Eddie Gurnee
COSC 592
Operating Systems

Chapter 7 & 8 Reading Questions:

Chapter 7

Exercise 1: Consider the situation of cars attempting to use a single lane bridge (see the diagram below).

1. Show that the four required conditions for deadlock hold in this example.
2. State a simple rule that could preclude deadlocks in this system.

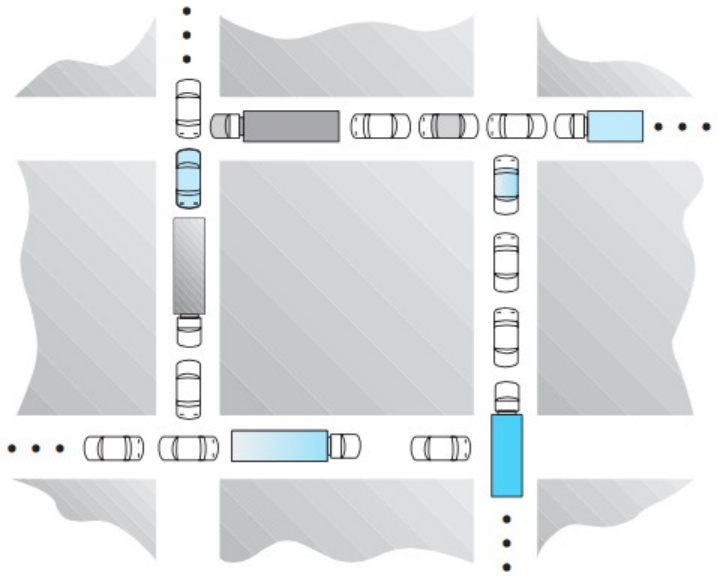

Figure 7.10 Traffic deadlock for Exercise 7.11.

Response:

1. Required conditions hold ->
   a) mutual exclusion: no position in traffic can hold multiple vehicles
   b) hold-and-wait: when a vehicle 'holds' its position in traffic while it 'waits' for other vehicles to get out of its way
   c) no preemption: barring ufos and teleport without error, cars cannot be simply removed from their location in traffic
   d) circular wait: a vehicle can only advance if the vehicle in front of it has advanced
2. A rule that could preclude deadlocks: no vehicle can enter the intersection if it is obvious that they could not immediately safely exit the intersection.

Exercise 2: The Java API for the Thread class contains a method destroy that has been deprecated. Consulting the API (you can find it), explain why the method was deprecated. Provide an example where the use of destroy might possibly lead to a deadlock.

Response: Firstly, Thread.destroy() was never actually implemented. The reason why it was never implemented is because the specifications for it were prone to causing deadlocks: if a Thread holds a lock on the monitor protecting a critical system resource is destroyed, no Thread can access that

resource. If a Thread tries to lock that monitor, deadlocks galore.

Exercise 3: Consider the following snapshot of a system:

	Allocation				Max				Available			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	1	2	3	4	3	5	6	6	3	1	2	3
P1	3	1	2	1	5	2	5	2				
P2	2	3	0	1	2	6	1	3				
P3	1	2	1	3	1	4	2	4				
P4	2	1	0	0	4	2	1	2				

Answer the following questions using the banker's algorithm:

1. Illustrate that the system is in a safe state by demonstrating an order in which the processes may complete.
2. If a request from process P1 arrives for (1, 0, 0, 1), can the request be granted immediately?
3. If a request from process P0 arrives for (0, 0, 2, 0), can the request be granted immediately?

Response:

	Need			
	A	**B**	**C**	**D**
P0	2	3	3	2
P1	2	1	3	1
P2	0	3	1	2
P3	0	2	1	1
P4	2	1	1	2

1. The sequence <P4, P3, P0, P1, P2> satisfies the safety condition.
   a) P4 → 5, 2, 2, 3
   b) P3 → 6, 4, 3, 6
   c) P0 → 7, 6, 6, 10
   d) P1 → 10, 7, 8, 11
   e) P2 → 12, 10, 8, 12
2. Yes. The resources are immediately available, and the system would still be in a safe state as the sequence <P4, P3, P0, P1, P2> would still satisfy the safety condition.
3. Negative. While the resources are immediately available, allocating those resources would result in an unsafe state: while P4 could still run, P3 would be waiting on access to non-available resource C.

Chapter 8

Exercise 4: Consider a logical address space of 256 pages of 4,096 words each, mapped onto a physical memory of 128 frames.

(a) How many bits are there in a logical address?
(b) How many bits are there in a physical address?

Response:

pages = 256 = 2 ^ 8
word/page = 4_096 = 2 ^ 12
frames = 128 = 2 ^ 7

1. Pages * Words/Page = 2 ^ 8 * 2 ^ 12 = 2 ^ 20 => 20 bits in a logical address
2. Frames * Words/Page = 2 ^ 7 * 2 ^ 12 = 2 ^ 19 = > 19 bits in a physical address

Exercise 5: Given six memory partitions of 200 KB, 600 KB, 350 KB, 300 KB, 750 KB, and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 358 KB, 500 KB, 150 KB, 125 KB, and 375 KB (in order)? Rank the algorithms in terms of how efficient their memory use is in this case.

Response:

Memory Partition	Memory Size		Process Name	Process Size
M0	200KB		P0	358KB
M1	600KB		P1	500KB
M2	350KB		P2	150KB
M3	300KB		P3	125KB
M4	750KB		P4	375KB
M5	125KB			

1. First-fit
   a) P0 → M1 (242KB remaining)
   b) P1 → M4
   c) (250KB remaining)
   d) P2 → M0 (50KB remaining)
   e) P3 → M1 (117KB remaining)
   f) P4 → wait for free memory
2. Best-fit
   a) P0 → M1 (242KB remaining)
   b) P1 → M4 (250KB remaining)
   c) P2 → M0 (50KB remaining)
   d) P3 → M5 (0KB remaining)
   e) P4 → wait for free memory
3. Worst-fit
   a) P0 → M4 (392KB remaining)
   b) P1 → M1 (100KB remaining)
   c) P2 → M4 (242KB remaining)
   d) P3 → M2 (225KB remaining)
   e) P4 → wait for free memory

After Algorithms run, KB used (percent):

	M0	M1	M2	M3	M4	M5
First	150/200 (.75)	483/600 (.805)	0/350 (0.0)	0/300 (0.0)	500/750 (.6667)	0/125 (0.0)
Best	150/200 (.75)	358/600 (.5967)	0/350 (0.0)	0/300 (0.0)	500/750 (.6667)	125/125 (1.0)
Worst	0/200 (0.0)	500/600 (.8333)	125/350 (.357)	0/300 (0.0)	508/750 (.6773)	0/125 (0.0)

Rank the algorithms in terms of efficiency of memory use:

Efficiency Model 1: as a percent of partitions used
Efficiency Model 2: largest remaining partition size
Efficiency Model 3: average partition use
Efficiency Model 4: average free partition size

	Partitions Used	Efficiency Model 1	Efficiency Model 2	Efficiency Model 3	Efficiency Model 4
First-fit	3	73.10%	350KB	74.06%	198.67
Best-fit	4	67.64%	350KB	75.34%	238.4
Worst-fit	3	66.65%	300KB	62.25%	198.67

This is a textbook case of too much work for little gain, the only measurement of efficiency needed here is model 3, which is the average percent use of the 6 partitions given a particular algorithm. For this set of data, Best-fit is the most efficient at memory use, followed by first-fit, then worst-fit.

Exercise 6: Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

1. 4123
2. 28224
3. 206928
4. 516000

Response:

ASIDE: Assuming 32-bit address size, because that's what it probably is; however with the largest being 516_000, all of these could fit into a 19-bit architecture. It just doesn't look or sound as cool.

Page = 1KB = 1024 B = 2 ^ 10 B => 10 bits in offset

$Address_{10}$	$Address_2$	$Page\ Number_2$	$Offset_2$	$Page\ Number_{10}$	$Offset_{10}$
4123	0000 0000 0000 0000 0001 0000 0001 1011	00 0000 0000 0000 0000 0100	00 0001 1011	4	27
28224	0000 0000 0000 0000 0110 1110 0100 0000	00 0000 0000 0000 0001 1011	10 0100 0000	27	576
206928	0000 0000 0000 0011 0010 1000 0101 0000	00 0000 0000 0000 1100 1010	00 0101 0000	202	80
516000	0000 0000 0000 0111 1101 1111 1010 0000	00 0000 0000 0001 1111 0111	11 1010 0000	503	928

1. 4123 →
   Page: 4
   Offset: 27
2. 28_224 →
   Page: 27
   Offset: 576
3. 206_928 →
   Page: 202
   Offset: 80
4. 516_000 →
   Page: 503
   Offset: 928