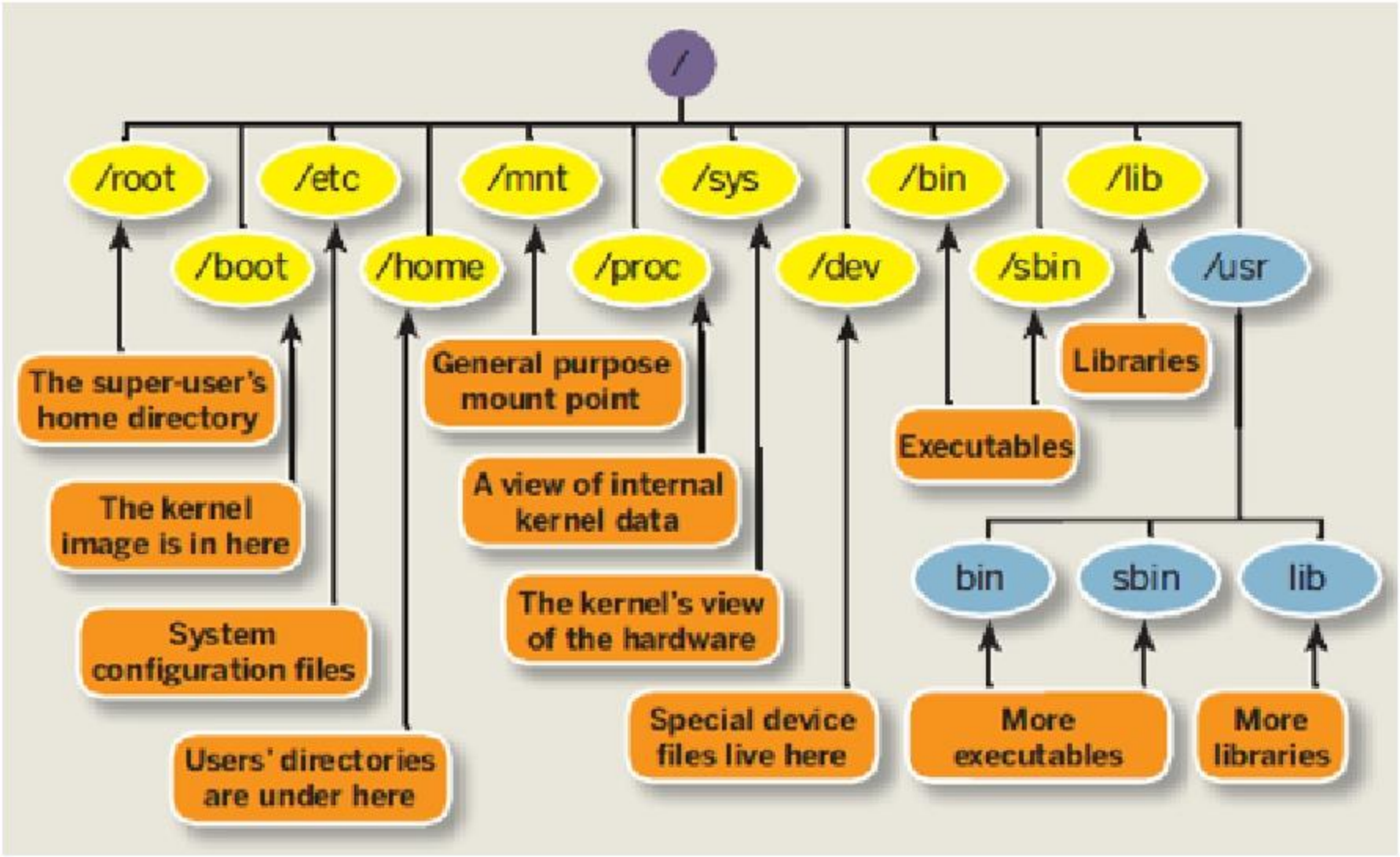
Logs on a Linux System

Getting started with Logging, Analysis and Auditing

Day: Modules

1. Linux System Admin Overview
2. Linux Security Overview
3. Logs

Sample Linux File System



http://www.tuxradar.com/content/take-linux-filesystem-tour

See what's running

Check for a process by name

- **`ps auxwww | grep apache`**
- **`lsof`**

```
sysadm@pc102:~$ ps auxwww | grep apache
root      1029  0.0 24.5 137524 125184 ?       Ss   01:29   0:02 /usr/sbin/apache2 -k start
www-data  1062  0.0 23.1 134788 117836 ?       S    01:29   0:00 /usr/sbin/apache2 -k start
www-data  1087  0.0 23.8 414236 121236 ?       Sl   01:29   0:00 /usr/sbin/apache2 -k start
www-data  1088  0.0 23.8 414236 121240 ?       Sl   01:29   0:00 /usr/sbin/apache2 -k start
sysadm    1426  0.0  0.1   3320   804 ttyS0    S+   02:40   0:00 grep --color=auto apache
```

Stop the process by PID (Process ID). From above listing:

- `sudo kill 1029` (why this one?)
- `Sudo kill -9 1029` (force stop if hung)

```
sysadm@pc102:~$ ps auxwww | grep apache
sysadm    1430  0.0  0.1   3320   808 ttyS0    S+   02:46   0:00 grep --color=auto apache
```

Viewing Files (part I)

Several ways to view a file:

1. cat <filename>!
2. more <filename!
3. less <filename>!

!

- cat is short for *conCATenate*
- “less is more”

Viewing files revisited

Sometimes files are viewed through a pager program (“more”, “less”, “cat”). Examples:

```
man sudo
less /etc/services
```

- Space bar for next page
- “b” to go backwards
- “q” to quit
- “/” and a pattern (/text) to search

Kinda looks like vim, no?

“`less` is more”

Obtaining help

To get help explaining commands you can do:

- man <command>!
- <command> --help!

man stands for “man”ual.

More on “man”

- man man!

More on Linux directory structure:

- man hier!

Auto-complete commands with tab

Core concept:

Once you type something unique, press TAB. If nothing happens, press TAB twice.

If text was unique text will auto-complete.

A command will complete, directory name, file name, command parameters will all complete.

If not unique, press TAB twice. All possibilities will be displayed.

Works with file types based on command!

Auto-complete commands with tab

Very, very, very useful

"The tab key is good", "the tab key is our friend", "press the tab key", "press it again" -

Tab works in the *bash* shell. Note, the *root* user might not use the *bash* shell by default.

Auto-completion

We'll do this now:

$ cat /etc ! ! !(TAB twice quickly)!

$ cat /etc/netw! ! ! ! !(TAB)!

$ cat /etc/network/in! !(TAB)!

Module 2: Security Overview

Goal

Understand the following:

- The Linux / Unix security model
- How a program is allowed to run
- Where user and group information is stored
- Details of file permissions

Security model

Numeric IDs

user id (uid 0 = "*root*", the superuser)
group id
supplementary groups

Mapped to names

/etc/passwd, /etc/group (plain text files)

Suitable security rules enforced

e.g. you cannot kill a process running as a different user,
unless you are "*root*"

Filesystem security

Each file and directory has three sets of permissions

- For the file's uid (user)
- For the file's gid (group)
- For everyone else (other)

Each set of permissions has three bits: **rwx**

- File: **r**=read, **w**=write, **x**=execute
- Directory: **r**=list directory contents, **w**=create/delete files within this directory, **x**=enter directory (e**x**ecutable)

Filesystem security

The permission flags are read as follows left to right:

-rw-r--r--!	**!**for regular files,
drwxr-xr-x!	**!**for directories

We will experiment on permissions later in the day.

Users and Groups

Linux understands Users and Groups

A user can belong to several groups

A file can belong to only one user and one group at a time

A particular user, the superuser “*root*” has extra privileges (uid = “0” in /etc/passwd)

Only *root* can change the ownership of a file

Users and Groups cont.

User information in `/etc/passwd`

Password info is in `/etc/shadow`

Group information is in `/etc/group`

`/etc/passwd` and `/etc/group` divide data fields using "`:`"

/etc/passwd:

```
joeuser:x:1000:1000:Joe User,,,:/home/joeuser:/bin/bash
```

/etc/group:

```
joeuser:x:1000:
```

A program runs...

A program may be run by a user, when the system starts or by another process.

Before the program can execute the kernel inspects several things:

- Is the file containing the program accessible to the user or group of the process that wants to run it?
- Does the file containing the program permit execution by that user or group (or anybody)?
- In most cases, while executing, a program inherits the privileges of the user/process who started it.

Piping and Sudo privileges of the user/process who started it.

A program in detail

When we type:

```
ls -l /usr/bin/top
```

We'll see:

```
-rwxr-xr-x 1 root root 68524 2011-12-19 07:18 /usr/bin/top
```

What does all this mean?

```
-r-xr-xr-x  1   root     root     68524   2011-12-19 07:18 /usr/bin/top


---------- --- -------  -------  -------- ------------     -------------
     |          |       |           |            |            |                        |
     |          |       |           |            |            |                  File Name
     |          |       |           |            |            |
     |          |       |           |            |           +---   Modification Time/Date
     |          |       |           |            |
     |          |       |           |           +-------------    Size (in bytes
     |          |       |           |
     |          |       |          +------------------------          Group
     |          |       |
     |          |       +---------------------------------          Owner
     |          |
     |          +---------------------------------------   “link count”
     |
     +-----------------------------------------------    File Permissions
```

Group

The name of the group that has permissions in addition to the file's owner.

Owner

The name of the user who owns the file.

File Permissions

The first character is the type of file. A "-" indicates a regular (ordinary) file. A "d” indicate a directory. Second set of 3 characters represent the read, write, and execution rights of the file's owner. Next 3 represent the rights of the file's group, and the final 3 represent the rights granted to everybody else.

(Example modified from **http://www.linuxcommand.org/lts0030.php**)

Access rights

Files are owned by a *user* and a *group* (ownership)

Files have permissions for the user, the group, and *other*

“*other*” permission is often referred to as “world”
The permissions are *Read*, *Write* and *Execute* (r, w, x)

The user who owns a file is always allowed to change its permissions

Some special cases

When looking at the output from "`ls -l`" in the first column you might see:

```
d = directory
- = regular file
l = symbolic link
s = Unix domain socket
p = named pipe
c = character device file
b = block device file
```

Some special cases cont

In the Owner, Group and other columns you might see:

`s = setuid`	[when in Owner column]
`s = setgid`	[when in Group column]
`t = sticky bit`	[when at end]

Some References

http://www.tuxfiles.org/linuxhelp/filepermissions.html

http://www.cs.uregina.ca/Links/class-info/330/Linux/linux.html

http://www.onlamp.com/pub/a/bsd/2000/09/06/FreeBSD_Basics.html

File permissions

There are two ways to set permissions when using the `chmod` command:

Symbolic mode:

testfile has permissions of `-r--r--r--`

```
                                    u   g   o*
$ chmod g+x testfile      ==>  -r--r-xr--
$ chmod u+wx testfile     ==>  -rwxr-xr--
$ chmod ug-x testfile     ==>  -rw--r--r—
u=user, g=group, o=other (world)
```

File permissions cont.

Absolute mode:

We use octal (base eight) values represented like this:

Letter	Permission	Value
r	read	4
w	write	2
x	execute	1
-	none	0

For each column, User, Group or Other you can set values from 0 to 7. Here is what each means:

0= **---** 1= **--x** 2= **-w-** 3= **-wx**

4= **r--** 5= **r-x** 6= **rw-** 7= **rwx**

File permissions cont.

Numeric mode cont:

Example index.html file with typical permission values:

```
$ chmod 755 index.html
$ ls -l index.html
-rwxr-xr-x  1 root  wheel  0 May 24 06:20 index.html

$ chmod 644 index.html
$ ls -l index.html
-rw-r--r--  1 root  wheel  0 May 24 06:20 index.html
```

Inherited permissions

Two critical points:

1. The permissions of a directory affect whether someone can see its contents or add or remove files in it.

2. The permissions on a file determine what a user can do to the data in the file.

Example:

If you don't have write permission for a directory, then you can't delete a file in the directory. If you have write access to the file you can update the data in the file.

Any questions?

?

Log files (a few examples)

```
/var
   /var/log
   /var/log/apache2
   /var/log/apache2/access.log
   /var/log/apache3/error.log
   /var/log/auth.log
   /var/log/boot.log
   /var/log/dmesg
   /var/log/kern.log
   /var/log/mail.info
   /var/log/mail.err
   /var/log/mail.log
   /var/log/messages
   /var/log/mysql
   /var/log/syslog
```

Log file: who & what's doing what

The most critical place to solve problems

- System messages, including:
  - Problems
  - Security issues
  - Configuration errors
  - Access issues
- Service messages, including:
  - Same as above

When something does not work…

…Look in your log files first!

Troubleshooting: Logfiles

Log files are critical to solve problems. They reside (largely) in /var/log/

Some popular log files include:

/var/log/messages or /var/log/syslog

/var/log/apache2/error.log

/var/log/mail.log

/etc/namedb/log/* (later in the week)

To view the last entry in a log file:

```
tail /var/log/messages
```

To view new entries as they happen:

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

There's More

But, hopefully this is enough to get us started...

Some Resources

http://www.ubuntu.com

http://ubuntuforums.org

http://www.debian.org

http://ubuntuguide.org

http://en.wikipedia.org/wiki/Debian

http://en.wikipedia.org/wiki/Ubuntu_(Linux_distribution)

GIYF (Google Is Your Friend)

End

Questions ?