IT5002 Tutorial 7

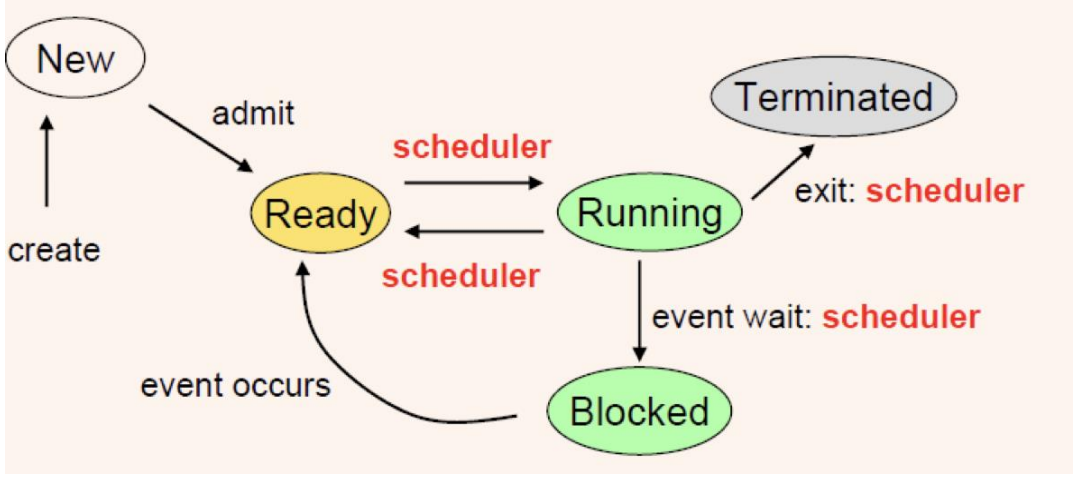
AY 2025/26 Semester 1

Prepared by Michael Yang

Slides adapted from Theodore Leebrant, Prof. Colin and Prof. Aaron

2. Consider a process P1. What state would P1 be in, in the following circumstances?

Situation	P1's State
The OS has just finished creating P1	Ready
P1 is pre-empted by another process	Ready
P1 is waiting for I/O	Blocked
P1's I/O has just completed	Ready
P1 is currently being executed	Running
P1 is suspended for 2 seconds	Blocked



(From Lecture 13 – Scheduling, Slide 4)

3. Suppose we have a **fixed-priority** operating system, where each process is given a priority, and when a higher priority process is ready, the OS pre-empts the current process and passes CPU control to the higher priority process.

a. If there are 255 possible priority levels, how many ways can you assign priorities to n processes, if no two processes can have the same priority level?

Out of 255 priority levels, choose n (assuming n <= 255)

For each of n processes, they can be permuted (assigned priority levels) in n! ways

Total number of ways:

$$\binom{255}{n} \cdot n! = \frac{255!}{n! \cdot (255 - n!)} \cdot n! = \frac{255!}{(255 - n)!} = P(255, n)$$

3b. A *periodic process* is one that runs for C_i CPU cycles every P_i cycles.

Rate Monotonic Scheduling (RMS): a process with a shorter period is assigned a higher priority than a process with a longer period

Suppose we sort our processes in ascending order of periods (T1 with shortest period i.e. highest priority).

For each task T_i, recursively compute $S_{i,x}$ until $S_{i,(x+1)} = S_{i,x}$. Call this $S_{i,F}$.

Prove that $S_{i,F}$ is the worst-case response time for a process T_i. The response time is the time between the first time a process runs, and when it finishes running

$$S_{i,0} = \sum_{j=1}^{i} C_j \qquad S_{i,(x+1)} = C_i + \sum_{j=1}^{i-1} C_j \times \left\lceil \frac{S_{i,x}}{P_j} \right\rceil$$

RMS is just another type of scheduler. (See https://www.youtube.com/watch?v=xgW8VhEOpFg for an example of how it works)

Rationale for assigning shorter period processes with higher priority:

- Shorter period processes have a tighter 'deadline' - they must complete their execution within the given window, so they are allocated higher priority so that they can pre-empt others
- Longer period processes have more 'leeway' and a larger window to complete their execution, so they are given lower priority so that they can yield

Think of $S_{i,0}$ as the total number of CPU cycles if all processes from 1 to *i* executed completely without any pre-emption of lower priority processes by higher priority processes

For some process *i*, it can be pre-empted by a higher priority process *j*.

The number of times this happens is given by $\left\lceil \frac{S_{i,x}}{P_j} \right\rceil$

Process j runs for C_j cycles, so it adds an additional $C_j \cdot \left\lceil \frac{S_{i,x}}{P_j} \right\rceil$ cycles

Under the worst case conditions, **every** process j with a higher priority than process i pre-empts process i exactly $\left\lceil \frac{S_{i,x}}{P_j} \right\rceil$ times

So the total contribution is $\sum_{j=1}^{i-1} C_j \cdot \left\lceil \frac{S_{i,x}}{P_j} \right\rceil$

Adding the time taken to execute T_i itself, the worst case time taken is $C_i + \sum_{j=1}^{i-1} C_j \cdot \left\lceil \frac{S_{i,x}}{P_j} \right\rceil$

Essentially, the algorithm calculates $S_{i,x}$ until it becomes 'stable' – deriving the worst possible time for process T_i to finish executing under the conditions that it can be pre-empted by higher priority processes

4. A system has one CPU and one I/O device. There are 3 processes P1, P2 and P3 where P1 has the highest priority and P3 the lowest. All 3 processes are ready to run at time 0.

When the I/O device is busy the next process will wait for it to be free before using it.

If two processes are waiting for I/O, the higher priority process uses it first.

There is no pre-emption for I/O.

Process	Execution Profile
P1	C2-IO3-C3-IO1-C2-IO3-C3
P2	C1-IO2-C1
P3	C3

a. Show the schedule for the three processes.

Cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CPU	P1	P1	P2	P3	P3	P1	P1	P1	P2	P1	P1	P3			P1	P1	P1
I/O			P1	P1	P1	P2	P2		P1			P1	P1	P1			

b. What is the response time for each process?

P1: 17 cycles
P2: 9 cycles
P3: 12 cycles

c. What is the CPU utilization to 2 decimal places?

Utilization = 15/17 x 100% = 88.24%

Slides uploaded to https://github.com/michaelyql/IT5002

Email: e1121035@u.nus.edu

Anonymous feedback: https://bit.ly/feedback-michael

(or scan the QR below)