COMP301 - OPERATING SYSTEMS

Fall-2023
ASSIGNMENT - 3
DUE: 17th January 2024, 11:59PM

Background

Virtual memory:

Virtual memory is a memory management technique employed by operating systems to provide an illusion to users and applications that there is more physical memory (RAM) available than is physically installed on the system. It allows running programs to use more memory than what is physically available by utilizing the combination of RAM and disk space.

Address Spaces

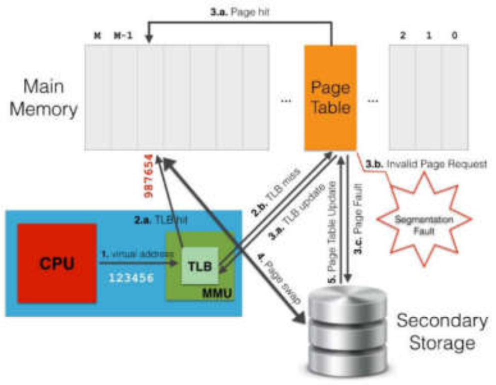
In a system with virtual memory, each process is given its own private address space, starting from address 0 and extending to the maximum address the process can use. This address space is divided into segments, including code, data, and stack.

Pages

Virtual memory is typically organized into fixed-size blocks called pages. Similarly, physical memory is divided into frames. The operating system manages the mapping of virtual pages to physical frames, allowing for efficient use of both RAM and disk storage.

Page Faults

When a process accesses a location in its virtual address space that is not currently in physical memory, a page fault occurs. The operating system responds by fetching the required page from the disk into an available physical frame.



†Figure

Page Replacement

In situations where physical memory is full, the operating system must decide which pages to keep in RAM and which to swap out to the disk. Various page replacement algorithms, such as LRU (Least Recently Used) or FIFO (First-In-First-Out), are used to make this decision.

†courtesy Gabriele Tolomei

Demand Paging

Virtual memory systems often use a technique called demand paging. Only the pages needed by a process are loaded into physical memory, and the remaining pages are brought in as needed. This optimizes memory usage and reduces the time required for process startup.

Task

Write a C program that simulates the creation and management of a page table.

Part 1: The program should perform the following tasks:

- Define a structure for a page table entry that includes necessary information such as page number, frame number, and status flags.
- Implement a function to initialize a page table with a specified number of entries.
- Create a function to perform a page table lookup, given a virtual page number.
- Display the contents of the page table before and after performing a lookup.

Part 2: Extend your program to simulate page faults. Implement the following:

- Generate a sequence of virtual page accesses. Use random numbers.
- Simulate page faults by handling cases where the requested virtual page is not present in the page table.
- Implement a basic demand paging strategy to load the missing page into memory.

Part 3: Implement a page replacement algorithm within your program. Choose a specific algorithm (e.g., LRU, FIFO, or Optimal) and perform the following:

- Extend the program to track page faults and page replacements.
- Implement the chosen page replacement algorithm to decide which page to replace when a page fault occurs.
- Display the page replacement decisions and the updated page table.

SUBMISSION DETAILS:

1. Upload C files named as *YourRollNumberPart1.c*, *YourRollNumberPart12.c*, *YourRollNumberPart123.c* etc.
2. Also Upload the Screenshots of outputs.
3. Always write the following at the start of code in comments:
    - Course Code + Course Name + Section,
    - Your Name
    - Roll Number
    - Date of Lab
4. Upload the files to Moodle.

†courtesy Gabriele Tolomei