CS232 Operating Systems
Assignment 3: Paging Simulator in C

CS Program
Habib University

Fall 2024
Release Date: 12 November 2024 @ 08:00AM
Due Date: 19 November 2024 @ 11:59PM

Overview

Develop C programs that simulate paging technique of memory management. Your simulator parameters should be dynamic i.e. it should work for any logical and physical address space sizes and any page size, in addition to any degree of multiprogramming. Degree of multiprogramming refers to the maximum number of processes that can reside in the main memory simultaneously.

Task: Paging Implementation

Inputs

- Physical Memory Size in bytes (e.g., 1048576 B for 1 MB).
- Logical Address Space Size in bits (e.g., 12 bits).
- Page size in bytes (e.g., 1024 B for 1 KB).
- List of `N` Processes (file paths to load in memory) (`N` = 1 .. k).

The paging simulator inputs would be through command line arguments: the physical address space size (bytes), the logical address size (bits), page size (bytes), followed by a list of processes to be loaded into memory.

Paging Simulator Run

Usage: `./paging <physical memory size in B> <logical address size in bits> <page size> <path to process 1 file> <path to process 2 file> ...<path to process n file>`
Example: `./paging 1048576 12 1024 "p1.proc" "./processes/p2.proc" "p3.proc"`
Details: This simulator will simulate a system having 1 MB physical memory, 12-bit logical address, 1 KB pages, and will load 3 processes given in the paths.

Steps

1. **Define Data Structures**:
   - Create a `PageTable` structure with entries that map logical pages to physical frames.
   - Each entry should store the frame number and a valid bit.
   - The Page Table for each process will be maintained in the respective `PCB` (see Process Management section of this file).

2. **Load Program**:
    - Parse the binary file to extract the code and data segments.
3. **Divide Logical Address Space (code+data) into Pages and Physical Address Space into Frames**:
    - Divide the process address space into pages based on the given page size.
    - Divide the physical memory into frames based on the given frame size.
    - Maintain a list of all free frames - not being used by any process.
4. **Allocate Frames and Internal Fragmentation**:
    - Map each logical page to a physical frame in memory. Update the page table accordingly.
    - Allocate the page to the first free frame found from the list of free frames. Use any data structure you want for free frame list maintenance.
    - Calculate internal fragmentation by summing up the unused space in the last allocated page.

Outputs

- Print a memory dump showing the pages and frames for each process.
- Print the free frame list.
- Display the total internal fragmentation.

Process Management

- The simulator must maintain a dynamic list of `PCBs` (process control blocks) which will have at least the following attributes:
    - Process id
    - Process size (code + data)
    - Process file name
    - Page table (must be a list of struct PTE)
- When a process file is to be loaded, create a new `PCB` and add it to the ready queue.
- Hint: A ready queue can simply be a dynamic array as the max no. of processes will be known when the simulator runs (through the command line arguments count.

File Format

- Each binary file for a process should follow this structure:
    - **Process id** (byte 0)
    - **Code Segment size** (bytes 1-2):
        - Size of the code segment (e.g., 0x0030 bytes) = 48 B
    - **Code Segment**:
        - Next 48 bytes will be loaded in the code segment
    - **Data Segment size** (2 bytes following the end of code segment):
        - Size of the data segment (e.g., 0x0100 bytes) = 256 B

  - **Data Segment** (Following the data segment size bytes):
    - Next 256 bytes will be loaded in the data segment
  - **End of Process**: 0xFF marks the end of the file. If after the data segment bytes 0xFF is not found, an exception must be generated and the process should be terminated.

Example File Content

```
10 // process id (1 byte)
00 1E // size of code segment (30 bytes)
// 30 bytes follow
00 3C // size of data segment (60 bytes)
// 60 bytes follow
FF // End of Process
```

Additional Considerations

- **Error Handling**: Ensure that binary files are parsed correctly. Handle any malformed input gracefully.
- **Testing**: Create multiple test process files to validate paging implementation.
- **Comments and Debugging**: Comment each section to explain its purpose, and implement optional debugging functions to print memory states.

Submission Instructions

Submit a single compressed zip file named `CSxxyyyy.zip` (replace with your student ID) via the Assignment 3 module on LMS by 11:59 PM on November 18, 2024. The zip file must contain:

1. A `makefile` with `compile`, `build`, and `clean` targets.
2. One C source file `paging.c`.

Penalties

You will be penalized for the following:

(a) submission does not follow the given instructions that is you have submitted files which are not required (for e.g. you are submitting test case evaluation data files) or have failed to comply to the instructions: -50% marks.

(b) code does not compile: -50% marks.

(c) code has warnings (compile with -Wall): -10% marks.

(d) makefile is absent: -10% marks.

(e) program does not work properly or crashes: -20% marks

(f) late submission: -10% marks for missing the deadline + -5% × num. of days.

(g) code readability: -10% marks for improper indentation, missing code comments, bad variable naming, etc.

Grading Rubric

- Defining required structs e.g. PageTable, PTE, PCB, etc. - 20 points
- Reading process file - 10 points
- Maintaining paging data structures in PCB - 20 points
- Maintaining ready queue of PCBs - 10 points
- Memory dump - 20 points
- Tracking free frames - 10 points
- Internal fragmentation - 10 points

Using chatGPT or any other AI Tool

You are not permitted to use any AI tool to obtain code for this assignment. The faculty may conduct a viva and use AI-detection tools to verify originality. If AI tools like ChatGPT are used, you must provide the chat history. Failure to comply or using AI-generated content without adaptation may result in a zero (0) and referral to the Office of Academic Conduct. This assignment is to be completed independently, and originality will be strictly enforced. You may refer to online resources but must not directly copy code.

Plagiarism Policy

We have zero tolerance for plagiarism. Every submission will be screened using a plagiarism detection software. Offenders will be reported to the Office of Academic Conduct and obtain zero (0). This applies even if the code is obtained from an online repository like github without proper attribution. We expect you to credit all sources used for completion of this assignment.