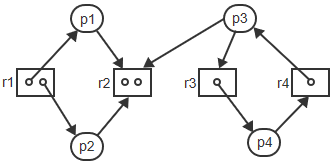
Jeremiah Webb

CS 420

Deadlock and Memory Management Exercises

5.1.1



(a)

Which processes are blocked?

(b)

Which processes are deadlocked?

5.1.2

The state transition diagram in the Participation Activity titled "A deadlock state in a state transition graph" represents the behavior of two processes sharing two resources.

(a)

Modify the diagram such that each process acquires the second resource without releasing the first and then releases both at the same time.

(b)

Modify the diagram such that process p1 only needs resource r2.

(c)

Does either modification eliminate the possibility of deadlock?

5.1.3

|  |  |
| --- | --- |
| p1:  while(1) {  request r1  request r2  release r1  release r2  ... } | p2:  while(1) {  request r2  request r1  release r2  release r1  ... } |

(a)

Draw a state transition diagram to model the above interactions.

(b)

Draw the resource graphs corresponding to states s10, s11, s12, s13, s14, s15, s23, s33.

5.1.4

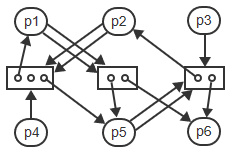
(a)

Modify the previous problem such that p2 requests and releases the resources in the same order as p1. Draw the modified diagram.

(b)

In which states are the processes blocked or deadlocked?

5.2.1



(a)

Using graph reduction, determine if the graph contains a deadlock.

5.2.2

Processes p1, p2, and p3 are executing concurrently. The variables x, y, and z are binary semaphores, all initialized to 1. The left arrow points to the currently executing instruction.

|  |  |  |
| --- | --- | --- |
| p1:  ...  P(x)  ...  P(z) <--  ... | p2:  ...  P(y) <--  ... | p3:  ...  P(z)  ...  P(x) <--  ... |

(a)

To determine if the state is a deadlock state, draw a resource allocation graph by interpreting each semaphore as a resource containing 1 unit and each P operations as request for the resource.

(b)

Reduce the graph to determine if the state contains a deadlock.

(c)

Would initializing any of the semaphores to a value greater than 1 avoid the deadlock?

5.2.3

The wait-for graph of a single-unit resource system contains the edges:  
p1 → p2, p2 → p3, p3 → p4, p5 → p4, p4 → p2

(a)

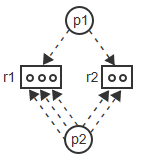
Does the graph represent a deadlock state?

5.2.4

(a)

Show by a counterexample that a cycle in a multi-unit resource graph is not a sufficient condition for a deadlock.

5.3.1



(a)

Show a sequence of operations leading from the given state to a deadlock state.

(b)

Show how the banker's algorithm would have prevented the deadlock.

5.3.2

A system has 2 processes, p1 and p2, and a resource R with 100 identical units. The numbers of maximum claim edges and current allocation edges are shown in the table:

|  |  |  |
| --- | --- | --- |
| Process | Claim edges | Allocation edges |
| p1 | 60 | 30 |
| p2 | 50 | 35 |

(a)

Determine if the current state contains a deadlock.

(b)

Two new processes, p3 and p4, arrive with the following maximum claims and request edges:

|  |  |  |
| --- | --- | --- |
| Process | Claim edges | Request edges |
| p3 | 45 | 15 |
| p4 | 40 | 25 |

Determine if either request can be granted without incurring the possibility of a deadlock.

5.3.3

A system has 2 processes and a resource with 3 identical units. The maximum claim of each process is 2.

(a)

Explain why the system is deadlock-free for all possible states.

5.3.4

A system has n processes and a resource with m identical units. The maximum claim of any process is k.

(a)

Explain why the system is deadlock-free for all possible states if m > n(k - 1).

5.4.1

To prevent inconsistencies when transferring money between two bank accounts, Ai and Aj, the system needs to lock both accounts prior to performing the transfer, then unlock both accounts after the transfer:  
lock(Ai) and lock(Aj) in either order; update Ai and update Aj in either order; unlock Ai and unlock Aj in either order.

(a)

Show how a deadlock can occur when multiple transactions are performed concurrently.

(b)

How could an ordered resource policy be implemented to prevent deadlock?

5.4.2

Process p1 needs to access resources r1, r2, r3, in some order.  
Process p2 needs to access resources r2, r3, r4, in some order.

(a)

Determine which of the following access sequences by p1 and p2 would violate an ordered resources policy and could lead to a deadlock:

|  |  |  |
| --- | --- | --- |
|  | Accesses by p1 | Accesses by p2 |
| 1 | r1, r2, r3 | r2, r3, r4 |
| 2 | r3, r2, r1 | r3, r4, r2 |
| 3 | r2, r1, r3 | r3, r2, r4 |
| 4 | r1, r2, r3 | r4, r2, r3 |
| 5 | r3, r1, r2 | r3, r4, r2 |
| 6 | r1, r3, r2 | r2, r3, r4 |

6.1.1

The animation in this section entitled "Static vs dynamic relocation" shows a load module in memory under static and dynamic relocation.  
An additional function occupying 50 memory words is to be linked in front of function f.  
The load module is then loaded into memory starting at location 1500.

(a)

Show how the addresses change in the load module

1. prior to loading
2. after loading under static relocation
3. after loading under dynamic relocation

6.1.2

(a)

A hole of size m is used to satisfy a request for n bytes of memory, where m > n. The holes are organized as a linked list. Is allocating the last n bytes of the hole better than allocating the first n bytes? Why or why not?

6.1.3

(a)

Memory contains m holes. To satisfy a request for 1 byte of memory, how many holes need to be visited under the different allocation strategies of first-fit, next-fit, best-fit, and worst-fit?

6.1.4

Memory contains 3 holes of 10 MB each. A sequence of 14 requests for 1 MB each is to be processed.

(a)

For each of the four memory allocation strategies, determine the sizes of the remaining holes after all 14 requests have been satisfied.

6.1.5

Memory contains 6 holes with the sizes: 190, 550, 220, 420, 650, and 110.  
A sequence of requests for 4 block is to be satisfied: A = 210, B = 430, C = 100, and D = 420.

(a)

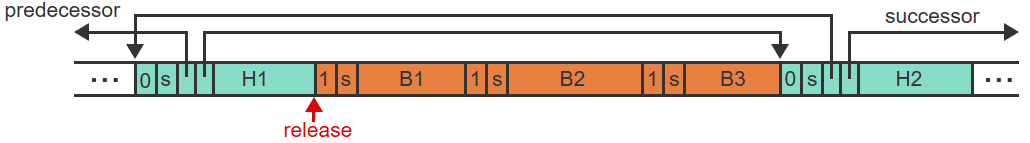
Determine which holes are allocated to which request by each of the 4 allocation schemes.

6.1.6

Main memory consists of allocated blocks and holes of various sizes.

* Each allocated block starts with a 1, followed by the block size, s.
* Each hole starts with a 0, followed by the hole size, s.
* Holes are linked together using a doubly linked list. Since space in the holes is unused, the links are kept inside the holes.

Example:



* The memory contains 3 contiguous blocks (B1, B2, B3) of different sizes.
* H1 and H2 are holes linked together with successor and predecessor pointers.
* Block B1 is to be released

(a)

When a block is released, the memory manager must check if the left and/or the right neighbor is a hole, in which case the newly created hole must be coalesced with the existing hole.

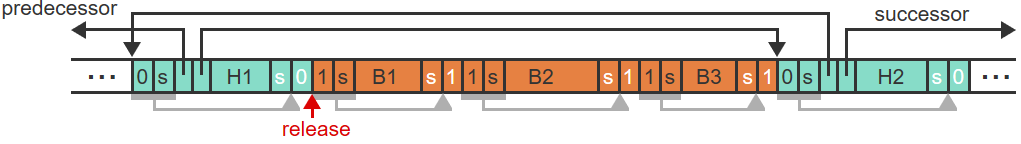
Describe the steps the memory manager must take to check if the right neighbor of block B1 is a hole.

(b)

Describe the steps the memory manager must take to check if the left neighbor of block B1 is a hole.

(c)

The checking of the left neighbor can significantly be simplified if the sizes and the 0/1 tags are replicated at the end of each occupied block and each hole:



Describe the steps the memory manager must take to check if the left neighbor of block B1 is a hole.

(d)

Why must the size fields be replicated at the end of each hole/block and not just the 0/1 tag?

6.2.1

Memory size is 18 MB. Hole size = block size = 1 KB. The 50% rule holds.

(a)

Determine the total number of holes.

(b)

Determine the total number of occupied blocks.

(c)

Determine the amount of space occupied by holes.

6.2.2

Hole size = block size. The 50% rule holds.

(a)

If half of all occupied blocks are released, how will the number and the average size of holes change?

6.2.3

The 50% rule refers only to the number of holes and blocks, but not the amounts of memory space taken up by the holes and blocks. The amounts of space depend on the average hole size vs the average block size.  
If k is the ratio between average hole size and average block size, then the fraction f of space occupied by holes can be determined using the formula f = k/(k + 2).

(a)

Determine the fraction of space wasted in holes if, on average, an occupied block is twice as large as a hole.

(b)

Determine the fraction of space wasted in holes if, on average, a hole is twice as large as an occupied block.

6.2.4

Memory size is 256 MB. Hole size = block size = 1 KB. The 50% rule holds. Reading or writing one 32-bit word takes 10 ns.

(a)

Assuming that all occupied blocks need to be moved, how long will it take to compact the memory?

6.2.5

A total of 4.2 MB of main memory is occupied by 3 blocks, X, Y, and Z:

|  |  |  |
| --- | --- | --- |
| Block | Starting address | Size |
| X | 1,000,000 | 1 MB |
| Y | 2,900,000 | 0.5 MB |
| Z | 3,400,000 | 0.8 MB |

The system uses first-fit allocation. Whenever a request fails, memory is compacted using one of two possible schemes:

1. Starting with the lowest-address block, all blocks are shifted toward address zero.
2. Starting with the lowest-address block, blocks are shifted toward address zero until a hole big enough for the current request has been created.

(a)

Show the memory layout after 3 new blocks, A, B, and C, have been allocated, with the respective sizes of 0.5 MB, 1.2 MB, and 0.2 MB:

(b)

How many bytes of memory were moved under each scheme?

6.2.6

When a memory request fails, a larger hole may be created by:

* Compaction, which involves moving one or more blocks within main memory.
* Swapping, which involves moving one or more blocks to the disk.

(a)

What are the main advantages and drawbacks of each approach?

6.2.7

Define a measure of memory utilization as n \* t where n is area (MB) and t is time (seconds). Processes p and q execute concurrently for m seconds. Halfway through execution, both processes may call a shared function f. The time to execute f is negligible compared to m. The size of each process without f is k. The size of f is also k.

(a)

Determine the space-time requirement for both processes in each case:

1. Static linking and no sharing.
2. Static linking and sharing of f.
3. Dynamic linking and sharing when neither processes calls f.
4. Dynamic linking and sharing when 1 process calls f.
5. Dynamic linking and sharing when both processes call f.

6.3.1

A logical address space consisting of 4 pages of 4096 words each and is mapped onto a physical memory of 64 frames.

(a)

Determine the size of the logical address.

(b)

Determine the size of the physical address.

6.3.2

Paging systems A through D use different combinations of page sizes and address sizes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| Page size (# of words) | 512 | 1024 | 512 | 1024 |
| Logical address size (# of bits) | 16 | 16 | 32 | 32 |

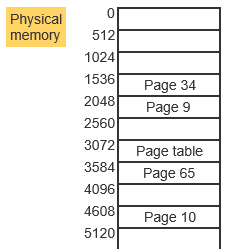
(a)

For each system determine:

* The page table size (# of pages).
* The size of the logical address space (# of words).

6.3.3

The physical memory of a paged system is occupied as follows:



Next, page 49 is loaded at location 0 and page 34 is replaced by page 12.

(a)

Show the contents of the page table before and after the changes.

(b)

Translate the logical address to physical addresses: 4608, 5119, 5120, 33300

6.3.4

Assume a page size of 512.

(a)

Separate the same logical addresses of the previous exercise (4608, 5119, 5120, 33300) into a page number p and an offset w, but following a different approach:

1. Translate the given logical address into an octal number.
2. Interpret the least significant 3 digits as the offset w and remaining digits as the page number p.

(b)

Would the same approach work for any page size? Why or why not?

6.4.1

A memory system employs both paging and segmentation:

* The logical address size is 32 bits.
* Page size is 512 words.
* The segment table contains 2¹³ entries.

(a)

What is the size of w?

(b)

What is the maximum number of pages per segment?

6.4.2

A system is using segmentation without paging.

* The segment table of the currently executing process has the following entries:

|  |  |  |
| --- | --- | --- |
|  | Size | Address |
| 0 | 854 | 1251 |
| 1 | 140 | 0 |
| 2 | 240 | 810 |
| 3 | 100 | 145 |
| 4 | 400 | 2601 |
| 5 | 15 | 380 |

* The process is currently executing in segment 2.
* References to segments other than the current segment are logical addresses of the form (s, w).
* Self-references are of the form (BR, w), where BR is a base register and w is an offset. The register BR is loaded with the starting address of the current segment 2, and w is added to complete the physical address.

(a)

Translate each of the following logical addresses into the corresponding physical address or indicate that the address is illegal:

(3, 0), (4, 250), (5, 20), (6, 10), (BR, 10), (BR, 245)

6.4.3

Three functions, each of length 600 words, are linked together into one process and loaded into memory. Consider four possible combinations of paging and segmentation:

1. Paging (no segmentation):  
   page size: 1024 words  
   page table occupies 1 page
2. Segmentation (no paging):  
   segment table size: 1024 words
3. Segmentation with paging (each function becomes a separate segment):  
   page and segment size: 1024  
   page and segment tables occupy 1 page each
4. Two-level paging (page table is paged):  
   page size: 1024  
   all page tables occupy 1 page each

(a)

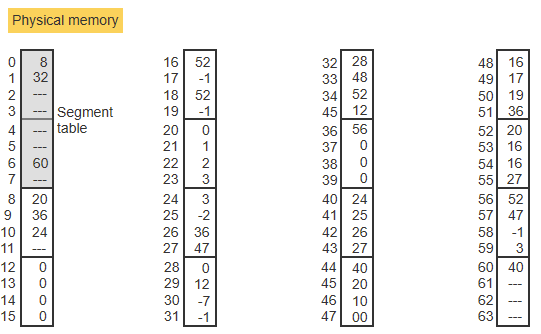
For each system, determine the total amount of occupied memory space, including all page or segment tables.

(b)

For each system, determine the amount of space wasted due to internal fragmentation.

6.4.4

A system using paging and segmentation is mapped on a physical memory of 64 words arranged into 16 frames. The frame size and page size is 4 words.



* The segment table starts at address 0 and occupies 2 frames.
* Page tables and pages occupy 1 frame each.
* A dash denotes a nonexistent page table or page.

(a)

Starting with the segment table at address 0, trace down the entire memory hierarchy and label each frame as: "page table of segment s," or "page p of segment s," where p and s are the actual page and segment numbers. Page frames that remain unmarked are free.

(b)

Translate the following logical addresses into the corresponding physical addresses and determine the content of each address:  
0, 27, 18, 96

6.4.5

In a system with paging and segmentation, each logical address (s, p, w) requires 3 memory accesses. To speed up the address translation, a TLB holds the components (s, p) together with the corresponding frame number.  
Accessing memory takes m nanoseconds. Accessing the TLB takes m/10 nanoseconds.

(a)

Determine the fraction h of memory accesses that need to find a match in the TLB, known as the hit ratio, such that the average address translation is reduced by 50%.