**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.

**Processes**

Cooks, Packers, Cashier, Customers

**Resources**

Burgers

**Other assumptions**

The tables are buffers

The restaurant is always open

The cooks never leave their workstations

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

/\* Semaphores & Mutexes \*/

sem\_cooks\_ready = n;

sem\_packers\_ready = n;

sem\_cashier\_ready = 1;

sem\_customer\_ready = 0;

sem\_table\_ready = n;

sem\_table\_occupied = n;

sem\_counter\_occupied = 0;

sem\_counter\_available = n;

mut\_table\_space\_available[] = n;

mut\_packing = 0;

mut\_get\_burger[] = n;

Cook {

while (true) {

makeBurger();

wait(sem\_table\_ready);

wait(mut\_table\_space\_available);

placeBurgerOnTable();

signal(mut\_table\_space\_available);

signal(sem\_table\_occupied);

}

}

Packer {

while(true) {

wait(sem\_table\_occupied);

wait(mut\_packing);

packBurger();

signal(mut\_packing);

signal(sem\_table\_ready);

wait(sem\_counter\_available);

placeBurgerOnCounter();

signal(sem\_counter\_occupied);

}

}

Cashier {

while(true) {

wait(sem\_customer\_ready);

takeOrderFromCustomer();

takeMoney();

signal(cashier\_ready);

}

}

Customer {

goToRestaurant();

signal(sem\_customer\_ready);

wait(sem\_cashier\_ready);

orderBurger();

pay();

wait(sem\_counter\_occupied);

wait(mut\_get\_burger);

getBurger();

signal(mut\_get\_burger);

signal(sem\_counter\_available);

eatBurger();

leave();

}

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

/\* Semaphores & Mutexes \*/

table\_ready;

table\_occupied;

table\_space\_available;

packing\_space;

counter\_available;

customer\_ready;

cashier\_ready;

counter\_occupied;

get\_burger;

Cook {

while (true) {

makeBurger();

lock (table\_ready) {

lock(table\_space\_available) {

placeBurgerOnTable();

}

}

}

}

Packer {

while(true) {

lock (table\_occupied) {

lock (packing\_space) {

packBurger();

}

}

lock (counter\_available) {

placeBurgerOnCounter();

}

}

}

Cashier {

while(true) {

lock (customer\_ready) {

takeOrderFromCustomer();

takeMoney();

}

}

}

Customer {

goToRestaurant();

lock (cashier\_ready) {

orderBurger();

pay();

}

lock (counter\_occupied) {

lock (get\_burger) {

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 conditions for a deadlock are mutual exclusion; hold and wait; no preemption; and circular wait. It is through circular wait, however, that makes a deadlock occur.

**Mutual exclusion**

Only one process may use a resource at a time. No process

may access a resource unit that has been allocated to another process.

**Hold and wait**

A process may hold allocated resources while awaiting assignment

of other resources.

**No preemption**

No resource can be forcibly removed from a process holding it.

**Circular wait**

A closed chain of processes exists, such that each process holds

at least one resource needed by the next process in the chain

There are three approaches for deadlock prevention.

**Prevention**

To prevent a deadlock, the system is designed such that it prevents the occurrence of one of the first three conditions above, in order to avoid the fourth condition, circular wait. It is quite conservative in allocation of resources and inefficient execution of processes.

**Avoidance**

The system dynamically predicts if the acquisition of a resource will potentially produce a deadlock. It requires knowledge of future process resource requests.

**Detection**

An algorithm is run periodically to determine if a circular wait condition has occurred. Once a deadlock has been detected, a number of remedies can be used: Abort all deadlocked processes; back up each deadlocked process to some previously defined checkpoint, and restart all processes; successively abort deadlocked processes until dealock no longer exists; successively preempt resources until deadlock no longer exists.

(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?

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

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

Log base 2 of 64,000 = 15.965784 = **16 bits**

* 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 = 15571
     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. FIFO
  2. LRU
  3. Optimal
  4. Simple clock (pre-paging: pages 1,2,3,4,5)
  5. Simple clock (no pre-paging)

(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.

This situation is not an issue unless it happens too many times. If it happens too many times, it is called Thrashing. Basically, the OS is sending a page of a process to the secondary storage to make room for a new page of another process. However, since the OS is not finished working with that particular page of the prior process, it needs to retrieve it from secondary memory again. When this happens too many times, thrashing occurs. In this particular situation, page P is being sent to the secondary memory, so that another page may occupy its space in main memory, however in the middle of the I/O request to send page P to secondary memory, the OS requests page P again, thus another I/O request is made to retrieve page P from secondary memory to main.

* 1. Indicate two strategies that could be employed by the memory management system such that the system executes with an optimal performance? Elaborate.

A replacement policy will be used to determine which process page will be swapped out. An effective policy chooses the page that will be least likely to be used in the near future. The more elaborate the policy, the greater the hardware and software overhead to implement it. Two replacement policies are **Fixed Allocation** and **Dynamic/Variable Allocation.**

**Fixed Allocation**

**Dynamic/Variable Allocation**

(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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A |  | B |  | C |  | D | E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| A |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A1 | A2 | E1 | C1 | A3 | E2 | C2 | A4 | E3 | C3 | D1 | B1 | A5 | E4 | C4 | D2 | B2 | A6 | E5 | C5 | D3 | B3 | A7 | C6 | B4 | A8 | B5 | A9 | A10 |

**Queue:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** |
| **Tr** | 29 | 16 | 22 | 13 | 20 |
| **Tr/Ts** | 29/10 = 2.9 | 16/5 = 3.2 | 22/6 = 3.67 | 13/3 = 4.33 | 20/5 = 4 |

* 1. RR q = 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A |  | B |  | C |  | D | E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| A |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Queue:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| E1 | A2 | C2 | B1 | E5 | D3 | A6 | C6 | B5 | A10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** |
| **Tr** | 28 | 22 | 22 | 13 | 22 |
| **Tr/Ts** | 28/10 = 2.8 | 22/5 = 4.4 | 22/6 = 3.67 | 13 / 3 = 4.33 | 22/5 = 4.4 |

* 1. SPN
  2. SRT
  3. HRRN
  4. 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.