1. Concurrency

- A processor may run two or more programs at the same time (multitasking)
- To do so, it saves its current state in one program’s fetch/execute cycle, and loads the state of another program’s cycle, to run a time slice of that program, numerous times per second
- A process is a programming abstraction that simulates ownership of all computer resources
8. Multithreading

A thread simulates ownership of a CPU

- One program may run multiple threads, e.g., for disk access and user I/O
- The processor executes multiple threads concurrently
- Multiple processors may share execution of threads

A pipe simulates interprocess communication

- A process that produces data may make results available before it completes
- A second process works immediately on the first data that enters the pipeline
- Familiar example: the conveyer on an assembly line
Examples of multithreading

- *Downloading* video while starting to play it
- *Printing* concurrently with document editing
- *Garbage collection*: Java garbage collector runs as a low-priority thread to free up memory previously allocated to objects but no longer used

### 2. Java threads

Methods:

- `start`
- `run`
- `sleep`
- `interrupt`
- `suspend`
- `stop`
- `isAlive`
- `currentThread`
Life cycle of a thread

[pic, Deitel and Deitel, 1997, p. 674]

Memory allocation and garbage collection

• Java and C++ new operator (function malloc in C) causes dynamic allocation of memory for objects
• References and pointers provide access to object [pic]
• When reference variable goes out of scope (e.g., enclosing method terminates), memory space is still allocated but object is inaccessible
• Memory must be deallocated
  – automatically by Java garbage collector or
  – using delete operator in C++, free in C
8. Multithreading

Time slicing

- Threads are each allocated a time quantum (slice) to execute
- After one thread’s time slice, the thread waits for other processes to take their turns
- All threads of equal priority take turns, round-robin style
- Threads are pre-empted by higher-priority threads or by threads whose turn has come

Java threads

- A Thread object is constructed within a running method and is launched by a call to the start method
- Upon launch, the thread executes concurrently with the calling thread
- sleep(n) method causes thread to sleep n milliseconds
- suspend and resume methods cause thread to pause and restart
### Thread states

- born $\rightarrow$ ready $\leftrightarrow$ running
- *ready* means waiting for processor resource
- running $\rightarrow$
  - waiting
  - sleeping
  - suspended
  - blocked (waiting for I/O)
  - dead

### 3. Priorities and scheduling

- Each thread has priority in 1 .. 10; 5 by default (*Thread.MIN_PRIORITY* = 1)
- Threads inherit priority of their creator threads
- Java scheduler ensures that highest priority thread executes at all times
- If OS platform supports time slicing and multiple highest-priority threads are running, then they execute *round robin*
8. Multithreading

Synchronization

- Objects with methods declared as `synchronized` may have these methods execute only under one thread at a time
- This is enforced by a thread obtaining a `lock` on the object
- Monitor objects enforce locking
- Example: a bank-account transaction requires locking the account object

4. Producers and consumers

- Example: Producer thread places data in a buffer, consumer thread prints it
- Producer and consumer may need to be synchronized to avoid excess data or loss of data to consumer
- Hence these methods are declared with the `synchronized` keyword
8. Multithreading

# Daemon threads

- **Definition:** threads that run in background to serve other threads
- **Example:** Java garbage collector
- `setDaemon(true)` causes a thread to be a daemon
- Program terminates when only daemons are running

## Java interface Runnable

Multithreading is enabled only in classes that
- extend `Thread`, or
- implement the Java interface `Runnable`
Multithreading vs. parallel processing

- Threading is a *programming abstraction*
- Parallelism refers to multi-processor *hardware* environment
- Multithreading can occur on single processor
- Multicore and multi-processor systems can speed up performance of multithreaded software

**Terms**

- concurrency
- consumer thread
- daemon thread
- dynamic allocation
- garbage collection
- interrupt
- lock
- monitor
- multithreading
- parallelism
- pre-emption
- priority
- producer thread
- quantum
- ready
- resume
- round robin
- Runnable
- scheduler
- sleep
- start
- starvation
- step
- suspend
- synchronization
- thread
- time slicing
- wait
References