# Name (Last, First): Clayton Samson

**Questions (10 points total)**

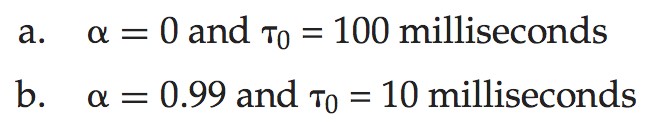
1. Describe the actions taken by a kernel to context-switch between processes. (1 point)

When an interrupt occurs the system needs to save the current context of the process that was currently running, so it can restore this context once the interrupt kernel code has finished executing. The system must save the current CPU register values, the process state noted by the program counter (PC) and memory management information in the process control block (PCB). Then loads the saved context of the new process scheduled to run.

1. Which of the following components of program state are shared across threads in a multithreaded process? Also explain the implications of such state sharing to the efficiency of creating a thread. (1 point)
   1. Register values
   2. Heap memory
   3. Global variables
   4. Stack memory

Since threads of the same process share the same address space, heap memory, and global variables multiple threads can be implemented more efficiently than multiple processes. However, multithreading can present problems as well, for one switching between threads (context switching) creates overhead. Also, if multiple threads are running at the same time and want to modify the same data, synchronization issues can arise, and methods such as a mutex locks must be introduced to ensure data consistency.

1. Consider the exponential average formula used to predict the length of the next CPU burst. What are the implications of assigning the following values to the parameters used by the algorithm? (1 point)



When assigning an alpha value for this equation you are defining how much weight you are putting towards the amount of time the last CPU burst took vs the entire history of the processes burst time. If you value the most recent CPU bust time (t*n*) over the average burst time for this process you would choose an alpha value close to 1, such as in example B above. If you want to give more consideration to the entire history (T*n*) you would choose an alpha value close to 0 such as in example A above. Setting alpha to 0 means the most recent burst time has no effect, setting alpha to 1 means the history has no effect and only the most recent burst time is considered. Setting alpha to ½ gives equal weight to both the most recent burst time and the entire history.

1. What resources are used when a thread is created? How do they differ from those used when a process is created? (1 point)

When a thread is created a small data structure must be created to hold a register set, stack, and thread priority values. Creating threads cost much less than creating a new process since the system must allocate a new process control block (PCB) which is a rather large data structure. Creating a new process creates a lot more overhead work than just creating a new thread within a process.

1. Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use non-preemptive scheduling, and base all decisions on the information you have at the time the decision must be made. (3 points)

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| *P*1 | 0.0 | 8 |
| *P*2 | 0.4 | 4 |
| *P*3 | 1.0 | 1 |

* 1. What is the average turnaround time for the FCFS scheduling algorithm?

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Completion Time |
| P1 | 0 | 8 |
| P2 | 0.4 | 12 |
| P3 | 1.0 | 13 |

8-0 = 8; 12 – 0.4 = 11.6; 13 – 1 = 12;

8 + 11.6 + 12 = 31.6

31.6 / 3 = **10.533**

**Average Turnaround Time = 10.5333**

* 1. What is the average turnaround time for the SJF scheduling algorithm?

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Completion Time |
| P1 | 0 | 8 |
| P2 | 0.4 | 13 |
| P3 | 1.0 | 9 |

For SJF algorithm the processes will execute in the following order: P1, P3, P4.

8 - 0 = 8; 9 – 1 = 8; 13 – 0.4 = 12.6

8 + 8 + 12.6 = 28.6

28.6 / 3 = **9.5333**

**Average Turnaround Time = 9.5333**

* 1. The SJF algorithm is supposed to improve performance, but notice that we chose to run process *P*1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes *P*1 and *P*2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Completion Time |
| P1 | 0 | 14 |
| P2 | 0.4 | 6 |
| P3 | 1.0 | 2 |

For the future-knowledge scheduling the processes will execute in the following order: P3, P2, P1.

14 – 0 = 14; 6 – 0.4 = 5.6; 2 – 1 = 1;

14 + 5.6 + 1 = 20.6

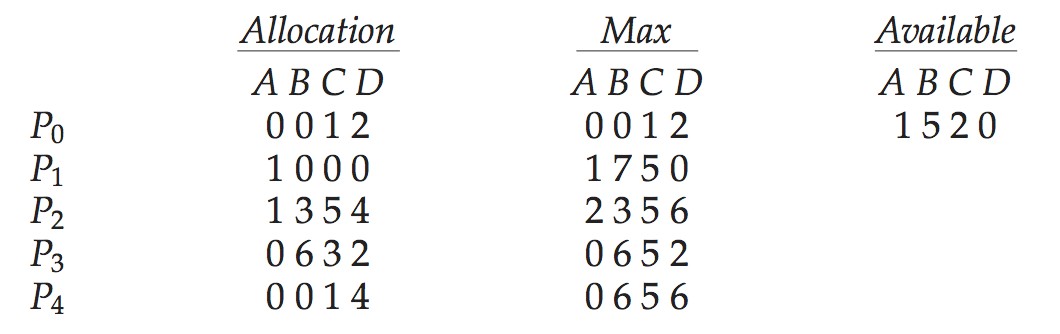
20.6 / 3 = **6.8666**

**Average Turnaround Time = 6.8666**

1. What is the meaning of the term busy waiting? What other kinds of waiting are there in an operating system? Explain why spinlocks are not appropriate for single-processor systems but are often used in multiprocessor systems. (1 point)

Spinlock or mutex lock implementations can create busy waiting, this is due to the process attempting to enter the critical area constantly ‘knocking on the door’ to see if the lock is now available. Spinlocks are not appropriate for single-processor systems since a single-processor system cannot execute two processes concurrently so there is no way for two processes to attempt to modify the same data at the same time in a single-processor system. Implementing spinlocks in a single-processor system would wasteful and creates pointless overhead.

1. Consider the following snapshot of a system:



Answer the following questions using the banker’s algorithm: (2 points)

* 1. What is the content of the matrix *Need*?

|  |  |
| --- | --- |
| Process | Need |
| P0 | 0 0 0 0 |
| P1 | 0 7 5 0 |
| P2 | 1 0 0 2 |
| P3 | 0 0 2 0 |
| P4 | 0 6 4 2 |

* 1. Is the system in a safe state?

Yes, the processes can successfully execute in the following order: P0, P3, P1, P2, P4

* 1. If a request from process *P*1 arrives for (0,4,2,0), can the request be granted immediately?

Yes, initially there is (1 5 2 0) resources available, which is more than P1 would be requesting in this example.