Recap 1

• Multiprogramming
  – As a way of increasing CPU utilization
  – As we have more processes running, we have less chance all are waiting for I/O at once
  – Rather than $1-P$ it is $1-(P^N)$ assuming independence
  – Limited by the amount of memory

• Fixed Partitions
  – Processes go in a partition larger than they need
  – Fragmentation
Recap 2

• Relocation
  – allow programs to be used
  – change every address access

• Protection
  – base & limit

• Variable Size Partitions
  – Holes & Coalescing
  – Best Fit, First Fit, Worst Fit, Quick Fit, Buddy, Next Fit
Recap 3

• Virtual Memory
  – Pages & Page Frames
  – Page Table maps between the two
  – Page Faults
  – Page Table per Process
    • Single Level Page Table
    • Multi Level Page Table
    • Inverted Page Table
    • TLB
  – Every memory access is a page table lookup
Recap 4

• Page Replacement Algorithms
  – How does the OS decide which page goes out
  – Working Set & Locality of Reference
  – PRAs: Optimal, NRU, FIFO, 2nd Chance, Clock, LRU, NFU, Aging, etc...
Modeling Paging Algorithms

• Intuition says that more page frames should decrease the number of page faults

• Counter example, for FIFO, is called Belady’s Anomaly
  – 0 1 2 3 0 1 4 0 1 2 3 4
  – three page frames gives 9 page faults
  – four page frames gives 10 page faults
Demand Paging

- A paging algorithm is said to be demand paging if it only fetches a page when there is demand i.e. the page is referenced.
- A paging algorithm that is aware of a process's working set (might be explicitly or might be based on probabilities) can pre-fetch pages.
- Most real-world paging algorithms are demand since analysis of what to pre-fetch is very difficult.
Design Issues for Paging

- A PRA is local if the victim frame is chosen from the frames the process already has.
- A PRA is global if the victim frame is chosen from the frames of all processes.
- No PRA is truly local, since at startup, a process has no frames, and needs to get some.
- Global algorithms run the risk of thrashing (getting nothing done because the system is only doing paging).
  - Global LRU and RR scheduling
  - By the time a process runs all its pages have gone.
Local vs Global PRA

• For global algorithms, we need to strike a balance between giving each process an equal number of pages and allowing one process to use all of the pages.

• If we give each process an equal number of pages, we have the possibility that some processes have many more than their working set and some have many fewer.

• One way to decide is to give processes with many page faults more pages; we can swap out processes that cannot be satisfied.
Page Size

- Must be disk block size multiple

- Large Pages
  - Good for User I/O. I/O less likely to cross page boundaries
  - Good for Demand Paging
    - One big page paged in rather than several small ones
    - However we run the risk that the page contains info that isn’t really local to each other
  - Large internal fragmentation
  - Smaller page table

- Small Pages have the opposite properties
Implementation Issues 1

• Instruction Backup - ignore this, more modern hardware tends to just get this right

• Pinning Pages - the OS has to have to ability to keep certain pages from being paged out
  – Some OS pages if the kernel is paged (like the process table)
  – Pages that have DMA I/O in process

• Shared Pages
  – Copy on write
  – Pages need to be reference counted
  – Can be difficult to get correct paging behavior