Announcements

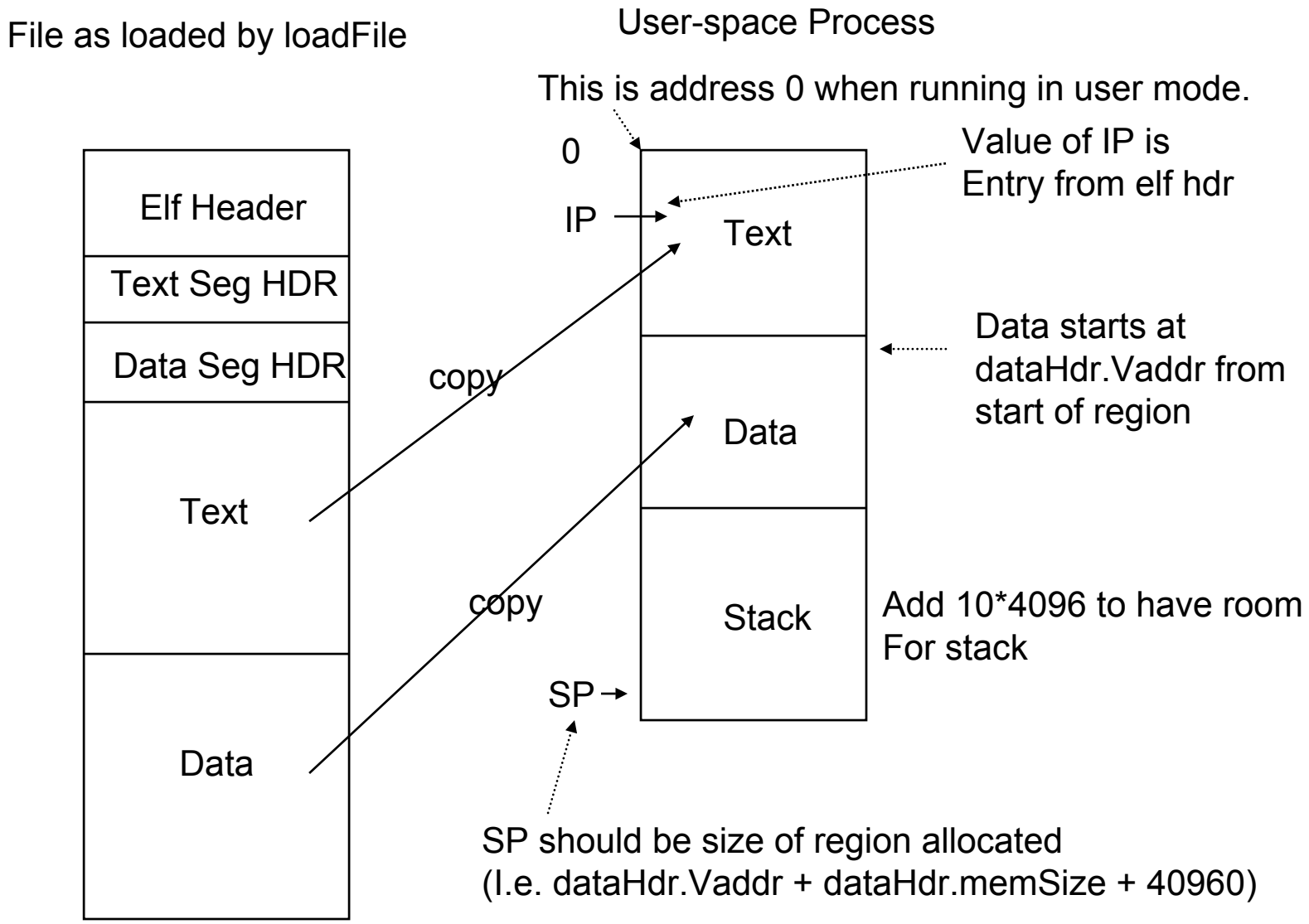
- Program #1
  - Additional info on elf file format is on the web
  - See slide from today about project issues

- Reading
  - Chapter 6
  - Chapter 7 (Tuesday)

Project Issues

- Use one TSS for entire system
- User Space Memory layout
  - Text segment
  - Data and stack are one segment
  - Stack grows down
- Kernel component of a process
  - Has own stack (in kernel memory)
- Malloc in libuser.a
  - Write an emply malloc_Atomic in libuser.c is should call Print_String to report and error and exit

User Process Memory Layout



Short-term scheduling algorithms

- First-Come, First-Served (FCFS, or FIFO)
  - as process becomes ready, join Ready queue, scheduler always selects process that has been in queue longest
  - better for long processes than short ones
  - favors CPU-bound over I/O-bound processes
  - need priorities, on uniprocessor, to make it effective

Algorithms (cont.)

- Round-Robin (RR)
  - use preemption, based on clock - time slicing
    - generate interrupt at periodic intervals
  - when interrupt occurs, place running process in Ready queue, select next process to run using FCFS
  - what's the length of a time slice
    - short means short processes move through quickly, but high overhead to deal with clock interrupts and scheduling
    - guideline is time slice should be slightly greater than time of "typical job" CPU burst
  - problem dealing with CPU and I/O bound processes

Algorithms (cont.)

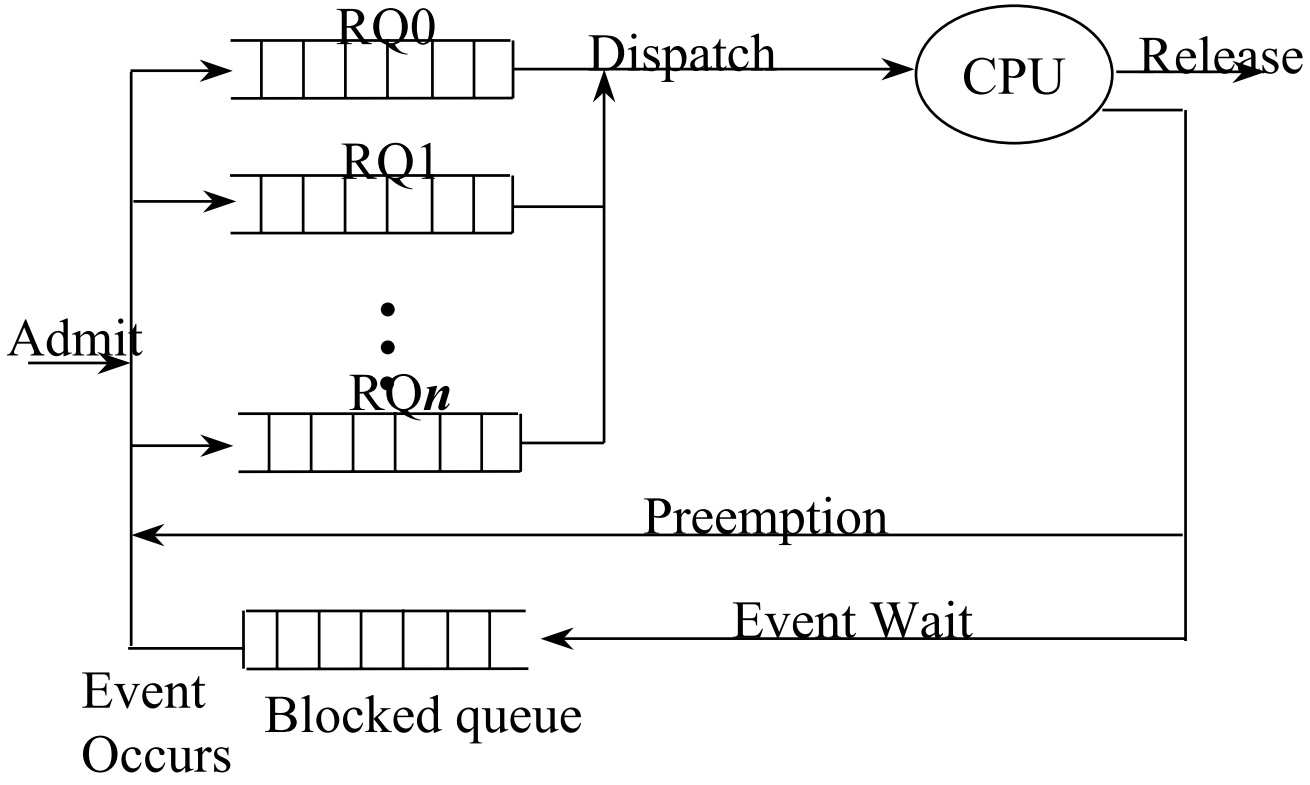
- Shortest Process Next (SPN)
  - non-preemptive
  - select process with shortest expected processing time
  - improves response time, but increases its variability, reducing predictability - provably decreases average waiting time
  - problem is estimating required processing time
  - risk of starving longer processes, as long as there are shorter processes around
  - not good for time sharing - non-preemptive

Algorithms (cont.)

- Shortest Remaining Time (SRT)
  - preemptive version of SPN
  - scheduler chooses process with shortest expected remaining process time
  - still need estimate of processing time, and can starve longer processes
    - no bias in favor of longer processes, as in FCFS
    - no extra interrupts as in RR, so reduced overhead
  - must record elapsed service times
  - should give better turnaround time than SPN

Priority Based Scheduling

- Priorities
  - assign each process a priority, and scheduler always chooses process of higher priority over one of lower priority
- More than one ready queue, ordered by priorities



Priority Algorithms

- Fixed Queues
  - processes are statically assigned to a queue
  - sample queues: system, foreground, background
- Multilevel Feedback
  - processes are dynamically assigned to queues
  - penalize jobs that have been running longer
  - preemptive, with dynamic priority
  - have ***N*** ready queues (RQ0-RQ***N***),
    - start process in RQ0
    - if quantum expires, moved to i + 1 queue

Cooperating Processes

- Often need to share information between processes
  - information: a shared file
  - computational speedup:
    - break the problem into several tasks that can be run on different processors
    - requires several processors to actually get speedup
  - modularity: separate processes for different functions
    - compiler driver, compiler, assembler, linker
  - convenience:
    - editing, printing, and compiling all at once

Interprocess Communication

- Communicating processes establish a link
  - can more than two processes use a link?
  - are links one way or two way?
  - how to establish a link
    - how do processes name other processes to talk to
      - use the process id (signals work this way)
      - use a name in the filesystem (UNIX domain sockets)
      - indirectly via mailboxes (a separate object)
- Use send/receive functions to communicate
  - send(dest, message)
  - receive(dest, message)