Lab Test 4 – Address Translation and Memory Mapped Files

Operating Systems Comp Sci 3SH3 Term 2, Winter 2018

Dr. Neerja Mhaskar

- **Submit your solution on Avenue under assessments -> assignments -> labtest4 submission folder.**
- Make sure to submit a version of your lab test ahead of time to avoid last minute uploading issues.
- Note that students copying each other's solution will get a zero. This includes (but is not limited to) sharing your solution with students taking the test during a different lab hour.
- You cannot share the lab test question with students taking the test in the next lab sections.
- Only one copy of your solution per group should be submitted.
- There is one deliverable for this lab test – labtest4.c

Outline

Assume that a system has a 16-bit virtual address with a 256 (=2^8) page size. The physical memory address is also a 16 bit address. Consider a small program that needs only 10 pages of memory.

In today's test you are to write a C program which simulates an MMU's address translation capability. To simulate a program's memory address requests we use the text file named **ltaddr.txt**. This file can be downloaded from Avenue -> Content -> Lab Tests and Solutions -> Lab Test4. This file contains a sample of **15** logical addresses generated for this test.

1. Your program should read each logical address from this file, compute its corresponding page number (p) and offset (d) using "Bitwise operators in C".

2. The page table is stored in the following binary file **'`pagetable.bin`'**. This file contains all the frame numbers in binary format. The first frame number stored in the file corresponds to the first page number, the second frame number in the file corresponds to the second page number, and so on. This file can be downloaded from Avenue -> Content -> Lab Tests and Solutions -> Lab Test4.

3. You are to memory map this file using the `mmap()` system call. After which you will copy each entry into an integer array named `page_table` using the `memcpy()` function. After copying all entries, you will un-map the file and close it.

4. All subsequent page table access should be done using the `page_table` array.

5. Your program should retrieve the frame number (f) corresponding to the page number (p) from the `page_table` array.

6. After which your program should compute the physical address using the frame number (f) and offset (d) using “Bitwise operators in C”.

Program Output: Your program should print each logical/virtual address (read from the file ltaddr.txt), its corresponding page number, page offset and physical address on the console.

Sample Output: `./labtest4`

Virtual addr is 2356: Page# = 9 & Offset = 52 frame number = 101 Physical addr = 25908.
Virtual addr is 527: Page# = 2 & Offset = 15 frame number = 10 Physical addr = 2575.
Virtual addr is 2129: Page# = 8 & Offset = 81 frame number = 33 Physical addr = 8529.
Virtual addr is 1312: Page# = 5 & Offset = 32 frame number = 9 Physical addr = 2336.
Virtual addr is 780: Page# = 3 & Offset = 12 frame number = 21 Physical addr = 5388.
Virtual addr is 297: Page# = 1 & Offset = 41 frame number = 192 Physical addr = 49193.
Virtual addr is 1117: Page# = 4 & Offset = 93 frame number = 13 Physical addr = 3421.
Virtual addr is 2622: Page# = 10 & Offset = 62 frame number = 0 Physical addr = 62.
Virtual addr is 72: Page# = 0 & Offset = 72 frame number = 17 Physical addr = 4424.
Virtual addr is 1557: Page# = 6 & Offset = 21 frame number = 176 Physical addr = 45077.
Virtual addr is 522: Page# = 2 & Offset = 10 frame number = 10 Physical addr = 2570.
Virtual addr is 1893: Page# = 7 & Offset = 101 frame number = 51 Physical addr = 13157.
Virtual addr is 1289: Page# = 5 & Offset = 9 frame number = 9 Physical addr = 2313.
Virtual addr is 2309: Page# = 9 & Offset = 5 frame number = 101 Physical addr = 25861.
Virtual addr is 317: Page# = 1 & Offset = 61 frame number = 192 Physical addr = 49213.