OS, Processes, Virtual Memory

- After an application is developed – source has been written, compiled, assembled, and linked into an executable – it is ready to execute.
- Applications are executed by submitting them to the OS execution queue.
    - Type the name of the executable at the command line, or
    - Double click the application icon from the GUI
- This is also known as "launching" the app.
- Once the application is launched, it is usually referred to as a process in the context of the OS.
- The process now executes via the von Neumann instruction cycle as described previously, in which the CPU fetches, decodes, and executes the instructions stored in the process image.

What happens at application launch?

- **Executable** starts off as a file within the file system on the HD.
- Executable image must be converted and expanded into a **process image** and copied into the **swap area**, a special **raw partition** region on the HD.
    - If there is not enough swap region to hold the process image, the application fails to launch.
    - Older OSes required a single contiguous block of available swap space for the image
    - Newer OSes still require all space to be available, but may not require it to be contiguous.
- A **process descriptor** is created to represent the process.
    - The descriptor is a small summary record.
    - It points to various larger resources associated with the process, most importantly the image.
    - All the descriptors are maintained in a queue by the OS.
- The **state** of the process indicates what the OS can do with the process.
    - **Created**: state assigned to a newly launched process
    - **Running**: currently being executed by the CPU, only one process can be running at once on a single processor, each core on a multicore processor can run a separate process, assuming the OS is capable of taking advantage of the hardware's capability.
    - **Ready** (or **Waiting**): process is "swapped in" (in RAM) but not running, it is ready to run as soon as it gets its chance.
    - **Swapped and Ready**: process is not running and is "swapped out" (not in RAM). It first needs to be "swapped in" (put back in RAM by VM) to return to Ready state.
    - **Blocked**: process is swapped in but cannot currently run because it is waiting on a resource (such as input) to become available.
    - **Swapped and Blocked**: process is blocked and is also swapped out.
    - **Terminated**: process has completed, OS can now release its resources (RAM, swap space).
- OS uses various scheduling algorithms to move ready processes to the running state.

- Time slicing is usually employed here to create the illusion to the user that multiple processes are running at the same time. This will result in multiple ready processes rapidly switching between ready and running states.

Virtual Memory (VM)

- VM is a feature of the OS that allows a process to run with only a small subset of its pages currently in RAM (swapped in).
- Just like we can bring a subset of a program's instructions from RAM into cache to speed up access by the CPU, VM brings in a subset of a program's pages from swap to RAM.
- Page: a logical division of the process image. We say that the process image is broken down into n pages of x bytes each.
- Page Frame: a physical chunk of RAM that holds exactly one logical page.
- Example Address Analysis
    - VM pages work similarly to cache blocks.
    - CPU issues a logical address to an instruction or data in process image.
    - Suppose address is 64 bits and pages have 1024 bytes each
    - Then the address is divided into
        - 54 bits logical page number
        - 10 bits byte offset.
    - Suppose there are 8 GB RAM
    - There are 2^{33} addresses in RAM, 2^{23} page frames, 2^{10} bytes per page.

Supporting Data Structures

- Page Table: A simple table that translates physical page number to logical page number
- The page table can also be swapped in or out, it must be swapped in first so that it can be read in order to determine if actual process pages need to be swapped in or out.
- Translation Buffer or Translation Lookaside Buffer (TLB): like a special purpose cache devoted to virtual-to-physical page translations. If we have a TLB miss, we have to go to the full page table, which might be swapped out.
- Inverted Page Table: the Page Table requires one entry for each logical page, while the inverted page table requires one entry per physical page frame, which can be substantially smaller. But it will require hash functions to allow quick lookup of virtual addresses.

Performance

No associativity is enforced like with cache, any virtual page can be assigned to any physical frame.

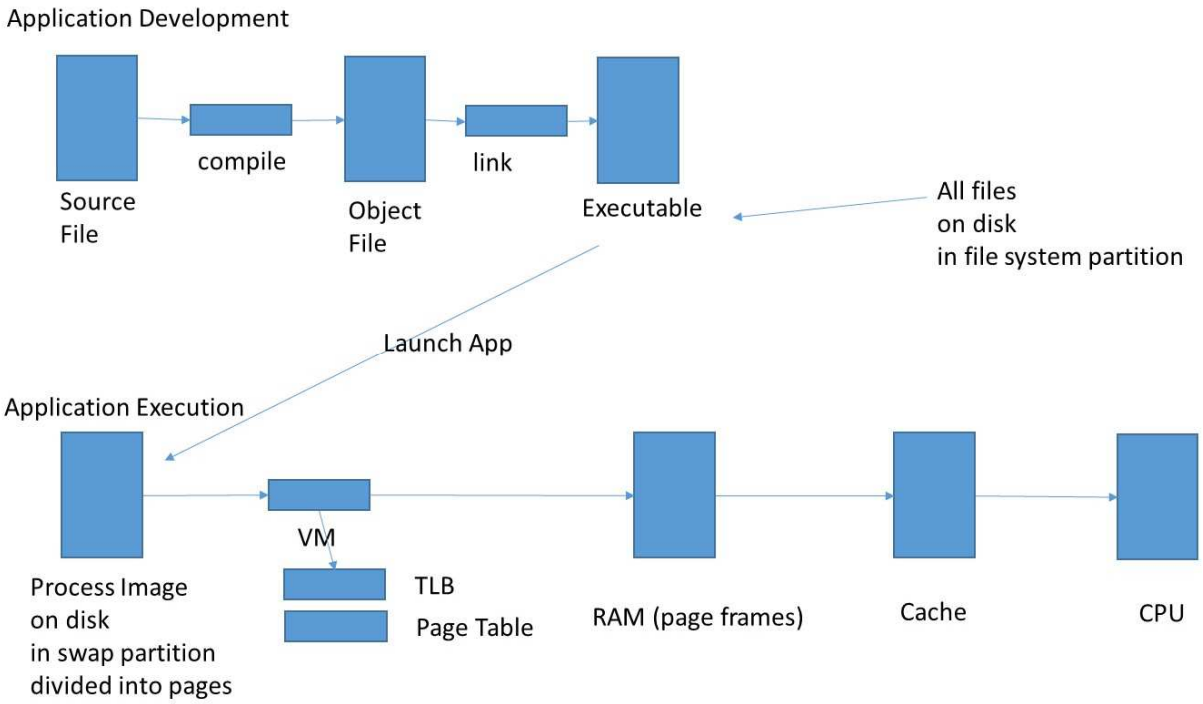
If a virtual page is in memory we say it is swapped in or resident.

If it is not in memory we say it is swapped out to disk.

If the CPU issues a virtual address and it is swapped out, the OS must pause and swap it in, potentially kicking out some resident page to make room. This is called a page fault (like a cache miss).

RSS: resident set size. This is the number of pages a process needs to have in memory to get best performance. This number is not always known in advance, it may have to be determined by monitoring performance.

Thrashing: if the OS does a bad job of managing the RSS for a process, the system may generate too many page faults, and the OS will spend most of its time moving pages in and out of memory and not executing the code. This is called thrashing.



The Big Picture: Where do applications come from? Where do they wind up?