Introduction to Real Life Administration in Linux
Welcome to Linux Lecture Series - Lecture No. 4

Orr Dunkelman

orrd@vipe.technion.ac.il

Why do I need this?

Just like other UNIX environments, Linux gives the system administrator unlimited power over the computer. This approach was developed in the early '70's when a system administrator, was someone who really knew what he is doing.

However, If you have Linux in your home, you might do damages to it without noticing. As several operations are irreversible, you might have at the end a non-working system.

Basic Concepts

Linux assumes one thing about *Root* (the system administrator) - that he knows what he is doing. This gives more power to the super user than in other cases, as root can do everything (!) he desires with the computer.

For example, root can delete all the file systems, without a second thought (unless the "*rm*" command was defined to be "*rm -i*"). Root can also declare the cd-rom to be writable (assume what would the system do when it tries to write over a read only drive).

Basic Concepts (cont.)

Rule # 1: Backup!

Before changing any configuration file, copy it to another name. For example, before changing the */etc/fsttab*, copy it to */etc/fsttab.old*.

Rule # 2: Learn your system!

As you will deal a lot with system configuration files, learn to read them when the system is up and running. Most configuration files live in */etc/*, home of configuration files. Learn how important files look like, and where you put them the last time you used them.

Linux' File System

Linux file system is named EXT2. Today EXT3 is developed, and Silicon Graphics (SGI) has donated her Jornaling File System to Linux.

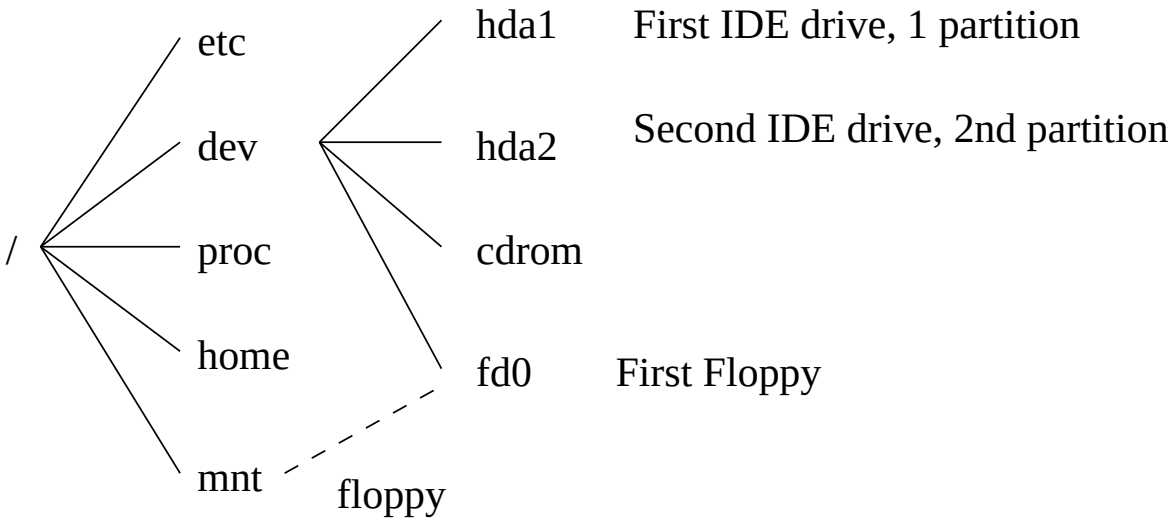
The root directory /, is the base for all the file system. The file system reminds a tree, where each file (or directory, as directories are a special type of file) has an ancestor.

Unlike Windows (and DOS) one can have an interaction between distinct branches. This is done using a symbolic link, which enable connecting two different branches.

When the target of the link is a device (like other drive) the command which is used is mount.

More information: ln, mount, umount, fstab.

Linux' File System Illustration



/mnt/floppy = /dev/fd0

Linux' File System - Important Directories

/bin, /sbin - These two directories contain the binaries (executables) for many programs.

/etc - In this directory, many configuration files are stored.

/dev - This is the directory related to the devices. Linux (as other UNIXes) handle devices as files.

/mnt - This is usually the directory for which mounting of removable media (CDs, floppys) is done (meaning, you create the connection between the regular file system and other drives through /mnt/<some directory>).

/var - This directory holds many important files like the logs of the system.

Other directories has also a meaning, but you can find it yourself.

Linux' File System - Access Control

Each file has an owner (as we have many users on the system) which is composed of a user and a group (in most system each user is either part of the users group, or a group containing only himself).

Like other UNIXes, Linux relates to each file and directory several security flags. The main 9 flags are grouped into 3 groups of 3 similar flags each. The three flags in each group are RWX - R stands for read permission, W for write permission and X for executing permission. The first group related to the permissions to the owner of the file, the second related to the permissions to the group of the file, and the last is related to all the users.

Thus, $rw-r--r--$, means that the owner can read and write to the file, and the group and rest of the world can only read.

How to Change Permissions

We assume that some file (or directory) has permissions. However, we want to change the permissions (allow read access to the whole world, for example). The command which is used is *chmod*.

We demonstrate one of several syntaxes available (see the manual pages for the other syntaxes). The permissions $rwx\ rwx\ rwx$ are considered as 3 groups - user, group and world. Each group has 8 combinations (from $---$ to rwx). Therefore, each permission can be written as 3-digit number. Each digit is between 0 ($---$) and 7 (rwx).

Therefore, to change the permissions of the file $just_file$ to $r-x\ r--\ \ ---$ we execute *chmod 540 just_file*.

Configuration Files

File 1: /etc/fstab - File Systems Table

```
/dev/hda1    /             ext2    defaults       1 1
/dev/hda2    swap          swap    defaults       0 0
/dev/fd0     /mnt/floppy   auto    noauoto,nodev,unhide 0 0
/dev/cdrom   /mnt/cdrom    auto    noauto,exec,nodev,ro 0 0
none         /proc         proc    defaults       0 0
none         /dev/pts      devpts  mode=0622      0 0
/dev/hdb1    /mnt/win      vfat    defaults       1 2
```

The fstab file contains a list of all file systems which are usually loaded and their parameters. Some of the file systems are mounted while the system is loading, and the rest when root asks for it.

Configuration Files - /etc/fstab

The first field in each line is the device name. The second is the place to mount the file system. The third is the file system type (ext2 = native Linux). The fourth field is the options field. For more information about the options (and available file systems) try "*man mount*" or "*man fstab*".

It is good to know what the fifth and sixth fields are but they are not a must. Look at the fstab manual for this information.

See also mount, umount, dump, fsck, df

Configuration Files

File 2: /etc/lilo.conf - The System Loader

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux
image=/boot/vmlinuz-2.2.14
      label=linux
      root=/dev/hda1
      initrd=/boot/initrd-2.2.9-19mdk.img
other=/dev/hdb1
      label=dos
      table=/dev/hdb
```

Configuration Files - /etc/lilo.conf

The lilo.conf holds the possible loader options. Thus, if you want to load windows, you need to specify this in the lilo.conf file.

boot keeps where Master Boot Record resides (MBR). *map* keeps the map file location. The *install*, is where the new boot record should be directed to. The *prompt* command forces the lilo to ask for instructions, and the *timeout* is the number of seconds lilo waits for the user's choice. *default* is the default image to be loaded.

Each image is either a Linux one, or of *other* type. If the image is Linux the kernel image which defined is loaded. The *label* field contains the name which lilo identify the system as.

If other is specified, then the *table* field is where the partition table lies.

Configuration Files - /etc/lilo.conf

After changing this file, the lilo command should be run (as root).

There is a special load option, when things get really bad, and the system refuse to load, try to ask for *linux single* from LILO. If lilo loads linux single, you have access without need of the password as root (great when you forget the root's password). This feature can be used against you so be sure the computer is unreachable by untrusted people.

See also lilo

Configuration Files

File 3: /etc/passwd - Where the Password Lies

```
root:x:0:0:The System Administrator:/root:/bin/tcsh
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
orrd:x:1164:100:Orr Dunkelman:/home/staff/orrd:/bin/tcsh
```

Configuration Files - /etc/passwd

Each user is kept in the /etc/passwd file. The file contains information about the user's name, password, uid, gid, real name, where is home directory is, and what shell to use when he logs in (according to order).

To change a password, the user runs the command *passwd* from it's own account. However, the change is done to a file, we don't want the users to be able to change... Imagine, a user, who can change the root password.

The solution is a special command. The command passwd, is owned by root, who let others use it, by giving the permissions $r-xr-xr-x$. However, this would not suffice, as the user which run the command will have uid of his own, which is not allowed to write over the /etc/passwd file.

Configuration Files - /etc/passwd

In order to allow the user to have the privileges of root, only for a short time and for a known (and authorized) purpose, the administrator, enable the "set-uid bit", a special status, which allows the user who run the program to have uid as if he was root (and thus change the /etc/passwd file).

To change user real name (fifth field), the user runs *chfn*.

To change the shell the user has, he runs the command *chsh*

See also chsh, chfn, pam (don't use pam without knowing perfectly what you are doing!), shadow.

How to Change Permissions - Special Permissions

In order to activate the "set-uid bit", the file must be changed so it is executable, and one add 4 in the beginning of the 3-digits (thus have a 4-digit number). For example *chmod 4540 just_file*.

Two more special bits exist, the "set-gid bit" and the "sticky-bit".

See also chmod, chmod(2), setuid, setgid.

How to Change Ownership

In order to change the owner of some file from one user to another, use the command *chown*. The command, takes two arguments - *new_user.new_group* and the file name.

In order to change only the group, use *chgrp*.

I forgot the configuration file name

Well you are not in trouble. As in the console text mode, you can access all files, you can search within the /etc/ files, and look for a known string.

If for example, you forgot where the file system table lies, and you know that the string "/dev/" is in it, use the command "*grep "/dev/" /etc/**" which will output all files (and lines) in /etc/ which contain the string.

Thus, you narrowed the search.

Another option is to try *rpm -qfl /a/relvent/file | grep /etc*

More advanced searching is also available, see more at the grep manual.

See also egrep

Basic Concepts

Rule # 3: Know Your Hardware

Write down (aside) the hardware details of your system. From CPU and memory, to the type of sound card, networking card, etc.

Many problems can be identified as hardware-related, and thus knowing exactly what is in the computer is essential.

Basic Concepts

Rule # 4: Learn and Update

Linux is very dynamic system. Though most of the stuff remains the same, there are things which are very dynamic (for example, supporting new hardware).

You need to live on the edge of new technology, and be aware for new kernels, new programs, and new patches. Useful links are found in the club's home page, or at *http://www.linux.org.il*.

Updating Software

Software in Linux can come in several formats. The most common is the .tar.gz files. Those files are usually the source code files and need to be compiled. But before compiling, first they need be extracted by running *tar -xzvf file-name.tar.gz*. It is highly recommended to open files in a new directory (i.e., create a directory and put it there).

After extracting the files, usually there is a README/INSTALL file, containing the exact installation process. However, most of installations require to perform (some of) the following steps:

1. *make config*
2. *make*
3. *make install*

Updating Software - Kernel

The following procedure is to be used:

1. Go to /usr/src, remove the linux directory (the linux directory should be a symbolic link to some other directory, like linux-2.2.3)
2. Extract the *linux-2.2.14.tar.gz* file there (it will create directory named linux).
3. Rename the directory (the mv command) to linux-2.2.14, and change to that directory.
4. *make menuconfig*, once the menu shows, check for the options you want.
5. *make dep*
6. *make clean*

7. *make modules*
8. *make modules_install*
9. *make bzImage* - bzImage will create a new kernel image under arch/i386/boot/bzImage. move this file to the /boot place, and update lilo.conf (do not forget to run lilo afterwards).
10. Create a soft link (*ln -s*) to the /usr/src/linux-2.2.14 directory named /usr/src/linux.

A new kernel is ready. Rebooting the system would restart the new kernel (update the default value in lilo.conf, but don't remove the entry of the previous kernel, for the case problems occur).

Updating Software - RPM

Another format to distribute software is the RPM format. The format is available under Red Hat, Mandrake, etc. and is used to distribute a software, without worry about dependency problem (as the RPM installation will know to alert the problem).

The usage is quite easy (and there is also a graphical interface available) - *rpm -Uvh rpm-file.rpm*.

To update - *rpm -U*.

You can find more options in the manual. **See Also** rpm, gnurpm, kpackage.

Rebooting the System

As in other operating systems, it is wise to alert the system of a shutdown (or reboot) instead of just shutting the power down physically. The command for doing that is the *shutdown*.

shutdown can be instructed to reboot (-r switch) or to halt (-h switch), when one can determine that the shutdown will occur later or now. The usage is of the form $shutdown - < rh > \ \ < time >$. Where time format is specified in the manual pages. For immediate shutdown - *shutdown -h now*.

See also fsck, init, halt, reboot

Rebooting the System - Hard Reboot

Assume that due to some problem a physical shutdown was forced (e.g., electricity break down). The file system were not cleanly unmounted. Thus once the system will be loaded again, an error message of the form */dev/hda2 was not cleanly unmounted* will appear.

The system knows to run the fsck utility, which is a check and fix utility to Linux native file systems. This utility is from the file system utilities family, which contain also the mkfs, which create Linux file systems.

See also fsck, mkfs, fstab

Basic Concepts

Rule # 5: Read The F. Manual (a.k.a. RTFM)

Whenever a new command or a problem arises, it is wise to try to first deal with it alone. Sometimes you succeed and feel good about yourself. To do it yourself, you need to know where to look. The first place is of course the manual pages (the *man* command).

A second place to look for, is the **HOWTO** section. HOWTO is a manual for a complex operation (like installing a new kernel). They vary in subjects (ISDN-HOWTO, to Hebrew-HOWTO), and can be found in *http://www.linuxberg.org* site.

A third place is the Linux Documentation Project (LDP) which documents Linux. It can be found on-line in many places, like *http://www.linux.org.il*.

There are also on-line guides, etc.

How to work as Root

Try not to **login as root**. Try the following scheme instead:

- Log in as a regular user
- Run *su* - and then enter Root's password.

Special Files

The following files contain much information about your system:

/proc/ - All the processes area.
/proc/meminfo - Information about yours memory settings.
/proc/cpuinfo - Information about yours CPU.
/var/log - Several important logs.

Useful Administration Utilities

The following utilities might be useful in many cases:

top - Gives information about what the system load is.
ps - The users processes.
kill - Caution, use only if you know what you are doing. The way to end a process if it gets stuck or something like that.
df - Lists the current mounted file systems.
netstat - Information about your current networking status.
gzip - Compressing and Decompressing files. Useful if the tar utility tells you it doesn't know the -z flag, in that case you first run *gzip -d*.
tar - Archiver (make many files a single file).

Processes

In the UNIX environment, every program which is executed is a process (or can become more than one process).

The system keeps for each process important data, like the process id (pid) and the running environment (memory, priority, nice priority), and a table named file descriptors, which contain information about files (or devices) which the process interacts with.

Processes - Fork

Process can create a son-process using the system call fork().

Once a process called for fork, the system creates a new process, with the same (!) parent process' running environment (memory, file descriptor table, etc.).

There is a mechanism to distinguish between the father and the son (or mother and daughter, for the politically correctness), which enables one process to run several sub-processes (sons).

Most common implementation for such a case is a system daemon (like SSH daemon or web-server), that for each new request, copy itself (fork) and one part (father) wait for more requests, and the second one (son) handle with the request.

Processes - Signals and Murderers

Each process has a possibility to get signals. Signals are a way of the system (or the user) to alert a process to a fact.

Signals are used to communicate between processes (the system itself is a pack of processes), and carry information, like the fact that soon the system is going down due to power failure (SIGPWR).

Processes - Signals and Murderers (cont.')

Signal may be caught for the process to handle. For example, we might want the process to be able to de-allocate resources he caught before exiting. However, some signal can not be caught.

If a process is stuck, we can send to it a special signal, called SIGKILL (use only if you are a murderer!). SIGKILL will terminate the process, and free all allocated things the process has.

Be very careful when using SIGKILL, you might kill something important, so use it only when you know what you are doing.

Signals are being sent by user (or administrator) using the command $kill \ < signal > \ < pid >$.

Processes - Signals and Murderers (cont.')

Common signals include:

- SIGINT - signal interrupt, sent to the process once CTRL Break is pressed.
- SIGKILL - signal kill, sent by the system to end the process. Cannot be caught.
- SIGSTOP - signal stop, the process is frozen. Can be sent from keyboard. CTRL+Z emulates this signal.
- SIGCONT - signal continue, defrost a frozen process.
- SIGSEGV - signal invalid memory reference. Also known as "Segmentation Fault (celebration)".
- SIGHUP - signal hang-up detection. Now days mostly used to restart programs after their configuration file has been changed.

Processes - Family Bonds

What is connection between father and son?

If the son terminates, the father gets a signal telling him that his beloved son died (SIGCHLD). This is the usual case (in the UNIX model) that the father watches the death of his children. But what if the father dies first?

The assumption is that father is killing his children when he terminates. If that doesn't happen, his children become what is called "orphants". They wait till the father properly terminates, if this happen they'll be terminated by the system.

Sometimes the father can't accept the loss, thus, the son process finished to run, but the father ignore his death so the process still hangs in air just like a "zombie".

Related Commands - Processes

- *ps* - Process Status. Gives information about processes.
- *ps fax* - Useful flag of the ps command, gives the information in a nice way.
- *pstree* - Process Tree. Shows the processes' trees.
- *top* - Presents the process information in table.

Related Commands - Signals

- *kill* - Send singal to process.
- *killall* - Send signal to many processes related by process name. Be extremly careful when using this command.
- *alarm* - Set a timer to get an alarm signal.
- *sleep* - Go to sleep for a period of time.