Operating Systems and UNIX
Operating Systems

Batch versus interactive processing

1. OK, so we have this computer now...
   (a) Imagine you have a program to run on given data set...
   (b) How do you get computer to execute?
   (c) We need system for management of jobs, programs, etc

2. Batch:
   (a) No user interaction, computer waits for prepared “jobs”
   (b) Jobs are processed in one sequence: job queue (FIFO)

(c) Programs accompanied by job control instructions and data

3. Interactive:
   (a) Computer continuously responds to user requests
   (b) Job control is partially and interactively in hands of user
   (c) Requires system to implement user interaction
   (d) “Real Time Processing”: coordination between machine and machine environment
Batch Processing
Interactive Processing
Time-Sharing and Multi-Tasking

Remember: A program is an ordered set of instructions

Problem:

1. Only one program runs at one specific time
2. What happens when user wants (a) to run several programs at same time? (b) More than one users use system?

Solutions:

1. Time-Sharing: divide computer’s time
2. Multi-Tasking: illusion of parallel execution
Time-Sharing and Multi-Tasking

1. Time-Sharing:
   (a) Computer’s time is divided into slices
   (b) Slices are allocated for different processes or users

2. Multi-Tasking
   (a) Single User systems
   (b) Illusion of running more than one program at one time
   (c) Essentially Time-Sharing
Time-Sharing
Setting up User Accounts on lab.cs.odu.edu

1. Anyone enrolled in CS course is entitled to account
   (a) On-Line registration at: www.cs.odu.edu/online
   (b) Requires Student ID number

2. Supervized enrollment only for this class
   (a) Registration in my office
   (b) During office hours tomorrow (March 26th), 2-5PM
   (c) All other: on appointment basis
Operating System

1. Just another program
   (a) Manages system and user experience
   (b) Implements “shell” or GUI user interface
   (c) Regulates start and termination of programs
   (d) Takes care of communication to external devices (HD, etc)

2. Functions as a buffer between user and core of system
Operating System - Components: Shell and Kernel

1. Shell: interfaces between user and OS
   (a) Command-line
   (b) Graphical User Interface (GUI)

2. Kernel
   (a) File and Memory Management
   (b) Interfaces with external devices
   (c) Activation/Termination of programs
Operating Systems

1. Windows
   (a) Microsoft product
   (b) Version 3.0: 1990: shell over DOS
   (c) DOS: Command-line shell dating back to 1981 (IBM PC)
   (d) Most recent version is XP

2. MacOS:
   (a) LISA computer: 1983
   (b) Largely based on ideas developed at Xerox PARC
   (c) Now based on Free BSD UNIX OS

3. UNIX/Linux
   (a) Created at Bell Labs as a result of MULTICS effort in late 60s
   (b) Purpose: multi-user system, reliable time-sharing
   (c) Philosophy: small applications combined produce big results
Bootstrapping - How does the OS start?

1. Bootstrapping:
   (a) RAM memory is volatile: erases when power is turned off
   (b) ROM: retains information when computer is turned off
   (c) ROM contains small program at address where CPU first looks for instruction to execute
   (d) Program directs CPU to transfer material from external storage to RAM memory: Operating System can be started

2. OS resides in:
   (a) Floppy
   (b) HD
   (c) CDROM
Bootstrapping the OS
External storage: File Management

Files and Directories:

1. File: stored data object, labeled by alphanumeric string
   (a) [8].[3] standard
   (b) Name + suffix (indication of type)
   (c) Long file names
   (d) Some non-alphanums are allowed: -, _, space, etc.

2. Directory: bundle of files and directory, labeled by alphanumeric string

Directory Tree:

1. Defines a tree structure
2. Tree has a root, in UNIX denoted by ’/’, Windows: drive letter, e.g A:
3. Each file location denoted by sequence of directory labels, separated by ’/’ (UNIX)

Examples:

Filename  Letter2Mommy.txt, letter_To_mommy-for_mday.txt
Directory  /home/jdoe/files/letters
Filename in Directory  /home/jdoe/letter.txt, /home/jdoe/data.txt
Some Conventions

1. UNIX: root has several standard subdirectories:
   (a) /bin: system commands
   (b) /usr/bin: user software
   (c) /home: user accounts, e.g. /home/jbollen
   (d) /mnt: location of external drives, e.g. /mnt/floppy

2. Your account:
   (a) login = jbollen
   (b) account: /home/jbollen
   (c) Cannot touch other parts of system: System Administrator only
Basic UNIX directory commands

1. You have a position: the directory you are in
   > cd /home/jdoe/MyFiles
2. Make a directory:
   (a) `mkdir {name}`
   (b) creates directory in your present directory
   > mkdir AphexTwin
   > cd AphexTwin
   > ls
3. Move to another directory:
   (a) `cd {directory name}`
   (b) Absolute: directory name starting from root
   (c) Relative: ..: one down tree, .: present directory
   > cd ../Autechre
4. List content of present directory
   (a) `ls {directory name}`
   (b) list directories as well as files
4. Where am I?
   (a) `pwd`
   (b) List present directory

5. Remove file or directory
   (a) `rm` {file name}
   (b) `rm -r` {directory}

6. Copy file
   (a) `cp` {Old Name} {New Name}
   (b) Can include directory names

7. Rename or move file
   (a) `mv` {Old Name} {New Name}
   (b) Can include directory names

8. Use of asterisk ’*’ and ?
   (a) Search for patterns
   (b) e.g. `ls *.mp3` or `aphextwin?`
Some remarks on UNIX file attributes

1. Multi-user system file management issues:
   (a) File Ownership
   (b) File Access Permissions

2. UNIX solution:
   (a) Each file has Read, Write and Execute permission
   (b) Permission can be set for user, group and “others”
   (c) File carries owner’s username and group
Example:

-rw-rw-r-- 1 jbollen cs 3663 Feb 25 11:48 test.txt

<table>
<thead>
<tr>
<th>User</th>
<th>Group</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Owner</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>jbollen</td>
<td>cs</td>
</tr>
</tbody>
</table>

Use `ls -l` to retrieve access permissions and ownership for directory and files
Changing File Access Permissions and Ownership: chmod

1. CHMOD: changes access permissions
   (a) Use: chmod \{u,g,o,a\}+,-\{r,w,x\}
   (b) Example: chmod o-x test.txt, removes execute permissions for other for file test.txt
   (c) There’s a method using permission bits, 3 x 3 user: rwx, group: rwx etc...

2. Important for any UNIX related work
File Editing in UNIX

1. To view a file: use More
   (a) Example: `more my_file.txt`
   (b) Allows searching: stop forward slash and search string

2. To view top portion of file: use `head -n {no. lines} {file}`
   (a) Example: `head -n 5 test.txt`
   (b) Displays first 5 lines

3. To view bottom portion of file: use `tail -{no. lines} {file}`
   (a) Example: `tail -5 test.txt`
   (b) Display last 5 lines

4. Try this: `sort` a file

5. Or this: `uniq` it

6. Even better: chain commands using pipes!
   (a) piping: move output of one command to input of other
   (b) example: `tail -5 test.txt|sort|uniq`

7. Redirection: redirect output to file
   (a) redirection: take output and redirect to file
   (b) example: `tail -5 test.txt|sort|uniq > result.txt`
Pico file Editing in UNIX

1. This class will use the pico editor
   (a) Found on most UNIX systems
   (b) Great for editing and programming
   (c) Not as bad as Emacs but still learning curve: command line driven!

2. Activation: pico your_file.txt

3. Main Commands:
   (a) Open File: CTRL-R, Save File: CTRL-O
   (b) Stop Emacs: CRTL-X
You need your own UNIX system now: Clients and Servers

1. Client and server division of labour
   (a) Relation:
      i. Client: service request
      ii. Server: fulfills service
   (b) Interaction:
      i. Client-server download
      ii. Client-server upload
      iii. Two-way Interaction

2. Example: WWW
   (a) Server identified by machine name
       stores pages
   (b) URL describes both protocol, machine name and page filename
   (c) Client (IE Explorer, Netscape)
       make network request for page
   (d) Server responds and send page back
   (e) Client renders page
Some notes on Client-Server Protocols

1. Need for client and server to exchange information
   (a) Sequential exchange of requests and information
   (b) Standardized language for such exchanges and what constitutes valid sequences: protocol

2. Protocols: standard for client-server exchange

3. File Transfer:
   (a) FTP
   (b) HTTP (WWW),
   (c) POP (e-mail)

4. Remote control
   (a) telnet
   (b) X windows

5. Model for this class: telnet
Telnet

1. Purpose: allow users to control accounts remotely
   (a) Account is part of system
   (b) Accessible through shell or remote telnet

2. Extension of terminal concept to network-environment:
   (a) Local computer transfer keystrokes and command to remote server
   (b) Server returns updates environment
   (c) Real-time interaction

3. Can be confusing to keep track of which environment you are working in
Client - Server Model

Local Computer

Log-In

/home/jbollen

/home/jdoe

/home/edoe

Server

Local Computer

Log-In
Telnet: a practical overview

1. Activate local telnet client applications
2. Open connection to lab.cs.odu.edu
3. Login to registered CS account:
   (a) User name (do not confuse OCCS password!)
   (b) Password
4. From that point: full access to your account
5. Independent from local OS and hardware
6. Do not confuse with your local environment!
Windows Telnet
Windows Telnet

![Windows Telnet Screenshot]
Windows Telnet
Windows Telnet

[Image of a Windows Telnet window with a login prompt and system information]