Report for project3

Overall design

473\_mm.c can be divided into four parts:

1. signal handling
2. FIFO method
3. Third chance method
4. circular linked list related operations

The signal handling part is firstly turning off the permissions of all the virtual memory range. Secondly register the added signal handling functions. And thirdly initialize the circular linked list data structures, also record the system parameters i.e. “vm”, “n\_frames” and “page\_size”. Since when the signal handler functions are called, these info are not passed as parameters.

Within the signal handling functions, the required info is calculated from the input. The formular is

a\_specific\_address=virtual\_memory\_start+page\_number\*page\_size+offset\_within\_page

Then enquires whether this virtual page is in the memory. If it is, then only some bits are needed to updated according to the input. If it is not in the memory, then either need to allocate a new slot within the circular linked list or evict some existed page.

For the FIFO method, the evict method is rather simple by just deleting the oldest item. For the third chance method, a separate function is written to do this. Within that function, a while(1) loop is used to traverse all the items in the circular linked list until find one item to be evicted. Besides, there are also some other details e.g. fault\_type=3 or 4. When the page is in already in the memory, mprotect function is also needed to be called for these two fault\_type(s). I have separated two methods into two independent functions.

For the circular linked list, a single one is used for two methods and I have designed it customized for this project. It has fixed capacity of “n\_frames” after the initialization stage. Later on, it is judging by another field “size” to decide how many effective items have been stored. Then “head” and “tail” field for FIFO method and only “head” field for the third chance method when load and store items from and to the circular linked list.

Challenges

The first challenge is the signal handling part. One thing is although the register REG\_ERR is pointed out in the material that can be referred to. The data structure of uc\_mcontext is not that easy to be found. Another thing is I didn’t formerly use GDB to debug the signal handler functions. By default, the GDB suggests failure just after the SIGSEGV signal occurs. At last, I find that “handle SIGSEGV nostop print” can solve this. A third thing is that I once let the program run in dead loop of handling the first read in project3.c. It was later found out that because the current mechanism is after return from the SIGSEGV signal handler function, the program is trying to executing the same place again. Then again, the program enters the SIGSEGV signal handler function in the exactly same state as the last call. I even wanted to adopt the advice listed in [https://stackoverflow.com/questions/2663456/how-to-write-a-signal-handler-to-catch-sigsegv#](https://stackoverflow.com/questions/2663456/how-to-write-a-signal-handler-to-catch-sigsegv) But finally found I forgot to use mprotect to update the page’s state, which can let the program proceed.

The second challenge is that the descriptions of the third chance method are not detailed. e.g. I don’t know how the chance bit get updated when a new item is added in the circular linked list. The provided video helps a lot. It explains how the intermediate steps are performed and helps debug my program step by step.

Trade-offs

Due to the initial understanding on FIFO and third chance method, I don’t merge the two methods into one signal handler function, but two separate functions. So, there should be much code redundancies.

Besides, due to the limited time, the interface for accessing the circular linked list is somehow not that straightforward when I take a look at the codes I wrote.

partition of work between team members

Another teammate dropped the course, so there is no partition of work.

any specifics/quirks

None.