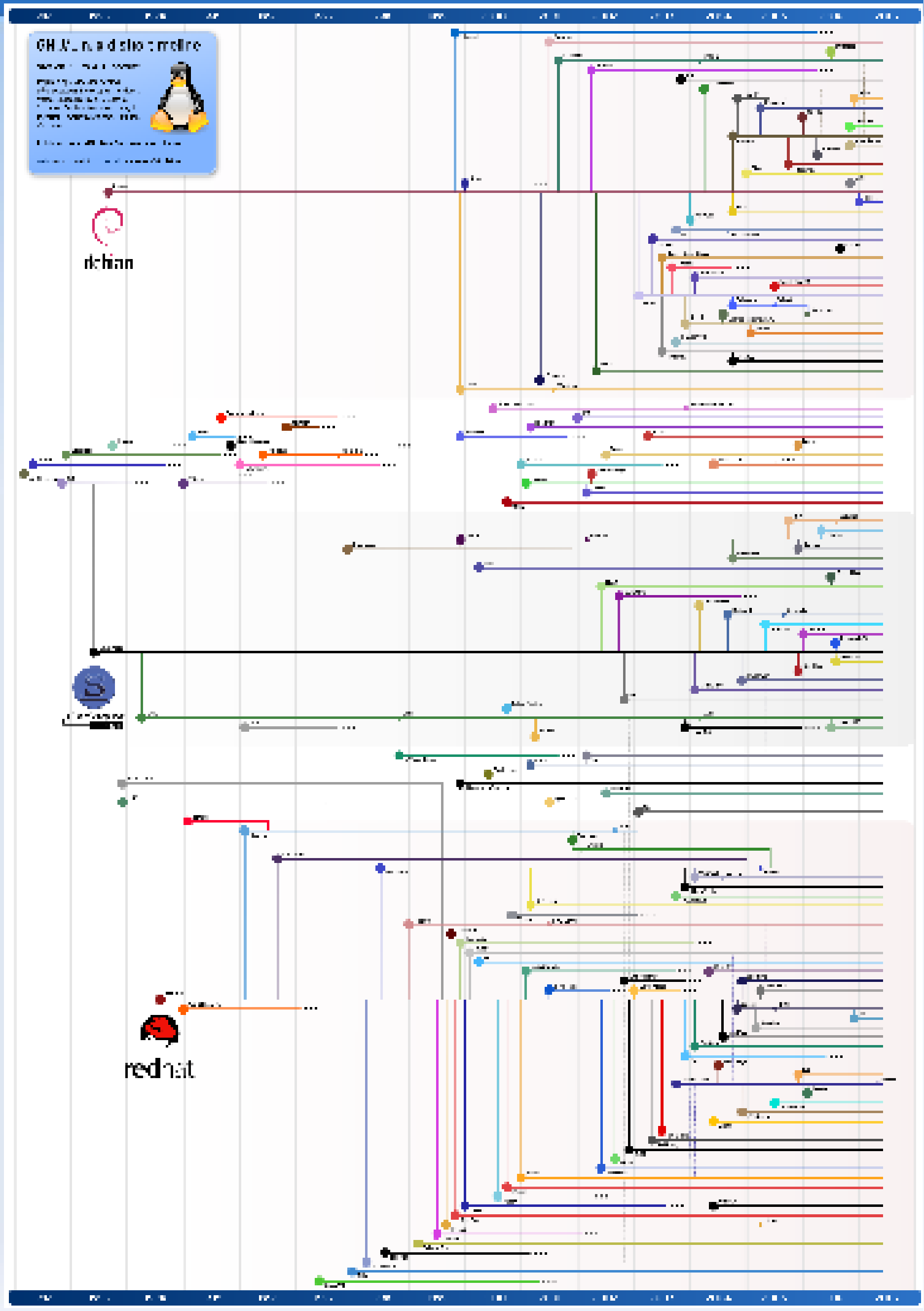
# Linux 101

- A Brief History of Linux
- Basic Concepts and Terminology
- Text Editors
- Finding Documentation
- Basic Shell Usage
- Network Tasks
- Package Management
- Administer Linux from Windows
- Introduction to Intermediate Topics

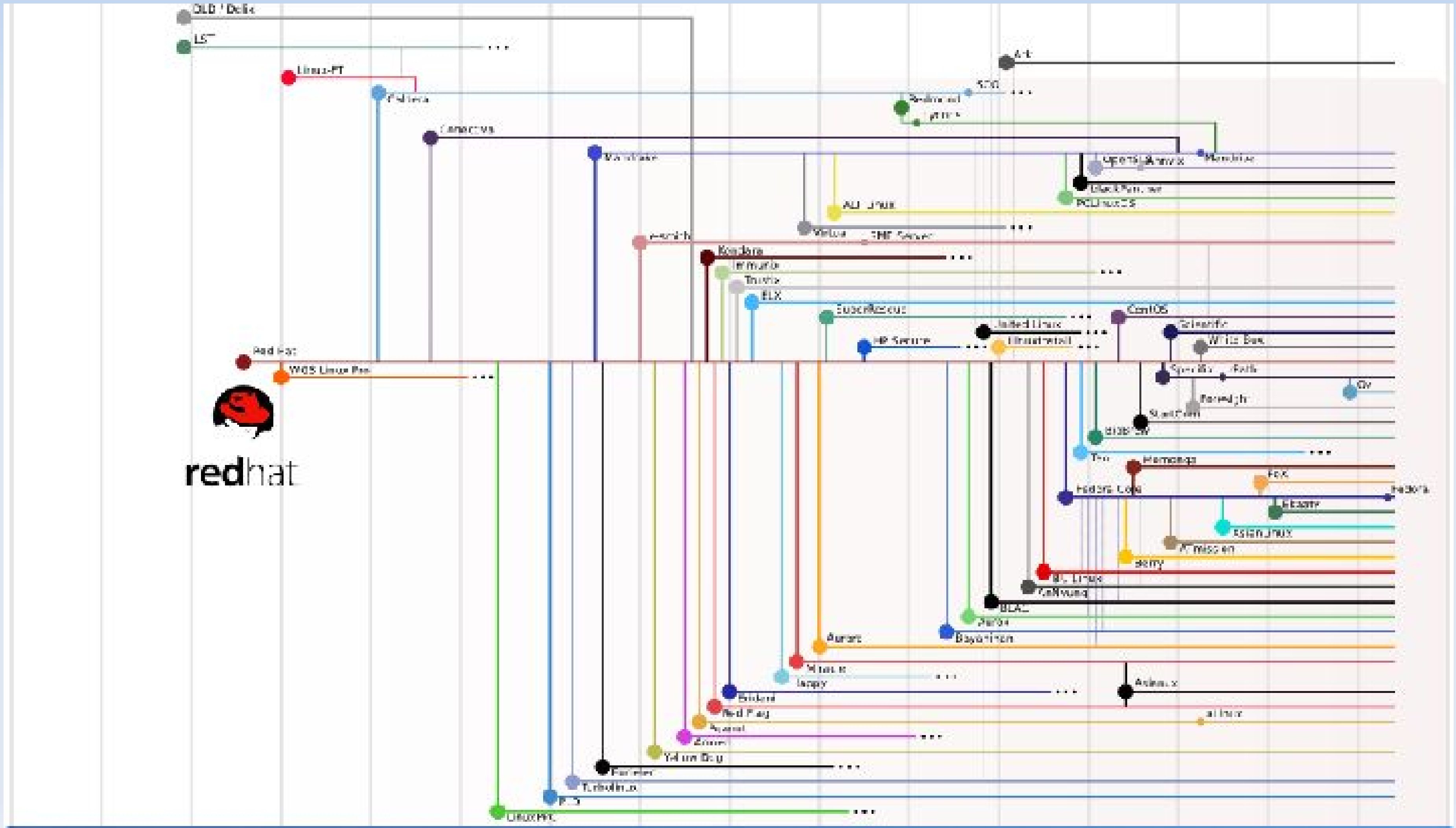
# A Brief History of Linux

- Linux has a HUGE family tree
  `http://futurist.se/gldt/`
- Why is that?
- How do they differ?



# A Brief History of Linux

- RedHat's branch



# Basic Concepts and Terminology

- CaSe senSITiviTy
  - `command` is not the same as `Command` or `COMMAND`
- first commands: `ls` (*l*i*s*t directory contents), `cd` (*c*hange *d*irectory), `pwd` (*p*rint *w*orking *d*irectory)
- Home Directories, `$HOME`, and ~
- Paths, filenames, and pathnames
  - `/home/igo/` vs `file.txt` vs `/home/igo/file.txt`
- Special directories `.` and `..`
- Absolute vs Relative Paths
  - `/home/igo/file.txt` vs `./file.txt` vs `../igo/file.txt`

## Hands On #1

- Open a shell (aka terminal, or `xterm`) on your computer and type `echo $HOME` Are you in `$HOME` now?
- Using full paths, change directories to `/tmp`.
- Using full paths, go back to your home directory.
- Using a relative path, change directories to `/home`.
- Using a relative path, enter your home directory.
- What files and directories can you find in your home directory?

## Basic Concepts and Terminology

- File Permissions, Owner, and Group
  - Who can read, write, or execute a file?
  - Running `ls -l` on a file shows something like this:
    - `-rwxr-xr--  1 igo  igo     25 2007-08-22 10:54 tmp`
  - The file *user* (owner) is shown in the third column. The *group* to which the file belongs is shown in the fourth column. Permissions for the owner, group, and other users are listed in the first column:
    - First triplet (`rwx`) is the permissions of the file's *user* (owner). Second triplet (`r-x`) is the *group*'s permissions. Third triplet (`r--`) is the permissions of any other users.
      - Someone trying to work with the file gets the most-permissive set of permissions for which they qualify.
      - `r` allows the file to be *r*ead; `w` allows the file to be (over)*w*ritten; `x` allows the file to be executed.

## Basic Concepts and Terminology

- Directory Permissions, Owner, and Group
  - Who can see inside, write in, or list the contents of a directory?
  - Running `ls -l` on a directory shows something like this:
    - `drwxr-xr--   1 igo   igo    4096 2007-08-22 10:54 dir`
  - Everything works the same as files, except:
      - `r` allows the directory contents to be seen; `w` allows the directory to accept new files; `x` allows someone to enter the directory.

## Basic Concepts and Terminology

- The Power of root
  - *The root account can override any permissions on local filesystems.*

## Text Editors

- There are many options, but these are the common ones:
  - `nano`, `emacs`, `vi`
  - `nano` and `emacs` provide a UI; `vi` does not
  - `vi` is always present; `nano` usually is; `emacs` often has to be installed
  - `nano` is easiest to learn, followed by `emacs`, and then `vi`
- Is there such a thing as the best editor?
  - Yes. And there are several best editors.

## Finding Documentation

- `man`
  - short for `manual`
  - e.g. `man vi`
- `info`
  - e.g. `info vi`
- `http://google.com`
- built-in
- books
  - paper (relatively cheap, but not updatable)
  - Safari Books Online: `http://safari.oreilly.com/` (expensive, but updated regularly)

## Hands On #2

- Using the free, offline methods listed previously, find documentation that shows you how to do things in `emacs` and `nano`.
- Think about when might you use each source, including books.

## BREAK

- 15-minute break

## Basic Shell Usage

- Common Commands
  - `cp` *`source target`*
    - *copy* a file or directory to another location and/or name.
  - `mv` *`source target`*
    - *move* a file or directory to another location and/or name.
  - `rm` *`target`*
    - *remove* (delete) a file or directory.
    - *Use parameters carefully!*
    - *Be very careful when root.*
    - *No trashcan or recycling bin with* `rm`*!*
    - *Use* `ls` *to test* `rm` *commands.*

## Hands On #3

- Use a text editor to create a file named `~/handson1.txt` that contains the following lines, then exit the editor:

```
user=skippy
password=perklang
home=/home/skippy
realname=Skippy Perklang
```

- When you are done, remove the file and re-create it using a different editor, repeating until you have used all the editors. *Do not delete the file that you created with the final editor.*

## Hands On #4

- Skippy has legally changed his first name to Herbert. Copy `~/handson1.txt` to `~/handson2.txt`, then choose an editor to change any instances of `Skippy` or `skippy` in `~/handson2.txt` to `Herbert` or `herbert`, as appropriate.
- When you are done, re-copy `~/handson1.txt` to `~/handson2.txt` and use a different editor for the task.
- Repeat until you have used all the editors listed earlier.

# Basic Shell Usage

- `mkdir` *`target`*
  - *makes* a *dir*ectory.
- `rmdir` *`target`*
  - *rem*oves an **empty** *dir*ectory. (Use `rm -r` to remove non-empty directories.)
- `ln` *`target source`*
  - Creates a *lin*k to a real file or directory.  Most common usage is `ln -s` *`target source`*

# Hands On #5

- Make a directory called `~/handson3/sub1/sub2/sub3/sub4`.
- While in `~/handson3`, produce a recursive directory listing. (Learn how with `man ls`.)
- Create a link from `~/handson3/link` to `~/handson3/sub1/sub2`. If you `cd` into `link`, do you end up in `sub2`?
- List the contents of `~/handson3` in a way that shows that `link` is a link.
- Remove `sub4`. Then remove `sub2` and all directories below it. Show how this affects `link`.

## Basic Shell Usage

- `chmod`
  - *ch*anges *mod*e (permissions) of a file or directory.
- `chgrp`
  - *ch*anges *gr*ou*p* of a file or directory.
- `su`
  - Logs you in as another user, often root (the *s*uper*u*ser).
- `sudo`
  - Runs commands with temporary root permissions (i.e. *do* things as *s*uper*u*ser).
- `whoami`
  - Displays which user the shell belongs to.

## Hands On #6

- Without logging in as the root user, create a directory called `~/handson3/root-owned` belonging to `root` and in `root`'s default group.
- Make your user account the owner of the directory.
- Change the group to be your user's default group.
  - hint: You can use `ls -l ~` to learn your default group.
- Change the permissions on the directory so that only the members of the group can enter the directory.
- Log in as `root` and see what username you have.
- Without logging in as `root`, run the same command you used above with root permissions.

## Hands On #7

- Log in as the root user and create a directory called `~/handson3/root-owned` belonging to `root` and in `root`'s default group.
- Change the owner to be your user account.
- Change the group to be your user's default group.
  - hint: You can use `ls -l ~` to learn your default group.
- Change the permissions on the directory so that only members of the group can enter the directory.

# Basic Shell Usage

- `which`
  - `which` *`file`* `:` Which executable named *`file`* will be run if you type *`file`* in a shell?
- `updatedb and locate`
  - Lets you build a file location table and search it quickly.
- `grep`
  - Find what you want amongst a bunch of text, e.g. `grep` *`pattern file`* or `ls | grep` *`file`* (*)
- `cat, more, and less`
  - Different ways to look at a file without an editor.
  - Example: `less` *`file`*

(*) That strange vertical line will be explained later.

# Hands On #8

- There is a file called `hands-on-5-instructions.txt` on your laptop.  Once you find it, read it in three different ways to find your next set of instructions.

## Basic Shell Usage

- `last`
  - See recent logins, shutdowns, reboots.
- `history`
  - See all your previous shell commands.
- `who` and `w`
  - Who has a shell on the system? What are they running?
- `ps` / `top`
  - Get a list of running processes.
- `fuser`
  - While pronounced a bit like an insult, it really isn't.
  - It tells you what processes are *users* of a *file*.

## Hands On #9

- What are some of the recent commands you have run, in order?
- Is anyone else logged into your computer?
- Who has been logged into your computer in the past?
- What processes are currently ”using” your home directory?

## LUNCH

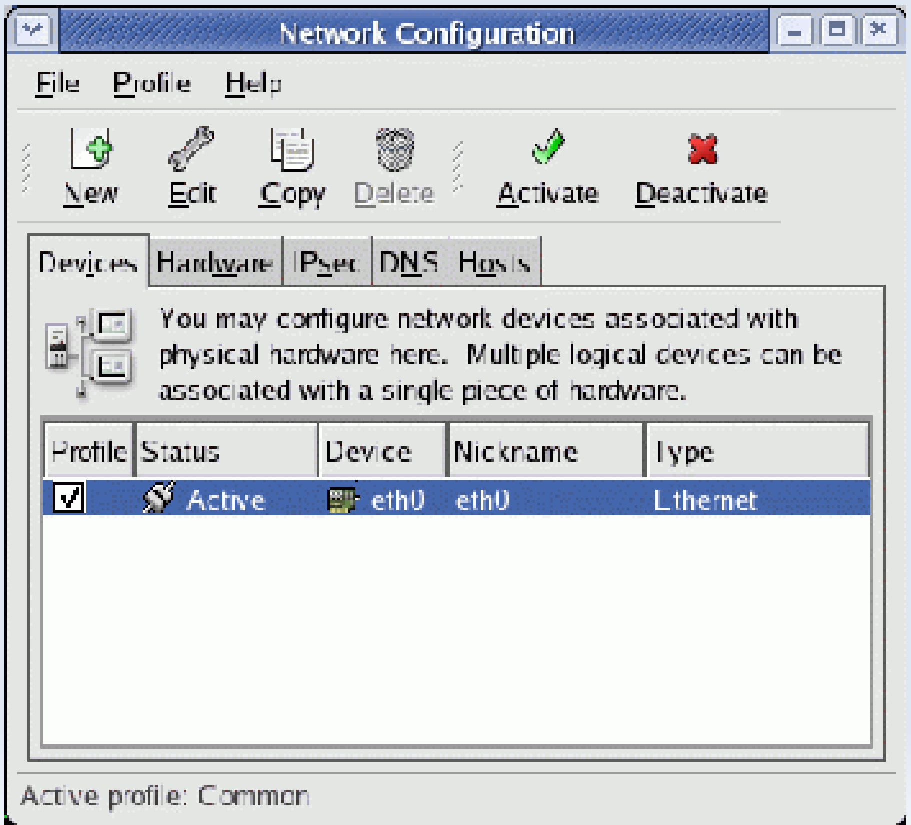
- 60 minute lunch

## Network

- `ifup` / `ifdown` : Turn network devices on or off.
- `ifconfig` : Check or set network parameters.
- `ping` : See if another networked system is up.
- `ssh`
  - Think: supercharged, encrypted `telnet`
  - Remotely administer any Linux system from another.
- `scp`
  - Think: supercharged, encrypted `ftp`
  - We'll get to it later.
- `telnet` : check open ports, e.g. `telnet` *`host port`*

# Network

- GUIs
  - The GUI equivalent of `ifup`/`ifdown` and `ifconfig`: System -> Administration -> Network launches the Network Configuration tool in Fedora. From the CLI, you can run it as `system-config-network-gui`


Network Configuration
File Profile Help
New Edit Copy Delete Activate Deactivate
Devices Hardware IPsec DNS Hosts

You may configure network devices associated with physical hardware here. Multiple logical devices can be associated with a single piece of hardware.

| Profile | Status | Device | Nickname | Type |
|---|---|---|---|---|
| ✓ | Active | eth0 | eth0 | Ethernet |

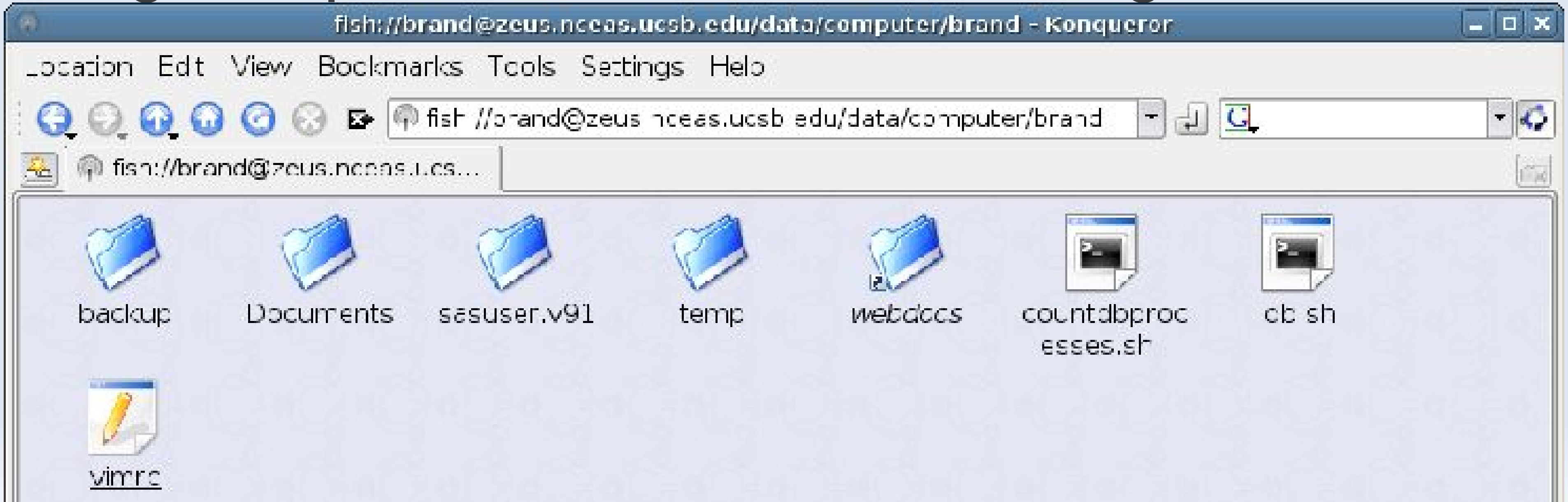
Active profile: Common

## Network

- `ping` and `ssh` don't exactly need GUIs, but there may be some out there.
- `telnet` doesn't need a GUI for the purpose of checking open ports.

# Network

- `scp`'s GUI in Fedora is essentially `konqueror` using the `fish://` protocol, which connects to another UNIX system over SSH. URL Syntax: `fish://`*`user`*`@`*`host`*`/`*`directory`*
- You can copy files between `konqueror` windows or between `konqueror` tabs, just like you would when using konqueror as a local file manager.


fish://brand@zeus.nceas.ucsb.edu/data/computer/brand - Konqueror
Location Edit View Bookmarks Tools Settings Help
fish://brand@zeus.nceas.ucsb.edu/data/computer/brand
fish://brand@zeus.nceas.ucs...
backup
Documents
sasuser.v91
temp
webdocs
countdbprocesses.sh
cb.sh
vimrc

## Hands On #10-1

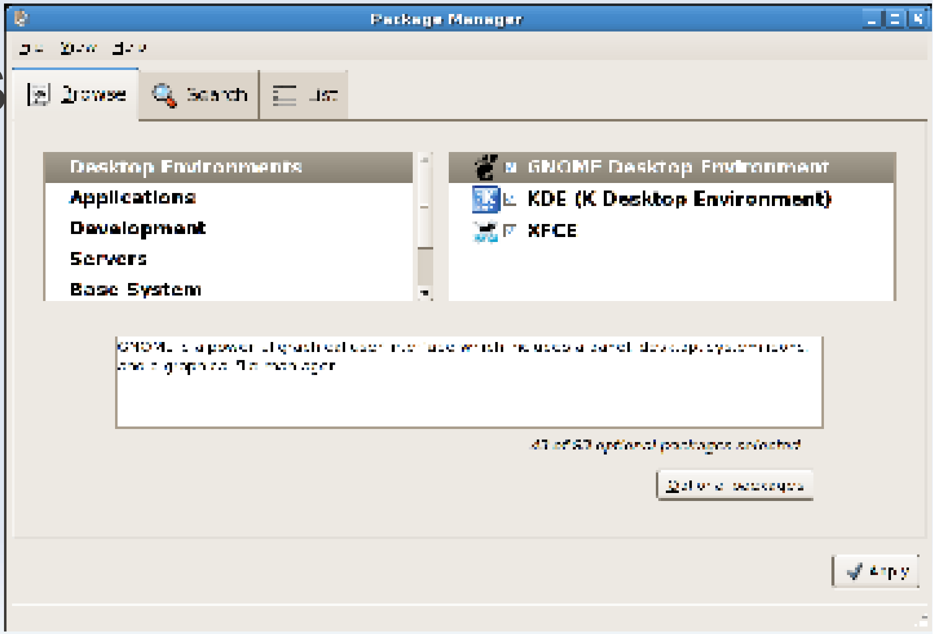
- Provide your IP address to your lab partner.  Each of you will `ping` the other's computer and then log into it securely and remotely using the CLI.
- Did the login proceed smoothly?
- Launch an `xterm` on your lab partner's computer that displays on yours.  How do you know it's not really running on your computer?
- Using software only, *one* of you should disconnect your computer from the network.
- Observe the behavior of both computers after one is no longer on the network.

## Hands On #10-2

- Restore the network connection.  Did you happen to do it in time to save any remote connections, or did they drop?
- Log in again remotely.  Did the login proceed smoothly?

# Package Management

- CLI
  - RPM (*R*edHat *P*ackage *M*anager) : Very low-level
    - *Be careful*.  Many distros use RPMs, but not all RPMs are compatible with Fedora.
  - YUM (*Y*ellowdog *U*pdater, *M*odified) : Wrapper for RPM
    - `http://en.wikipedia.org/wiki/Yellow_dog_Updater%2C_Modified`
- GUI
  - Fedora-based distributions use `pirut`, aka Package Manager.  Essentially a GUI wrapper for YUM.
  - Applications -> Add/Remove Software



## Hands On #11

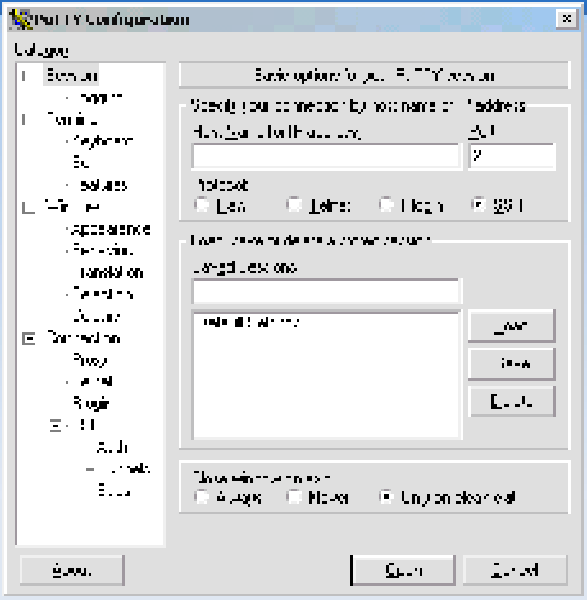
- Using each of the methods described, get a list of installed packages that have the text `xorg` in them.
- Using only the *wrappers* described, *show* how you would remove the `emacs` package, but do not actually attempt it.
- Using only the *wrappers* described, *show* how you would install a new package and its dependencies, but do not actually attempt it.

## BREAK

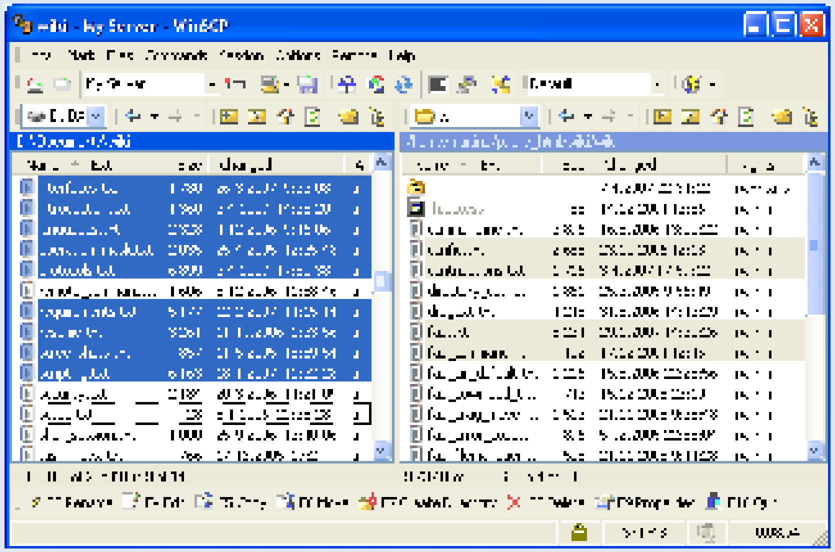
- 15-minute break

# Administer Linux from Windows

- putty
  - ssh for Windows
  - http://www.chiark.greenend.org.uk/~sgtatham/putty/



- WinSCP
  - scp for Windows
  - http://winscp.net



- cygwin
  - Lets you run remote Linux applications and display them in Windows. *It doesn't always work.*

# Introduction to Intermediate Topics

- `$PATH`
  - A colon-separated list of directories where your shell looks for executable files.
  - You may want to add directories to `$PATH` for 3rd-party software that installs in nonstandard locations (e.g. `java`).
- `echo`
  - Actually, a basic command, but often only needed for more involved reasons. `echo` will, among other things, tell you the value of any shell variable, like `$HOME` or `$PATH`.

# Hands On #12

- Open a *new* xterm and see what your $PATH is:
  - `echo $PATH`
- Set your `$PATH` there to nothing:
  - `export PATH=""`
- Try to run `ls`. It's gone away!
- Can you still run `ls` in your old xterm?
- Add an arbitrary path to your empty `$PATH` :
  - `export PATH="/some/dir"`
- Use a command listed above to check its value.
- Close the new xterm that you opened.

# Introduction to Intermediate Topics

- backups
  - `scp`
    - e.g. to do a full copy of localdir to remotehost,
    - `scp -r /my/localdir user@remotehost:/directory/`
  - `rsync`
    - Uses `ssh/scp` by default to *syn*chronize (backup) files between locations, either locally or remotely.
    - e.g. to do a full or incremental copy of localdir to remotehost, deleting any files on remotehost that have been deleted in localdir:
    - `rsync -avz –delete /my/localdir user@remotehost:/directory/`

# Hands On #13

- Use `scp` to copy the contents of `~/.nautilus` to `/tmp` on your lab partner's computer.
- Do it again with `rsync`.
- Was anything different the second time using `rsync`?
- Do the same with `scp`.

## Introduction to Intermediate Topics

- `keychain`
  - Helps automate ssh logins by using pre-shared keys between systems.
  - Convenience vs Security
  - http://www.ibm.com/developerworks/linux/library/l-keyc.html
  - http://www.ibm.com/developerworks/linux/library/l-keyc2/

## Introduction to Intermediate Topics

- To set it up, do the following for each user, on the server:
  - Login as *`user`*
  - `ssh-keygen -t dsa -b 1024 -f ~/.ssh/id_dsa`
  - Enter a passphrase for the key.
  - `cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys`
  - `chmod go-rw ~/.ssh/authorized_keys`

## Introduction to Intermediate Topics

- Copy `~/.ssh/id_dsa*` from the server to the client.
  - e.g. `scp ~/.ssh/id_dsa*` *`username@clientbox`*`:/home/`*`username`*`/.ssh/`
- Next, on the client system, make sure `keychain` is installed (it is on your classroom computer), and make the following additions to the user's `.bashrc`:
  - `keychain $HOME/.ssh/id_dsa`
  - `. $HOME/.keychain/${HOSTNAME}-sh`
- Log out and log back in on the client computer to activate `keychain`.

## Hands On #14

- Set up a `keychain`-based login to your lab partner's computer.
- Verify that it works without a password once you enter your passphrase on your local computer.
- Once verified, repeat the previous backup exercise with `rsync`.  Now you don't have to enter your password for your account on your lab partner's computer.

## BREAK

- 10-minute break

## Introduction to Intermediate Topics

- Advanced `grep` options
  - You can write *regular expressions* to do pattern-based grepping.
  - Learning regular expressions completely can take weeks. A good place to start: `http://www.regular-expressions.info/`
  - Short examples:
    - `grep -e ".ob..go" /etc/passwd`

      ```
      igo:x:1000:1000:Bob Igo,,,:/home/igo:/bin/bash
      ```
    - `grep -e "Igo.*bash" /etc/passwd`

      ```
      igo:x:1000:1000:Bob Igo,,,:/home/igo:/bin/bash
      ```

## Introduction to Intermediate Topics

- Chaining shell commands
  - The | character is referred to as *pipe* in Linux. *Piping* output from one command to another command's input is common. Some examples:
    - `ps aux | grep term | less`
    - `cat ~/handson1.txt | grep home`
  - < and > can be used to redirect input or output to from or to a file, overwriting any previous contents. Some examples:
    - `ps aux | grep term > terms-running.txt`
    - `cat ~/handson1.txt | grep home > homes-found-in-handson1.txt`
  - >> can be used to redirect output to a file, adding to its previous contents. Examples are the same as above, but with >> instead of >.

## Hands On #15

- Get a list of the previous commands you have run in your shell that contain the text:

  `tmp`

  and put them into the file `~/tmps.txt`
- Now repeat the above, but add to `~/tmps.txt` instead of overwrite it.

## Introduction to Intermediate Topics

- shell scripting
  - A shell script is a collection of shell commands.  They range from the trivial to the complicated.
- system startup scripts
  - Shell scripts that regulate how your system starts up. Distros like Fedora store them in `/etc/init.d/`
  - Basically, when your system boots, scripts under `/etc/init.d/` start services that have been configured to run.  For example, all the server processes that you need for K12LTSP are run for you this way.
  - If you want to manually stop, start, or restart one of these services, run `/etc/init.d/`*`scriptname`* with no parameters to see your options.

## Introduction to Intermediate Topics

- find
  - *"I don't know where I saved my OpenOffice document, but I did it yesterday after lunch." -- Joe User*
  - `find` could get its own full-day course. Here's a basic example that you can extend or modify as needed:

```
find /some/directory -name 'pattern' -type f -exec grep greppattern {} \; -print
```

  - This searches in and under `/some/directory` for a regular file (`-type f`) named *`pattern`* (`-name 'pattern'`). If it finds it, it will execute a `grep` for text that matches *`greppattern`* (`-exec grep greppattern {} \;`) and finally print the name of the file (`-print`).

## Introduction to Intermediate Topics

- `ps`
  - See what processes are running.
- `kill` *`signal PID`*
  - Stop (or, ironically, restart) a process with Process ID *`PID`*. Examples:
    - `kill -1` *`PID`* : kill, then start the process with Process ID *`PID`*
    - `kill` *`PID`* : kill the process
    - `kill -9` *`PID`* : kill the process a *lot*
- `mount` and `umount`
  - Mount and unmount filesystems.
  - Either by hand or invoking pre-defined shortcuts in `/etc/fstab`

## Conclusion

- That was a lot of information in a relatively short amount of time.
- Don't try to memorize the details at this stage.  The concepts are more important.
- As you implement your new knowledge, use this presentation as a reference on what can be done.
- Remember: There are lots of ways to do the same thing.

## Questions?