**Computer Systems 2019 – Operating Systems Part**

Copied from the tutorials

**Question 3:**

a)

(a) Share – data, programs, hardware, resource allocation

(b) Concurrency – support multiple parallel activities, switch activities, safe concurrency

(c) Non-determinism – interrupts, input errors, network packet loss

(d) Data storage – persistent storage

b)

Deadlock – each process is blocked waiting for an event that only another process can cause.

Livelock – processes/threads are not blocked, but they or the system as a whole does not make progress.

Necessary conditions – mutual exclusion, hold and wait, no preemption, circular wait.

c)

The answer would depend on X. Shortest job first is the best way to minimise average turnaround time.

0 < X ≤ 3: X, 3, 5, 6, 9

3 < X ≤ 5: 3, X, 5, 6, 9

5 < X ≤ 6: 3, 5, X, 6, 9

6 < X ≤ 9: 3, 5, 6, X, 9

X > 9: 3, 5, 6, 9, X

d)

This depends on whether the system is a uniprocessor or a multiprocessor system. If the system is a multiprocessor system, the implementation using threads is more efficient if each thread could execute on a separate processor. However, if the system is a uniprocessor system, the implementation using threads would be less efficient due to the additional overhead incurred by spawning threads and switching contexts between the threads.

e)T

(a) Disable interrupts

(b) Read value of semaphore (S)

(c) If it is a down operation and S is equal to zero, put the calling process on a list of blocked processes

associated with S

(d) If it is doing an up, check to see if any processes are blocked on S. If one or more processes are

blocked, one of them is removed and made runnable

(e) When all these operations have been completed, enable interrupts

f)

At least twice and at-most thrice.

P0 completes loop, prints exam and waits on S0,

P1 runs and wakes P0 so P0 prints exam

P2 runs and wakes P0 so P0 prints exam i.e. three times.

However, if P2 runs immediately after P1, then S0 will still be 1 so P0 runs only once more and hence

prints exam twice.

**Question 4:**

a)

Device driver – a piece of software that is part of the OS and is responsible for interacting with a

particular device.

Device controller – a piece of hardware that controls the device and implements an interface between the device and rest of the system.

b)

Large block sizes – significant internal fragmentation.

Small block sizes – multiple blocks dispersed through disk, poor performance due to many seeks.

c)

For small files, this would be advantageous since it would give us ‘spatial locality’. Extra disk

seek operations would be reduced since the data can be written to the spare blocks of the inode which

are not being used. This way, we also reduce wastage in the data blocks used to store the inode.

d)

Not in tutorials, so my own answer:

The working set of a process is the set of pages that a process used within a certain time interval. Larger pages may mean that certain memory is combined into fewer pages. Hence the number of pages (the technical definition of the working set) is lower. However, the total memory in the working set is larger as larger pages result in more internally fragmented space.

The working set of a process is the set of pages in the virtual address space of the process that are currently resident in physical memory.

e)

Since we need to use R multiple times , we calculate it apriori.

R = 1/100 = 10 ms

For the first request:

Tseek = 0.1 \* 100 + 5 = 15 ms

Tlatency = 1/2 \* 10 ms = 5 ms

Ttransfer = 1 sector

10 sectors \* 10 ms = 1 ms

Tservice = Tseek + Tlatency + Ttransf er = 21 ms

For the second request:

Tseek = 0. Tlatency and Ttransfer are the same as for the first request. Assumption here is that the delay between requests is unknown, so on average the disk is equally likely to be at any rotational position when the second request arrives.

Tservice = Tseek + Tlatency + Ttransfer = 6 ms

Total = 21 + 6 = 27 ms

f)

(a) Page size = 2size of of f set = 28 = 256 bytes

(b) Process size = 256 KB ⇒ Total number of pages = 256 KB /256 bytes = 2^10

Third-level page table holds total of 2^6 references, so number of third-level pages required = 2^10/

2^6 =2^4

Second-level page table holds total of 2^8 references, so number of second-level pages required = 2^4/2^8 = 1

Similarly, number of outer page-table pages required = 1

Total size = Num of third-level pages \* their size + Num of second-level pages \* their size + Num

of outer page-table pages \* their size

Page table entry = 2 bytes ⇒ Total size = 2^4 \* 2^6 \* 2 + 1 \* 2^8 \* 2 + 1 \* 2^10 \* 2 = 4608 bytes