This Week

- Tutorials on lab 3 tomorrow, Thursday.
- Homework for 6 May

  Chap 8, probs 9, 14; Chap 9, prob 10

  A system incorporates a system call, sleep(t), that puts a process to sleetp for t seconds. Describe how you would implement sleep() in Nachos. How accurate would you expect the system call to be, i.e., how close would the difference between the time a process invokes sleep(t) and is woken up be to t?

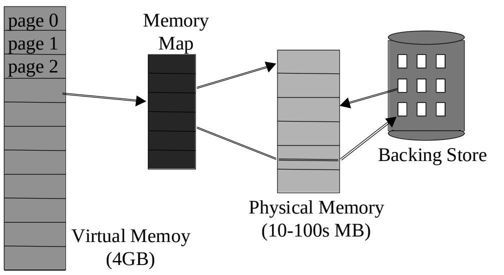
Virtual Memory: Demand Paging

Comp 305, Lecture 7

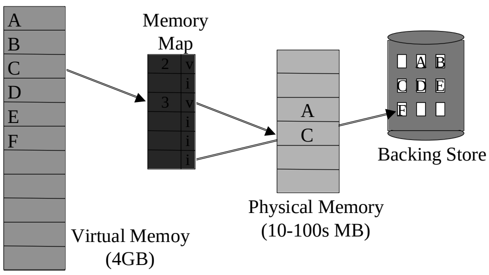
The Size and Nature of Programs

- Programs
  - Handle many rate exception conditions
  - Allocate more memory than needed
  - Have rarely used features
  - Make distinct passes
- Advantage in not allocating memory to full program.

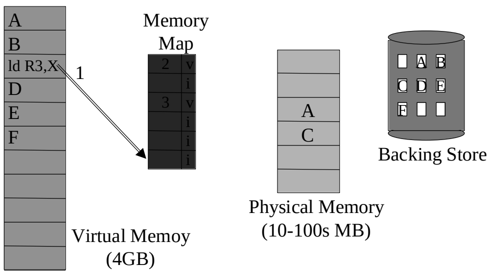
Virtual v Physical Memory



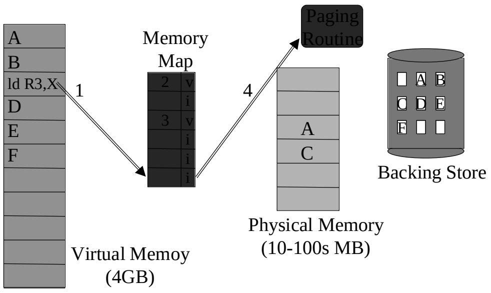
Demand Paging



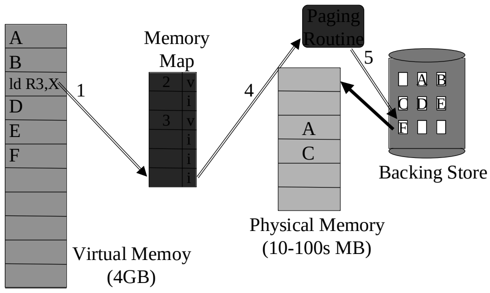
A Page Fault



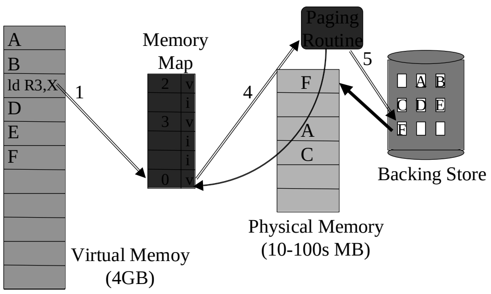
A Page Fault



A Page Fault



A Page Fault



Details

- Which page frames are free?
- How do I create more free page frames?
- When are `writes' reflected on disk?
- Retrying complex instructions

Performance of Demand Paging

- Let p = probability of a page fault
- Then effective access time
  = (1-p) * memory access + p* fault access
- One fault in 200,000 references can cause 10% slow down.
- Each process uses less memory
- Other processes use CPU during fault

Managing Memory

- Process's memory grows and grows
- How do we reclaim memory?
  - Process terminates
  - Swap process out
  - Reclaim from active process

Page Replacement

- Program with 2 code and 4 data pages

0	1	2	3	4	5	Virtual Memory

Page Replacement

- Program with 2 code and 4 data pages
- Execute from page 0

0	1	2	3	4	5	Virtual Memory

0				Physical Memory

Page Replacement

- Program with 2 code and 4 data pages
- Execute from page 0
- Reference page 2

0	1	2	3	4	5	Virtual Memory

0				Physical Memory
0	2			

Page Replacement

- Program with 2 code and 4 data pages
- Execute from page 0
- Reference page 2
- ... Reference page 4

0	1	2	3	4	5	Virtual Memory

0				Physical Memory
0	2			
0	2	4		

Page Replacement

- Program with 2 code and 4 data pages
- Execute from page 0
- Reference page 2
- ... Reference page 4
- ... Reference page 3

0	1	2	3	4	5	Virtual Memory

0				Physical Memory
0	2			
0	2	4		
0	2	1	3	

Page Replacement

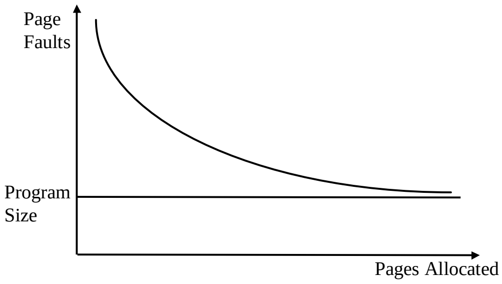
- Program with 2 code and 4 data pages
- Execute from page 0
- Reference page 2
- ... Reference page 4
- ... Reference page 3
- ...Execute from page 1

0	1	2	3	4	5	Virtual Memory

0				Physical Memory
0	2			
0	2	4		
0	2	1	3	

?

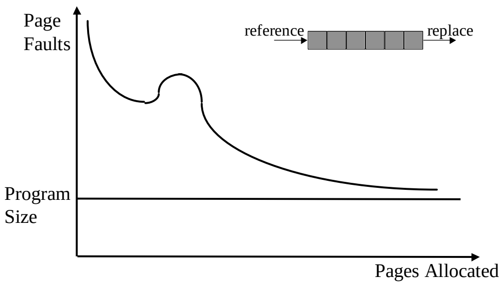
Performance of a Single Process



Page Replacement Algorithms

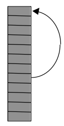
- Within a fixed memory allocation
- Optimal
  - Replace page used furthest into the future
  - Cannot be implemented
- FCFS
  - Queue implementation
  - Anomolous behaviour

FCFS Replacement Anomoly



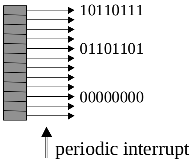
Replacement Algorithms cont.

- Least Recently Used
  - Stack implementation
  - Inefficient



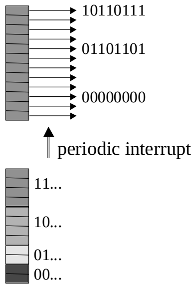
Replacement Algorithms cont.

- Least Recently Used
  - Stack implementation
  - Inefficient
- LRU Approximation
  - Set Ordering



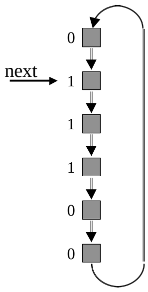
Replacement Algorithms cont.

- Least Recently Used
  - Stack implementation
  - Inefficient
- LRU Approximation
  - Set Ordering



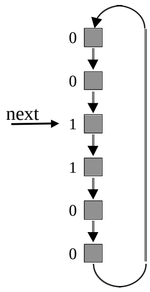
Replacement Algorithms cont.

- Least Recently Used
  - Stack implementation
  - Inefficient
- LRU Approximation
  - Set Ordering
  - Second Chance



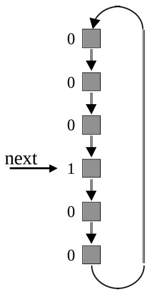
Replacement Algorithms cont.

- Least Recently Used
  - Stack implementation
  - Inefficient
- LRU Approximation
  - Set Ordering
  - Second Chance



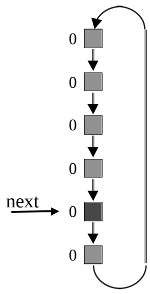
Replacement Algorithms cont.

- Least Recently Used
  - Stack implementation
  - Inefficient
- LRU Approximation
  - Set Ordering
  - Second Chance

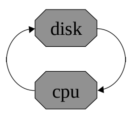


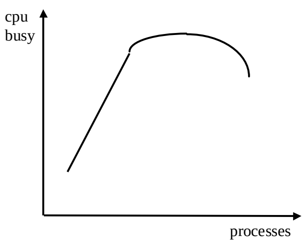
Replacement Algorithms cont.

- Least Recently Used
  - Stack implementation
  - Inefficient
- LRU Approximation
  - Set Ordering
  - Second Chance



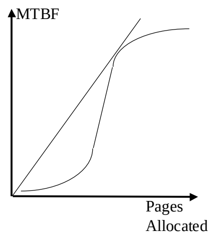
System Perfromance





Program Behaviour

- Do processes always need the same memory?
- What is optimal performance point?



Working Set Model

- Dynamically ajusts frames allocated
- $W_t(\tau)$ = set of pages referenced in $(t-\tau,t)$
- $w_t(\tau) = |W_t(t)|$ = working set size at time t

...2 6 1 5 7 7 7 7 5 1 6 2 3 4 1 2 3 4 4 4 3 4 3 4 4 4 1 3 2 3 4 4 4 3 4 ...

τ τ

W={1,2,5,6,7} W={3,4}

UNIX Memory Management

- Clusters = integral number of page frames
- Core Map

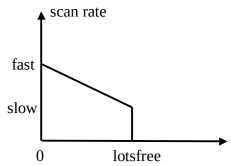
  free
  next, prev on free list
  page type (system, text, data, stack)
  pointer to owner process
  location in owner's virtual memory
  lock bit and lock wanted bit
  disk location

Demand Paging

- Page Allocated from Free List
- Logical page found and mapped
  - filled from file
  - zero filled
  - paged from swap space
- Text segments, prefetched by cluster

Page Replacement

- Free list is kept full by a clock algorithm
- Triggered when free list falls below limit



- Two handed clock