Introduction to Linux

SANOG IV Workshop 2004

July 2004

Presented by Hervey Allen
Network Startup Resource Center

Welcome

- Introduction
  - Instructors:
    Hervey Allen
    Joe Abley
    Philip Hazel
  - Helpers:
    Kishor Panth
- At what level are we?
- How the class runs
- What we'll do today

Schedule

Morning
- 08:45-10:45 Class
- 10:45-11:00 Break
- 11:00-13:00 Class

Lunch
- 13:00-14:00

Afternoon
- 14:00-16:00 Class
- 16:00-16:15 Break/Tea
- 16:15-18:15 Class

Evening sessions

Optional. Around 19:30-21:00.

Labs

Open after dinner (19:30) until around 21:00. If there is class you can use machines not in use by those attending class.

How the workshop runs

With your participation!

Please ask questions. If you don't understand something due to language, ask us to clarify.

We are sharing workstations. This is good. You'll learn more working in groups.

At the end of the week we'll have a written test.

A Certificate of Attendance will be given as well.

How the workshop runs – practical

We have a server in use for the workshop. IP and server name will be given in class.

The PCs have Internet access, but during class we'll often work outside of XWindows.

The root password on your machine is "SANOG2K4".

We have several people assisting during the week to help with language issues and during exercises.

Outline Part 1

- What distribution?
- Red Hat Fedora Core 2 installation and Kickstart Notes
- System commands (cp, ls, cd, rm, etc.).
- Basic editor usage with vi.
- Create and remove user accounts.
- Discuss /etc/passwd, /etc/group, /etc/shadow.
- Commands, programs, shells and paths.
- Use of 'su' command for 'root', and /etc/sudoers.
- Download RPM packages using FTP and install them.
- Directory structures – what's where.
- Learn how to shutdown and restart the server. Initialization levels.
- Discuss /etc/ and /etc/rc.d/init.d/ (services).
- Practice what we've learned if there is time.

What distribution?

Red Hat, Fedora, SuSE, Debian, Conectiva, Turbolinux, Mandrake, United Linux, Gentoo, Slackware, other? etc?

Or, do we use FreeBSD, Solaris, OpenBSD, NetBSD, HP/UX, AIX, Mac OS X, SCO, etc?

What do you run?

Our response:

- Currently Red Hat 9 appeared *almost* to be a standard
- If you know Red Hat Linux using commands and text file, then you know enough to use other distributions.
- Lots of local as others are likely running RH/Fedora.
- We want to teach how to scale services and "good network practices" using a reasonable system.

Fedora Core 2 and Kernel 2.6.x

Fedora Core 2 uses Kernel version 2.6.5-1.358 (see /proc/version). There are some changes (refer, also, to /usr/src/linux/Documentation/Changes):

- `mount` has more options and supports more filesystems.
- Modules work differently (/etc/modprobe.conf)
- Kernel building is a bit different.
- Larger disk (TB) and memory (64GB) support.
- Pcmcia support partially built in to kernel.
- Updates to NFS support and Quota support.
- More...

Installing Fedora Core 2

- How can you install? Upgrade?
  - Using CD/DVD.
  - Hard drive partition
  - Floppy/CD/DVD and an ftp/tftp/http/nfs server.
  - Floppy/CD/DVD and Kickstart file.
  - NIC PXE boot.
- Install options using the Kickstart graphicl tool.
- We might install at the end of the workshop.

Basic Commands

- cp, cd*, ls, mv, rm y man
  - (*built in command shell commands).
- Where are commands located?
- /bin, /usr/bin, /usr/local/bin, /sbin, /usr/sbin
  - The difference between “sbin”, “bin” and “ /usr”
- If you know DOS:
  - cp = copy
  - cd/chdir = cd/chdir
  - ls = dir
  - mv = move (before it was copy and delete/erase
  - rm = del[ete] and/or erase

Basic commands continued

Not a command, but we'll practice starting a detached process.

To do this you use the “&” symbol *after* the command you wish to run that opens a separate window.

For example, to open another terminal from within your terminal under XWindows type:

```
/usr/bin/gnome-terminal &
```


Some more commands...

- bg
- bzip
- cat
- chgroup*
- chmod
- clear
- chown*
- exec
- df
- dmesg
- du
- export
- find
- gcc
- grep
- groupadd*
- groupdel*
- gzip
- history
- ifconfig*
- info
- init*
- insmod*
- kill
- ln
- locate
- lsmod
- lsof
- mkdir
- “|” (pipe)
- man
- mount*
- more
- netstat
- nmap
- ping
- printenv
- ps
- pwd
- route*
- rpm*
- rmdir
- rmmod*
- set
- su
- tail
- tar
- telinit*
- top
- touch
- traceroute
- uname
- unset
- unzip
- usermod*
- users
- watch
- whereis
- which
- whoami
- apropos
- “ctrl-u”
- date*
- hexdump
- hwclock*
- mkisofs
- tcpdump

* = root

Looking for more information

Not only can you use commands to find information about your system, but you can look inside several files, and you can explore the /proc directory, as well.

Example of files with useful information:

- /etc/motd
- /etc/issue
- /etc/resolve.conf
- /etc/services
- /etc/X11/XF86Config
- /etc/modules.conf and with Kernel 2.6 /etc/modprobe.conf
- /etc/fstab

The vi editor

- Why use vi? Why not emacs, xemacs, joe, pico, etc.? (Make note of “pico -w”)
- vi exists in almost all distributions of Linux/Unix/BSD.
- If you have to work on a new machine, then vi will almost always be available to you.
- In reality, you are likely to use a different editor for more complex editing, but let's see what we can do with vi -->

Basic vi commands

Impress your friends...

- **Open:** vi fn, vi -r fn, vi + fn, vi +n fn, vi +/pat fn
- **Close:** :w, :wq, :q, :q!
- **Movement:** h,j,k,l y w, W, b, B, :n (+arrow keys)
- **Edit:** i, o, x, D, dd, yy, p
- **Search:** /pattern, ?pattern, n, N

Note: vi is alias to vim, and view is link to /usr/bin/vim under Fedora

Create and eliminate user accounts

- /etc/passwd, /etc/group, /etc/shadow, /etc/sudoers
- /usr/sbin/adduser --> /usr/sbin/useradd
- /usr/sbin/userdel
- /etc/default/useradd
- /etc/skel
- /etc/login.defs
- /etc/profile
- chsh, passwd, groupadd*, groupdel*, groupmod*, usermod*

Note: * requires root/admin access

/etc/passwd

The /etc/password file has the following format:

```
hervey:x:500:500:Hervey Allen:/home/hervey:/bin/bash
```

i.e.:

```
userid:pw:UID:GID:name:directory:shell
```

Using /etc/shadow the “pw” is represented by an “x”. If the user entry is actually something like a service, then the “shell” is represented with “/sbin/nologin”.

/etc/shadow

This file is used to hide encoded user passwords. Only root can (or should) read this file.

/etc/shadow has the following format:

```
hervey:$1$w!@F62adfk3$LCYjTI3udsd/tGP1pux1:12452:0:99999:7:::
```

i.e.:

- userid:
- password encoded with crypt:
- Days after Jan. 1, 1970 on which the password was last changed:
- Days until the user can change the password:
- Number of days before a password expires:

/etc/shadow continued

```
hervey:$1$w!@F62adfk3$LCYjTI3udsd/tGP1pux1:12452:0:99999:7:::
```

- Days before a password expires that a user is notified:
- Days after a password expires that an account is deactivated:
- Days since Jan. 1, 1970 that a user has been deactivated:
- Reserved:

/etc/group

In this file group names are specified (no more than 8 characters), password, the Group IDentifying number (GID), and a list of group members separated by commas:

For example:

```
mail:x:12:mail,postfix
```

The “x” - *If* there is a group password, then it is stored in encoded format in /etc/shadow.

What are groups used for?

Commands - programs – shell – path

What's a “command” and a “program”?

Why can't you always run all commands and programs on a system?

How do you “fix” this?

How do you see how things are configured for a user?

- /etc/skel
- /etc/profile
- /home/user/.bashrc
- /home/user/.bash_profile
- set, printenv, export

Using the `su` command

The “su” command is used to become a different userid, like root, without having to logout and log back in.

To use “su” to become root your userid has to be given permission to do this in “/etc/sudoers”.

You can allow users to run specific privileged commands using “/etc/sudoers” and “sudo”.

You can assign users to the “wheel” group and using “/etc/sudoers” you can allow them to run all or some commands.

More uses for the su command

Instead of having to open a root shell, you can run a privileged command like this:

```
sudo command
```

For example:

```
sudo less /etc/shadow
```

And, if you wish to open a different user shell *and* run their login scripts do:

```
su - userid
```

What happens if you don't use the “-”?

Package installation using RPM

- RPM stands for “Red Hat Package Manager”
- There are several systems to control and install software (rpm, dpackage/apt-get, yum, source). Each one has its advantages and disadvantages.
- RPM allows you to install and remove software on your server. You can, also, see the files associated with a package and where they install.
- RPM, however, cannot resolve software dependencies between software packages.

Using RPM

You can do quite a few things with RPM, but typically you install like this:

rpm -Uvh package-name.rpm

rpm -ivh package-name.rpm

Found out if something is already installed:

rpm -qa | grep package-name

Get more information about an already installed RPM:

rpm -qi package-name

rpm -ql package-name

Usando RPM continued

People use RPM differently. See the difference between "`rpm -Uvh`" and "`rpm -ivh`". Is it really necessary to use the "`-h`" option?

The key for anything that trys to organize your software is does it really function? RPM helps, but it does not solve the problems of knowing what you have installed, where, and what belongs to what software package.

If you don't believe me see how many times you end up using "`rpm –nodeps`" in the future... ;-)

Shutdown and restarting a server

How do you shutdown a Linux box?

- shutdown -t 60 message
- halt
- init/telinit 0

And, to restart?

- reboot (linked to /sbin/halt)
- shutdown -r now

Run levels

And, what was "init/telinit 0"?

Linux works with the concept of run levels. Each level has a meaning. These are:

```
0 – Shutdown. Never make this "default"
1 – Single user mode
2 – Multi-user mode, but without network file
    system (NFS).
3 – Multi-user mode with NFS.
4 – Not used
5 – Multi-user mode with X11 (Gnome, KDE,
    XWindows, etc.).
6 – restart. Never make this "default"
```


Run levels continued

The level your machine boots in to is defined in /etc/inittab.

For a server you will almost always want to start in run level 3, with the occasional exception going to “single user” mode (level 1).

The basic rule of thumb; “Server, level 3, computer for personal use, level 5”.

You can change run levels using “telinit --> init”, that is “telinit” is an alias for “init”.

Run levels continued

To shut down XWindows (Gnome, KDE) and work only in a shell (text), you can open a terminal in XWindows and type:

```
init 3
```

To restart XWindows type:

```
init 5
```

Now we are going to see how you can control what runs in each run level...

Services that run in each run level

This is a bit complicated, but the bits and pieces we use with Fedora (System V) are:

- /etc/rc.d/
- /etc/rc.d/rc0.d to rc6.d
- /etc/rc.d/init.d/
- /etc/inittab
- /sbin/service script option
- chkconfig

Controlling Services

First look at /etc/rc.d/init.d. Here we have the scripts that control most running services (note xinetd!).

Let's stop and start “nfslock”.

Let's see in which run levels “nfslock” runs:

```
/sbin/chkconfig -list | grep nfslock
```

Now we are going to use “chkconfig” to remove “nfslock” as a running service in all run levels.

Controlling services continued

First let's go to /etc/rc.d/init.d/rc3.d and rc2.d to explain what happens in these directories.

After this, let's configure things so that “nfslock” does not run the next time we restart our server (in /etc/rc.d/init.d):

```
chkconfig -del nfslock
```

Now let's play with “chkconfig”, the rcn.d directories, and services to understand what's happening.

Services summary

Perhaps you've seen /etc/rc.d/rc, rc.sysinit, or rc.local on other systems. These files, also, control how services run, but in a different manner.

The directory structure we have seen is called “System V” and comes from operating system structures before Linux was created.

Understanding how to start and stop services temporarily and permanently is key to being able to properly run and maintain a Linux server.

Summary part one

We have seen a bit about how to run a Linux server. Tomorrow we'll discuss much more about the structure of a Linux server.

Now, if there's time, we'll return to the command list near the start of this presentation and practice using these commands, or read about them using the command “`man`”.

Part two

Introduction to Linux Continued

Overview*

- Discussion about Linux partition schemes and options - /etc/fstab, dev.
- Permissions in Linux. Review commands “chmod” and “chown”.
- “ifconfig” to configure network cards and interfaces.
- Discuss Linux services and how to know what's running.
- The /proc directory.
- Network configuration changes using /etc/sysconfig/network-scripts
- We'll present /etc/crontab and practice using cron commands.
- The Linux kernel and how to recompile it.
- /etc/modules.conf and /etc/modprobe.conf.
- We mention firewalls.
- Installing compiled software (vs. RPM).
- Gnome vs. KDE and XWindows. What are they. Why are they not necessary for a server?
- Logs and where they exist. We'll inspect logs, noting /etc/syslog.conf.

* This is an example of how *not to* use a slide in a presentation...;-)

Linux filesystems

- ext2, *ext3*, reiserfs, jfs, hpfs, etc...
  - The current “winner” , ext3 (journaling)
- Structure of partitions:
  - / (“root”)
  - /usr
  - /var
  - /tmp
  - /home
  - swap

/ (“root”)

The root partition is where critical system files live, including the programs necessary to boot the system in to “single user” mode (run level 1).

The idea is that this part of the system does not grow or change, but rather stays isolated from the rest of the operating system.

The default Fedora install looks like this:

- /boot (approx. 100Mb)
- swap (1 to 2 x installed RAM)
- / (the rest of the hard drive)

/usr

Is used for system software like user tools, compilers, XWindows, etc.

If one has to expand* this partition for additional software, then having it separate makes this possible.

*We'll discuss this. We don't always install Linux with a separate /usr partition.

/var

This is where files and directories that change a lot are kept. For example, webserver logs, email directories, print spools, etc.

On a server it is a good idea to have /var in a separate partition. If someone, by accident or on purpose, creates a large amount of email, web activity, etc. then it's possible to fill this partition. If /var is not a separate partition this could cause your server to crash or hang.

/tmp

This is where users and application can save temporary files. By default quotas are not enabled in Linux, thus it's possible for a user, or program, to fill the /tmp partition on purpose or by accident.

Sometimes /tmp resides in /var/usr/tmp, but many Linux applications expect to have access to /tmp.

Another convention in the Unix world is to use /scratch, or /var/tmp instead of /tmp.

/home

This is where user directories are kept. Often user's email is not kept in this partition.

If you don't use quotas (common in Linux) then users can easily fill this partition.

An example: a user writes a program that produces a text file. On accident they create an infinite loop that writes text to a file and they don't notice this. This could fill your /home partition rapidly.

Swap

Swap is where virtual memory lives. Swap is it's own filesystem.

You can run without swap, and your PC may run faster, but this is dangerous if you run out of memory.

There are several opinions about what is the optimal swap size. This can depend on what type of services you run (databases need more swap). The general rule of thumb is that swap size should be somewhere between your RAM and twice your server's RAM.

Mounting filesystems

- If you want to mount a filesystem not listed in /etc/fstab then you need to use the mount command.
- First, you need to know what entry in the /dev directory describes the device you wish to mount (a cd, floppy, another hard drive, etc.).
- You, also, need to know what type of filesystem.
- For example, mounting a dos formatted floppy:
  - mount -t msdos /dev/fd0 /mnt/floppy

File permissions

- There are five categories and three types of permissions that you need to consider.
- The default file permissions are set with the `umask` command.
- There are two categories of permissions that relate to the user or group that is going to run a file (setuid, setgid).
- In addition you can set a "sticky" bit on directories so that only root and owner can make changes.
- The permissions available are "r" (read), "w" (write), and "x" (execute).
- You can assign permissions to world (a), group (g), and user (u).

File permissions continued

- A file belongs to a user. You can assign a file to another user or another group using the `chown` ("CHange OWNer") command.
- You can change permissions and/or owners for a group of files or for all files and all files in subdirectories using the `chmod` and `chown` commands.
- You can change directory permissions using the `chmod` command.
- As root you can modify file write access using the `chattr` command.

Ifconfig

During the week you are going to use this command *a lot*.

ifconfig is used to view the status of and to configure ethernet interfaces.

For example:

- `/sbin/ifconfig`
- `/sbin/ifconfig eth0 192.188.58.66 netmask 255.255.255.224`

/etc/hosts

In this file you should have, at least, this line:

```
127.0.0.1localhost.localdomain    localhost
```

For a private network you can use this file instead of using a DNS server. Linux looks in /etc/hosts before asking DNS to resolve an IP address.

You can change this order in /etc/nsswitch.conf.

We'll add an entry in /etc/hosts for “noc”.

Running services and ports

- To view all services (UNIX us “-Af”):
  - `ps -aux | more`
- What tcp ports are they using?
  - `sudo /usr/sbin/lsof -i`
  - `/bin/netstat -natu`
- What starts at boot time? See in /etc/rc.d/init.d
  - `/sbin/chkconfig –list | more`
  - `Don't forget xinet.d and inet.d`

The special directory /proc

- The /proc directory represents the state of your machine. It is an abstraction of your kernel and does not exist as a filesystem on your hard drive.
- For example /proc/cpuinfo has information about the cpu (or cpu's) of your machine. You can read it like this:
  - `cat /proc/cpuinfo`

But, you need to be root. The “/proc” directory is a security hole.

The /proc directory continued

Some of the files in /proc:

/proc/meminfo	Memory usage status.
/proc/version	Kernel version.
/proc/net/dev	Information about each network interface.
/proc/interrupts	IRQ usage in your system.
/proc/kcore	RAM contents (be careful!).

More /proc

If you think about it, there are several commands that simply show what's in /proc.

For example, the command `lsmod` is basically the same as:

```
cat /proc/modules
```

And, the command `ps` uses information you can see by looking in /proc/processid.

Configuration changes

If you want to make sure that your network interface configuration remains the same upon rebooting your machine you need to edit the file /etc/sysconfig/network-scripts/ifcg-eth0 (if it's for eth0).

If the “default route” cannot be calculated using the IP address and the netmask, then you need to use the command:

- route add default gw nnn.nnn.nnn.nnn

in order to access networks outside your local net.

Crontab

- The “cron” service allows you to automatically run programs when you want. Fedora uses “anacron” as well (better on personal PCs).
- This is configured in /etc/cron.allow, and /etc/cron.deny, if they exist.
- Use the command `crontab` in order to change the files that control how the crond daemon works.
- Or edit system-wide files in /etc/cron.hourly/daily/weekly/monthly.
- The cron files have a very specific format.

Crontab continued

A cron file that shows how a service is going to run has the following format:

```
Minute Hour Day Month Weekday Command
```

An example:

```
1 4 1 4 * /bin/mail user@dot.com < /home/user/joke
```

Send an email on the first of April.

The process “anacron” is used in Fedora. Anacron uses /etc/anacrontab and is used for machines that are not always running (i.e. desktops vs. servers).

The Linux Kernel

The kernel, or heart of Linux, can be adjusted and configured as you like. You must install the kernel source to do this. This resides in /usr/src.

If you want to recompile the kernel things have changed a bit with kernel version 2.6.x in Fedora. In the /usr/src/linux2-6.x directory you might do:

- `make mrproper` (removes old and "stale" files)
- `make xconfig` (".config" in configs/kernel-2.6.xx)
- `make` (builds bzImage in arch/i386/boot)
- `make modules_install`
- `mv arch/i386/boot/bzImage /boot/vmlinuz-2.6-new`
- `vi /boot/grub/grub.conf`

The Linux kernel continued

Recompiling your kernel is a bit of an "art", but you can make your kernel smaller, more efficient, and more secure by building a custom kernel for your hardware.

For much more detailed information go to:

http://www.kernel.org/

And download and read the documentation describing how to compile the kernel.

Hint: kernel.org docs are still not "grub" savvy.

Kernel loadable modules

- Changed with kernel 2.5.48+. `modprobe` program was simplified and now uses /etc/modprobe.conf vs. /etc/modules.conf to describe the "drivers" (modules) that will be loaded to support hardware on your PC.
- You can view what modules are in use, load a module manually, remove a module, and view module information:
  - `/sbin/insmod`
  - `/sbin/lsmod` logical link to `/sbin/insmod`
  - `/sbin/modprobe` logical link to `/sbin/insmod`
  - `/sbin/rmmod` logical link to `/sbin/insmod`
  - `/sbin/modinfo`

Firewalls

We mention this topic here as firewalls involve several aspects of Linux:

- With a modern Linux distribution you can expect that the kernel already has support for "netfilter" built in.
- You can control incoming, outgoing and forwarded network packets (tcp/udp/icmp) using "iptables" (netfilter).
- In Fedora you initialize iptables as a service in /etc/rc.d/init.d that reads the script /etc/sysconfig/iptables (this is just one method).
- To make a set of network filter rules that work well as a firewall for your server you need to understand network protocols and configuration very well as well as what you are running and how you wish to control these services.
- iptables-save, iptables-restore (ip6tables-restore, etc.) are available to make setting up rules easier.

Compiled software installation

It's likely you'll want to install software that's either not available as an RPM package, or that you need to change or reconfigure before installation.

In such cases, you need to compile the software from source code.

It's very typical that software comes as a single “tar” archive that is compressed (tar.gz or .tgz)

An example of how to install from source -->

Compiled software installation cont.

- Download a file fn.tar.gz to /usr/local/src.
- tar -xvzf /usr/local/src/fn.tar.gz
- cd /usr/local/src/fn-version
- ./configure
- make
- make install

This is all you need if it works, but now you don't have any good way to uninstall the software...

XWindows – Gnome – KDE

The first thing to understand is that Gnome and KDE use the XWindows graphical subsystem. Generally KDE programs run in Gnome and vice-versa.

For a server you do not need to run, or install, any of these.

If you are going to run one (Gnome?), you can run both (KDE and Gnome), or even more graphical windowing systems.

XWindows – Gnome – KDE cont.

- Which windowing system is better. There's no correct answer to this.
- To configurar how XWindows runs you specify this in the file /etc/X11/XF86Config.
- You can configure everything using menu-based tools, but if you understand how things work you can edit /etc/X11/XF86Config directly.
- To exit a graphical interface you may need to change to run level 3 (or 2 or 1).
- You can, also, go directly to a terminal using alt-ctrl-f2 through f6.

Logs – how to know what's happening

- To configure what services will report events to be logged see the file /etc/syslog.conf.
- Take a look at the file /var/log/messages. The "`tail`" command is very useful for this.
- Get used to using /var/log/messages (among others) to resolve problems. For example, problems running a service, type the command:

```
tail -f /var/log/messages
```

while you start and stop the service.

Logs continued

There are many log files. For example, if you run a webserver, like apache, all of the webserver logs are likely to be in /var/log/httpd

sendmail uses /var/log/maillog.

There are multiple software packages to read and automatically generate reports based on log files. See:

- http://nsrc.org/security/index.html#logging

to see some examples of available packages.

Summary part two

The Linux operating system is built in a modular manner to ensure stability, and to allow for extensibility and security.

Having access to the Linux source code allows for rapid development and the resolution of problems in a timely manner.

With Kernel 2.4, and 2.6, Linux comes closer in terms of stability and scalability of operating systems like FreeBSD and Solaris when offering network services.

More resources

- http://www.google.com/linux
- http://www.linux.org/
- http://www.linuxdocs.org/
- O'Reilly books (http://www.oreilly.com/)
  - "Linux Server Hacks" by Rob Flickenger
- http://www.sourceforge.net/
- http://www.redhat.com/

Hervey Allen - hervey@nsrc.org