CS 537
Lecture 10
Swapping

Michael Swift

Swapping Pages

- In this lecture:
  - When do pages get swapped
  - Where do they get put on disk?

When to swap

- The OS may write pages to disk and free memory two times
  - in the background, to make sure there are free pages in the future
  - on demand, when there is memory available
- Why have both?

Background swapping

- **Swap Daemon** (a kernel process) periodically wakes up and scans pages
  - Runs clock algorithm or adjusts working set sizes
  - Moves pages from “active” list – in use - to “inactive list” – candidate for eviction
- Clean and dirty pages treated differently
  - If a page is clean, it can be reused immediately
    - can put on free list
  - If a page is dirty, is must be written back first
    - swap daemon tries to write sets of pages at a time

Page fault to an inactive / free page

- What happens if a program references a page that is in the process of being written?
  - It can still use it without delay; page still contains data
- When should the OS clear the contents of a page?
  - When put on free list: don't have to clear it before returning it
  - When returned from allocator: can still use data on page fault to original virtual address

Where do pages go on disk?

- Kernel organizes regions of virtual memory as "areas" or segments according to how they are swapped
  - Data that gets swapped to the same file all goes to a segment
  - Multiple memory areas can get swapped to the same file in different places, or to anywhere in a "swap file" or "swap partition"
- How do you find a place to swap a page in the swap file?
  - Swap daemon maintains a "**swap map**" of
    - Which blocks on disk are in use
    - Which virtual pages are stored in those blocks

When is a page allocated space?

- When should a page be allocated/assigned space in swap?
  - When it is allocated?
    - ensures space available
    - Saves time on swapping
    - Total memory usage = swap
  - When it is evicted?
    - May never need to do it
    - Can put it in a better place, so write pages sequentially
    - total memory usage = swap + ram