|  |  |
| --- | --- |
| History | * Early computers allowed only one program to execute at a time   + Complete control of the system   + Access to all the systems resources (memory, disk, etc) * Modern computers allow for more than one program (process) to operate at a time |
| Process | * Definition: a program in execution * Stack: Contains temporary data such as function parameters, return addresses, local variables * Heap: Memory dynamically allocated during process run time * Data Section: Global variables * Text Section: Program code   As the stack grows down, the heap grows up. When they meet, an insufficient memory error occurs. |
| Multiple Processes | * A single program could run as multiple processes   + For example, running two copies of Word or a command line app   + These processes aren’t associated with each other and are separate copies |
| Process States | * New – Process is being created * Running – Instructions are executed * Waiting – The process is waiting for some event to occur * Ready – The process is waiting to be assigned toa processor * Terminated – The process has finished execution |
| Process Control Block | Process Control Block - each process represented in the operating system by a process control block |
| * Process State – {New, Ready, Running, Waiting, Terminated} * Program Counter – indicates the address of the next instruction to be executed * CPU registers – saves state of cpu registers for when an interrupt occurs. May include accumulators, index registers, stack pointers, . . . |
| * CPU-scheduling information – process priority, pointers to scheduling queues, . . . * Memory-management information – base and limit registers, page tables, segment tables (covered in Chapter 7) * Accounting Information – CPU time used, time limits, account numbers, process numbers * I/O status information – list of I/O devices allocated, list of open files, . . . |
| Threads | * A single thread allows a process to perform only one task at a time * A multithreaded process allows a GUI to be responsive and to perform a task at the same time |
| Process Scheduling | * Multiprogramming – goal is to have a process running at all times * Time Sharing – switch processes so frequently that users are able to interact with each other |
| Scheduling Queues | * Job Queue – all processes in a system * Ready Queue – processes residing in main memory, ready and waiting to execute * Device Queue – processes waiting for an I/O device. Each device has its own queue |
| **Figure 3.6**: Queueing Diagram  3 | |
| Schedulers | * Long-term scheduler or job scheduler – selects processes from a pool to be executed. The processes selected are loaded into memory * Short-term scheduler – selects from processes already in memory |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |