

XSEDE Scholars Program

Introduction to the Linux Environment

Tips and Tricks for Linux Users

John Lockman III

June 5th, 2012

Tools for this Course

- You will need a Linux terminal to do the examples as we go along.
- Using Linux already?
 - Open the terminal
- Using Mac OSX?
 - Use spotlight search for “Terminal”
- Using Windows?
 - Download a copy of Cygwin
 - www.cygwin.com
 - Install all options.

Linux in the Real World

- Q: Before we begin, does anyone have a feel for how many machines in the November 2011 Top500 list ran variants of the Linux operating System?
- Q: How about Windows?

Linux in the Real World

A: 97.4% are Linux-like

Operating System	# of Systems	Percentage
Linux	457	91.4%
Unix	30	6%
Mixed	11	2.2%
BSD Based	1	0.20%
Windows	1	0.20%

Unix

A Little History

- Q: How old is Unix (5, 10, 20 years, or greater)?



Unix

A Little History

- Q: How old is Unix (5, 10, 20 years, or greater)?
A: > 40 Years
- Unix dates back to 1969 with a group at Bell Laboratories
- The original Unix operating system was written in assembler
- First 1972 Unix installations had 3 users and a 500KB disk



DEC PDP-11, 1972

Linux

Bringing Unix to the Desktop

- Unix was very expensive
- Microsoft DOS was the mainstream OS
- MINIX, tried but was not a full port
- An open source solution was needed!

Linux 0.02 – October 5, 1991

- “Do you pine for the nice days of minix-1.1, when men were men and wrote their own device drivers?
Are you without a nice project and just dying to cut your teeth on a OS you can try to modify for your needs? Are you finding it frustrating when everything works on minix? No more all-nighters to get a nifty program working? Then this post might be just for you :-)” - Linus Benedict Torvalds
- "I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design :-)" (Andrew Tanenbaum to Linus Torvalds)

1990's Movers and Shakers

**Richard Stallman,
father of the GNU Project**



Linus Torvalds



What is Linux?

- Linux is a clone of the Unix operating system written from scratch by Linus Torvalds with assistance from developers around the globe (technically speaking, Linux is not Unix)
- Torvalds uploaded the first version - 0.01 in September 1991
- Only about 2% of the current Linux kernel is written by Torvalds himself but he remains the ultimate authority on what new code is incorporated into the Linux kernel.
- Developed under the [GNU General Public License](#), the source code for Linux is freely available
- A large number of Linux-based distributions exist (for free or purchase)

Why use LINUX?

- **Performance:** as we've seen, supercomputers generally run Linux; rich-multi user environment
- **Functionality:** a number of community driven scientific applications and libraries are developed under Linux (molecular dynamics, linear algebra, fast-fourier transforms, etc).
- **Flexibility/Portability:** Linux lets you build your own applications and there is a wide array of support tools (compilers, scientific libraries, debuggers, network monitoring, etc.)

Why Linux is Still Used

- 40+ years of development (Unix)
 - Linux 1991
- Many academic, scientific, and system tools
- Open Source
- System Stability
- Lightweight
- Easy Development

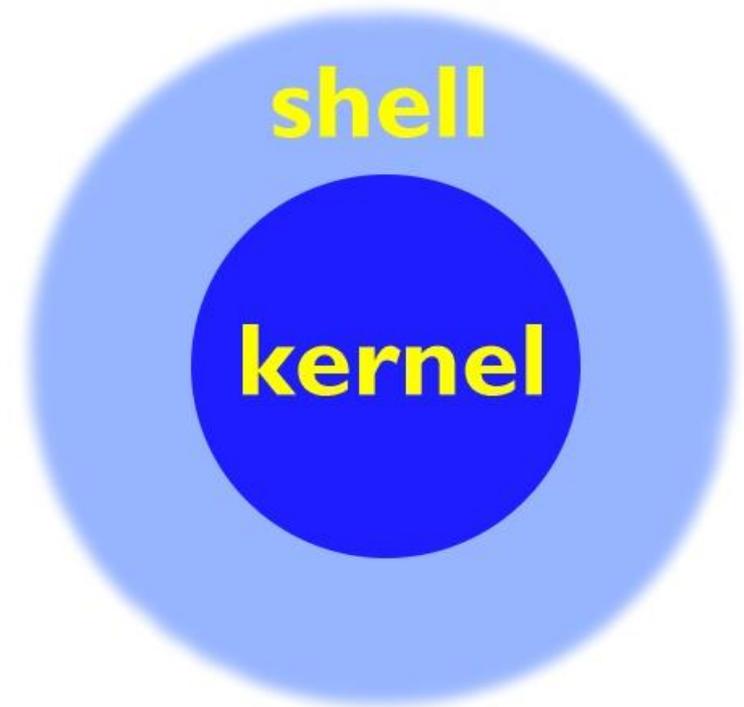
The Basics

- **The Command Line**
 - Interaction with Linux is based on entering commands to a text terminal
 - Often there are no ‘warnings’ with commands, no ‘undo’
- **The Shell**
 - The user environment that enables interaction with the kernel, or lower-system OS.
 - Windows Explorer would be a shell for Microsoft Windows.

The Basics

How does Linux work?

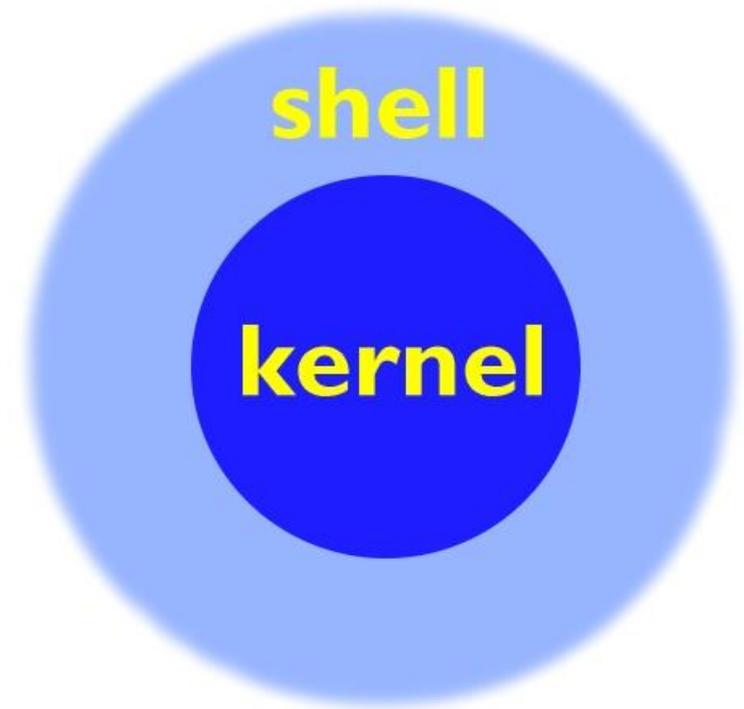
- Linux has a kernel and one or more shells
- The kernel is the core of the OS; it receives tasks from the shell and performs them
- The shell is the interface with which the user interacts



The Basics

How does Linux work?

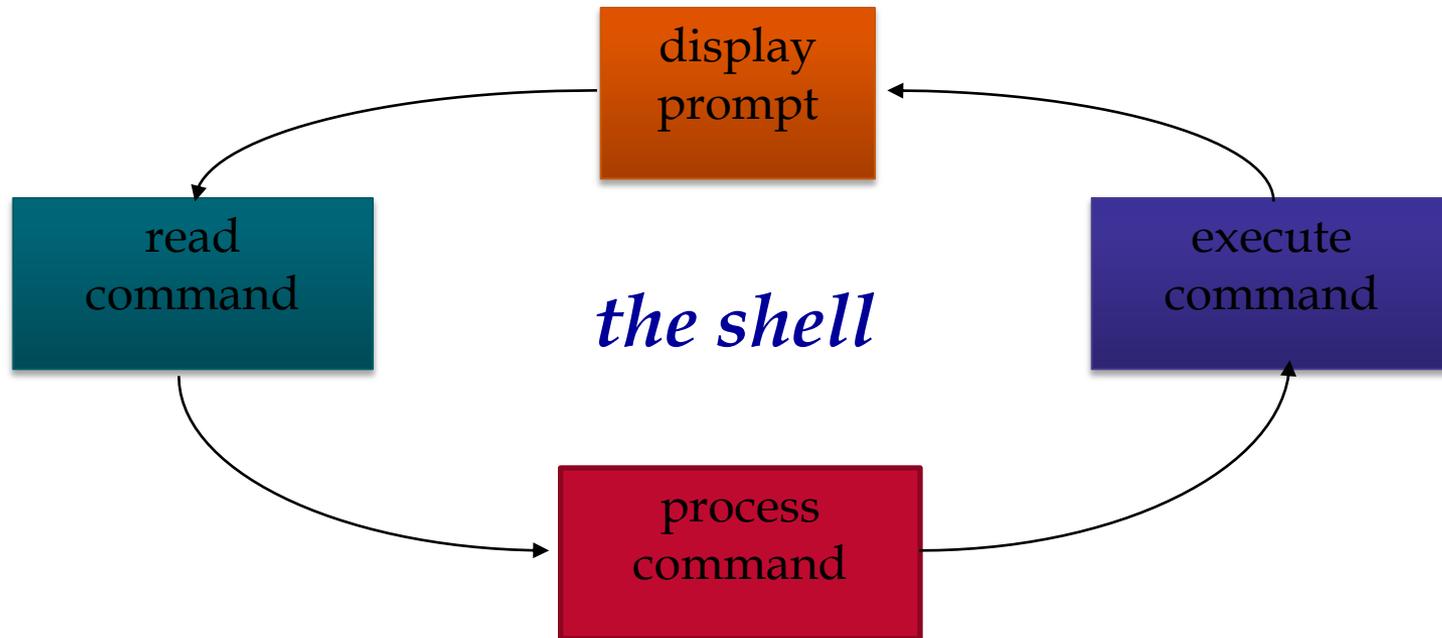
- Everything in Linux is either a file or a process
- A process is an executing program identified by a unique PID (process identifier). Processes may be short in duration or run indefinitely
- A file is a collection of data. Files are created by users using text editors, running compilers, etc
- The Linux kernel is responsible for organizing processes and interacting with files: it allocates time and memory to each processes and handles the filesystem and communications in response to system calls



The Basics

What does the Shell Do?

- The user interface is called the *shell*.
- The shell tends to do 4 jobs repeatedly:



The Basics

Common Shells

- **sh** – the original Unix shell, still located in `/bin/sh`
- **bash** – a Linux shell written for the GNU Project and is installed on most Linux systems
- **csh** – C Shell, modeled after the C programming language used by Linux systems
- **tcsh** – C Shell with modern improvements such as file name completion

- `echo $SHELL` or `echo $0` – displays what shell your account is using
- `chsh` – change your shell

The Basics

Linux Interaction

- The user interacts with Linux via a shell
- The shell can be graphical (X-Windows) or text-based (command-line) shells like tcsh and bash
- To remotely access a shell session on TACC production resources, use ssh (secure shell)

How to Get Help

Before we go further...

- Read the Manual.
 - `man command`
 - `man [section] command`
 - `man -k keyword` (search all manuals based on keyword)
- Most commands have a built-in manual, even the **man** command!
- Commands without manuals have help too, with **-h**, **--help**, or **/?** option.

Linux Accounts

- To access a Linux system you need to have an *account*
- Linux account includes:
 - username and password
 - userid and groupid
 - home directory
 - a place to keep all your snazzy files
 - may be quota'd, meaning that the system imposes a limit on how much data you can have
 - a default shell preference

Shell “Preferences”

- Shells execute startup scripts when you login
- You can customize these scripts with new *environment variables* and *aliases*
 - For bash: `~/.profile`
 - For tcsh: `~/.cshrc`

Customizing Your Startup Script

bash

```
export ENVAR=value  
export PATH=$PATH:/new/path  
alias ll='ls -lrt'
```

- Customize your command prompt

```
export PS1="\u@\h:\W\$ "
```

tcsh

```
setenv ENVAR value  
set PATH = ( $PATH /new/path)  
alias ll "ls -lrt"
```

```
setenv PROMPT "[%n@%m:%c]%"
```

Linux Accounts

Groups

- Linux includes the notion of a "group" of users
- A Linux group can share files and active processes
- Each account is assigned a "primary" group
- The *groupid* is a number that corresponds to this primary group
- In Linux-speak, groupid's are known as *GID's*
- A single account can belong to many groups (but has only one primary group)

Files and File Names

- A file is a basic unit of storage (usually storage on a disk)
- Every file has a name
- File names can contain any characters (although some make it difficult to access the file)
- Unix file names can be long!
 - how long depends on your specific flavor of Linux

File Contents

- Each file can hold some raw data
- Linux does not impose any structure on files
 - files can hold any sequence of bytes
 - it is up to the application or user to interpret the files correctly
- Many programs interpret the contents of a file as having some special structure
 - text file, sequence of integers, database records, etc.
 - in scientific computing, we often use binary files for efficiency in storage and data access
 - Fortran unformatted files
 - Scientific data formats like NetCDF or HDF have specific formats and provide APIs for reading and writing

More about File Names

- Every file must have a name
- Each file in the same directory must have a unique name
- Files that are in different directories can have the same name
- Note: Linux is case-sensitive
 - So, “texas-fight” is different than “Texas-Fight”
 - **Mac caveat: MacOS is NOT cAsE sEnSiTiVe**

Directories

- A *directory* is a special kind of file - Linux uses a directory to hold information about other files
- We often think of a directory as a container that holds other files (or directories)
- Mac and Windows users can relate a *directory* to the same idea as a *folder*

Directories

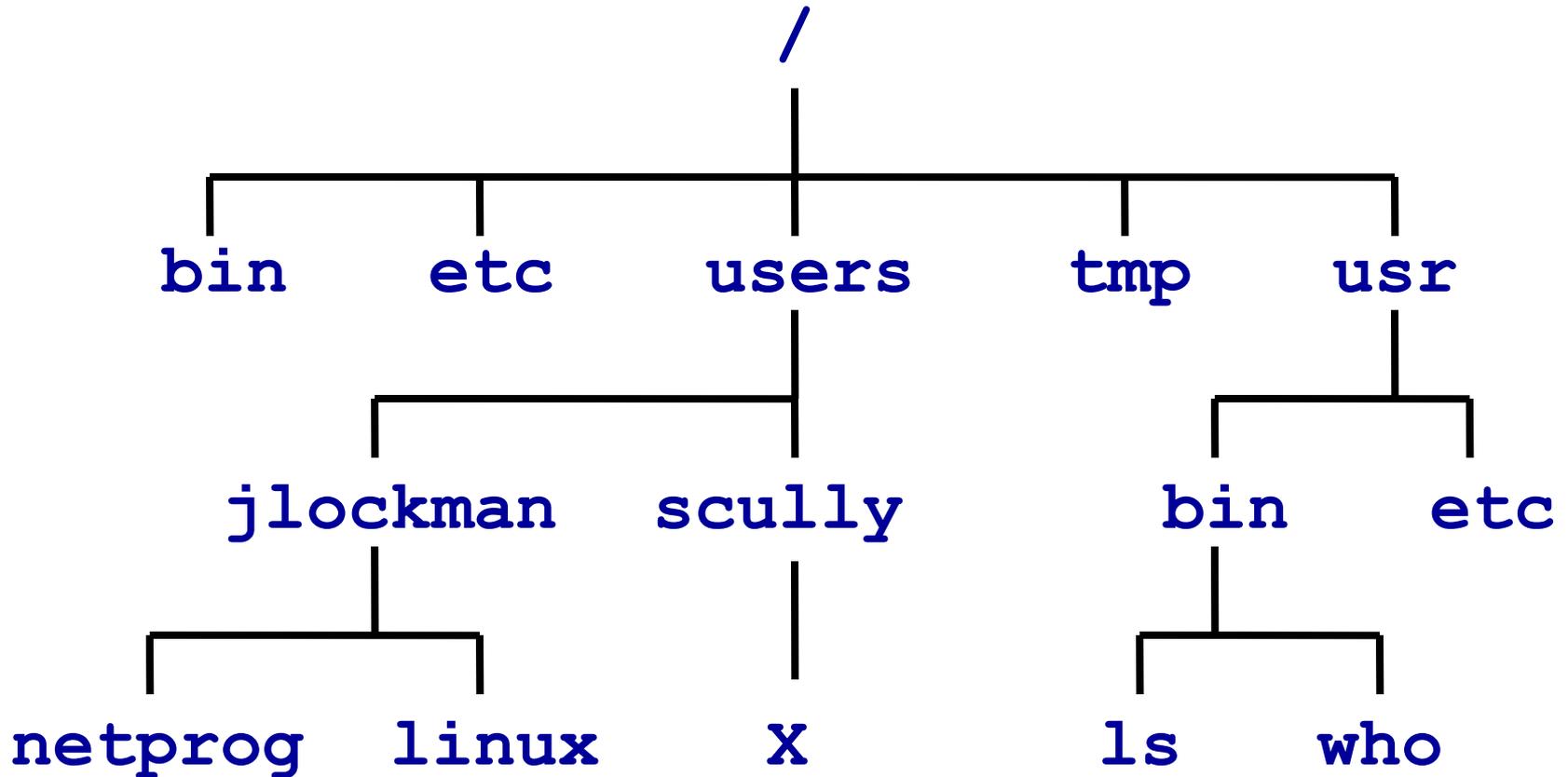
What is a *working directory*?

The directory your shell is currently associated with. At anytime in the system your login is associated with a directory

pwd – view the path of your working directory

ls – view your working directory

Linux File System (an upside-down tree)



Finding your home

Each user has a home directory which can be found with:

```
cd
```

```
cd ~jlockman
```

```
cd $HOME
```

The tilde character ‘~’ will tell the shell to auto-complete the path statement for the **cd** command

`$HOME` refers to an *environment variable* which contains the path for home.

Relative vs.. Absolute Path

Commands expect you to give them a path to a file. Most commands will let you provide a file with a relative path, or a path relative to your working directory.

`../directory` - the `..` refers to looking at our previous directory first
`./executable` - `.` says this directory, or our working directory

Absolute, or Full paths are complete. An easy way to know if a path is absolute is does it contain the `/` character at the beginning?

`/home/user/directory/executable` - a full path to file executable

More file commands

cd *directory* - change your current working directory to the new path

ls *-a* – show hidden files

Hidden files are files that begin with a period in the filename ‘.’

mv - moves one file to another

cp – copies files or directories

rm – remove files & directories

rm *-rf* – remove everything with no warnings

rm *-rf ** - most dangerous command you can run!

rename from to filenames – can rename lots of files at once

- rename file file0 file?.txt (i.e. would move file1.txt to file01.txt)

Recursive Directories

Oftentimes a manual will refer to ‘recursive’ actions on directories. This means to perform an action on the given directory and recursively to all subdirectories.

cp *-R source destination* – copy recursively all directories under source to destination

Poking around in \$HOME

How much space do I have?

quota – command to see all quotas for your directories are, if any.

How much space am I taking up?

du - command to find out how much space a folder or directory uses.

df – display space information for the entire system

Helpful Hints on Space

Almost all commands that deal with file space will display information in Kilobytes, or Bytes. Nobody finds this useful.

Many commands will support a ‘-h’ option for “Human Readable” formatting.

ls -lh - displays the working directory files with a long listing format, using “human readable” notation for space

Permissions

- Linux systems are multi-user environments where many users run programs and share data. Files and directories have three levels of permissions: World, Group, and User.
- The types of permissions a file can contain are:

Read Permissions	Write Permissions	Execute Permissions
r	w	x

Permissions Cont.

- File permissions are arranged in three groups of three characters.
- In this example the owner can read & write a file, while others have read access

User (owner)	Group	Others (everyone else)
rw-	r--	r--

Changing Permissions

- **chmod** – change permissions on a file or directory
- **chgrp** and **chown** – change group ownership to another group (only the superuser can change the owner)
 - Both options support ‘-R’ for recursion.

What everyone else is up to

- **top** – show a detailed, refreshed, description of running processes on a system.
- **uptime** – show the system load and how long the system has been up.
- ‘load’ is a number based on utility of the cpu’s of the system. A load of 1 indicates full load for one cpu.

```
login1$ uptime
```

```
13:21:28 up 13 days, 20:12, 23 users, load average: 2.11, 1.63, 0.91
```

Killing Badly Behaving Processes

- Commands or programs on the system are identified by their filename and by a process ID which is a unique identifier.
 - ps – display process information on the system
 - kill pid – terminates the process id
 - ^c (control+c) terminates the running program
 - ^d (control+d) terminates your session.
- Only you and the superuser (root) has permissions to kill processes you own.

Advanced Program Options

- Often we must run a command in the background with the ampersand ‘&’ character
`command -options &`
runs command in background, prompt returns immediately
- Match zero or more characters wildcard ‘*’
`cp * destination`
copy everything to destination
This option can get you into trouble if misused

Background and Foreground

- `^z` (control+z) suspends the active job
- **bg** – resumes a suspended job in the background and returns you to the command prompt
- **fg** – resumes a background job in the foreground so you can interact with it again

Editing and Reading Files

• **emacs** vs. **vim**

- Among the largest ‘nerd battle’ in history. **emacs** relies heavily on key-chords (multiple key strokes), while **vim** is mode based. (editor mode vs. command mode)
- **vim** users tend to enter and exit the editor repeatedly, and use the Linux shell for complex tasks, whereas **emacs** users usually remain within the editor and use **emacs** itself for complex tasks

• **less**

- If you only need to read a file (not edit it), programs like less give you “read only” access and a simplified interface

• Others:

- **nano** and **pico** are very simple text editors

Searching for files

A large majority of activity on Linux systems involve searching for files and information.

find – utility to find files

```
login1$ find . -name foobar
./test_dir/foobar
login1$ cat ./test_dir/foobar
=====
*
This is the file I searched for!
*
=====
```

Input and Output

- Programs and commands can contain an input and output. These are called 'streams'. Linux programming is oftentimes stream based.
 - Programs also have an error output. We will see later how to catch the error output.
- 0 - STDIN** – 'standard input,' or input from the keyboard
- 1 - STDOUT** – 'standard output,' or output to the screen
- 2 - STDERR** – 'standard error,' error output which is sent to the screen.

File Redirection

- Oftentimes we want to save output (stdout) from a program to a file. This can be done with the 'redirection' operator.

```
myprogram > myfile
```

using the '>' operator we redirect the output from myprogram to file myfile

- Similarly, we can append the output to a file instead of rewriting it with a double '>>'

```
myprogram >> myfile
```

using the '>' operator we append the output from myprogram to file myfile

Input Redirection

- Input can also be given to a command from a file instead of typing it to the screen, which would be impractical.

```
cat programinput > mycommand
```

- This command series starts with the command 'cat' which prints a file to the screen. programinput is printed to stdout, which is redirected to a command mycommand

Pipes

- Using a pipe operator ‘|’ commands can be linked together. The pipe will link the standard output from one command to the standard input of another.
- Helpful for using multiple commands together
example: `ls -l */* | wc -l`

Other Useful Commands

head `file.txt`

– prints the first 10 lines of a file

tail `-n 5 file.txt`

– prints the last 5 lines of a file

history

– prints your command history

example: `history | grep "sed"`

Compression using gzip

- `slogin1$ du -h bigfile`
- `32K bigfile`
- `slogin1$ gzip bigfile`
- `slogin1$ du -h bigfile.gz`
- `4.0Kbigfile.gz`

Linux vs. Windows files

- File formats are different between the two operating systems
- Use the UNIX command `dos2unix` to convert files – especially script files - created on Windows, so they will work on Linux

File Transfers

- **scp** is a simple file transfer tool.
- scp usage:
 - scp [options] SOURCE DESTINATION
 - Example:

```
login1$ scp myfile.txt jlockman@lonestar.tacc.utexas.edu:
```

- This will copy the file “myfile.txt” to Lonestar in my home folder (/share/home/00944/jlockman)
- You could also provide the full path

```
login1$ scp myfile.txt jlockman@lonestar.tacc.utexas.edu:/work/00944/jlockman/foo
```

Run Linux on your Desktop/Laptop

- Try out a LiveCD first
 - Just download and boot from CD or USB
 - No reconfiguring of your existing setup
- Fedora
 - <http://fedoraproject.org/en/get-fedora>
- Ubuntu
 - <http://www.ubuntu.com/download/ubuntu/download>