Daniel Frey
CS 4500-001
Homework 1
3/16/19

I have neither given or received unauthorized assistance on this work
Name: Daniel Frey *Date:* 3/16/19

- **Chapter 1**
  1. **What are the differences between a trap and an interrupt?**
     The difference between a trap and an interrupt is that a trap is when a user program requests OS services and an interrupt is when a hardware device requests OS services.

  2. **What are the main structures of contemporary operating systems, and their advantages and disadvantages?**
     One of the main structures of contemporary OSs is monolithic kernel. The advantages of this structure are that it's quick and easy to implement. The disadvantages are a large kernel that is harder to maintain, there is no protection between kernel components, and the dependencies among components are complex.
     Another structure of contemporary OSs is microkernel. The advantages of this structure are that the modular design make it easily extensible, it is easy to maintain, and it is more reliable and secure. The disadvantages are that there is a performance loss, and it has a complicated process management.

  3. **How can I/O devices notify an OS of the completion of jobs? List three ways.**
     a. Polling – I/O devices puts info in a status register.
     b. Interrupt – I/O device gets attention from processor by interrupting what it is doing
     c. DMA (Direct Memory Access) – delegates I/O responsibility from CPU

  4. **Describe the actions an OS must take to process an interrupt.**
     The CPU will write commands into command registers, then the device will signal the interrupt controller. The interrupt controller informs the CPU, which will then accept the interrupt and triggers the service routine/interrupt handler. After the interrupt handler completes, control is returned to the program.

  5. **What is a system call? How is the transition between the user-mode and the kernel-mode performed?**
     A system call is a way programs can interact with the OS. The transition between user-mode and the kernel-mode is performed with traps. The library procedure executes a trap instruction which causes the shift form user-mode to kernel-mode.

- **Chapter 2**
  1. **What is a process? How is a process different from a program?**
     A process is an abstraction of a running program. A process differs from a program in the sense that a process is the dynamic instantiation of code, data, files, etc. while a program is the static code and data.

  2. **Given the five-state process model, explain how does a process transit among these states and on what events.**

Considering the five states (new, ready, running, blocked, terminated), a process begins by being admitted where it is loaded into memory. From there, the process is ready and willing to run but must wait for the CPU. The scheduler will dispatch the process to the CPU to put it in the running state. In the running state, the process may receive an interrupt or preemption to stop the process from running and putting it back in a ready state. It may also get an I/O or event wait that will block the process. The process will become unblocked when the I/O or event is completed, once again putting it in a ready state. After the process is scheduled again, it may run to completion and exit, thus being terminated.

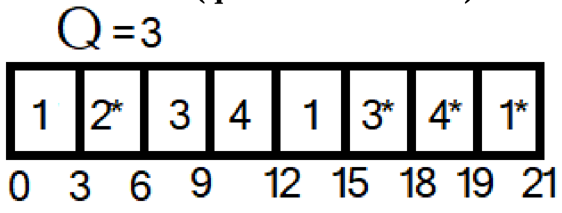
3. **What are the differences of threads and processes?**
   Differences between threads and processes are that threads are used for small, lightweight tasks, whereas processes are used for more heavyweight tasks. Also, threads within the same processes share memory space when processes do not.

4. **What is multiprogramming and why is it needed?**
   Multiprogramming is a form a parallel processing that allows several programs to run on a uniprocessor. Multiprogramming is used to maximized CPU usage, especially for uniprocessor systems.

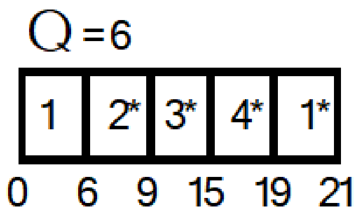
5. **Discuss the advantages and disadvantages of user-level threads and kernel-level threads.**
   Advantages of user-level threads are that they do not require OS-thread support, they are lightweight, and they each have their own customized scheduling algorithm. Disadvantages of user-level threads are that there is a lack of coordination between threads and OS kernel, and they require non-blocking system calls, like with page faults.
   Advantages of kernel-level threads are that they have knowledge of all threads so that they may be better scheduled, and they are good for applications that frequently block. Disadvantages of kernel-level threads are that they are slow and inefficient, and they are expensive to create and switch.

6. **Discuss the goals of CPU scheduling on different computer systems.**
   On a user-oriented system, CPU scheduling is used to minimize the response time, the turnaround time, and unpredictability. On a system-oriented system, CPU scheduling is designed to maximize throughput, utilization, and fairness.

7. **Assume that the following processes are to be executed on a uniprocessor system. Based on their arrival time and CPU burst, calculate the average turnaround time and response time of these processes under the following scheduling policies:**
   a. **FCFS**
      Turnaround time: ((8-0)+(11-1)+(17-2)+(21-3))/4 = 12.75
      Response time: ((0-0)+(8-1)+(11-2)+(17-3))/4 = 7.5
   b. **Round Robin (quantum = 3 and 6)**

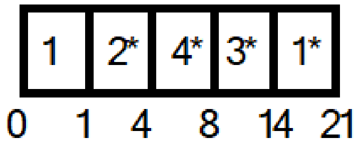


Turnaround time: ((21-0)+(6-1)+(18-2)+(19-3))/4 = 14.5
Response time: ((0-0)+(3-1)+(6-2)+(9-3))/4 = 3

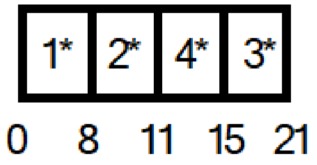


Turnaround time: ((21-0)+(9-1)+(15-2)+(19-3))/4 = 14.5
Response time: ((0-0)+(6-1)+(9-2)+(15-3))/4 = 6

c. **Shortest Job First (both preemptive and non-preemptive)**
Preemptive:



Turnaround time: ((21-0)+(4-1)+(14-2)+(8-3))/4 = 10.25
Response time: ((0-0)+(1-1)+(8-2)+(4-3))/4 = 1.75
Non-preemptive:



Turnaround time: ((8-0)+(11-1)+(21-2)+(15-3))/4 = 12.25
Response time: ((0-0)+(8-1)+(15-2)+(11-3))/4 = 7

Process	**Arrival Time**	**CPU Burst**
P1	**0**	**8**
P2	**1**	**3**
P3	**2**	**6**
P4	**3**	**4**

8. **What are the additional requirements of multiprocessor scheduling compared with uniprocessor scheduling? What are the possible issues?**
The additional requirements of multiprocessor scheduling are time-sharing and load-balancing. Possible issues include load-balancing for parallelism, time-sharing for groups of threads, and processor affinity.

9. **List different ways to ensure mutual exclusion.**
   a. Critical region cannot have two processes simultaneously.
   b. Assumptions about speed or numbers of CPUs should not be made.
   c. No process running outside its critical region may block another process.
   d. No process must wait forever to enter its critical region.

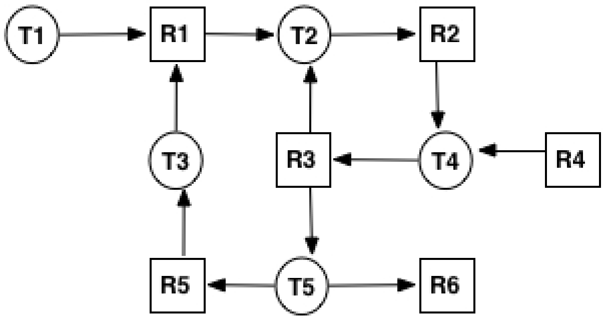
10. **What are the advantages and disadvantages of busy-waiting and sleep-and-wake approaches for mutual exclusion?**
An advantage of busy-waiting is that it avoids expensive context switch when critical region is short. Disadvantages of busy-waiting are that it wastes CPU cycles and it may prevent other things from running on single-core systems.
An advantage of sleep-and-wakeup are that it addresses the problem of wasted CPU time like in busy-waiting. A disadvantage of sleep-and-wakeup is that the time to put threads to sleep or

wake up may end up decreasing runtime performance.

11. **Why is semaphore needed? What are the commonalities and differences between semaphore and mutex?**
    A semaphore is needed to solve the critical region issue and to achieve process synchronization in multiprocessor systems. A similarity between semaphores and mutexes is that they both provide synchronization services. Differences include the possible values they can take. Also, a semaphore is more of a signaling mechanism, while a mutex is considered a locking mechanism.

12. **In the readers/writers problem (Pg 171, Fig. 2-48), explain what semaphores** mutex **and** db **do, and how they work in reader's and writer's processes, respectively.**
    Semaphore mutex controls the access to rc, the number of processes reading or wanting to read. It works in the reader process to allow the number of readers to be changed. It is not present in the writer process.
    Semaphore db controls access to the database. This works in both the writer and reader processes. In the reader process, it restricts or grants access to the database based on the number of readers. In the writer process, it restricts and grants access to the database if the writer is active.

- **Chapter 6**
  1. **Given the following resource graph, show how deadlock detection works by finding out which processes and resources are deadlocked.**



[T1] → [T1, R1] → [T1, R1, T2] → [T1, R1, T2, R2] → [T1, R1, T2, R2, T4] → [T1, R1, T2, R2, T4, R3] → [T1, R1, T2, R2, T4, R3, T2]
Since T2 appears twice in the above list, there exists a cycle ∴ T2, R2, T4, and R3 are deadlocked.

[T1] → [T1, R1] → [T1, R1, T2] → [T1, R1, T2, R2] → [T1, R1, T2, R2, T4] → [T1, R1, T2, R2, T4, R3, T5] → [T1, R1, T2, R2, T4, R3, T5] → [T1, R1, T2, R2, T4, R3, T5, R5] → [T1, R1, T2, R2, T4, R3, T5, R5, T3] → [T1, R1, T2, R2, T4, R3, T5, R5, T3, R1]
Since R1 appears twice in the list above, there exist a cycle ∴ R1, T2, R2, T4, R3, T5, R5, and T3 are deadlocked.

2. **Problem 26 in Ch. 6, 4th ed.** NOTE: **Change the row of process B as Allocated: 20111, Maximum: 22211, and the row of C as Allocated: 11110, Maximum: 21510.**

A system has four processes and five allocable resources. The current allocation and maximum needs are as follows:

	Allocated	*Maximum*	*Available*
Process A	1 0 2 1 1	1 1 2 1 3	0 0 x 1 1
Process B	2 0 1 1 1	2 2 2 1 1	
Process C	1 1 1 1 0	2 1 5 1 0	
Process D	1 1 1 1 0	1 1 2 2 1	

What is the smallest value of *x* for which this is a safe state?

Allocated: $\begin{bmatrix} 1 & 0 & 2 & 1 & 1 \\ 2 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$ Maximum: $\begin{bmatrix} 1 & 1 & 2 & 1 & 3 \\ 2 & 2 & 2 & 1 & 1 \\ 2 & 1 & 5 & 1 & 0 \\ 1 & 1 & 2 & 2 & 1 \end{bmatrix}$ Needed: $\begin{bmatrix} 0 & 1 & 0 & 0 & 2 \\ 0 & 2 & 1 & 0 & 0 \\ 1 & 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$

Resources: $[0 \quad 0 \quad 3 \quad 1 \quad 1]$

Result D: $[1 \quad 1 \quad 4 \quad 2 \quad 1]$
Result C: $[2 \quad 2 \quad 5 \quad 3 \quad 1]$
Result B: $[4 \quad 2 \quad 6 \quad 4 \quad 1]$
Result A: $[5 \quad 2 \quad 8 \quad 5 \quad 2]$

Since all processes were able to run with x = 3, the minimum value is 3.