Announcements

• Reading
  – Project #1 – due in 1 week at 5:00 pm
  – Scheduling
    • Chapter 6 (6th ed) or Chapter 5 (8th ed)
Relationship between Kernel mod and User Mode

User Process

Unique:
  Program Stack Heap

Kernel

Initial Thread

Kernel Mode thread of A user process

Idle Thread

System Calls

User Process

Unique:
  Program Stack Heap

Kernel Threads:
  Each has own stack (separate from user mode)
  Share heap with other kernel threads
  Run same program (kernel) as other kernel threads
Threads

- processes can be a heavy (expensive) object
- threads are like processes but generally a collection of threads will share
  - memory (except stack)
  - open files (and buffered data)
  - signals
- can be user or system level
  - user level: kernel sees one process
    - easy to implement by users
    - I/O management is difficult
    - in an multi-processor can’t get parallelism
  - system level: kernel schedules threads
Important Terms

- **Threads**
  - An execution context sharing an address space
- **Kernel Threads**
  - Threads running with kernel privileges
- **User Threads**
  - Threads running in user space
- **Processes**
  - An execution context with an address space
  - Visible to and scheduled by the kernel
- **Light-Weight Processes**
  - An execution context sharing an address space
  - Visible to and scheduled by the kernel
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**
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
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
Types of Scheduling

- At least 4 types:
  - long-term - add to pool of processes to be executed
  - 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
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
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
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
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.)
Process State Transitions

- New
- Ready, suspend
- Blocked, suspend
- Ready
- Blocked
- Running
- Exit

- Long-term scheduling
- Medium-term scheduling
- Short-term scheduling
- Event wait