linux-cmd

xingwenpeng

August 22, 2014

Contents

1 Linux-edu **1**
1.1 基本命令 . . . 1
1.1.1 shell . . . 1
1.1.2 目录和文件 . . . 3
1.1.3 文件属性和用户用户组 . . . 9
1.1.4 查找与检索 . . . 11
1.1.5 安装卸载软件 . . . 12
1.1.6 磁盘管理 . . . 13
1.1.7 压缩包管理 . . . 20
1.1.8 进程管理 . . . 22
1.1.9 环境变量 . . . 26
1.1.10 用户管理 . . . 26
1.1.11 网络管理 . . . 27
1.1.12 常用服务器构建 . . . 28
1.1.13 其它命令 . . . 30
1.1.14 练习 . . . 31
1.1.15 需要安装的组件 . . . 31
1.2 vim . . . 31
1.2.1 vi 简介 . . . 31
1.2.2 vim 工作模式 . . . 32
1.2.3 vim 基础操作 . . . 32
1.2.4 vim 打造 IDE . . . 34
1.3 gcc . . . 35
1.4 toolchain . . . 35
1.5 静态库和共享库 . . . 36
1.5.1 静态库 . . . 36
1.5.2 共享库 . . . 37

1.5.3 共享库加载 37
1.6 Makefile 项目管理 38
1.6.1 用途 38
1.6.2 基本规则 38
1.7 gdb 调试工具 39
1.7.1 gdb 调试模式 41

1 Linux-edu

1.1 基本命令

1.1.1 shell

1. shell 家族 shell：命令解释器，根据输入的命令执行相应命令。

   察看当前系统下有哪些 shell：

   cat /etc/shells

   察看当前系统正在使用的 shell

   echo $SHELL

   常见 shell：

   /bin/sh (已经被/bin/bash 所取代)

   /bin/bash (就是 Linux 默认的 shell)

   /bin/ksh (Kornshell 由 AT&T Bell lab. 发展出来的，兼容于 bash)

   /bin/tcsh (整合 C Shell ，提供更多的功能)

   /bin/csh (已经被/bin/tcsh 所取代)

   /bin/zsh (基于 ksh 发展出来的，功能更强大的 shell)

2. bash bash 是一个为 GNU 计划编写的 Unix shell。它的名字是一系列缩写：Bourne-Again SHell —这是关于 Bourne shell（sh）的一个双关语（Bourne again / born again）

   bash 是许多 Linux 平台的内定 Shell，事实上，还有许多传统 UNIX 上用的 Shell，像 tcsh、csh、ash、bsh、ksh 等等，Shell Script 大致都类同，当您学会一种 Shell 以后，其它的 Shell 会很快就上手，大多数的时候，一个 Shell Script 通常可以在很多种 Shell 上使用。

   bash 是大多数 Linux 系统以及 Mac OS X v10.4 默认的 shell，它能运行于大多数 Unix 风格的操作系统之上，甚至被移植到了 Microsoft Windows 上的 Cygwin 系统中，以实现 windows 的 POSIX 虚拟接口。此外，它也被 DJGPP 项目移植到了 MS-DOS 上。

3. 命令和路径补齐 在 bash 下敲命令时，Tab 键可以补全已经敲了一部分的文件名和目录名。如果是 Ubuntu 系统，系统默认启用了 bash completion，还可以补全命令的某些参数、Makefile 目标等等。如果是 Debian 系统，可以用以下命令启用 bash completion：

   $ source /etc/bash$_{\text{completion}}$

   建议将这一行加入~/.bashrc 启动脚本中。比如使用 sudo 后面接命令，如果没有 bash completion 则只有 sudo 可以补全，后面的命令不能补全。如果启用了 bash completion，则后面的命令，包括命令的某些参数（比如 aptitude 命令的 install）都可以补全了。

   比如在主目录下要列出桌面目录的内容，输入（不回车）

   $ ls De

   然后敲 Tab 键，如果以 De 开头的文件或文件夹只有 Desktop 一个，就自动补全为

   $ ls Desktop

   否则，再敲一次 Tab 键，将会把所有以 De 开头的文件或文件夹列在下面供你选择 (在这里我们手动创建另外一个以 De 开头的文件)

   $ touch Death

   $ ls De

   Death Desktop/

   你可以再补敲一个 s 再 Tab，这次 Desktop 就会补全到命令后面了。有的人是从 DOS 时代过来的，留下一个很不好的习惯就是在找一个文件时反复地 cd、ls、cd、ls。。。等找到了要找的文件时再想回到先前的目录，已经不记得先前是从哪个目录转到这里来的了。

   我们从上面可以看出，Tab 补全本身就具备了 ls 的功能，上面的 Tab 补全相当于 ls -Fd De* 命令。所以我们完全不必反复地 cd 到别的目录然后 ls 去找文件，多按几次 Tab 就可以一条命令完成了，这样的好处是我们的当前目录不用变，不需要找完了文件再 cd 回来，同时省去了大量的按键次数。更重要的是，自动补全同时兼具了检查拼写错误的功能，如果前面几个字母拼写错了，就补全不出东西来，用户就知道拼写错了，如果前面几个字母没有拼写错，那么由系统补全出来的文件名肯定也不会有拼写错误，避免了用户在敲很长的文件名时易犯的拼写错误。

4. 历史记录

history	查看历史命令

历史记录是另外一个非常方便的功能。按上下移动光标键（或者 Ctrl-p、Ctrl-n）可以一条一条浏览以前输过的命令。如果有需要重复输入的命令就不用输第二次了。如果你能记住以前输过的某条命令中的某个关键字，可以按 Ctrl-r，然后输入关键字，随着你每输入一个字母，bash 会做增量式（increasingly）查找，也可以反复按 Ctrl-r 或 Ctrl-s 向前向后查找。如果找到了，按左右移动光标键或 Home 键 (Ctrl-a) 或 End 键 (Ctrl-e) 将该命令带回提示符下进一步修改，或者直接按 Enter 键原封不动地执行该命令。

5. 主键盘快捷键 bash 的快捷键和 emacs 保持一致，用惯其中之一再用另一个程序会很顺手的。请记住一条原则：尽量使用主键盘快捷键而不使用移动光标键和编辑键。因为手不必离开主键盘是效率最高的，这样在你一生之中所节省的来回移动手的时间绝对可以用星期来计算，是绝对值得你花十分钟的时间记住这些快捷键的。

功能	快捷键	助记
上	Ctrl-p	previous
下	Ctrl-n	next
左	Ctrl-b	backward
右	Ctrl-f	forward
Del	Ctrl-d	delete 光标后面的
Home	Ctrl-a	the first letter
End	Ctrl-e	end
Backspace	Backspace	delete 光标前面的

1.1.2 目录和文件

1. 类 Unix 系统目录结构 ubuntu 没有盘符这个概念，只有一个根目录/所有文件都在它下面

   / 根目录

   bin //系统可执行程序，如命令

   boot //内核和启动程序，所有和启动相关的文件都保存在这里

   grub //引导器相关文件

   dev //设备文件

   etc //系统软件的启动和配置文件，系统在启动过程中需要读取的文件都在这个目录。如 LILO 参数、用户账户和密码。

   home //用户的主目录。下面是自己定义的用户名的文件夹

lib //系统程序库文件, 这个目录里存放着系统最基本的动态链接共享库，类似于 Windows 下的 system32 目录，几乎所有的应用程序都需要用到这些共享库。

media //挂载媒体设备，如光驱、U 盘等

mnt //目录是让用户临时挂载别的文件系统，如挂载 Windows 下的某个分区，ubuntu 默认还是挂载在/media 目录。

opt //可选的应用软件包（很少使用）

proc //这个目录是系统内存的映射，我们可以直接访问这个目录来获取系统信息。也就是说，这个目录的内容不在硬盘上而是在内存里。

sbin //管理员系统程序

selinux

srv

sys //udev 用到的设备目录树，/sys 反映你机器当前所接的设备

tmp //临时文件夹

usr //这是个最庞大的目录，我们要用到的很多应用程序和文件几乎都存放在这个目录下。]

bin // 应用程序

game //游戏程序

include

lib //应用程序的库文件

lib64

local //包含用户程序等

sbin //管理员应用程序

2. 用户目录 位于/home/user，称之为用户工作目录或家目录

   表示方式：

   /home/user

   ~

   (a)

3. 相对路径和绝对路径

(a) 绝对路径 从/目录开始描述的路径为绝对路径，如：
cd /home
ls /usr

(b) 相对路径 从当前位置开始描述的路径为相对路径，如：
cd ../../
ls abc/def

(c) . 和.. 每个目录下都有. 和..
. 表示当前目录
.. 表示上一级目录，即父目录
根目录下的. 和.. 都表示当前目录

4. ls ls [OPTION]... [FILE]...

ls 是英文单词 list 的简写，其功能为列出目录的内容。这是用户最常用的一个命令，因为用户需要不时地查看某个目录的内容。该命令类似于 DOS 下的 dir 命令。对于每个目录，该命令将列出其中的所有子目录与文件。对于每个文件，ls 将输出其文件名以及所要求的其他信息。默认情况下，输出条目按字母顺序排序。当未给出目录名或是文件名时，就显示当前目录的信息。

主要的 OPTION 有：

-a 列出隐藏文件，文件中以“.”开头的均为隐藏文件，如：~/.bashrc

-l 列出文件的详细信息

-R 连同子目录中的内容一起列出

用 `ls -l` 命令显示的信息中，开头是由 `10` 个字符构成的字符串，其中第一个字符表示文件类型，它可以是下述类型之一：
`-` 普通文件
`d` 目录
`l` 符号链接
`b` 块设备文件
`c` 字符设备文件
后面的 `9` 个字符表示文件的访问权限，分为 `3` 组，每组 `3` 位。第一组表示文件属主的权限，第二组表示同组用户的权限，第三组表示其他用户的权限。每一组的三个字符分别表示对文件的读、写和执行权限。各权限如下所示：
`r` 读
`w` 写
`x` 可执行。对于目录，表示进入权限。

`s` 当文件被执行时，把该文件的 `UID` 或 `GID` 赋予执行进程的 `UID`（用户 `ID`）或 `GID`（组 `ID`）。
`t` 设置标志位（`sticky bit`）。如果是有 `sticky bit` 的目录，在该目录下任何用户只要有适当的权限即可创建文件，但文件只能被超级用户、目录拥有者或文件属主删除。如果是有 `sticky bit` 的可执行文件，在该文件执行后，指向其正文段的指针仍留在内存。这样再次执行它时，系统就能更快地装入该文件。
`-` 没有相应位置的权限。

访问权限后面的数字表示与该文件共享 `inode` 的文件总数，即硬链接数（参见下面 `ln` 命令）。

5. cd change dir 改变当前所在路径

   cd ~

   cd dir1/dir2

   cd ..

6. which 查看指定命令所在路径

   which ls

7. pwd 查看当前所在路径

   pwd

8. mkdir mkdir [OPTION] DIRECTORY...

   创建目录 DIRECTORY，可以一次创建多个。OPTION 如果是 -p，表示可以连同父目录一起创建。

9. rmdir rmdir [OPTION]... DIRECTORY...

   删除空目录，可以一次删除多个。OPTION 如果是 -p，表示可以连同空的父目录一起删除。mkdir 和 rmdir 的用法举例：

```
$ mkdir a
$ mkdir a/b
$ ls a
b
$ rmdir a/b
$ ls a
$ rmdir a
$ mkdir a/b
```

```
mkdir: cannot create directory `a/b': No such file or directory
$ mkdir -p a/b
$ rmdir -p a/b
```

10. touch touch [OPTION]... FILE...

    将每个文件的访问及修改时间都更新为目前的时间。如果文件不存在，则创建一个字节数为 0 的文件。

11. rm 删除文件：

    rm file

    删除目录：

    rm dir -rf

12. mv 重命名：

    mv file1 file2

    移动文件

    mv file1 ~/

13. cp 拷贝文件：

    cp file1 file2

    cp file1 dir/

    cp file1 ../

    拷贝目录:

    cp dir1 dir2 -r

    cp dir1 ~/ -r

14. cat 查看文件里内容，输出到终端，如果 cat 时没跟文件名，则读标准输入，遇到 \n 后，输出到标准输出，终端下输入 Ctrl-d 表示结束

15. more more [OPTION] [FILE]...

    查看文本文件的内容，屏幕显示完一屏就等待用户按下任意键再滚动到下一屏，如果中途不想继续看下去了，可以按 Ctrl+C 或 q 终止显示。

16. less less [OPTION] [FILE]...

    查看文本文件的内容，屏幕显示完一屏就等待用户按键，用户可以向上或向下查看，如果中途不想继续看下去了，可以按 Ctrl+C 或 q 终止显示。

17. head head [OPTION]... [FILE]...

    显示指定文件的前面几行。如果没有指定文件，将从标准输入（键盘）上读取。如果没有指定要显示的行数，则默认显示前 10 行。如果要显示文件的前 5 行：

    $ head -5 file1

18. tail tail [OPTION]... [FILE]...

    显示文件的最后几行。若没有指定显示的行或字符数，则默认显示末尾 10 行。如果要显示文件末 5 行：

    $ tail -5 file1

19. ln 链接有两种，一种被称为硬链接（Hard Link），另一种被称为符号链接（Symbolic Link）。建立硬链接时，链接文件和被链接文件必须位于同一个文件系统中，并且不能建立指向目录的硬链接。而对符号链接，则不存在这个问题。默认情况下，ln 产生硬链接。如果给 ln 命令加上 -s 选项，则建立符号链接。举例如下，注意 ls -l 列出文件的硬链接数和字节数：

    硬链接：

    touch hello

    ln hello $word_h$

    软链接：

    ln -s hello $word_s$

20. tree 这个命令需要下载安装，ubuntu 下

    sudo apt-get install tree

    按结构树的形状显示目录和文件

21. wc 利用 wc 指令我们可以计算文件的 Byte 数、字数、或是列数, 若不指定文件名称、或是所给予的文件名为“-”, 则 wc 指令会从标准输入设备读取数据。

    wc -l ./*

    -c 或 –bytes 或 –chars 只显示 Bytes 数。

    -l 或 –lines 只显示列数。

    -w 或 –words 只显示字数。

22. od od -tcx file1

- t 指定数据的显示格式，主要的参数有：

c ASCII 字符或反斜杠序列

d[SIZE] 有符号十进制数, 每个整数 SIZE 字节。

f[SIZE] 浮点数, 每个整数 SIZE 字节。

o[SIZE] 八进制（系统默认值为 02）, 每个整数 SIZE 字节。

u[SIZE] 无符号十进制数, 每个整数 SIZE 字节。

x[SIZE] 十六进制数, 每个整数 SIZE 字节。

23. du 查看某个目录的大小：

    以 M 为单位

    du -hm /home/xingwenpeng/test

    以 B 为单位

    du -hb ./*

    以 K 为单位,4k 的整数倍

    du -hk ./*

24. df df 查看磁盘使用情况

    df –block-size=GB

    df –block-size=MB

1.1.3 文件属性和用户用户组

1. whoami 查看当前登陆用户
2. chmod

    1．文字设定法
    chmod [who] [+|-|=] [mode] 文件名
    操作对象 who 可是下述字母中的任一个或者它们的组合：
    u 表示“用户（user）”，即文件或目录的所有者。
    g 表示“同组（group）用户”，即与文件属主有相同组 ID 的所有用户。
    o 表示“其他（others）用户”。
    a 表示“所有（all）用户”。它是系统默认值。
    操作符号可以是：
    + 添加某个权限。

- 取消某个权限。
= 赋予给定权限并取消其他所有权限（如果有的话）。
设置 mode 所表示的权限可用下述字母的任意组合：
r 可读。
w 可写。
x 可执行。

2. 数字设定法
chmod [mode] 文件名
我们必须首先了解用数字表示的属性的含义：0 表示没有权限，1 表示可执行权限，2 表示可写权限，4 表示可读权限，然后将其相加。所以数字属性的格式应为 3 个从 0 到 7 的八进制数，其顺序是（u）（g）（o）。
例如，如果想让某个文件的属主有“读/写”二种权限，需要把 4（可读）+2（可写）= 6（读/写）。
比如设置一个文件允许所有用户可写
$ chmod a+w file1
设置一个文件允许所有用户可读、可写、不可执行
$ chmod 666 file1

user			group			other		
r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1
5			6			3		

3. chown chown [OPTION]... [OWNER][:[GROUP]] FILE...

chown [OPTION]... –reference=RFILE FILE...

更改某个文件或目录的属主和属组。这个命令也很常用。例如 root 用户把自己的一个文件拷贝给用户 xxxx，为了让用户 xxxx 能够存取这个文件，root 用户应该把这个文件的属主设为 xxxx，否则，用户 xxxx 无法存取这个文件。

OPTION 的主要参数：

-R 递归式地改变指定目录及其下的所有子目录和文件的拥有者。

-v 显示 chown 命令所做的工作。

比如把一个文件改为 akaedu 用户和 nogroup 用户组所有

$ sudo chown akaedu:nogroup file1

注意：

(a) chown 需要特权用户才能执行

(b) 一个文件的 owner 和 owning group 是没有关联的。一个文件属于用户 A，也属于用户组 B，并不表示用户 A 属于用户组 B。

4. chgrp chgrp [OPTION]... GROUP FILE...

   chgrp [OPTION]... –reference=RFILE FILE...

   该命令改变（指定）指定文件所属的用户组。其中 group 可以是用户组 ID，也可以是/etc/group 文件中用户组的组名。文件名是以空格分开的要改变属组的文件列表，支持通配符。如果用户不是该文件的属主或超级用户，则不能改变该文件的组。OPTION 的主要参数：

   -R 递归式地改变指定目录及其下的所有子目录和文件的属组。

1.1.4 查找与检索

1. 根据文件名查找

   (a) find find [OPTION] [path...] [expression]

   在目录中搜索文件，path 指定目录路径，系统从这里开始沿着目录树向下查找文件。它是一个路径列表，相互用空格分离，如果不写 path，那么默认为当前目录。Expression 是 find 命令接受的表达式，find 命令的所有操作都是针对表达式的。

   一条最常用的 find 命令——在当前目录及子目录下查找所有以 file 开头的文件名。

   $ find . -name 'file*'

   $ find / -name 'vimrc'

   $ find ~ -name '*.c'

2. 根据内容检索

   (a) grep grep [options] PATTERN [FILE...]

   grep [options] [-e PATTERN | -f FILE] [FILE...]

   在指定文件中搜索特定的内容，并将含有这些内容的行输出到标准输出。若不指定文件名，则从标准输入读取。

   [options] 部分包含的主要参数：

   -c：只输出匹配行的计数。

   -I：不区分大小写（只适用于单字符）。

   -h：查询多文件时不显示文件名。

   -l：查询多文件时只输出包含匹配字符的文件名。

-n：显示匹配行及行号。
-s：不显示不存在或无匹配文本的错误信息。
-v：显示不包含匹配文本的所有行。
-R: 连同子目录中所有文件一起查找。
比如到系统头文件目录下查找所有包含 printf 的文件
$ grep 'printf' /usr/include -R

1.1.5 安装卸载软件

1. apt-get 更新源服务器列表

   sudo vi /etc/apt/sources.list

   更新完服务器列表后需要更新下源

   sudo apt-get update 更新源

   sudo apt-get install package 安装包

   sudo apt-get remove package 删除包

   sudo apt-cache search package 搜索软件包

   sudo apt-cache show package 获取包的相关信息，如说明、大小、版本等

   sudo apt-get install package –reinstall 重新安装包

   sudo apt-get -f install 修复安装

   sudo apt-get remove package –purge 删除包，包括配置文件等

   sudo apt-get build-dep package 安装相关的编译环境

   sudo apt-get upgrade 更新已安装的包

   sudo apt-get dist-upgrade 升级系统

   sudo apt-cache depends package 了解使用该包依赖那些包

   sudo apt-cache rdepends package 查看该包被哪些包依赖

   sudo apt-get source package 下载该包的源代码

   sudo apt-get clean && sudo apt-get autoclean 清理无用的包

   sudo apt-get check 检查是否有损坏的依赖

2. aptitude

3. deb 包安装 安装 deb 软件包命令：sudo dpkg -i xxx.deb
   删除软件包命令：sudo dpkg -r xxx.deb
   连同配置文件一起删除命令：sudo dpkg -r –purge xxx.deb
   查看软件包信息命令：sudo dpkg -info xxx.deb
   查看文件拷贝详情命令：sudo dpkg -L xxx.deb
   查看系统中已安装软件包信息命令：sudo dpkg -l
   重新配置软件包命令：sudo dpkg-reconfigure xxx

4. 原码安装 1. 解压缩源代码包 2.cd dir

   (a) ./configure

   检测文件是否缺失，创建 Makefile, 检测编译环境

   (a) make

   编译源码，生成库和可执行程序

   (a) sudo make install

   把库和可执行程序，安装到系统路径下

1.1.6 磁盘管理

1. mount 命令格式：

   mount [-t vfstype] [-o options] device dir

   其中：

   1.-t vfstype 指定文件系统的类型，通常不必指定。mount 会自动选择正确的类型。常用类型有：

   光盘或光盘镜像：iso9660

   DOS fat16 文件系统：msdos

   Windows 9x fat32 文件系统：vfat

   Windows NT ntfs 文件系统：ntfs

   Mount Windows 文件网络共享：smbfs

   UNIX(LINUX) 文件网络共享：nfs

   2.-o options 主要用来描述设备或档案的挂接方式。常用的参数有：

loop：用来把一个文件当成硬盘分区挂接上系统

ro：采用只读方式挂接设备

rw：采用读写方式挂接设备

iocharset：指定访问文件系统所用字符集

3.device 要挂接 (mount) 的设备。

4.dir 设备在系统上的挂接点 (mount point)。

挂接光盘镜像文件

由于近年来磁盘技术的巨大进步，新的电脑系统都配备了大容量的磁盘系统，在 Windows 下许多人都习惯把软件和资料做成光盘镜像文件通过虚拟光驱来使用。这样做有许多好处：一、减轻了光驱的磨损; 二、现在硬盘容量巨大存放几十个光盘镜像文件不成问题，随用随调十分方便; 三、硬盘的读取速度要远远高于光盘的读取速度，CPU 占用率大大降低。其实 linux 系统下制作和使用光盘镜像比 Windows 系统更方便，不必借用任何第三方软件包。

1、从光盘制作光盘镜像文件。将光盘放入光驱，执行下面的命令。

#cp /dev/cdrom /home/sunky/mydisk.iso 或

#dd if=/dev/cdrom of=/home/sunky/mydisk.iso

注：执行上面的任何一条命令都可将当前光驱里的光盘制作成光盘镜像文件/home/sunky/mydisk.iso

2、将文件和目录制作成光盘镜像文件，执行下面的命令。

#mkisofs -r -J -V mydisk -o *home/sunky/mydisk.iso /home/sunky* mydir

注：这条命令将/home/sunky/mydir 目录下所有的目录和文件制作成光盘镜像文件/home/sunky/mydisk.iso，光盘卷标为：mydisk

3、光盘镜像文件的挂接 (mount)

#mkdir /mnt/vcdrom

注：建立一个目录用来作挂接点 (mount point)

#mount -o loop -t iso9660 /home/sunky/mydisk.iso /mnt/vcdrom

注：使用/mnt/vcdrom 就可以访问盘镜像文件 mydisk.iso 里的所有文件了。

挂接移动硬盘

对 linux 系统而言，USB 接口的移动硬盘是当作 SCSI 设备对待的。插入移动硬盘之前，应先用 fdisk －l 或 more /proc/partitions 查看系统的硬盘和硬盘分区情况。

[root at pldyrouter /]# fdisk -l

Disk /dev/sda: 73 dot 4 GB, 73407820800 bytes

255 heads, 63 sectors/track, 8924 cylinders

Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot Start End Blocks Id System

/dev/sda1 1 4 32098+ de Dell Utility

/dev/sda2 * 5 2554 20482875 7 HPFS/NTFS

/dev/sda3 2555 7904 42973875 83 Linux

/dev/sda4 7905 8924 8193150 f Win95 Ext'd (LBA)

/dev/sda5 7905 8924 8193118+ 82 Linux swap

在这里可以清楚地看到系统有一块 SCSI 硬盘/dev/sda 和它的四个磁盘分区/dev/sda1 – /dev/sda4, /dev/sda5 是分区/dev/sda4 的逻辑分区。接好移动硬盘后，再用 fdisk －l 或 more /proc/partitions 查看系统的硬盘和硬盘分区情况

[root at pldyrouter /]# fdisk -l

Disk /dev/sda: 73 dot 4 GB, 73407820800 bytes

255 heads, 63 sectors/track, 8924 cylinders

Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot Start End Blocks Id System

/dev/sda1 1 4 32098+ de Dell Utility

/dev/sda2 * 5 2554 20482875 7 HPFS/NTFS

/dev/sda3 2555 7904 42973875 83 Linux

/dev/sda4 7905 8924 8193150 f Win95 Ext'd (LBA)

/dev/sda5 7905 8924 8193118+ 82 Linux swap

Disk /dev/sdc: 40.0 GB, 40007761920 bytes

255 heads, 63 sectors/track, 4864 cylinders

Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot Start End Blocks Id System

/dev/sdc1 1 510 4096543+ 7 HPFS/NTFS

/dev/sdc2 511 4864 34973505 f Win95 Ext'd (LBA)

/dev/sdc5 511 4864 34973473+ b Win95 FAT32

大家应该可以发现多了一个 SCSI 硬盘/dev/sdc 和它的两个磁盘分区/dev/sdc1?、/dev/sdc2，其中/dev/sdc5 是/dev/sdc2 分区的逻辑分区。我们可以使用下面的命令挂接/dev/sdc1 和/dev/sdc5。

#mkdir -p /mnt/usbhd1

#mkdir -p /mnt/usbhd2

注：建立目录用来作挂接点 (mount point)

#mount -t ntfs /dev/sdc1 /mnt/usbhd1

#mount -t vfat /dev/sdc5 /mnt/usbhd2

注：对 ntfs 格式的磁盘分区应使用 -t ntfs 参数，对 fat32 格式的磁盘分区应使用 -t vfat 参数。若汉字文件名显示为乱码或不显示，可以使用下面的命令格式。

#mount -t ntfs -o iocharset=cp936 /dev/sdc1 /mnt/usbhd1

#mount -t vfat -o iocharset=cp936 /dev/sdc5 /mnt/usbhd2

linux 系统下使用 fdisk 分区命令和 mkfs 文件系统创建命令可以将移动硬盘的分区制作成 linux 系统所特有的 ext2、ext3 格式。这样，在 linux 下使用就更方便了。使用下面的命令直接挂接即可。

#mount /dev/sdc1 /mnt/usbhd1

挂接 U 盘

和 USB 接口的移动硬盘一样对 linux 系统而言 U 盘也是当作 SCSI 设备对待的。使用方法和移动硬盘完全一样。插入 U 盘之前，应先用 fdisk －l 或 more /proc/partitions 查看系统的硬盘和硬盘分区情况。

[root at pldyrouter root]# fdisk -l

Disk /dev/sda: 73 dot 4 GB, 73407820800 bytes

255 heads, 63 sectors/track, 8924 cylinders

Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot Start End Blocks Id System

/dev/sda1 1 4 32098+ de Dell Utility

/dev/sda2 * 5 2554 20482875 7 HPFS/NTFS

/dev/sda3 2555 7904 42973875 83 Linux

/dev/sda4 7905 8924 8193150 f Win95 Ext'd (LBA)

/dev/sda5 7905 8924 8193118+ 82 Linux swap

插入 U 盘后，再用 fdisk －l 或 more /proc/partitions 查看系统的硬盘和硬盘分区情况。

[root at pldyrouter root]# fdisk -l

Disk /dev/sda: 73 dot 4 GB, 73407820800 bytes

255 heads, 63 sectors/track, 8924 cylinders

Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot Start End Blocks Id System

/dev/sda1 1 4 32098+ de Dell Utility

/dev/sda2 * 5 2554 20482875 7 HPFS/NTFS

/dev/sda3 2555 7904 42973875 83 Linux

/dev/sda4 7905 8924 8193150 f Win95 Ext'd (LBA)

/dev/sda5 7905 8924 8193118+ 82 Linux swap

Disk /dev/sdd: 131 MB, 131072000 bytes

9 heads, 32 sectors/track, 888 cylinders

Units = cylinders of 288 * 512 = 147456 bytes

Device Boot Start End Blocks Id System

/dev/sdd1 * 1 889 127983+ b Win95 FAT32

Partition 1 has different physical/logical endings:

phys=(1000, 8, 32) logical=(888, 7, 31)

系统多了一个 SCSI 硬盘/dev/sdd 和一个磁盘分区/dev/sdd1,/dev/sdd1 就是我们要挂接的 U 盘。

#mkdir -p /mnt/usb

注：建立一个目录用来作挂接点 (mount point)

#mount -t vfat /dev/sdd1 /mnt/usb

注：现在可以通过/mnt/usb 来访问 U 盘了, 若汉字文件名显示为乱码或不显示，可以使用下面的命令。

#mount -t vfat -o iocharset=cp936 /dev/sdd1 /mnt/usb

挂接 Windows 文件共享

Windows 网络共享的核心是 SMB/CIFS，在 linux 下要挂接(mount)windows 的磁盘共享，就必须安装和使用 samba 软件包。现在流行的 linux 发行版绝大多数已经包含了 samba 软件包，如果安装 linux 系统时未安装 samba 请首先安装 samba。当然也可以到 www.samba.org 网站下载.....新的版本是 3.0.10 版。

当 windows 系统共享设置好以后，就可以在 linux 客户端挂接(mount) 了，具体操作如下：

mkdir - p /mnt/samba

注：建立一个目录用来作挂接点 (mount point)

mount -t smbfs -o username=administrator,password=pldy123 //10.140.133.23/c$ /mnt/samba

注：administrator 和 pldy123 是 ip 地址为 10.140.133.23 windows 计算机的一个用户名和密码，c$ 是这台计算机的一个磁盘共享

如此就可以在 linux 系统上通过/mnt/samba 来访问 windows 系统磁盘上的文件了。以上操作在 redhat as server 3、redflag server 4.1、suse server 9 以及 windows NT 4.0、windows 2000、windows xp、windows 2003 环境下测试通过。

挂接 UNIX 系统 NFS 文件共享

类似于 windows 的网络共享，UNIX(Linux) 系统也有自己的网络共享，那就是 NFS(网络文件系统)，下面我们就以 SUN Solaris2.8 和 REDHAT as server 3 为例简单介绍一下在 linux 下如何 mount nfs 网络共享。

在 linux 客户端挂接 (mount)NFS 磁盘共享之前，必须先配置好 NFS 服务端。

1、Solaris 系统 NFS 服务端配置方法如下：

(1) 修改/etc/dfs/dfstab, 增加共享目录

share -F nfs -o rw /export/home/sunky

(2) 启动 nfs 服务

/etc/init.d/nfs.server start

(3)NFS 服务启动以后，也可以使用下面的命令增加新的共享

share /export/home/sunky1

share /export/home/sunky2

注：/export/home/sunky 和/export/home/sunky1 是准备共享的目录

2、linux 系统 NFS 服务端配置方法如下：

(1) 修改/etc/exports, 增加共享目录

/export/home/sunky 10.140.133.23(rw)

/export/home/sunky1 *(rw)

/export/home/sunky2 linux-client(rw)

注：/export/home/目录下的 sunky、sunky1、sunky2 是准备共享的目录，10.140.133.23、*、linux-client 是被允许挂接此共享 linux 客户机的 IP 地址或主机名。如果要使用主机名 linux-client 必须在服务端主机/etc/hosts 文件里增加 linux-client 主机 ip 定义。格式如下：

10.140.133.23 linux-client

(2) 启动与停止 NFS 服务

/etc/rc.d/init.d/portmap start (在 REDHAT 中 PORTMAP 是默认启动的)

/etc/rc.d/init.d/nfs start 启动 NFS 服务

/etc/rc.d/init.d/nfs stop 停止 NFS 服务

注：若修改/etc/export 文件增加新的共享，应先停止 NFS 服务，再启动 NFS 服务方能使新增加的共享起作用。使用命令 exportfs -rv 也可以达到同样的效果。

3、linux 客户端挂接 (mount) 其他 linux 系统或 UNIX 系统的 NFS 共享

mkdir - p /mnt/nfs

注：建立一个目录用来作挂接点 (mount point)

#mount -t nfs -o rw 10.140.133.9:/export/home/sunky /mnt/nfs

注：这里我们假设 10.140.133.9 是 NFS 服务端的主机 IP 地址，当然这里也可以使用主机名，但必须在本机/etc/hosts 文件里增加服务端 ip 定义。/export/home/sunky 为服务端共享的目录。

如此就可以在 linux 客户端通过/mnt/nfs 来访问其它 linux 系统或 UNIX 系统以 NFS 方式共享出来的文件了。以上操作在 redhat as server 3、redflag server4.1、suse server 9 以及 Solaris 7、Solaris 8、Solaris 9 for x86&sparc 环境下测试通过。

(a) 检测存储设备名称

sudo fdisk -l

(a) 挂载存储设备 sdb1 到挂载点/mnt 目录

sudo mount /dev/sdb1 /mnt

(a) 访问/mnt
(b) 卸载/mnt

sudo umount /mnt

2. umount 卸载命令

   umount 挂在点

3. dd dd: 拷贝

   例 1：拷贝光碟 (注意，你的光碟是标准的 iso9660 格式才可以这么做唷！)

   dd if=/dev/cdrom of=cdrom.iso

   例 2：将文件 sfile 拷贝到文件 dfile 中。

   $ dd if=sfile of=dfile

   例 3：创建一个 100M 的空文件

   dd if=/dev/zero of=hello.txt bs=100M count=1

/dev/null，外号叫无底洞，你可以向它输出任何数据，它通吃，并且不会撑着！

/dev/zero, 是一个输入设备，你可你用它来初始化文件, 从里面读出来的数据都是 0。

1.1.7 压缩包管理

1. tar tar [主选项 + 辅选项] 文件或者目录

   tar 可以为文件和目录创建档案。利用 tar 命令用户可以为某一特定文件创建档案（备份文件），也可以在档案中改变文件，或者向档案中加入新的文件。使用该命令时，主选项是必须要有的，辅选项是辅助使用的，可以选用。

   主选项包括：

   c 创建新的档案文件。如果用户想备份一个目录或是一些文件，就要选择这个选项。

   r 把要存档的文件追加到档案文件的末尾。

t 列出档案文件的内容，查看已经备份了哪些文件。

u 更新文件。用新增的文件取代原备份文件，如果在备份文件中找不到要更新的文件，则把它追加到备份文件的最后。

x 从档案文件中释放文件。（常用）

辅选项包括：

f 使用档案文件或设备，这个选项通常是必选的。（常用）

k 保存已经存在的文件。

m 在还原文件时，把所有文件的修改时间设定为现在。

M 创建多卷的档案文件，以便在几个磁盘中存放。

v 详细报告 tar 处理的文件信息。如无此选项，tar 不报告文件信息。（常用）

w 每一步都要求确认。

z 用 gzip 来压缩/解压缩文件，加上该选项后可以将档案文件进行压缩，但还原时也一定要使用该选项进行解压缩。（常用）

j 用 bzip2 来压缩/解压缩文件，加上该选项后可以将档案文件进行压缩，但还原时也一定要使用该选项进行解压缩。（常用）

要将文件备份到一个特定的设备，只需把设备名作为备份文件名。

打包：

tar cvf dir.tar dir

tar xvf dir.tar dir

打 gz 压缩包：

tar zcvf dir.tar.gz dir

tar zxvf dir.tar.gz

打 bz2 压缩包:

tar jcvf dir.tar.bz2 dir

tar jxvf dir.tar.bz2

指定目录解压缩：

tar zxvf dir.tar.gz -C ~/test

2. rar 打包：把 dir 压缩成 newdir.rar

   rar a -r newdir dir

   解包：把 newdir.rar 解压缩到当前目录

   unrar x newdir.rar

3. zip 打包：

   zip -r dir.zip dir

   解包：

   unzip dir.zip

1.1.8 进程管理

1. who 查看当前在线上的用户情况。所有的选项都是可选的，不使用任何选项时，who 命令将显示以下三项内容：

   login name：登录用户名；

   terminal line：使用终端设备；

   login time：登录到系统的时间。

```
xingwenpeng@ubuntu:~/demo$ who -uH
名称         线路           时间               空闲         进程
号 备注
xingwenpeng tty2         2014-08-14 13:31   .          6798
xingwenpeng tty7          2014-08-14 01:31  旧          2423
xingwenpeng pts/1      2014-08-14 01:31 12:00      2843 (:0)
xingwenpeng pts/3       2014-08-14 10:39   .        2843 (:0)
```

2. ps ps [选项]

   ps 命令用于监控后台进程的工作情况，因为后台进程是不和屏幕键盘这些标准输入/输出设备进行通信的，所以如果需要检测其情况，便可以使用 ps 命令了。选项部分如下：

   -e 显示所有进程。

   -f 全格式。

   -h 不显示标题。

   -l 长格式。

   -w 宽输出。

   a 显示终端上的所有进程，包括其他用户的进程。

   r 只显示正在运行的进程。

   x 显示没有控制终端的进程。

   这个命令参数有很多，但一般的用户只需掌握一些最常用的命令参数就可以了。最常用的三个参数是 u、a、x，我们首先以 root 身份登录系统，查看当前进程状况

```
xingwenpeng@ubuntu:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   3672  2008 ?        Ss   08:46   0:01 /sbin/init

xingwenpeng@ubuntu:~$ ps ajx
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
 4592  6948  6948  4592 pts/3    6948 R+    1000   0:00 ps ajx

xingwenpeng@ubuntu:~$ ps -Lf 2423
UID        PID  PPID   LWP  C NLWP STIME TTY      STAT   TIME CMD
1000      2423  2282  2423  0   4 08:46 ?        Ssl    0:00 gnome-session --session=ubunt
1000      2423  2282  2465  0   4 08:46 ?        Ssl    0:00 gnome-session --session=ubunt
1000      2423  2282  2466  0   4 08:46 ?        Ssl    0:00 gnome-session --session=ubunt
1000      2423  2282  2468  0   4 08:46 ?        Ssl    0:00 gnome-session --session=ubunt
```

Head 标头：

USER　用户名

UID　用户 ID（User ID）

PID　进程 ID（Process ID）

PPID　父进程的进程 ID（Parent Process id）

SID　会话 ID（Session id）

%CPU　进程的 cpu 占用率

%MEM　进程的内存占用率

VSZ　进程所使用的虚存的大小（Virtual Size）

RSS　进程使用的驻留集大小或者是实际内存的大小，Kbytes 字节。

TTY　与进程关联的终端（tty）

STAT　进程的状态：进程状态使用字符表示的（STAT 的状态码）

R 运行　Runnable (on run queue)　正在运行或在运行队列中等待。

S 睡眠　Sleeping　休眠中，受阻，在等待某个条件的形成或接受到信号。

I 空闲　Idle

Z 僵死　Zombie（a defunct process）　进程已终止，但进程描述符存在，直到父进程调用 wait4() 系统调用后释放。

D 不可中断　Uninterruptible sleep (ususally IO)　收到信号不唤醒和不可运行，进程必须等待直到有中断发生。

T 停止　Terminate　进程收到 SIGSTOP, SIGSTP, SIGTIN, SIGTOU 信号后停止运行运行。

P 等待交换页

W 无驻留页 has no resident pages 没有足够的记忆体分页可分配。
X 死掉的进程
< 高优先级进程 高优先序的进程
N 低优先 级进程 低优先序的进程
L 内存锁页 Lock 有记忆体分页分配并缩在记忆体内
s 进程的领导者（在它之下有子进程）；
l 多进程的（使用 CLONE_THREAD, 类似 NPTL pthreads）
+ 位于后台的进程组
START 进程启动时间和日期
TIME 进程使用的总 cpu 时间
COMMAND 正在执行的命令行命令
NI 优先级 (Nice)
PRI 进程优先级编号 (Priority)
WCHAN 进程正在睡眠的内核函数名称；该函数的名称是从/root/system.map 文件中获得的。
FLAGS 与进程相关的数字标识

3. jobs

   用来显示当前 shell 下正在运行哪些作业（即后台作业）。

   $ cat

   （按下 Ctrl-z 挂起当前进程）

4. fg fg [job…]

   把指定的后台作业或挂起作业移到前台运行。参数 job 是一个或多个进程的 PID，或者是命令名称，或者是作业号（作业号前面要带一个% 号）。

   通常在 shell 中输入命令启动进程后，如果该进程需要与用户交互，那么此后用户的键盘输入都被该进程读取，直到该进程退出后才出现 shell 提示符 $，这种进程为前台进程。

   如果在命令行的末尾加上 & 字符，则 shell 为这个命令创建一个后台进程，它虽然也可以输出到屏幕，但是不能读取键盘输入，不管执行命令的进程有没有退出都立刻回到 shell 提示符接受下一条命令的输入。如果该进程也需要读取键盘输入，则被挂起等待直到用户用 fg 命令把它变成前台进程。如果一个命令需要较长的处理时间并且不需要与用户交互，就适合把它放在后台执行。

5. bg bg [job…]

   把被挂起的进程提到后台执行。其中，job 是一个或多个进程的 PID、命令名称或者作业号，在参数前要带% 号。

   $ cat

   （按下 Ctrl-z 挂起当前进程）

6. kill 向指定进程发送信号

   kill [-signal | -s signal] pid ...

   查看信号编号

   kill -l [signal]

   给一个进程发信号，或终止一个进程的运行。

```
$ cat
（按 Ctrl-z 挂起当前进程）
[1]+  Stopped                 cat
$ ps
      PID TTY          TIME CMD
     5819 pts/1    00:00:00 bash
     5893 pts/1    00:00:00 cat
 5894 pts/1    00:00:00 ps
$ kill -SIGKILL 5893
$（再次按回车键）
[1]+  Killed                  cat
$
```

   kill 命令如果不带参数而直接跟 pid，就是发给该进程 SIGTERM 信号，大部分进程收到该信号就会终止。但是被挂起的进程不能处理信号，所以必须发 SIGKILL 信号，由系统强制终止进程。

1.1.9 环境变量

1. env 查看当前进程环境变量
2. ~/.bashrc 配置当前用户环境变量
3. /etc/profile 配置系统环境变量, 配置时需要有 root 权限

1.1.10 用户管理

1. 创建用户 sudo useradd -s /bin/bash -g xingwenpeng -d /home/xingwenpeng -m xingwenpeng

   sudo useradd -s /bin/sh -g group -G adm,root xwp

   此命令新建了一个用户 xwp，该用户的登录 Shell 是/bin/sh，他属于 group 用户组，同时又属于 adm 和 root 用户组，其中 group 用户组是其主组。

   -s 指定新用户登陆时 shell 类型

   -g 指定所属组，该组必须已经存在

   -G 指定附属组，该组必须已经存在

   -d 用户家目录

   -m 用户家目录不存在时，自动创建该目录

2. 设置用户组 sudo groupadd xingwenpeng

3. 设置密码 sudo passwd xingwenpeng

4. 切换用户 su 用户名

   su xingwenpeng

   (a) root 用户 变成 root 用户
   sudo su
   设置 root 密码
   passwd

5. 删除用户 userdel 选项用户名

   常用的选项是 -r，他的作用是把用户的主目录一起删除。

   例如：

   sudo userdel -r xingwenpeng

   此命令删除用户 xingwenpeng 在系统文件（主要是/etc/passwd，/etc/shadow，/etc/group 等）中的记录，同时删除用户的主目录。

1.1.11 网络管理

1. ifconfig 1. 查看网卡信息

   ifconfig

2. 关闭网卡

sudo ifconfig eth0 down

3. 开启网卡 eth0

sudo ifconfig eth0 up

4. 给 eth0 配置临时 IP

sudo ifconfig eth0 IP

2. ping ping [选项] 主机名/IP 地址查看网络上的主机是否在工作。它向该主机发送 ICMP ECHO$_{\text{REQUEST}}$ 包。有时我们想从网络上的某台主机上下载文件，可是又不知道那台主机是否开着，命令中各选项的含义如下：-c 数目在发送指定数目的包后停止。-d 设定 SO$_{\text{DEBUG}}$ 的选项。-f 大量且快速地送网络封包给一台机器，看它的回应。-I 秒数设定间隔几秒送一个网络封包给一台机器，预设值是一秒送一次。-l 次数在指定次数内，以最快的方式送封包数据到指定机器（只有超级用户可以使用此选项）。-q 不显示任何传送封包的信息，只显示最后的结果。-r 不经由网关而直接送封包到一台机器，通常是查看本机的网络接口是否有问题。-s 字节数指定发送的数据字节数，预设值是 56，加上 8 字节的 ICMP 头，一共是 64ICMP 数据字节。

3. netstat netstat [选项] 显示网络连接、路由表和网络接口信息，可以让用户得知目前都有哪些网络连接正在运作。命令中各选项的含义如下：-a 显示所有 socket，包括正在监听的。-c 每隔 1 秒就重新显示一遍，直到用户中断它。-i 显示所有网络接口的信息，格式同“ifconfig -e”。-n 以网络 IP 地址代替名称，显示出网络连接情形。-r 显示核心路由表，格式同“route -e”。-t 显示 TCP 协议的连接情况。-u 显示 UDP 协议的连接情况。-v 显示正在进行的工作。

4. nslookup nslookup [-option] [name | -] [server] 查询一台机器的 IP 地址和其对应的域名。它通常需要一台域名服务器来提供域名服务。如果用户已经设置好域名服务器，就可以用这个命令查看不同主机的 IP 地址对应的域名。不带参数使用 nslookup 命令时，出现提示符”>”，在后面输入要查询的 IP 地址或域名并回车即可。如果要退出该命令，输入 exit 并回车即可。

5. finger finger [-lmsp] [user ...] [user@host ...] 查询用户的信息，通常会显示系统中某个用户的用户名、主目录、停滞时间、登录时间、登录 shell 等信息。如果要查询远程机上的用户信息，需要在用户名后面接“@ 主机名”，采用 [用户名 @ 主机名] 的格式，不过要查询的网络主机需要运行 finger 守护进程。命令中各选项的含义如下：-s 显示用户的注册名、实际姓名、终端名称、写状态、停滞时间、登录时间等信息。-l 除了用 -s 选项显示的信息外，还显示用户

主目录、登录 shell、邮件状态等信息，以及用户主目录下的.plan、.project 和.forward 文件的内容。-p 除了不显示.plan 文件和.project 文件以外，与 -l 选项相同。

1.1.12 常用服务器构建

1. ftp

   (a) ftp 服务器

   1. 安装 vsftpd 服务器

   sudo apt-get install vsftpd

   2. 配置 vsftpd.conf 文件

   sudo vi /etc/vsftpd.conf

   添加下面设置

```
anonymous_enable=YES
anon_root=/home/xinwenpeng/ftp
no_anon_password=YES
write_enable=YES
anon_upload_enable=YES
anon_mkdir_write_enable=YES
```

   3. 重启服务器，重新加载/etc/vsftpd.conf 配置文件

   sudo /etc/init.d/vsftpd restart

   4. 进入你的/home/xingwenpeng/ftp 目录下创建一个空目录，供用户上传

   cd ~/ftp

   mkdir anonymous

   chmod 777 anonymous

   5. 测试上传功能，登陆 ftp 服务器，进入到 anonymous 目录

   ftp IP

   cd anonymous

   6. 上传命令，可以把你当前目录下的文件上传到 ftp 服务器的 anonymous 目录

   put somefile

   (b) ftp 客户端

   (c) lftp 客户端 lftp 也是一种 ftp 客户程序。它是以文本方式操作的，但是比起图形界面更为方便。Lftp 几乎具有 bash 的所有方便功

能，Tab 补全，bookmark, queue, 后台下载等可以得到支持。用法与 ftp 类似，主要的指令如下：

put 上传文件

mput 上传多个文件

get 下载文件

mget 下载多个文件

mirror 下载整个目录及其子目录

mirror － R 上传整个目录及其子目录

!command 调用本地 shell 执行命令 command

注意，有的发行版可能缺省没有安装 lftp 工具，需要用户自己安装。如果是 Debian 或 Ubuntu 系统，则安装 lftp 软件包。

2. nfs

1. 安装 nfs 服务器

sudo apt-get install nfs-kernel-server

2. 设置/etc/exports 配置文件

sudo vi /etc/exports

添加这行配置

/home/用户名/nfs *(rw,sync,$\text{no}_{\text{root squash}}$)

3. 在用户目录下创建 nfs 目录

mkdir /home/用户名/nfs

4. 重启服务器，重新加载配置文件

sudo /etc/init.d/nfs-kernel-server restart

5. 在/home/用户名/nfs 目录下创建测试文件 hello

cd /home/用户名/nfs

touch hello

6. 测试服务器，把服务器共享目录 nfs 挂在到/mnt 节点

sudo mount -t nfs -o nolock -o tcp IP:/home/用户名/nfs /mnt

7. 进入/mnt 目录可以看到 hello 文件，表示构建成功

8. 卸载网络共享目录

sudo umount /mnt

3. ssh 1. 安装 ssh 服务器

   sudo apt-get install openssh-server

   2. 远程登陆

   ssh 用户名 @IP

4. telnet

1.1.13 其它命令

1. 终端翻页 Shift-pageup

   Shift-pagedown

2. man 看手册 (叫做 manual 或 man page)。每一个命令和系统函数都有自己的 man page。

   man man

   man read 查看 read 命令的 man page

   man 2 read 查看 read 系统函数的 man page(在第二个 section 中，表示为 read(2))

   man -k read 以 read 为关键字查找相关的 man page

3. clear 清屏。使光标和提示符回到屏幕第一行。

   快捷键：Ctrl-l

4. alias alias [-p] [name[=value] ...] 将 value 字符串起个别名叫 name，以后在命令行输入 name，shell 自动将其解释为 value，如果不带参数执行本命令，或以参数 -p 执行，则显示当前定义的别名列表。 $ alias alias ls='ls –color=auto' alias rm='rm -i'

5. echo echo [-n] 字符串

   在显示器上显示一段文字，一般起到一个提示的作用。其中选项 n 表示输出文字后不换行；字符串可以加引号，也可以不加引号。用 echo 命令输出加引号的字符串时，将字符串原样输出；用 echo 命令输出不加引号的字符串时，将字符串中的各个单词作为字符串输出，各字符串之间用一个空格分割。

   查看上一个程序退出数值，正常情况程序退出值是 0

   echo $?

6. date

7. umask umask [-p] [-S] [mode] umask 指定用户创建文件时的掩码，其中的 mode 和 chmod 的命令中的格式一样。如果不用 mode 参数，则显示当前的 umask 设置。如果用 -S 参数，则以符号形式显示设置。$ umask 0022 $ umask -S u=rwx,g=rx,o=rx 比如该用户 touch 或 gedit 创建一个文件，则其默认权限为 -rw-r–r–，如果该用户创建一个可执行文件 (比如编译生成的程序)，则其默认权限为 -rwxr-xr-x。也就是说，由于 umask 的设定，创建的文件默认是不具有 g 的 w 权限和 o 的 w 权限的，除非用 chmod 更改权限。

8. 关机重启

   (a) poweroff

   (b) shutdown

   (c) reboot

9. 查看系统信息 查看内核版本信息

   uname -a

   查看发行版信息

   $\text{lsb}_{\text{release}}$ -a

1.1.14 练习

1. 创建 test 目录，在里面创建 aa bb cc 三个目录，在 aa 里创建 hello 文件，在 bb 里创建 world 目录，在 cc 里创建 itcast.c, 然后执行 tree/ls -R, 最后删除 test

1.1.15 需要安装的组件

```
sudo apt-get install openssh-server
sudo apt-get install nfs-kernel-server
sudo apt-get install vsftpd
```

1.2 vim

1.2.1 vi 简介

i 简介 Vi 是“Visual interface”的简称，它在 Linux 上的地位就仿佛 Edit 程序在 DOS 上一样。它可以执行输出、删除、查找、替换、块操作等众多文本操作，而且用户可以根据自己的需要对其进行定制。Vi 不是一个排版程序，它不象 Word 或 WPS 那样可以对字体、格式、段落等其他属性

进行编排，它只是一个文本编辑程序。Vi 没有菜单，只有命令，且命令繁多。

Vi 有三种基本工作模式：命令模式、文本输入模式和末行模式。

命令行模式

任何时候，不管用户处于何种模式，只要按一下 ESC 键，即可使 Vi 进入命令模式；我们在 shell 环境 (提示符为 $) 下输入启动 Vi 命令，进入编辑器时，也是处于该模式下。在该模式下，用户可以输入各种合法的 Vi 命令，用于管理自己的文档。此时从键盘上输入的任何字符都被当做编辑命令来解释，若输入的字符是合法的 Vi 命令，则 Vi 在接受用户命令之后完成相应的动作。但需注意的是，所输入的命令并不在屏幕上显示出来。若输入的字符不是 Vi 的合法命令，Vi 会响铃报警。

文本输入模式

在命令模式下输入插入命令 i、附加命令 a 、打开命令 o、修改命令 c、取代命令 r 或替换命令 s 都可以进入文本输入模式。在该模式下，用户输入的任何字符都被 Vi 当做文件内容保存起来，并将其显示在屏幕上。在文本输入过程中，若想回到命令模式下，按键 ESC 即可。

末行模式

末行模式也称 ex 转义模式。在命令模式下，用户按“:”键即可进入末行模式下，此时 Vi 会在显示窗口的最后一行 (通常也是屏幕的最后一行) 显示一个“:”作为末行模式的提示符，等待用户输入命令。多数文件管理命令都是在此模式下执行的 (如把编辑缓冲区的内容写到文件中等)。末行命令执行完后，Vi 自动回到命令模式。例如：

:sp newfile

则分出一个窗口编辑 newfile 文件。如果要从命令模式转换到编辑模式，可以键入命令 a 或者 i；如果需要从文本模式返回，则按 Esc 键即可。在命令模式下输入“:”即可切换到末行模式，然后输入命令。

1.2.2 vim 工作模式

1.2.3 vim 基础操作

进入插入模式:
i: 插入光标前一个字符
I: 插入行首
a: 插入光标后一个字符
A: 插入行未
o: 向下新开一行, 插入行首
O: 向上新开一行, 插入行首
进入命令模式:
ESC: 从插入模式或末行模式进入命令模式

移动光标:
h: 左移
j: 下移
k: 上移
l: 右移
M: 光标移动到中间行
L: 光标移动到屏幕最后一行行首
G: 移动到指定行, 行号 -G
w: 向后一次移动一个字
b: 向前一次移动一个字
{: 按段移动, 上移
}: 按段移动, 下移
Ctr-d: 向下翻半屏
Ctr-u: 向上翻半屏
Ctr-f: 向下翻一屏
Ctr-b: 向上翻一屏
gg: 光标移动文件开头
G: 光标移动到文件末尾
删除命令:x: 删除光标后一个字符, 相当于 Del
X: 删除光标前一个字符, 相当于 Backspace
dd: 删除光标所在行,n dd 删除指定的行数 D: 删除光标后本行所有内容, 包含光标所在字符
d0: 删除光标前本行所有内容, 不包含光标所在字符
dw: 删除光标开始位置的字, 包含光标所在字符
撤销命令:
u: 一步一步撤销
U: 一次性撤销当前行所作的所有操作
Ctr-r: 反撤销
重复命令:
.: 重复上一次操作的命令
文本行移动:
»: 文本行右移
«: 文本行左移
复制粘贴:
yy: 复制当前行,n yy 复制 n 行
p: 在光标所在位置向下新开辟一行, 粘贴
可视模式:
v: 按字符移动, 选中文本
V: 按行移动, 选中文本可视模式可以配合 d, y, », « 实现对文本块的删除, 复制, 左右移动

替换操作:
r: 替换当前字符
R: 替换当前行光标后的字符
分屏操作:
sp: 上下分屏, 后可跟文件名
vsp: 左右分屏, 后可跟文件名
Ctr+w+w: 在多个窗口切换
执行 shell 下命令: 末行模式里输入!, 后面跟命令
查找命令:/: 查找
查看 Man Page: 光标移动到函数上,Shift-k 光标移动到函数上,3Shift-k, 查看第三章的 ManPage
查看宏定义:[-d: 可以查看宏定义, 必须先包含此宏所在的头文件
代码排版:gg=G: 代码自动缩进排版

1.2.4 vim 打造 IDE

vimrc 是 vim 的配置文件，可以两个位置

1. /etc/vim/vimrc

2.~/.vimrc
~/.vimrc 优先级高
1. 拷贝我提供的 vim-ide.tar.gz，保证你的 vim 版本大等于 7.4，vim –v
2. 解包到当前用户目录下, 得到.vim 隐藏文件
tar zxvf vim-ide.tar.gz -C ~/
3. 创建 vim 配置文件，vimrc 的符号链接
ln -s .vim/vimrc .vimrc
4. 拷贝出.vim/.ycm$_{extra conf}$.py 到用户目录或是你的源代码目录
cp ~/.vim/.ycm$_{extra conf}$.py ~/
,ta ,nn
5.vim-ide 里会用到 Ctr-space 键，把系统的输入法快捷键切换成 Ctr-Shift，空出 Ctr-space 键
6.vim-ide 常用操作，备注：这些操作可以通过修改.vimrc 文件进行设置
7. 修改.vimrc 中邮箱，把 xingwenpeng 出现的地方换成你的名字和邮箱
let g:DoxygenToolkit$_{authorName}$=”xingwenpeng <wenpeng.xing@gmail.com>”
8.vim-ide 里的快捷方式,dd 在函数开头生成函数说明,da 在文件开头生成文件说明,dl 生成采用发布协议
,jd 跳转到函数定义或头文件所在地,f 在头文件和源文件之间跳转, 创建对应名字的头文件
,ta 打开函数列表，在右侧,o 关闭多窗口，只留当前窗口,bf 显示已经打开的文件列表

,nn 打开或关闭文件列表,nl 取消搜索后的高亮显示
backspace 关闭一个 tab buffer
,tab 在 tab 标签移动
C-t 创建新标签 tab
:ta fun 跳到指定函数定义
V 选中函数
,zc 折叠当前函数
,zr 打开当前函数
,zR 打开所有折叠
,8 在当前目录下生成 tags 文件
C-] 跳转到函数定义，利用 Ctags
C-o 返回

1.3 gcc

-v / –v / –version 查看 gcc 版本号
-I<DIR> 指定头文件目录，注意 -I 和 <DI> 之间没有空格
-g 包含调试信息
-On n=0˜3 编译优化，n 越大优化得越多
-Wall 提示更多警告信息
-D<DEF> 编译时定义宏，注意 -D 和 <DEF> 之间没有空格
-E 生成预处理文件
-M 生成.c 文件与头文件依赖关系以用于 Makefile，包括系统库的头文件
-MM 生成.c 文件与头文件依赖关系以用于 Makefile，不包括系统库的头文件

1.4 toolchain

binutils 一组用于编译、链接、汇编和其它调试目的的程序，包括 ar、as、ld、nm、objcopy、objdump、ranlib、readelf、size、strings、strip 等
gcc 编译器
glibc 该库实现 Linux 系统函数，例如 open、read 等，也实现标准 C 语言库，如 printf 等。几乎所有应用程序都需要与 glibc 链接
本节主要介绍 binutils 中的几种主要工具的作用。
ar 打包生成静态库
as 汇编器
ld 链接器。本节前面介绍用 gcc 完成链接步骤，其实是 gcc 调用链接器 ld，将用户编译生成的目标文件连同系统的 libc 启动代码链接在一起形成最终的可执行文件

nm 查看目标文件中的符号（全局变量、全局函数等）

objcopy 将原目标文件中的内容复制到新的目标文件中，可以通过不同的命令选项调整目标文件的格式，比如去除某些 ELF 文件头

objdump 用于生成反汇编文件，主要依赖 objcopy 实现，a.out 编译时需要 -g，objdump -dSsx a.out > file

ranlib 为静态库文件创建索引，相当于 ar 命令的 s 选项

readelf 解读 ELF 文件头

1.5 静态库和共享库

* 本节就如何创建和使用程序库进行论述。所谓" 程序库"，简单说，就是包含了数据和执行码的文件。其不能单独执行，可以作为其它执行程序的一部分来完成某些功能。库的存在，可以使得程序模块化，可以加快程序的再编译，可以实现代码重用，可以使得程序便于升级。程序库可分静态库 (static library) 和共享库 (shared object)。

1.5.1 静态库

是在可执行程序运行前就已经加入到执行码中，成为执行程序的一部分；共享库，是在执行程序启动时加载到执行程序中，可以被多个执行程序共享使用。

建议库开发人员创建共享库，比较明显的优势在于库是独立的，便于维护和更新；而静态库的更新比较麻烦，一般不做推荐。然而，它们又各有优点，后面会讲到。

本节所讲述的执行程序和库都采用 ELF(Executable and Linking Format) 格式, 尽管 GNU GCC 工具可以处理其它格式，但不在本节的讨论范围。

静态库可以认为是一些目标代码的集合。按照习惯，一般以".a" 做为文件后缀名。使用 ar(archiver) 命令可以创建静态库。因为共享库有着更大的优势，静态库已经不经常使用。但静态库使用简单，仍有使用的余地，并会一直存在。有些 Unix 系统，如 Solaris 10，已经基本废弃了静态库。

静态库在应用程序生成时，可以不必再编译，节省再编译时间。但在编译器越来越快的今天，这一点似乎已不重要。如果其他开发人员要使用你的程序，而你又不想给其源码，提供静态库是一种选择。从理论上讲，应用程序使用了静态库，要比使用动态加载库速度快 1-5%，但实际上可能并非如此。由此看来，除了使用方便外，静态库可能并非一种好的选择。

要创建一个静态库，或要将目标代码加入到已经存在的静态库中，可以使用以下命令：** 静态库

ar rcs libmylib.a file1.o

file2.o 以上表示要把目标码 file1.o 和 file2.o 加入到静态库 libmylib.a 中 (ar 的参数 r)。若 libmylib.a 不存在，会自动创建 (ar 的参数 c)。然后更新.a 文件的索引，使之包含新加入的.o 文件的内容 (ar 的参数 s)。

静态库创建成功后，需要链接到应用程序中使用。使用 gcc 的 -l 选项来指定静态库，使用 -L 参数来指定库文件的搜索路径。比如上述例子应指定 -lmylib，所有库文件名都以 lib 开头，开头的 lib 在指定参数时应省略。-l 和 -L 之后都直接带参数而不跟空格。

在使用 gcc 时，要注意其参数的顺序。-l 是链接器选项，一定要放在被编译的文件名称之后；若放在文件名称之前则会连接失败，并会出现莫名其妙的错误。这一点切记。

1.5.2 共享库

共享库的创建比较简单，基本有两步。首先使用 -fPIC 或 -fpic 创建目标文件，PIC 或 pic 表示位置无关代码，然后就可以使用以下格式创建共享库了：gcc -share -Wl,-soname,$your_{soname}$ -o $library_{name}$ $file_{list}$ $library_{list}$ 下面是使用 a.c 和 b.c 创建库的示例：

基础班使用

gcc -fPIC -c a.c

gcc -fPIC -c b.c

gcc -shared -Wl -o libmyab.so a.o b.o

就业班使用

gcc -share -Wl,-soname,libmyab.so.1 -o libmyab.so.1.0.1 a.o b.o

1.5.3 共享库加载

在所有基于 GNUglibc 的系统中，在启动一个 ELF 二进制执行程序时，一个特殊的程序” 程序装载器” 会被自动装载并运行。在 linux 中，这个程序装载器就是/lib/ld-linux.so.X(X 是版本号)。它会查找并装载应用程序所依赖的所有共享库。被搜索的目录保存在/etc/ld.so.conf 文件中。当然，如果程序的每次启动，都要去搜索一番，势必效率不堪忍受。Linux 系统已经考虑这一点，对共享库采用了缓存管理。ldconfig 就是实现这一功能的工具，其缺省读取/etc/ld.so.conf 文件，对所有共享库按照一定规范建立符号连接，然后将信息写入/etc/ld.so.cache。/etc/ld.so.cache 的存在大大加快了程序的启动速度。

1. 修改/etc/ld.so.conf

sudo vi /etc/ld.so.conf

添加你的共享库路径

1. 更新查找共享库的路径

sudo ldconfig -v
3. 测试你的程序可否找到共享库
ldd a.out

1.6 Makefile 项目管理

1.6.1 用途

- 项目代码编译管理
- 节省编译项目时间
- 一次编写终身受益
- 操作示例文件：add.c sub.c mul.c dive.c main.c

1.6.2 基本规则

1. 三要素 目标
   条件
   命令
2. Makefile 工作原理
   - 分析各个目标和依赖之间的关系
   - 根据依赖关系自底向上执行命令
   - 根据修改时间比目标新，确定更新
   - 如果目标不依赖任何条件，则执行对应命令，以示更新
3. clean
   - 用途：清除编译生成的中间.o 文件和最终目标文件
   - make clean 如果当前目录下有同名 clean 文件，则不执行 clean 对应的命令
   - 伪目标声明：.PHONY:clean
   - clean 命令中的特殊符号
     - "-" 此条命令出错，make 也会继续执行后续的命令。如："-rm main.o"

  - ”@” 不显示命令本身，只显示结果。如：”@echo ”clean done””
- 其它
  - make 默认执行第一个出现的目标，可通过 make dest 指定要执行的目标
  - distclean 目标
  - install 目标

4. 隐含规则和模式规则

1.7 gdb 调试工具

程序中除了一目了然的 Bug 之外都需要一定的调试手段来分析到底错在哪。到目前为止我们的调试手段只有一种：根据程序执行时的出错现象假设错误原因，然后在代码中适当的位置插入 printf，执行程序并分析打印结果，如果结果和预期的一样，就基本上证明了自己假设的错误原因，就可以动手修正 Bug 了，如果结果和预期的不一样，就根据结果做进一步的假设和分析。本章我们介绍一种非常强大的调试工具 gdb，可以完全操控程序的运行，使得程序就像你手里的玩具一样，叫它走就走，叫它停就停，并且随时可以查看程序中所有的内部状态，比如各变量的值、传给函数的参数、当前执行的语句位置等。掌握了 gdb 的用法以后，调试的手段就更加丰富了。但要注意，即使调试的手段非常丰富了，其基本思想仍然是“分析现象 -> 假设错误原因 -> 产生新的现象去验证假设”这样一个循环，根据现象如何假设错误原因，以及如何设计新的现象去验证假设，这都需要非常严密的分析和思考，如果因为手里有了强大的工具就滥用，而忽视了严谨的思维，往往会治标不治本地修正 Bug，导致一个错误现象消失了但 Bug 仍然存在，甚至是把程序越改越错。本章通过几个初学者易犯的错误实例来讲解如何使用 gdb 调试程序，在每个实例后面总结一部分常用的 gdb 命令。

gcc -g main.c -o main

命令	简写	作用
help	h	按模块列出命令类
help class		查看某一类型的具体命令
list	l	查看代码，可跟行号和函数名
quit	q	退出 gdb
run	r	全速运行程序
start		单步执行，运行程序，停在第一行执行语句
next	n	逐过程执行
step	s	逐语句执行，遇到函数，跳到函数内执行
backtrace	bt	查看函数的调用的栈帧和层级关系
info	i	查看函数内部局部变量的数值
frame	f	切换函数的栈帧
finish		结束当前函数，返回到函数调用点
set		设置变量的值
run argv[1]argv[2]		调试时命令行传参
print	p	打印变量和地址
break	b	设置断点，可根据行号和函数名
delete	d	删除断点 d breakpoints NUM
display		设置观察变量
undisplay		取消观察变量
continue	c	继续全速运行剩下的代码
enable breakpoints		启用断点
disable breakpoints		禁用断点
x		查看内存 x /20xw 显示 20 个单元，16 进制，4 字节每单元
watch		被设置观察点的变量发生修改时，打印显示
i watch		显示观察点
core 文件		ulimit -c 1024 开启 core 文件，调试时 gdb a.out core

[1]+ Stopped cat
$ fg %1
cat
（按 Ctrl-d 输入文件结束符）
$

[2]+ Stopped cat

第一列方括号中的数字表示作业序号，它是由当前运行的 shell 分配的，而不是由操作系统统一分配的。在当前 shell 环境下，第一后台作业的作业号为 1，第二作业的作业号为 2，等等。第二列中的“+”号表示相应作业的优先级比“-”号对应作业的优先级高。第三列表明作业状态，是否为运行、中断、等待输入或停止等。最后列出的是创建当前这个作业所对应的命令行。

1.7.1 gdb 调试模式

- run 全速运行
- start 单步调试