (1500-0)+(4999-0)+(4999-3000)=8498
- **Total seek distance is 8498 cylinders**
- Note: This is assuming it won't go back to 1500 since it started there already.

f) C-LOOK

Circular-LOOK (C-LOOK) schedule is 1500, 1500, 1250, 500, 4,000 and 3,000.

Calculations
- (1500-500)+(4000-500)+(4000-3000)=5500
- **Total seek distance is 5500 cylinders**
- Note: This is assuming it won't go back to 1500 since it started there already.

Problem 6. Consider a disk system with 100 pairs of mirrored disks, i.e., there are a total of 200 disks. A pair of mirrored disks is considered a failure if both disks are down. The disk system is considered a failure if a pair of mirrored disks are a complete failure. Suppose the mean time to failure (MTTF) of a single disk is 9000 hours and the mean time to repair (MTTR) a disk is 24 hours.

a) (5 pts) What is the MTTF of a pair of mirrored disks (your answer can be approximate but it has to be reasonably accurate)?

100 pairs of mirrored disks
MTTF = 9000 hours
MTTR = 24 hours

Calculations:
- Single Disk
  - Probability of Failure = 1/9000
  - Probability of Repair = 1/24
- Mirror Disk
  - Probability of Failure = 2*(1/9000)=1/4500

- MTFF=4500
- Mean # failures until complete failure = $\frac{p_r}{p_f} = \frac{9{,}000}{24} = 375$
- **Mean time to complete failure=$\frac{9000^2}{2*24} = 1.69 * 10^6 Hours$**

b) (5 pts) What is the MTTF of the disk system (your answer can be approximate but it has to be reasonably accurate)?

Probability of complete failure = $\frac{1}{1.69*10^6}$

Probability of complete failure of 100 systems = $100 * \frac{1}{1.69*10^6} = \frac{1}{1.69*10^4}$

MTFF of complete failure for 100 system = 16875 Hours

Problem 7. Consider a computer memory system with a 64-bit logical address and 2KB page size. The system supports up to 32 GB of physical memory. How many entries are there in each of the following? You may assume that an entry in a page table is always 16 bytes.

a) (5 pts) How many bytes are there in the page table in a single-level page table system?

Given:
- 64 bit logical address
- 2kB page size
- 32 GB physical memory

Calculating…
- 2kB ≈ 2048B = 2^{11}B
  - 11 bits offset
  - 2^{11} *bytes per page table*
- (logical address) = (bits for page #) + (bits for offset)
- (bits for page #) = (logical address) – (bits for offset)
- (bits for page #) = 64 – 11
- (bits for page #) = 53
  - 2^{53} *page table*
- 32GB physical memory = 2^{35} bytes
- $\frac{(2^{35}\ bytes\ of\ memory)}{(2^{11}\ bytes\ per\ page)} = 2^{24}\ physical\ pages$
  - 24 bits needed for physical page #

Page table entry
- 24 bit physical page # = 3 bytes needed
- Access control bits = 1 byte
- Page table entry = 3+1 = 4 bytes

$Page\ table\ size = (2^{53}\ page\ table\ entries) * (\ 2^2\ bytes\ for\ page\ table) = \ 2^{55} bytes$

Page table size ≈ 32768 TB

b) (5 pts) Consider a hierarchical page table system with two levels. How many bytes are there in the page tables, i.e., the sum of both outer and inner page tables?

Given:
- Hierarchical page table system with 2 levels

Calculated
- 2^{24} physical pages
- Assuming each entry to be 16 bytes = 2^4 bytes
- Total number of bytes in 2nd page table is $2^{24} * 2^4 = 2^{28}\ bytes = \mathbf{256MB}$
- This table is also paged. We refer to this as first page table. It has $\frac{2^{28}}{2^{11}} = 2^{17} = 128k$
  - First table has 128k entries
  - 64bit = 8 bytes
  - First page table has size = $128k * 8bytes = \mathbf{1MB}$

Result
256MB+1MB=**257MB**

c) (5 pts) Consider a hierarchical page table system with three levels. Assume a 64-bit logical adddress, 2KB page size, and 16 bytes per page entry. What is the largest amount of physical memory (in bytes) that can be supported? Leave your answer as a power of 2 (e.g., 2^20) or sums of powers of 2.

Given:
- Hierarchical page table system with 3 levels

Calculated
- 2^{24} physical pages
- Assuming each entry to be 16 bytes = 2^4 bytes
  - Total entries in first table is $\frac{2^{20}}{2^{11}} = 2^9\ entries$
  - $2^9 * 16\ bytes = 8kB$

Result:
- 8kB + 1MB + 256MB = **257.008MB**

Problem 8 (10 pts). Consider memory partitions shown in the following table. Consider placing processes of 75 KB, 50 KB, 100 KB, 175KB in order. Fill the table with these processes. Indicate any processes that are blocked from memory.

Memory Partitions	**First-Fit**	**Best-Fit**	**Worst-Fit**
90KB	75kB	75kB	

200KB	50kB	175kB	75kB
100KB	100kB	50kB	100kB
150KB		100kB	50kB

For First-Fit, 175kB was blocked from memory.
For Best-Fit, no processes were blocked from memory.
For Worst-Fit, 175kB was blocked from memory.