EECE.4810/EECE.5730: Operating Systems

Spring 2017

Homework 3 Solution

1. *(16 points) Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:*

Process	*Burst*	*Priority*
P1	*20*	*4*
P2	*5*	*3*
P3	*30*	*2*
P4	*2*	*3*
P5	*5*	*1*

a. (12 points) Assume the processes arrived in the order P1, P2, P3, P4, P5, all at time 0.

*What is the turnaround time (i.e., time of completion) of each process for each of the following four scheduling algorithms: FCFS (First Come First Serve), Round Robin (quantum=1), SJF (Shortest Job First), and a non-preemptive priority (a smaller priority number implies a higher priority)***?**

__Solution:__ The solutions are shown below, with each column representing a scheduling metric, and the start/end times for each process shown. The turnaround time for each is the end time, shown in bold, since all processes arrive at time 0. Remember that, as with the example in class, a process with a burst time of 1 starts and ends in the same cycle. You may have used slightly different notation.

In most cases, processes run to completion once they have started, with round robin being the only preemptive scheme shown in this problem. In round robin, all 5 processes alternate, with P4 finishing after two cycles (time 9), P2 (time 20) and P5 (time 22) finishing after 5 cycles, P1 finishing after 20 cycles (time 51), and P3 finishing after 30 cycles (time 62).

Process	**FCFS**	**RR**	**SJF**	**Priority**
P1	1 → **20**	1 → **51**	13 → **32**	43 → **62**
P2	21 → **25**	2 → **20**	3 → **7**	36 → **40**
P3	26 → **55**	3 → **62**	33 → **62**	6 → **35**
P4	56 → **57**	4 → **9**	1 → **2**	41 → **42**
P5	58 → **62**	5 → **22**	8 → **12**	1 → **5**

b. (4 points) For each of the four scheduling algorithms listed above, state whether your answer would have changed if each process arrived 1 millisecond apart (P1 at time 0, P2 at time 1, etc.) and briefly explain why. (Note: You do <u>not</u> have to determine the turnaround time for each process with varying arrival times; you simply have to explain if your answer changes and why.)

<u>Solution:</u> With varying arrival times, P1 always runs first, since there are no other jobs to choose from at time 1. Therefore, your answer will change for those metrics that would have run something else first, namely <u>shortest job first (SJF) and priority scheduling.</u> Scheduling metrics that ran P1 first even when all five processes were available do not change, so <u>FCFS and round robin</u> give the same results.

2. *(9 points) Assume you have a multi-programmed system managing main memory using a base and bounds scheme. You have the following lists of holes and address space requests, each of which exists in the order shown:*

Available holes: *500 KB, 200 KB, 350 KB, 750 KB, 125 KB*
Address space requests: *300 KB, 175 KB, 400 KB, 200 KB, 95 KB*

How would these address spaces be placed using (a) first fit, (b) best fit, and (c) worst fit allocation? As part of your solution to each part, show the list of remaining holes after all address spaces are placed.

Solution: First, the solution for (a) first fit:

Space request	Hole used	Remaining holes (modified hole in red)
300 KB	500 KB	200 KB, 200 KB, 350 KB, 750 KB, 125 KB
175 KB	200 KB	25 KB, 200 KB, 350 KB, 750 KB, 125 KB
400 KB	750 KB	25 KB, 200 KB, 350 KB, 350 KB, 125 KB
200 KB	200 KB	25 KB, 350 KB, 350 KB, 125 KB (200 KB hole completely filled)
95 KB	350 KB	25 KB, 255 KB, 350 KB, 125 KB

Next, the solution for (b) best fit:

Space request	Hole used	Remaining holes (modified hole in red)
300 KB	350 KB	500 KB, 200 KB, 50 KB, 750 KB, 125 KB
175 KB	200 KB	500 KB, 25 KB, 50 KB, 750 KB, 125 KB
400 KB	500 KB	100 KB, 25 KB, 50 KB, 750 KB, 125 KB
200 KB	750 KB	100 KB, 25 KB, 50 KB, 550 KB, 125 KB
95 KB	100 KB	5 KB, 25 KB, 50 KB, 550 KB, 125 KB

Finally, the solution for (c) worst fit:

Space request	Hole used	Remaining holes (modified hole in red)
300 KB	750 KB	500 KB, 200 KB, 350 KB, 450 KB, 125 KB
175 KB	500 KB	325 KB, 200 KB, 50 KB, 450 KB, 125 KB
400 KB	450 KB	325 KB, 25 KB, 50 KB, 50 KB, 125 KB
200 KB	325 KB	125 KB, 25 KB, 50 KB, 50 KB, 125 KB
95 KB	125 KB	30 KB, 25 KB, 50 KB, 550 KB, 125 KB

3. *(10 points) On a system using segmentation to manage main memory, the segment table for the currently running process is listed below:*

Segment #	*V*	*Base*	*Bounds*	*Access*
0	*1*	*1020*	*500*	*read*
1	*0*	*74*	*12*	*read/write*
2	*1*	*19*	*78*	*read/write*
3	*1*	*4810*	*5730*	*read*
4	*1*	*3220*	*2160*	*read/exec*

For each of the given memory accesses, determine if (i) the access is valid, and (ii) what the physical address is for each valid access. If the access is invalid, briefly explain why.

Solution notes: Remember that:

- Accesses can be invalid for one of three reasons: the segment is not in physical memory (V = 0), the offset is greater than the bound for that segment, or the process does not have the necessary access rights (for example, writing a read-only segment)
- For a valid access, the physical address is simply the segment base address + offset from the virtual address

a. *Read from virtual address 2, 16*

Solution: (i) Access is valid, (ii) physical address is 16 + 19 = 35

b. *Write to virtual address 1, 10*

Solution: (i) Access is invalid—segment is not in physical memory (V = 0)

c. *Write to virtual address 3, 5000*

Solution: (i) Access is invalid—process only has read permission to segment 3

d. *Read from virtual address 0, 600*

Solution: (i) Access is invalid—offset is greater than bound for segment 0 (600 > 500)

e. *Read from virtual address 4, 2000*

Solution: (i) Access is valid, (ii) physical address = 4 + 2000 = 2004

4. *(15 points) On a system using paging to manage main memory, the currently running process uses the page table below:*

Virtual page #	Valid bit	Reference bit	Dirty bit	Frame #
0	1	0	0	5
1	1	0	1	0
2	0	0	0	--
3	1	1	0	3
4	0	0	0	--
5	1	1	1	2
6	0	0	0	--
7	1	1	0	1

a. *(5 points) Which pages above are candidates to be evicted on a page fault? Which, if any, are better candidates to evict?*

Solution: Any valid page with reference bit = 0 is a candidate for eviction—pages 0 and 1. Page 0 is arguably a better candidate, as it is not dirty and therefore would not require a write to disk.

b. *(10 points) Assuming 8 KB pages, what physical addresses would the virtual addresses below map to? Note that some virtual addresses may not have a valid translation, in which case you should note that address causes a page fault.*

Solution notes: Note that 8 KB = 2^{13} byte pages imply a 13-bit offset, meaning the page number is the upper 3 bits of the virtual address. In each solution below, the page number is underlined, as is the frame number that replaces it in the physical address.

- *0xABCD*

Solution: 0xABCD = 1010 1011 1100 1101 → page #5, which maps to frame #2
→ physical address = 0100 1011 1100 1101 = 0x4BCD

- *0x1792*

Solution: 0x1792 = 0001 0111 1001 0010 → page # 0, which maps to frame #5
→ physical address = 1011 0111 1001 0010 = 0xB792

- *0x4680*

Solution: 0x4680 = 0100 0110 1000 0000 → page #2, which is invalid → no physical address

- *0x5701*

Solution: 0x5701 = 0101 0111 0000 0001 → page #2, which is invalid → no physical address