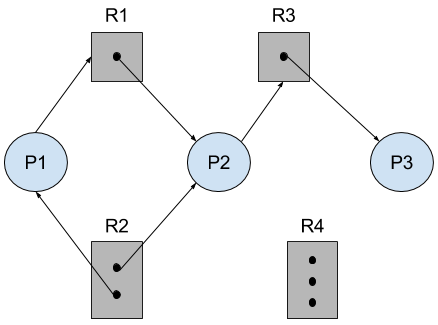
**Short answers**

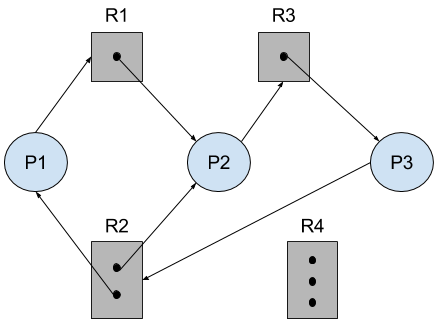
Chapter 7:

1. Example of system:

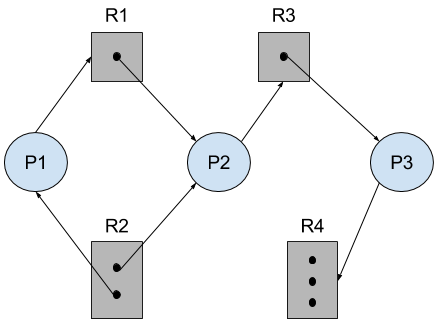


a. If P3 waits on R2 deadlock will happen. Because P2 is waiting for R3 which is held by P3 which is waiting for R2 which is held by both P1 which waits for R1 and also P2.

The RAG if this is the case:



b. If P3 waits on R4 there will be no deadlock.



1.

a.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Allocation | Max | Available |
|  | A B C D | A B C D | A B C D |
| P\_0 | 4 2 1 2 | 4 2 1 2 | 1 1 1 0 (5 3 2 2) |
| P\_3 | 1 4 2 4 | 1 4 2 4 | 5 2 1 0 (6 6 3 4) |
| P\_1 | 5 2 5 2 | 5 2 5 2 | 4 5 0 3 (9 7 5 5) |
| P\_2 | 2 3 1 6 | 2 3 1 6 | 9 5 4 2 ( 11 8 5 8) |
| P\_4 | 3 6 6 5 | 3 6 6 5 | 9 6 2 5 (12 12 8 10) |

b. Yes, if we allocated P\_1 (1, 1, 0, 0) the available remaining resources is (2, 2, 2, 1) and then if we follow the same sequence as before we will end up with a safe state

|  |  |  |  |
| --- | --- | --- | --- |
|  | Allocation | Max | Available |
|  | A B C D | A B C D | A B C D |
| P\_0 | 4 2 1 2 | 4 2 1 2 | 0 0 1 0 (4 2 2 2) |
| P\_3 | 1 4 2 4 | 1 4 2 4 | 4 1 1 0 (5 5 3 4) |
| P\_1 | 5 2 5 2 | 5 2 5 2 | 4 5 0 3 (9 7 5 5) |
| P\_2 | 2 3 1 6 | 2 3 1 6 | 9 5 4 2 ( 11 8 5 8) |
| P\_4 | 3 6 6 5 | 3 6 6 5 | 9 6 2 5 (12 12 8 10) |

c. No, if we allocated P\_4(0, 0, 2, 0) the available remaining resources is (3, 3, 0, 1) and neither of the remaining processes can finish (this includes P\_4) because there is not enough resources to even satisfy any process to completion.

d. The request will not be granted immediately, there is not enough available resources to satisfy P\_3's request for 3 instances of resource D because the current available instances of D is only 1. Depending on the system, it will either deny the request, place it on hold, or only allocate the remaining resources it has if it even has any left.

Chapter 8:

1. It is most efficient to break the address into X page bits and Y offset bits, rather than perform arithmetic on the address to calculate the page number and offset. Because each bit position represents a power of 2, splitting an address between bits results in a page size that is a power of 2.

2. Internal fragmentation is the wasted space within each allocated block because of rounding up from the actual requested allocation to the allocation granularity. External fragmentation is the various free spaced holes that are generated in either your memory or disk space. External fragmented blocks are available for allocation, but may be too small to be of any use.

3.

|  |  |  |  |
| --- | --- | --- | --- |
|  | External Fragmentation | Internal Fragmentation | Ability to share code |
| contiguous memory allocation (fixed size) | There is no external fragmentation | There is internal fragmentation | Does not allow processes to share code. |
| contiguous memory allocation (variable size) | There is external fragmentation | There is no internal fragmentation | Does not allow processes to share code. |
| pure segmentation | There is external fragmentation | There is no internal fragmentation | Able to share code between processes |
| pure paging | There is no external fragmentation | There is internal fragmentation | Able to share code between processes |

4. In certain situations the page tables could become large enough that by paging the page tables, one could simplify the memory allocation problem (by ensuring that everything is allocated as fixed-size pages as opposed to variable-sized chunks) and also enable the swapping of portions of page table that are not currently used.