LPIC-1 TRAINING COURSE

Topic 102: Linux Installation and package management

Contents

Objectives

- Design a disk partition scheme for a Linux system
- Select, install and configure a boot manager
- Build and install program from source
- Determine and install the shared libraries that executable programs depend on
- Perform package management for Linux distribution that use RPMs for package distribution
- Perform package management using the Debian package manager

1. Hard Disk Layout

File System Overview

- Linux filesystem is a single tree with the / directory as its *root* directory.
- You create the single tree view of the filesystem by *mounting* the filesystems on different devices at a point in the tree called a *mount point*
- Files or subdirectories that were already in mountpoint are no longer visible when new filesystem is mounted there

Filesystem Hierarchy Standard

bin	Essential command binaries
boot	Static files of the boot loader
dev	Device files
etc	Host-specific system configuration
lib	Essential shared libraries and kernel modules
media	Mount point for removable media
mnt	Mount point for mounting a filesystem temporarily
opt	Add-on application software packages
sbin	Essential system binaries
srv	Data for services provided by this system
tmp	Temporary files
usr	Secondary hierarchy
var	Variable data

Partition

- Three types of partition on hard drives: *primary*, *logical*, and *extended*
- The *partition table* is located in the *master boot record (MBR)* of a disk
- When more than 4 partitions are required, one of the *primary* partitions must become an *extended* partition
- Linux numbers primary or extended partitions as 1 through 4
  - If logical partitions are defined, they are numbered starting at 5

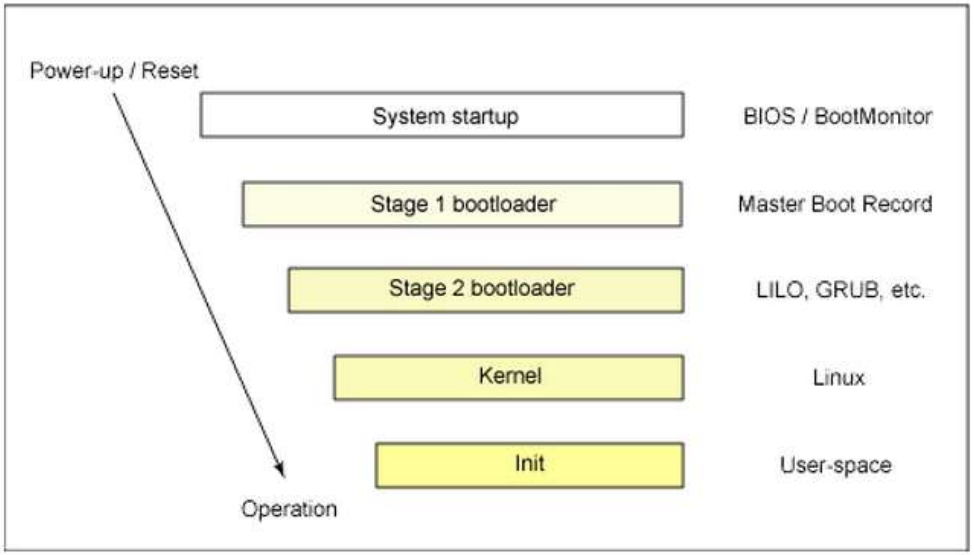
Recommended Partition Scheme

Mountpoint	Size	Description
/	4GB or more	Contains all directories not present on other filesystems
swap	2 x RAM size	used to support virtual memory
/boot	100MB	Contains the Linux kernel and boot files
/home	200MB per user	Default location for user home directories
/var	2GB or more	Contains log files and spools
/tmp	As much as possible	Holds temporary files created by programs

2. Boot manager

Linux Boot Process

- System is boot/reset: processor executes code in BIOS to determine boot device
- Boot device is found: the 1st-stage boot loader in MBR is loaded into RAM and load the 2nd-stage boot loader.
- 2nd-stage boot loader load Linux and an optional initial RAM disk (temporary root file system) into memory

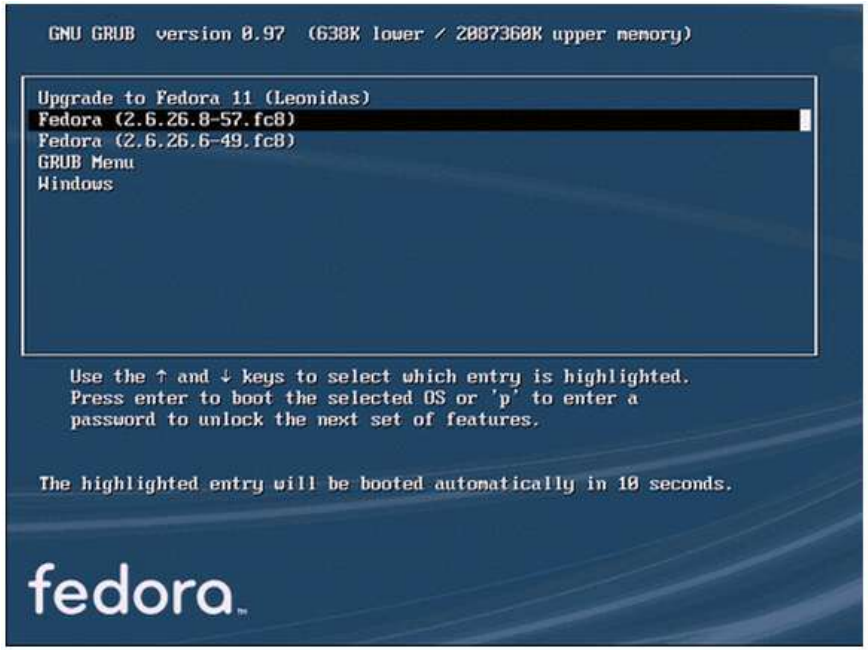


LILO

- The **LI**nux **Lo**ader
- Can install LILO into the MBR or into the partition boot record
- Configuration file: ***/etc/lilo.conf***

GRUB

- The **GR**and **U**nified **B**ootloader
- Can install GRUB into the MBR or into the partition boot record
- Configuration file: ***/boot/grub/grub.conf***



GRUB vs LILO

- LILO has no interactive command interface, whereas GRUB does.
- LILO does not support booting from a network, whereas GRUB does.
- LILO stores information on the MBR.

⇒If you change your LILO config file, you have to rewrite the LILO stage one boot loader to the MBR

⇒Much more risky option than GRUB

3. Make and install program from source

Why install from source

- A program that is not part of your distribution
- A program that is only available as source
- Need some feature of a program that is only available by rebuilding the from source
- Want to learn more about how a program works or want to participate in its development

Download and unpack

- Source can be get from sites like SourceForge.net
- Mostly distributed as compressed ***tarballs*** *(.tar.gz, .tar.Z, .tgz, .tar.bz2)*
  - *Source may be packaged for specific distro in a source package (eg: .src.rpm)*
- Check installation documentation and install required libraries & tools before start building your program

Download and unpack (cont')

- Unpacking compressed tar files
  - Two stage:
    1. Decompressing: ***gunzip*** (.tar.Z, .tar.gz, .tgz) or ***bunzip2*** (.tar.bz2)
    2. Extracting tar files:
       `tar –xvf <filename>.tar`
  - Can be done in one
    - **`tar –zxvf <filename>.tar.Z`**
      (or ***.tar.gz, .tgz)***
    - **`tar –jxvf <filename>.tar.bz2`**

Build the program

- Read a **README** or **INSTALL** file
- Find and run the **configure** script from source directory to create **Makefile**

  `./configure --help`
  - Remove **config.cache** file if you need to run **./configure** again
- Install the program
  - build the program from source: **`make`**
  - install the program you built: **`make install`**
- Remove the program
  - Read the **README** or **INSTALL** file to know how
  - Maybe **`make uninstall`** will work!

Why your build won't work?

- Missing prerequisite packages
- Wrong level of prerequisite packages
- Wrong value for some parameter that you should have passed to configure or make
- Missing compiler
- Bugs in the configure script or generated Makefile
- Source code bugs

Exercise

- Install ***netcat*** utility from source code
  1. Download netcat-0.7.1.tar.gz to ***/tmp*** directory
     - `cd /tmp`
     - `wget http://nchc.dl.sourceforge.net/project/netcat/netcat/0.7.1/netcat-0.7.1.tar.gz`
  2. Extract ***netcat*** source code
     - `tar –xvzf netcat-0.7.1.tar.gz`
  3. View installation instruction and install ***netcat***
     - `cd ./netcat-0.7.2`
     - `less README`
     - `less INSTALL`
     - `./configure`
     - `make`
     - `make install`
  4. Test ***netcat*** after installed
     - `which netcat`
     - `netcat -h`

4. Manage shared libraries

Static and dynamic executables

- Statically linked executable
  - Contain all the library that they need to execute
  - Do not depend on external library to run
  - Work without installing prerequisites
- Dynamically linked executable
  - Require functions from external shared libraries
  - Prerequisite libraries must be installed first
  - Many running programs share one copy of a library
  - Most programs to day use dynamic linking

Shared Libraries Directory

- **/lib**: main shared libraries
- **/usr/lib**: supplement libraries
- **/usr/X11R6/lib**: shared libraries for X Window
- Shared libraries's name:
  <**libraryname>-<major>-<minor>-<patch>.so**
  <**libraryname>.so** (link to the previous file)
  - Example: **libgcc_s-4.1.2.so**, **libgcc_s.so**

Managing Shared Libraries

- Viewing required shared libraries
  `ldd <filename>`
- Setting library paths
  `export LD_LIBRARY_PATH=/path/to/lib`
- Configuring shared libraries:
  - add the new directory to **/etc/ld.so.conf**
  - updating the **/etc/ld.so.cache**:
    `ldconfig`

5. Red Hat Package Manager

Package management overview

- Formalize the notion of prerequisites and versions
- Standardize file location on your system
- Provide a tracking mechanism that helps determining what packages are installed

⇒Easier software installation, maintenace and removal

rpm commands

- Options are grouped into 3 subgroups
  - Querying and verifying packages
  - Installing, upgrading and removing packages
  - Performing miscallaneous functions
- RPM is now the package management system used for packaging in the *Linux Standard Base* (LSB)

Installing RPM packages

- ***rpm*** can install package from local file systems or from internet (using *http* or *ftp*)
- Installing rpm packages:
  - **rpm –ivh </path/to/filename.rpm>**
- Forcibly installing an rpm:
  - **rpm –ivh --force </path/to/filename.rpm>**
  - **rpm –ivh --nodeps </path/to/filename.rpm>**
- Upgrading an rpm:
  - **rpm –Uvh </path/to/filename.rpm>**
  - **rpm –Fvh </path/to/filename.rpm>**

Removing RPM packages

- Removing rpm package:
  - **`rpm -e <package>`**
- Forcibly removing rpm package:
  - **`rpm -e --nodeps <package>`**

Querying RPM packages

- RPM maintains an internal database of installed packages

⇒installed packages can be manipulated using the package name

- Querying installed package:
  - **`rpm -q [-i] [-l] <package>`**
- Querying package files:
  - **`rpm -qp [-i] [-l] </path/to/filename.rpm>`**
- Querying all installed packages:
  - **`rpm -qa`**

Querying RPM packages (cont')

- Finding the owner package for a file:
  - **`rpm -qf </path/to/executable>`**
- Finding dependencies for installed package:
  - **`rpm -qR <package>`**
- Finding dependencies for package file:
  - **`rpm -qpR </path/to/filename.rpm>`**
- Querying all installed packages:
  - **`rpm -qa`**

Verifying package integrity

- Checking the integrity of package file:
  - **`rpm -Kv </path/to/filename.rpm>`**
- Verifying an installed package:
  - **`rpm -V <package>`**

Repositories and other tools

- rpm packages can be download from ditributor's *repository*
- Some tools are provided for installing packages from the repository or updating entire system
  - ***YaST (***SUSE)
  - ***up2date*** (Red Hat)
  - ***yum*** (Fedora & others)
  - ***Mandrake Software Management*** (Mandriva)
- Good resource for locating RPM: rpmfind.net

Yellowdog Updater Modified

- An interactive, automated update program for maintaining systems using rpm
- ***yum*** searches numerous *repositories* for package and their dependencies and install them together
- Allow system admin to configure a local repositories to supplement packages provided by Red Hat

Configure *yum* repositories

- Repositories information is contained in **/etc/yum.conf**

```
[repository ID]
name=repository name
baseurl=url, file or ftp://path to repository
```

- Steps to create a local repository
  - Create a repo folder: **`mkdir -p /path/to/repo`**
  - Copy all the RPMs into that directory
  - cd to that directory and run: **`createrepo`**
  - clear repo cache: **`yum clean all`**
  - Add repo information to **`/etc/yum.conf`**

yum commands

- Install the latest version of packages
  - **`yum install <package/s>`**
- Update specified packages to latest version
  - **`yum update <package/s>`**
- Remove specified packages along with any other packages that dependent on
  - **`yum remove <package/s>`**
- Find any packages containing keyword
  - **`yum search <keyword>`**

Exercise

1. Map the installation .iso file to your Virtual Machine
   - **VM -> Settings -> CD/DVD -> Use ISO imange file**:
2. Mount the installation CD to /media directory
   - `mount /dev/cdrom /media`
3. Change to **/media/CentOS** (or **/media/Server** for RedHat) and list the content of this directory
   - `cd /media/CentOS`
   - `ls –a`
4. Find and Install **createrepo-XYX.rpm**
   - `ls | grep createrepo`
   - `rpm –ivh createrepo-0.4.4-2.fc6.noarch.rpm`
5. Find and install **gcc-c++-XYX.rpm** without and then with ***--nodeps*** option
   - `ls | grep gcc-c++`
   - `rpm –ivh gcc-c++-4.1.2-14.el5.i386.rpm`
   - `rpm –ivh --nodeps gcc-c++-4.1.2-14.el5.i386.rpm`
6. Verify that **gcc-c++** is installed:
   - `rpm –qa | grep gcc-c++`
7. Remove **gcc-c++-XYZ.rpm** and verify that this package is uninstalled
   - `rpm –ev gcc-c++`
   - `rpm –qa | grep gcc-c++`

Exercise (con't)

7. Make new directory in **/tmp** for ***yum*** repository
   - `mkdir /tmp/localrepo`
8. Copy all the **.rpm** files from **/media/CentOS** (or **/media/Server** if you are using RedHat) to **/tmp/localrepo**
   - `cp /media/CentOS/*.rpm  /tmp/localrepo`
9. Run **createrepo**
   - `createrepo /tmp/localrepo`
10. (For CentOS only) Copy all the ***repo*** files in **/etc/yum.repos.d** to **/etc/yum.repos.d/backup**
    - `cd /etc/yum.repos.d`
    - `mkdir backup`
    - `mv *.repo backup`
11. Create a new file that describe you new repository in **/etc/yum.repos.d**
    - `nano localrepo.repo`

      ```
      [LocalRepo]
      name=Local Repository
      baseurl=file:///tmp/localrepo
      enabled=1
      gpgcheck=0
      ```
12. Check the **/etc/yum.conf** to ensure **gpgcheck** is disabled (**gpgcheck=0**)
    - `nano /etc/yum.conf`
13. Clear yum cache and list the content of you new repository
    - `yum clean all && yum list`
14. Reinstall gcc-c++ with yum
    - `yum install gcc-c++`

6. Debian package management

rpm vs. dpkg

rpm

- Installing package
  - ***rpm –ivh <filename.rpm>***
- Updating package
  - ***rpm –Uvh <filename.rpm>***
- Removing package
  - ***rpm –e <package>***
- List all installed packages
  - ***rpm –qa***
- Show package information
  - ***rpm –qpi <filename.rpm>***

dpkg

- Installing package
  - ***dpkg –i <filename.deb>***
- Updating package
  - ***dpkg –i <filename.deb>***
- Removing package
  - ***dpkg –r <package>***
- List all installed package
  - ***dpkg –l***
- Show package information
  - ***dpkg –l <filename.deb>***

yum vs. apt

yum

- Configuration file
  - ***/etc/yum.conf***
- Repositories information
  - ***/etc/yum.repos.d/***
- Installing package
  - ***yum install <package>***
- Updating package
  - ***yum update <package>***
- Removing package
  - ***yum remove <package>***
- Find package w/ keyword
  - ***yum search <keyword>***

apt

- Configuration file
  - ***/etc/apt/apt.conf***
- Repositories information
  - ***/etc/apt/sources.list***
- Installing package
  - ***apt-get install <package>***
- Updating package
  - ***atp-get upgrade <package>***
- Removing package
  - ***apt-get remove <package>***
- Find package w/ keyword
  - ***apt-cache search <keyword>***

SUMMARY

SUMMARY

- Understand how to allocate filesystem and swap space to the intended use of the system
- Understand boot manager role and how to install and configuring a boot loader such as GRUB
- Understand how to unpack a file of source
- Identifying shared libraries, know the typical locations of system libraries
- Installing, upgrading and removing Debian and Red Hat binary packages
- Finding packages and obtaining package information

Thank You !

BACKUP SLIDES

Partition naming on Linux

Description	Linux Name	Windows Name
First primary partition on the primary master HDD	hda1	C:
Second primary partition on the primary master HDD	hda2	D:
Third primary partition on the primary master HDD	hda3	E:
Fourth primary partition on the primary master HDD (EXTENDED)	hda4	F:
First logical drive in the extended partition on the primary master HDD	hda5	G:
Second logical drive in the extended partition on the primary master HDD	hda6	H:
Third logical drive in the extended partition on the primary master HDD	hda7	I:

Example of lilo.conf

```
image=/boot/vmlinuz-2.6.31-14-generic
        label="Lin 2.6.31-14"
        initrd=/boot/initrd.img-2.6.31-14-generic
        read-only

image=/boot/vmlinuz-2.6.31-20-generic
        label="Lin 2.6.31-20"
        initrd=/boot/initrd.img-2.6.31-20-generic
        read-only

image=/boot/memtest86+.bin
        label="Memory Test+"
        read-only

# If you have another OS on this machine (say DOS),
# you can boot if by uncommenting the following lines
# (Of course, change /dev/sda1 to wherever your DOS partition is.)
other=/dev/sda6
        label="Fedora 8"

 other=/dev/sda1
   label="Windows XP"
```

Example of grub.conf

```
default=1
timeout=10
splashimage=(hd0,5)/boot/grub/splash.xpm.gz
#hiddenmenu
password --md5 $1$RW1VW/$4XGAklxB7/GJk0uO47Srx1
title Upgrade to Fedora 11 (Leonidas)
        kernel /boot/upgrade/vmlinuz preupgrade \
      repo=hd::/var/cache/yum/preupgrade stage2=\
      hd:UUID=8b4c62e7-2022-4288-8995-5eda92cd149b:/boot/upgrade/install.img \
      ks=hd:UUID=8b4c62e7-2022-4288-8995-5eda92cd149b:/boot/upgrade/ks.cfg
        initrd /boot/upgrade/initrd.img
title Fedora (2.6.26.8-57.fc8)
        root (hd0,5)
        kernel /boot/vmlinuz-2.6.26.8-57.fc8 ro root=LABEL=FEDORA8 rhgb quiet
        initrd /boot/initrd-2.6.26.8-57.fc8.img
title Fedora (2.6.26.6-49.fc8)
        root (hd0,5)
        kernel /boot/vmlinuz-2.6.26.6-49.fc8 ro root=LABEL=FEDORA8 rhgb quiet
        initrd /boot/initrd-2.6.26.6-49.fc8.img
title GRUB Menu
        rootnoverify (hd0,1)
        chainloader +1
title Windows
        rootnoverify (hd0,0)
        chainloader +1
```

/etc/yum.repos.d/*.repo

```
[ian@echidna ~]$ cat /etc/yum.repos.d/fedora-updates.repo
[updates]
name=Fedora $releasever - $basearch - Updates
failovermethod=priority
#baseurl=http://download.fedoraproject.org/pub/fedora/linux/updates/$releasever
/$basearch/
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=updates-released-f$r
eleasever&arch=$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$basearch

[updates-debuginfo]
name=Fedora $releasever - $basearch - Updates - Debug
failovermethod=priority
#baseurl=http://download.fedoraproject.org/pub/fedora/linux/updates/$releasever
/$basearch/debug/
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=updates-released-deb
ug-f$releasever&arch=$basearch
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$basearch

[updates-source]
name=Fedora $releasever - Updates Source
failovermethod=priority
#baseurl=http://download.fedoraproject.org/pub/fedora/linux/updates/$releasever
/SRPMS/
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=updates-released-sou
rce-f$releasever&arch=$basearch
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$basearch
```

/etc/apt/sources.list

```
ian@pinguino:~$ cat /etc/apt/sources.list
#deb cdrom:[Ubuntu 9.10 _Karmic Koala_ - Release i386 (20091028.5)]/ karmic main restrict
ed
# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.

deb http://us.archive.ubuntu.com/ubuntu/ karmic main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic main restricted

## Major bug fix updates produced after the final release of the
## distribution.
deb http://us.archive.ubuntu.com/ubuntu/ karmic-updates main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic-updates main restricted

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team. Also, please note that software in universe WILL NOT receive any
## review or updates from the Ubuntu security team.
deb http://us.archive.ubuntu.com/ubuntu/ karmic universe
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic universe
deb http://us.archive.ubuntu.com/ubuntu/ karmic-updates universe
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic-updates universe
```