Betriebssysteme und Systemnahe Programmierung

Kapitel 5 • Scheduling

Winter 2016/17

Marcel Waldvogel

Process Behavior (1)

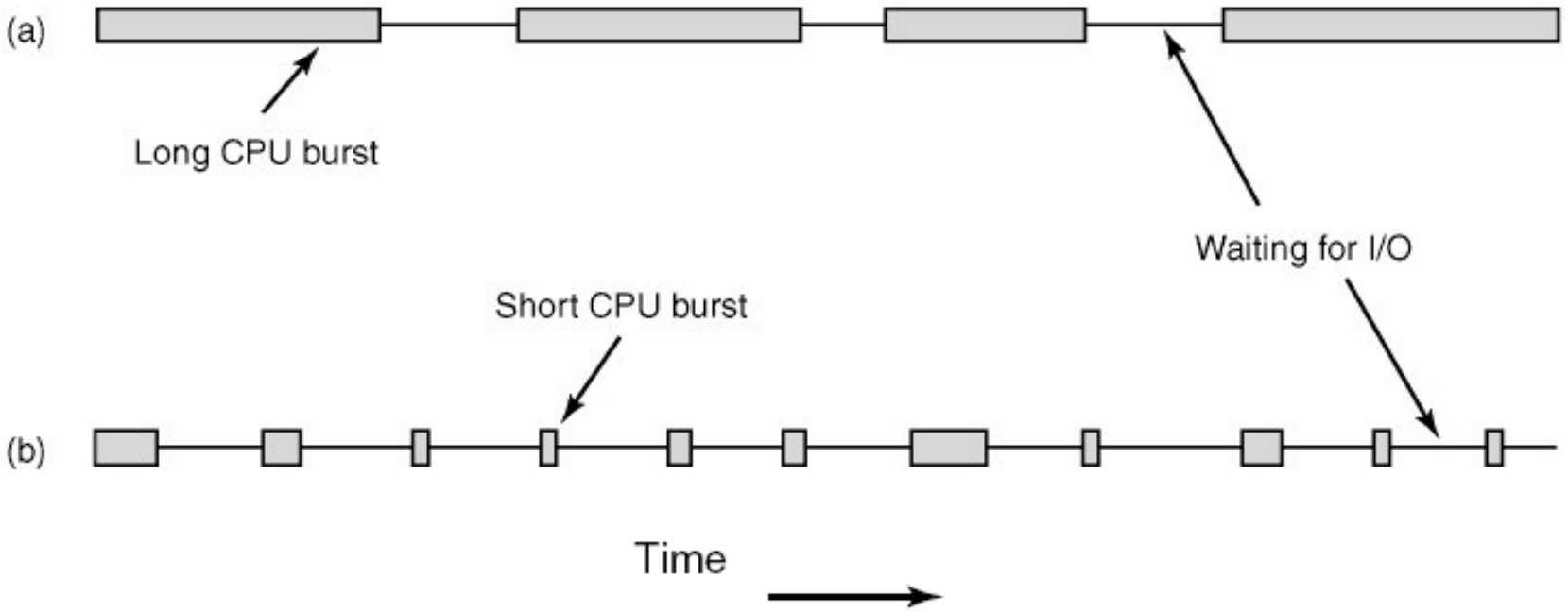


Figure 2-22. Bursts of CPU usage alternate with periods of waiting for I/O. (a) A CPU-bound process. (b) An I/O-bound process.

When to Schedule

When scheduling is absolutely required:

1. When a process exits.
2. When a process blocks on I/O, or a semaphore.

When scheduling usually done (though not absolutely required)

1. When a new process is created.
2. When an I/O interrupt occurs.
3. When a clock interrupt occurs.

Scheduling Algorithms (2)

All systems

Fairness — giving each process a fair share of the CPU
Policy enforcement — seeing that stated policy is carried out
Balance — keeping all parts of the system busy

Batch systems

Throughput — maximize jobs per hour
Turnaround time — minimize time between submission and termination
CPU utilization — keep the CPU busy all the time

Interactive systems

Response time — respond to requests quickly
Proportionality — meet users' expectations

Real—time systems

Meeting deadlines — avoid losing data
Predictability — avoid quality degradation in multimedia systems

Figure 2-23. Some goals of the scheduling algorithm under different circumstances.

Scheduling Algorithms (2)

Figure 2-24. An example of shortest job first scheduling.
(a) Running four jobs in the original order.
(b) Running them in shortest job first order.

Three Level Scheduling (1)

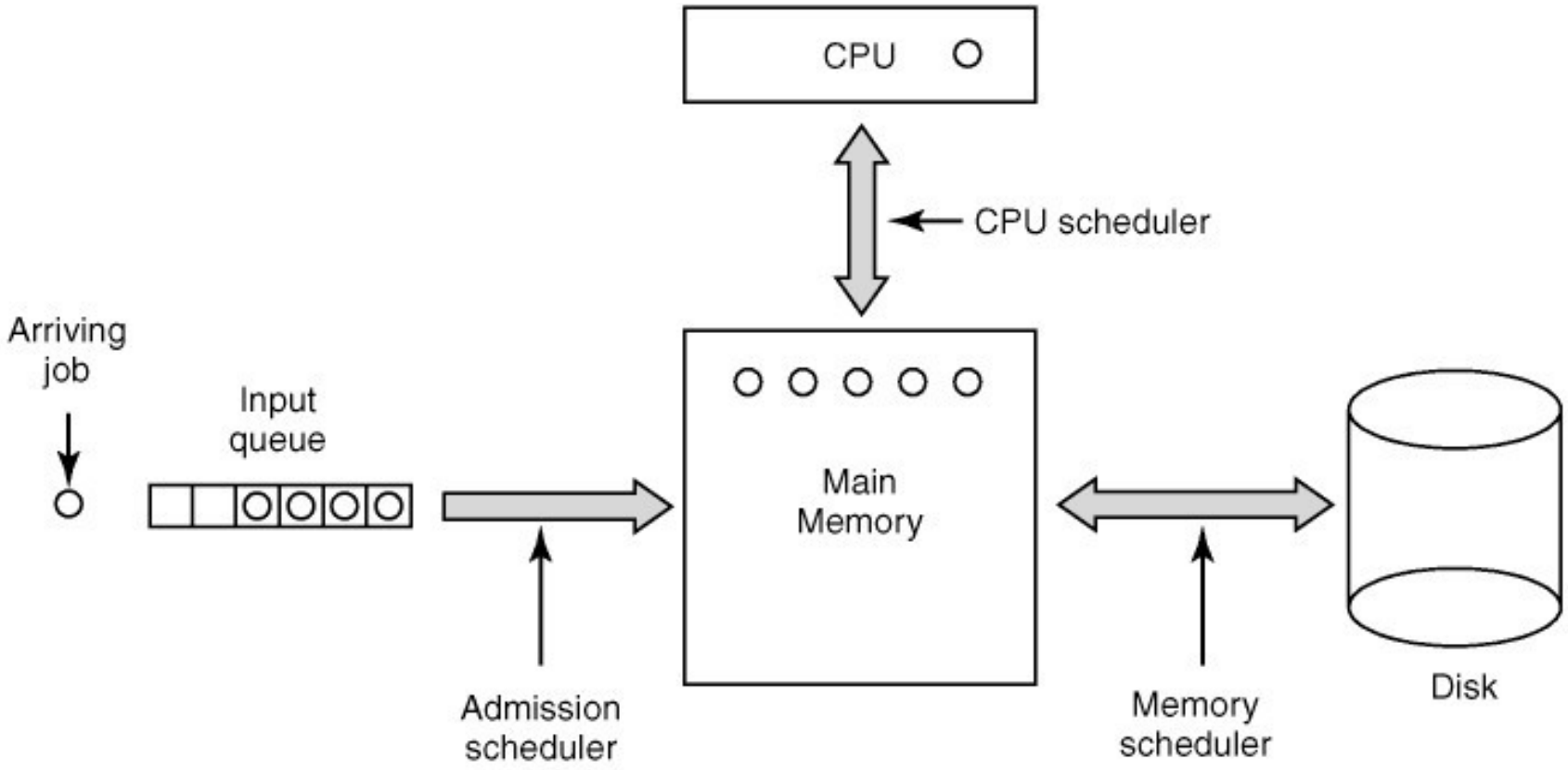


Figure 2-25. Three-level scheduling.

Three Level Scheduling (2)

Criteria for deciding which process to choose:

- How long has it been since the process was swapped in or out?
- How much CPU time has the process had recently?
- How big is the process? (Small ones do not get in the way.)
- How important is the process?

Round-Robin Scheduling

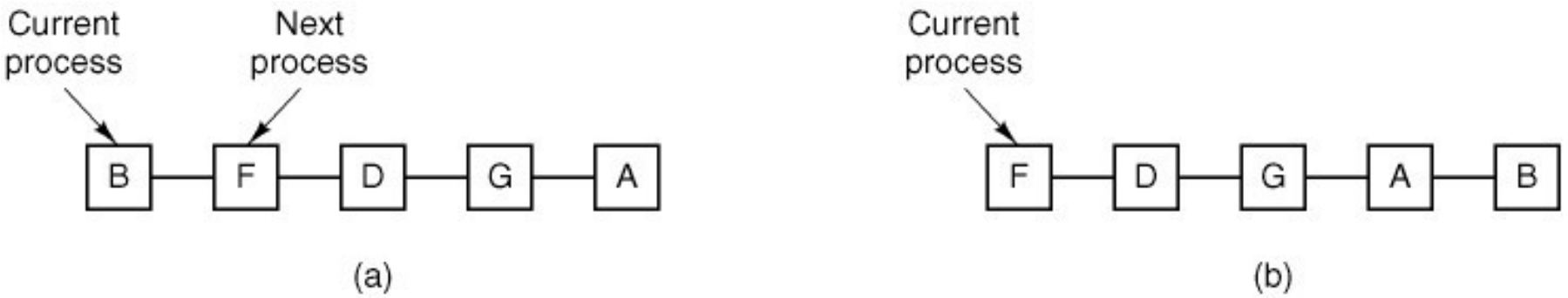


Figure 2-26. Round-robin scheduling.

(a) The list of runnable processes.

(b) The list of runnable processes after B uses up its quantum.

Priority Scheduling

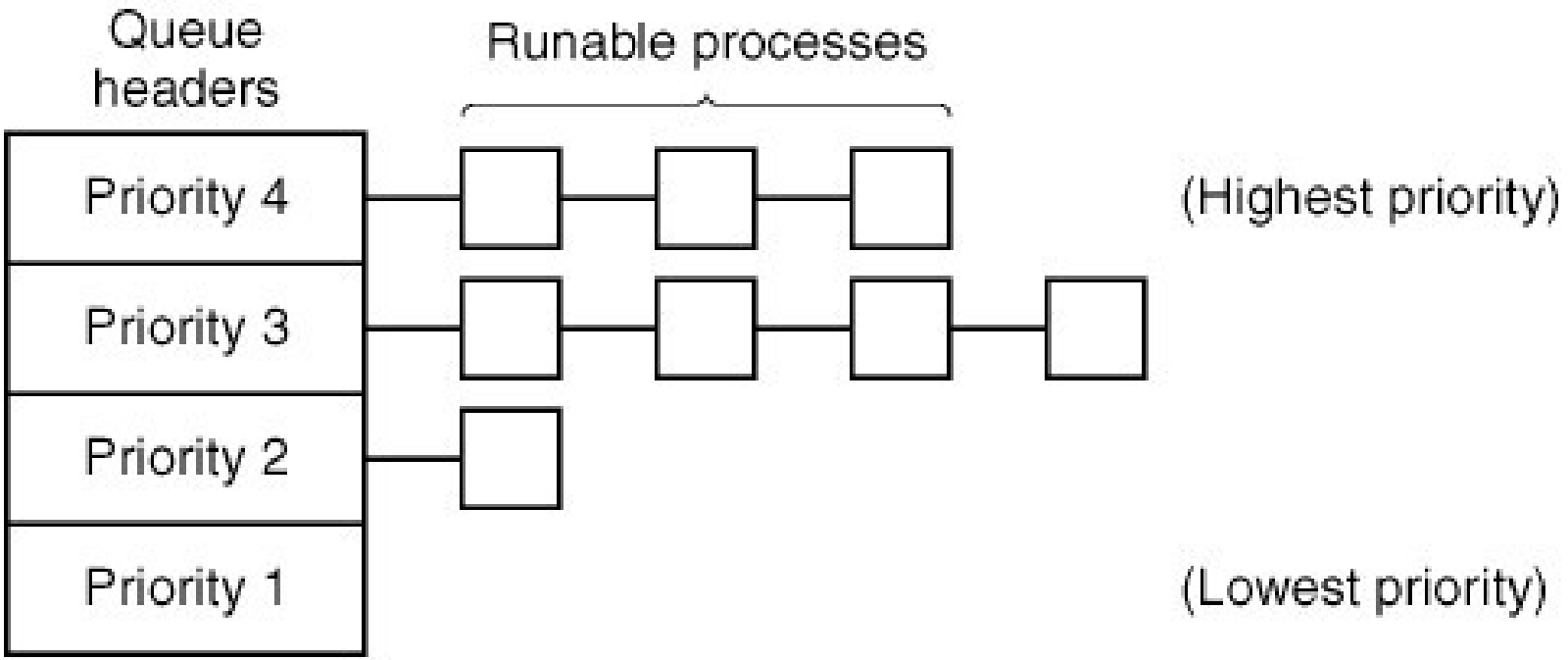
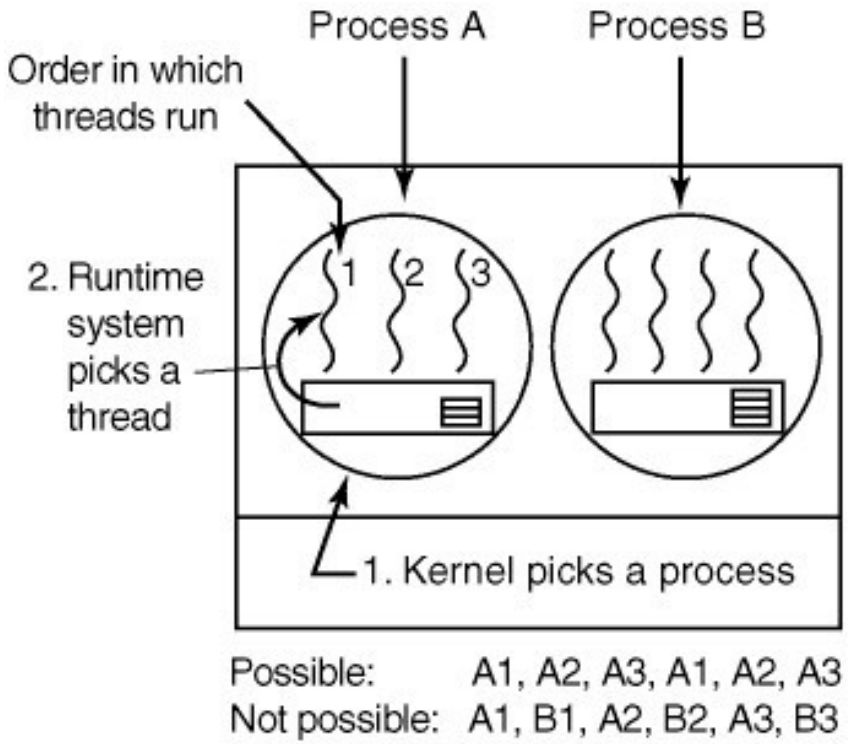


Figure 2-27. A scheduling algorithm with four priority classes.

Thread Scheduling (1)



(a)

Figure 2-28. (a) Possible scheduling of user-level threads with a 50-msec process quantum and threads that run 5 msec per CPU burst.

Thread Scheduling (2)

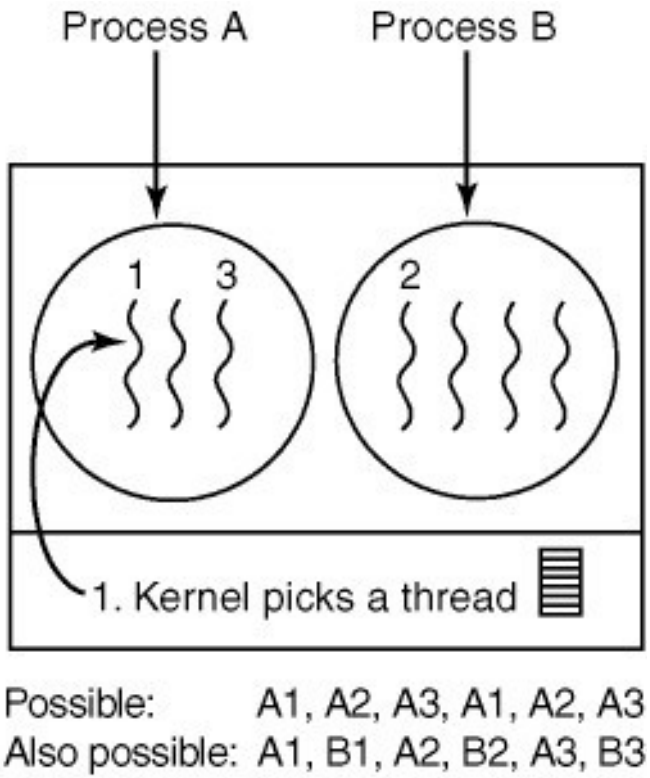


Figure 2-28. (b) Possible scheduling of kernel-level threads with the same characteristics as (a).