EN 600.318/418 Operating Systems: Final Exam

Assigned: May 7^{th} 2009
Due: May 10^{th} 2009

Frontmatter

You must answer all questions. Please identify yourself clearly on the answer sheet. Please identify your start date & time and stop date & time on your answer sheet. For Yes/No Why/Why-not style questions, you must pick exactly one side, and substantiate it. Ambivalent or unsubstantiated answers will receive no credit.

The exam must be submitted to cs418instructors@cs.jhu.edu by Sunday, May 10^{th} before 11:59:59 pm. We expect you will need about 2 or 3 hours to complete the exam, but there is no time limit other than when it is due. The only resources you may use are your notes, your textbook, and course reading material. Reading material and other media posted on the Syllabus are included in the course reading material category, so if you have not printed them, you may fetch them on-line. No other on-line material is allowed.

You must complete the exam in one sitting, though you may use all time given. You may take breaks as necessary, though you may not interact with anyone during these breaks. You must complete the exam entirely on your own.

I Address Spaces

(1) (5 points) 64-bit address spaces in AMD-64 long mode are fabricated from 12-bit page offsets and 9-bit table indices, and are only 4 tables in height. This creates an address space that is only 48 bits long. Next, they constrain valid addresses by requiring all 16 most-significant bits of the address match the most significant bit in the address space. This produces a large hole in the middle of all address spaces that can never contain valid mappings. What would the address space look like if this constraint were removed and the most significant bits of an address were ignored? How would this impact expected application behavior?

For the next two questions, consider the following architecture:

- Register size: 64-bit.
- Virtual Address Size: 64-bit
- Physical Address Size: 32-bit.
- Pages size is: 32768 bytes.

Consider the following address space mappings.

- 0x0140482102250000 => 0x00010000
- 0x0140482104490000 => 0x00020000
- 0x0142002118840000 => 0x00030000
- 0x1540482102250000 => 0x00040000
- 0x1540492102250000 => 0x00050000

(2) (20 points) Describe that address space using hierarchical page tables.

(3) (10 points) Describe the same address space using inverted page tables.

II Synchronization and Deadlocks

Consider the dining philosophers problem.

(1) (5 points) Use a resource allocation graph to illustrate a deadlocked scenario.

(2) (10 points) Use the banker's algorithm to illustrate a deadlocked scenario.

(3) (10 points) For each of the conditions for deadlock, alter the scenario so that deadlock is not possible.

III Disks and File Systems

(1) (10 points) Explain briefly the Shortest seek time first (SSSF) disk scheduling algorithm, and identify the problems in this approach.

(2) (10 points) How would you modify/improve the SSSF algorithm to ameliorate the above shortcoming(s)?

(3) (15 points) How would you adapt the SSSF algorithm to work in a RAID system that does (a) mirroring? (b) striping?

(4) (10 points) File systems such as LFS, WALF, etc. optimize for write performance rather than read performance. Can you think of a scenario where this model does not fit well?

IV Access Control and Security

(1) (5 points) Why is security not the job of the operating system? If not security, what is?

(2) (10 points) Recall that a domain is confined when no information can escape, though information may flow inward. Given the operations described by Lampson, describe how a domain can violate confinement.

(3) (5 points) Give a trivial example of two isolated domains in a system using capabilities for access control. (yes, this really should be trivial)

(4) (10 points) How can an application enforce a systemic security policy?

V Microkernels

(1) (15 points) Bershad and Chen argue that microkernels are inherently flawed. Hartig et al. claim that microkenrels impose relatively little overhead. Which argument do you agree with? Why?

VI Virtual Machines

The IA-32 hardware supports both segmentation and paging as mechanisms of virtualization. A simplified description is presented below:

- Virtual addresses are of the form `seg:offset`
- `seg` is one of code segment (`cs`), data segment (`ds`), or stack segment `ss` registers.
- `cs`, `ds` and `ss` store indices into a 256 entry table called global descriptor table (GDT). The entries of the GDT are called segment descriptors, which store information of the individual segments, such as base address, limit, permissions (read, write, execute), privilege (user/supervisor) and type (code, data, stack).
- All code addresses are interpreted as being from the `cs` segment (the segment identified by the `cs` register), data accesses through the `ds` segment and stack accesses through the `ss` segment.
- There is a `mov` instruction which can be used to load/store the segment registers. At the time of segment load, the hardware caches the base, limit, and other information internally. Updates to the GDT without an explicit `mov` to load a segment register have no effect on the MMU operations.

- The `offset` is a 32 bit value that specifies the offset within a specific segment.
- The segmentation hardware calculates a 32 bit *linear address* by adding the offset to the segment base addresses.
- Trying to access beyond the segment limit, or violating permissions results in a general protection (GP) fault, also called commonly as "segmentation fault."
- Linear addresses are them translated through the paging system (10 bit page directory, 10 bit page table, 12 bit offset) to obtain a 32 bit *physical address*

We now consider the problem of constructing a Type I virtual machine that faithfully emulates this hardware. The virtual machine is said to "go wrong" if is behavior differs from that of the original hardware in any way.

(1) (15 points) In class we studied the shadow paging technique to virtualize a machine that uses paging. Explain this technique briefly. In particular explain:

- How the virtual machine monitor (VMM) protects itself from Guest-OS.
- How the guest-OS is protected from guest-user processes.
- How guest-user processes are protected from each other.
- How accessed, dirty bits are propagated to the guest-OS page table.
- Any problems/limitations of this approach.

(2) (10 points) Suppose that a guest operating system does not use segmentation but only uses paging. That is, it sets the `cs`, `ds` and `ss` registers at start-up to point to segments whose size is all of addressable memory (0 to 4GB) and never changes their value. Is it possible virtualize this operating system by exploiting the segmentation facility in the hardware, without using shadow paging? If so, please explain.

(3) (10 points) Suppose that a guest operating system uses only segmentation but not paging (IA-32 provides a facility to turn off paging, where linear addresses are interpreted directly as physical addresses). Can we use a shadow paging like technique to virtualize the segmentation support for this operating system? If so, please explain.

(4) (15 points) Recall that a virtual machine will need to save the registers of a guest operating system to switch contexts to another guest. This will cause the `cs`, `ds` and `ss` registers to be reloaded when it is restored. Given this, and the fact that the hardware caches segment information at register load time, does your answer in the previous question emulate the IA-32 hardware precisely, or can it go wrong?