CS730 - Topics in OS

Assignment 1

1. [Part-1] Track number of TLB misses for a range of virtual address. Your module will be provided a virtual address range from the user code.
    a. You need to maintain a single count for the total number of TLB misses for all the addresses.
    b. Additionally, for each page in the range maintain a TLB miss count and display top pages with highest misses.
2. [Part-2] Same as question 1 but with PTI enabled.
3. [Part 3] Given a virtual address range, you need to find the number of pages in the range that are read, the number of pages that are written and the number of pages that are not accessed at all. Any page that is written shall not be accounted for reads even if it was read earlier. There are two subtasks for this part :-
    a. Maintain the list of read-only, written and unused pages and display their count whenever asked for.
    b. From the read-only lists, display the top pages that are read the most along with each of their read counts and from the written list, display the top pages that are written a lot along with each of their write counts.

Working of the module

The module starts with creating a char device (/dev/memtrack) which will be used by the user process to communicate. The module also creates multiple sysfs variables for displaying the total TLB misses (tlb_misses) for parts 1 and 2, the number of read (*readwss*), written (*writewss*) and *unused* pages counts for part 3 of the assignment. Lastly, there will be a writable sysfs variable (*command*) which you can use to specify the part itself. You can do so by running the following command in super user mode to enable part-1:-

```
echo 0 > /sys/kernel/memtrack/command
```

Similarly pass value 1 for part-2 and 2 for part-3. You need to implement the corresponding sysfs function (memtrack_command_set) for this to work.

Once the *command* is set, you can start an user process that creates a virtual memory region and pass it to the char device. An example of such implementation is given in **usertest.c** file. You should create your own user process for testing purpose and communicate to the char device in the following order :-

1. Allocate multiple pages and pass the pointer to the device using *write* call (usertest.c:52). All the pages should be present in the memory (use MAP_POPULATE in mmap).
2. Send FAULT_START (usettest.c:61) command (not to be confused with sysfs variable *command*) via *read* call to the device (although conventionally *read* API is used to read from device, here we will use it to pass commands). At this point, your module should start counting TLB misses or reads and writes on these pages depending on the sysfs *command*.
3. Access the memory in some pattern. The sample test case does so randomly but the final test cases will have some pattern to test the correctness of your module.
4. Send TLBMISS_TOPPERS command (usertest.c:79) again via *read* call. Here, your module should populate top 5 (defined as MAX_TOPPERS in module/interface.h) pages that encountered TLB miss till now. Use other commands (READ_TOPPERS, WRITE_TOPPERS) for part 3 of the assignment.
5. All the sysfs variables should be up to date from device open till device close.

Note:

1. You can assume only one thread is used for all the questions.
2. The virtual address range will be aligned with page boundaries.
3. Only one part of the assignment will be executed between every open and close of the char device. You can make use of global variables inside the template file. All the memory allocated should be freed before the module end. Anything leftover shall attract penalties. The module shouldn't crash during execution.
4. You shall modify only the template file and submit the same. If you have modified other files for debugging purposes make sure your solution works with their original versions.
5. Make sure there is no print statements on your submission.
6. Your submission will be a single file (hook.c) and DO NOT rename the same while submitting.