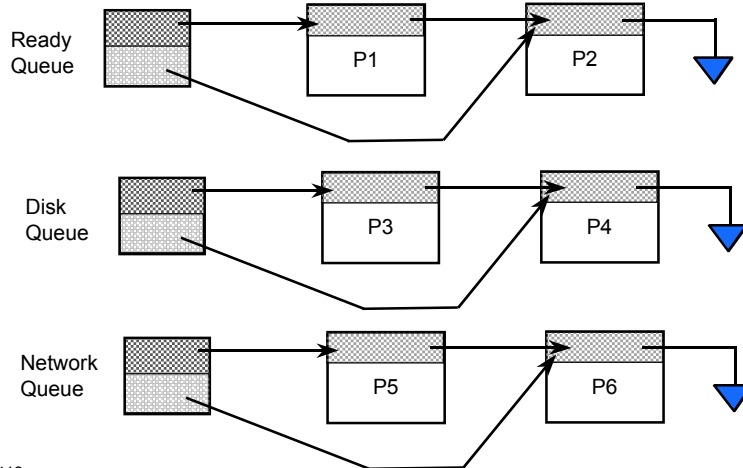


Queues of Processes

- Store processes in queues based on state



CMSC 412

1

forking a new process

- create a PCB for the new process
 - copy most entries from the parent
 - clear accounting fields
 - buffered pending I/O
 - allocate a pid (process id for the new process)
- allocate memory for it
 - could require copying all of the parents segments
 - however, text segment usually doesn't change so that could be shared
 - might be able to use memory mapping hardware to help
 - will talk more about this in the memory management part of the class
- add it to the ready queue

CMSC 412

2

Process Termination

- **Process can terminate self**
 - via the exit system call
- **One process can terminate another process**
 - use the kill system call
 - can any process kill any other process?
 - No, that would be bad.
 - Normally an ancestor can terminate a descendant
- **OS kernel can terminate a process**
 - exceeds resource limits
 - tries to perform an illegal operation
- **What if a parent terminates before the child**
 - called an orphan process
 - in UNIX becomes child of the root process
 - in VMS - causes all descendants to be killed

CMSC 412

3

Termination (cont.) - UNIX example

- **Kernel**
 - frees memory used by the process
 - moved process control block to the terminated queue
- **Terminated process**
 - signals parent of its death (SIGCHLD)
 - is called a zombie in UNIX
 - remains around waiting to be reclaimed
- **parent process**
 - wait system call retrieves info about the dead process
 - exit status
 - accounting information
 - signal handler is generally called the reaper
 - since its job is to collect the dead processes

CMSC 412

4

Dispatcher

- The inner most part of the OS that runs processes
- Responsible for:
 - saving state into PCB when switching to a new process
 - selecting a process to run (from the ready queue)
 - loading state of another process
- Sometimes called the short term scheduler
 - but does more than schedule
- Switching between processes is called context switching
- One of the most time critical parts of the OS
- Almost never can be written completely in a high level language

CMSC 412

5

Selecting a process to run

- called scheduling
- can simply pick the first item in the queue
 - called round-robin scheduling
 - is round-robin scheduling fair?
- can use more complex schemes
 - we will study these in the future
- use alarm interrupts to switch between processes
 - when time is up, a process is put back on the end of the ready queue
 - frequency of these interrupts is an important parameter
 - typically 3-10ms on modern systems
 - need to balance overhead of switching vs. responsiveness

CMSC 412

6

Process Priority

- Use multiple run queues, one for each priority
- Who decides priority
 - dispatcher - that mixes policy and mechanism too much
 - when the process is created, assign it a priority
 - have a second level scheduler (often called medium term scheduler) to manage priorities
 - mechanism is to move processes between different queues

CMSC 412

7

CPU Scheduling

- Manage CPU to achieve several objectives:
 - maximize CPU utilization
 - minimize response time
 - maximize throughput
 - minimize turnaround time
- Multiprogrammed OS
 - multiple processes in executable state at same time
 - scheduling picks the one that will run at any give time (on a uniprocessor)
- Processes use the CPU in bursts
 - may be short or long depending on the job

CMSC 412

8

Types of Scheduling

- At least 4 types (mainly for batch schedulers):
 - long-term - add to pool of processes to be executed (admittance)
 - medium-term - add to number of processes partially or fully in main memory
 - short-term - which available process will be executed by the processor
 - I/O - which process's pending I/O request will be handled by an available I/O device
- Scheduling changes the **state** of a process

CMSC 412

9

Long-term scheduling

- Determine which programs admitted to system for processing - controls degree of multiprogramming
- Once admitted, program becomes a process, either:
 - added to queue for short-term scheduler
 - swapped out (to disk), so added to queue for medium-term scheduler
- Batch Jobs
 - Can system take a new process?
 - more processes implies less time for each existing one
 - add job(s) when a process terminates, or if percentage of processor idle time is greater than some threshold
 - Which job to turn into a process
 - first-come, first-serve (FCFS), or to manage overall system performance (e.g. based on priority, expected execution time, I/O requirements, etc.)

CMSC 412

10

Medium vs. Short Term Scheduling

- **Medium-term scheduling**
 - Part of swapping function between main memory and disk
 - based on how many processes the OS wants available at any one time
 - must consider memory management if no virtual memory (VM), so look at memory requirements of swapped out processes
- **Short-term scheduling (dispatcher)**
 - Executes most frequently, to decide which process to execute next
 - Invoked whenever event occurs that interrupts current process or provides an opportunity to preempt current one in favor of another
 - Events: **clock interrupt, I/O interrupt, OS call, signal**

CMSC 412

11

Scheduling criteria

- **Per processor, or system oriented**
 - CPU utilization
 - maximize, to keep as busy as possible
 - throughput
 - maximize, number of processes completed per time unit
- **Per process, or user oriented**
 - turnaround time
 - minimize, time of submission to time of completion.
 - waiting time
 - minimize, time spent in ready queue - affected solely by scheduling policy
 - response time
 - minimize, time to produce first output
 - most important for interactive OS

CMSC 412

12

Scheduling criteria non-performance related

- Per process
 - predictability
 - job should run in about the same amount of time, regardless of total system load
- Per processor
 - fairness
 - don't starve any processes, treat them all the same
 - enforce priorities
 - favor higher priority processes
 - balance resources
 - keep all resources busy