**Computer Systems 2017 – OS Part**

Copied from tutorials

**Question 3:**

a)

int E, W = 0; // Counts of east and west-bounwd baboons initially = 0

semaphore Emutex, Wmutex, Bridgefree;

// Binary semaphores, so all set to 1

Emutex = 1;

Wmutex = 1;

Bridgefree = 1;

// East-bound baboon

Earrive () {

down (Emutex);

E++;

// First East-bound checks if bridge is free and if not, it blocks.

// Any following East-bound baboons will be held on Emutex.

if E == 1

down (Bridgefree);

up (Emutex);

}

Edepart () {

down (Emutex);

E--;

// If last East-bound baboon, then indicate bridge is free

if E == 0

up (Bridgefree);

up (Emutex);

}

// West-bound baboon

Warrive () {

down (Wmutex);

W++;

// First West-bound checks if bridge is free and if not, it blocks.

// Any following West-bound baboons will be held on Wmutex.

if W == 1

down (Bridgefree);

up (Wmutex);

}

Wdepart () {

down (Wmutex);

W--;

// If last West-bound baboon, then indicate bridge is free

if W == 0

up (Bridgefree);

up (Wmutex);

}

Fairness – No, a solid stream of baboons either east-bound or west-bound can result in starvation from the opposite direction.

b)

The assumption is that with FCFS, once a process is initiated it runs to completion. Such a scheme would prevent the system from guaranteeing good interactive response times. For example, suppose a large batch process enters a uniprocessor system. While that process executes, no other processes can execute. A user that attempts to load a web page or send an instant message must wait until the batch process completes before the system will respond to those requests.

Round-robin is a preemptive scheme that makes use of the interrupt clock. Long processes cannot delay shorter ones, because the shorter ones are assured of getting the processor periodically. Interactive users will thus receive the processor frequently enough to maintain good response times. In our example, the large batch job will be interrupted to service the processes that try to load a web page or send an instant message.

**Question 4:**

a)(own answer)

In paging data, instructions and the stack share an address space, while in segmentation they are separated. This allows for protection of certain memory, e.g. preventing overwriting of the instructions by out-of-bounds memory access.

The process running has no awareness of paging, while it must be aware of segmentation. Pages have a fixed size, while segments are variable. Segmentation can lead to more external fragmentation and segments may need to be moved if their size changes.

b)

(a) Low-level format - Done to create the disk sector layout of the hard drive. Usually done by the manufacturer.

(b) Seek Time - Time taken for the necessary track to appear under the read/write head of the hard drive.

(c) SCAN scheduling - Process requests that result in the shortest seek time in preferred direction. The direction is only changed after reaching the outermost/innermost cylinder (or no further requests in preferred direction).

(d) C-SCAN scheduling - Service requests in one direction only and change jump to outermost request once you have processed the innermost request.

c)

(own addition) For a closed file only the inode location is kept in memory.

Additional information contained - location of the original inode by maintaining the disk device number and inode device number in order that the inode can itself be written back to disk, if it had been modified. Additionally contains information about the number of process that have opened this file.

d)

This does not always work. If an interrupt occurs between the statements InterruptEnabled=true and PC=Mem[0], the value of PC is lost and the system would be in an unstable state. To avoid this

problem, the two statements (restoring PC from Mem[0] and re-enabling interrupts) are made into one special instruction (rti) that is indivisible and hence, un-interruptible.

e)

