Virtual Memory I

Virtual Memory:

- Pretending that entire process is in main-memory, even though only a subset of process' pages are in the main-memory at any given time.
- Loading only some of all the pages a process has into main-memory to conserve space, and maximize multi-threaded work.

Thrashing:

- When the process/computer spends majority of time only swapping memory pages in and out of memory.
- This is a feature of I/O limited processes.

Main Memory as a Cache: Concept of Virtual Memory treats main-memory as a cache for *pages*.

Memory Reference Process

- Check if the memory-address is valid or not.
  - If invalid, terminate the program
  - If valid:
    - If *page* is in main-memory, continue instruction execution
    - If *page* not in main-memory (*Page fault*), free the main-memory for a new *page* using a cache-eviction technique
- If cache-eviction had to be done, write back to disk any changes
- Request a disk-read to load the requested *page* into the main-memory
- When disk-read complete, update the *page table*
- Restart instruction execution with the requested *page* in memory

  NOTE:

  - request for memory will cause the process to get blocked if *page* wasn't in cache.
  - Processor will do something useful while the process gets the requested *page* from hard-disk

Instruction Repetion:

- Attempting to execute an instruction might lead to multiple *page faults.*
- Neccessary to be able to repeat an instruction that caused the *page fault.*
  - *e.g.*: consider `ADD C, B, A`, and assume that the `ADD` instruction, variable A, B, and C are all in different *pages.*

    - First *page fault* happens when accessing the instruction.
    - Second *page fault* happens when accessing variable B. Instruction repeats. B does not cause a page fault the second time.
    - Third *page fault* happens when accessing variable A. Instruction repeats.A does not cause a page fault the second time.
    - Fourth *page fault* happens when accessing variable C. Instruction repeats. C does not cause a page fault the second time.

Virtual Memory Performance

- **Cache Hierarchy:**
  - Main Cache behaves like a typical processor-based cache.

    - Main-memory behaves like a cache for *pages.*
    - Hard-disk plays the role of the slower memory-storage.
- **Effective Access Time:**

Copy-on-Write

- Process spawning requires *cloning* the parent to create a child process.
- Quite often, the child process will use `exec()` and change its memory contents.
    - Possibly quite slow since some of processes' pages will cause *page faults.*
    - This leads to make copying the entire parent-process seem resources wastage.
- UNIX lets the child and the parent process *share* the same *pages* until either one of the processes' attempts to modify its memory contents
- Only the modifed pages will now be actually copied.