PROCESSES & THREADS

Charles Abzug, Ph.D.

Department of Computer Science
James Madison University
Harrisonburg, VA 22807

Voice Phone: ***540-568-8746*;** Cell Phone: ***443-956-9424***
E-mail: ***abzugcx@JMU.edu*** OR ***CharlesAbzug@ACM.org***
Home Page: ***https://users.cs.jmu.edu/abzugcx/public/index.htm***

Tanenbaum, Andrew S. (2008). *Modern Operating Systems. Third Edition.* Upper Saddle River, NJ: Prentice-Hall. ISBN: 0-13-031358-0.

CHAPTER 2: Processes and Threads

Sobell, Mark G. (2005). *A Practical Guide to Linux Commands, Editors, and Shell Programming.* Upper Saddle River, NJ: Prentice-Hall Professional Technical Reference. ISBN: 0-13-147823-0 (alk. paper).

CHAPTER 3:

CPU-Scheduling Algorithms (for Processes or Threads)

1. First-Come-First-Served
2. Shortest Job First
3. Shortest Remaining Time Next
4. Round-Robin Scheduling
5. Priority Scheduling
6. Priority Scheduling with Multiple Queues
7. Shortest Process Next
8. "Guaranteed" Scheduling
9. Lottery Scheduling
10. Fair-Share Scheduling

Part I: PROCESSES

Multiprogramming

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-1.
© 2008 Pearson Education

Initiators of Process Creation

1. **System Initialization (Boot-Up)**
2. **Explicit User-Initiation (*e.g.,* issuance of a CLI command)**
3. **Existing Process via issuance of a Process-Creation System Call**
4. **Long-Term Scheduler, in accommodating a submitted batch job**

Initiators of Process Termination

1. Completion of assigned task: normal termination (voluntary)
2. Exit upon encountering a specifically chosen error (voluntary)
3. Occurrence of a fatal error (involuntary)
4. Explicitly ordered by another (usually ancestral) process (involuntary)

Kinds of Processes

1. Foreground
2. Background
3. Daemon

Simple Model of Process States and Process Transitions

Causal events for transition:

NOTE: Fourth state (not shown): Terminated

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-2.
© 2008 Pearson Education

Individual Processes, and the Underlying OS for Interrupt-Handling & Scheduling

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-3.
© 2008 Pearson Education

Contents of a Process Descriptor

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-4.
© 2008 Pearson Education

Sequence of Activities Following the Occurrence of an Interrupt

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-5.
© 2008 Pearson Education

% CPU Utilization Depends on the Degree of Multiprogramming

$$\boxed{U_{\mathrm{CPU}} = 1 - p^{n}}$$

Degree of Multiprogramming ⟶

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-6.
© 2008 Pearson Education

Part II: THREADS

Word Processor with Multiple Threads

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-7.
© 2008 Pearson Education

1. **Waits for and then handles input from keyboard.**
2. **Reformats the document in background.**
3. **Autosaves the document at the specified time interval, copying out the current content onto the disk (foreground).**
4. **Sends the document to the print SPOOLer.**

Web Server with Multiple Threads

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-8.
© 2008 Pearson Education

Web Server: Dispatcher Thread

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-9a.
© 2008 Pearson Education

Web Server: Worker Thread

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-9b.
© 2008 Pearson Education

Alternative Approaches to the Design of a Web Server

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-10.
© 2008 Pearson Education

Single-Threaded and Multi-Threaded Processes

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-11.
© 2008 Pearson Education

Items Shared by All the Threads of a Single Process, and Items Unique to Each Thread

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-12.
© 2008 Pearson Education

Separate Stack for Each Thread

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-13.
© 2008 Pearson Education

Function Calls for POSIX Threads (Pthreads)

Tanenbaum (2008). *Modern Operating Systems, 3rd Edition.*
Figure 2-14.
© 2008 Pearson Education

Example of a Program that Uses *Pthreads*

Tanenbaum (2008). *Modern Operating Systems, 3rd Edition.*
Figure 2-15.
© 2008 Pearson Education

User-Level Threads *vs.* Kernel-Level Threads

Tanenbaum (2008). *Modern*
Operating Systems. 3rd Edition.
Figure 2-16.
© 2008 Pearson Education

Multiplexing of Several User-Level Threads Onto a Single Kernel-Level Thread

Tanenbaum (2008). *Modern*
Operating Systems. 3rd Edition.
Figure 2-17.
© 2008 Pearson Education

Creation of a New Thread in Response to the Arrival of a Message

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-18.
© 2008 Pearson Education

Inter-Thread Conflicts in the Use of a Global Variable

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-19.
© 2008 Pearson Education

Private Global Variables

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-20.
© 2008 Pearson Education

Race Conditions

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-21.
© 2008 Pearson Education

Mutual Exclusion from a Critical Region

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-22.
© 2008 Pearson Education

Use of a Spin Lock to Achieve Mutual Exclusion

Process 0:

Process 1:

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-23.
© 2008 Pearson Education

Peterson's Algorithm for Achieving Mutual Exclusion

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-24.
© 2008 Pearson Education

Use of an Atomic "Test and Set Lock" (TSL) Machine Instruction to Achieve Mutual Exclusion

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-25.
© 2008 Pearson Education

Entering and Leaving the Critical Region Using "XCHG"

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-26.
© 2008 Pearson Education

Race Condition in the "Producer-Consumer Problem"

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-27.
© 2008 Pearson Education

Use of Semaphores in the "Producer-Consumer Problem"

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-28.
© 2008 Pearson Education

Use of *mutex_lock* and *mutex_unlock* to Achieve Mutual Exclusion

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-29.
© 2008 Pearson Education

POSIX Threads: Procedure Calls for Mutual Exclusion

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-30.
© 2008 Pearson Education

POSIX Threads: Procedure Calls Pertaining to Condition Variables

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-31.
© 2008 Pearson Education

POSIX Threads: Use of Mutexes and Condition Variables to Solve the Producer-Consumer Problem

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-32a.
© 2008 Pearson Education

POSIX Threads: Use of Mutexes and Condition Variables to Solve the Producer-Consumer Problem

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-32b.
© 2008 Pearson Education

Use of the Monitor, a Programming-Language Construct, to Achieve Mutual Exclusion

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-33.
© 2008 Pearson Education

Use of a Monitor to Effect Mutual Exclusion in the "Producer-Consumer Problem"

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-34.
© 2008 Pearson Education

The "Producer-Consumer Problem: Solution in Java (1)

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-35"a".
© 2008 Pearson Education

The "Producer-Consumer Problem: Solution in Java (2)

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-35"b".
© 2008 Pearson Education

Message-Passing Used to Solve the "Producer-Consumer Problem:

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-36.
© 2008 Pearson Education

Use of a Barrier to Enforce the Synchronization of Multiple Processes

All processes are approaching the barrier. None has yet reached it.

Several processes have reached the barrier and are waiting until all are present.

Last process has reached the barrier. All processes may now pass through.

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-37.
© 2008 Pearson Education

CPU-Bound and I/O-Bound Processes

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-38.
© 2008 Pearson Education

Goals of the Scheduling Algorithm: Dependent upon the Computing Environment

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-39.
© 2008 Pearson Education

Long-Term (Admission) Scheduler, Medium-Term (Memory) Scheduler, and Short-Term (CPU) Scheduler

NOTE: This figure did not come from our course text.

CPU-Scheduling Algorithms

1. First-Come-First-Served (FCFS): Batch Environment, *NON*-Preemptive

CPU-Scheduling Algorithms

1. First-Come-First-Served (FCFS): Batch Environment, *NON*-Preemptive

2. Shortest Job First (SJF): Batch Environment, *NON*-Preemptive

Effectiveness of "Shortest-Job First" Scheduling

Jobs run in order of arrival:
First-Come-First-Served (FCFS)

Shortest Jobs run First (SJF)

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-40.
© 2008 Pearson Education

CPU-Scheduling Algorithms

1. First-Come-First-Served (FCFS): Batch Environment, *NON*-Preemptive

2. Shortest Job First (SJF): Batch Environment, *NON*-Preemptive

3. Shortest Remaining Time Next (SRT): Batch Environment, ***PREEMPTIVE***

CPU-Scheduling Algorithms

1. First-Come-First-Served (FCFS): Batch Environment, *NON*-Preemptive

2. Shortest Job First (SJF): Batch Environment, *NON*-Preemptive

3. Shortest Remaining Time Next (SRT): Batch Environment, ***PREEMPTIVE***

4. Round-Robin Scheduling (with Time Quantum): Interactive, ***PREEMPTIVE***

Round-Robin Scheduling

Job 'B' gets the processor:

Job 'B' uses up its quantum:

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-41.
© 2008 Pearson Education

CPU-Scheduling Algorithms

1. **First-Come-First-Served (FCFS): Batch Environment, *NON*-Preemptive**

2. **Shortest Job First (SJF): Batch Environment, *NON*-Preemptive**

3. **Shortest Remaining Time Next (SRT): Batch Environment, *PREEMPTIVE***

4. **Round-Robin Scheduling (with Time Quantum): Interactive, *PREEMPTIVE***

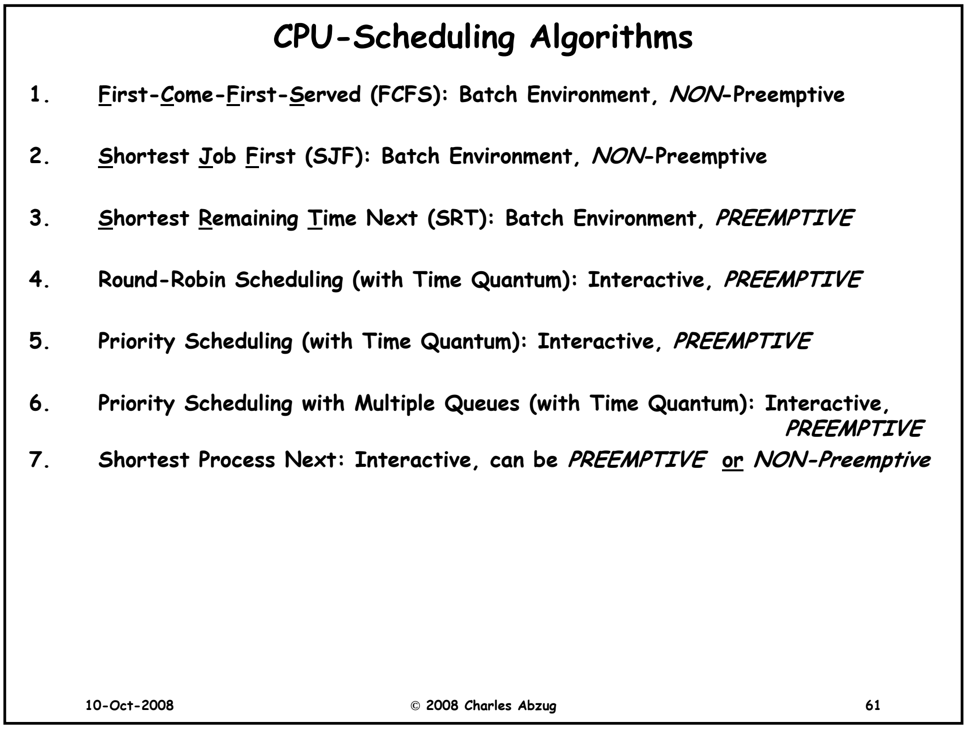
5. **Priority Scheduling (with Time Quantum): Interactive, *PREEMPTIVE***

CPU-Scheduling Algorithms

1. **First-Come-First-Served (FCFS): Batch Environment, *NON*-Preemptive**
2. **Shortest Job First (SJF): Batch Environment, *NON*-Preemptive**
3. **Shortest Remaining Time Next (SRT): Batch Environment, *PREEMPTIVE***
4. **Round-Robin Scheduling (with Time Quantum): Interactive, *PREEMPTIVE***
5. **Priority Scheduling (with Time Quantum): Interactive, *PREEMPTIVE***
6. **Priority Scheduling with Multiple Queues (with Time Quantum): Interactive,**

Priority-Based Scheduling

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-42.
© 2008 Pearson Education



CPU-Scheduling Algorithms

1. First-Come-First-Served (FCFS): Batch Environment, *NON*-Preemptive

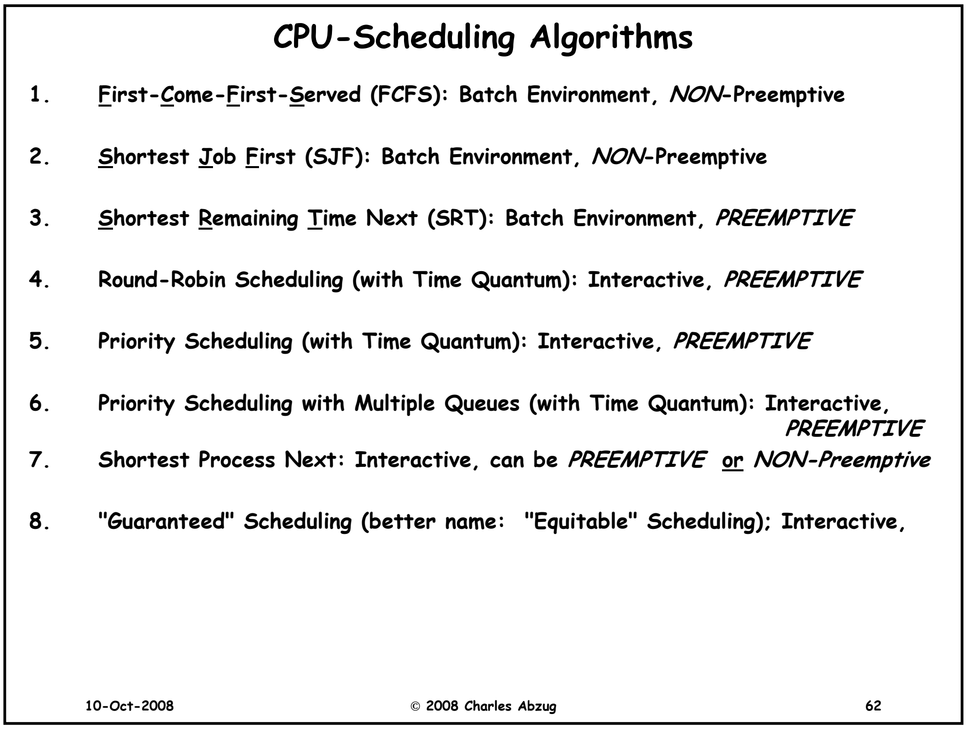
2. Shortest Job First (SJF): Batch Environment, *NON*-Preemptive

3. Shortest Remaining Time Next (SRT): Batch Environment, ***PREEMPTIVE***

4. Round-Robin Scheduling (with Time Quantum): Interactive, ***PREEMPTIVE***

5. Priority Scheduling (with Time Quantum): Interactive, ***PREEMPTIVE***

6. Priority Scheduling with Multiple Queues (with Time Quantum): Interactive, ***PREEMPTIVE***
7. Shortest Process Next: Interactive, can be ***PREEMPTIVE*** or ***NON-Preemptive***



CPU-Scheduling Algorithms

1. First-Come-First-Served (FCFS): Batch Environment, *NON*-Preemptive

2. Shortest Job First (SJF): Batch Environment, *NON*-Preemptive

3. Shortest Remaining Time Next (SRT): Batch Environment, ***PREEMPTIVE***

4. Round-Robin Scheduling (with Time Quantum): Interactive, ***PREEMPTIVE***

5. Priority Scheduling (with Time Quantum): Interactive, ***PREEMPTIVE***

6. Priority Scheduling with Multiple Queues (with Time Quantum): Interactive, ***PREEMPTIVE***
7. Shortest Process Next: Interactive, can be ***PREEMPTIVE*** or ***NON-Preemptive***

8. "Guaranteed" Scheduling (better name: "Equitable" Scheduling); Interactive,

CPU-Scheduling Algorithms

1. First-Come-First-Served (FCFS): Batch Environment, *NON*-Preemptive
2. Shortest Job First (SJF): Batch Environment, *NON*-Preemptive
3. Shortest Remaining Time Next (SRT): Batch Environment, *PREEMPTIVE*
4. Round-Robin Scheduling (with Time Quantum): Interactive, *PREEMPTIVE*
5. Priority Scheduling (with Time Quantum): Interactive, *PREEMPTIVE*
6. Priority Scheduling with Multiple Queues (with Time Quantum): Interactive, *PREEMPTIVE*
7. Shortest Process Next: Interactive, can be *PREEMPTIVE* or *NON-Preemptive*
8. "Guaranteed" Scheduling (better name: "Equitable" Scheduling); Interactive, *PREEMPTIVE*
9. Lottery Scheduling (with Time Quantum): Interactive , *PREEMPTIVE*

CPU-Scheduling Algorithms

1. First-Come-First-Served (FCFS): Batch Environment, *NON*-Preemptive
2. Shortest Job First (SJF): Batch Environment, *NON*-Preemptive
3. Shortest Remaining Time Next (SRT): Batch Environment, *PREEMPTIVE*
4. Round-Robin Scheduling (with Time Quantum): Interactive, *PREEMPTIVE*
5. Priority Scheduling (with Time Quantum): Interactive, *PREEMPTIVE*
6. Priority Scheduling with Multiple Queues (with Time Quantum): Interactive, *PREEMPTIVE*
7. Shortest Process Next: Interactive, can be *PREEMPTIVE* or *NON-Preemptive*
8. "Guaranteed" Scheduling (better name: "Equitable" Scheduling); Interactive, *PREEMPTIVE*
9. Lottery Scheduling (with Time Quantum): Interactive , *PREEMPTIVE*
10. Fair-Share Scheduling (with Time Quantum): Interactive, *PREEMPTIVE*

Scheduling of User-Level Threads

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-43a.
© 2008 Pearson Education

Scheduling of Kernel-Level Threads

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-43b.
© 2008 Pearson Education

The "Dining Philosophers Problem"

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-44.
© 2008 Pearson Education

Deadlock or Starvation in the "Dining Philosophers Problem"

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-45.
© 2008 Pearson Education

Use of Semaphores to Solve the "Dining Philosophers Problem" (1)

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-46a.
© 2008 Pearson Education

Use of Semaphores to Solve the "Dining Philosophers Problem" (2)

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-46b.
© 2008 Pearson Education

Use of Semaphores to Solve the "Readers & Writers Problem"

Tanenbaum (2008). *Modern Operating Systems. 3rd Edition.*
Figure 2-47.
© 2008 Pearson Education

END