<table>
<tr><td>History</td><td>

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
</td></tr>
<tr><td>Process</td><td>

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.</td></tr>
<tr><td>Multiple Processes</td><td>

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
</td></tr>
<tr><td>Process States</td><td>

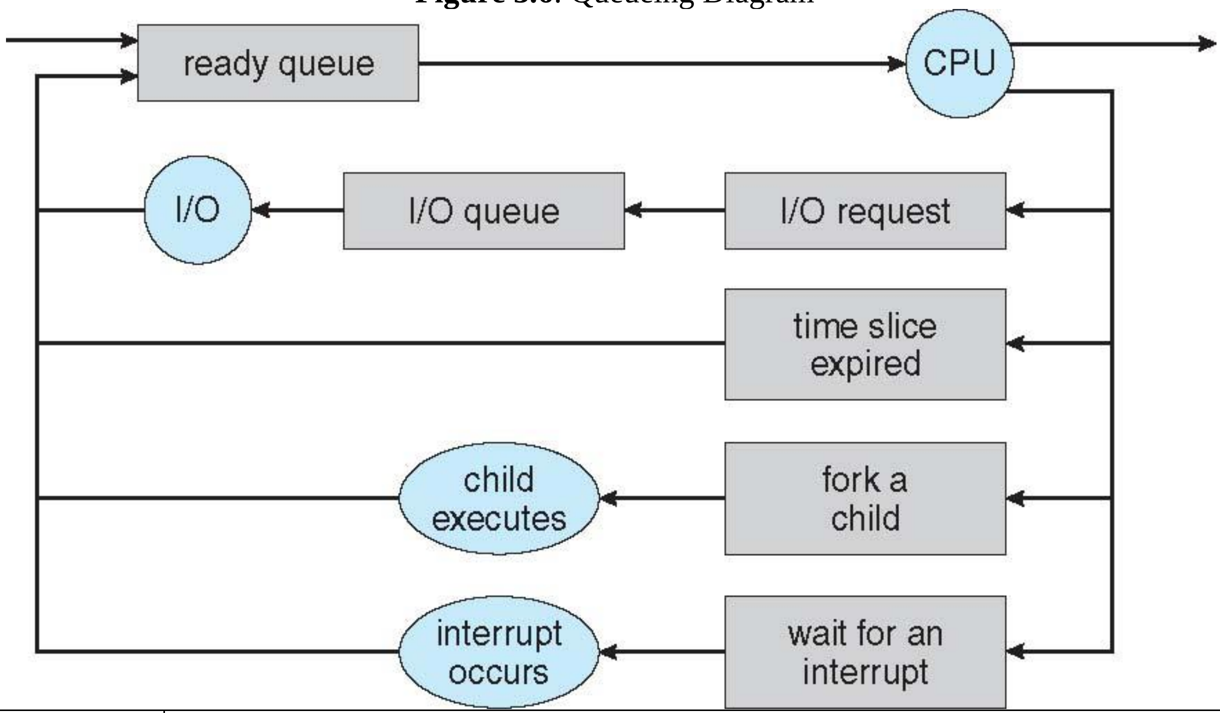
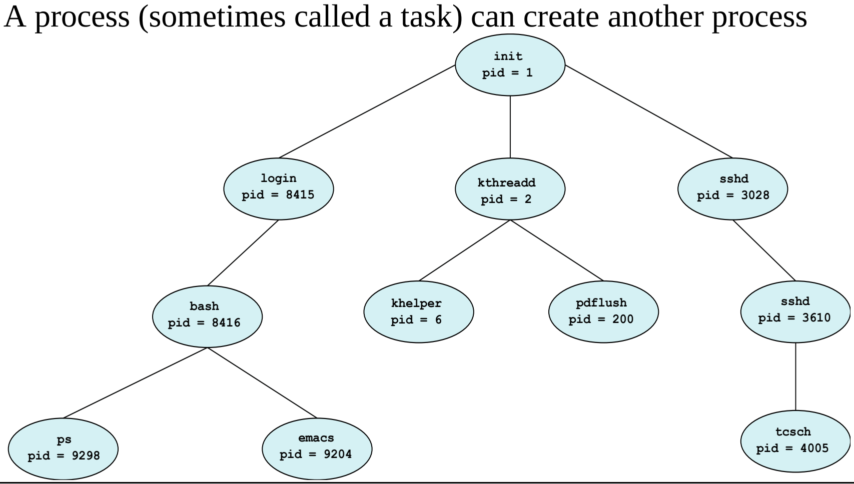
New – Process is being created
Running – Instructions are executed
Waiting – The process is waiting for some event to occur
Ready – The process is waiting to be assigned toa processor
Terminated – The process has finished execution
</td></tr>
<tr><td rowspan="3">Process Control Block</td><td>Process Control Block - each process represented in the operating system by a process control block</td></tr>
<tr><td>

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, . . .
</td></tr>
<tr><td>

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, . . .
</td></tr>
<tr><td>Threads</td><td>

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
</td></tr>
<tr><td>Process Scheduling</td><td>

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
</td></tr>
</table>

<table>
<tr><td>Scheduling Queues</td><td>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</td></tr>
<tr><td colspan="2">Figure 3.6: Queueing Diagram

</td></tr>
<tr><td>Schedulers</td><td>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</td></tr>
<tr><td>Processes</td><td>I/O bound – spends more time doing I/O than doing computations
CPU bound – spends more time doing computations than waiting for I/O</td></tr>
<tr><td>Process Creation</td><td>A process (sometimes called a task) can create another process

</td></tr>
</table>