**Computer Systems 2018 – Operating Systems Part**

Copied from the tutorials

**Question 3**

Hope a and b don't show up again

c) (my own answer)

i) (18+21+31+36+47)/5 = 30.2min

ii) Order of the length: 3, 5, 8, 10, 11, 18

(3+8+18+29+47)/5 = 21 min

**Question 4**

a)

The operating system must locate the page table for the process that is to start running. It must set the base register in the MMU so that it points to the page table in memory, and it must set the length register if present. Finally, it must clear any now-invalid cached address translations from the TLB.

b) (own answer)

Character-oriented devices transfer one character at a time, e.g. a keyboard or a modem

Block-oriented devices transfer a larger block of data at a time, e.g. ethernet

c)

Not in general. When the disk controller signals “operation complete” after a write to disk, the OS must be able to rely on that page staying written even in the event of an immediate crash (e.g. due to power failure). Reporting “operation complete” before the write to disk is finished violates this rule. A controller can do write-behind if its local memory is battery-backed for long enough and if at restart, it performs the writes that were reported completed before the crash.

d)

a) Essentially the OS has to run a garbage collection algorithm on the files to recover the free block information. First it would reinitialise the free block bitmap. Then, it would start at the root of the file system (the ‘/’ directory), and mark every block used by every file found through a recursive descent through the file system. When finished, it would update the stored free block bitmap. This is essentially what the UNIX utility fsck does to recover from metadata corruption.

(b) There could be multiple copies of the free block bitmap stored at different locations of the disk. In addition, updates to it would have to be atomic.

e)

Maximum size of the virtual address depends on the number of page table entries – can be calculated using the maximum page table size (given) and the size of a page table entry.

Size of a page table entry = valid bit + dirty bit + permission bits + X, where X = bits required to store physical page number

X = total physical address size (32) - size of page offset

Total bits to address a page of size 8 KB (2^13) = 13 ⇒ size of page offset = 13

X = 32 - 13 = 19

Size of page table entry = 1 + 1 + 3 + 19 = 24 bits

Number of virtual pages = page table size/size of page table entry = 96 MB/24 bits = 2^25

Total number of virtual addresses = Number of virtual pages \* Page size = 2^25 ∗ 8 KB = 2^38, which requires 38 bits to address

Therefore, length of virtual address space = 38 bits