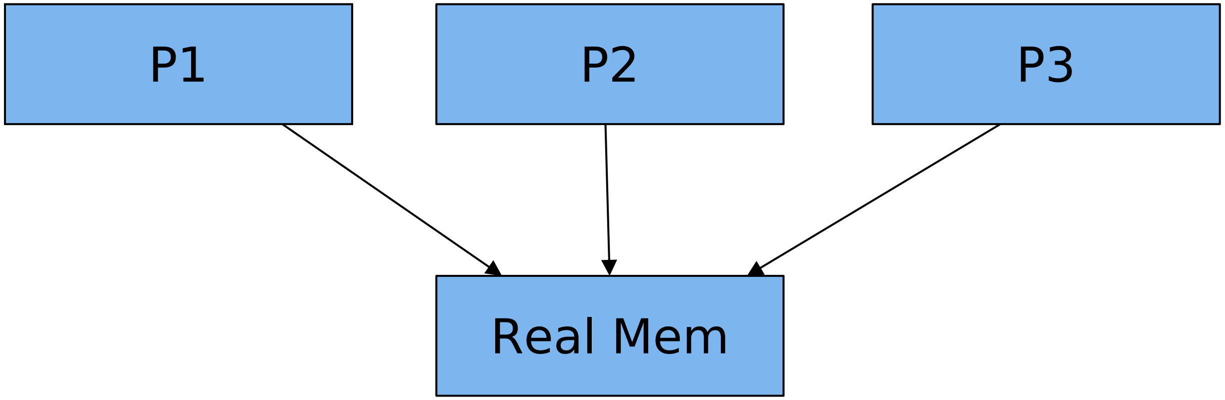
Thrashing, Demand Paging and Page Swapping

Thrashing

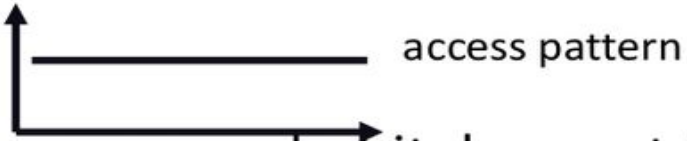
- Thrashing: processes on system require more memory than it has.



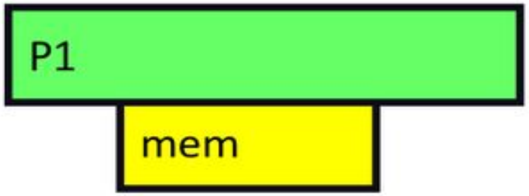
- Each time one page is brought in, another page, whose contents will soon be referenced, is thrown out.
– Processes will spend all of their time blocked, waiting for pages to be fetched from disk
– I/O devs at 100% utilization but system not getting much useful work done

- What we wanted: virtual memory with the size of disk with access time of of physical memory
- What we have: memory with access time = disk access

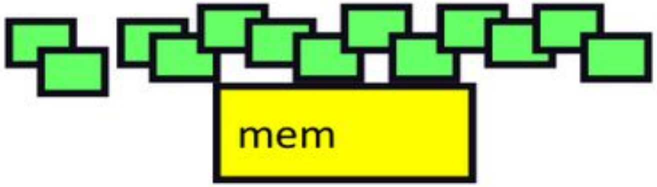
- Process(es) “frequently”reference page not in mem
  - Spend more time waiting for I/O then getting work done
- Three different reasons
  - process doesn’t reuse memory, so caching doesn’t work (past != future)

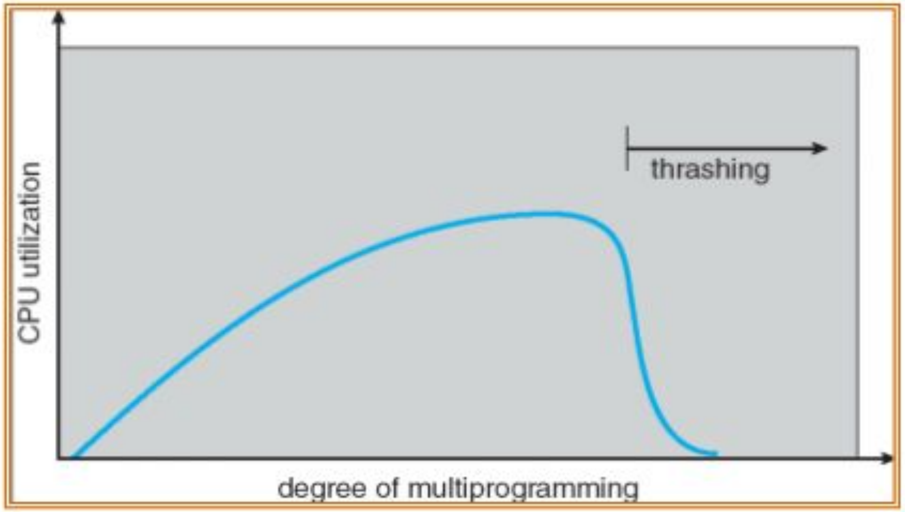


  - process does reuse memory, but it does not “fit”



  - individually, all processes fit and reuse memory, but too many for system.





- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
    - low CPU utilization
    - operating system spends most of its time swapping to disk
- Thrashing ≡ a process is busy swapping pages in and out
- Questions:
    - How do we detect Thrashing?
    - What is best response to Thrashing?

- Single process thrashing?
  - If process does not fit or does not reuse memory, OS can do nothing except contain damage. (cs140?).
- System thrashing?
  - If thrashing arises because of the sum of several processes then adapt:
    - figure out how much memory each process needs
    - change scheduling priorities to run processes in groups whose memory needs can be satisfied (load shedding)
    - if new processes try to start, can refuse (admission control)
- Careful: example of technical vs social.
  - OS not only way to solve this problem (and others).
  - "Social" solution: buy more memory.
  - Another: use 'ps' to find idiot killing machine and yell

Methodology for solving?

- Approach 1: working set
  - thrashing viewed from a caching perspective: given locality of reference, how big a cache does the process need?
  - Or: how much memory does process need in order to make "reasonable" progress (its working set)?
  - Only run processes whose memory requirements can be satisfied.
- Approach 2: page fault frequency
  - thrashing viewed as poor ratio of fetch to work
  - PFF = page faults / instructions executed
  - if PFF rises above threshold, process needs more memory
    - not enough memory on the system? Swap out.
  - if PFF sinks below threshold, memory can be taken away

Objective

- Learn demand paging, pages of data are only brought into the main memory when a program accesses them
- Learn swapping technique that uses magnetic or other media to store the state of programs that are not currently running on the processor
- Understand the use of swapping by the operating system to treat all of a program ’ s data as an atomic unit and moves all of the data into or out of the main memory at one time

Demand paging

- Pages of data are only brought into the main memory when a program accesses them
- When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory
- Instead, it just begins executing the new program and fetches that program's pages as they are referenced

Demand-paging systems advantage

- Only fetch the pages of data that a program actually uses from the disk
- If a program only needs to reference a fraction of its data during each timeslice of execution, this can significantly reduce the amount of time spent copying data to and from the disk
- Individual pages of a program's data can be brought into the memory as needed, making the limit on the maximum amount of data a program can reference, the amount of space available on the disk, not the amount of main memory

Swapping

- A related technique that uses magnetic or other media to store the state of programs that are not currently running on the processor

- In a system that uses swapping, the operating system treats all of a program's data as an atomic unit and moves all of the data into or out of the main memory at one time

- When the operating system on a computer that uses swapping selects a program to run on the processor, it loads all of the program's data into the main memory, evicting other programs from the main memory if necessary

Swapping

- Programs being executed on a computer fitting into the main memory
  - If all of the (counting both their instructions and data) fit into the main, both demand paging and swapping allow the computer to operate in a multiprogrammed mode without having to fetch data from disk

Swapping systems advantage

- Once a program has been fetched from disk, all of the program's data is mapped in the main memory

- This makes the execution time of the program more predictable, since page faults never occur during a program's use of the CPU

Swapping disadvantage over demand paging

- Systems that use swapping typically cannot use their magnetic storage to allow a single program to reference more data that fits in the main memory
- All of a program ’ s data must be swapped into or out of the main memory as a unit