

## Motivation, Terminology, Layered systems (and other random stuff)

---

---

---

---

---

---

---

---

## History and Motivation

- Early computers were highly centralized.
  - Single point of failure
  - User has to “go to” the computer.
- Proliferation of low cost computers made it possible to get past these 2 primary disadvantages (with a network).

---

---

---

---

---

---

---

---

## Motivation

- Sharing of resources is more efficient
- Price/Performance
- Use each piece of equipment for what it is best at
- Centralize administration
- Computers as communication tools

---

---

---

---

---

---

---

---

## Computer Networks are now everywhere

- PCs <-> Mainframes
- Automated Tellers
- Embedded Systems
- Communications Systems
- The Internet

---

---

---

---

---

---

---

---

## Networked Computers - Traditional Uses

- Communication (email)
- File exchange, disk sharing
- Sharing peripherals (printers, tape drives)
- Remote execution

---

---

---

---

---

---

---

---

## New(er) Uses for Networked Computers

- Information Sharing
- Entertainment, distributed games
- MP3s!
- Commerce
- Automation of business processes
- Collaborative computing
- Homework Submission

---

---

---

---

---

---

---

---

## Wide variety of types of networks

- circuit switched
  - telephone system
- packet switched:
  - The Internet (TCP/IP)

---

---

---

---

---

---

---

---

## Network Models

- Using a formal model allows us to deal with various aspects of Networks abstractly.
- We will look at a popular model (OSI reference model).
- The OSI reference model is a *layered* model.

---

---

---

---

---

---

---

---

## Layering

- Divide a task into pieces and then solve each piece independently (or nearly so).
- Establishing a well defined interface between layers makes porting easier.
- Major Advantages:
  - ◆ Code Reuse
  - ◆ Extensibility

---

---

---

---

---

---

---

---

## Layering Example: Federal Express

- Letter in envelope, address on outside
- FedX guy adds addressing information, barcode.
- Local office drives to airport and delivers to hub.
- Sent via airplane to nearest city.
- Delivered to right office
- Delivered to right person

Netprog 2001 Network Terminology

10

---

---

---

---

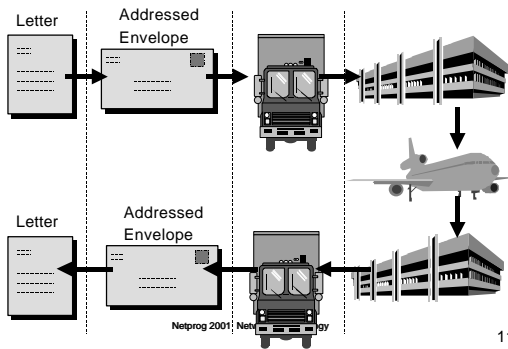
---

---

---

---

## FedX Layers



Netprog 2001 Network Terminology

11

---

---

---

---

---

---

---

---

## Layered Software Systems

- Network software
- Operating systems
- Windowing systems

Netprog 2001 Network Terminology

12

---

---

---

---

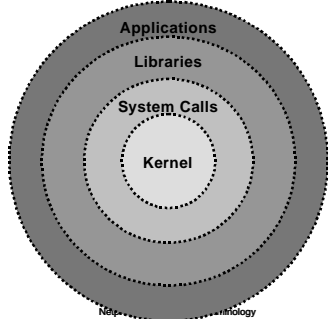
---

---

---

---

## Unix is a Layered System



13

---

---

---

---

---

---

---

---

## OSI Reference Model

The International Standards Organization (ISO) proposal for the standardization of the various protocols used in computer networks (specifically those networks used to connect open systems) is called the Open Systems Interconnection Reference Model (1984), or simply the OSI model.

Netprog 2001 Network Terminology

14

---

---

---

---

---

---

---

---

## OSI Model

Although the OSI model is a just a model (not a specification), it is generally regarded as the most complete model (as well it should be - nearly all of the popular network protocol suites in use today were developed before the OSI model was defined).

Netprog 2001 Network Terminology

15

---

---

---

---

---

---

---

---



## What's a Protocol?

- An agreed upon convention for communication.
  - both endpoints need to *understand* the protocol.
- Protocols must be formally defined and unambiguous!
- We will study lots of existing protocols and perhaps develop a few of our own.

---

---

---

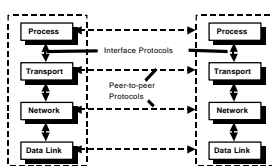
---

---

---

---

---



## Interface and Peer-to-peer Protocols

- Interface protocols describe the communication between layers on the same endpoint.
- Peer-to-peer protocols describe communication between peers at the same layer.

---

---

---

---

---

---

---

---

## Thought Exercise

- Come up with an example of a layered system.
- Describe the interface and peer-to-peer protocols for your example.



---

---

---

---

---

---

---

---

## Programs & Processes

- A *program* is an executable file.
- A *process* or *task* is an instance of a program that is being executed.
- A single program can generate multiple processes.

---

---

---

---

---

---

---

---

## Client - Server

- A *server* is a process - not a machine !
- A server waits for a request from a client.
- A client is a process that sends a request to an existing server and (usually) waits for a reply.

---

---

---

---

---

---

---

---

## Client - Server Examples

- Server returns the time-of-day.
- Server returns a document.
- Server prints a file for client.
- Server does a disk read or write.
- Server records a transaction.

---

---

---

---

---

---

---

---

## Servers

- Servers are generally more complex (more interesting).
- 2 Basic types of servers:
  - ◆ *Iterative* - server handles one client at a time.
  - ◆ *Concurrent* - server handles many clients at a time.
- We will study the differences later.

---

---

---

---

---

---

---

---

## Unix and Processes

- In Unix the way to create a new process is the **fork**( ) system call.
- In Unix the way to run a new program is with the family of **exec**( ) system calls.
- You should already know this!

---

---

---

---

---

---

---

---