COMP 3430

Operating Systems

July 15th, 2019

Goals

By the end of today's lecture, you should be able to:

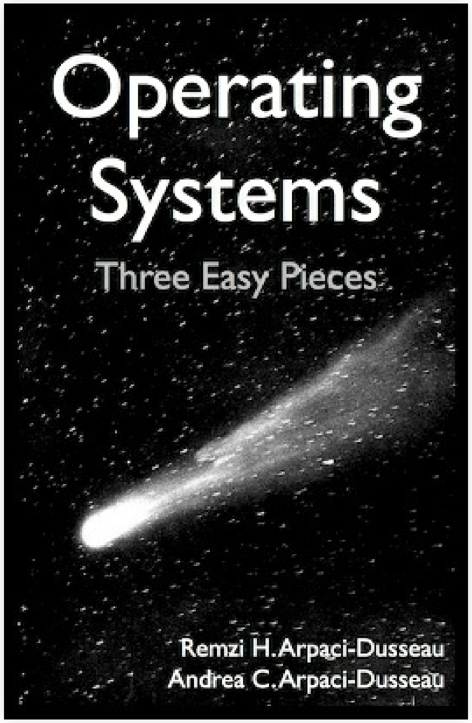
- Compare and contrast data structures for implementing paging (chapters 19,20, in-class)
- Describe a page table entry and how it supports swapping (Chapter 18, 21)
- Compare and contrast types of swapping policies (Chapter 22, in-class)
- Select appropriate memory policies for a type of workload (Chapter 22)

Pixabay License

OSTEP Q ’n A

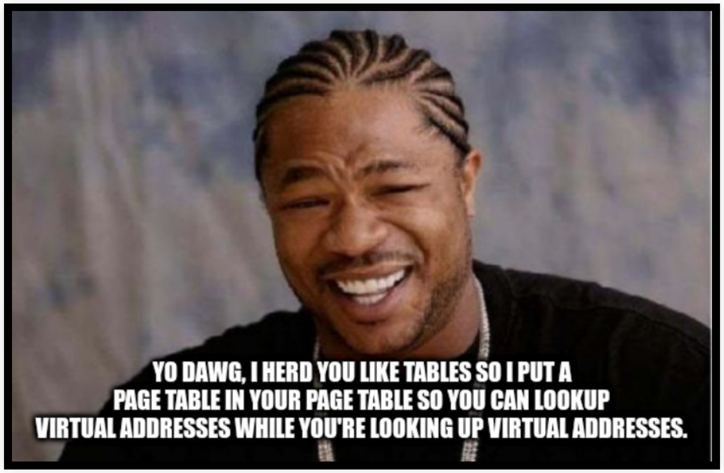
Chapters for this week:

- Chapter 21
- Chapter 22



Page directories

- A page directory is an *alternative* implementation for address translation lookups.
- We want to compare and contrast **page tables** and **page directories**.


YO DAWG, I HERD YOU LIKE TABLES SO I PUT A
PAGE TABLE IN YOUR PAGE TABLE SO YOU CAN LOOKUP
VIRTUAL ADDRESSES WHILE YOU'RE LOOKING UP VIRTUAL ADDRESSES.

Page directories.

Tables vs directories

Let’s answer some questions:

1. What is the **main difference** between the two?
2. What **data structures** can be used to implement each?
3. What **trade offs** are we making when choosing tables or directories?
4. How much **hardware support** does each require?
5. Can you use a TLB with *both*?

Paging

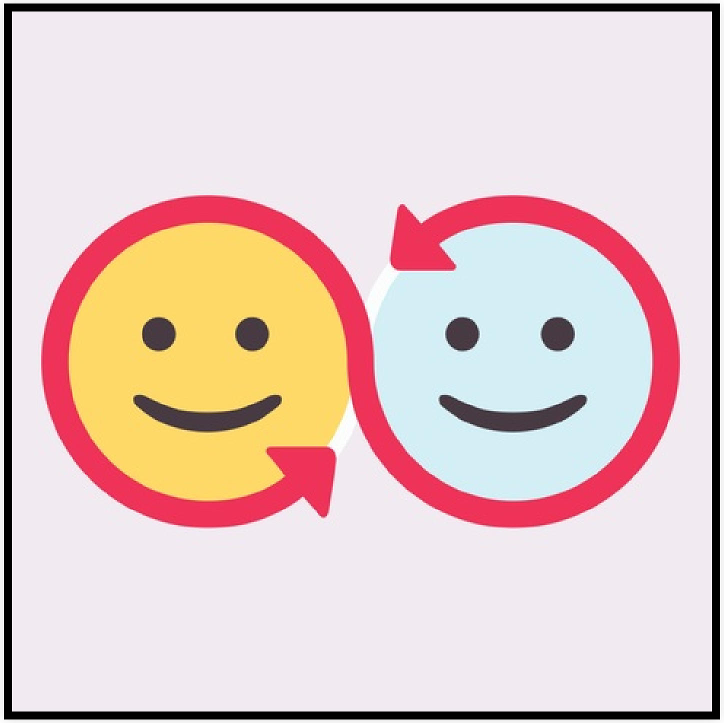
- Paging and segmentation solve the same problem.
    - How do I make processes *think* they have an entire address space by mapping virtual addresses to physical addresses?
- Paging avoids external fragmentation, but suffers from speed and size issues.
    - Speed issues are addressed with a TLB (cache)
    - Size issues *can* be addressed with different data structures.
        - … at the cost of speed/performance.

Yeesh. (Pixabay License)

Swapping

Problems with assumptions we've made:

- *Actual* physical memory sizes are limiting.
  - We made some promises we can't keep.
  - Let's quickly check out `addresses.c` again.
- Not all processes are actively running all the time.



© Laan Labs (CC BY-SA 4.0)

Oversubscription

- Idea: Let's *oversubscribe* memory.
  - We'll *pretend* that there's more memory than we actually have.
- We'll take advantage of our **disk** being huge compared to memory.
  - Let's look at the **memory hierarchy**.

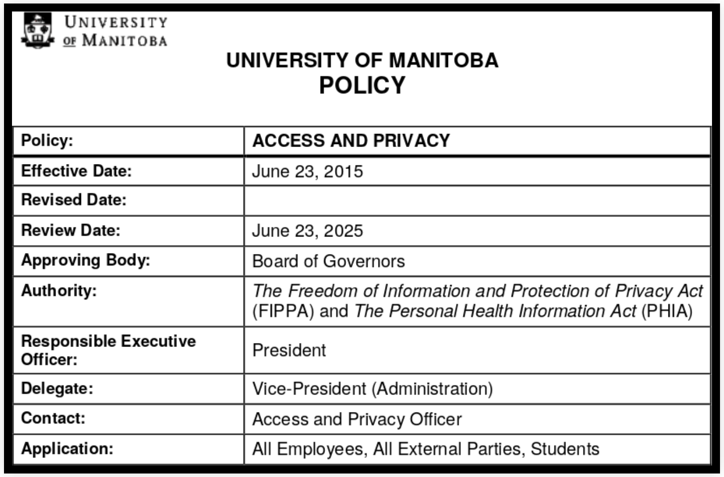
Pixabay License

Swapping

- Main idea: treat memory as a *cache*.
- We can move pages between memory and disk:
  1. Read pages into memory when requested.
  2. Write pages to disk temporarily when they aren’t used.

Replacement policies

- When our system requires a new page, we have to decide *which one* to move to disk.
- Replacement policies are algorithms to *choose* a page to evict.



UNIVERSITY OF MANITOBA
POLICY

Policy:	**ACCESS AND PRIVACY**
Effective Date:	June 23, 2015
Revised Date:	
Review Date:	June 23, 2025
Approving Body:	Board of Governors
Authority:	*The Freedom of Information and Protection of Privacy Act* (FIPPA) and *The Personal Health Information Act* (PHIA)
Responsible Executive Officer:	President
Delegate:	Vice-President (Administration)
Contact:	Access and Privacy Officer
Application:	All Employees, All External Parties, Students

A policy

Evaluating policies

Let's evaluate some policies.

- Optimal
- FIFO
- Random
- LRU

Pixabay License

Swapping

- Swapping lets us *extend* physical memory beyond its limits.
- We can oversubscribe memory, further enabling interactivity/concurrency.
- Hardware helps us out (a bit), but swapping is *mostly* an OS responsibility.

Final exam

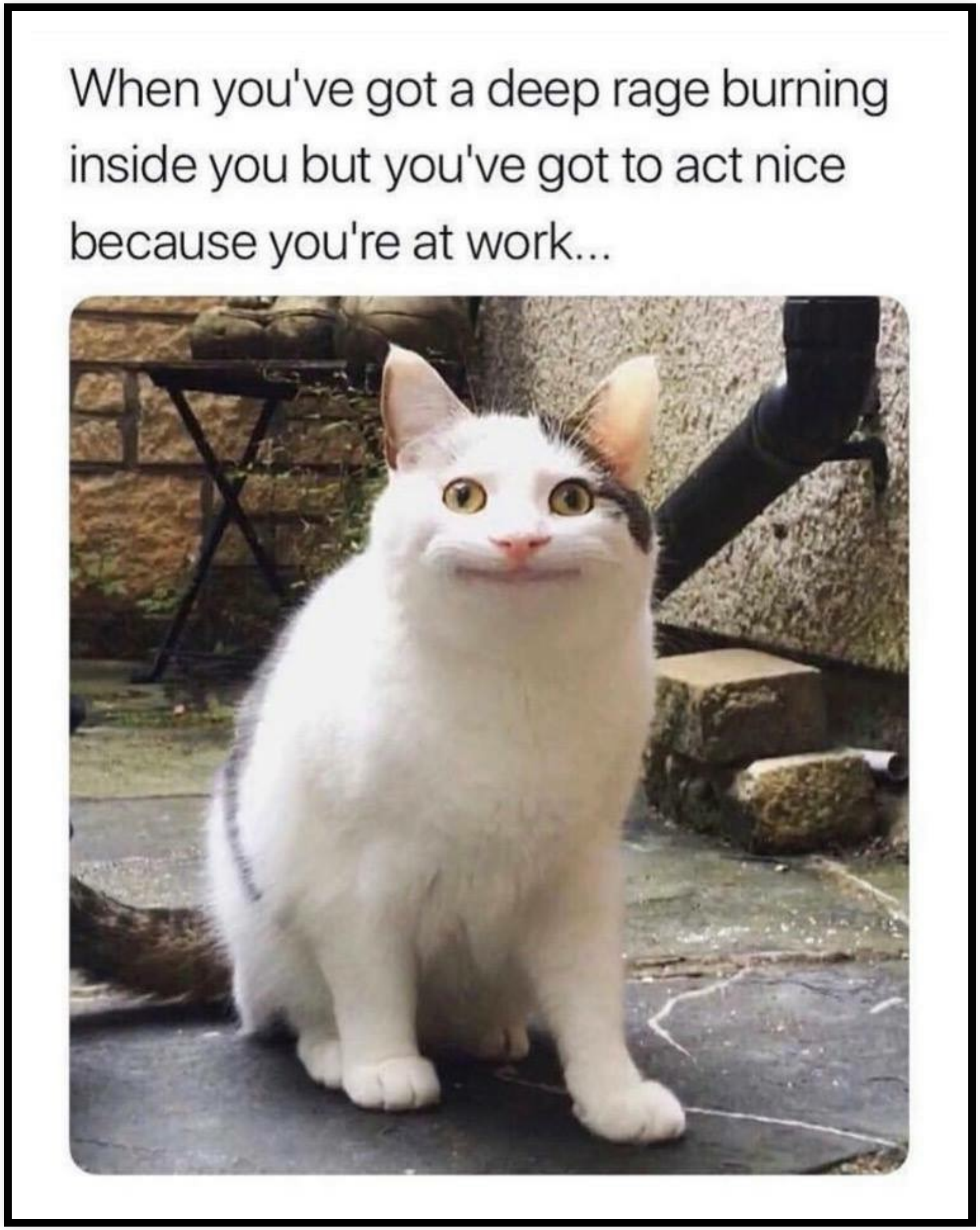
.

Goals

You should be able to:

- Compare and contrast data structures for implementing paging (chapters 19,20 in-class)
- Describe a page table entry and how it supports swapping (Chapter 18, 21)
- Compare and contrast types of swapping policies (Chapter 22, in-class)
- Select appropriate memory policies for a type of workload (Chapter 22)

Pixabay License



When you've got a deep rage burning inside you but you've got to act nice because you're at work...