Operating systems and multiprogramming

G-assignment 4

Alexander Worm Olsen - bdj816

Allan Martin Nielsen - jcl187

Troels Thompsen - qvw203

March 9 2014

Department of Computer Science
University of Copenhagen

TLB exception handling

We were asked to implement the handling of exceptions caused by TLB misses. We have chosen to implement this using a random inserting of a page, which is not at all effective or efficient but easy to implement.

First we removed any occurrences of the function tlb_fill, and instead implemented the functionality of tlb_store and tlb_load, where tlb_load simply calls tlb_store. Firstly we get the exception state with the pre implemented function _tlb_get_exception_state. Then we find our thread and the pagetable associated with this. With this we can look through our pagetable and find the appropriate virtual page number. If no entry in the pagetable is found then an error is thrown otherwise we write the entry randomly in the TLB.

Please note that we've chosen to send kernel exceptions into userland exceptions, e.g. they are handled in the same way.

Dynamic allocation for user processes

To complete this task we were to implement the functionality of dynamic memory allocation for user processes. First we implemented the system call memlimit. In this we find the pagletable associated with the current thread, then we calculate the number of entries to be made and add these to the valid cound of the pagetable and writes the entries to the pagetable.

In order to use the functionality of memlimit we've changed the implementations of free and malloc from the user library to use memlimit. In malloc we've changed the lists of free blocks to be of length 0 at initialising, and then in malloc, when no more blocks are available we use syscall_memlimit to create an appropriate number of new pages depending on the needed number of bytes.

The changes to free was very limited. These were automatically taken care of when we change the size of our free block list. Therefore you will find no changes done specifically in the function free.

Extended tests

We've made three separate test files to test the TLB exception handling and Dynamic memory allocation. These are found in tlbtest1, tlbtest2 and tlbtest3.

The first, tlbtest1, tests for a user process that tries to allocate to much memory, and therefore ends in the TLB exception handling, only here the memory that is to be allocated is bigger than what the TLB exception handling can manage.

The tlbtest2 file shows the example of a TLB exception handling where the process doesn't exceed the avaiable amount of memory.

The last test file, tlbtest3, shows the library functions malloc and free. We've used print statements in the implementation of the two functions to see that they worked perfectly, but these have been deleted in our hand-in, and therefore nothing is shown, but neither no errors are thrown.