Operating System Design

Fall 2018

Name:

You are in charge of designing an operating system for a server machine which will host all students of class of 2020 at Bucknell (50 students). The server will have 8GB RAM installed on it which supports frame size of 4KB and will be shared by all the students. Think about the following design specifications and how you can use the concepts that you have learnt so far in the course to address those. Finally, how can you make the system even better if you had **super powers** and were able to improve the existing methods? Think in terms of efficiency, performance and reliability of the system.

	Challenges	What concept (that you have learned so far in the course) can help you with overcoming these challenges and how	What are some shortcomings of this design technique? Can you improve it?
I. The server should be capable of running all students processes, which sum to less than 64MB simultaneously. Note that students are running different processes at different times (transient processes to/from memory).	1. managing memory efficiently among diff users 2. security	- Any memory management technique: Contiguous mem allocation: Best fit or in order to reduce external fragmentation ⇒ use segmentation. Use of base & limit registers could prevent malicious access.	
II. Each student's process address in the RAM should be hidden from other students.		Base & Limit for each process. Or with paging Only frames that are holding processes pages are accessible through valid/invalid bit	

	Challenges	What concept (that you have learned so far in the course) can help you with overcoming these challenges and how	What are some shortcomings of this design technique? Can you improve it?
III. Assuming a paging system, we would like to have a mapping between student's process page numbers and their frame number (on RAM) to be able to access the process faster. Where should we keep this information? How does this work when we are context switching between different processes running by the same student or different students?	1. where to keep the page table? 2. How does EAT changes? 3. How much memory?	- Page tables could have up to millions of entries ⇒ reside on RAM. But this will double each memory access time. - We can use a Cache (TLB) to keep some of the mappings closer to MMU ⇒ lower the latency. - we should delete the TLB (TLB Flush) after each Context Switch.	
IV. We realize that towards the end of the semester, students total RAM usage (for executing different processes) can become as big as 512MB each! We still should support concurrent execution of their processes. However, we know that their processes has independent parts, which can be partially loaded into RAM as necessary. Also almost all of them use emacs text editor when coding their projects. How can we support this with 8GB RAM?	1. How can all this fit into memory? 2. How can different pages be shared among processes?	- Virtual Memory: each process access its logical address space & it is the job of MMU + OS to provide necessary pages for the process. ? This is done through the page fault process.	