Linux Basic 2

2013/6

Linux Popular Release

- Redhat

  Phase1 : Redhat 5.x~9.x

  Phase2 : Redhat Enterprise Linux (RHEL)

  Desktop WS Server AS DataCenter ...

  Free Edition : Fedora

  Academic Release Supported by top-research lab, from RHEL AS

  CentOS ScientificLinux

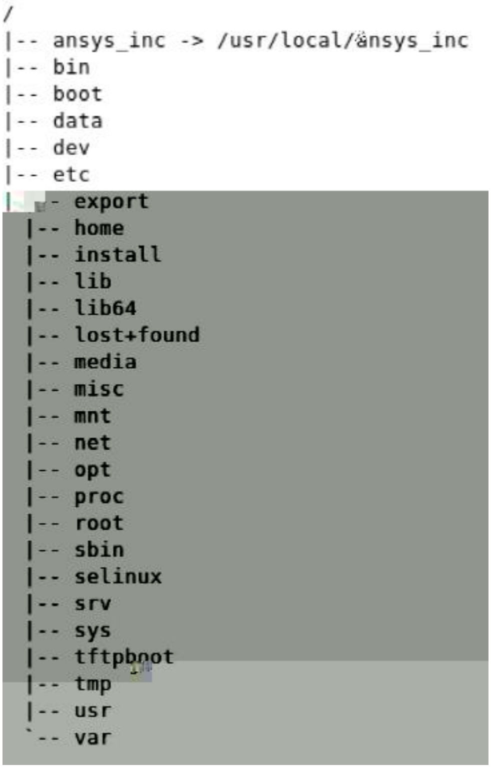
- Novell SuSE

  Business : SLED SLES

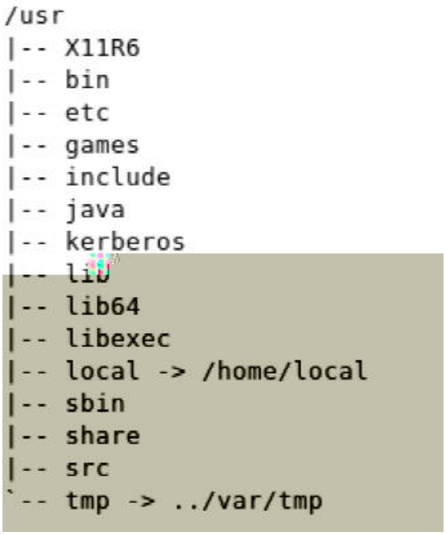
  Free Editon : OpenSuSE

- Ubuntu,

Typical Directory Tree



```
/
|-- ansys_inc -> /usr/local/ansys_inc
|-- bin
|-- boot
|-- data
|-- dev
|-- etc
|-- export
|-- home
|-- install
|-- lib
|-- lib64
|-- lost+found
|-- media
|-- misc
|-- mnt
|-- net
|-- opt
|-- proc
|-- root
|-- sbin
|-- selinux
|-- srv
|-- sys
|-- tftpboot
|-- tmp
|-- usr
`-- var
```



```
/usr
|-- X11R6
|-- bin
|-- etc
|-- games
|-- include
|-- java
|-- kerberos
|-- lib
|-- lib64
|-- libexec
|-- local -> /home/local
|-- sbin
|-- share
|-- src
`-- tmp -> ../var/tmp
```

Typical Directory Tree

The root directory (/), as was discussed previously, is primarily used to hold all other directories. It is bad karma to store any file in the root (other than what Linux stores there).

The /bin directory stores binary executable files (programs). The name bin is derived from binary. Only Linux system binaries should be stored in this directory.

The /dev directory holds the files that refer to devices. If you recall from the previous section, everything in Linux is a file, and devices (such as a printer) are no exception.

The /etc directory holds Linux-specific configuration files.

The /home directory contains the home directories for users known to the system. When you log in to the system, you are taken to your home directory, which is found under /home.

The /lib or /lib64 directory is used to hold shared library files. These shared library files contain common function routines used by programs. Library files are referred to as shared because more than one program can access routines found within them. This fact keeps most programs small (and the system smaller) because each program does not have to store those routines.

Typical Directory Tree

The /proc directory holds process and kernel runtime information. The information is actually in memory but is accessed through /proc.

The /tmp directory, as you may have guessed, stores temporary files. Most of these temporary files are created by running processes. It is a good idea to visit this directory from time to time to see if any (large) files are left lingering around. The best time to do this is just after logging in to the system.

The /usr directory is used to contain most of the software packages that you install. This directory contains other directories, such as /usr/bin, /usr/etc, /usr/lib or /usr/lib64, /usr/local, /usr/man, and /usr/src. Let's take a look at these directories. Executables are stored in /usr/bin (the same as /bin does). Various configuration files not found in /etc are stored in /usr/etc - mainly configuration files used by the installed software packages.

Typical Directory Tree

The /usr/lib or /usr/lib64 directory stores shared library files for the software packages.

The man pages (help files) are stored in /usr/man. The /usr/man directory will also contain a number of directories.

Source code for software can be found in /usr/src. The size of this directory can be quite large if you opt to install source code for all the software packages.

The /usr/local directory is used for nonessential files and programs. The structure of /usr/local will normally be different between UNIX systems. As a rule, however, it will contain /usr/local/bin, /usr/local/etc, and /usr/local/lib.

Files that fluctuate in size can be found in /var. The /var directory typically contains two directories: /var/adm and /var/spool.

The /var/adm directory contains system error messages and log files. These files are reviewed and maintained by the system administrator. The /var/spool directory contains files that are used by programs such as news and mail.

Login ,Exit, Shutdown

- ssh client
- shutdown

```
shutdown –h now
```

- exit

```
exit
```

Logon , Exit, Shutdown

ssh client

xftp
— download or upload
  files from server
— support ssh and ftp, etc.

ssh
— login server.

Some Basic Commands

- Current Working Directory
- The Home Directory
- Viewing a Directory's Contents
- Moving Around in the Linux Directory System
- Creating New Directories
- Deleting Files and Directories
- Viewing Files
- Copying Files
- Moving Files
- Getting Help - The Man Pages

Some Basic Commands

- Current Working Directory

The pwd command can be used to print the current working directory as a full pathname.

```
$ pwd
```

You can change the current working directory using the cd command.

```
$ cd /bin
```

Some Basic Commands

- ## The Home Directory

  Every user known to the system has a home directory. Optionally, you can use the tilde character (~) to specify your home directory. Peter tells you that the file is named myfile.c and is found in her home directory (peter). The following is a dialog to use to copy that file to your home directory.

  ```
  $ cp  ~peter/myfile.c  ~
  ```

Some Basic Commands

- Moving Around in the Linux Directory System

```
cd  [dir]
stimpy $ cd  ..
stimpy $ cd  ~
stimpy $ cd  /
stimpy $ cd
stimpy $ cd  /
stimpy $ cd  ../home/mtobler
```

Some Basic Commands

- Creating New Directories

  mkdir [options] directories

```
$ mkdir database documents source
$ ls
data
docs
source
```

...

Some Basic Commands

- Deleting Files and Directories

```
rm   [options]   files
```

```
$ rm *ity
```

all files that end with ity will be removed.

A second method of deleting files exists. deleting a file using /dev/null:

```
$ mv  myfile.txt   /dev/null
```

The syntax for rmdir is as follows:

```
rmdir   [options]   directories
rm  -rf  docs
```

Some Basic Commands

Some Basic Commands

- Copying Files

```
cp  [options]  file1  file2
cp  [options]  files  directory
cp  -r  dir1   dir2
```

- Moving Files

```
mv  [options]  source  target
$mv names.txt ..
$mv names.txt names
```

- Getting Help - The Man Pages

```
man  [options]   [section]   [title]
```

Some Basic Commands

- Check Dir Space Usage

```
du [options] [file]
du -sh docs
```

- Check Disk Space Usage

```
df [options] [file]
df
```

Network

- Show/Manipulate IP route table

  route [options]

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
202.119.112.128 *               255.255.255.128 U     0      0        0 eth1
192.168.9.0     *               255.255.255.0   U     0      0        0 eth0
192.168.90.0    *               255.255.255.0   U     0      0        0 ib0
link-local      *               255.255.0.0     U     1003   0        0 eth0
link-local      *               255.255.0.0     U     1004   0        0 eth1
link-local      *               255.255.0.0     U     1005   0        0 ib0
default         202.119.112.254 0.0.0.0         UG    0      0        0 eth1
```

Network

- Configure a network interface

```
ifconfig [interface]
ifconfig interface options| address
```

```
$ /sbin/ifconfig
```

User & Group

- Add New User

  useradd   user1

- Modify Password

  passwd user1
  mypass

- Delete An User

  userdel   user1

- Add New Group

  groupadd   grp1

- Delete Group

  groupdel   grp1

- whoami
- who

RPM Package

- Query which rpm package the command belongs to

```
rpm   -qf     /bin/hostname
```

- Show the rpm package information

```
rpm    -qpi      /export/home/jointforce/rpm-helper-0.9.1-4sls.noarch.rpm
```

- Install new package

```
rpm      –i   xxxx.rpm
rpm      -Uvh    xxxx.rpm
rpm      -ivh      xxxx.rpm
```

- Delete package

```
rpm   –e   xxxx
```

YUM tools

- Install a package

  ```
  yum install -y libstdc++
  ```

- Remove a package

  ```
  yum remove libstdc++
  ```

Unpack/Pack TAR Ball

```
tar  -zxvf  lammps.tar.gz
tar  -zcvf  Si.tar.gz   Si_case/
tar  -jxvf  lammps.bz2
```

Login cluster nodes

- Login computing nodes and other nodes

```
rsh   node1
ssh   node1
```

- Remote copy

```
scp  -r user1@server1:~/data   ./
```

Process

- Looking at Processes

  Even as you sit down at your computer, there are processes running. Every executing program uses one or more processes. Each process in a Linux system is identified by its unique process ID, sometimes referred to as pid.
- top
- ps

  The ps command displays the processes that are running on your system.

  ```
  $ ps   -aux  | grep  xxxx
  $ ps   -ef
  ```
- kill (kill -9 pid, kill)
- Ctrl-C Ctrl-Z
- xxx&
- nohup   xxx&

Developing Toolkits

- GNU (free)

Compiler	gcc/g++, f77/gfortran
debug	gdb
IDE	kdevelop

- Pgi group (business)

  pgf90/pgcc

- Intel (non-commercial Edition)

  Intel Compiler (c/c++, fortran, MKL)

Getting Started

- Compiling a source file

The -c option tells gcc to compile the program to an object file only; The -I option is used to tell GCC where to search for header files. By default, GCC looks in the current directory and in the directories where headers for the standard libraries are installed. If you need to include header files from somewhere else, you'll need the -I option.

Sometimes you'll want to define macros on the command line. It's easier to simply define NDEBUG on the command line, like this:

```
$ g++   -c   -D   NDEBUG   myfile.cpp
$ g++   -c   -D   NDEBUG=2   myfile.cpp
$ g++   -c   -O2    myfile.cpp
```

Getting Started

linking Object Files

The -o option gives the name of the file to generate as output from the link step. If you had needed to link in another library (such as a graphical user interface toolkit), you would have specified the library with the -l option. In Linux, library names almost always start with lib. To link in libpam.a . As with header files, the linker looks for libraries in some standard places, including the /lib and /usr/lib directories that contain the standard system libraries. If you want the linker to search other directories as well, you should use the -L option,

```
$ g++   -o  myfile  myfile.o   -lpam
$ g++   -o  myfile  -D   NDEBUG   myfile.cpp
```

You can use this line to instruct the linker to look for libraries in the /usr/local/lib/pam directory before looking in the usual places:

```
$ g++ -o  myfile  myfile.o   -L/usr/local/lib/pam    -lpam
```

Intel Compiler

- Intel Compiler

  icc/ifort -o prog -O3 -xSSE4.2 prog.c

  Our Cluster   -O3 -xSSE4.2

```
$ head  -24 /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 26
model name      : Intel(R) Xeon(R) CPU           X5560  @ 2.80GHz
stepping        : 5
cpu MHz         : 1596.000
cache size      : 8192 KB
physical id     : 0
siblings        : 4
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 11
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp
             lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf pni dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm
             dca sse4_1 sse4_2 popcnt lahf_lm ida dts tpr_shadow vnmi flexpriority ept vpid
bogomips        : 5601.15
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
```

Getting Started

- Automaking the Process with GNU Make

You can see that targets are listed on the left, followed by a colon and then any dependencies. The rule to build that target is on the next line. The line with the rule on it must start with a Tab character, or make will get confused. The $(CFLAGS) is a make variable.You can define this variable either in the Makefile itself or on the command line. GNU make will substitute the value of the variable when it executes the rule.

```
$make CFLAGS=-O2
```

MPI

- Compile

```
/usr/local/mvapi2/bin/mpif90  -o mpiprog  -O3 xSSE4.2  progf.f90
/usr/local/openmpi1.6.2/bin/mpif90  -o mpiprog  -O3 xSSE4.2  progf.f90
```

- Deploy/Run

```
/usr/local/mvapi2/bin/mpiexec  -launcher rsh –n 24  -f hostfile  ./mpiprog
/usr/local/openmpi1.6.2/bin/mpirun  --mca btl openib,self --mca orte_rsh_agent rsh –np 24  -hostfile  hostfile  ./mpiprog
```

Job Submit

- Submit Job

```
qsub  job.sh
```

- Check Job State

```
qstat
```

- Delete a Job

```
qdel
```

Matlab

- matlab.sh

```
#!/bin/sh
#
#$ -S /bin/sh
#$ -N  mjob3              Job Name
#$ -j y
#$ -o ./
#$ -e ./
#$ -cwd
#$ -q short.q

source ~/.bash_profile
#source ~/.bashrc
hash -r
export path=$TMPDIR:$path

#      drive.m  --- input file       mat.out   ---   stdout  message
/usr/local/Matlab2010a/bin/matlab -nodisplay -nojvm < drive.m >> mat.out
```

Ansys

- ansys.sh

```
#!/bin/sh
#___INFO__MARK_BEGIN__
# Welcome to use  EasyCluster V1.6 All Rights Reserved.
#
#___INFO__MARK_END__
#
Project=STAMP
#$ -S /bin/sh
#$ -N STAMP
#$ -j y
#$ -o ./
#$ -e ./
#$ -cwd
#$ -q short.q
#$ -pe mvapi 8-8
source ~/.bashrc
hash -r
export path=$TMPDIR:$path
cp $TMPDIR/machines hosts
cat hosts
MAC=`head -1 hosts`:$NSLOTS
/usr/local/ansys121/v121/ansys/bin/ansys121 -b -pp -dis -j=$Project -np=$NSLOTS -machines $MAC -i $Project.txt -
         o $Project.log
```

Fluent

- Fluent_job.sh

```
#!/bin/sh
#___INFO__MARK_BEGIN__
# Welcome to use  EasyCluster V1.6 All Rights Reserved.
#
#___INFO__MARK_END__
#
#$ -S /bin/sh
#$ -N flu1
#$ --
```

Fluent

- Command file --- fluentin

```
; Read case file
rc 100-an-cui-dao-52-771.cas
/file/auto-save/root-name /home/user001/cases/flunet/100-an-cui-dao-52
/file/confirm-overwrite? no
/file/auto-save/case-frequency 100
/file/auto-save/data-frequency 100
; Initialize the solution
/solve/initialize/initialize-flow
; Calculate 1000 iterations
it 1000
; Exit FLUENT
exit
yes
```

Q&A