CS 3204
Operating Systems

Lecture 18

Godmar Back

Virginia Tech

Announcements

- Fix remaining project 2 bugs (if any)
  - Reminder: by end of semester, passing students will have provided a deliverable that achieves >= 90% test score on project 2 (or a 100% test score on project 3 or 4's regression tests.) – multi-oom not part of these
- Project 3 Help Sessions
  - Thursday 22nd 7-9pm McB 216
  - Friday 23rd 5-7pm McB 216
- Project 3 Design Milestone
  - Monday 26th 11:59pm – no extensions!
- Midterm March 29
  - See announcement + sample midterms on class website

Virtual Memory

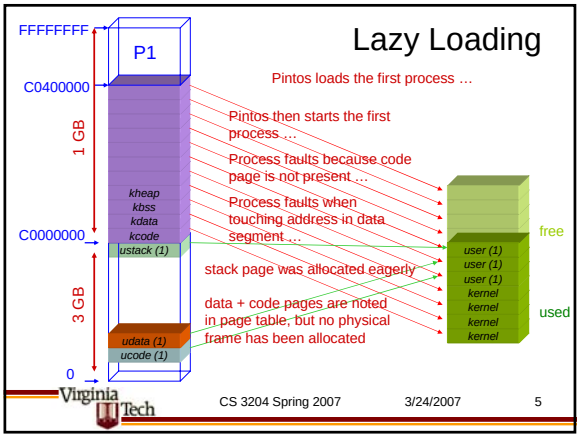
Paging Techniques

Virginia Tech

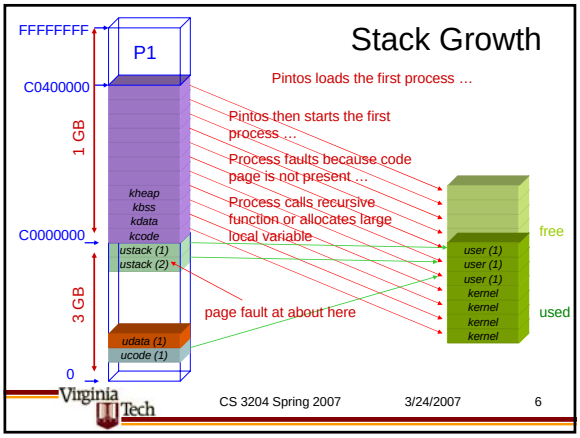
Demand paging

- Idea: only keep data in memory that's being used
  - Needed for virtualization – don't use up physical memory for data processes don't access
- Requires that actual allocation of physical page frames be delayed until first access
- Many variations
  - Lazy loading of text & data, mmapped pages & newly allocated heap pages
  - Copy-on-write

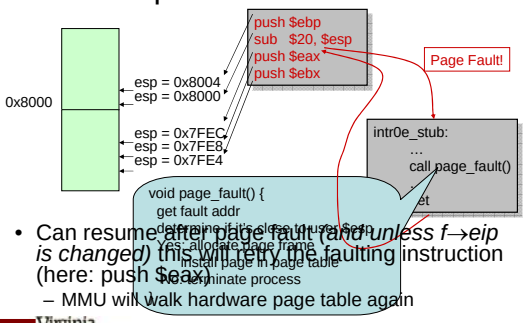
Lazy Loading



Stack Growth



Microscopic View of Stack Growth



- Can resume after page fault (and *unless f→eip is changed)* this will retry the faulting instruction (here: push $eax)
  - MMU will walk hardware page table again

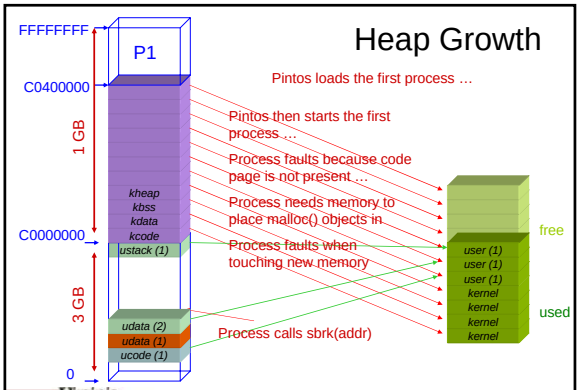
Fault Resumption

- Requires that faulting CPU instruction be restartable
  - Most CPUs are designed this way
- Very powerful technique
  - Entirely transparent to user program: user program is frozen in time until OS decides what to do
- Can be used to emulate lots of things
  - Programs that just ignore segmentation violations (!?) (here: resume with next instruction – retrying would fault again)
  - Subpage protection (protect entire page, take fault on access, check if address was to an valid subpage region)
  - Virtual machines (vmware, qemu – run entire OS on top of another OS)
  - Garbage collection
  - Distributed Shared Memory

Distributed Shared Memory

- Idea: allows accessing other machine's memory as if it were local
- Augment page table to be able to keep track of network locations:
  - local virtual address → (remote machine, remote address)
- On page fault, send request for data to owning machine, receive data, allocate & write to local page, map local page, and resume
  - Process will be able to just use pointers to access all memory distributed across machines – fully transparent
- Q.: how do you guarantee consistency?
  - Lots of options

Heap Growth



mmap()