Chapter 10

Multithreading, Networks, and Client/Server Programming

At a Glance

Instructor's Manual Table of Contents

Lecture Notes

Overview

Chapter 10 provides an overview of multithreading, networks, and client/server programming with Python. Students learn the basic principles behind multithreading, and how threads are manipulated in Python. The problem of synchronization is introduced, and conditions are used to solve a simple synchronization problem. Students also learn to use IP addresses, ports, and sockets to create simple client/server applications. Finally, they learn how to restructure existing applications for deployment as client/server applications on a network.

Objectives

After completing this chapter, students will be able to:

- Describe what threads do and how they are manipulated in an application
- Code an algorithm to run as a thread
- Use conditions to solve a simple synchronization problem with threads
- Use IP addresses, ports, and sockets to create a simple client/server application on a network
- Decompose a server application with threads to handle client requests efficiently
- Restructure existing applications for deployment as client/server applications on a network

Teaching Tips

Threads and Processes

1. Provide a brief explanation of the evolution of threads and processes. Discuss *time-sharing systems*, *multiprocessing systems*, *networked and distributed systems*, and *parallel systems*.

2. Explain the term *parallel computing* as it applies to hardware architectures, operating systems, and algorithms.

Threads

1. Explain how threads are used in modern computers to represent processes and that different processes may be run in separate threads.

2. Explain that, in Python, a thread is an object like any other in that it can hold data, be run with methods, be stored in data structures, and be passed as parameters to methods. However, threads can also be executed as a process if it implements the `run` method.

Teaching Tip	"Python piggybacks on top of the OS' underlying threads system. A Python thread is a real OS thread. If a Python program has three threads, for instance, there will be three entries in the `ps` output. However, Python imposes further structure on top of the OS threads. Most importantly, there is a global interpreter lock, the famous (or infamous) GIL. It is set up to ensure that (a) only one thread runs at a time, and (b) that the ending of a thread's turn is controlled by the Python interpreter rather than the external event of the hardware timer interrupt." *Reference:* http://heather.cs.ucdavis.edu/~matloff/Python/PyThreads.pdf.

3. Use Figure 10.1 to describe the states in the life of a thread. Point out how, when, and why a thread changes from one state to another one. Introduce the following terms: *ready queue*, *time slicing*, and *context switch*.

4. Use an example to show that the most common way to create a thread is to define a class that extends the class `threading.Thread`.

5. Use Table 10.1 to describe some of the most useful `Thread` methods. Note that a thread's `run` method is invoked automatically by `start`.

Teaching Tip	The method `threading.Thread.join()` avoids the need for wasteful looping in `main()`, while the latter is waiting for the other threads to finish. *Reference:* http://heather.cs.ucdavis.edu/~matloff/Python/PyThreads.pdf.

Sleeping Threads

1. Note that the function `time.sleep` puts a thread to sleep for the specified number of seconds.

2. Use Figure 10.2 and the sample program provided in the book to show how to use `time.sleep` to put a thread to sleep.

Producer, Consumer, and Synchronization

1. Explain that threads that interact by sharing data are said to have a *producer/consumer relationship*. Use the assembly line in a factory example (or another example) to help illustrate this concept.

2. Use Figure 10.3, Table 10.2, and the code provided in the book to show how the producer/consumer relationship can be simulated in a program. Explain how *synchronization problems* may arise in this program.

3. Use the pseudocode provided in the book to explain how to solve the synchronization problems by using a shared cell with a Boolean flag and an instance of `threading.Condition`. Introduce the concept of locking a resource.

Teaching Tip	"Python offers a higher-level class, `threading.Event`, which is just a wrapper for `threading.Condition`, but which does all the lock operations behind the scenes, alleviating the programmer from having to do this work." *Reference:* http://heather.cs.ucdavis.edu/~matloff/Python/PyThreads.pdf.

4. Use Table 10.3 to describe the role of the methods of the `Condition` class.

Teaching Tip	For more information on the `Condition` class, visit: http://docs.python.org/lib/condition-objects.html.

Quick Quiz 1

1. What is parallel computing?
   Answer: Parallel computing is the discipline of building the hardware architectures, operating systems, and specialized algorithms for running a program on a cluster of processors.

2. True or False: Before it can execute, a thread's class must implement a `run` method.
   Answer: True

3. Threads that interact by sharing data are said to have a(n) ____________________ relationship.
   Answer: producer/consumer

4. A `Condition` object is used to maintain a(n) ____________________ on a resource.
   Answer: lock

Networks, Clients, and Servers

1. Introduce the following concepts: *client*, *server*, *IP address*, and *socket*.

Teaching Tip	For more information about socket programming in Python, visit: http://docs.python.org/howto/sockets.

IP Addresses

1. Describe the role of *IP addresses*, *IP numbers*, and *IP names* in identifying computers in a network.

2. Explain that Python's `socket` module includes two functions that can look up the IP name and IP number of a computer. Use Table 10.4 to describe the `socket` functions for IP addresses, and use a few examples to show how to implement these functions.

3. Introduce the terms *local host* and *Internet host*, and explain how these types of computers can be used to test a network application.

Ports, Servers, and Clients

1. Explain that clients connect to servers via *ports*, and provide an overview of the structure and functionality of different ports.

Sockets and a Day/Time Client Script

1. Introduce the term *socket*, and explain how it functions in a network.

2. Use Figures 10.4, 10.5, and the code provided in the book to show how to use sockets to implement a day/time client script.

Teaching Tip	Python supports non-blocking sockets; i.e. sockets for which `send`, `recv`, `connect` and `accept` can return without having done anything. *Reference:* http://docs.python.org/dev/howto/sockets.html.

A Day/Time Server Script

1. Use Figure 10.6 and the code provided in the book to show how to use sockets to implement a day/time server script. Be sure to guide students through the pseudocode provided on Page 415 of the book to clarify what the code is trying to achieve.

A Two-Way Chat Script

1. Explain the basic steps needed to implement a two-way chat script. Use the code provided in the book to show how to implement these steps.

Handling Multiple Clients Concurrently

1. Explain that to solve the problem of giving many clients timely access to the server, we can assign the task of handling the client's request to a client-handler thread (see Figure 10.7).

2. Use the code provided in the book to show how to implement this client-handler thread.

Setting Up Conversations for Others

1. Use Figure 10.8 and the code provided in the book to explain how to support multiple two-way chats.

Quick Quiz 2

1. What is an IP address?
   Answer: Every computer on a network has a unique identifier called an IP address (IP stands for Internet Protocol). This address can be specified either as an IP number or as an IP name.

2. True or False: When developing a network application, the programmer can first try it out on a virtual host—that is, on a standalone computer that may or may not be connected to the Internet.
   Answer: False

3. What is a port?
   Answer: Clients connect to servers via objects known as ports. A port serves as a channel through which several clients can exchange data with the same server or with different servers. Ports are usually specified by numbers.

4. The ____________________ `socket` function returns the IP name of the host computer running the Python interpreter.
   Answer: `gethostname()`

Case Study: A Multi-Client Chat Room

1. Guide students as they step through this section.

Request

1. Ask students to write a program that supports an online chat room.

Analysis

1. Use Figure 10.9 to describe the user interface of the program.

Design

1. Review the structure that the chat room program would need to have in order to function according to the specified requirements.

2. Note the similarities and differences between the chat room’s program structure and the online therapy server described earlier.

Implementation (Coding)

1. Guide students through the script, explaining lines of code that are unclear and making sure that students understand the purpose of each section of code.

2. Note that the script differs a bit from the earlier examples because it must prompt the human user for a user name and send it to the server before entering its conversation loop.

Quick Quiz 3

1. What synchronization problem may arise from the code of the multi-way chat room?
   Answer: You might have noticed that the chat record is actually shared among several client-handler threads. This presents a potential synchronization problem of the type discussed earlier in this chapter. If one handler is timed out in the middle of a mutation to the record, some data might be lost or corrupted for this or other clients. The solution of this problem is left as an exercise.

Class Discussion Topics

1. Ask your students to talk about any previous programming experience with threads and/or sockets. How does these features in another language compare with the way Python handles them?

2. Have your students taken an Operating Systems course? If so, ask them to describe other synchronization mechanisms that they may be aware of.

Additional Projects

1. Ask students to do some research on how the `threading.Thread.join()` method can be used to avoid wasteful looping in `main()`.

2. Ask your students to do some research on other synchronization mechanisms provided by Python (e.g., `threading.Event`, `threading.Semaphore`, and `threading.RLock`).

3. Ask students to do some research on non-blocking sockets in Python. How can they be created/used?

Additional Resources

1. Tutorial on Threads Programming with Python:
   http://heather.cs.ucdavis.edu/~matloff/Python/PyThreads.pdf

2. Socket Programming HOWTO:
   http://docs.python.org/dev/howto/sockets.html

3. `socket` -- Low-level networking interface:
   www.python.org/doc/lib/module-socket.html

4. Thread Synchronization Mechanisms in Python:
   http://effbot.org/zone/thread-synchronization.htm

Key Terms

- **client/server relationship:** A means of describing the organization of computing resources in which one resource provides a service to another resource.
- **context switch:** The process of saving or restoring a thread's state.
- **distributed system:** See networked system.
- **front:** The end of a queue from which elements are removed.
- **IP address:** The unique location of an individual computer on the Internet.
- **IP name:** A representation of an IP address that uses letters and periods.
- **IP number:** A representation of an IP address that uses digits and periods.
- **multiprocessing system:** a system that allows a single user to run multiple programs at once.
- **network:** A collection of resources that are linked together for communication.
- **networked system:** a system that allows processes associated with a program to be distributed across several CPUs linked by high-speed communication lines.
- **parallel computing:** building hardware architectures, operating systems, and specialized algorithms for running a program on a cluster of processors**.**
- **parallel system:** A system that runs a single program on several CPUs at once.
- **port:** A channel through which several clients can exchange data with the same server or with different servers.
- **producer/consumer relationship:** A means of describing the organization of computing resources in which one resource produces a data item and provides it to the consumer for consumption.
- **ready queue:** A data structure used to schedule processes or threads for CPU access.
- **rear:** The end of a queue to which elements are added.
- **server:** A computational object that provides a service to another computational object.
- **socket:** An object that serves as a communication link between a single server process and a single client process.
- **synchronization problem:** A type of problem arising from the execution of threads or processes that share memory.
- **time sharing:** The scheduling of multiple programs so that they run concurrently on the same computer.
- **time slicing:** A means of scheduling threads or processes wherein each process receives a definite amount of CPU time before returning to the ready queue.
- **time-sharing operating system:** A computer system that can run multiple programs in such a manner that its users have the illusion that they are running simultaneously.