Questions for OSC

Q1 location 6.4.1: contention scope p.285:

PCS scheduling is within the process the different user threads onto LWP and thus onto kernel-threads. Is SCS scheduling how the system decides to assign a kernel-level thread to the CPU? In a none one-to-one model then, the PCS would hook up the user-thread to the LWP and thus to the Kernel-thread, to which the SCS would then come in and decide which of these kernel-threads available is going to be executed by the CPU?

Answer:

You are basically correct that in a m-2-m or m-2-1 model, the user level selection of which thread to run is done with PCS. Once one thread per process is identified, then scheduling of the cpu among these is with SCS. In a 1-2-1 model, you don't need the first, PCS, step.
The pthread scheduling system call however allows a user to select PCS or SCS for this first selection step in the m-2-m model.

Q2 location 6.8.2: queueing models p.309:

I don't understand the Queueing models strategy of Algorithmic evaluation. How does it work?

Answer:

Queueing theory is a mathematical model that predicts how several queues placed back to back and/or in parallel with each other will behave. It allows you to ask questions like how long will an element be in a queue, what is the average queue size and so forth. From our perspective in this class, this is as much as you need to know. The other methods are the important ones for us: deterministic modelling using gantt charts and simulation/implementation.

Q3 location 6.8.3: simulations p. 310:

the clock variable in simulation, is it suppose to represent a clock cycle or time? Once we have all the statistics from the simulation would we run deterministic modeling or analytical evaluation to finally determine which is of the simulations achieves our criteria, since it seems that after the simulation all we have is a bunch of data that still needs to be further analyzed?

Answer:

the clock variable in simulation represents the time passing as a process is moved from queue to queue or is executed on the processor. Its just a variable that is increment whenever something happens that would have taken time. You are right that you then would neeed to evaluate some metric on the data from simulation/implementation, same as you would in deterministic modeling.

Q4 location 7.2: Swapping p. 333:

Does the swapped out process need to swap back in to same physical addresses ? It doesn't have to because everything involving the process is mapped through the logical address right? even with pending i/o is going to be looking for the logical address not the physical address?

Answer:

If th ememory mapping supports relocation, as does paging or segementation for example, then no, a page/segemented that is swapped back in does not have to be at the same location.

IO is an issue. Either the OS must buffer the IO so that the information is fed to the new age correctly, or else the page has to be locked into memory until the IO completes.

Q5 location 7.3.2: memory allocation p.337:

would the worst fit strategy cause the least amount of external fragmentation, or take the longest amount of time to have external fragmentation, in the variable-partition method? Since the memory would be allocated from bottom to top, increasing the chance that the processes at the bottom of memory will finish before all the high memory is used up in a linear method?

Answer:

best fit and first fit are typicall better in terms of storage ultization than worst fit, despite the intuition about worst fit.

Q6 location 7.3.2: memory allocation p.336:

Multiple-partition method of contiguous memory allocation, has no problems with external fragmentation correct? But possibly large ones with internal fragmentation?

Answer:

This is an old method and not really that useful. But you are right that it would suffer from internal not external fragmentation.

Q7 location 7.5 paging p.345:

In paging the logical address is given, int division is done to get the page number, and mod is done to get the page offset, the page number is looked up in the page table and the page offset is put in the physical address, does the page table pass the physical address of the frame? Or does it pass the frame number to the frame table and then the frame table passes the address to the physical address?

Answer:

If you think about it, the frame number 'is' the physical address that the frame starts at, once you place it at the rightmost (ie highest) part of the address.

Q8 location 7.5.2 hardware support (paging) p. 348:

Do TLB entries that are wired down survive a flush? Or are they simply never replaced by other entries ?

Answer:

Depends on the exact TLB interface, either option is reasonable.

Q9 location 7.6.2 hashed page tables p.355:

In clustered page tables, each entry refers to several pages, does that mean that each entry has multiple pointers to multiple linked list, and that the system is searching through multiple linked list in parallel and checking to see if the virtual page numbers match?

Answer:

Each entry has multiple pages as a single entry, and a linked list of single entries (which each are multiple pages). So the clusters rather than the pages are the elements of the hash table.

Q10 location 8.4

Will counting based, page buffering and applications replacement be on the exam?

Answer:

The concept of LRU or MFU is one that you should be familiar with and capable of executing on a reference string. But Page buffering and applications/page replacement - no.

Q11 location 8.8.1 buddy system p. 410

Has the buddy system pretty much been discontinued ? It seems that though slab is more complicated it is more effective in pretty much every way? Would buddy system be more effective on a small scale with scarce resources?

Q12 location 10.2.2 direct access p.488

In direct access does the write(n) command append the content of the block or overwrite it? Does the position_file(n) followed by write next append or overwrite the content of the block?

Q13 location 10.3.6 acyclic-graph directories p. 497

I don't really understand the difference between tree-structured directory, acyclic-graph directory, and general graph directory. It seems that they are all essentially a tree-structured directory, that vary slightly with the inclusion of pointers to ease sharing among different subdirectories?

Also I can't seem to really understand what a cycle is, is it simply that when doing a search operation on a acyclic or general graph structure and the system checks the same place twice (since some of the files are connected to multiple directories)?

Q14 location 10.4 file system mounting p. 500

Is the mount point always the root of the file structure to be mounted ? How does the system know where the current file structure to mount the mount point? Is it in the instructions that come with the file-system? Or does the system figure it out?