Kaya Nelson

CST – 221

Micheal Landreth

05/27/2024

GitHub Link: [Kdeshun/CST221-Week-4 (github.com)](https://github.com/Kdeshun/CST221-Week-4)

**Memory Management**

Memory Management in Computer Systems

The Memory Management Unit (MMU) plays a crucial role in managing a computer system's virtual memory. The MMU organizes the virtual memory into a logical address space and maps it to the physical address space of the physical memory.

The MMU uses paging techniques to dynamically link the virtual memory to the physical memory based on individual processes. Paging systems, such as demand paging, segmentation systems, or demand segmentation, utilize a reference string to record the addresses of memory references.

When a process calls for a page reference, the following steps occur:

1. The Translation Lookaside Buffer (TLB) is checked for the page reference.
2. If the page reference is not found in the TLB, the main memory is checked.
3. If the page reference is not found in the main memory, a page fault is triggered, and the operating system retrieves the page from the disk.

Pseudocode:

*Read in process page request*

*Retrieve base register*

*Determine physical memory address*

*Check TLB*

*If TLB miss, check main memory*

*If both TLB and main memory miss, trigger page fault*

*If page fault:*

*Determine if it's an invalid reference or a genuine page fault*

*Check other tables to see if the reference exists*

*Secure an empty frame in memory*

*If necessary, swap out the old page*

*Swap in the new frame and reset the tables*

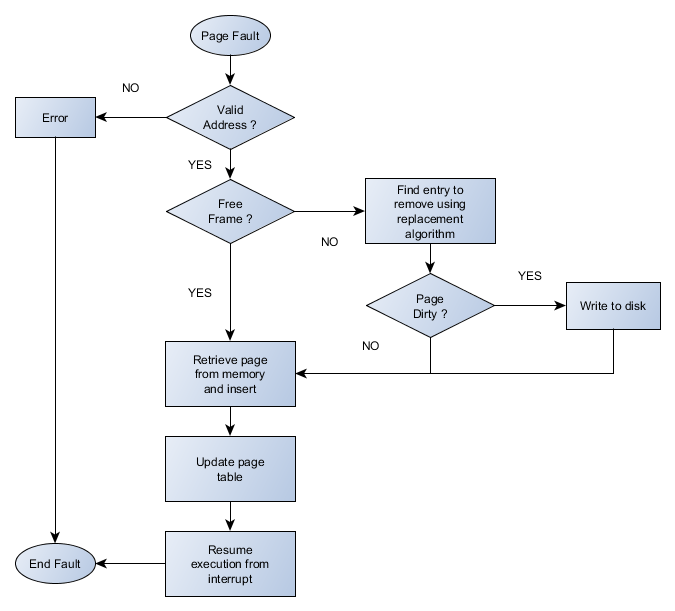
*While the operating system is retrieving the page from the disk, the CPU can be allocated to other programs.*

*Interrupt, save the current program (Program 2)*

*Restart the process (Program 1) that caused the page fault*

The key aspects of this memory management process are the use of paging techniques, the TLB, and the handling of page faults. By dynamically linking virtual memory to physical memory and swapping pages in and out as needed, the MMU enables efficient utilization of the available memory resources.

My Flowchart –



Page Fault Handling in Virtual Memory Management -

When a process references a page that is not currently present in the main memory, a page fault occurs. The Memory Management Unit (MMU) is responsible for handling these page faults and fetching the required page from the disk.

The process of handling a page fault involves the following steps:

1. The MMU determines if there is a free frame available in the main memory.

2. If no free frame is available, the MMU selects a page to be evicted from the main memory using a page replacement algorithm, such as FIFO (First-In, First-Out), Optimal Page, or Least Recently Used (LRU).

3. The MMU then fetches the intended page from the disk and loads it into the available or evicted frame.

4. The MMU updates the necessary tables, including the page table and the Translation Lookaside Buffer (TLB), to reflect the new page mapping.

5. Finally, the process that caused the page fault is restarted from the point where the fault occurred.

To improve efficiency and save time, the operating system can allocate the CPU to other programs while the page is being fetched from the disk.

Abstraction in Virtual Memory Management

The separation of policy and mechanism, known as abstraction, is a fundamental concept in virtual memory management.

In a virtual memory system, the virtual memory addresses used by a process are not directly mapped to the physical addresses in the main memory. The exact physical address of the data is not known to the process.

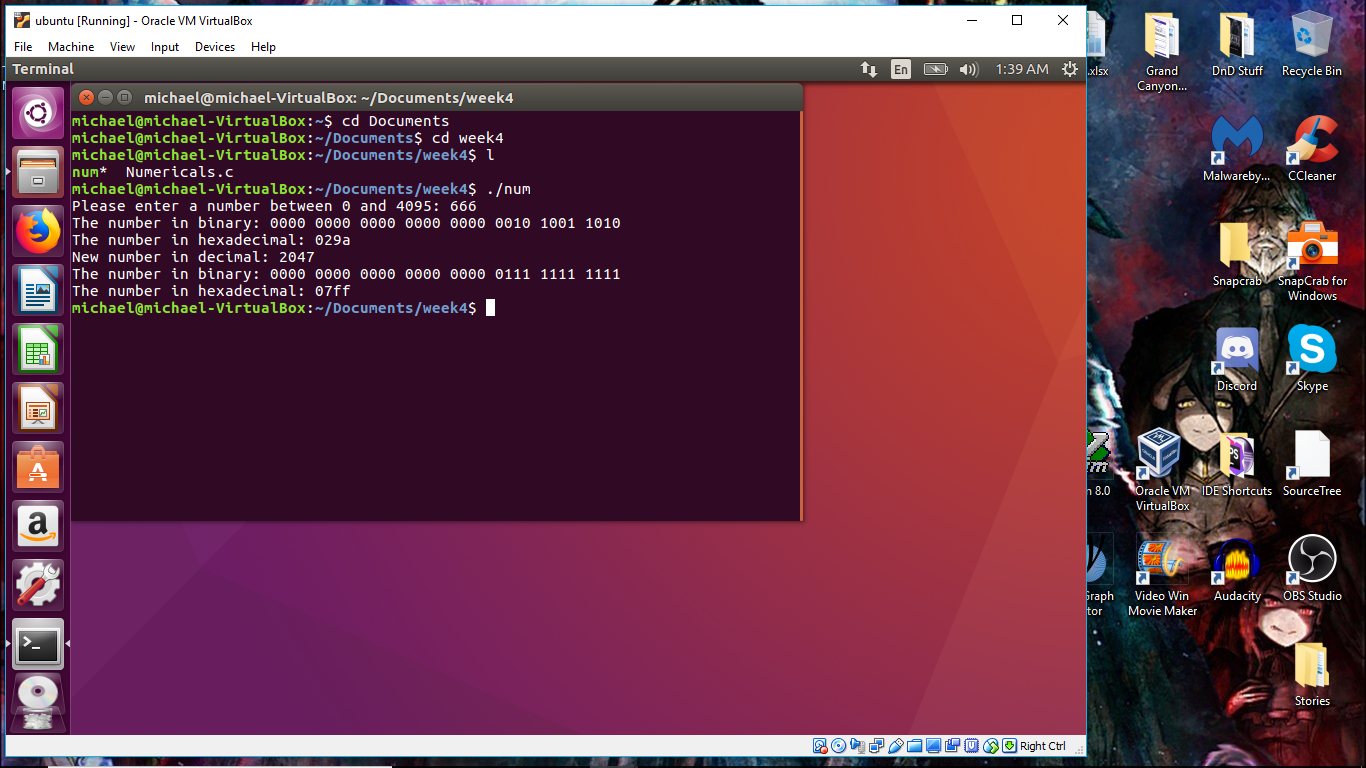
The process simply uses a virtual memory address, which is passed as a parameter to the MMU. The MMU then references its own internal tables to determine if the requested page is present in the main memory. If the page is not found, a page fault occurs.

The page fault handler then passes the page reference to the operating system's external pager function, which is responsible for retrieving the page from the disk and returning it to the page fault handler. The page fault handler, in turn, updates the page tables and the TLB, and restarts the process that caused the fault.

Throughout this process, the different components (the process, the MMU, the page fault handler, and the external pager function) are abstracted from each other. Each component only knows its own responsibilities and the parameters it receives, without having any knowledge of the internal workings of the other components.

This abstraction is what enables the virtual memory system to function effectively, allowing the separation of policy (how the system should behave) and mechanism (how the system actually works).

Programs Results –



Analysis of the Programming Assignment

The programming assignment you completed was much more straightforward than you initially expected. The use of unsigned int data types and bitwise operators helped streamline the most challenging aspects of the implementation.

Overall, the implementation process went rather smoothly for you. The incorporation of the necessary concepts, such as the unsigned int fields and bitwise operations, seemed to have simplified the completion of the assignment.

Regarding the output log requirement, if there is no explicit need for an output log in the assignment instructions, then you do not need to generate one. However, if an output log is required, please make sure to message the individual forum so you can adjust your program accordingly and avoid losing points.

The absence of an output log requirement in the assignment instructions suggests that the focus was primarily on the functionality and implementation, rather than on generating additional output. As long as your program meets the specified requirements and produces the expected results, the lack of an output log should not be an issue.

If there are any further clarifications or concerns regarding the assignment, feel free to reach out to the instructor or the teaching assistants through the appropriate channels. They will be able to provide more guidance and information to ensure you complete the assignment successfully.

Overall, your assessment of the program being more straightforward than initially anticipated is a positive outcome, and it demonstrates your ability to adapt and navigate the challenges presented in the assignment.

References:

Abualrob, M., & Abualrob, M. (2019, January 15). *Policy vs mechanism in operating system*. 8 BIT AVENUE. <http://www.8bitavenue.com/2016/12/policy-vs-mechanism-in-operating-system/>

*Operating system - Virtual memory*. (n.d.). <https://www.tutorialspoint.com/operating_system/os_virtual_memory.htm>

*Operating System - Memory management*. (n.d.). <https://www.tutorialspoint.com/operating_system/os_memory_management.htm>

*Clear lower 16 bits*. (n.d.). Stack Overflow. <https://stackoverflow.com/questions/6867489/clear-lower-16-bits>

*Bitwise operators in c*. (n.d.). <https://www.tutorialspoint.com/cprogramming/c_bitwise_operators.htm>

*Working with Hexadecimal values in C programming language  - IncludeHelp*. (n.d.). <https://www.includehelp.com/c/working-with-hexadecimal-values-in-c-programming-language.aspx>