Nguyễn Thế Đức - Bkacad

Cài đặt phần mềm trên Linux

1. Redhat Package Manager

Được phát triển đầu tiên bởi Redhat, sau đó được các phiên bản linux khác sử dụng rộng rãi:
CentOS, Fedora, Mandrake, SuSe, Oracle Enterprise Linux
Gói RPM binary có dạng: *Tên package-phiên bản-số hiệu.kiến trúc.rpm*

. i386.rpm, .i686.rpm, .x86_64.rpm *» Kiến trúc Intel*
.sparc.rpm *» Kiến trúc Sun*
.noarch.rpm *» Không phụ thuộc vào kiến trúc*
Quản lý cài đặt package bằng trình quản lý gói *Redhat Package Manager*
- Đây là kiểu cài đặt phổ biến nhất của các bản phân phối Redhat linux
- Dễ cài đặt, dễ remove
Trình quản lý Redhat Package Manager gồm 2 thành phần quan trọng:
RPM database: lưu giữ thông tin về các gói đã được cài. Database này được lưu trong */var/lib/rpm*
rpm tool: công cụ cho người dùng cài đặt/gỡ bỏ các gói rpm và truy vấn RPM database. Người dùng gõ lệnh *rpm -tham số …* trong đó tham số để chỉ ra loại thao tác:
-i (--install), -U (--upgrade), -e (--erase), -q (--query)
Cài đặt package **rpm -ivh đường_dẫn_package**
Tham số -v (--verbose): hiển thị thông tin về gói cài
Tham số -h (--hash): hiển thị bảng phần trăm (thể hiện bằng dấu hash "#") quá trình cài đặt
Một số trường hợp lỗi

- *Package đã cài rồi : package is already installed*
- *Xung đột tập tin : …package…conflicts with…*
- *Phụ thuộc vào package khác : failed dependecies*

Một số option khác sử dụng trong cài đặt
--nodeps: cho phép cài đặt, bỏ qua các gói phụ thuộc.
--force: bắt buộc upgrade, bỏ qua conflicts.
--test: không cài đặt, upgrage, chỉ test.
--requires: liệt kê các gói phụ thuộc.
Mount **đĩa DVD CentOS 6** vào thư mục **/media**
mount /dev/cdrom /media
df -hT
Cài đặt package **rpm -ivh đường_dẫn_package**

```
# cd /media/Packages
# rpm -ivh mc-4.7.*.rpm
# ls gcc*
# rpm -ivh gcc-4.4.7-11.el6.x86_64.rpm
error: Failed dependencies:
        cloog-ppl >= 0.15 is needed by gcc-4.4.7-11.el6.x86_64
        cpp = 4.4.7-11.el6 is needed by gcc-4.4.7-11.el6.x86_64
# rpm -ivh gcc-4.4.7-11.el6.x86_64.rpm cpp-4.4.7-11.el6.x86_64.rpm cloog-ppl-0.15.7-
1.2.el6.x86_64.rpm
error: Failed dependencies:
        libmpfr.so.1()(64bit) is needed by cpp-4.4.7-11.el6.x86_64
        libppl.so.7()(64bit) is needed by cloog-ppl-0.15.7-1.2.el6.x86_64
        libppl_c.so.2()(64bit) is needed by cloog-ppl-0.15.7-1.2.el6.x86_64
```

✍ Tìm kiếm rpm http://www.rpmfind.net


```
# rpm -ivh gcc-4.4.7-11.el6.x86_64.rpm cpp-4.4.7-11.el6.x86_64.rpm cloog-ppl-0.15.7-
1.2.el6.x86_64.rpm mpfr-2.4.1-6.el6.x86_64.rpm ppl-0.10.2-11.el6.x86_64.rpm
```

Upgrade package **rpm -Uvh đường_dẫn_package**

Khi upgrade RPM sẽ xóa các phiên bản cũ của package. Có thể dùng lệnh này để cài đặt package

Truy vấn một package **rpm -qa | grep tên_package**

- *Liệt kê tất cả các phần mềm đã cài:* ***rpm -qa***
- *Lọc kết quả truy vấn qua lệnh* ***grep***

```
# rpm -qa | grep gcc
gcc-4.4.7-11.el6.x86_64
```

Gỡ bỏ package **rpm -e tên_package**

Một số trường hợp lỗi:

- *Package được xóa có liên quan đến package khác : "removing these packages would break dependecies"*

```
# rpm -e mc
```

Kiểm tra file thuộc package nào **rpm -qf đường_dẫn_tên_file**

```
# rpm -qf /etc/passwd
setup-2.8.14-20.el6_4.1.noarch
```

```
# which setup
# rpm -qf /usr/sbin/setup
setuptool-1.19.9-4.el6
# rpm -qi setup
```

-qa: hiển thị danh sách các package đã cài đặt.
-qf <file_name>: hiển thị package sở hữu <file_name>
-qd <package_name>: hiển thị danh sách tập tin tài liệu
-qi <package_name>: hiển thị thông tin của package
-ql <package_name>: hiển thị file chứa trong package_name
-qc <package_name>: hiển thị danh sách tập tin cấu hình

Phần mềm dạng RPM source

*Gói phần mềm có phần mở rộng tên file: **.src.rpm** » Phần mềm dạng RPM source*
Dịch RPM source sang RPM binary rồi cài đặt

2. Yum Package Manager

yum *(yellowdog updater modified)* là một công cụ quản lý và cài đặt phần mềm cho các hệ thống RedHat Linux

Mặc định **yum** tìm kiếm và tải về các gói nằm trong các repo server, việc cài đặt phần mềm với **yum** yêu cầu phải có kết nối Internet tốc độ cao để việc tải gói về được nhanh chóng.

Repository (gọi tắt là repo) là hệ thống kho phần mềm của Linux được lưu trữ trên mạng Internet thông qua hệ thống repo server cho phép các máy khác lấy file về cài đặt

Trong CentOS (và các dòng Redhat) thông tin về repo server được lưu trong các file **.repo** của thư mục **/etc/yum.repos.d**

```
# ls -l /etc/yum.repos.d
```

File **.repo** chứa thông tin mô tả về đường dẫn đến các server repo. Để thêm vào một repo mới, cần tạo file **.repo** trong thư mục **/etc/yum.repos.d** hoặc cài đặt gói rpm để tạo thêm repo mới.

Hiển thị các repo

```
# yum repolist
# yum repolist all
```

File cấu hình yum **/etc/yum.conf**

```
# vim /etc/yum.conf
[main]
cachedir=/var/cache/yum
keepcache=1
```

- cachedir: Thư mục chứa cache
*- keepcache=0: Xóa package download sau khi cài đặt, 1: package download lưu trong **/var/cache/yum/<repo-name>/packages/***

Xóa cache yum

```
# yum clean all
# ls -l /var/cache/yum
```

Add repo EPEL

EPEL là từ viết tắt của *Extra Packages for Enterprise Linux* do 1 nhóm người dùng Fedora (*Fedora Special Interest Group*) tạo ra, duy trì, và quản lý các phần mềm cho các bản phân phối Enterprise Linux như *Red Hat Enterprise Linux* (RHEL), CentOS và *Scientific Linux* (SL). Repo EPEL có hơn 8000 gói phần mềm và có thể sử dụng trên các bảng phân phối dựa trên Enterprise Linux như Fedora, RHEL, CentOS, Scientific Linux và không gây xung đột với các phần mềm khác trên các bản phân phối này.EPEL là 1 repo mở, có thể được dùng bởi bất cứ người dùng Linux nào để cài đặt các phần mềm tồn tại trên Fedora và không có trên phiên bản RHEL.

```
# yum install epel-release -y
# cat /etc/yum.repos.d/epel.repo
```

Add repo rpmforge

RHEL/CentOS 6 32-Bit

```
# wget http://apt.sw.be/redhat/el6/en/i386/rpmforge/RPMS/rpmforge-release-0.5.3-1.el6.rf.i686.rpm
```

RHEL/CentOS 6 64-Bit

```
# wget http://apt.sw.be/redhat/el6/en/x86_64/rpmforge/RPMS/rpmforge-release-0.5.3-1.el6.rf.x86_64.rpm
# rpm -ivh rpmforge-release-*.rpm
```

Quản lý và cài đặt package với Yum Package Manager

Cài đặt gói phần mềm **yum install package-name**

Mặc định khi cài đặt bằng lệnh **yum**, lệnh yêu cầu xác nhận cài đặt package, để bỏ qua bước này chỉ cần thêm option **-y**

Cài đặt phần mềm remote desktop

```
# yum install remmina remmina-plugins-rdp remmina-plugins-vnc remmina-plugins-xdmcp -y
```

Cài đặt phần mềm monitor hệ thống

```
# yum install htop -y
# yum install python-pip python-devel python-psutil python-setuptools gcc -y
# pip install Glances supervisor
# pip list | grep Glances
# glances
```

Glances Color Codes

GREEN: OK (everything is fine)

BLUE: CAREFUL (need attention)

VIOLET: WARNING (alert)

RED: CRITICAL (critical)

Cài đặt phần mềm web console

yum install w3m -y

Cài đặt phần mềm so sánh file

yum install meld -y

Cài đặt tiện ích yum

yum install yum-utils -y

Download package nhưng không cài đặt **yumdownloader <package>**

yumdownloader gcc

Cài đặt theo repo chỉ định **yum --enablerepo=repo-name install package-name**

Cập nhật package mới **yum update package-name**

Cập nhật Kernel

yum update kernel -y

Gỡ bỏ gói phần mềm **yum remove package-name**

yum remove w3m -y

Tìm một phần mềm **yum search package-name**

Liệt kê tất cả các package có trong **repo yum list package-name**

yum list w3m

Hiển thị thông tin cài đặt package

yum info w3m

Cài đặt giao diện đồ họa GNOME - KDE

Kiểm tra cài đặt các group *X Window System*, *GNOME* và *KDE*

yum grouplist

yum grouplist | grep -i --color gnome

Kiểm tra thông tin chi tiết các package trong group

yum groupinfo 'X Window System'

Cài đặt X Window System nếu chưa được cài đặt

CentOS 5

yum groupinstall 'X Window System' -y

yum groupinstall 'GNOME Desktop Environment' -y

yum groupinstall 'KDE (K Desktop Environment)' -y

Chọn loại giao diện đồ họa Linux khởi động mặc định

vim /etc/sysconfig/desktop

DESKTOP="KDE"

DISPLAYMANAGER="KDE"

Cấu hình KDE cho phép user **root** login

find / -name "kdmrc"

/usr/share/config/kdm/kdmrc

vim /usr/share/config/kdm/kdmrc

AllowRootLogin=true

Cài đặt tiện ích ***switchdesk***

yum install switchdesk -y

Chuyển đổi KDE và GNOME

switchdesk kde

switchdesk gnome

CentOS 6

yum groupinstall 'X Window System' -y

yum groupinstall 'Desktop' 'Desktop Platform' -y

yum groupinstall 'KDE Desktop' -y

yum groupinstall 'Fonts' -y

Cấu hình KDE cho phép user **root** login

find / -name "kdmrc"

/etc/kde/kdm/kdmrc

vim /etc/kde/kdm/kdmrc

AllowRootLogin=true

✍ Tìm kiếm package http://pkgs.org

Kiểm tra log YUM

tail -f /var/log/yum.log

Bài tập: Cấu hình **yum package manager** sử dụng **local repository**

1. *Mount* đĩa cài **Linux** *lên thư mục* **/media**
   # mount /dev/cdrom /media
2. *Cài đặt gói* **createrepo-*.rpm**
   # cd /media/Packages
   # rpm -ivh createrepo-*.rpm python-deltarpm-*.rpm deltarpm-*.rpm
3. *Copy file****.rpm*** *từ* **/media/Packages** *vào thư mục* **/localrepo**
   # mkdir /localrepo
   # cp /media/Packages/mc-*.rpm /localrepo
4. *Chạy câu lệnh:* **createrepo /localrepo** *để chuẩn bị repository*
   # createrepo /localrepo
5. *Move tất cả file có đuôi****.repo*** *trong thư mục* ***/etc/yum.repos.d*** *sang một thư mục khác*
   # mkdir /etc/yum.repos.d/backup
   # mv /etc/yum.repos.d/*.repo /etc/yum.repos.d/backup/
6. *Tạo một file mới tên là* **local.repo** *trong thư mục* **/etc/yum.repos.d** *và đưa các thông tin mô tả về repository này vào file đó*
   # vim /etc/yum.repos.d/local.repo
   [LocalRepo]
   name=Local Repository
   baseurl=file:///localrepo

```
enabled=1
gpgcheck=0
# yum clean all
# yum install mc -y
```

3. Cài đặt phần mềm từ mã nguồn mở

Trình biên dịch GNU C Compiler (GCC)

- gcc/ g++ Trình biên dịch ngôn ngữ C/C++
- Trình biên dịch *gcc* thường được đặt trong **/usr/bin**. Khi biên dịch, *gcc* cần nhiều file hỗ trợ như các tập tin thư viện liên kết trong **/usr/lib**, các file C header trong **/usr/include**...
- Chương trình sau khi biên dịch có thể đặt bất kỳ đâu trên hệ thống sao cho Hệ điều hành có thể tìm thấy trong biến môi trường **PATH** hoặc theo đường dẫn tuyệt đối trong dòng lệnh.

Lưu ý khi biên dịch trong Linux

- *Dùng g++ nếu chương trình có chứa mã C lẫn C++*
- *Dùng gcc nếu chương trình chỉ có mã C*
- *File thực thi tạo ra không có đuôi .exe, .dll như môi trường Windows*

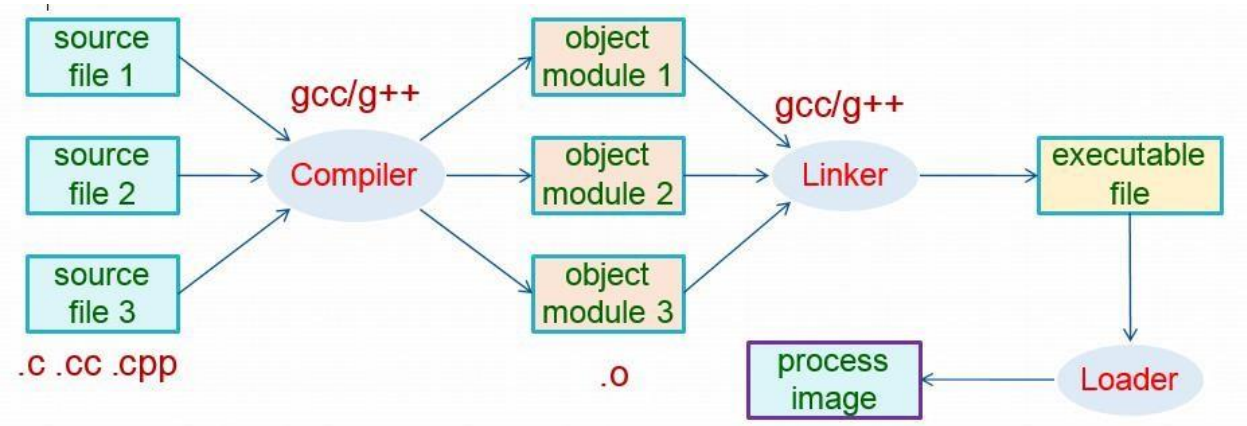
Cú pháp: gcc [options] sources

Tùy chọn

-c: sinh ra tập tin đối tượng .o

-o: sinh ra tập tin thực thi

gcc -v



Ví dụ ngôn ngữ C

Soạn thảo 2 files: main.c và func.c

vim func.c

```
/* func.c */
#include <stdio.h>
void hello(){
  printf("Hello World\n");
}
```


```
# vim main.c
/* main.c */
main(){
hello();
}
```

Biên dịch và chạy

```
# gcc -c func.c main.c
# gcc -o main main.o func.o
# ./main
Hello World
```

Ví dụ ngôn ngữ C++

```
# vim hello.cpp
#include  <iostream>
using namespace std;
int main()
{
  cout << "Hello World \n"; // prints Hello World
  return 0;
}
# g++ -c hello.cpp
# g++ -o main hello.o
# ./main
Hello World
```

Công cụ hỗ trợ biên dịch make và Makefile

Makefile là một file đặc biệt dạng script chứa các thông tin

- *Cấu trúc project (file, sự phụ thuộc)*
- *Chứa các quy tắc biên dịch*

Lệnh make sẽ đọc các bước dịch trong Makefile**,** hiểu kiến trúc của project và thực thi các lệnh

```
# vim Makefile
# Makefile
CC = gcc
main: main.o func.o
        $(CC) -o main main.o func.o
main.o: main.c
        $(CC) -c main.c
func.o: func.c
        $(CC) -c func.c
# make
```

✍ Copy nội dung Makefile trên vào trình soạn thảo Vi có thể gặp lỗi biên dịch sau:
"*Makefile:4: *** missing separator (did you mean TAB instead of 8 spaces?). Stop.*"
Thay 8 spaces bằng ***phím Tab*** các dòng sau rồi biên dịch lại bằng lệnh **make**:

```
$(CC) -o main main.o func.o
$(CC) -c main.c
$(CC) -c func.c
```

```
# ./main
Hello World
```

Cài đặt phần mềm từ source

Cài phần mềm từ mã nguồn tương thích với mọi phiên bản Linux, được đóng gói sử dụng kiểu GNU Zip (.gz) hoặc BZip2 (.bz2)
Giải nén bằng lệnh **tar**

```
tar -xvf <filename>.tar.gz
tar -xvf <filename>.tar.bz2
tar -xvf <filename>.tar.bz2 -C /tmp
```

-x: bung (extract) file lưu trữ
-v: xem quá trình extract
-C: Thư mục extract
Đọc file INSTALL hoặc README để có những chỉ dẫn riêng biệt của gói cài đặt.
Sau khi giải nén, chuyển đến thư mục của gói source:

```
cd <extracted_dir_name>
```

Cấu hình phần mềm: chạy script *./configure* kiểm tra hệ thống, thư viện cần thiết và các gói phụ thuộc nào cần để cài đặt phần mềm, nếu không có lỗi nào *./configure* sẽ tạo ra *Makefile* để hướng dẫn biên dịch mã nguồn của gói ra dạng thực thi
Đọc file README, INSTALL để có những tham số lựa chọn cần thiết dùng cho lệnh *./configure*

```
Type "./configure --help" for more information.
Remove config.cache file if you need to run ./configure again
# ./configure
```

Biên dịch mã nguồn bằng lệnh *make*

```
# make
```

Cài đặt vào hệ thống

```
# make install
```

Lệnh *make install* sẽ copy các file thực thi vào đúng vị trí của nó trên hệ thống. File thực thi thường được cài đặt vào thư mục */usr/local/bin*
Khi có thay đổi trong source, cần biên dịch và cài đặt lại
Sau khi cài đặt xong, để gỡ bỏ gói source, dùng lệnh *make uninstall*
Nếu cần thiết xóa bỏ luôn thư mục source cài đặt rm -rf <extracted_dir_name>

Cài đặt bash shell

Cài đặt công cụ biên dịch **gcc** và **patch** tool

```
# yum install gcc gcc-c++ patch -y
```

Download mã nguồn bash mới nhất và các bản vá lỗi (Patch File) từ http://ftp.gnu.org/gnu/bash

```
# cd /usr/local/src
# wget http://ftp.gnu.org/gnu/bash/bash-4.3.tar.gz
# tar -xvf bash-4.3.tar.gz
# cd bash-4.3
# ./configure && make && make install
# which bash
/usr/local/bin/bash
# echo $PATH
.:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
# bash -version
GNU bash, version 4.1.2(1)-release
# bash
# bash -version
GNU bash, version 4.3.0(1)-release
# env x='() { :;}; echo vulnerable' bash -c "echo this is a test"
vulnerable
this is a test
```

Patch source cho **bash** và cài đặt lại

```
# cd /usr/local/src/bash-4.3
# for i in $(seq -f "%03g" 0 33); do wget http://ftp.gnu.org/gnu/bash/bash-4.3-patches/bash43-$i; done
# for i in $(seq -f "%03g" 0 33);do patch -p0 < ./bash43-$i; done
# ./configure && make && make install
# bash -version
GNU bash, version 4.3.33(2)-release
# env x='() { :;}; echo vulnerable' bash -c "echo this is a test"
this is a test
```

✍Kiểm tra **shellshock** bằng lệnh env x='() { :;}; echo vulnerable' bash -c "echo this is a test"

Nếu kết quả trả về:

vulnerable

this is a test

Thì bash chưa được fix shellshock

Nếu kết quả là *this is a test*, thì bash đã được fix.

Cài đặt bộ gõ tiếng việt xvnkb

Cài đặt group **Libraries**

CentOS 5

```
# yum groupinstall 'Development Libraries' -y
```

CentOS 6

```
# yum groupinstall 'Desktop Platform Development' -y
```

Địa chỉ xvnkb: http://xvnkb.sourceforge.net/

Tải source **xvnkb** về

```
# wget http://ncu.dl.sourceforge.net/project/xvnkb/xvnkb/0.2.11/xvnkb-0.2.11.tar.gz
```

Giải nén và cài đặt

```
# tar -xvf xvnkb-0.2.11.tar.gz
# cd xvnkb-0.2.11
# ./configure
# make
# make install
```

Chuyển vào giao diện đồ họa X Windows, mở Terminal và chạy *xvnkb*

```
# startx
# xvnkb
```

Tạo link xvnkb ra Desktop

Ấn chuột phải Desktop và chọn : "Create Launcher..."

→ chọn **Command**: "xvnkb" → OK

Cài đặt font chữ unicode windows

Copy font ***unicode:*** *arial.ttf, tahoma.ttf, times.ttf, verdana.ttf* từ C:\Windows\Fonts đến thư mục /usr/share/fonts/default/Type1

```
# cd /usr/share/fonts/default/Type1
# ls *.ttf
arial.ttf  tahoma.ttf  times.ttf  verdana.ttf
```

Cập nhật font chữ

```
# fc-cache -fv
```

Mở **Open Office** hoặc **Libre Office** kiểm tra font unicode và sử dụng **xvnkb** để gõ tiếng việt

Quản lý thư viện

Thư viện là file chứa các đoạn mã lệnh và dữ liệu được tổ chức thành các hàm (*function*), các lớp (*class*) nhằm cung cấp dịch vụ, chức năng nào đó cho các chương trình chạy trên máy tính.

Thư viện gồm 3 loại: **Static, Dynamic và Shared**. Thường thì các thư viện ở dạng mã nhị phân, không phải dạng văn bản thuần túy (*plain text*)

Khi biên dịch 1 chương trình đang ở dạng *source code* (gồm tập các câu lệnh, khai báo được viết bằng 1 ngôn ngữ lập trình cấp cao như C/C++, Java…) sang dạng *executable* (hay binary - tập các mã máy nhị phân mà chỉ CPU mới hiểu được) thì nhiều hàm chức năng của chương trình được liên kết từ các thư viện. Quá trình biên dịch này do bộ biên dịch (*Compiler*) đảm nhiệm.

Các thư viện được liên kết động được dùng chung bởi nhiều ứng dụng được gọi là **shared library**. Trên Windows các file thư viện có phần mở rộng là ***.dll*** , còn Linux các file thư viện có phần mở rộng ***.a***, ***.so***, ***.sa*** và được bắt đầu bằng tiếp đầu ngữ "*lib*": *libutil.a* hoặc *libc.so…*
File thư viện ***.so*** được ưu tiên hơn file ***.a***, nếu không tìm thấy file ***.so*** thì file ***.a*** mới được dùng
Khi biên dịch, thông thường chương trình liên kết (*ld*) sẽ tìm thư viện trong 2 thư mục chuẩn ***/usr/lib*** và ***/lib***
Các hệ thống Linux có hai dạng chương trình thực thi

- *Chương trình thi hành được* liên kết tĩnh *(statically linked) chứa tất cả các hàm thư viện mà chúng cần để thi hành. Chúng là chương trình đầy đủ, chạy không phụ thuộc vào thư viện bên ngoài. Một lợi thế của các chương trình liên kết tĩnh là chúng sẽ làm việc mà không cần cài đặt các điều kiện cần có trước.*
- *Chương trình thi hành được* liên kết động *(dynamically linked) là các chương trình nhỏ hơn nhiều, chưa đầy đủ và cần các hàm từ các thư viện chia sẻ bên ngoài để chạy. Các chương trình liên kết động cho phép một gói chỉ rõ các thư viện cần có trước mà không cần phải chứa sẵn các thư viện trong gói. Việc sử dụng liên kết động cũng cho phép nhiều chương trình đang chạy chia sẻ một bản sao của một thư viện thay vì chiếm bộ nhớ để chứa nhiều bản sao của cùng một mã. Với những lý do này, hầu hết các chương trình hiện nay sử dụng liên kết động.*

Tổ chức các file thực thi trên Linux
- Thư mục **/usr** chứa các chương trình và thư viện
- Thư mục **/usr/bin** chứa các file thực thi các gói đã cài đặt như: *firefox, gedit ...*
- Thư mục **/usr/lib** chứa các files có phần mở rộng là .so *(shared object)* là các hàm thư viện liên kết động hoặc .a *(archive)* là các hàm thư viện liên kết tĩnh. Đặc tính căn bản của 2 dạng thư viện này là hàm thư viện liên kết tĩnh sẽ được liên kết thẳng với files thực thi luôn trong quá trình liên kết, còn hàm thư viện liên kết động thì sẽ được liên kết trong quá trình thực thi, vì vậy sau khi chương trình được biên dịch và liên kết, các thư viện liên kết tĩnh có thể bỏ đi nhưng thư viện liên kết động thì bắt buộc phải đi kèm với chương trình.
- Thư mục **/usr/share** chứa các icon, manual hoặc info của gói

Ví dụ: lệnh ***ln*** tạo ra các liên kết giữa các tập tin. Lệnh *ln* sử dụng các thư viện chia sẻ, do đó nếu các thư viện liên kết động không làm việc, thì lệnh *ln* không thể chạy. Để tránh trường hợp này, hệ thống Linux có một phiên bản liên kết tĩnh của *ln* là lệnh ***sln***

```
# ls -lh /sbin/sln /bin/ln
-rwxr-xr-x. 1 root root  45K Th10 15 11:51 /bin/ln
-rwxr-xr-x. 1 root root 671K Th10 15 09:00 /sbin/sln
```

Quản lý shared library

Lệnh **ldd** *(List Dynamic Dependencies)* hiển thị thông tin về các thư viện cần thiết của một chương trình thi hành được

```
# ldd /sbin/sln /bin/ln
/sbin/sln:
        not a dynamic executable
/bin/ln:
        libc.so.6 =>
```

Lệnh ldconfig xử lý tất cả các thư viện trong ld.so.conf và trong thư mục /lib, /usr/lib

ldconfig [OPTION]

Tùy chọn

***-p**: hiển thị nội dung hiện tại của cache, không tạo lại cache*

***-v**: hiển thị quá trình thực hiện tạo lại cache*

Lệnh ldconfig để hiển thị *ld.so.cache*

```
# ldconfig -p | less
# ldconfig -p | grep /usr/lib | less
# cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
```

Ví dụ: Quản lý thư viện liên kết

Kiểm tra thư viện liên kết với ldd <filename>

```
# ldd /usr/bin/top
        linux-vdso.so.1 =>  (0x00007fffe237e000)
        libproc-3.2.8.so => /lib64/libproc-3.2.8.so (0x00007fc84458c000)
        libncursesw.so.5 => /lib64/libncursesw.so.5 (0x00007fc84435d000)
        libc.so.6 => /lib64/libc.so.6 (0x00007fc843fc8000)
        libtinfo.so.5 => /lib64/libtinfo.so.5 (0x00007fc843da7000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007fc843ba3000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fc8447bf000)

# mv /lib64/libproc-3.2.8.so  /tmp
# top
top: error while loading shared libraries: libproc-3.2.8.so: cannot open shared object file: No such file or directory
```

Lỗi không tìm thấy thư viện: ***error while loading shared libraries: .... cannot open shared object file: no such file or directory***

Cài đặt đường dẫn thư viện với biến môi trường LD_LIBRARY_PATH=/path/to/lib

```
# export LD_LIBRARY_PATH=/tmp
# top
```

Cấu hình thư viện liên kết với **ldconfig**

unset LD_LIBRARY_PATH

echo $LD_LIBRARY_PATH

top

top: error while loading shared libraries: libproc-3.2.7.so: cannot open shared object file: No such file or directory

Thêm đường dẫn thư viện mới vào /etc/ld.so.conf cập nhật /etc/ld.so.cache

vim /etc/ld.so.conf

include ld.so.conf.d/.conf*

/tmp

ldconfig -p | grep /tmp

ldconfig -v

ldconfig -p | grep /tmp

libproc-3.2.8.so (libc6,x86-64) => /tmp/libproc-3.2.8.so

top