Assignment 3: Memory Management

Part 1: System call to display current memory usage by the process

Due: April 8, 2017. This project will be done with your partner.

Pre-requisite: You must be familiar with how to write a system call in xv6 and memory layout of the user process on xv6.

Description:

This assignment is to be done by modifying the xv6 code. Like most of the operating systems, xv6 uses page tables to maintain the mapping between the virtual address space of a process and the physical memory of the system. This allows xv6 to multiplex the address spaces of different processes onto a single physical memory, and to protect the memories of different processes.

Each process has a separate page table, and xv6 tells the page table hardware to switch page tables when xv6 switches between processes. When a process asks xv6 for more memory, xv6 first finds free physical pages to provide the storage, and then adds Page Table Entries to the process's page table that point to the new physical pages.

Some entries in the table which correspond to pages required by the kernel appears in page table of all the user processes and always maps to virtual address between KERNBASE:KERNBASE+PHYSTOP to 0:PHYSTOP. A defect of this arrangement is that xv6 cannot make use of more than 2 GB of physical memory i.e., the address space of a process can only grow up to 2 GB. Also, xv6 does not support demand paging hence constraining the size of user programs.

To do task:

Create a new system call which displays the current memory usage of the process calling it, in terms of number of pages allocated to the process. Further display the number of pages that are accessible and writable by user program. Make sure you write several test programs which tries to allocate varying amounts of memory (until no more memory can be allocated) and record how allocation affects page table for the process. Your modifications must not prevent xv6 from functioning normally. Working code (65%) Document (15%) Test code (20%)

Tips and guidance:

1. Make sure you understand how xv6 uses a page directory and page tables to map a process's virtual memory to physical memory. In particular, understand what the different bits in a Page Directory Entry and Page Table Entry mean. Chapter 2 of the **xv6 textbook** is a useful reference. How does a Page Table Entry differ for a valid page compared to an invalid page?
2. Have a look at vm.c file of xv6 to understand how page tables are handled in xv6.

Submission instructions:

- Submissions must be done through blackboard only. No email submissions will be accepted.
- Submit a zip file with three folders (one containing xv6 files, second one containing the part 2 code and the third folder containing documentation).
- Only one submission per team is required. Please mention team members' names in a file *team.txt* and place in the documents folder.
- You need to write up a short summary of how memory is currently managed in xv6 in a file *summary.txt*, put this file in the documentation folder. Mention what happens where and why. Mention all the functions you made changes to and why did you do those changes.