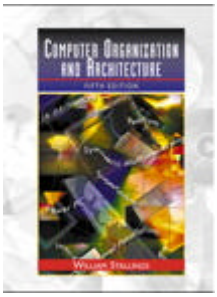
Campus de Gualtar
4710-057 Braga

UNIVERSIDADE DO MINHO
ESCOLA DE ENGENHARIA

Departamento de
Informática



Computer Organization and Architecture

5th Edition, 2000

by William Stallings

Table of Contents

II. THE COMPUTER SYSTEM.

3. ...
4. ...
5. ...
6. ...
7. Operating System Support. (30-Mar-98)

Introduction (7.1)

- From an architectural viewpoint, the most important function of an operating system is resource management
    - It controls the movement and storage of data
    - It allows access to peripheral devices
    - It controls which sets of instructions are allowed to process data at a particular time
- But it is an unusual control mechanism, in that:
    - It functions in the same way as ordinary computer software -- it is a program executed by the CPU
    - It frequently relinquishes control and must depend on the CPU to allow it to regain control

Scheduling (7.2)

- The central task of modern operating systems is multiprogramming - allowing multiple jobs or user programs to be executed concurrently.
- A better term than job (which is rooted in the old batch systems) is process. Several definitions:
    - A program in execution
    - The “animated spirit” of a program
    - That entity to which a processor is assigned

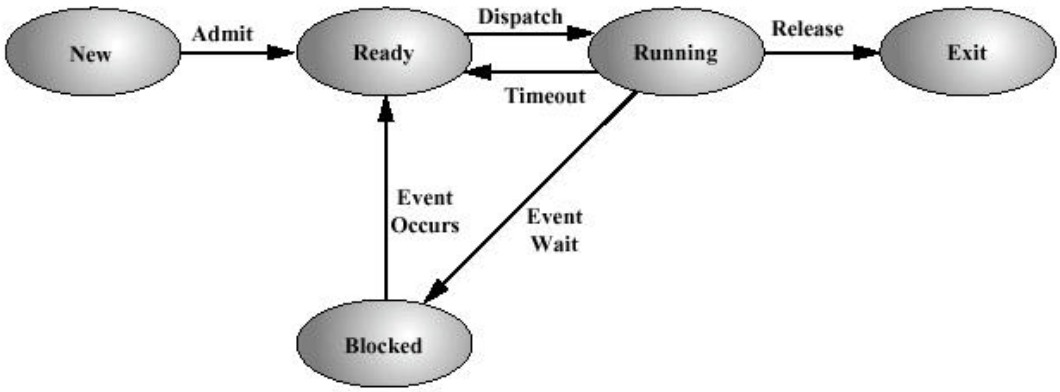
High-Level Scheduling

- High-level scheduling
    - Determines which programs are admitted to the system for processing
    - Executes relatively infrequently
    - Controls the degree of multiprogramming (number of processes in memory)
    - Batch systems can make system optimizing decisions on which jobs to add, priorities to assign, etc.
    - Once admitted, a job or program becomes a process and is added to a queue for the short-term scheduler

Short-Term Scheduling

- The short-term scheduler, or dispatcher
    - determines which process (of those permitted by the high-level scheduler) gets to execute next
    - executes frequently

- Process States - for short-term scheduling, a process is understood to be in one of 5 basic states
    - New - admitted by the high-level scheduler, but not yet ready to execute
    - Ready - needs only the CPU
    - Running - currently executing in the CPU
    - Waiting - suspended from execution, waiting for some system resource
    - Halted - the process has terminated and will be destroyed by the operating system



- Memory Pointers - starting and ending points of the process in memory (used for memory management
- Context Data - processor register values
- The OS maintains state information for each process in a process control block (PCB), which contains:
    - Scheduling Techniques
    - The OS maintains queues of PCB's for each process state
    - The long term scheduler determines which processes are admitted from New to Ready
    - The short-term scheduler determines which processes are dispatched from Ready to Running
        - Usually done round-robin
        - Priorities may be used
- At some point in time, a process in the Running state will cease execution and the OS will begin execution. 3 reasons:
    - The running process issued a service call to the OS, such as an I/O request
    - The running process (or some device associated with it) causes an interrupt, such as from an error or a timer interrupt
    - Some event unrelated to the running process causes an interrupt, such as I/O completion
- When the OS begins execution to handle an interrupt or service request (which is often handled as an interrupt), it:
    - Saves the context of the running process in the PCB.
    - Preempts that PCB from RUNNING to READY if it still has all resources it needs, or blocks to WAITING (usually in an I/O queue) if it does not
    - Handles the interrupt

  - Executes the short-term scheduler to pick the next process to dispatch from RUNNING to READY
- Organizational support
  - The mode bit in the CPU
  - Privileged instruction support in the CPU
  - The CPU timer device (a module on the bus)

Memory Management (7.3)

- Swapping
  - Used when all processes currently in the system become blocked waiting for I/O
  - Some waiting processes are swapped out to an intermediate queue in disk storage
  - Other processes (which have all the resources they need) are admitted from the long-term queue or swapped in from the intermediate queue to keep the processor busy
- Partitioning
  - Memory is partitioned among running processes
    - the OS kernel (or nucleus) occupies a fixed portion of main memory
    - remaining memory is partitioned among the other processes
      - fixed-size partitions - different sized partitions are pre-set, and processes are loaded into the smallest that will work. Low processing overhead, but wasteful.
      - variable-size partitions - a process is allocated exactly as much memory as it requires. Fragmentation can still occur to waste memory, however.
    - Requires that program addresses be logical addresses - relative to beginning of program.
  - Organizational support in the CPU for partitioning
    - base address register
    - limit register
- Paging
  - Extends partition idea by dividing up both memory and processes into equal size pieces
    - pieces of memory are called frames
    - pieces of a process are called pages
  - Each process has a list of frames that it is using, called a page table, stored in the PCB
  - The OS maintains a list of free frames that it can assign to new processes
  - Only waste is a end of a process's last page
  - Logical addresses become (frame #, rel. addr)
  - Organizational support for paging
    - page table address register in the CPU
    - cache support for page table lookups
- Virtual Memory
  - Refines paging by demand paging - each page of a process is brought in only when needed, that is, on demand.
  - Obeys the principle of locality, similar to caching
  - If a page is needed which is not in memory, a page fault is triggered, which requires an I/O operation

- Pages currently in memory may have to be replaced. In some situations this can lead to thrashing, where the processor spends too much time swapping the same pages in and out.
- Advantages
  - Less pages per process, more processes at a time
  - Unused pages are not swapped in and out anyway
  - Programs can be longer than all of main memory
    - Main memory, where processes execute is referred to as real memory
    - The memory perceived by the programmer or user is called virtual memory
- Page Table Structure
  - Basic mechanism for reading a word from memory involves using a page table to translate
    - a virtual address - page number and offset

    into
    - a physical address - frame number and offset
- Page tables may be very large
  - they cannot be stored in registers
  - they are often stored in virtual memory (so are subject to paging!)
  - sometimes a page directory is used to organize many pages of page tables. Pentium uses such a two-level structure
  - sometimes an inverted page table structure is used to map a virtual address to a real address using a hash on the page number of the virtual address. AS/400 and PowerPC use this idea.
- Organizational support
  - Translation Lookaside Buffer (TLB)
    - Avoids problem that every virtual memory reference can cause 2 physical memory accesses
      - one to fetch appropriate page table entry
      - one to fetch desired data
    - TLB is a special cache, just for page table entries
    - Result of address resolution then uses regular cache for fetch

- Segmentation
  - Segmentation allows programmer to view memory as multiple address spaces, or segments
    - Unlike virtual memory, segmentation is not transparent to the programmer
    - Advantages
      - Simplifies handling of growing data structures - OS will expand or shrink segment as needed
      - Simplifies separate compilation
      - Lends itself to sharing among processes
      - Lends itself to protection - programmer or sysadmin can assign access privileges to a segment