
CSE 391

Lecture 1

introduction to Linux/Unix environment

slides created by Marty Stepp, modified by Jessica Miller, Ruth Anderson, and Brett Wortzman

<http://www.cs.washington.edu/391/>

Lecture summary

- Course introduction and syllabus
- Unix and Linux operating system
- Introduction to Bash shell

Course Introduction

- Instructor:
 - Brett Wortzman, brettwo@cs, CSE446
 - Office hours: TBD
- TA:
 - Omeed Magness, omag01@cs
- Website: <http://cs.washington.edu/391>
- Collection of tools and topics not specifically addressed in other courses that CSE majors (and interested others) should know
 - CSE 351 may be the first course you take that uses Linux
- Credit / No Credit course, determined by weekly assignments
 - Graded primarily on effort/completion
- “Textbook” – *Linux Pocket Guide*
 - Optional but recommended; very useful guide

Course Topics

- Linux command line interface (CLI)
- Shell commands
- Users and groups
- Permissions
- Shell scripting
- Regular expressions
- Project management tools (e.g. makefiles)
- Version control (e.g. git)

Homework/Grading

- ~Nine weekly assignments
 - Released after lecture
 - Due following Tuesday, 11:59pm (no late work accepted)
- Based on material covered in that week's lecture
 - A few “self-discovery” extensions
 - All required information in lecture, slides, book, and/or man pages
- Graded out of 2 points each
 - Primarily determined by effort/completion (see syllabus)
 - Total of 14 points required to receive credit
- To be completed on Linux/Unix systems (next slide)
- Collaboration allowed/encouraged, but **ALL SUBMITTED WORK MUST BE YOUR OWN**

Accessing Linux/Unix

Roughly in suggested order...

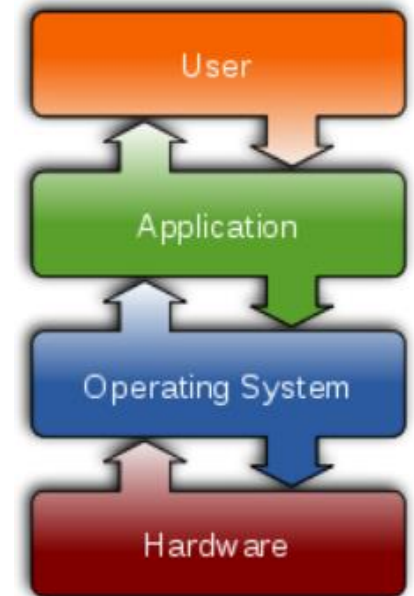
- `ssh` to attu (CSE majors), linuxNN (EE majors), or ovid (all UW students)
- Download/run CSE VM
- Visit CS or EE basement labs
- Set up Linux on your own machine
- See “Working at Home” on course website for more info

Operating systems

- What is an OS? Why have one?
- What is a Kernel?

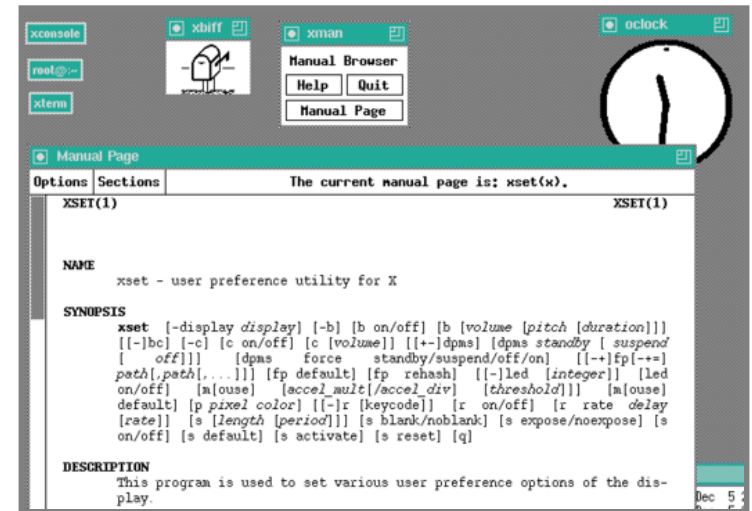
Operating systems

- **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



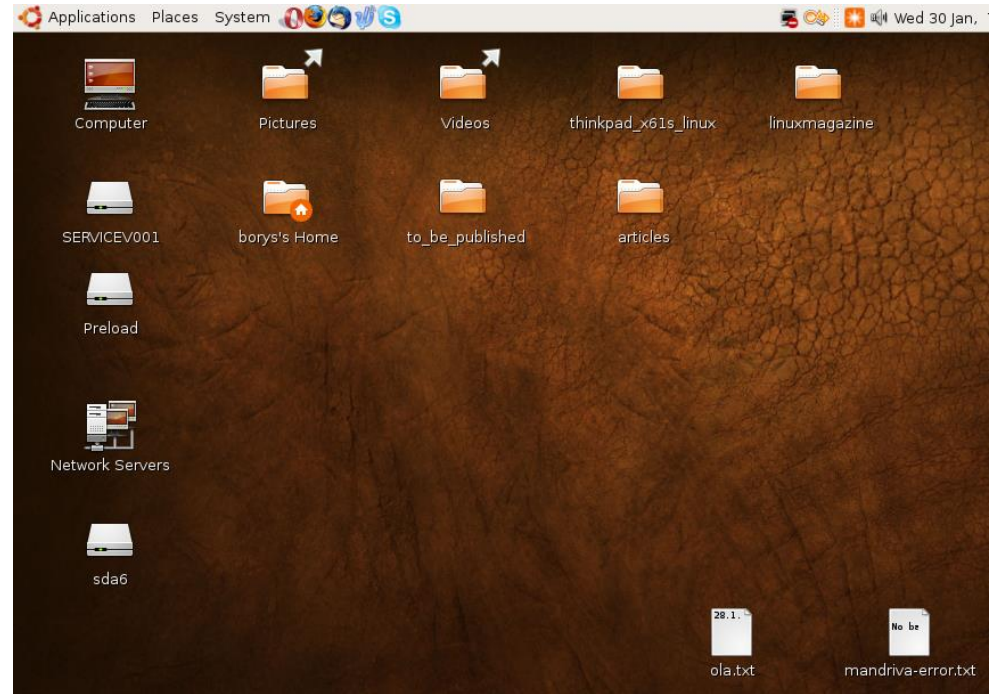
Linux

- **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, CentOS
- key features of Linux:
 - **open source software:** source can be downloaded
 - free to use
 - constantly being improved/updated by the community



Linux Desktop

- X-windows
- window managers
- desktop environments
 - Gnome
 - KDE
- How can I try out Linux?
 - CSE Virtual machine
 - CSE basement labs
 - attu shared server



Things you can do in Linux

- Load the course web site in a browser
- Install and play games
- Play MP3s
- Edit photos
- IM, Skype

Shell

- **shell:** An interactive program that uses user input to manage the execution of other programs.
 - A command processor, typically runs in a text window.
 - User types commands, the shell runs the commands
 - Several different shell programs exist:
 - bash : the default shell program on most Linux/Unix systems
 - We will use bash
 - Other shells: Bourne, csh, tsch
- Why should I learn to use a shell when GUIs exist?

Why use a shell?

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

Example shell commands

command	description
pwd	<u>p</u> rint the current <u>w</u> orking <u>d</u> irectory
cd	<u>c</u> hanges the working <u>d</u> irectory
ls	lists files in a directory
man	brings up the manual for a command
exit	logs out of the shell

```
$ pwd
/homes/iws/rea
$ cd CSE391
$ ls
file1.txt file2.txt
$ ls -l
-rw-r--r-- 1 rea    fac_cs 0 2017-03-29 17:45 file1.txt
-rw-r--r-- 1 rea    fac_cs 0 2017-03-29 17:45 file2.txt
$ cd ..
$ man ls
$ exit
```

System commands

command	description
<code>man</code> or <code>info</code>	get help on a command
<code>clear</code>	clears out the output from the console
<code>exit</code>	exits and logs out of the shell
<code>date</code>	output the system date
<code>cal</code>	output a text calendar
<code>uname</code>	print information about the current system

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

Relative directories

directory	description
.	the directory you are in ("working directory")
..	the parent of the working directory (../.. is grandparent, etc.)
~	your <u>home</u> directory (on many systems, this is /home/ <i>username</i>)
~ <i>username</i>	<i>username</i> 's <u>home</u> directory
~/Desktop	your desktop

Unix file system

directory	description
/	root directory that contains all others (drives do not have letters in Unix)
/bin	programs
/dev	hardware devices
/etc	system configuration files <ul style="list-style-type: none">▪ /etc/passwd stores user info▪ /etc/shadow stores passwords
/home	users' home directories
/media, /mnt, ...	drives and removable disks that have been "mounted" for use on this computer
/proc	currently running processes (programs)
/tmp, /var	temporary files
/usr	user-installed programs

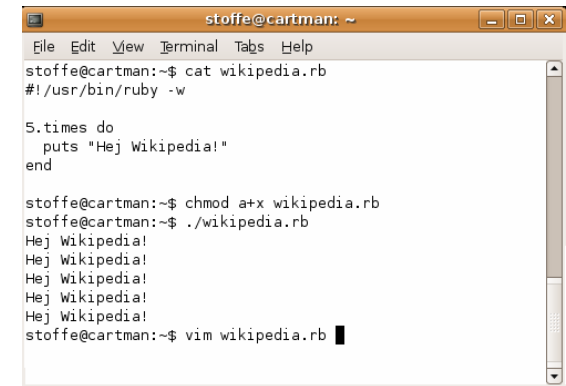
Directory commands

command	description
ls	list files in a directory
pwd	<u>p</u> rint the current <u>w</u> orking <u>d</u> irectory
cd	<u>c</u> hanges the working <u>d</u> irectory
mkdir	create a new directory
rmdir	delete a directory (must be empty)

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

Command-line arguments

- 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

A terminal window titled 'stoffe@cartman: ~' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following sequence of commands and output:

```
stoffe@cartman:~$ cat wikipedia.rb
#!/usr/bin/ruby -w

5.times do
  puts "Hej Wikipedia!"
end

stoffe@cartman:~$ chmod a+x wikipedia.rb
stoffe@cartman:~$ ./wikipedia.rb
Hej Wikipedia!
Hej Wikipedia!
Hej Wikipedia!
Hej Wikipedia!
Hej Wikipedia!
stoffe@cartman:~$ vim wikipedia.rb
```

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
- options can be combined: `ls -l -a -r` can be `ls -lar`
- many programs accept a --help or -help option 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 argument, if you omit the parameter, it will read from standard input (your keyboard)

File commands

command	description
cp	copy a file
mv	move or rename a file
rm	delete a file
touch	create a new empty file, or update its last-modified time stamp

- 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.

Exercise Solutions

- caution: the cp, rm, mv commands do not prompt for confirmation
 - easy to overwrite/delete a file; this setting can be overridden (how?)
 - Use “-i” with the command, “interactive” to prompt before overwrite
- *Exercise* : Given several albums of .mp3 files all in one folder, move them into separate folders by artist.
 - `mkdir U2`
 - `mkdir PSY`
 - `mkdir JustinBieber`
 - `mv GangnamStyle.mp3 PSY/`
 - `mv Pride.mp3 U2/`
- *Exercise* : Modify a .java file to make it seem as though you finished writing it on Dec 28 at 4:56am.
 - `touch -t "201812280456" Hello.java`