# Luke Pepin - Homework 1

**Released: Feb 6, 2024**

**Due: Feb 13, 2024, 11:59pm**

**Problem 1 (5 points):** What is a system call? How is system call handled by the OS?

A system call is the way for user programs to request a service from the operating system which then is executed.

When a system call is initiated by a user program it transitions control from user space to kernel space. In the kernel space, the system call handler receives the system call number and its arguments. The kernel then executes the appropriate kernel function corresponding to the system call number, performing the requested operation. After execution, the kernel returns the result to the user program. The user program evaluates the system call return value and acts accordingly.

**Problem 2 (10 points):** Let us assume that the average process runs for T sec before gets blocked for I/O. A process switch requires a time of S sec. For round-robin algorithm with quantum Q, give a formula for CPU efficiency for the followings:

1. Q>T

CPU Efficiency =

1. S<Q<T

CPU Efficiency =

1. Q=S

CPU Efficiency =

1. Q is almost 0

CPU Efficiency =

**Problem 3 (5 points):** How can you implement mutex variables? Please give the pseudo code.

Mutex = false

Locked = false

mutex\_lock(mutex):

while (mutex is locked):

wait(mutex) // Wait until the lock becomes available

Locked = true

mutex\_unlock(mutex):

locked = false

wakeup() // Signal waiting threads

**Problem 4 (5 points):** Suppose a computer has 2GB of memory. Operating System takes 512 MB of memory. Imagine each task takes up 200 MB of memory and the average I/O wait is 80%. What would be the CPU utilization if you have 3 jobs? What would be the CPU utilization if you have 10 jobs?

Formulas:

Total memory used by tasks = Number of tasks × Memory used by each task

Memory available for tasks after OS and I/O = Total memory - OS memory - Memory used by tasks

Memory used by I/O wait = Memory available for tasks × Average I/O wait

Memory used by CPU for tasks = Memory available for tasks - Memory used by I/O wait

Total CPU Utilization = (Memory used by CPU for tasks / Memory available for tasks) × 100%

3 Jobs:

Total memory used by tasks = 3 × 200MB = 600MB

Memory available for tasks after OS and I/O = 2048MB – 512MB – 600MB = 936MB

Memory used by I/O wait = 936 \* 0.8 = 748.8MB

Memory used by CPU for tasks = 936MB – 748.8MB = 187.2MB

Total CPU Utilization = (187.2MB/936MB) \* 100 = 20%

10 Jobs:

Total memory used by tasks = 10 × 200MB = 2000MB

Memory available for tasks after OS and I/O = 2048MB – 512MB – 2000MB = -464MB

Negative value here means insufficient memory

**Problem 5 (10 points):** Suppose there are four processes A, B, C and D, which arrive at 2, 3, 50, 60 msec respectively. The lengths of process A, B, C and D are 50, 100, 50 and 25 msec, respectively. Ignore the process switching overhead.

1. Compute the average turnaround time if you are using shortest job first algorithm (non-preemptive).

TAT = CT – AT

CT = AT + Length of Process

TAT = AT + Length of Process – AT= Lenght of process

D(60, 25)

A(2,50)

C(50,50)

B(3,100)

(25+50+50+100)/4 = 56.25ms

1. Compute the average turnaround time if you are using round robin algorithm, assuming the CPU quanta is 30 ms (preemptive).

A arrives at 2

B arrives to queue at 3

C arrives to queue at 50

D arrives to queue at 60

A 32 to B(-30A)

B 62 to C(-30B)

C 92 to D(-30C)

D ends at 117 to A(-25D)

A ends at 137 to B(-20A)

B 167 to C(-30B)

C ends at 187 to B(-20C)

B ends at 207(-40B)

TAT = CT – AT

A TAT = 137 – 2 = 135

B TAT = 207 – 3 = 204

C TAT = 187 – 50 = 137

D TAT = 117 – 60 = 47

(135+204+137+47)/4 = 130.75ms

**Problem 6 (5 point):** Can priority inversion problem happen if you use round-robin scheduling instead of priority scheduling? Explain.

Yes, priority inversion problem can arise when using round-robin scheduling. Priority inversion problems occur when a lower-priority task holds a shared resource when needed by a higher-priority task. Round-robin scheduling is based on their arrival time and the time quantum assigned to them; it works with no respect of priority causing priority inversion problem.

**Problem 7 (10 points):**

Suppose that two jobs arrive at the same time (time 0), and needs 20 minutes of CPU time each to finish. However, each job waits 50% time for I/O wait.

1. When will the last job finish if they are scheduled sequentially and the scheduling is non-preemptive? In this case, as you are scheduling sequentially, if you schedule job A and it gets blocked, you can not schedule job B even if B is ready to run.

Job A runs for 10 min, it is then blocked by the I/O for 10 minutes. During that time Job B still cannot run. Job A runs for another 10mins and finishes, Job B immediately starts runs for 10 mins and the I/O stops it for 10 mins it then finishes Job another 10 mins later: 60mins total (1hr)

1. When will the last job finish if the scheduling is preemptive?

If the scheduling is preemptive jobs can be completed during the I/O wait time as a result of Job A runs for 10 mins, Job B immediately starts after 10 mins during Job A’s I/O wait time, Job B then enters its wait time Job A finished and then Job B does lasting a total of 40 mins

**Problem 8 (5 points):**

Assume that aging algorithm with a=0.8 is used to predict run times. The previous four runs, from oldest to most recent, are 30, 25, 55, and 35 msec. What is the prediction for the next time?

Prediction = a \* previous prediction + (1 – a) \* actual runtime

*(using 30 msec as the “previous prediction” of the oldest runtime)*

2nd = 0.8 \* 30 + 0.2 \* 30 = 30 msec

3rd = 0.8 \* 25 + 0.2 \* 30 = 26 msec

4th = 0.8 \* 55 + 0.2 \* 26 = 49.2 msec

5th = 0.8 \* 35 + 0.2 \* 49.2 = 37.84 msec

The prediction for the next runtime is 37.84 msec