**CSCE 611 600: OPERATING SYSTEMS Homework #2**

**Name:** Rohan Chaudhury

**Email Address:** [rohan.chaudhury@tamu.edu](mailto:rohan.chaudhury@tamu.edu)

**UIN:** 432001358

**List of people I have worked with:** None

**Question 1: Consider the following segment table:**

**Segment Base Length**

**0 219 600**

**1 2300 14**

**2 90 100**

**3 1327 580**

**4 1952 96**

**What are the physical addresses for the following logical addresses?**

**hint: the logical address is formatted as: segment number, offset**

1. **1, 10**
2. **4, 112**

**Answer:** We know that:

Physical address = base address of segment number + offset

So:

1. Logical address of 1,10 = 2300 + 10 = 2310
2. Logical address of 4, 112 will cause an error. Reason: The length of segment 4 is 96 while the offset given is 112. Since offset is greater than the length of the segment thus it will cause an error.

**Question 2: A computer has four page frames. The time of loading, time of last access and the R(referenced) and M(modified) bits are shown below (the times are in clock ticks).**

**Page Loaded Last Ref. (ticks) R bit M bit**

**0 126 280 1 0**

**1 230 265 0 1**

**2 140 270 0 0**

**3 110 285 1 1**

1. **Which page will Not Recently Used (NRU) replace?**
2. **Which page will FIFO replace?**
3. **Which page will Least Recently Used (LRU) replace?**
4. **Which page will second chance replace?**

**Answer:**

1. NRU replaces the page which has not been referenced or modified first. Hence NRU would replace **Page 2** first.
2. FIFO will replace the page which has the earliest time of loading. Hence it will replace **Page 3** with time of loading 110 ticks.
3. LRU will replace the page which was referenced the earliest. Hence **Page 1** which has the earliest time of reference (265 ticks) will be replaced by LRU.
4. Second chance will replace the page with the earliest time of loading but with the R bit set to 0. Hence **page 2** will be replaced by second chance.

Reference: <https://www.geeksforgeeks.org/second-chance-or-clock-page-replacement-policy/>

**Question 3: If you implement a correct solution to the multiple producer-consumer problem, why should your output (and therefore your order of producers and consumers executing) vary between runs?**

**Answer:**

A correct solution to the multiple producer-consumer problem would include a lock that limits the access to the buffer which is the critical section in this case. The lock would ensure only one producer or consumer has access to the buffer at any given time. Ordering of the producer or consumer with regards to when they are allowed to access the buffer is not defined in this correct solution. So, any producer or consumer can access the buffer at any given time in a serial manner provided that it is not locked by some other producer or consumer and can hold the buffer as long as it requires to complete its work. Once it has done its work, it releases the lock and again any producer or consumer can access and block the buffer. Since there is no order as to which the producers or consumers are accessing the buffer, our output will vary between runs depending on which producers and consumers are accessing the buffer in that particular run.

**Question 4: Why are output files for the printer normally spooled on disk before being printed?**

**Answer:** In general, the rate at which a printer is printing is slower than the rate at which it has received the instructions and output files to print from the system applications. So, if multiple print instructions and output files are needed to be sent from the system to the printer without spooling, the system would have to wait for the current print job to finish before sending the next print instruction. This problem can be avoided by spooling the output files required to be printed on disk in a queue so that the system and the applications don’t have to wait for the current print job to get finished by the printer and can continue their execution. The spooled output files are then sent to the printer one by one.

Reference: <https://www.quora.com/Why-are-output-files-for-the-printer-normally-spooled-on-disk-before-being-printed>

**Question 5: What is the working set page replacement algorithm? How does the working set algorithm work? Explain locality of reference with respect to memory references and how this applies to the working set page replacement algorithm.**

**Answer:** The working set refers to the set of pages that a process is currently using. The **working set page replacement** algorithm is a page replacement algorithm that attempts to keep track of each process's working set and tries to ensure that it is in memory before allowing it to start.

A parameter, Δ, is used to define the working-set window in this algorithm. The idea is to look at all the most current Δ page references. As demonstrated in Figure 1, the set of pages in the most recent Δ page references is the working set. If a page is in active use, it will be in the working set. It will be removed from the working set Δ time units after its last reference if it is no longer in use.

For instance, given the memory reference sequence depicted in Figure 1, if Δ = 10 memory references, the working set at time t1 is {1, 2, 5, 6, 7}. The working set will change to {3, 4} by time t2.

Graphical user interface, application

Description automatically generated

**Figure 1**

According to the **Locality of reference** model, a process goes from one locality to the other as it executes. A locality is a collection of pages that are actively used together.  A running program is made up of numerous separate localities that may or may not overlap. When a function is called, for example, it defines a new locality. Memory references to the function call's instructions, local variables, and a subset of global variables are made in this locality. The process exits this locality when we exit the function because the local variables and instructions are no longer in use. We may come back to this locality in the future.

Figure 2 depicts the concept of locality and how it changes over time for a process. The locality at time (a) is the collection of pages {18, 19, 20, 21, 22, 23, 24, 29, 30, 33}. The locality changes to {18, 19, 20, 24, 25, 26, 27, 28, 29, 31, 32, 33} at time (b). We can see that some pages (for example, 18, 19, and 20) are part of both localities, so there is an overlap.

Diagram, schematic

Description automatically generated

**Figure 2**

The working set page replacement algorithm works on the assumption of the locality of reference, that is, a running program is made up of numerous separate localities that may or may not overlap. It assumes that the page accesses for a particular part of a process is not random but is confined in different localities, which is why using the working set page replacement algorithm makes sense as it uses the working-set window to keep a track of these localities and keeps them in memory when a particular part of the process corresponding to a particular locality is executing. This will cause the process to run without causing many page faults and also avoid the issue of thrashing.

References:

1. <https://www.informit.com/articles/article.aspx?p=25260&seqNum=9>

2. Page 421, 422 of the book ‘Operating System Concepts, 10th Edition by Abraham Silberschatz, Greg Gagne, Peter B. Galvin’