**BCIT**

**Comp 4735 Operating Systems**

**Instructor: Mirela Gutica**

**Fall 2015**

**Mark: \_\_\_\_\_\_\_\_ / 110**

**Assignment 2**

Note: To receive any credit whatsoever, your answers must be legible and readily readable in the judgement of the grader. Add brief explanatory comments as necessary to make sure your answers are clear and unambiguous to the grader. **When you solve a problem, show all the steps, similar with the examples in the lectures**. Just the answer will not give you credits for a problem. **It is required to have a professional layout for the assignment**.

The assignment should be handed-in no later than **Friday, December 4, 2015, at 11:30pm. No late assignments will be accepted.**

**Solve the problems:**

(20p)

1. Consider a fast food restaurant with cooks, packers and one cashier. Consider that cooks prepare burgers and place them on a table. Packers get burgers from that table, them and place them on a counter for customers to get them. Customers pay first and then are allowed to walk to the table and get burgers.
   1. Define processes and resources.

Assumptions:

* Cooks never leaves their workstation
* Cooks will be in the kitchen when the restaurant is open
* Two buffers; table and counter

Processes:

* Cooks
* Packers
* Cashiers
* Customers

Resources

* Burgers
  1. Solve this problem using semaphores (in pseudo-code).

Semaphores/Mutexes:

semaphore:

cooks\_ready = n

packers\_ready = n

customer = 0

cashier\_ready = 1

table\_ready = n

table\_occupied = 0

counter\_occupied = 0

counter\_avaliable = n

customer\_ready = n

mutexes:

table\_avaliable = 1

wrapping = 0

getBurger = 1

Cook {

while (true) {

makeBurger();

wait(table\_ready);

wait(table\_avaliable);

placeBurgerOnTable();

signal(table\_avaliable);

signal(table\_occupied);

}

}

Packer {

while (true) {

wait(table\_occupied);

wait(wrapping);

wrapBurger();

signal(wrapping);

signal(table\_ready);

wait(counter\_avaliable);

placeBurgerOnCounter();

signal(counter\_occupied);

}

}

Cashier {

while (true) {

wait(customer\_ready);

takeOrder();

takeCurrency();

signal(cashier\_ready);

}

}

Customer {

goToRestaurant();

signal(customer\_ready);

wait(cashier\_ready);

orderBurger();

pay();

wait(counter\_occupied);

wait(getBurger);

getBurger();

signal(getBurger);

signal(counter\_avaliable)

eatBurger();

leave();

}

* 1. Solve this problem with monitors (in pseudo-code).

Monitor/Mutexes:

Monitor:

table\_ready = 1

table\_occupied = 0

counter\_occupied = 0

counter\_avaliable = 1

cashier\_ready = 1

customer\_ready = 0

mutexes:

table\_avaliable = 1

wrapping = 0

getBurger = 1

Cook {

while (true) {

makeBurger();

lock (table\_ready); {

lock (table\_avaliable) {

placeBurgerOnTable();

}

}

}

}

Packer {

while (true) {

lock(table\_occupied) {

lock(wrapping) {

wrapBurger();

}

}

lock(counter\_avaliable) {

placeBurgerOnCounter();

}

}

}

Cashier {

while (true) {

lock(customer\_ready) {

takeOrder();

takeCurrency();

}

}

}

Customer {

goToRestaurant();

lock(cashier\_ready) {

orderBurger();

pay();

}

lock(getBurger) {

getBurger();

}

eatBurger();

leave();

}

(6p)

1. Consider that you have to design an operating system for critical situations and want to implement a deadlock policy. Indicate three possible solutions and weight the side effects of each.

The first 3 conditions that make the deadlock possible, they are:

* **Mutual Exclusion:** Only a single process may use a resource at a time. No other process may access a resource that is currently occupied by another process.
* **Hold-and-wait:** A process may hold resources while awaiting assignment of other resources.
* **No Preemption:** No resource can be taken by force from a process that is holding it.

The last condition is what allows the deadlock to occur:

* **Circular Wait:** A closed chain of processes where each process holds at least one resource required by the next process in the chain.

The first approach, Prevention, is used to design a system that excludes the possibility of a deadlock from occurring. This approach is described as conservative and under commits resources. Prevention constrains resource requests to prevent at least one out of the four conditions of a deadlock thus preventing a deadlock from occurring. Prevention has two different implementations; indirect method and direct method. Indirect method prevents the first 3 conditions from being satisfied while direct method only prevents the last condition from being satisfied. The indirect method handles Mutual Exclusion by allowing some resources multiple accesses for reading but only exclusive access for writing. Hold-and-wait can be prevented by having the process request all its required resources at once and block the process until all the requests can be granted at once. The advantage of this is it works well for processes that perform a single burst of activity. However the cons for using this method is that the process can be held up for an indefinite amount of time while waiting for its request to be granted. No Preemption has multiple approaches; denying further request to a process holding certain resources and have them release their current resources and re-request them along with additional resources, and allow a process that is requesting a resource that is held by another process to force its release.

The second approach is avoidance. This approach resolves deadlocks by allowing the first 3 conditions but makes choices to prevent the actual deadlock from occurring. Avoidance does not need to implement the first approach, no preemption. It checks the knowledge of future process requests of resources to determine if it may lead to a deadlock. There are two implementations of this method; Process Initiation Denial and Resource Allocation Denial. Process Initiation Denial is that a process may not start if its demands lead to a deadlock. It checks if the maximum claims of resources of all processes are met. Resource Allocation Denial is that an incremental resource request will not be granted if its allocation leads to a deadlock. This is also known as the Banker’s Algorithm (see question 7).

The third and last approach is Detection. It grants resources to processes that request it and uses a deadlock detection algorithm to detect if a circular wait condition will occur. This algorithm is periodically executed. Using this method results with no delay with process initiation (doesn’t have to wait to get requested resources), and handling is facilitated online. The first two approaches places constraints and limitations on resources. The system checks and determines whether a process should receive their requested resources or not. The third approach however, does not place any restrictions or constraints on the resources.

(8p)

1. Consider a dynamic partitioning system with the following available free blocks (in this order from left to right): 150KB, 360KB, 400KB, 625KB and 200KB. Assume that the memory starts with an occupied block of 100KB and ends with an occupied block of 450KB. The system receives the following requests in this order: P1: 215KB, P2: 171KB, P3: 86KB, and P4: 481KB. (1) Indicate if all new processes can be allocated in the memory and (2) describe the final content of memory indicating the free space for:
   1. Best-fit
   2. First-fit
   3. Next-fit
   4. Worst-fit

Based on this problem and on what we discussed in class, contrast and compare the four placement strategies.

Note: Assume that the last allocation was just before the 400KB block.

(8p)

1. Consider simple paging. Consider that the main memory has 128Mbytes and the size of a page is 2Kbytes. Consider that the memory is byte addressable. A process P has 5 logical pages (first logical page has the address 0).

Answer the questions:

* 1. What is the number of frames in this system?

128Mbytes -> 128,000,000bytes

2Kbytes -> 2,000bytes

128,000,000bytes / 2,000bytes = **64,000 frames**

* 1. How many bits are allocated to the page address?

Log base 2 (64,000) = 15.965784

**Requires 16bits to be allocated**

* 1. Consider that P has been mapped on frames: 10, 16, 31, 32 and 65 (in this order). What is the physical address for:
     1. 000000000000011110011010011
     2. 000000000000000000111101111
     3. 000000000000010000000000010
     4. 000000000000001001001001001
     5. 000000000000001100000000110

(6p)

1. Explain the advantages of virtual segmentation over virtual paging. Explain why virtual segmentation is not used in modern operating systems.

Segmentation:

Segmentation represents an organization of memory in multiple address spaces or segments. Segments can be set as unequal, dynamic size modules. Advantages of using segmentation is that it simplifies the handling of growing data structures. Because segments can be organized as unequal, dynamic size modules, the programmer does not require to set the size of a particular data structure. The OS handles expanding or shrinking the segment as needed. If a segment in main memory needs to expand and there is insufficient room, the OS may move the segment to a larger area of main memory, or the segment will be swapped out and back in at the next opportunity. Segments allows programs to be modified and compiled independently without needing the set of programs to be relinked and reloaded. Segments also lends itself to sharing data among processes. Segments can lend itself to protection due to the fact that segments can be constructed to contain defined set of data and the programmer can assign access privileges accordingly. Segments are visible to the programmer and eliminates internal fragmentation

Although using segments provides a variety of benefits, it is no longer used in modern operating systems. This is because segmentations provide limitations for modern programmers. In modern operating systems, a combination of segmentation and paging is used instead. Paging is transparent to the programmer, prevents external fragmentation, and provides efficient use of the main memory. Combining both allows the possibility to have the pros from both without their disadvantages.

(6p)

1. A system has a virtual address space of 48 bits and a physical address space of 32 bits. A page is 4Kbytes, the memory is byte addressable, and a page entry has 8Bytes.
2. How many virtual pages are in this system?
3. How many pages are needed for page tables assuming that the page tables are allocated continuously in the memory?
4. Is it possible for the page tables to be stored in the main memory? If the answer is yes, what percentage of the main memory is used by page tables?

(6p)

1. Give as many reasons as you can why locality is a phenomenon that is common during process execution. Give as many reasons as you can of situations when the principle of locality is not happening.

Program and data references within a process is expected to cluster. The principle of locality is a fundamental that suggests the majority of future references will be directed towards recently brought in pages. Pages that were just referenced is likely to be referenced again in the near future. As a result, after a period of time the number of page faults should decrease to a very low level; this suggests that with the use of special algorithms, virtual memory may work efficiently. To apply this principle, the OS should have an implementation of special algorithms which can be used to dynamically predict which pages and segments are required next based on statistics and given data. However there are some situations where the principle of locality could not occur. Examples of this is that if a reference to a recently brought in page is only occurs once and no longer referenced again in the future, such as a value only being referenced once.

(20p)

1. Consider that the execution of a process requests the following page references: (pg 363)

1, 1, 3, 5, 2, 2, 6, 8, 7, 6, 2, 1, 5, 5, 5, 1, 4, 9, 7, 7

Consider fixed allocation, local scope; 5 frames per process. Show the page allocation and calculate the page fault ratio because page replacement for the following policies:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |

1, 1, 3, 5, 2, 2, 6, 8, 7, 6, 2, 1, 5, 5, 5, 1, 4, 9, 7, 7

* 1. FIFO

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 1 | |  | |  | |  | |  | | |  | | --- | | 1 | |  | |  | |  | |  | | |  | | --- | | 1 | | 3 | |  | |  | |  | | |  | | --- | | 1 | | 3 | | 5 | |  | |  | | |  | | --- | | 1 | | 3 | | 5 | | 2 | |  | | |  | | --- | | 1 | | 3 | | 5 | | 2 | |  | | |  | | --- | | 1 | | 3 | | 5 | | 2 | | 6 | | |  | | --- | | 8 | | 3 | | 5 | | 2 | | 6 | | |  | | --- | | 8 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 8 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 8 | | 7 | | 5 | | 2 | | 6 | |
|  |  |  |  |  |  |  | F | F |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 8 | | 7 | | 1 | | 2 | | 6 | | |  | | --- | | 8 | | 7 | | 1 | | 5 | | 6 | | |  | | --- | | 8 | | 7 | | 1 | | 5 | | 6 | | |  | | --- | | 8 | | 7 | | 1 | | 5 | | 6 | | |  | | --- | | 8 | | 7 | | 1 | | 5 | | 6 | | |  | | --- | | 8 | | 7 | | 1 | | 5 | | 4 | | |  | | --- | | 9 | | 7 | | 1 | | 5 | | 4 | | |  | | --- | | 9 | | 7 | | 1 | | 5 | | 4 | | |  | | --- | | 9 | | 7 | | 1 | | 5 | | 4 | |  |
| F | F |  |  |  | F |  |  |  |  |

F = 5/10

1, 1, 3, 5, 2, 2, 6, 8, 7, 6, 2, 1, 5, 5, 5, 1, 4, 9, 7, 7

* 1. LRU

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 1 | |  | |  | |  | |  | | |  | | --- | | 1 | |  | |  | |  | |  | | |  | | --- | | 1 | | 3 | |  | |  | |  | | |  | | --- | | 1 | | 3 | | 5 | |  | |  | | |  | | --- | | 1 | | 3 | | 5 | | 2 | |  | | |  | | --- | | 1 | | 3 | | 5 | | 2 | |  | | |  | | --- | | 1 | | 3 | | 5 | | 2 | | 6 | | |  | | --- | | 8 | | 3 | | 5 | | 2 | | 6 | | |  | | --- | | 8 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 8 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 8 | | 7 | | 5 | | 2 | | 6 | |
|  |  |  |  |  |  |  | F | F |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | |  | | --- | | 8 | | 7 | | 1 | | 2 | | 6 | | |  | | --- | | 5 | | 7 | | 1 | | 2 | | 6 | | |  | | --- | | 5 | | 7 | | 1 | | 2 | | 6 | | |  | | --- | | 5 | | 7 | | 1 | | 2 | | 6 | | |  | | --- | | 5 | | 7 | | 1 | | 2 | | 6 | | |  | | --- | | 5 | | 4 | | 1 | | 2 | | 6 | | |  | | --- | | 5 | | 4 | | 1 | | 2 | | 9 | | |  | | --- | | 5 | | 4 | | 1 | | 7 | | 9 | | |  | | --- | | 5 | | 4 | | 1 | | 7 | | 9 | |  |
|  | F | F |  |  |  | F | F | F |  |  |

F = 7/10

1, 1, 3, 5, 2, 2, 6, 8, 7, 6, 2, 1, 5, 5, 5, 1, 4, 9, 7, 7

* 1. Optimal

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 1 | |  | |  | |  | |  | | |  | | --- | | 1 | |  | |  | |  | |  | | |  | | --- | | 1 | | 3 | |  | |  | |  | | |  | | --- | | 1 | | 3 | | 5 | |  | |  | | |  | | --- | | 1 | | 3 | | 5 | | 2 | |  | | |  | | --- | | 1 | | 3 | | 5 | | 2 | |  | | |  | | --- | | 1 | | 3 | | 5 | | 2 | | 6 | | |  | | --- | | 1 | | 8 | | 5 | | 2 | | 6 | | |  | | --- | | 1 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 1 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 1 | | 7 | | 5 | | 2 | | 6 | |
|  |  |  |  |  |  |  | F | F |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 1 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 1 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 1 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 1 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 1 | | 7 | | 5 | | 2 | | 6 | | |  | | --- | | 1 | | 7 | | 5 | | 2 | | 4 | | |  | | --- | | 1 | | 7 | | 5 | | 9 | | 4 | | |  | | --- | | 1 | | 7 | | 5 | | 9 | | 4 | | |  | | --- | | 1 | | 7 | | 5 | | 9 | | 4 | |  |
|  |  |  |  |  | F | F |  |  |  |

F = 4/10

1, 1, 3, 5, 2, 2, 6, 8, 7, 6, 2, 1, 5, 5, 5, 1, 4, 9, 7, 7

* 1. Simple clock (pre-paging: pages 1,2,3,4,5)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |

1, 1, 3, 5, 2, 2, 6, 8, 7, 6, 2, 1, 5, 5, 5, 1, 4, 9, 7, 7

* 1. Simple clock (no pre-paging)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 11 | |  | |  | |  | |  | | |  | | --- | | 11 | |  | |  | |  | |  | | |  | | --- | | 11 | | 31 | |  | |  | |  | | |  | | --- | | 11 | | 31 | | 51 | |  | |  | | |  | | --- | | 11 | | 31 | | 51 | | 21 | |  | | |  | | --- | | 11 | | 31 | | 51 | | 21 | |  | | |  | | --- | | 11 | | 31 | | 51 | | 21 | | 61 | | |  | | --- | | 81 | | 30 | | 50 | | 20 | | 60 | | |  | | --- | | 81 | | 71 | | 50 | | 20 | | 60 | | |  | | --- | | 81 | | 71 | | 51 | | 20 | | 61 | |
|  |  |  |  |  |  |  | F | F |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 81 | | 71 | | 51 | | 21 | | 61 | | |  | | --- | | 80 | | 70 | | 11 | | 20 | | 60 | | |  | | --- | | 80 | | 70 | | 11 | | 51 | | 60 | | |  | | --- | | 80 | | 70 | | 11 | | 51 | | 60 | | |  | | --- | | 80 | | 70 | | 11 | | 51 | | 60 | | |  | | --- | | 80 | | 70 | | 11 | | 51 | | 60 | | |  | | --- | | 80 | | 70 | | 11 | | 51 | | 41 | | |  | | --- | | 91 | | 70 | | 11 | | 51 | | 41 | | |  | | --- | | 91 | | 71 | | 11 | | 51 | | 41 | | |  | | --- | | 91 | | 71 | | 11 | | 51 | | 41 | |
|  | F | F |  |  |  |  | F |  |  |

F = 5/10

(6p)

1. Suppose that a memory management system has chosen a modified page P for replacement. Page P must be sent to secondary storage before the new page is placed in its frame. Therefore, an I/O request is issued. Consider that during the time when the I/O request, the page P is requested again.
   1. What phenomenon describes this situation? Elaborate.
   2. Indicate two strategies that could be employed by the memory management system such that the system executes with an optimal performance? Elaborate.

(24p)

1. Suppose jobs A, B, C, D, and E arrived in a system at time 0, 2, 4, 6, and 7. Assume that the job lengths are: A = 10, B= 5, C = 6, D = 3 E = 5 time units. Draw the time diagrams and calculate Tr and Tr/Ts for each process:
   1. RR q = 1
   2. RR q = 4
   3. SPN
   4. SRT
   5. HRRN
   6. FB, three queues, q1 = 1, q2 = 2, q3 = 4, n1 = 1, n2 = 2.

Note: a. If a process executes its time slice, it cannot be interrupted by the arrival of a new process.

b. If two processes join the queue in the same time: one just finishing the time slice and one new process, the new process has priority over the older one.