**COMP4735 - Assignment 2**

Apr. 11th, 2017

Do-Yoon(Rick) Kim

Student#: A00816364

Set: 4A

**BCIT**

**Comp 4735 Operating Systems**

**Instructor: Mirela Gutica**

**Winter 2017**

**Mark: \_\_\_\_\_\_\_\_ / 145**

**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 **Monday, April 10, 2017, at 3:30pm. No late assignments will be accepted.**

**Solve the problems:**

(20p)

1. Consider a situation when an OS exam is organized as a combination of a computer test and an oral examination. In order to pass the exam, a student should pass both components. If the student fails the computer test, he/she can retake the computer test in the same day (only once). The oral examination cannot be retaken. Each student should enter a waiting room with ten chairs. The school regulations don’t allow more than ten students in the waiting room. The waiting room has two doors one to the computer room and one to the oral examination room. Students should take the computer test first. The computer room has five computers, so five students can write the computer test in the same time. The oral examination room has two teachers, so only two students can take the oral examination in the same time. If a student fails once the computer test, he/she gets a priority ticket in e-mail and should enter the waiting room again. Because the computer test should be retaken in the same day, a student who failed once has priority over a student who just arrived. If a student fails the computer test the second time, he/she cannot take the oral examination.

Solve this synchronization problem using semaphores.

sem Room = 10

sem Chair = 10

sem ChairAccess = 1

sem CompTestRoom = 5

sem CompTestRoomDeskAccess = 1

sem OralTestRoom = 2

sem OralTestRoomDeskAccess = 1

**Student** {

if(TakingTest) {

if(PriorityStudent) {

GoToTestRoomFirst();

} else {

GoToTestRoom();

}

**0** Wait(Room);

**1** Wait(Chair);

**2** Wait(ChairAccess);

Sit();

**2** Signal(ChairAccess);

**3** Wait(CompTestRoom);

Leave-Chair();

**1** Signal(Chair);

**4** Wait(CompTestRoomDeskAccess);

EnterCompTestRoom();

**4** Signal(CompTestRoomDeskAccess);

**6** Wait(Test.ComputerTest);

**5** Signal(CompAvailForStudent);

TakingCompTest();

**3** Signal(CompTestRoom);

if (CompTestPass) {

**a** Wait(OralTestRoom);

**b** Wait(OralTestRoomDeskAccess);

EnterOralTestRoom();

**b** Signal(OralTestRoomDeskAccess);

**c** Wait(Test.OralTest);

**d** Signal(StudentAvailForOral);

TakingOralTest();

**a** Signal(OralTestRoom);

GetOralTestResult();

**0** Signal(Room);

} else {

if (SecondFail) {

**0** Signal(Room);

} else {

GetPriorityEmailTicket();

RetakeCompTest();

**0** Signal(Room);

}

}

}

}

**Test** {

While(TestHours) {

ComputerTest {

**5** Wait(CompAvailForStudent);

TestGiven\_Comp();

**6** Signal(ComputerTest);

}

OralTest {

**d** Wait(StudentAvailForOral);

TestGiven\_Oral();

**c** Signal(OralTest);

}

}

}

(15p)

1. Consider simple paging. Consider that the main memory has 1Gbyte and the size of a page is 1Kbyte. Consider that the memory is byte addressable. A process P has 8 logical pages (first logical page has the address 0).

Answer the questions:

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

* **Ans:** 230 bytes (=Main Memory) / 210 bytes (=page size) = 220 frames

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

* 20 bits for frame and 10 bits for offset ∴ **30 bits** are needed

1. Consider that P has been mapped on frames: 31, 32, 40, 50, 51, 52, 63 and 64 (in this order). What is the physical address for the following logical addresses:
   * 1. 0000 0000 0000 0000 0001 1011111110
        + Logical index: 1 🡪 Physical frame: 32
        + **Ans:** 0000 0000 0000 0010 0000 1011111110
     2. 0000 0000 0000 0000 0011 1100111000
        + Logical index: 3 🡪 Physical frame: 50
        + **Ans:** 0000 0000 0000 0011 0010 1100111000
     3. 0000 0000 0000 0000 0000 0000110101
        + Logical index: 0 🡪 Physical frame: 31
        + **Ans:** 0000 0000 0000 0001 1111 0000110101
     4. 0000 0000 0000 0010 0101 0001001110
        + Logical index: 37 🡪 Physical frame: Not Available
        + **Ans:** Page table has index number up to 7, ∴ physical address is not available
     5. 0000 0000 0000 0000 0111 0000100000
        + Logical index: 7 🡪 Physical frame: 64
        + **Ans:** 0000 0000 0000 0100 0000 0000100000
     6. 0000 0000 0000 0000 0101 0000100000
        + Logical index: 5 🡪 Physical frame: 52
        + **Ans:** 0000 0000 0000 0011 0100 0000100000

(15p)

1. Consider a virtual paging system implemented on a very powerful machine that has the following characteristics: 264 of bytes of physical memory; page size of 4 Kbytes, 2116 pages of virtual address space.
2. How many frames are in the physical memory?

Ans: 264 / 212 = 252

1. How large is the virtual memory?

Ans: 2116 pages\* 212 bytes = 2128 bytes of virtual address space

1. How many bits are in the virtual address?

Ans: 128 bits are in the virtual address

1. How many bytes are in a frame?

Ans: Sames as page = 212 bytes

1. How many bits are in the physical address that specifies the frame?

Ans: 52 bits 🡪 264 / 212 = 252

1. How many entries are in total in page tables?

Ans: One entry per page 🡪 therefore, 2116 entries

1. Consider that each page table entry has a bit that indicates validity and 11 more control bits. How many bits are in each page table entry?

Ans: 52 + 1 + 11 = 64 bits

1. How large are the page tables (maximum size)? Assume that the page tables are allocated continuously in the memory. Give the answer in bytes and in pages.

Ans: Each page table entry has 64 bits ( = 8 Bytes) and 2116 pages.

* + - * 2116 pages \* 8 bytes = 2119 bytes.
      * 2119 bytes / 212 bytes = 2107 pages only to store page tables

1. 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? If the answer is “no”, where should the page tables stored?

Ans: Main memory = 252 frames. 2107 pages exceed the size of MM, therefore, Page tables should also be stored in secondary memory (= Segmentation fault).

1. Would this system benefit from a larger frame/page? What would be a disadvantage of a larger frame/page?

Ans: Larger pages result in less space consumed with page tables; but there will be more internal fragmentation

(30p)

1. Consider that a process was allocated on 9 pages as follows: pages 1 to 3 the heap, page 4 the user stack, page 5 the system stack and pages 6 to 9 the code. Consider that the execution of the process requests the following page references:

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

1. 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:

**For all algorithms in Q4-A, my assumption is to include all pages from beginning; therefore it’s counted as 28, not 22, but the page fault will begin once the frames are filled.**

* + 1. **FIFO**

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

Fault Ratio = 10/28 = **0.357143 (36%)**

* + 1. **LRU**

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

Fault Ratio = 15/28 = **0.535714 (54%)**

* + 1. **Optimal**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Optimal | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **6** | **6** | **1** | **6** | **2** | **7** | **4** | **1** | **8** | **9** | **3** | **7** | **5** | **8** | **6** | **9** | **2** | **2** | **6** | **8** | **7** | **5** | **9** | **6** | **2** | **1** | **7** | **8** |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 1 | 1 | 1 |
|  |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|  |  |  |  | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 9 | 9 | 9 | 9 | 9 | 9 |
|  |  |  |  |  | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
|  |  |  |  |  |  | 4 | 4 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
|  |  |  |  |  |  |  |  | F | F | F |  | F |  |  |  | F |  |  |  |  |  | F |  |  | F |  |  |
| \* Assuming 6 is loaded later than 2, 9 at the end | | | | | | | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |

Fault Ratio = 7/28 = **0.25 (25%)**

* + 1. **Simple clock (pre-paging: pages 1, 2, 4, 6, 7)**

Simple Clock (Pre-paging: pages 1, 2, 4, 6, 7)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** | **6** | **1** | | **6** | | **2** | | **7** | | **4** | | **1** | | **8** | | **9** | | **3** | | **7** | | **5** | | **8** | | **6** |
| >1(0) | >1(0) | >1(1) | | >1(1) | | >1(1) | | >1(1) | | >1(1) | | >1(1) | | 8(1) | | 8(1) | | 8(1) | | 8(1) | | 8(1) | | 8(1) | | >8(0) |
| 2(0) | 2(0) | 2(0) | | 2(0) | | 2(1) | | 2(1) | | 2(1) | | 2(1) | | >2(0) | | 9(1) | | 9(1) | | 9(1) | | 9(1) | | 9(1) | | 9(0) |
| 4(0) | 4(0) | 4(0) | | 4(0) | | 4(0) | | 4(0) | | 4(1) | | 4(1) | | 4(0) | | >4(0) | | 3(1) | | 3(1) | | 3(1) | | 3(1) | | 3(0) |
| 6(1) | 6(1) | 6(1) | | 6(1) | | 6(1) | | 6(1) | | 6(1) | | 6(1) | | 6(0) | | 6(0) | | >6(0) | | >6(0) | | 5(1) | | 5(1) | | 5(0) |
| 7(0) | 7(0) | 7(0) | | 7(0) | | 7(0) | | 7(1) | | 7(1) | | 7(1) | | 7(0) | | 7(0) | | 7(0) | | 7(1) | | >7(1) | | >7(1) | | 6(1) |
|  |  |  | |  | |  | |  | |  | |  | | F | | F | | F | |  | | F | |  | | F |
| **9** | **2** | **2** | | **6** | | **8** | | **7** | | **5** | | **9** | | **6** | | **2** | | **1** | | **7** | | **8** | |
| >8(0) | 2(1) | 2(1) | | 2(1) | | 2(1) | | 2(1) | | 2(0) | | >2(0) | | 6(1) | | 6(1) | | 6(1) | | 6(0) | | > | |
| 9(1) | >9(1) | >9(1) | | >9(1) | | 9(0) | | 9(0) | | 5(1) | | 5(1) | | >5(1) | | 5(0) | | 5(0) | | 7(1) | | 7(1) | |
| 3(0) | 3(0) | 3(0) | | 3(0) | | 8(1) | | 8(1) | | >8(1) | | 8(0) | | 8(0) | | 2(1) | | 2(1) | | >2(1) | | 2(0) | |
| 5(0) | 5(0) | 5(0) | | 5(0) | | >5(0) | | 7(1) | | 7(1) | | 7(0) | | 7(0) | | >7(0) | | 1(1) | | 1(1) | | 1(0) | |
| 6(1) | 6(1) | 6(1) | | 6(1) | | 6(1) | | >6(1) | | 6(0) | | 9(1) | | 9(1) | | 9(1) | | >9(1) | | 9(0) | | 8(1) | |
|  | F |  | |  | | F | | F | | F | | F | | F | | F | | F | | F | | F | |

Fault Ratio = 15/28 = **0.535714 (54%)**

* + 1. **Simple clock (no pre-paging)**

Simple Clock (No Pre-paging)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** | **6** | **1** | **6** | **2** | **7** | **4** | **1** | **8** | **9** | **3** | **7** | **5** | **8** |
| 6(1) | 6(1) | 6(1) | 6(1) | 6(1) | 6(1) | >6(1) | >6(1) | 8(1) | 8(1) | 8(1) | 8(1) | >8(1) | >8(1) |
| > | > | 1(1) | 1(1) | 1(1) | 1(1) | 1(1) | 1(1) | >1(0) | 9(1) | 9(1) | 9(1) | 9(1) | 9(1) |
|  |  | > | > | 2(1) | 2(1) | 2(1) | 2(1) | 2(0) | >2(0) | 3(1) | 3(1) | 3(1) | 3(1) |
|  |  |  |  | > | 7(1) | 7(1) | 7(1) | 7(0) | 7(0) | >7(0) | >7(1) | 7(0) | 7(0) |
|  |  |  |  |  | > | 4(1) | 4(1) | 4(0) | 4(0) | 4(0) | 4(0) | 5(1) | 5(1) |
|  |  |  |  |  |  |  |  | F | F | F |  | F |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** | **9** | **2** | **2** | **6** | **8** | **7** | **5** | **9** | **6** | **2** | **1** | **7** | **8** |
| 8(0) | 8(0) | 2(1) | 2(1) | 2(1) | 2(1) | >2(1) | 2(0) | 2(0) | 6(1) | 6(1) | >6(1) | 6(0) | 6(0) |
| 9(0) | 9(1) | >9(1) | >9(1) | >9(1) | 9(0) | 9(0) | 5(1) | 5(1) | >5(1) | 5(0) | 5(0) | 7(1) | 7(1) |
| 3(0) | 3(0) | 3(0) | 3(0) | 3(0) | 8(1) | 8(1) | >8(1) | 8(0) | 8(0) | 2(1) | 2(1) | >2(1) | 2(0) |
| 6(1) | 6(1) | 6(1) | 6(1) | 6(1) | >6(1) | 6(0) | 6(0) | 9(1) | 9(1) | >9(1) | 9(0) | 9(0) | 8(1) |
| >5(1) | >5(1) | 5(0) | 5(0) | 5(0) | 5(0) | 7(1) | 7(1) | >7(1) | 7(0) | 7(0) | 1(1) | 1(1) | >1(1) |
| F |  | F |  |  | F | F | F | F | F | F | F | F | F |

Fault Ratio = 15/28 = **0.535714 (54%)**

1. Consider the working set with a window of 5.
   * 1. Show the page allocation in memory (using the working set policy) during execution starting from the beginning of execution. Calculate the number of page faults (when a page was added to the working set because it was not in the memory).

**The page faults are based on assumption that page fault start after working set size of 5 are filled**

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

**Number of page faults = 16 pages**

1. Answer the questions:
   * 1. What values should be given to the protection bit for each page?
        + System Stack 🡪 Give value of 1 for page 5
     2. Give an explanation of why pages 1, 2, 4, 6, and 7 were pre-loaded in the case of pre-paging.
        + Pre-loaded was needed to reduce time for swapping from secondary memory; the swapping would take much more time than accessing memory, in order to reduce execution time, the pages are pre-loaded.

(15p)

1. Consider the page replacing algorithms and memory management policies.
   1. What algorithms and policies are most suitable for modern operating systems? Why?
      * + Clock Algorithm is most suitable for modern OS because it would avoid huge overhead required by LRU algorithm, and clock is simpler and easier than Optimal to implement. Clock algorithm has the other advantage for OS due to variant of styles that can be made more powerful by increasing the amount of bits flags.
   2. What algorithms and policies are used for Linux, UNIX, Windows and iOS? Give a short description using your own words. Important: Give citations: state the resources from where you obtained the information.
      * + *Most sources are from Chapter 8 from the textbook*
        + **Linux (Prior to 2.6.28) – page 385**

Prior to 2.6.28, Linux page replacement algorithm is based on simple clock algorithm – page with 0 age is considered as candidate for replacement. When page is referenced, the age variable increments. Lastly, if the pool is filled with incremented age variables, the pointer sweeps through the pool and decrement the variable and when it reaches page with replaced variable 0 again, replace the page. But now, Linux uses LRU.

* + - * **UNIX (SVR4) – page 380**

This OS uses two-handed clock algorithm, the modified version of clock. And this uses 2 parameters to determine operation: 1) Scanrate and 2) Handspread. Moreover, frontend sets the policy and backend determines whether it is 0 or 1.

* + - * **Windows – page 388**

Windows uses clock that is based on LRU because when the memory is running short, virtual memory manager recovers memory by removing least recently used(LRU) pages from process

* + - * **iOS[[1]](#footnote-1)**
* It seems like iOS is using two-handed clock algorithm for paging out: 1) when page that was recently referenced is inactive, page moves to inactive list 2) When pages in inactive list is not being used, kernel finds the page’s VM object. 3) If VM object has never been paged before, initialization begins and sets default pager object. 4) the default pager attempts to write out to backing store 5) if pager succeeds, the kernel frees occupied memory

(10p)

1. Suppose that a process P is mainly executing algorithms. The process sends I/O requests to update data in an on-line database located on a different server (the database update will be performed by a server process/thread S). Answer the questions:
2. What kind of thrashing is possible to occur during the execution of the process P? Elaborate and give different scenarios.
   * + - **Case A:** Lots of page faults due to small size of pages – this would result in thrashing
       - **Case B:** Page faults occurs due to not enough size in main memory, causing the thrashing
       - **Case C:** Large volume of I/O requests when on-line database located on different server could result in downtime. Decreasing process utilization that could cause thrashing.
3. What kind of thrashing is it possible to occur during the execution of the server process S? Elaborate and give different scenarios.
   * + - **Case A:** When threads are excessively assigned making process slower and not going to assign additional threads any soon – the thrashing occurs
       - **Case B:** If process S’s cache size is smaller than requested query, also larger than main memory, it could cause thrashing
       - **Case C:** Other processes on server may have higher priority than process S which may not get required resources – resulting thrashing
4. What policies can be employed by the memory management system in each case such that the number of page faults is kept very low?
5. Fetch Policy
6. Placement policy
7. Replacement policy
8. Resident set management
9. Cleaning policy
10. Load control

(15p)

1. Consider the scheduling algorithms discussed in this course. Consider table 9.5 and figures 9.11, 9.12, 9.13, 9.14 and 9.15.
   1. What processes should be given preference in order to improve the normalized turnaround time at higher levels of utilization? Why?
      * + d
   2. How priority and priority with pre-emption policies affect (i) short and (ii) long processes?
   3. How is the normalized turn-around time influenced by the scheduling algorithm? Describe.
   4. How is the wait time influenced by the scheduling algorithm? Describe.
   5. Which type of processes is generally favoured by a multilevel feedback scheduler in that all processes arrive first in a high priority queue: a processor-bound process or an I/O bound process that returns to the high priority queue after each I/O request? Why?

(25p)

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.

1. **RR q = 1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Scheduling Policy: | | | | | | | **Round Robin** | | | | | | | | | | |  | | |
| Other Policy: | | | | | | | **q = 1** | | | | | | | | | | |  | | |
| **Process** | | | | | | | ***A*** | | | ***B*** | | ***C*** | | ***D*** | | ***E*** | |
| **Arrival Time** | | | | | | | ***0*** | | | ***2*** | | ***4*** | | ***6*** | | ***7*** | |
| **Service Time(Ts)** | | | | | | | ***10*** | | | ***5*** | | ***6*** | | ***3*** | | ***5*** | |
| **Finish Time** | | | | | | | 29 | | | 18 | | 26 | | 19 | | 27 | | Average | | |
| **Turnaround Time(Tr)** | | | | | | | 29 | | | 16 | | 22 | | 13 | | 20 | | 20 | | |
| **Tr/Ts** | | | | | | | 2.9 | | | 3.2 | | 3.7 | | 4.3 | | 4 | | 3.62 | | |
| 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 (from --> To)** | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | A | A | B | A | B | C | A | B | D | C | E | A | B | D | C | E | A | B | D | C | E | A | C | E | A | C | E | A | A |

1. **RR q = 4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scheduling Policy: | **Round Robin** | |  |  |  |  |
| Other Policy: | **q = 4** |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Process** | ***A*** | ***B*** | ***C*** | ***D*** | ***E*** |  |
| **Arrival Time** | *0* | *2* | *4* | *6* | *7* |  |
| **Service Time(Ts)** | *10* | *5* | *6* | *3* | *5* |  |
| **Finish Time** | 28 | 24 | 26 | 19 | 29 | Average |
| **Turnaround Time(Tr)** | 28 | 22 | 22 | 13 | 22 | 21.4 |
| **Tr/Ts** | 2.8 | 4.4 | 3.67 | 4.33 | 4.4 | 3.92 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 (from --> To)** | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | A | -- | -- | -- | B | -- | -- | -- | C | -- | -- | -- | A | -- | -- | -- | D | -- | -- | E | -- | -- | -- | B | C | -- | A | -- | E |

1. **Shortest Process Next (SPN)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scheduling Policy: | **SPN** |  |  |  |  |  |
| Other Policy: |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Process** | ***A*** | ***B*** | ***C*** | ***D*** | ***E*** |  |
| **Arrival Time** | *0* | *2* | *4* | *6* | *7* |  |
| **Service Time(Ts)** | *10* | *5* | *6* | *3* | *5* |  |
| **Finish Time** | 10 | 18 | 29 | 13 | 23 | Average |
| **Turnaround Time(Tr)** | 10 | 16 | 25 | 7 | 16 | 14.8 |
| **Tr/Ts** | 1 | 3.2 | 4.17 | 2.33 | 3.2 | 2.78 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 (from --> To)** | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | A | -- | -- | -- | -- | -- | -- | -- | -- | -- | D | -- | -- | B | -- | -- | -- | -- | E | -- | -- | -- | -- | C | -- | -- | -- | -- | -- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | B, C, D & E are in; start from shortest | | | | | | | | | | |  |  |  |  |  |  |  |  |

1. **Shortest Remaining Time (SRT)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scheduling Policy: | **SRT** |  |  |  |  |  |
| Other Policy: |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Process** | ***A*** | ***B*** | ***C*** | ***D*** | ***E*** |  |
| **Arrival Time** | *0* | *2* | *4* | *6* | *7* |  |
| **Service Time(Ts)** | *10* | *5* | *6* | *3* | *5* |  |
| **Finish Time** | 29 | 7 | 21 | 10 | 15 | Average |
| **Turnaround Time(Tr)** | 29 | 5 | 17 | 4 | 8 | 12.6 |
| **Tr/Ts** | 2.9 | 1 | 2.83 | 1.33 | 1.6 | 1.93 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 (from --> To)** | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | A | -- | B | -- | -- | -- | -- | D | -- | -- | E | -- | -- | -- | -- | C | -- | -- | -- | -- | -- | A | -- | -- | -- | -- | -- | -- | -- |

1. **Highest Response Ratio Next (HRRN)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scheduling Policy: | **HRRN** |  |  |  |  |  |
| Other Policy: |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Process** | ***A*** | ***B*** | ***C*** | ***D*** | ***E*** |  |
| **Arrival Time** | *0* | *2* | *4* | *6* | *7* |  |
| **Service Time(Ts)** | *10* | *5* | *6* | *3* | *5* |  |
| **Finish Time** | 10 | 15 | 24 | 18 | 29 | Average |
| **Turnaround Time(Tr)** | 10 | 13 | 20 | 12 | 22 | 15.4 |
| **Tr/Ts** | 1 | 2.6 | 3.3 | 4 | 4.4 | 3.07 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 (from --> To)** | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | A | -- | -- | -- | -- | -- | -- | -- | -- | -- | B | -- | -- | -- | -- | D | -- | -- | C | -- | -- | -- | -- | -- | E | -- | -- | -- | -- |
|  | By 10: | | Rb = { (8(wait) + 5 (service)) / 5} = 2.6 | | | | | | | | | | |  | By 15: | | Rc = 2.83 | | |  | By 18: | | Rc = 3.33 | | |  |  |  |  |
|  |  |  | Rc = { (6(wait) + 6 (service)) / 6} = 2 | | | | | | | | | | |  |  |  | Rd = 4 | |  |  |  |  | Re = 3.2 | | |  |  |  |  |
|  |  |  | Rd = { (4(wait) + 3 (service)) / 3} = 2.3 | | | | | | | | | | |  |  |  | Re = 2.6 | | |  |  |  |  |  |  |  |  |  |  |
|  |  |  | Re = { (3(wait) + 5 (service)) / 5} =1.6 | | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. **Feedback (FB), , three queues, q1 = 1, q2 = 2, q3 = 4, n1 = 1, n2 = 2.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scheduling Policy: | **Feedback** |  |  |  |  |  |
| Other Policy: | **q1 = 1, q2 = 2, q3 = 4** | | *(length of quantum)* | | |  |
|  | **n1 = 1, n2 = 2** | | *(number of execution)* | | |  |
| **Process** | ***A*** | ***B*** | ***C*** | ***D*** | ***E*** |  |
| **Arrival Time** | *0* | *2* | *4* | *6* | *7* |  |
| **Service Time(Ts)** | *10* | *5* | *6* | *3* | *5* |  |
| **Finish Time** | 29 | 19 | 28 | 23 | 15 | Average |
| **Turnaround Time(Tr)** | 29 | 17 | 24 | 17 | 8 | 19 |
| **Tr/Ts** | 2.9 | 3.4 | 4 | 5.67 | 1.6 | 3.51 |
| \* Assuming n3 = infinite | |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 (from --> To)** | | | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **q1** | A | B | C | E | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **q2** | A | -- | A | -- | B | -- | C | -- | E | -- | D | -- | B | -- | C | -- | D | -- |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **q3** | A | -- | -- | -- | C | A |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Memory Usage Performance Guideline, “https://developer.apple.com/library/content/documentation/Performance/Conceptual/ManagingMemory/Articles/AboutMemory.html” [↑](#footnote-ref-1)