CSE 390a
Lecture 1

introduction to Linux/Unix environment

slides created by Marty Stepp, modified by Jessica Miller
http://www.cs.washington.edu/390a/
Lecture summary

• Course introduction and syllabus

• Unix and Linux operating system

• introduction to Bash shell
Course Introduction

• Me:
  - Jessica Miller, jessica@cs
  - Office hours: TBA

• CSE390a
  - Collection of tools and topics not specifically addressed in other courses that CSE majors should know
    • *nix command line interface (CLI), Shell scripting, compilation tools (makefiles), version control...
  - Credit / No Credit course, determined by short weekly assignments and a “final” assignment
Operating systems

• What is an OS? Why have one?

• What is a Kernel?
Operating systems

• **operating system**: Manages activities and resources of a computer.
  - software that acts as an interface between hardware and user
  - provides a layer of abstraction for application developers

• **features provided by an operating system**:
  - ability to execute programs (and multi-tasking)
  - memory management (and virtual memory)
  - file systems, disk and network access
  - an interface to communicate with hardware
  - a user interface (often graphical)

• **kernel**: The lowest-level core of an operating system.
Unix

• brief history:
  ▪ Multics (1964) for mainframes
  ▪ Unix (1969)
  ▪ K&R
  ▪ Linus Torvalds and Linux (1992)

• key Unix ideas:
  ▪ written in a high-level language (C)
  ▪ virtual memory
  ▪ hierarchical file system; "everything" is a file
  ▪ lots of small programs that work together to solve larger problems
  ▪ security, users, access, and groups
  ▪ human-readable documentation included
On to Linux

LINUX: A TRUE STORY:
WEEK ONE
Hey, it's your cousin I got a new computer but don't want Windows. Can you help me install "Linux"?
Sure.

WEEK TWO
It says my Xorg is broken. What's an "Xorg"? Where can I look that up?
Hmm, lemme show you man pages.

WEEK SIX
Due to auto-config issues, I'm leaving Ubuntu for Debian.
Uh or Gentoo. Uh oh.

WEEK TWELVE
You haven't answered your phone in days. Can't sleep. Must compile kernel.
I'm too late.

Parents: talk to your kids about Linux... before somebody else does.
Linux

• **Linux**: A kernel for a Unix-like operating system.
  ▪ commonly seen/used today in servers, mobile/embedded devices, ...

• **GNU**: A "free software" implementation of many Unix-like tools
  ▪ many GNU tools are distributed with the Linux kernel

• **distribution**: A pre-packaged set of Linux software.
  ▪ examples: Ubuntu, Fedora

• key features of Linux:
  ▪ **open source software**: source can be downloaded
  ▪ free to use
  ▪ constantly being improved/updated by the community
Features of Linux

- X-windows
- window managers
- desktop environments
  - Gnome
  - KDE

- How can I try out Linux?
  - CSE basement labs
  - at home (install Linux via Live CD, virtual machine, etc.)
  - attu shared server

- The Linux help philosophy: "RTFM" (Read the F***ing Manual)
Exercises

- Install Linux and boot it up successfully.
- Load the course web site in Linux.
- Install a new game on Linux and play it.
- Get Linux to play an MP3.
Shell

- **shell**: An interactive program that uses user input to manage the execution of other programs.
  - bash: the default shell program on most Linux/Unix systems

- Why should I learn to use a shell when GUIs exist?
Shell

- **shell**: An interactive program that uses user input to manage the execution of other programs.
  - bash : the default shell program on most Linux/Unix systems

- Why should I learn to use a shell when GUIs exist?
  - faster
  - work remotely
  - programmable
  - customizable
  - repeatable
## Shell commands

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit</td>
<td>logs out of the shell</td>
</tr>
<tr>
<td>ls</td>
<td>lists files in a directory</td>
</tr>
<tr>
<td>pwd</td>
<td>outputs the current working directory</td>
</tr>
<tr>
<td>cd</td>
<td>changes the working directory</td>
</tr>
<tr>
<td>man</td>
<td>brings up the manual for a command</td>
</tr>
</tbody>
</table>

$ pwd
/homes/iws/dravir
$ cd CSE390
$ ls
file1.txt  file2.txt
$ ls -l
-rw-r--r-- 1 dravir vgrad_cs 0 2010-03-29 17:45 file1.txt
-rw-r--r-- 1 dravir vgrad_cs 0 2010-03-29 17:45 file2.txt
$ cd ..
$ man ls
$ exit
## Relative directories

<table>
<thead>
<tr>
<th>directory</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>the directory you are in (&quot;working directory&quot;)</td>
</tr>
<tr>
<td>..</td>
<td>the parent of the working directory (../.. is grandparent, etc.)</td>
</tr>
<tr>
<td>~</td>
<td>your home directory (on many systems, this is /home/username)</td>
</tr>
<tr>
<td>~username</td>
<td>username's home directory</td>
</tr>
<tr>
<td>~/Desktop</td>
<td>your desktop</td>
</tr>
</tbody>
</table>
Directory commands

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ls</td>
<td>list files in a directory</td>
</tr>
<tr>
<td>pwd</td>
<td>output the current working directory</td>
</tr>
<tr>
<td>cd</td>
<td>change the working directory</td>
</tr>
<tr>
<td>mkdir</td>
<td>create a new directory</td>
</tr>
<tr>
<td>rmdir</td>
<td>delete a directory (must be empty)</td>
</tr>
</tbody>
</table>

• some commands (cd, exit) are part of the shell ("builtins")
• others (ls, mkdir) are separate programs the shell runs
Shell commands

• many accept **arguments** or **parameters**
  ▪ example: `cp` (copy) accepts a source and destination file path

• a program uses 3 streams of information:
  ▪ stdin, stdout, stderr (standard in, out, error)

• **input**: comes from user's keyboard
• **output**: goes to console
• **errors** can also be printed (by default, sent to console like output)

• parameters vs. input
  ▪ **parameters**: before Enter is pressed; sent in by shell
  ▪ **input**: after Enter is pressed; sent in by user
Command-line arguments

- most options are a - followed by a letter such as -c
  - some are longer words preceded by two - signs, such as --count

- parameters can be combined: `ls -l -a -r` can be `ls -lar`

- many programs accept a --help or -help parameter to give more information about that command (in addition to man pages)
  - or if you run the program with no arguments, it may print help info

- for many commands that accept a file name parameter, if you omit the parameter, it will read from standard input (your keyboard)
# Shell/system commands

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>man or info</td>
<td>get help on a command</td>
</tr>
<tr>
<td>clear</td>
<td>clears out the output from the console</td>
</tr>
<tr>
<td>exit</td>
<td>exits and logs out of the shell</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>output the system date</td>
</tr>
<tr>
<td>cal</td>
<td>output a text calendar</td>
</tr>
<tr>
<td>uname</td>
<td>print information about the current system</td>
</tr>
</tbody>
</table>

- "man pages" are a very important way to learn new commands
  - `man ls`
  - `man man`
File commands

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cp</td>
<td>copy a file</td>
</tr>
<tr>
<td>mv</td>
<td>move or rename a file</td>
</tr>
<tr>
<td>rm</td>
<td>delete a file</td>
</tr>
<tr>
<td>touch</td>
<td>create a new empty file, or update its last-modified time stamp</td>
</tr>
</tbody>
</table>

- caution: the above commands do not prompt for confirmation
  - easy to overwrite/delete a file; this setting can be overridden (how?)

- Exercise: Given several albums of .mp3 files all in one folder, move them into separate folders by artist.
- Exercise: Modify a .java file to make it seem as though you finished writing it on Dec 28 at 4:56am.