

# Operating Systems Supervision 1

Neelu Saraswatibhatla (srns2)

## 1 Example Sheet 1

1. (a)
  - i. Cache memory is a small amount of high-performance storage that is faster for the processor to access than main memory. The computer uses it to temporarily store data that the processor can retrieve easily.
  - ii. Main memory is the main RAM in a computer and it is where files that are currently in use by processes are stored.
  - iii. Registers are very fast pieces of memory located on the chip itself. Data is loaded from main memory (or cache memory) onto registers to be operated upon, and then loaded back onto main or cache memory.
- (b) If the size of the data being used by processes in progress exceeds the size of main memory, then some of that data is offloaded into disk storage temporarily.
2. (a)
  - i. An unsigned integer is simply stored as a binary number directly corresponding to the decimal representation of the unsigned integer.
  - ii. A signed integer is generally stored as a two's-complement integer. The most significant bit represents the sign (0 is positive, 1 is negative) and the remaining bits represent the value (but not directly). To convert from  $x$  to  $-x$ , invert every bit (including the MSB, turning the integer from positive to negative) and add 1.
  - iii. Memory can only store binary strings so a text string has to be converted into binary. The two main standards for this are ASCII and Unicode. ASCII is a 7-bit code that holds American letters, numbers, punctuation, and some other characters. Unicode is an 8-, 12-, or 32-bit code that aims to support all characters and symbols in every language and used by anyone.
  - iv. An instruction comprises of an opcode (specifies what operation to perform), zero or more operands (registers where values used in the operations the stored), and the register into which to put the result.
- (b) If it wants to interpret the contents of the register and convert them into a human-readable decimal number, for example, then it would need to know whether or not the integer is unsigned. However if it is simply adding or subtracting two integers then it does not need to know this information as adding or subtracting integers is done using the same method for both signed and unsigned integers, adding or subtracting 1.
- (c) During a context switch, the operating system saves the state of a thread, restores the state of another thread, and also switches process state if the two threads were

in different processes. Then when context switching back into this first thread, the state ('context') of the thread is restored.

### 3. TODO

4. With non-preemptive process scheduling, a scheduling decision is only taken when a running process blocks or terminates, while with preemptive process scheduling, a scheduling decision can be taken when a timer expires or a waiting process unblocks in addition to these.

Non-preemptive scheduling is simpler to implement than preemptive as the OS only needs to make a decision when a process blocks or terminates so does not need to be constantly doing anything, while preemptive is more complex as the OS needs to be watching for waiting processes unblocking and needs to have a timer.

Non-preemptive is less fair than preemptive as a longer running process could stop a lot of other shorter process from running, while preemptive allows more processes to be given running time by temporarily stopping a longer process. For the same reason, preemptive offers better performance than non-preemptive.

Non-preemptive is open to denial-of-service, so a process could just refuse to block or terminate, stopping the computer from doing anything else entirely. Preemptive has a timer so the OS can watch for this.

By virtue of non-preemptive being simpler, it has fewer hardware requirements, while preemptive has more hardware requirements for things such as the timer.

5. (a) Each job is given a priority (multiple jobs can have the same priority), which is an integer, and the smaller the integer the higher the priority. Every time the CPU makes a scheduling decision (on process blocking/termination in non-preemptive scheduling, or one of these two or timer expiry/waiting process unblocking in preemptive scheduling), it selects the job with the highest priority to run. If multiple jobs have the same priority, then ties can be broken using a round robin with time-slicing, allocating quanta to each process with the same highest priority.
- (b)
  - FCFS is a priority scheduling algorithm with jobs that arrived first having higher priority (e.g. priority = time of arrival). This is a static priority scheduling algorithm as the time a job arrived never changes.
  - Shortest job first is a priority scheduling algorithm with jobs of shortest time required having highest priority (e.g. priority = time required). This is a static priority scheduling algorithm as the total time a job requires never changes.
  - Shortest time remaining is a priority scheduling algorithm with jobs of shortest remaining time required having highest priority (e.g. priority = remaining time required). This is a dynamic priority scheduling algorithm as the time remaining of a job can change, e.g. when the process blocks and then unblocks or when a new job that requires shorter time is created.

- Round robin is a dynamic priority scheduling algorithm, with the job currently running having highest priority, and then as soon as its quantum expires, giving it lowest priority and increasing every other job's priority evenly.
- (c) The biggest problem with static priority scheduling is starvation, as a low priority process is not guaranteed to ever run. A solution is to use a dynamic priority scheduling algorithm, where the priority of a process increases after it has starved for a given amount of time.
- (d) Many CPU scheduling algorithms try to favour IO intensive jobs as they block more often waiting for IO, which allows more jobs to run.
6. (a) It is being used on a time-sharing system, with many jobs needing to be done.
- (b) Since the algorithm gives each job a certain amount of time on a rotating basis, all jobs are equally prioritised (based on priorities) rather than longer ones stopping other ones from being done, so one user running a long process won't stop the other users from getting anything done.

- (c)
- $P_1 : 15 - 0 = 15$
  - $P_2 : 10 - 1 = 9$
  - $P_3 : 12 - 7 = 5$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P1 (9)	P2 (4)							P3 (2)								
	P1(6)			P2(1)			P1(3)			P2(0)	P3(0)	P1(0)				

- (d) An advantage of using a small quantum is that it is more fair, giving more jobs the chance to run and allowing newer jobs to start running sooner. A disadvantage, however, is that context switching is not instant, so every time the current process is changed to a different one (and therefore context switching occurs), the CPU wastes time not doing any useful work.

7. TODO

## 2 Tripod Question 2012/II/3

- (a) (i) This mechanism is called context switching, and it involves storing the state of a thread and restoring the state of a different thread (and also process state if the threads are of different processes).
- (ii) TODO
- (iii) Two other elements of the process context are the program counter and page tables.
- (b) TODO

### 3 Tripod Question 2010/II/3b

(i)

Time	0	5	10	15	20	25	30	35	40	45	50
Job created (Time required)	P1 (25)	P2 (15)	P3 (5)	P4 (5)							
Job running		P1					P3	P4	P2		

$$\text{average waiting time} = \frac{(25 - 25) + (50 - 20) + (30 - 15) + (35 - 20)}{4} = 15$$

(ii)

Time	0	5	10	15	20	25	30	35	40	45	50
Job created (Time required)	P1 (25)	P2 (15)	P3 (5)	P4 (5)							
Job running (Time Remaining)		P1 (20)	P2 (10)	P3 (0)	P4 (0)	P2 (0)		P1 (20)			

$$\text{average waiting time} = \frac{(50 - 25) + (30 - 20) + (15 - 15) + (20 - 20)}{4} = 8.75$$

(iii)

Time	0	5	10	15	20	25	30	35	40	45	50
Job created (Time required)	P1 (25)	P2 (15)	P3 (5)	P4 (5)							
Job running (Time Remaining)		P1 (15)		P2 (5)		P3 (0)	P4 (0)	P1 (5)		P2 (0)	P1 (0)

$$\text{average waiting time} = \frac{(50 - 25) + (45 - 20) + (25 - 15) + (30 - 20)}{4} = 17.5$$

- (iv) (As above in Example Sheet 1 question 6d) An advantage of using a small quantum is that it is more fair, giving more jobs the chance to run and allowing newer jobs to start running sooner. A disadvantage, however, is that context switching is not instant, so every time the current process is changed to a different one (and therefore context switching occurs), the CPU wastes time not doing any useful work.