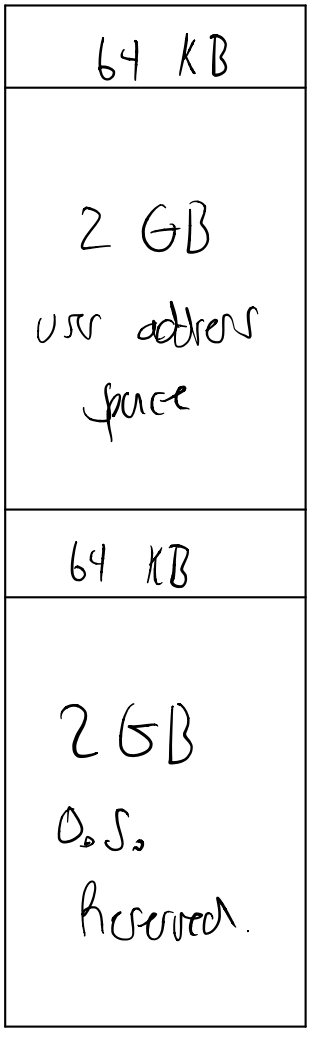
- Windows

  Intel and Amd 4KB
  Intel Itenium 8KB

32 bit windos



Null parts assigned.

for bad assignments

64 bit 8TB usr address space

- O.S. Manages the addresses by breaking it up into contigyus regions of 64 KB in size.

$8 \cdot 2^{40} \approx 2^{43} - 2^{27}$ pages

$\frac{}{2^{16}}$ = 16

- A region can be in 1 of 3 states
  - Available
  - Reserved
  - Committed - init to be used by process
    - page may be in disk or ram
    - if on disk if mapped it is in file otherwise it will be in pagefile.sys

- windows uses variable allocation with local scope.

Linux Memory Management
3 level page table

Virtual address	Global dir.	Middle dir	Page table	offset

Decs alpha process - hardware support for 3 level page table

32 bit pentium x86 supports 2 level page tables in hardware.
Set size of middle directory = 1

linux page replacement Clock algorithm.

use bit replaced by 8 bit age field
age is incremented each time page is accessed

in the background clock hand periodically sweeping through global page pool and decrements age field

Smallest use field best candidate for replacement.

Unix Memory Management

Page table entry

Page frame #	age	copy on write	Modify	ref	Valid	Protect

age - how long in memory without being referenced.
Copy on write - if one process edits page make others use original
Modify - has page been modified.
Valid - Memory resident
Protect - Are we allowed to change this page?

Disk block descriptor

swap device number	device block number	type of storage

Swap partition or actual file.

Page frame data table

page State	refrence count	logical device	block #	fr data pointer

page State - page is available
refrence count - number of process using page
logical device - device that contains page
block number - where the copy is
fr data pointer - pointer to other page from data table entries. or to list of free pages. or hash table of pages

Swap use table entry

ref count	Page / Storage unit #

- ref count - number of page table entries.

page replacement - unix use a two handed clock algorithm

11/7/18

Day 29

- Hw problem

① @ 660 + 198 = 858

ⓑ 272 + 156 = 378

ⓒ

ⓓ 986 + 444 = 1430

ⓔ 660 + 220 = 882

② 1000001 1010
offset

1110 00000 1010 = 7196

10 0010101101 page fault

101 0101111 01

= 379

7012030423032

Fifo	LRU	Clock	Optimal
7	7	7	7
70	70	70	701
701	701	701	201
201	201	201	203
231	203	203	243
230	403	403	203
430	407	423	201
420	432	023	701
423	032		
023			
7	6	5	

③ # 8.10

page = 4KB 2^{64} address space

⑤ page faults?

when an address points to a page that is not memory resident.

⑥ fewest page faults n

upper bound p

⑦

11123456 = 0001 0001 0001 001000 | 110100010110

page num | offset

(8) good demand page environment

① Stack - only push and pop so good
② hashed symbol table =
② sequential search - good chronologic
③ binary search - not good jumping.
④ pure code - probably good
⑤ vector operations - good, in chronological order
⑥ indirection - dereferencing pointer jumping around.

(9) effective access time

p = probability of page fault

$200\,ns = (1-p)\,100ns + p(.70(20_{ms} + 100_{ns}) + .3(8_{ms} + 100_{ns}))$

$200ns = 100_{ns} - 100p + p(.70(20{,}000{,}000 + 100) + .3(8{,}000{,}000 + 100))$

$100ns = -100p + p(.70(20{,}000{,}100) + .3(8{,}000{,}100))$

$100ns = -100p + p(14000070 + 2400030)$

$100_{ns} = 14000000_p + 2400000p$

$100ns = 16400000p$

$p = \left(\frac{100}{16400000}\right) = \left(\frac{1}{164000}\right)$

⑪ - Cause of thrashing.
- to detect page fault frequency algorithm.

⑩ Operating system and cycle problem

$$\text{Tef} = .99(1\mu s) + .01(.8(1us + 1us) + 0.01(1000\ \mu s + 2\ \mu s) + .001(2000\ \mu s + 2\mu s)$$
$$= 4.01\ \mu s$$

11/9/18 Day 30

Continue work on Hw problems

#12 2 byte page, an int is 4 bytes

```
for (int j=0; j<100; j++)
    for (int i=0; i<100; i++)
        A[i,j] = 0;
```

```
for (int i=0; i<100; i++)
    for (int j=0; j<100; j++)
```

2 per row
200 page faults

A[i, j] = 0;

(13) numer of page fauls with Fifo

(14) demand paging problem

$$t_{eff} = 0.8(1\,\mu s) + .18(2\,\mu s) + 0.02(20{,}000\,\mu s + 2\,\mu s)$$
$$= 401.2\,\mu s$$

(15) improving systen problem.

(a)
(b)
(c)
(d) Less multi programming
(e) memory
(f) faster hdd - thrash faster.
(g) pre paging - maybe
(h) increase page site - fewer in memory each.

off brand thrasher called thrashing with a stick of ram! :)

Scheduling

long term - Job Scheduling
medium term - Memory manager - swapping.
Short term - cpu scheduler

Goal of schedular

- fairness
- efficiency
- response time
- turn around
- throughput

Priorities

- process category
- original assigned priority
- anticipated resource needs
- running timer
- association with interactive terminal
- frequency of I/O requests
- resources held by process
- time waiting for resources.

Static priority system - can cause starvation.

dynamic priority system

non-preemptive scheduling
preemptive scheduling
> resource is cpu

Low overhead algorithms are a good idea.

Batch processing
- All data available at beginning
- no interaction
- only show information at end

- we use long term scheduling
  - user priorities.
    - resources usage estimates, ownership, ...
  - Jobs are read in
  - placed in input job queue
  - Queue foreach class or a priority queue
    - within Queue fifo

- Medium term : not needed

- Short term - when cpu ready
  - Job selected based on some kind of priority.
  - When process is running it runs until
    > it terminates
    > process requests, IO
    > stopped by system because of time out
    > higher priority process becomes ready and preemption is used.

Common Short term scheduling algorithms

1. Fifo - all processes have equal priority service in order of arrival.
2. Static priority scheduling - select process with highest priority fifo with priority class
3. Best known - Shortest job Next - SJN
   priority based on expected run time.