**The 60 was supposed to be 60% I/O wait (actually, whether sequential or parallel): 1 minute \* 0.6 is 36 seconds. That means it’s using the CPU for (60 seconds - 36 seconds) or 24 seconds out of every minute. Divide 10 min by 24 seconds and you get the sequential execution time of 1 job. Then multiply by 2 for two jobs running in sequence. The parallel time is based on the utilization formula 1 – percent wait^number of processes. Since they're running in parallel, you don't need to multiply by 2 for that one.**

Extended machine-OS abstracts interfaces to coder

Resource Manager-OS controls everything itself

**There are four principal events that cause a process to be created: System initialization. Execution of process creation system call by running a process. A user request to create a new process. Initiation of a batch job.**

The value of a semaphore is the number of units of the resource which are free. (If there is only one resource, a "binary semaphore" with values 0 or 1 is used.) The P operation busy-waits (uses its turn to do nothing) or maybe sleeps (tells the system not to give it a turn) until a resource is available, whereupon it immediately claims one. The V operation is the inverse; it simply makes a resource available again after the process has finished using it. The P and V operations must be atomic, which means that no process may ever be preempted in the middle of one of those operations to run another operation on the same semaphore.

**Microkernel-smaller/lightweight/portable/trusted codebase**

**Monolithic-more abstract/highly controllable**

Page replacement:

NRU: when a page is refer/mod the r/m bit is set-4classes

2nd chance: FIFO + r-bit check if r=1 page put to bottom

Clock: FIFO with no reordering by linked list

LRU: linked list has of all pages. Back=lru, front=mru

NFU: pages have incremented counters tracking their use

**CPU time to load page table: page size => bits (2^x),2^(address space – page size) = number of pages. Number of pages \* time to load = time to load pages**

Pthread\_join(x) makes calling process wait for x thread

**Page table size: Single level: 232 virtual address space mapped to 4KB (212 bytes) pages: the number of virtual pages is 220 (= 232 / 212). However, if the page size is increased to 32KB (215 bytes), only 217 pages are required.**

**Multi-level: first table: 2^PT1 + 2^PT2**

Real-time scheduling: while sum(actual/period) <= 1