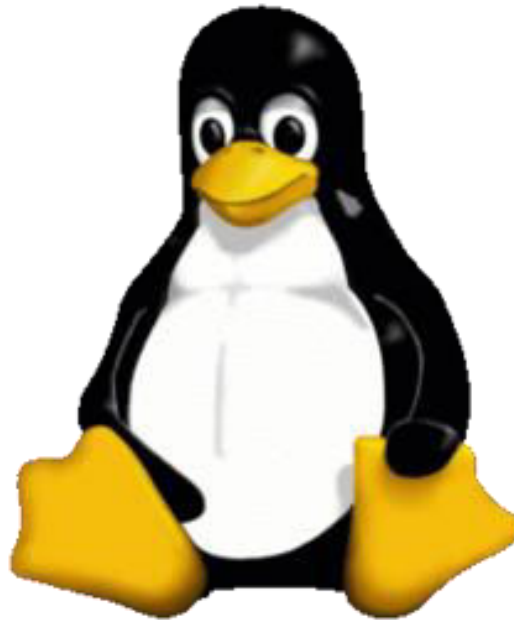


Introduction to Linux



Ms Krushima Soma

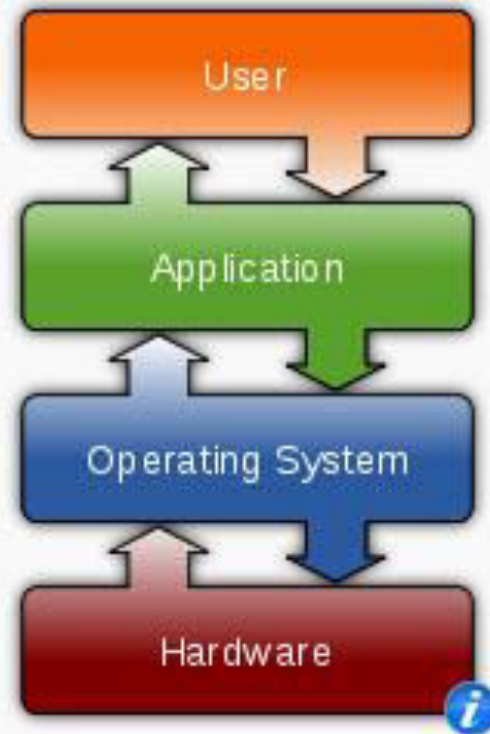
About the class...

- ▶ We'll start with a sign in sheet that include questions about your Linux experience and goals.
- ▶ We'll end with a class evaluation.
- ▶ We'll cover as much as we can in the time allowed, starting with the easiest and most important material. Don't feel rushed; if we don't cover everything, you'll pick it up as you continue working with Linux.
- ▶ This is a hands-on, lab class; ask questions at any time.
- ▶ Commands for you to type are in **BOLD**
- ▶ We'll take a break at the half-way point.

What is Linux?

It's an Operating System

Operating systems

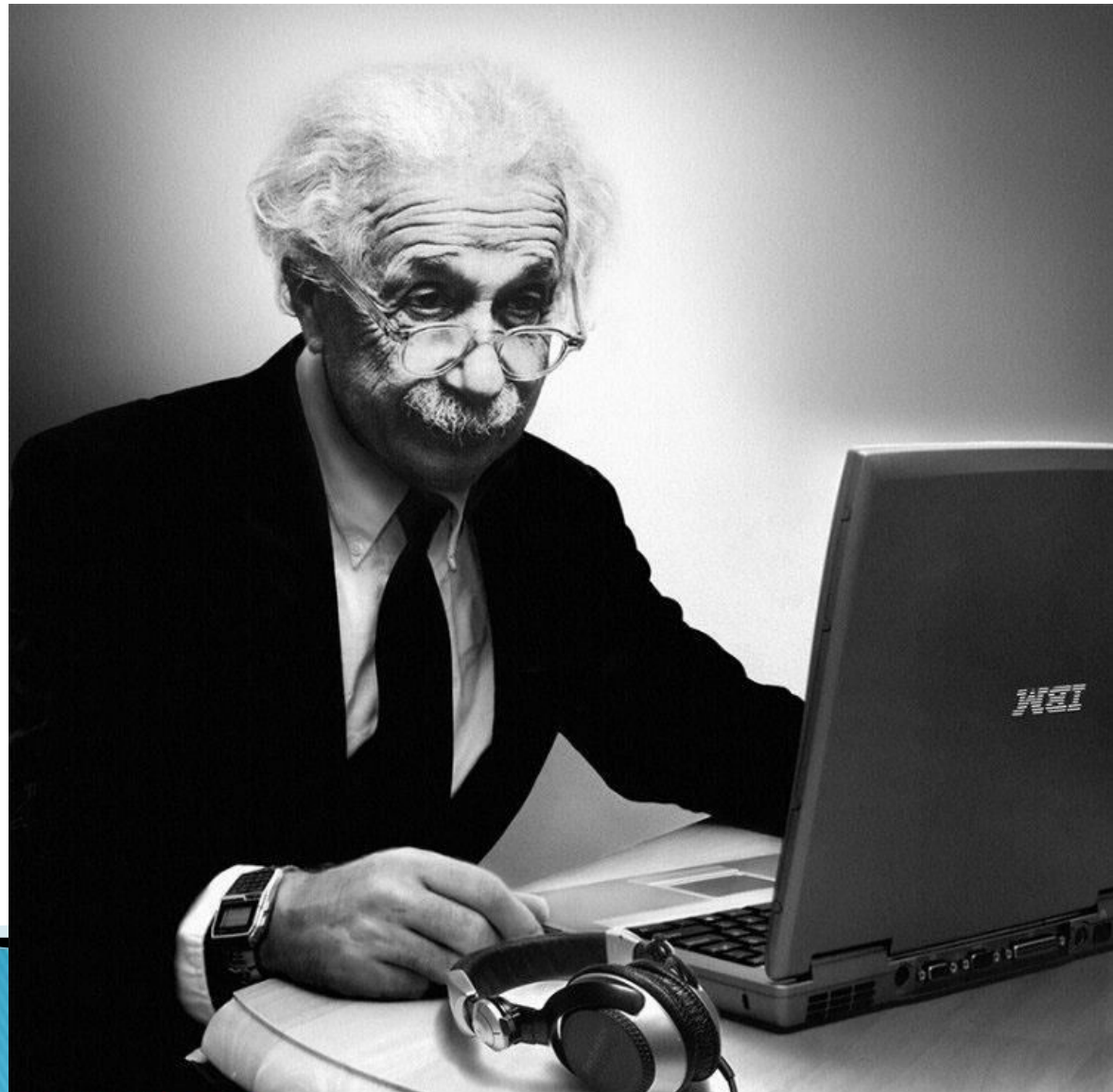


Common features

- Process management
- Interrupts
- Memory management
- File system
- Device drivers
- Networking (TCP/IP, UDP)
- Security (Process/Memory protection)
- I/O



What is Linux?



The Most
Common O/S
Used By BU
Researchers When
Working on a
Server or
Computer Cluster

What is Linux?

- ▶ Linux is a Unix clone written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net.
- ▶ Unix is a multitasking, multi-user computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs.
- ▶ Linux and Unix strive to be POSIX compliant.
- ▶ 64% of the world's servers run some variant of Unix or Linux. The Android phone and the Kindle run Linux.



The Linux Philosophy

*The *Nix Philosophy of Doug McIlroy*

- (i) Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new features.
- (ii) Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.
- (iii) Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

Linux Has Many Distributions



Linux Has Many Distributions

BU uses CentOS in its Linux cluster which is a free version of RedHat Enterprise Linux with the trademarks removed



What is Linux?

Linux + GNU Utilities = Free Unix



- ▶ Linux is an O/S core written by Linus Torvalds and others AND



- ▶ a set of small programs written by Richard Stallman and others. They are the GNU utilities.

<http://www.gnu.org/>

What is Linux?

“Small programs that do one thing well”

(see [unix-reference.pdf](#))

- ▶ **Network:** ssh, scp, ping, telnet, nslookup, wget
- ▶ **Shells:** BASH, TCSH, alias, watch, clear, history, chsh, echo, set, setenv, xargs
- ▶ **System Information:** w, whoami, man, info, which, free, echo, date, cal, df, free, man, info
- ▶ **Command Information:** man, info
- ▶ **Symbols:** |, >, >>, <, &, >&, 2>&1, ;, ~, ., .., \$!, !:<n>, !<n>
- ▶ **Filters:** grep, egrep, more, less, head, tail
- ▶ **Hotkeys:** <ctrl><c>, <ctrl><d>
- ▶ **File System:** ls, mkdir, cd, pwd, mv, ln, touch, cat, file, find, diff, cmp, /net/<hostname>/<path>, mount, du, df, chmod, find
- ▶ **Line Editors:** awk, sed
- ▶ **File Editors:** vim, gvim, emacs -nw, emacs

What is Linux?

“Small programs that do one thing well”

We will not cover the commands below in this class, but you need to know them. See the **man** pages for the process commands and the “sge” folder inside of the “cheat sheets and tutorials” folder for the SGE (Sun Grid Engine) command tutorials: qsh-interactive.pdf, qsh-interactive-matlab.pdf, qsub-batch.pdf, qsub-batch-matlab.pdf, and qstat-qhost.pdf.

- ▶ **Process Management:** ps, top, kill, killall, fg, bg
- ▶ **SGE Cluster:** qsh, qstat, qsub, qhost

Connecting to a Linux Host

- ▶ You need a “xterm” emulator: software that emulates an “X” terminal and connects using the “SSH” secure shell protocol.
- ▶ You are sitting at the “client,” either a Windows, Macintosh or even possibly a Linux machine.
- ▶ You are connecting to a “server,” typically the “head” or “gateway” node of a cluster of computers. You will be working on the head node or submitting jobs to execution nodes, all of them, Linux machines.
- ▶ You can also connect to a Linux machine by using VNC to get a whole desktop if it’s supported by the server.

Connecting to a Linux Host – Windows Client Software

- ▶ You need a “xterm” emulation – software that emulates an “X” terminal and that connects using the “SSH” Secure Shell protocol.
 - Windows
 - If you don’t need windowing, “putty” is good:
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>
 - If you need windowing, use StarNet “X-Win32:”
<http://www.bu.edu/tech/desktop/site-licensed-software/xwindows/xwin32/>

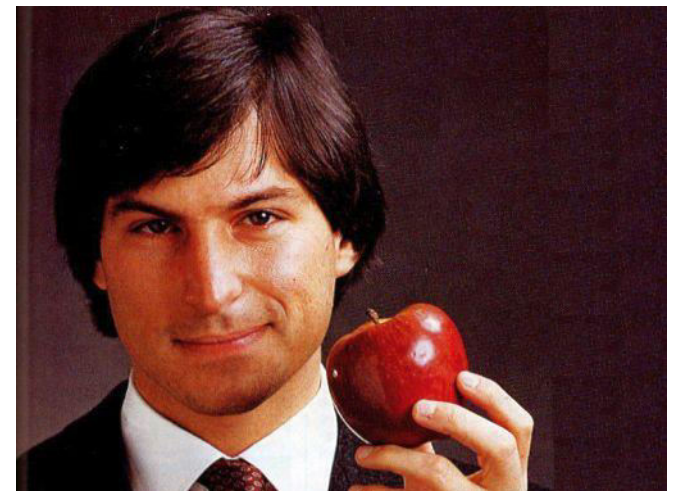




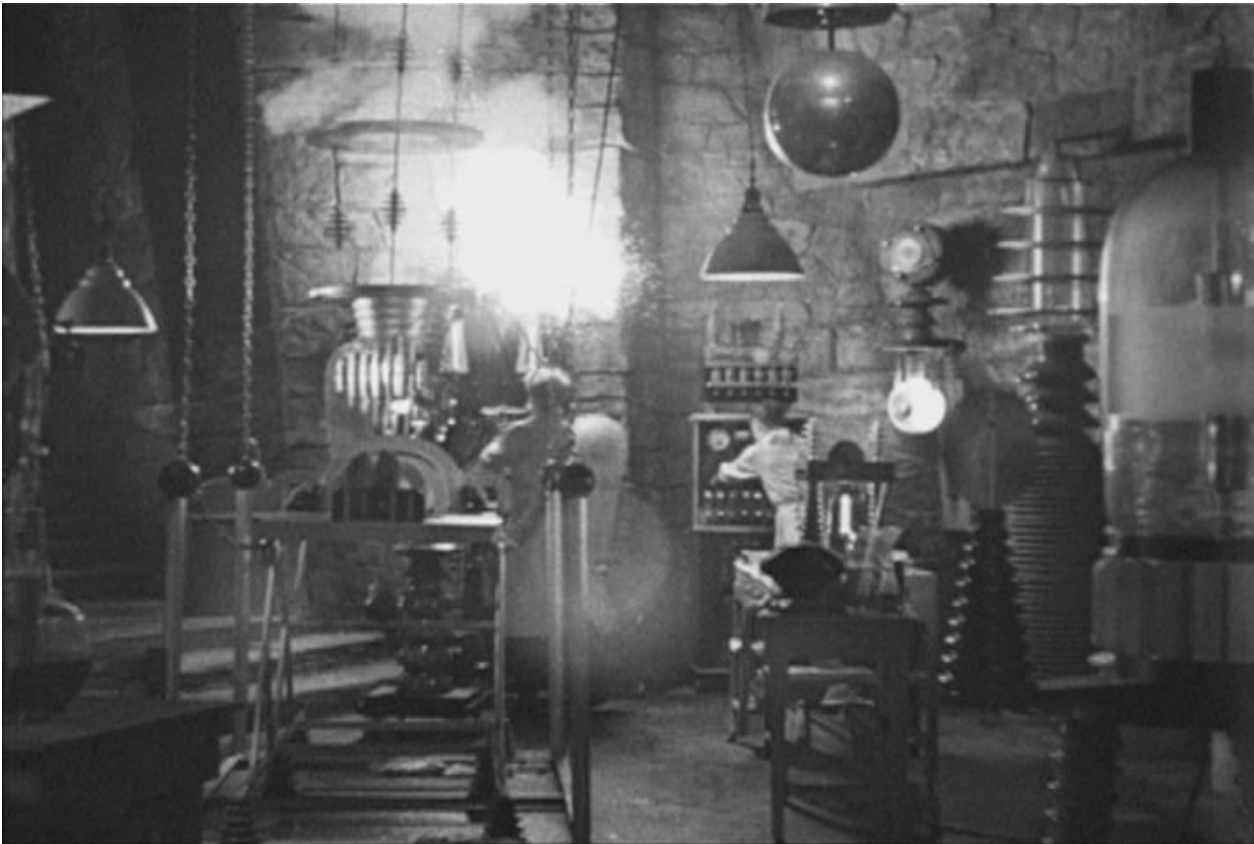
Connecting to a Linux Host – Mac OS X Client Software



- Mac OS X
 - “Terminal” is already installed
 - Why? Darwin, the system on which Apple's Mac OS X is built, is a derivative of 4.4BSD-Lite2 and FreeBSD. In other words, the Mac is a Unix system!



Let the Linux Lab Begin!



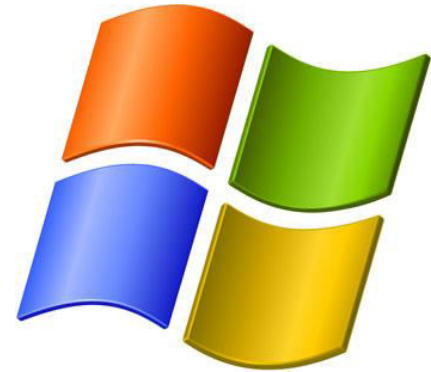
The Ideal Lab Facility



Your Instructor Today



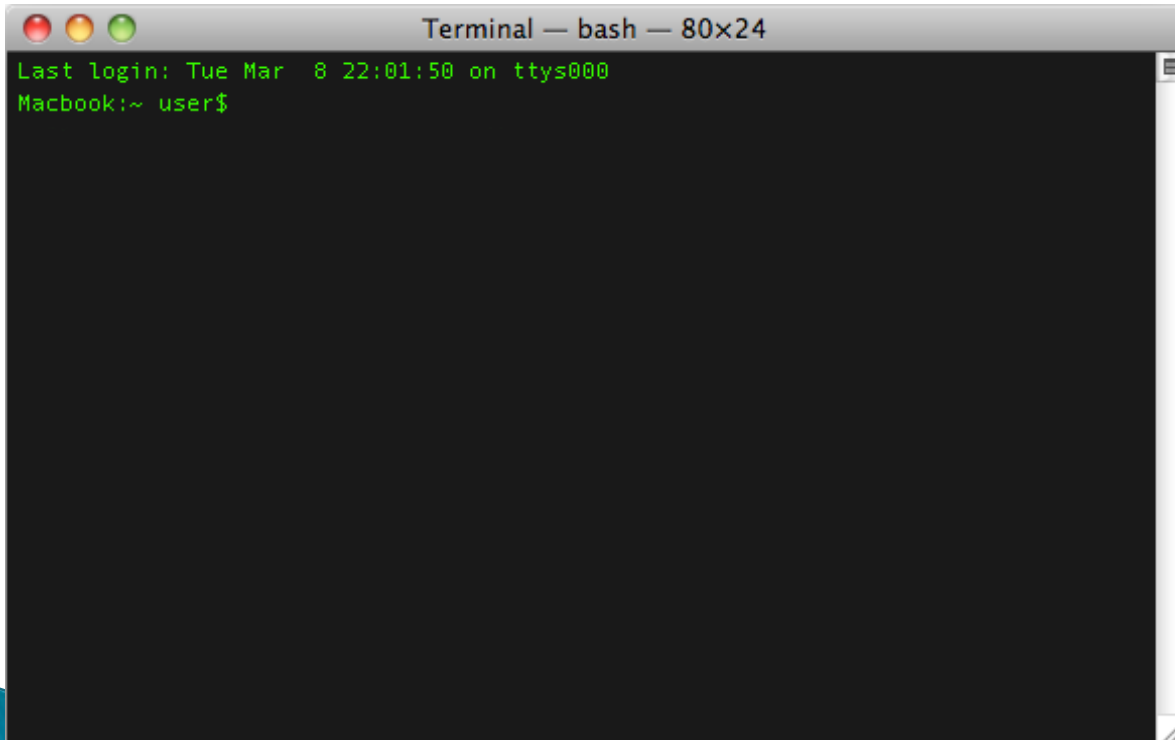
Connecting to a Linux Host – Windows Client



- ▶ X-Win32 / X-Config
 - Wizard
 - Name: katana
 - Type: ssh
 - Host: katana.bu.edu (Off-campus, must include domain “bu.edu”)
 - Login: <userID>
 - Password: <password>
 - Command: Linux
 - Click “katana” then “Launch”
 - Accept the host server public key (first time only)

Connecting to a Linux Host – Mac OS X Client

- ▶ Terminal
 - Type `ssh -X katana.bu.edu` or `ssh -Y katana.bu.edu` (less secure)



```
Terminal — bash — 80x24
Last login: Tue Mar  8 22:01:50 on ttys000
Macbook:~ user$
```





Connection Problems

When there are problems connecting to a login host, try:

- ▶ ping katana.bu.edu
- ▶ telnet katana.bu.edu 22

```
donj@crsos:~$ ping katana.bu.edu
PING katana.bu.edu (128.197.160.66) 56(84) bytes of data.
64 bytes from katana.bu.edu (128.197.160.66): icmp_seq=1 ttl=61 time=0.254 ms
64 bytes from katana.bu.edu (128.197.160.66): icmp_seq=2 ttl=61 time=0.250 ms
64 bytes from katana.bu.edu (128.197.160.66): icmp_seq=3 ttl=61 time=0.243 ms

--- katana.bu.edu ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.243/0.249/0.254/0.004 ms
donj@crsos:~$ telnet katana.bu.edu 22
Trying 128.197.160.66...
Connected to katana.bu.edu.
Escape character is '^]'.
SSH-1.99-OpenSSH_4.3
^]
telnet> quit
Connection closed.
donj@crsos:~$
```

Obtaining the Course Material

▶ Windows

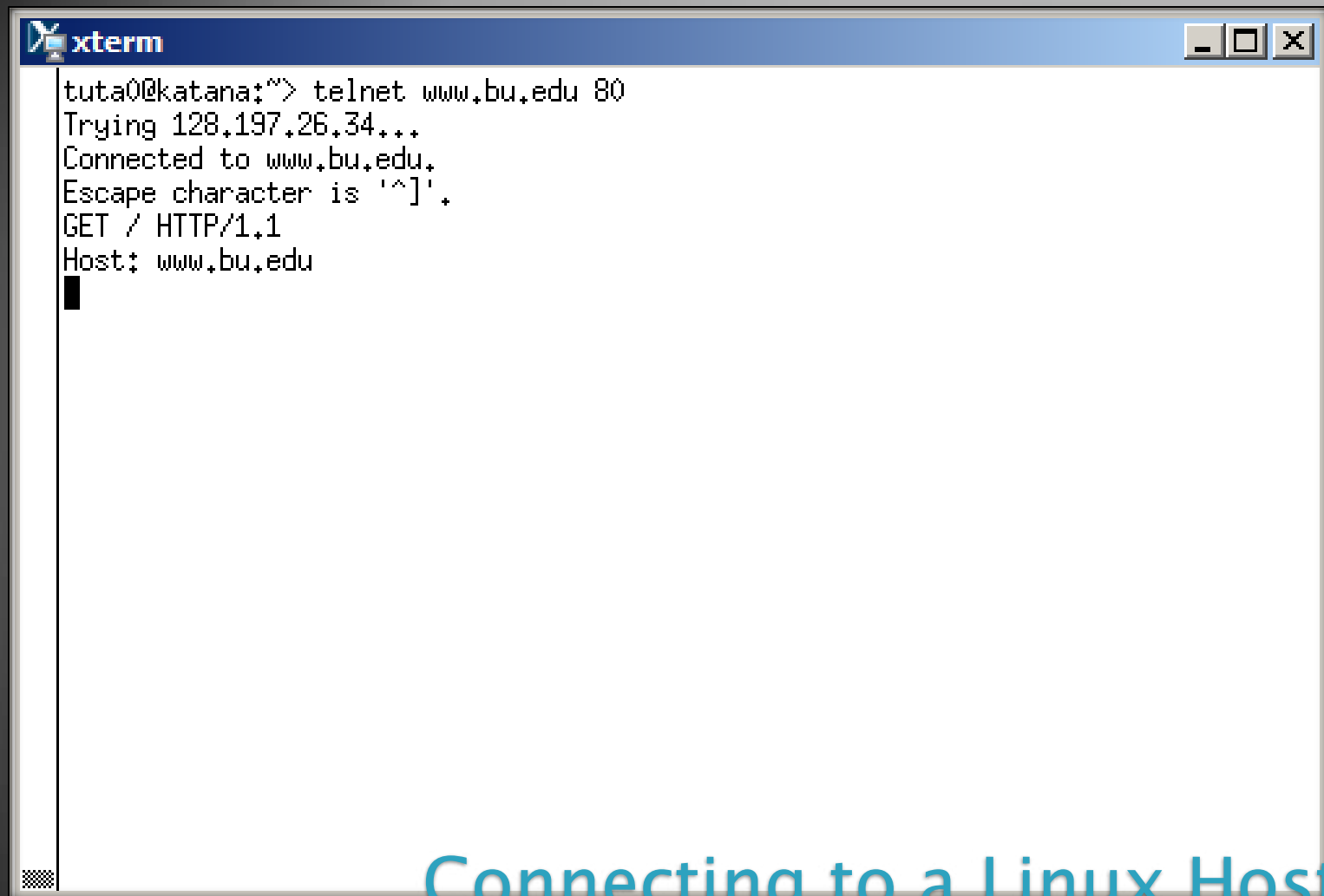
- Using File Explorer, copy the directory “\\scv-files.bu.edu\SCV\Training\Introduction to Linux” to “My Documents” on your lab machine

▶ Linux

- Connect to **katana.bu.edu** using X-Win32 and run this command:
 - **cp -Rv /project/ssrcsupp/linux_class ~/**

Connecting to a Linux Host: Emulate a Browser

- ▶ Note: <CR> is short for “carriage return” and equals the ASCII press the “Enter” or “Return” key. It tells the shell that you finished sending one line (see [ascii-table.pdf](#)).
- ▶ Try
 - **telnet www.bu.edu**
 - **GET / HTTP/1.1**
 - **Host:www.bu.edu<CR>**
 - **<CR>**
- ▶ What happened?



The image shows a terminal window titled 'xterm' with standard window controls. The terminal text is as follows:

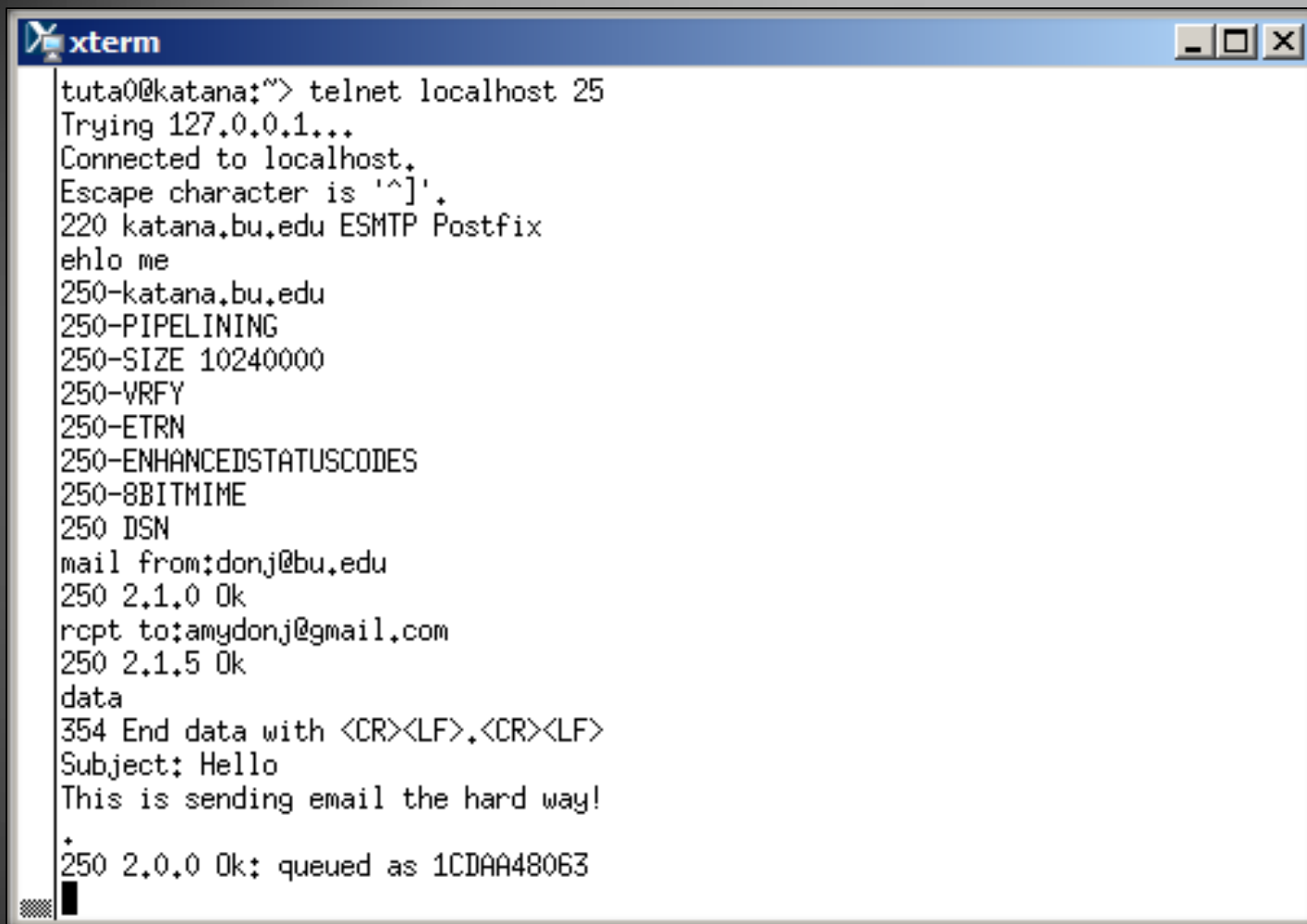
```
tuta0@katana:~> telnet www.bu.edu 80
Trying 128.197.26.34...
Connected to www.bu.edu.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.bu.edu
█
```

Connecting to a Linux Host >>

Emulate a Browser

Connecting to a Linux Host: Send and Email

- ▶ Try
 - telnet localhost 25
 - ehlo me
 - mail from:<your email address>
 - rcpt to:<destination email address>
 - data
 - Subject:<subject of email>
 - <Body of email>
 - .
 - <CR>
- ▶ What Happened?



```
xterm
tuta00@katana:~> telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 katana.bu.edu ESMTP Postfix
ehlo me
250-katana.bu.edu
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
mail from:donj@bu.edu
250 2.1.0 Ok
rcpt to:amydonj@gmail.com
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: Hello
This is sending email the hard way!
.
250 2.0.0 Ok: queued as 1CDAA48063
```

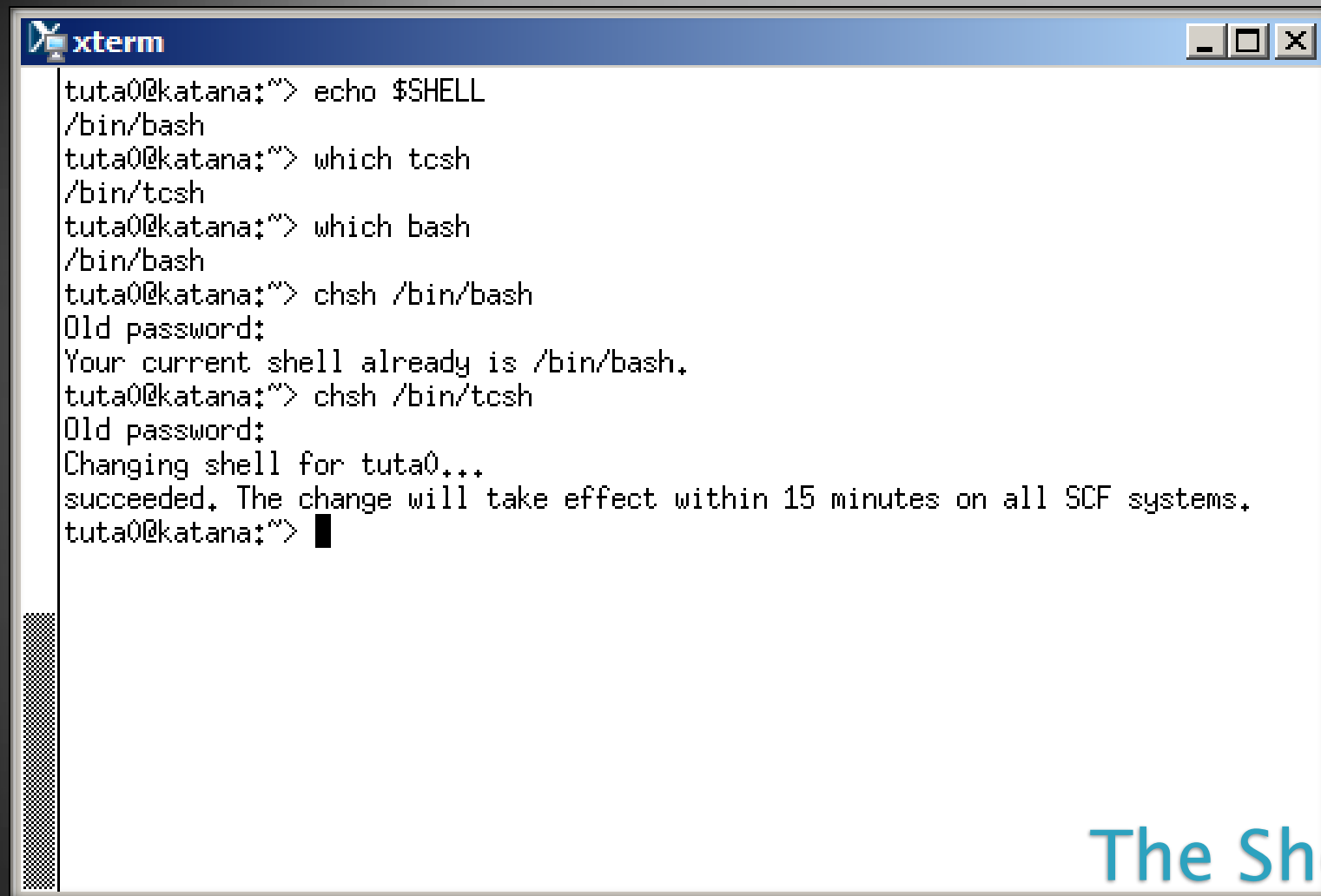
Connecting to an Linux Host >>

Send and Email



The Shell

- ▶ A shell is a computer program that interprets the commands you type and sends them to the operating system. Secondly, it provide a programming environment consisting of environment variables.
- ▶ Most BU systems, including the BU Linux Cluster, support at least two shells: TCSH and BASH. The default shell for your account is TCSH. The most popular and powerful Linux shell today is BASH.
- ▶ To determine your shell type:
 - `echo $SHELL` (shell prints contents of env
 - `echo "$SHELL"` (shell still processes env. variable)
 - `echo '$SHELL'` (shell treats env. variable as simple literal)
- ▶ The complete environment can be printed with **set**, **setenv** (TCSH) and **set** (BASH).
- ▶ To determine the path to the shell program, type:
 - `which bash`
 - `which tcsh`
- ▶ Change the shell with "**chsh /bin/bash**" (provide path to new shell as a "parameter," meaning to be explained soon)

A screenshot of an xterm window. The title bar says 'xterm'. The terminal content shows a user 'tuta0' at host 'katana' running several commands. The first command is 'echo \$SHELL', which outputs '/bin/bash'. The second is 'which tcsh', outputting '/bin/tcsh'. The third is 'which bash', outputting '/bin/bash'. The fourth is 'chsh /bin/bash', which prompts for an old password and then outputs 'Your current shell already is /bin/bash.'. The fifth is 'chsh /bin/tcsh', which prompts for an old password and then outputs 'Changing shell for tuta0... succeeded. The change will take effect within 15 minutes on all SCF systems.'. The prompt returns to 'tuta0@katana:~>'.

```
tuta0@katana:~> echo $SHELL
/bin/bash
tuta0@katana:~> which tcsh
/bin/tcsh
tuta0@katana:~> which bash
/bin/bash
tuta0@katana:~> chsh /bin/bash
Old password:
Your current shell already is /bin/bash.
tuta0@katana:~> chsh /bin/tcsh
Old password:
Changing shell for tuta0...
succeeded. The change will take effect within 15 minutes on all SCF systems.
tuta0@katana:~> █
```

The Shell >>

Output of the echo, which and chsh commands

System Information

- ▶ After you connect, type
 - shazam
 - whoami
 - hostname
 - date
 - cal
 - free
- ▶ Commands have three parts; *command*, *options* and *parameters*. Example: **cal -j 3 1999**. “cal” is the command, “-j” is an option (or switch), “3” and “1999” are parameters.
- ▶ Options have long and short forms. Example:
 - **date -u**
 - **data --universal**

What is the nature of the prompt?

What was the system's response to the command?

```
xterm
tuta0@katana:~$ shazam
-bash: shazam: command not found
tuta0@katana:~$ whoami
tuta0
tuta0@katana:~$ hostname
katana
tuta0@katana:~$ date
Sun Sep  9 12:14:51 EDT 2012
tuta0@katana:~$ cal
    September 2012
Su Mo Tu We Th Fr Sa
                1
 2  3  4  5  6  7  8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
tuta0@katana:~$ free
              total        used        free      shared    buffers     cached
Mem:      8109704      7759508      350196           0       100204      5794856
-/+ buffers/cache:      1864448      6245256
Swap:      9244240       502656      8741584
tuta0@katana:~$ date -u
Sun Sep  9 16:15:01 UTC 2012
tuta0@katana:~$ date --universal
Sun Sep  9 16:15:07 UTC 2012
tuta0@katana:~$ █
```

System Information >>

Output of the whoami, hostname, date, cal and free

Command History and Simple Command Line Editing

- ▶ Try the **history** command
- ▶ Try **<Ctrl><r>** (only works in BASH shell)
- ▶ Choose from the command history by using the up ↑ and down ↓ arrows
- ▶ What do the left ← and right → arrow do on the command line?
- ▶ Try the **** and **<Backspace>** keys

Help with Commands

- ▶ Type
 - `hostname --help`
 - `man hostname`
 - `info hostname` (gives the same or most information, but must be paged)
- ▶ And “Yes,” you can always Google it





Connect Commands Together with the Pipe Symbol “|” and Using Filters

- ▶ The pipe “|” feeds the OUTPUT of one command into the INPUT of another command. Our first example will use the pipe symbol to filter the output of a command. Try:
 - **w**
 - **w | grep 'root'**
 - **ps -e -o ruser,comm | grep 'tut'**
- ▶ The **ps** command is using both “options (dash)” and parameters
- ▶ Try both “**man grep**” and “**info grep**”. See the difference?



Editing the Command Line with Emacs Keys (see emacs-editing-mode.pdf)

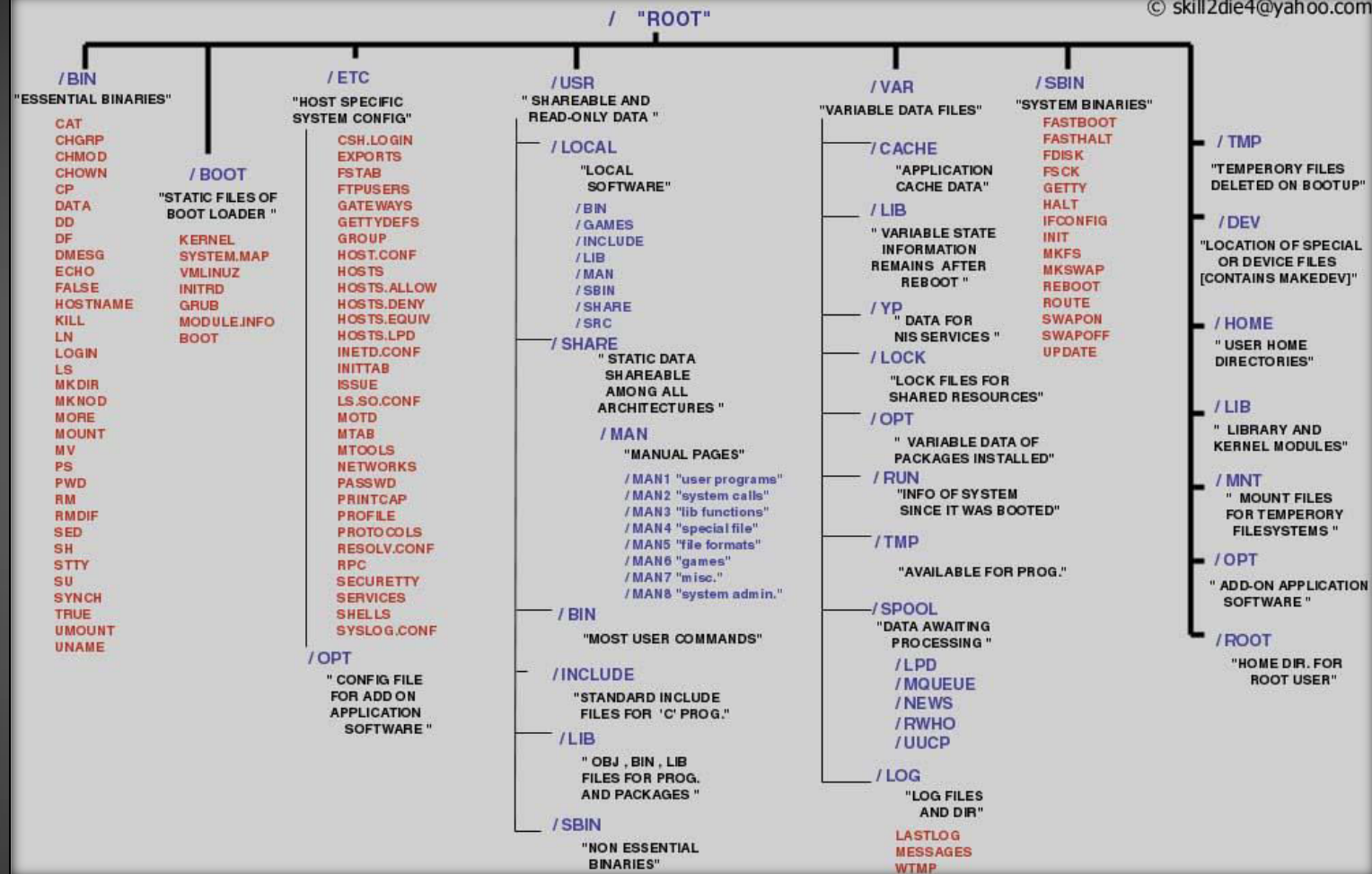
- ▶ **<Ctrl-a>** go to beginning
- ▶ **<Ctrl-e>** go to end
- ▶ **<Alt-f>** forward one word
- ▶ **<Alt-b>** back one word
- ▶ **<Ctrl-f>** forward one character
- ▶ **<Ctrl-b>** back one character
- ▶ **<Ctrl-d>** delete character
- ▶ **<Alt-d>** delete word
- ▶ **<Ctrl-u>** delete from cursor to beginning of line
- ▶ **<Ctrl-k>** delete from cursor to end of line

See emacs-editing-mode.pdf and emacs-editing-mode-short.pdf

Go to through command history in shell and practice editing.

The Linux File System

- ▶ The *Nix (Unix or Linux) file system is a hierarchical directory structure
- ▶ The structure resembles an upside down tree
- ▶ Directories are collections of files and other directories. The structure is recursive with many levels.
- ▶ Every directory has a parent except for the root directory.
- ▶ Many directories have children directories.
- ▶ Unlike Windows, with multiple drives and multiple file systems, a *Nix system only has ONE file system.
- ▶ The Linux Standard Base (LSB) specifies the structure of a Linux file system.



The Linux File System >>

A Typical Linux File System

Examining the File System

► Try

- `nautilus --browser --no-desktop`
- `tree -L 3 -d / | less`
- `tree -L 3 / | less`
- `file /bin/alsac` then press **<tab>**
- `cd ~; pwd` (This is your home directory where application settings are kept and where you have write privileges)
- `ls`
- `mkdir myPics;mkdir myPics/work;mkdir myPics/friends;mkdir myPics/friends/BU; mkdir myPics/friends/MIT`
- `tree myPics`

```
xterm
katana:~ % tree -L 3 -d / | head -3
/
|-- bin
|-- boot
katana:~ % tree -L 3 / | head -3
/
|-- bin
|   |-- alsacard
katana:~ % file /bin/alsacard
/bin/alsacard: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV), for GNU/
Linux 2.6.9, dynamically linked (uses shared libs), for GNU/Linux 2.6.9, strippe
d
katana:~ % cd ~ ; pwd
/usr4/tutorial/tuta0
katana:~ % ls
Desktop/          bupage.html      helloworld.c      untitled folder/
bin/              helloworld*      test.sh*
katana:~ % mkdir myPics ; mkdir myPics/work ; mkdir myPics/friends ; mkdir myPic
s/friends/BU ; mkdir myPics/friends/MIT
katana:~ % tree myPics
myPics
|-- friends
|   |-- BU
|   |-- MIT
|-- work

4 directories, 0 files
katana:~ % █
```

Examining the File System >>

Output from the tree, file, pwd and ls commands
Demonstration of using the mkdir command

Navigating the File System

- ▶ There are two types of pathnames
 - Absolute (Abs) – the full path to a directory or file; begins with the root symbol /
 - Relative (Rel) – a partial path that is relative to the current working directory
- ▶ Examples
 - Abs **cd /usr/local/lib**
 - **echo \$HOME** (one of many environment variables maintained by the shell)
 - Abs **cd `echo \$HOME`**
 - **pwd**
 - Rel **cd ..**
 - Rel **cd ..**
 - Abs **cd /lib** (location OS shared libraries)
 - **ls -d */** (a listing of only the directories in /lib)

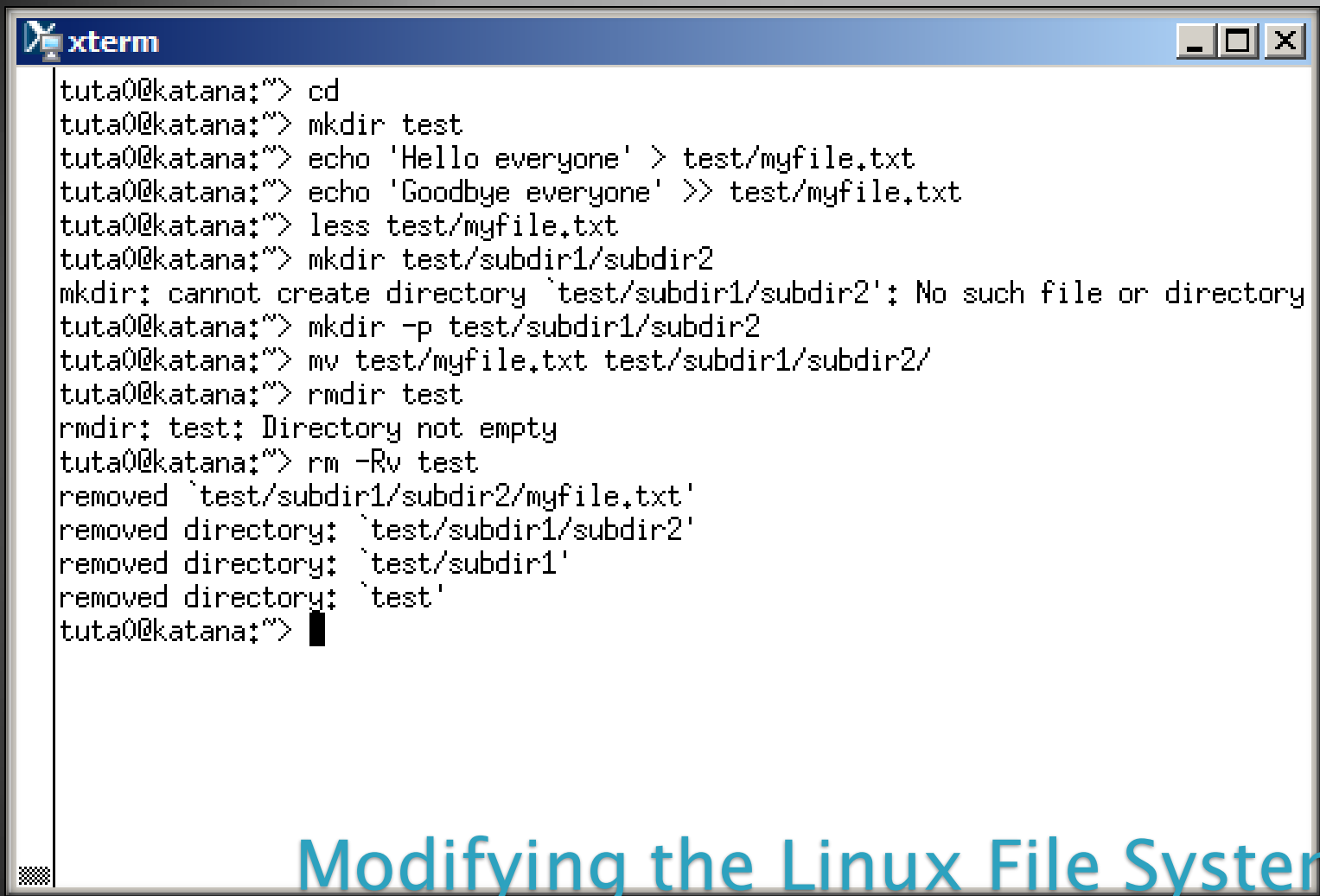
```
xterm
tuta0@katana:/lib> cd /usr/local/lib
tuta0@katana:/usr/local/lib> echo $HOME
/usr4/tutorial/tuta0
tuta0@katana:/usr/local/lib> cd `echo $HOME`
tuta0@katana:~> pwd
/usr4/tutorial/tuta0
tuta0@katana:~> cd ..
tuta0@katana:/usr4/tutorial> cd ..
tuta0@katana:/usr4> cd /lib
tuta0@katana:/lib> ls -d */
bdevd/    firmware/ kbd/    modules/ security/
dbus-1.0/ i686/     lsb/    rtkaio/  udev/
tuta0@katana:/lib> █
```

Navigating the File System >>

Moving around the file system using the cd command

Modifying the Linux File System

- ▶ More useful commands
 - `cd` (also takes you to your home directory like `cd ~`)
 - `mkdir test`
 - `echo 'Hello everyone' > test/myfile.txt`
 - `echo 'Goodbye all' >> test/myfile.txt`
 - `less test/myfile.txt`
 - `mkdir test/subdir1 /subdir2` (FAILS)
 - `mkdir -p test/subdir1 /subdir2` (Succeeds)
 - `mv test/myfile.txt test/subdir1 /subdir2`
 - `rmdir test` (FAILS)
 - `rm -Rv test` (Succeeds)

A terminal window titled 'xterm' with standard window controls (minimize, maximize, close) in the top right corner. The terminal displays a series of commands and their outputs. The user 'tuta00' is at the 'katana' prompt. The commands executed are: 'cd', 'mkdir test', 'echo 'Hello everyone' > test/myfile.txt', 'echo 'Goodbye everyone' >> test/myfile.txt', 'less test/myfile.txt', 'mkdir test/subdir1/subdir2' (which fails with an error), 'mkdir -p test/subdir1/subdir2' (which succeeds), 'mv test/myfile.txt test/subdir1/subdir2/', 'rmdir test' (which fails with an error), and 'rm -Rv test' (which succeeds, showing the removal of the file and directories). The prompt returns to 'tuta00@katana:' after the final command.

```
tuta00@katana:~$ cd
tuta00@katana:~$ mkdir test
tuta00@katana:~$ echo 'Hello everyone' > test/myfile.txt
tuta00@katana:~$ echo 'Goodbye everyone' >> test/myfile.txt
tuta00@katana:~$ less test/myfile.txt
tuta00@katana:~$ mkdir test/subdir1/subdir2
mkdir: cannot create directory `test/subdir1/subdir2': No such file or directory
tuta00@katana:~$ mkdir -p test/subdir1/subdir2
tuta00@katana:~$ mv test/myfile.txt test/subdir1/subdir2/
tuta00@katana:~$ rmdir test
rmdir: test: Directory not empty
tuta00@katana:~$ rm -Rv test
removed `test/subdir1/subdir2/myfile.txt'
removed directory: `test/subdir1/subdir2'
removed directory: `test/subdir1'
removed directory: `test'
tuta00@katana:~$
```

Modifying the Linux File System >>

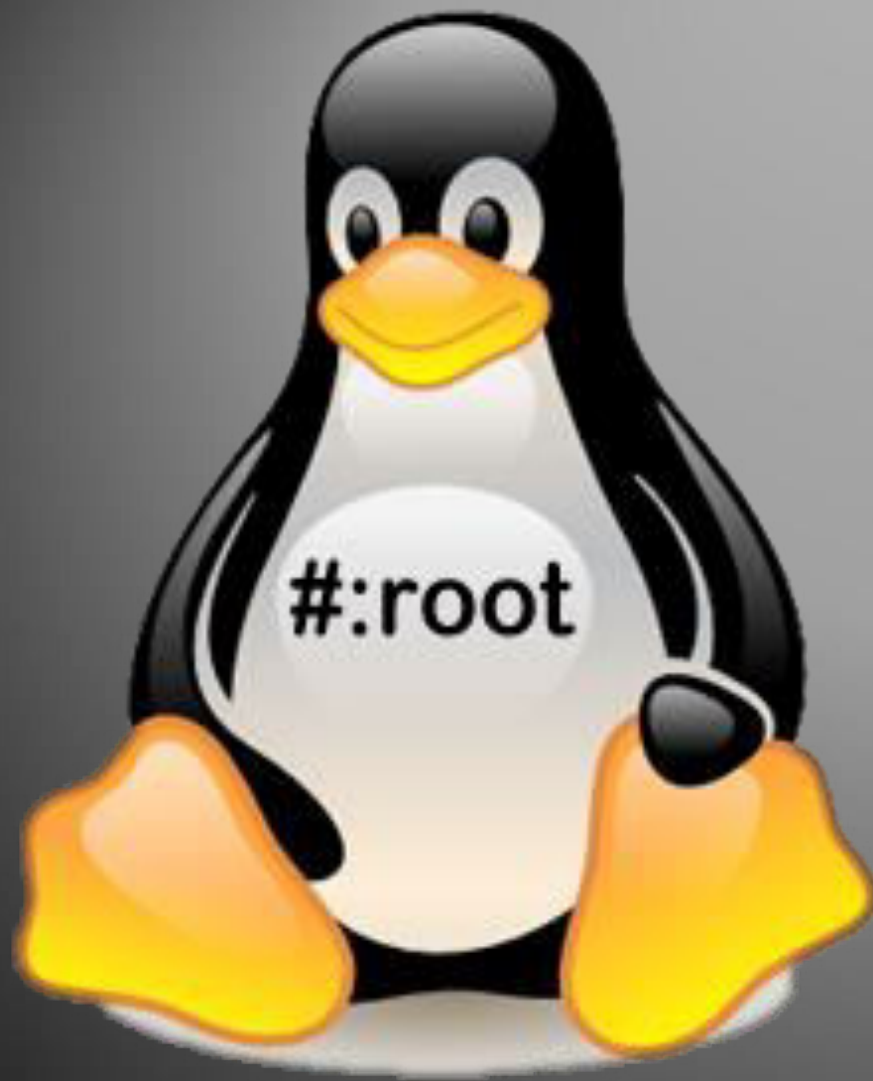
Demonstration of the mkdir, less, mv, rmdir and rm commands

The List Command

- ▶ Useful options for the “ls” command:
 - **ls -a** List all file including hidden file beginning with a period “.”
 - **ls -ld *** List details about a directory and not its contents
 - **ls -F** Put an indicator character at the end of each name
 - **ls -l** Simple long listing
 - **ls -lh** Give human readable file sizes
 - **ls -lS** Sort files by file size
 - **ls -lt** Sort files by modification time

File System Ownership and Permissions

- ▶ All files and directories have a individual and a group *ownership*.
- ▶ All files and directories have read (r), write (w), and execute (x) *permissions* assigned as octets to the individual owner (u), the group (g) owner and all others (o) that are logged into the system.
- ▶ You can change permissions if you are the individual owner or a member of the group.
- ▶ Only root can change ownership.



root >>

The root user is the master

File and Directory Ownership and Permissions

- ▶ Try
 - **cd**
 - **touch myfile** (create file)
 - **mkdir mydir** (create directory)
 - **ls -l myfile** (examine file)
 - **ls -ld mydir** (examine directory)
 - **chmod g+w myfile** (add group write permission)
 - **ls -l myfile**
 - **chmod ugo+x myfile** (add user, group and other execute permission)
 - **ls -l myfile**
 - **chmod ugo+w mydir** (add user, group and other write permission)
 - **ls -ld mydir**
 - **chmod a-w** (a=ALL, remove user, group and other write permission)

```
xterm
tuta0@katana:~$ cd /usr/local
tuta0@katana:/usr/local$ cd
tuta0@katana:~$ touch myfile
tuta0@katana:~$ mkdir mydir
tuta0@katana:~$ ls -l myfile
-rw-r--r-- 1 tuta0 tutorial 0 Sep  9 13:39 myfile
tuta0@katana:~$ ls -ld mydir
drwxr-xr-x 2 tuta0 tutorial 4096 Sep  9 13:39 mydir
tuta0@katana:~$ chmod g+w myfile
tuta0@katana:~$ ls -l myfile
-rw-rw-r-- 1 tuta0 tutorial 0 Sep  9 13:39 myfile
tuta0@katana:~$ chmod ugo+x myfile
tuta0@katana:~$ ls -l myfile
-rwxrwxr-x 1 tuta0 tutorial 0 Sep  9 13:39 myfile
tuta0@katana:~$ chmod ugo+w mydir
tuta0@katana:~$ ls -ld mydir
drwxrwxrwx 2 tuta0 tutorial 4096 Sep  9 13:39 mydir
tuta0@katana:~$ chmod a-w mydir
tuta0@katana:~$ ls -ld mydir
dr-xr-xr-x 2 tuta0 tutorial 4096 Sep  9 13:39 mydir
tuta0@katana:~$
```

File and Directory Ownership and Permissions >>

Examining and changing file and directory permissions



Editing Output Lines With *awk* (see [awk-tutorial.pdf](#))

- ▶ Syntax:

```
BEGIN { Actions }
```

```
{ ACTION } # Action for every line in a file
```

```
END { Actions }
```

- ▶ Try

- `ls -l /usr`

- `ls -l /usr | awk '{print $9 "\t" $5}'`

- `ls -l /usr > usr.txt`

- `awk 'print $9 "\t" $5' usr.txt` (gives same results as 2nd command line, but `awk` is acting on a file instead of saved output)

- `ls -lh /lib | awk '{printf "%20s\t%s\n", $9, $5}'`

- `ls -l /lib | awk 'BEGIN {sum=0} {printf "%20s\t%s\n", $9, $5; sum+=$5} END {sum/=1000000; printf "\nTotal: %d GB\n", sum}'`

```

donj@crsos:~
tuta0@katana:~> ls -l /usr | tail -3
drwxr-xr-x 245 root root 12288 Aug  4 17:09 share
drwxr-xr-x  5 root root  4096 Aug  4 17:08 src
lrwxrwxrwx  1 root root   10 Dec  3 2007 tmp -> ../var/tmp
tuta0@katana:~> ls -l /usr | awk '{print $9 "\t" $5}' | tail -3
share 12288
src 4096
tmp 10
tuta0@katana:~> ls -l /lib | awk '{printf "%20s\t%s\n", $9, $5}' | tail -3
          rtkaio 4096
          security 4096
          udev 4096
tuta0@katana:~> ls -l /lib | awk 'BEGIN {sum=0} {printf "%20s\t%s\n", $9, $5; sum+=
=$5} END{sum/=1000000; printf "\nTotal: %d GB\n", sum}' | tail -7
          lsb 4096
          modules 4096
          rtkaio 4096
          security 4096
          udev 4096

Total: 17 GB
tuta0@katana:~> █

```

Editing Output Lines With awk >>

Output from awk commands

Editing Output Lines With *sed*

- ▶ **sed** replaces one substring with another
- ▶ **sed** operates on every line in a file or processes every line piped into it
- ▶ **sed** matches patterns using *regular expressions* (See regular-expressions.pdf cheat sheet)
- ▶ Common regular expression metacharacters:
 - . – any character
 - ? – quantified zero or one
 - * – quantifier none or more
 - + – quantifier one or more
 - ^ – beginning of line
 - \$ – end of line
 - [XxYy] – character class matching upper or lower case “X” or “Y”

Editing Output Lines With *sed* – continued (see sed-tutorial.pdf)

► Try

- `echo "The rain in Spain stays mainly in the plain." > easy_sed.txt; cat easy_sed.txt`
- `sed -i.bak 's/rain/snow/;s/Spain/Sweden/;s/plain/mountains/' easy_sed.txt; cat easy_sed.txt`
- `ls -l /lib | awk 'BEGIN{sum=0}{printf "%s\t%s\n", $9, $5; sum += $5} END{printf "\nTotal: %d\n", sum}' | sed -e 's/\.so\(.\[0-9\]*\)*/' | less`
(challenge: get rid of soname extension)
- `ls -l /lib | awk 'BEGIN{sum=0}{printf "%s\t%s\n", $9, $5; sum += $5} END{printf "\nTotal: %d GB\n", sum}' | sed -e 's/\.so\(.\[0-9\]*\)*/' | awk '{printf "%20s\t%s\n", $1, $2}' | less` (pretty print)

```
donj@crsos:~  
tuta0@katana:~$ echo "The rain in Spain stays mainly in the plain." > easy_sed.txt  
tuta0@katana:~$ cat easy_sed.txt  
The rain in Spain stays mainly in the plain.  
tuta0@katana:~$ sed -i.bak 's/rain/snow/;s/Spain/Sweden/;s/plain/mountains/' easy_sed.txt  
tuta0@katana:~$ cat easy_sed.txt  
The snow in Sweden stays mainly in the mountains.  
tuta0@katana:~$ ls -l /lib | awk 'BEGIN {sum=0} {printf "%s\t%s\n", $9, $5; sum+=$5} END{printf "\nTotal: %d\n", sum}' | sed -e 's/\.so\(\.[0-9]*\)*/' | tail -7  
lsb          4096  
modules      4096  
rtkaio       4096  
security     4096  
udev         4096  
  
Total: 17279594  
tuta0@katana:~$ ls -l /lib | awk 'BEGIN {sum=0} {printf "%s\t%s\n", $9, $5; sum+=$5} END{printf "\nTotal: %d GB\n", sum}' | sed -e 's/\.so\(\.[0-9]*\)*/' | awk '{printf "%20s\t%s\n", $1, $2}' | tail -7  
lsb          4096  
modules      4096  
rtkaio       4096  
security     4096  
udev         4096  
  
Total: 17279594  
tuta0@katana:~$
```

Editing Output Lines With sed >>

Output from sed commands



Editing Files with Emacs and Vim >>

You don't have to take sides and there is always "nedit"

Editing Files with Emacs and Vim

- ▶ Cheat sheet: [emacs.pdf](#)
- ▶ Movement: <C-b>, <C-n>, <C-p>, <C-f>, <M-b>, <M-e>, <C-a>, <C-e>, <M-'>, <'>, <M-'>', >, <M-'>'>, >, <M-'>'>, >
- ▶ Change/Delete/Replace: <C-d>, <M-d-esc>, <M-d>, <C-kk>, <C-d'char'>, <Insert>
- ▶ Copy/Paste: <C-space>, <C-y>, <C-_>, <M-w>, <C-aky>
- ▶ Search/Replace: <C-s>, <C-s>, <C-r>, <M-x, 'replace-string'<CR>'srchstr'<CR>'replacement'<CR>
- ▶ Save/Quit: <C-xs>, <C-xw>, <C-xc, 'n', 'yes'<CR>>

- ▶ Cheat sheet: [vim.pdf](#)
- ▶ Movement: <h>, <j>, <k>, <l>, , <e>, <0>, <\$>, <gg>, <G>
- ▶ Change/Delete/Replace: <x>, <cw>, <dw>, <dd>, <r>, <R>
- ▶ Copy/Paste: <v>, <P>, <u>, <y>, <yy>
- ▶ Search/Replace: </>, <n>, <N>, <:%s/'regex'/'replacement'/g>
- ▶ Save/Quit: <:q>, <:w>, <:q!>

Emacs – Control Keys
C=Ctrl and M=Meta (Alt)



Vim – Modal
Cmd, Insert, and Visual





Mission Possible: Editing Files with Emacs and Vim

- Someone has corrupted Edgar Allen Poe's poem, "The Raven." Your mission, should you decide to accept it, is to repair the damage with **emacs** or **vim** and then confirm with the "**diff**" command. Hint: Also use **diff** to find corruption.
- `emacs -nw bad-the-raven.txt`
or
- `vim bad-the-raven.txt`
- After editing and saving your file, confirm you work with:
 - `diff bad-the-raven.txt good-the-raven.txt`



Finis

