Linux Kernel Hacking Free Course, 4th edition

Distributions for Linux

Vincenzo Laurenziello

University of Rome - Tor Vergata

Outline of the Talk

- What is a distribution
- Distributions considered:
  - Fedora
  - Slackware
  - Ubuntu
- Filesystems commonly used in a CDROM
- Common problems and how to solve them

What is a Distribution (1/2)

A *distribution* includes:

- a kernel which:
  - can boot from any block device, i.e., Hard disk, cdrom, pen drive, etc. . .
  - recognizes the I/O devices included in the computer
  - supports several filesystems, i.e., `ext2/3`, `ISO9660`, `procfs`, `sysfs`, etc. . .
- a set of packages that contain:
  - applications
  - libraries
  - configuration files

What is a Distribution (2/2)

There are over 500 Linux distributions. They can be classified according to:

- User Tipology
  - *Newbie Users*: never used a *nix OS
  - *Normal Users*: use graphical configuration tools, they prefer user friendliness
  - *Experienced Users*: use advanced tools and configure manually everything, they know Linux quite well
- Workload Tipology
  - *Desktop Distributions*: general-purpose, easy to use, handles multimedia applications
  - *Live Distributions*: doesn't use the hard disk, can be used for data recovery or demo
  - *Enterprise Distributions*: specialized for managing critical applications
  - *Real-Time Distributions*: specialized for real-time applications
  - *Embedded Distributions*: tailored for specific hardware with limited resources

Differences Among Distributions (1/2)

- *User-Friendliness*:
  - Fedora uses Anaconda, it can work in graphical mode or in text mode
  - Slackware uses only a textual interface called `dialog`. It's simple and powerful
  - Ubuntu runs like an LiveCD, thus we can run other tasks, i.e., surfing Internet, during the installation
- *Booting*:
  - Fedora uses a SystemV style. Every runlevel is stored in the directory `/etc/rc.d/rc.`*X*
  - Slackware uses the `BSD` style. Every runlevel is described in a file called `/etc/rc.d/rc.`*X*, but it supports also SystemV init files
  - Ubuntu uses a SystemV style. Every runlevel is stored in the directory `/etc/rc`*X*`.d`

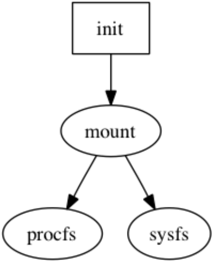
Differences Among Distributions (2/2)

- *Package Types*:
    - Fedora packages are `cpio` archives with modified headers
      Package managers: `rpm` or `yum`
    - Slackware packages are gzipped `tar` archives
      Package managers: `installpkg`, `removepkg`, `upgradepkg`, and `pkgtool`
    - Ubuntu packages are `ar` archives
      Package managers: `dpkg`, `apt-get`, `Synaptic`
- *Personalization*:
    - on Fedora we can use `rpm-build` to create a personal package
    - on Slackware we can use `makepkg` to create a personal package
    - on Ubuntu we can use `dh-make`, `debuild` to create a personal package

Common Elements

All distributions considered:

1. boot from CDROM or DVDROM using a bootloader like: `isolinux` or `GRUB`
2. mount a `miniroot` provisional filesystem derived from `initrd` or `initramfs`
3. mount the `procfs` and `sysfs` filesystems



initrd

- it is the *initial ramdisk*
- it is a gzipped file that contains a filesystem
- it is used during kernel start up (the pathname of initrd is passed as a bootloader parameter)
- it is mounted on a RAM-disk, aka a RAM based block device
- the kernel executes the `/linuxrc` file stored in it

```
$ dd if=/dev/zero of=my_initrd.img bs=1024 count=1000
$ mkfs.ext2 -F my_initrd.img
$ mkdir initd_dir; mount -oloop my_initrd.img initrd_dir
$ cp -ar /data/* initrd_dir/
$ umount initrd_dir; rmdir initrd_dir

$ gzip my_initrd.img
```

initramfs

- it is the successor of `initrd`
- it is a gzipped file that contains a `cpio` archive
- as in `initrd`, it is used during kernel start up (the pathname of initrd is passed as a bootloader parameter)
- it uses `ramfs`
- the kernel executes the `/init` file stored in it

```
$ cd /data
$ find .  | cpio -o -H newc > ~/my_initramfs.img

$ cd ~; gzip my_initramfs.img
```

initrd vs initramfs

`initrd`	`initramfs`
Uses a block device with fixed amount of memory	Uses the necessary space
Uses a specific filesystem with cache memory	Uses the built-in filesystem `ramfs`
Calls `pivot_root`	Calls `switch_root`

Steps required to build one of them

Creates a file	Get a list of files
Formats it	Stores data
Mount it	
Stores data	
Umount it	

Current distributions use `initramfs`.

procfs

`procfs` is a pseudo-filesystem that:

- displays information about running processes:

```
$ readlink /proc/self/exe
/bin/readlink

$
```

- reads, and eventually edits, some kernel parameters:

```
$ cat /proc/sys/kernel/ctrl-alt-del
0
$ echo 1 > /proc/sys/kernel/ctrl-alt-del
$ cat /proc/sys/kernel/ctrl-alt-del
1

$
```

sysfs

`sysfs` is another important pseudo-filesystem.
It reacts to plug-ins and plug-outs by adding and removing files in /sys
The most important subdirectories are:

- `/sys/devices`: it contains all devices recognized by the kernel. They are ordered by tipology of device;
- `/sys/bus`, `/sys/block`, `/sys/class`: these directories contain symlinks to the objects present in `/sys/devices`:
  - `/sys/bus`: ordered by tipology of bus used from a device;
  - `/sys/block`: it shows only the block devices;
  - `/sys/class`: it organize the informations into many hierarchical classes of devices.
- `/sys/modules`: contains all modules (statically or dinamically linked) that use sysfs APIs

sysfs Example

Using udevmonitor we can check what sysfs is doing

```
# udevmonitor &
```

If we insert a module, for example

```
# modprobe usb-storage
```

sysfs reacts and populates /sys with new files and directory, for example

```
/module/usb_storage/drivers
/bus/usb/drivers/usb-storage
/block/sdb

/class/usb_device/usbdev1.5
```

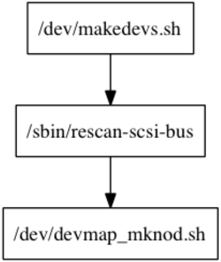
ramfs, tmpfs

- ramfs is a filesystem that store files in RAM. Only root can write on this filesystem.
- tmpfs is an extension of ramfs. Contrary to ramfs, the pages of tmpfs can be swapped out if necessary. Users can create their own tmpfs.

Slackware

- Language: `bash script`
- Three kernels to use:
  - `huge.s`: IDE+SCSI
  - `hugesmp.s`: IDE+SCSI with SMP support
  - `speakup.s`: IDE+SCSI with speech synthesizers
- `/dev` is populated by `/dev/makedevs.sh`
- it calls a shell. To install this distribution the user must issue the `setup` command

Detecting Hardware on Slackware

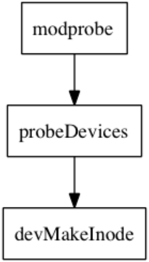


- `/dev/makedevs.sh`: parses `/proc/partitions` and populates `/dev` using `mknod`
- `/sbin/rescan-scsi-bus`: loads `sg` module, removes and adds all devices found in `/sys/class/scsi_host/` or in `/proc/scsi/scsi` file
- `/dev/devmap_mknod.sh`: creates `/dev/mapper/control` for LVM devices

Fedora

- Language: `C` & `python`
- It loads modules using the `init_module` syscall
- `kudzu` is used to probe devices (in Fedora 9 kudzu will be removed)
- It populates `/dev` using the `mknod` syscall
- Starts user interface directly and spawn shells.

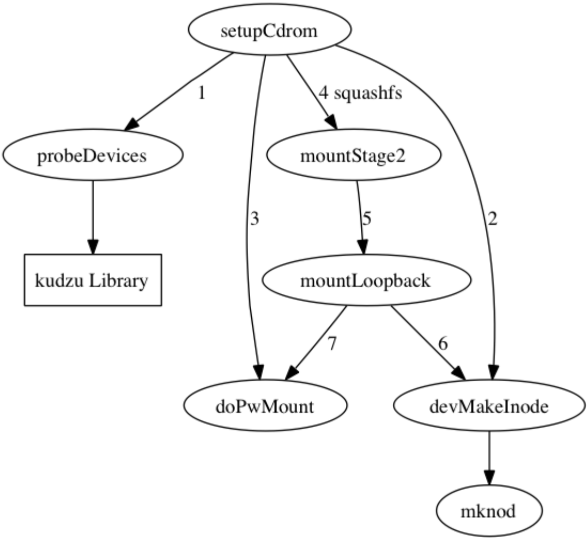
Detecting Hardware on Fedora (1/2)



- loads essentials modules about filesystems, IDE, SCSI, USB, firewire, and RAID
- calls `probeDevices` implemented in `kudzu` library that parses `/proc/ide` for IDE devices and `/sys/bus/scsi/devices` for USB, SCSI or SATA devices
- calls `devMakeInode` to create new node devices using `mknod` syscall

Detecting Hardware on Fedora (2/2)

Example to find an installation CDROM



squashfs

- is a read-only filesystem that compresses both files, inodes and directories;
- designed for archivial use (LiveCD/DVD) and for embedded systems (Flash Memory);
- we can sort files into the archive according to a fixed priority.
- isn't in the mainline kernel.

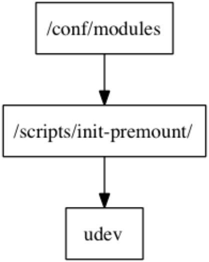
```
$ mkdir -p test/a_directory
$ touch test/a_file
$ ln -s ../a_file test/a_directory/a_link
$ mksquashfs test/ test.fs >/dev/null
$ unsquashfs -l test.fs
squashfs-root
squashfs-root/a_directory
squashfs-root/a_directory/a_link
squashfs-root/a_file

$
```

Ubuntu

- Language: `bash script`
- `udev` recognizes the hardware
- it starts a graphic interface which allows the user either to play with a LiveCD or to install the software on a hard disk

Detecting Hardware on Ubuntu



- loading modules listed in `/conf/modules`
- launches `udevd`, `udevtrigger`
- `udev` uses his rules to load modules about IDE, SCSI, MMC and populate `/dev`

Build your own distribution (1/2)

Main components:

- a script bash that builds an installation CDROM
- a configuration file which specifies the list of packages
- these packages can be fetched from a Slackware repository (official or not) or from your hard disk (personalized package)

`http://vinx.tuxfamily.org/my_distro`

Build your own distribution (2/2)

Main characteristics:

- every tool is built statically
- the tools used are: `busybox`, `e2fstools`, `util-linux`, a Linux kernel and a bootloader (`isolinux` or `GRUB`)
- hard disks are detected using the following table

Device	Path
USB	/sys/bus/usb/drivers/*nmodule*/ \ *symlink*/host[0-9]/scsi_host:host[0-9]/ \ proc_name
IDE	/sys/bus/ide/drivers/*nmodule*/ \ *symlink*/media
SCSI SATA	/sys/class/scsi_host/host[0-9]/proc_name

Common Problems: Module Not Found

- We must select the `dd` bootloader option offered by Fedora
  1. loads a *driver disk*
  2. this driver disk contains an image called `drivers.img`
  3. we can build a new `drivers.img` using the `dd` tool
- Slackware offers a shell to load manually a particular module
- Using Ubuntu, we can:
  - add the additional `break` bootloader option to load manually a particular module, or
  - using a shell in graphical interface to do the same things

Common Problems: Kernel Hangup

In some unlucky cases, the distribution kernel may hangup before offering a shell
We must rebuild a kernel to take care of the problem and create a new ISO image

- Fedora has many variants of official ISO images called *spins*, we must create a new spin using a tool such as `pungi` and add a different kernel
- The Slackware CD offers a tutorial file called `README.TXT` in `isolinux` directory that describes the steps to build a new ISO image
- Ubuntu has many tools, like `Ubuntu Customization Kit`, to create customized ISO images

Questions?!?