目录

2024年4月2日 12:49

8.1 基本命令

终端与指令

Linux发展史

开源文化
特点：开源、一切皆文件、多用户、多任务、优异性能与稳定性
分支：Ubuntu、Debian、centos(社区企业操作系统)、Redhat等

一、终端与指令

1.1终端

一系列输入输出设备的总称
1、终端的打开方式
鼠标右键-打开终端：在哪个界面上打开，打开的就是哪个路径
Ctrl+Alt+t：用户
切换终端：Alt+数字

1.2 Shell

命令解释器，根据输入的命令执行相应的命令
echo $SHELL　　//查看当前终端的命令解释器
结果：/bin/bash

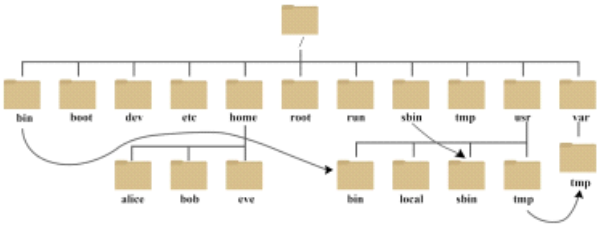
不同的解释器：shell、bash、dash

1.3指令

1、指令的标准格式
command [-options] parameter1 parameter2 …
命令　　选项　　参数1　　参数2　…

list - ls 列出
ls //显示当前路径下的文件
ls -l　//以列表形式显示
ls -l /　//查看根目录下的文件，以列表形式显示
ls 文件名　// 查看该文件名下的文件
ls -a　　// 显示当前目录下的所有文件（包括隐藏文件）
ls --all　// 显示当前目录下的所有文件（包括隐藏文件）
ls --help　// 查看ls指令如何使用
cd 目录　// 前往某个目录
cd ..　// 返回到上一级目录
Tab键可以补齐命令

1.4 根目录



1、bin目录：存二进制文件（可执行文件），存的是一些经常使用的命令
2、dev目录：device 设备文件，例如输入输出设备等

3、etc目录：配置文件，etc 是 Etcetera(等等) 的缩写,这个目录用来存放所有的系统管理所需要的配置文件和子目录。
4、home目录：用户的主目录，在 Linux 中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的。
5、lib目录：lib 是 Library(库) 的缩写这个目录里存放着系统最基本的动态连接共享库，其作用类似于 Windows 里的 DLL 文件。几乎所有的应用程序都需要用到这些共享库。
6、proc： 是 Processes(进程) 的缩写，进程在内存中，/proc 是一种伪文件系统（也即虚拟文件系统），存储的是当前内核运行状态的一系列特殊文件，这个目录是一个虚拟的目录，它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。这个目录的内容不在硬盘上而是在内存里。
7、root：该目录为系统管理员家目录，也称作超级权限者的用户主目录。

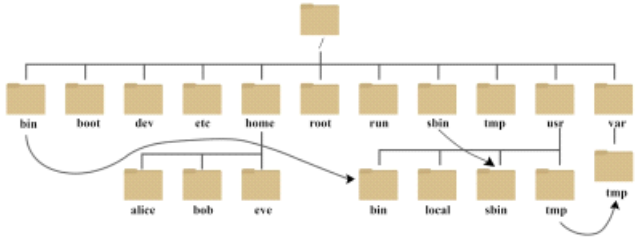
8、tmp：tmp 是 temporary(临时) 的缩写这个目录是用来存放一些临时文件的。
9、usr：存下载的软件，usr 是 unix shared resources(共享资源) 的缩写，这是一个非常重要的目录，用户的很多应用程序和文件都放在这个目录下，类似于 windows 下的 program files 目录

文件基本属性

2024年4月2日 11:40

二、文件基本属性

2.1 Linux 系统目录结构



1、bin目录：存二进制文件（可执行文件），存的是一些经常使用的命令
2、dev目录：device 设备文件，例如输入输出设备等

```
weihong@weihong:/dev/input$ ls
by-id    event0  event2  event4  js0   mouse0  mouse2
by-path  event1  event3  event5  mice  mouse1  mouse3
weihong@weihong:/dev/input$
```

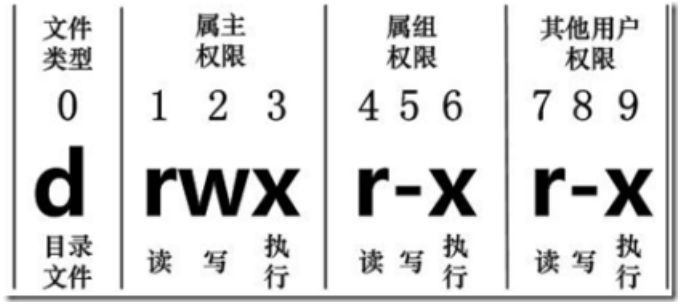
3、etc目录：配置文件，etc 是 Etcetera(等等) 的缩写,这个目录用来存放所有的系统管理所需要的配置文件和子目录。
4、home目录：用户的主目录，在 Linux 中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的。
5、lib目录：lib 是 Library(库) 的缩写这个目录里存放着系统最基本的动态连接共享库，其作用类似于 Windows 里的 DLL 文件。几乎所有的应用程序都需要用到这些共享库。
6、proc： 是 Processes(进程) 的缩写，进程在内存中，/proc 是一种伪文件系统（也即虚拟文件系统），存储的是当前内核运行状态的一系列特殊文件，这个目录是一个虚拟的目录，它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。这个目录的内容不在硬盘上而是在内存里。
7、root：该目录为系统管理员，也称作超级权限者的用户主目录。

8、tmp：tmp 是 temporary(临时) 的缩写这个目录是用来存放一些临时文件的。
9、usr：usr 是 unix shared resources(共享资源) 的缩写，这是一个非常重要的目录，用户的很多应用程序和文件都放在这个目录下，类似于 windows 下的 program files 目录

2.2、Linux 文件基本属性

ls -l 的第一列就是文件的属性，显示一个文件的属性以及文件所属的用户和组
1、文件的属性

```
weihong@weihong:/dev$ ls -l
总计 0
crw-r--r--  1 root    root     10, 235  4月  2 10:23 autofs
drwxr-xr-x  2 root    root         320  4月  2 10:23 block
drwxr-xr-x  2 root    root          80  4月  2 10:23 bsg
crw-------  1 root    root     10, 234  4月  2 10:23 btrfs-control
drwxr-xr-x  3 root    root          60  4月  2 10:23 bus
lrwxrwxrwx  1 root    root           3  4月  2 10:23 cdrom -> sr0
drwxr-xr-x  2 root    root        3780  4月  2 12:38 char
crw--w----  1 root    tty        5,   1  4月  2 10:23 console
lrwxrwxrwx  1 root    root          11  4月  2 10:23 core -> /proc/kcore
```



7个文件类型

在 Linux 中第一个字符代表这个文件是目录、文件或链接文件等，一个有7个文件类型。

- d 则是目录；
- - 则是文件；
- l 则表示为链接文档(link file)、软连接；
- c 表示是字符设备文件
- b 表示是块设备文件
- p pipe 管道文件
- s socket网络传输文件

接下来的字符中，以三个为一组，且均为 rwx 的三个参数的组合。其中， r 代表可读(read)、 w 代表可写(write)、 x 代表可执行(execute)。 要注意的是，这三个权限的位置不会改变，如果没有权限，就会出现减号 - 而已。

2、文件的属主与属组

```
weihong@weihong:~$ ls -l
总计 36
drwxr-xr-x 2 weihong weihong 4096  4月  1 21:51 公共的
drwxr-xr-x 2 weihong weihong 4096  4月  1 21:51 模板
drwxr-xr-x 2 weihong weihong 4096  4月  1 21:51 视频
```

三个文件属主属组都是weihong，属主权限为rwx，属组权限为r-x，其他用户权限为r-x

2.3、更改文件属性

1、chgrp：更改文件属组

chgrp [-R] 属组名 文件名

-R：递归更改文件属组，就是在更改某个目录文件的属组时，如果加上 -R 的参数，那么该目录下的所有文件的属组都会更改。

2、chown：更改文件所有者（owner），也可以同时更改文件所属组。

chown [- R] 所有者 文件名

chown [-R] 所有者:属组名 文件名

3、chmod：更改文件9个属性（change mode）

Linux文件属性有两种设置方法，一种是数字，一种是符号。

（1）数字

我们可以使用数字来代表各个权限，各权限的分数对照表如下：r:4、w:2、x:1

chmod [-R] xyz 文件或目录
xyz : 就是刚刚提到的数字类型的权限属性，为 rwx 属性数值的相加。
-R : 进行递归(recursive)的持续变更，以及连同次目录下的所有文件都会变更
例如：chmod 700 snap是将snap的文件属性改为rwx------

（2）符号类型改变文件权限
可以使用 chmod u=rwx,g=rx,o=r 文件名 来更改设定

文件与目录管理

2024年4月2日 11:40

三、文件与目录管理

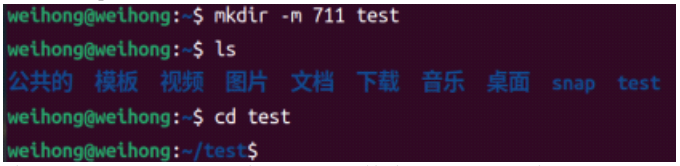
绝对路径与相对路径。
绝对路径：
路径的写法，由根目录 / 写起，例如： /usr/share/doc 这个目录。
相对路径：
路径的写法，不是由 / 写起，例如由 /usr/share/doc 要到 /usr/share/man 底下时，可以写成： cd ../man 这就是相对路径的写法。

3.1 处理目录的常用命令

ls（英文全拼：list files）：列出目录及文件名
cd（英文全拼：change directory）：切换目录
pwd（英文全拼：print work directory）：显示目前的目录
du（）：显示当前目录下的磁盘使用大小
mkdir（英文全拼：make directory）：创建一个新的目录
rmdir（英文全拼：remove directory）：删除一个空的目录
cp（英文全拼：copy file）：复制文件或目录
rm（英文全拼：remove）：删除文件或目录
mv（英文全拼：move file）：移动文件与目录，或修改文件与目录的名称
可以使用 man [命令] 来查看各个命令的使用文档，如 ：man cp。

1、ls（英文全拼：list files）：列出目录及文件名
ls > a.txt 把当前目录下的文件名放入a.txt （一个大于号覆盖，两个大于号叠加）

2、mkdir（英文全拼：make directory）：创建一个新的目录
mkdir [-mp] 目录名称
选项与参数：
-m ：配置文件的权限喔！直接配置，不需要看默认权限 (umask) 的脸色～
-p ：帮助你直接将所需要的目录(包含上一级目录)递归创建起来！
mkdir -p a/b/c 创建多级目录，a、b不存在自动创建



在当前目录下创建一个 test文件夹 设置权限为 711

3、touch命令用于修改文件或者目录的时间属性，包括存取时间和更改时间。若文件不存在，系统会建立一个新的文件

4、rmdir（英文全拼：remove directory）：删除一个空的目录
如果该目录下有内容，则无法删除

5、cp（英文全拼：copy file）：复制文件或目录
[root@www ~]# cp [-adfilprsu] 来源档(source) 目标档(destination)
[root@www ~]# cp [options] source1 source2 source3 .... Directory

weihong@weihong:~/test$ cp -r test1 test2
在test目录下复制test1，重命名为test2

-a 所有都复制到目标档（甚至连时间属性也能复制过去）

weihong@weihong:~/test$ cp test1/hello.txt test2/hello2.txt
复制test1下的hello.txt，到test2下并命名为hello2.txt

6、rm（英文全拼：remove）：删除文件或目录

rm [-fir] 文件或目录
-f ：就是 force 的意思，忽略不存在的文件，不会出现警告信息；
-i ：互动模式，在删除前会询问使用者是否动作
-r ：递归删除啊！最常用在目录的删除了！这是非常危险的选项！！！

7、mv（英文全拼：move file）：移动文件与目录，或修改文件与目录的名称

-f ：force 强制的意思，如果目标文件已经存在，不会询问而直接覆盖；
-i ：若目标文件 (destination) 已经存在时，就会询问是否覆盖！
-u ：若目标文件已经存在，且 source 比较新，才会升级 (update)

mv a.txt b.txt 把a.txt 改名为 b.txt

3.2 文件内容查看

cat 由第一行开始显示文件内容(查看文件内容)
cat -n 补齐行号
cat a.txt b.txt > c.txt（将a.txt 和 b.txt 的内容 添加到c.txt中，> 是覆盖添加 >> 是追加）

tac 从最后一行开始显示，可以看出 tac 是 cat 的倒着写！（倒序查看，查看日志文件）

nl 显示的时候，顺道输出行号！
more 一页一页的显示文件内容
less 与 more 类似，但是比 more 更好的是，他可以往前翻页！
head 只看头几行（默认前十行）

wc 对文件的统计信息 -l（行号） -w（单词数） -c（字节）

tail 只看尾巴几行
-f 动态监测文件内容

find 查找文件
find ./ -name "*.txt"（按名字查找当前路径下所有的txt文件）
find ./ -type f （按类型查找当前路径下所有文件类型的文件） -maxdepth 数字 选择搜素深度
find ./ -size +1 -size -100（按照大小查找当前目录下大于1小于100的文件）

> 输出重定向 ls > a.txt 把ls显示的内容 输出到 a.txt（本来ls是输出到终端，现在被重新定向输出到a.txt，覆盖输出）

cat a.txt b.txt > c.txt，把 a.txt b.txt 合并到 c.txt

>> 追加重重定向

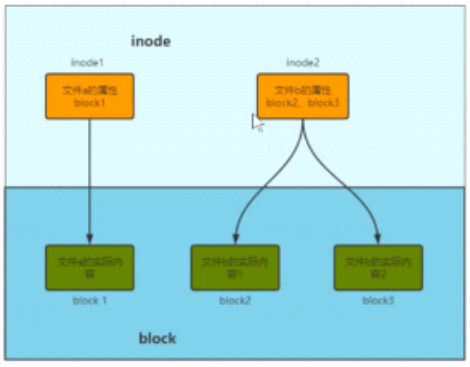
| 管道符：用来连接两个指令，将前一个指令的返回值传给下一个指令
ls / | grep b
查看根目录下的所有文件，将返回值通过管道符给grep，过滤文件中带b字母的

ls /etc | wc -l
查看根目录/etc下的所有文件，将返回值通过管道符给wc，即文件数量

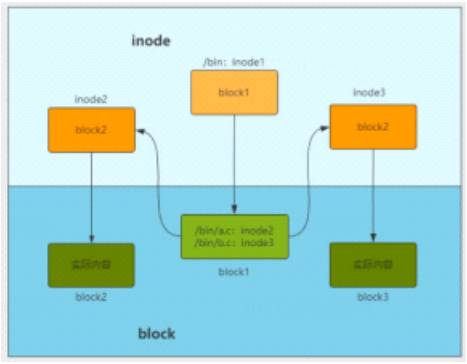
3.3 Linux 链接概念

链接

在 Linux 的文件系统中，大小等属性，称之为元数据，所有的元数据都存在inode区（没有文件名）、而文件内容存放在block区，目录的block存文件名和其他文件夹的inode



文件夹的block中存放的是文件夹内文件的inode



Linux 链接分两种，一种被称为硬链接（Hard Link），另一种被称为符号链接（Symbolic Link）。默认情况下，ln 命令产生硬链接。

硬连接 ln a.txt d.txt 链接a.txt 和 d.txt（inode号一样）

在创建硬链接时，整个文件系统的inode不变，block一般不变（满了才加），所以创建硬链接不占空间

在对硬链接读写时，操作的是源文件

删除硬链接或源文件，不会影响其他inode

硬链接计数为0时，inode节点会释放

不能跨文件系统

不能链接目录，对文件或者目录创建硬链接，将树结构改为环、图结构，可能死循环，增加复杂度

硬连接指通过索引节点来进行连接。在 Linux 的文件系统中，保存在磁盘分区中的文件不管是什么类型都给它分配一个编号，称为索引节点号(Inode Index)。在 Linux 中，多个文件名指向同一索引节点是存在的。比如：A 是 B 的硬链接（A 和 B 都是文件名），则 A 的目录项中的 inode 节点号与 B 的目录项中的 inode 节点号相同，即一个 inode 节点对应两个不同的文件名，两个文件名指向同一个文件，A 和 B 对文件系统来说是完全平等的。删除其中任何一个都不会影响另外一个的访问。

硬连接的作用是允许一个文件拥有多个有效路径名，这样用户就可以建立硬连接到重要文件，以防止“误删”的功能。其原因如上所述，因为对应该目录的索引节点有一个以上的连接。只删除一个连接并不影响索引节点本身和其它的连接，只有当最后一个连接被删除后，文件的数据块及目录的连接才会被释放。也就是说，文件真正删除的条件是与之相关的所有硬连接文件均被删除。

软连接

ln -s a.txt e.txt

相当于快捷方式

存文件路径，软连接文件小

使用绝对路径

源文件被修改、软链接失效

另外一种连接称之为符号连接（Symbolic Link），也叫软连接。软链接文件有类似于 Windows 的快捷方式。它实际上是一个特殊的文件。在符号连接中，文件实际上是一个文本文件，其中包含的有另一文件的位置信息。比如：A 是 B 的软链接

（A 和 B 都是文件名），A 的目录项中的 inode 节点号与 B 的目录项中的 inode 节点号不相同，A 和 B 指向的是两个不同的 inode，继而指向两块不同的数据块。但是 A 的数据块中存放的只是 B 的路径名（可以根据这个找到 B 的目录项）。A 和 B 之间是“主从”关系，如果 B 被删除了，A 仍然存在（因为两个是不同的文件），但指向的是一个无效的链接。

总结

创建一个文件f1，创建f1的一个硬连接文件f2，创建f1的一个符号连接文件f3

1).删除符号连接f3,对f1,f2无影响；

2).删除硬连接f2，对f1,f3也无影响；

3).删除原文件f1，对硬连接f2没有影响，导致符号连接f3失效；

4).同时删除原文件f1,硬连接f2，整个文件会真正的被删除。

vim

四、vim

4.1 一些指令

1、安装与卸载

重启网络
sudo nmcli network off
sudo nmcli network on

更新文件
sudo apt-get update

安装文件
sudo apt-get install sl （安装sl）

卸载文件
sudo apt-get remove sl

安装软件包
sudo dpkg -i 软件包名字

卸载软件包
sudo dpkg -r 软件包名字

软件包后缀.deb

2、压缩、解压缩

压缩文件 tar
tar -zcvf abcd.tar.gz a.txt b.txt c.txt d.txt
-z 压缩为gz格式
-c 创建文件
-v 显示压缩过程
-f 输入压缩后的文件名

解压缩
tar -zxvf abcd.tar.gz
-x 解压文件

查看压缩包内文件
tar -ztvf abcd.tar.gz

压缩文件 zip
zip abcd.zip a.txt b.txt c.txt d.txt
解压缩
unzip abcd.zip

压缩文件 rar
rar a -r abcd.rar a.txt b.txt c.txt d.txt

解压缩 unrar
unrar x abcd.rar

3、时间、日历
显示时间
data

显示年月日
date +%Y-%m-%d
date +%F
显示时分秒
date +%H:%M:%S
date +%T

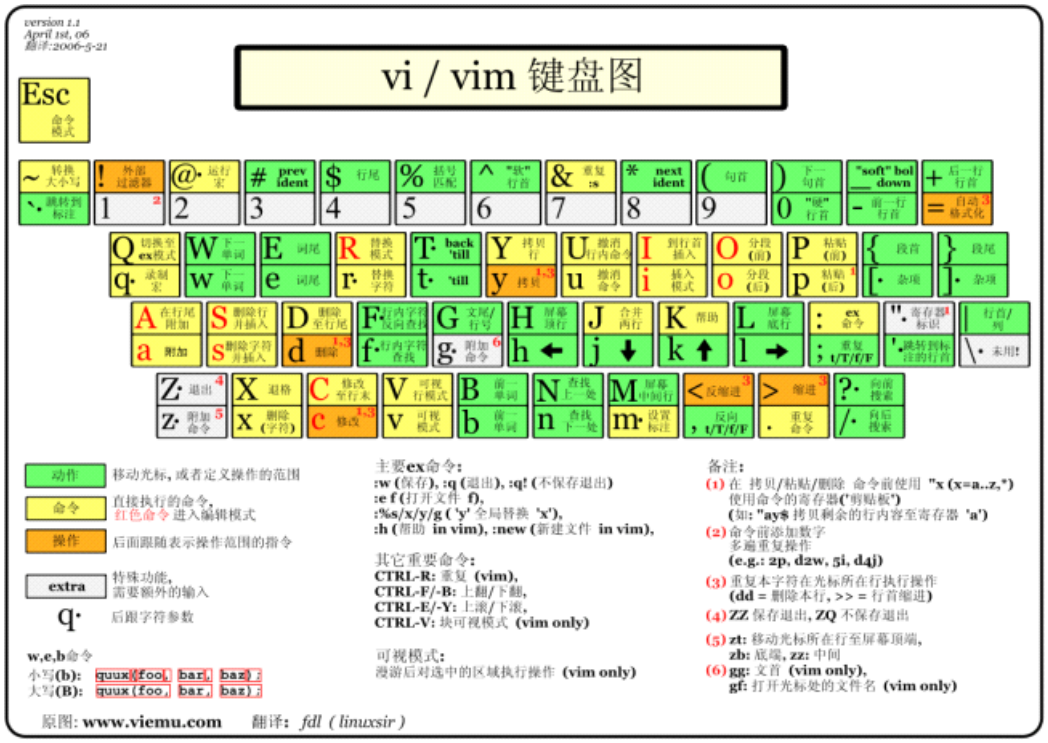
查看日历 cal

4、其他操作
shutdown -h 时间 在何时关机
reboot 重启
clear 清除(ctrl l)

man 指令 查看手册

4.2 vim指令

Vim 是从 vi 发展出来的一个文本编辑器。代码补全、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。



基本上 vi/vim 共分为三种模式，**命令模式（Command Mode）、输入模式（Insert Mode）和命令行模式（Command-Line Mode）**。

命令模式

用户刚刚启动 vi/vim，便进入了命令模式。
此状态下敲击键盘动作会被 Vim 识别为命令，而非输入字符，比如我们此时按下 `i`，并不会输入一个字符，`i` 被当作了一个命令。

以下是普通模式常用的几个命令：

- `i` -- 切换到输入模式，在光标当前位置开始输入文本。
- `x` -- 删除当前光标所在处的字符。
- `:` -- 切换到底线命令模式，以在最底一行输入命令。
- `a` -- 进入插入模式，在光标下一个位置开始输入文本。
- `o`：在当前行的下方插入一个新行，并进入插入模式。
- `O` -- 在当前行的上方插入一个新行，并进入插入模式。
- `dd` -- 剪切当前行。
- `yy` -- 复制当前行。
- `p`（小写） -- 粘贴剪贴板内容到光标下方。
- `P`（大写）— 粘贴剪贴板内容到光标上方。
- `u` -- 撤销上一次操作。
- `Ctrl + r` -- 重做上一次撤销的操作。
- v 可视模式
- V 可视行
- Ctrl v 可视块
- `:w` -- 保存文件。
- `:q` -- 退出 Vim 编辑器。
- `:q!` -- 强制退出Vim 编辑器，不保存修改。

若想要编辑文本，只需要启动 Vim，进入了命令模式，按下 `i` 切换到输入模式即可。
命令模式只有一些最基本的命令，因此仍要依靠**底线命令行模式**输入更多命令。

输入模式

在命令模式下按下 `i` 就进入了输入模式，使用 `Esc` 键可以返回到普通模式。
在输入模式中，可以使用以下按键：

- **字符按键以及Shift组合**，输入字符
- ENTER，回车键，换行
- BACK SPACE，退格键，删除光标前一个字符
- DEL，删除键，删除光标后一个字符
- **方向键**，在文本中移动光标
- HOME/END，移动光标到行首/行尾
- Page Up/Page Down，上/下翻页
- Insert，切换光标为输入/替换模式，光标将变成竖线/下划线
- ESC，退出输入模式，切换到命令模式

底线命令模式

在命令模式下按下 `:`（英文冒号）就进入了底线命令模式。
底线命令模式可以输入单个或多个字符的命令，可用的命令非常多。
在底线命令模式中，基本的命令有（已经省略了冒号）：

- :w：保存文件。
- :q：退出 Vim 编辑器。
- :wq：保存文件并退出 Vim 编辑器。
- :q!：强制退出Vim编辑器，不保存修改。
- :? 查找模式
- :/ 查找模式
- :s/替换内容/替换后内容
- :s/替换内容/替换后内容/g 全部替换（本行）
- :s/替换内容/替换后内容/g 全部替换（整个文件）
- :bp 查看下一个文件
- :bn 查看上一个文件

按 `ESC` 键可随时退出底线命令模式。

vim a.txt passwd 同时进入两个文件，使用底线命令模式下使用 bp bn 查看下一 或上一文件

配置vim格式（全局、个人）

全局在根目录下配置，在etc文件下存储配置

Linux 用户和用户组管理

五、Linux 用户和用户组管理

涉及到多用户，就涉及到权限管理，为了简化权限管理，又出现了用户组。
在创建用户时，如果没有写明所属用户组，默认就会按照用户名自动创建用户组（名字与用户名相同），是主组，除了主组，还有附加组。
存储用户信息的文件在配置文件中，/etc/passwd
存储用户组信息的文件:/etc/group
存储密码的文件：/etc/shadow

新建的用户id 默认从1000开始，然后累加，1000以内是linux自己创建的系统用户

5.1 Linux系统用户账号的管理

1、查看当前用户：whoami
whoami （who am i ，我是谁）

2、添加用户：useradd 组名 用户名
useradd 选项 用户名
参数说明：
选项:
-c comment 指定一段注释性描述。
-d 目录 指定用户主目录，如果此目录不存在，则同时使用-m选项，可以创建主目录。
-g 用户组 指定用户所属的用户组。
-G 用户组，用户组 指定用户所属的附加组。
-s Shell文件 指定用户的登录Shell。
-u 用户号 指定用户的用户号，如果同时有-o选项，则可以重复使用其他用户的标识号。
-gid 指定组id，组id必须存在
用户名:
指定新账号的登录名。

3、修改账号：usermod
sudo usermod -g weihong weihong413 修改weihong413的组为weihong
sudo usermod -u 1002 weihong413 修改用户weihong413的id为1002
sudo usermod -l weihong1 weihong413 修改用户weihong413 的用户名 为 weihong1（home目录下名字不改）

4、用户口令的管理
passwd 选项 用户名
可使用的选项：

-l 锁定口令，即禁用账号。
-u 口令解锁。
-d 使账号无口令。
-f 强迫用户下次登录时修改口令。
如果默认用户名，则修改当前用户的口令。

5、删除

userdel 选项 用户名

常用的选项是 -r，它的作用是把用户的主目录一起删除。

5.2 Linux系统用户组的管理

将user改为group

系统指令、GCC、动态库、静态库

2024年4月13日 20:57

六、系统指令、GCC、动态库、静态库

6.1 Linux系统指令

1、sudo
查看sudo配置文件 sudo vim /etc/sudoers

2、df 查看磁盘使用情况 proc 进程（在内存中，不占磁盘空间，虚拟文件） 挂载点 / 根目录 proc sys

3、free 查看内存使用情况

4、ps 查看进程使用情况
ps -ef e：全部进程 f：全部输出（详细） 输出全部进程

进程头

UID	PID	PPID	C	STIME	TTY	TIME CMD
用户id	进程id	父进程id	进程占用CPU的情况（百分比）	进程开始时间	终端	

系统开始时产生 0号进程id，0号id产生 1号2号id，剩下的所有进程都是由1 2号进程产生的，也就是说，进程也是树状结构

ps aux 显示所有用户的进程

ps ajx 显示进程关系

5、kill 杀死进程

kill 信号 进程id

kill -9 进程id 强制杀死进程

6、top 实时查看进程

7、hostname 查看主机名

8、id 查看用户id

9、ifconfig 查看IP地址

10、uptime 查看系統啓動時間以及負載

11、uname 獲取計算機操作系統相關信息
uname 查看操作系統類型
uname -a 查看全部的系統信息

6.2 GCC

1、gcc 编译.c 文件 生成 .out 可执行文件，

2、预处理

gcc -E test.c -o test.i 预处理 test.c 文件，生成 test.i 文件，预处理后生成 test.i 文件

预处理工作 将加载头文件，、

3、编译

gcc -S test.i 得到一个 test.s 文件 ：汇编文件

4、汇编

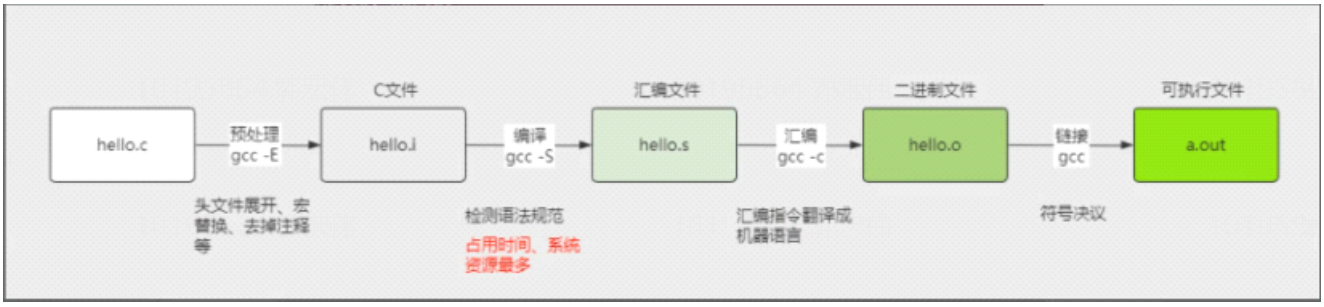
gcc -c test.s 得到一个 test.o 文件（可重定位文件） 已经汇编为机器语言

5、链接

gcc test.o 链接之后得到一个可执行目标文件 .out

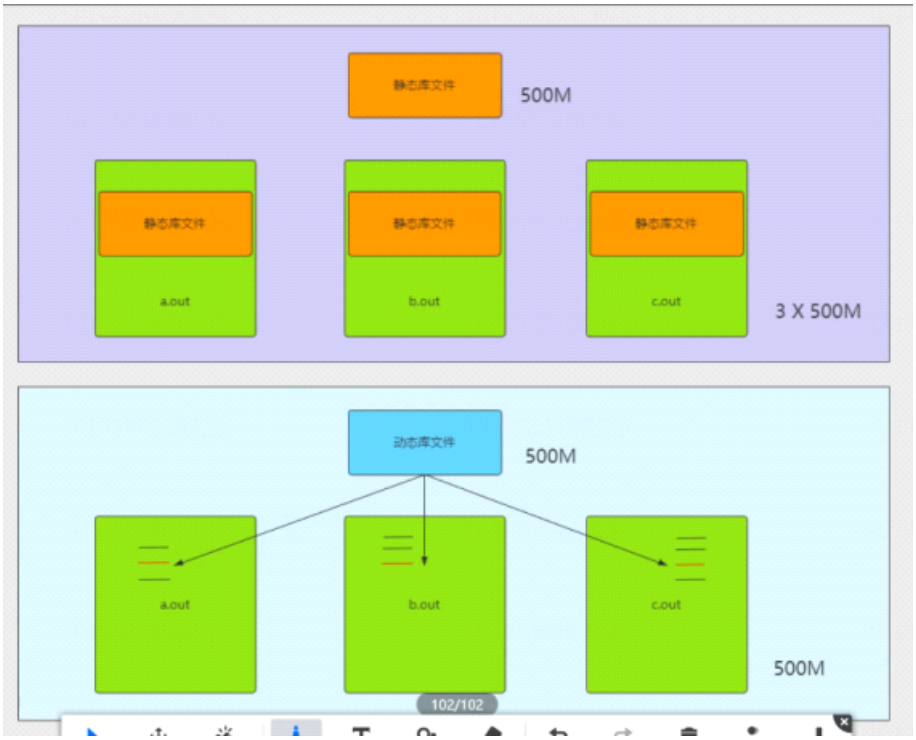
gcc test.o 得到可执行文件，适合多文件链接，例如 3个.c文件：hello.c、a.c、b.c 三个文件经过预处理、编译、汇编、共同链接，生成一个可执行文件.out， 当你要修改hello.c中的内容后，如果修改后只 gcc hello.c，那么a.c、b.c 就不在可执行文件中了，只能再次挨个编译 a.c、b.c ， 如果项目中有很多文件，这样就很麻烦了

因此我们可以拿 .o文件 进行 gcc，（gcc hello.o a.o b.o）,直接将这三个文件进行链接，如果 hello.c 被修改,再重新生成一个 hello.o，然后再和a.o、b.o 进行链接即可，不用再操作其他的文件



6.3 库文件（动态库、静态库）

库文件分为两类，动态库、静态库，可以通过man 查看库文件，不过使用man的时候，需要知道你要查看的库文件在哪个章节， man 章节号 库函数名



静态库：libxxx .a 结尾

如果 a.c b.c c.c 三个文件中都包含了同一静态库，在编译的时候，这个三个文件的静态库都会被编译进去，浪费资源

动态库：libxxx .so结尾

用到的时候，才去动态库查找函数

静态库执行更快，动态库节省空间

1、静态库

静态库的创建：ar rcs 静态库名 要编入的文件

在libdemo文件夹下创建 4个源文件 加法、减法、乘法、测试

将 加法、减法、乘法 编译为可重定位文件，然后编译为静态库
ar rcs libmath.a add.o sub.o mul.o （math是静态库名）

使用静态库

gcc -static test.c -o test -l math -L ./
-static：链接静态库
test.c ： 要链接的源文件
-o test: 生成可执行文件的名字
-l math: 要链接的静态库名字
-L ./ ： 要链接静态库的位置

2、动态库

动态库的创建

gcc -c Add.c -o add.o -fPIC
将 加法、减法、乘法 编译为可重定位文件， -fPIC 生成与位置无关的代码，与静态库区别，静态库是存绝对路径，动态库存相对路径

gcc -shared -o libmath.so add.o sub.o mul.o (创建动态库)

gcc test.c -o test -l math -L ./ （链接生成可执行文件）

动态库的使用

使用可执行文件报错：./test: error while loading shared libraries: libmath.so: cannot open shared object file: No such file or directory 打开动态库的时候报错，没有该文件

通过 ldd test 可以查看哪些库链接到了可执行文件
linux-vdso.so.1 (0x00007ffddff6d000)
libmath.so => not found（自己生成的动态库，没有路径）
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x000070e2a0600000)
/lib64/ld-linux-x86-64.so.2 (0x000070e2a0969000)

静态库是在链接生成可执行文件时直接将整个库复制到了可执行文件。
而动态库是函数执行的时候才会调用动态库中的函数，这个时候 已经 链接完成了，无法找到库中的函数，这个时候就需要告诉动态库的位置，

通过设置环境变量来解决这一问题
export LD_LIBRARY_PATH=./
但 该环境变量的是临时生效的，只在当前终端内，关闭终端后，再次打开就失效了

如果想要环境变量永久有效，应该将其放入配置文件，配置文件有两个位置（根目录/ 和 家目录 ~ ）

家目录
家目录里要放入到 .bashrc 中
vim ~/.bashrc
在该配置文件中加入 export LD_LIBRARY_PATH=./ （即当前目录下）

在配置完成后，还未生效
使用 source ~/.bashrc 使环境变量生效

根目录
根目录要放在 /etc/ld.so.conf 文件下
该配置文件中加入 export LD_LIBRARY_PATH=./ 或者 直接 ./

然后使配置文件生效 sudo ldconfig

根目录下的lib文件内，使存放标准库的，也可以直接将自己的库复制一份放入这个文件内，但一般都不这么做
因为该文件下的都是标准库，自己写的不是标准库

3、动态库和静态库的区别

到这里我们大致了解了静态库和动态库的区别了，静态库被使用目标代码最终和可执行文件在一起(它只会有自己用到的)，而动态库与它相反，它的目标代码在运行加载时链接。正是由于这个区别，会导致下面所介绍的这些区别。

（1）可执行文件大小
从前面也可以观察到，静态链接的可执行文件要比动态链接的可执行文件要大得多，因为它将需要用到的代码从二进制文件中“拷贝”了一份，而动态库仅仅是复制了一些重定位和符号表信息。
（2）扩展性与兼容性
如果静态库中某个函数的实现变了，那么可执行文件必须重新编译，而对于动态链接生成的可执行文件，只需要更新动态库本身即可，不需要重新编译可执行文件。
正因如此，使用动态库的程序方便升级和部署。
（3）依赖原库文件
静态链接的可执行文件不需要依赖其他的内容即可运行，而动态链接的可执行文件必须依赖动态库的存在。所以如果你在安装一些软件的时候，提示某个动态库不存在的时候也就不奇怪了。
即便如此，系统中一般存在一些大量公用的库，所以使用动态库并不会有什么问题.
（4）加载速度
由于静态库在链接时就和可执行文件在一块了，而动态库在加载或者运行时才链接，因此对于同样的程序，静态链接的要比动态链接加载更快。所以选择静态库还是动态库是空间和时间的考量。
但是通常来说，牺牲这点性能来换取程序在空间上的节省和部署的灵活性时值得的

Makefile

2024年4月14日　　17:38

七、makefile

在对源文件进行gcc时，会多次编译文件，如果每次修改就通过gcc来编译的话，比较繁琐。因此我们可以通过makefile来创建一个脚本文件，通过脚本文件来简化操作。

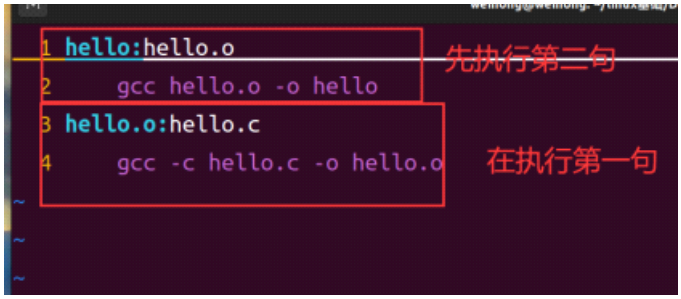
记住1 2 3
1 一个规则：　目标文件：依赖文件　tab　命令
2 两个函数
3 三个自动变量

7.1 一个规则

Makefile的创建
vim makefile

使用时 直接使用 命令 make即可

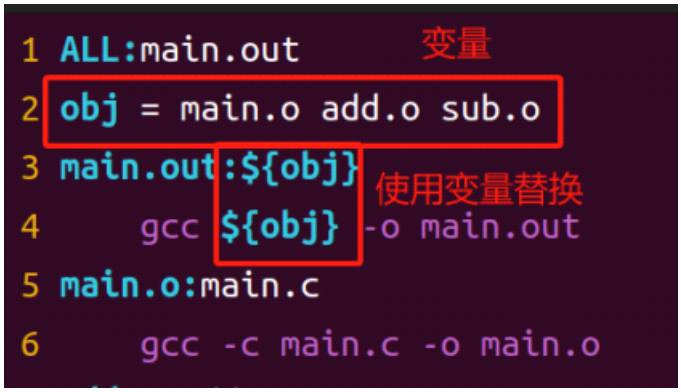
多步骤的makefile创建
先生成可重定位文件，在生成而可执行文件



在makefile中，需要先执行生成可执行文件的命令，再执行生成可重定位文件的命令。这是因为，在执行第一句时，对于第一句的依赖文件 hello.o，因为还没生成.o 文件，他就会执行下面的语句。

make clean -n 查看一下 clean 的命令

makefile文件中创建变量

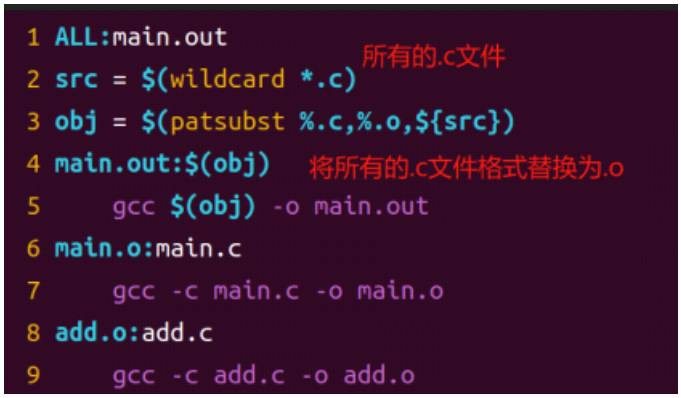


7.2 两个函数

两个函数

makefile文件中创建函数（**优点：后续在添加文件时，无需再修改代码**）

wildcard：通配符，patsubst：替换



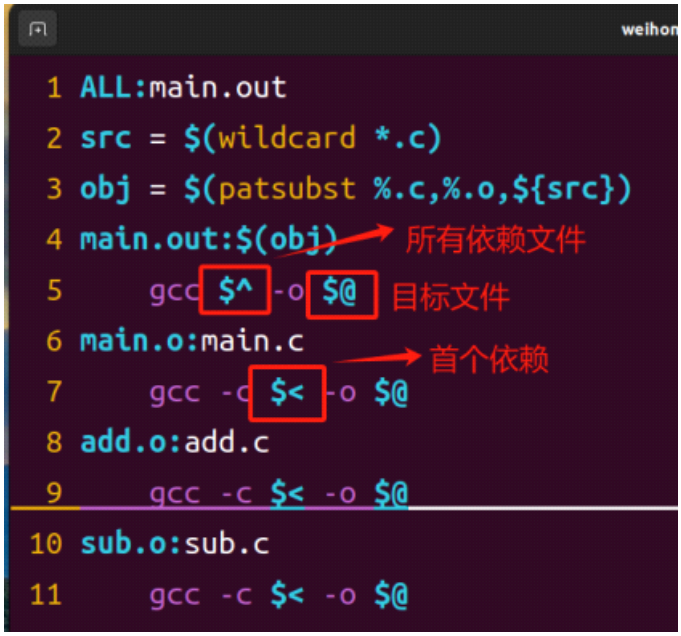
7.3 三个自动变量

三个自动变量

$@ 在规则命令中表示目标

$^ 在规则中的依赖（所有的依赖）

$< 在规则中的依赖（第一个依赖）（在 模式规模中，他会将依赖挨个取出）




```
ALL:main.out
src = $(wildcard *.c)
obj = $(patsubst %.c,%.o,${src})
main.out:$(obj)
    gcc $^ -o $@
main.o:main.c
    gcc -c $< -o $@
add.o:add.c
    gcc -c $< -o $@
sub.o:sub.c
    gcc -c $< -o $@
```

模式规则： $< 会将依赖文件挨个取出， 然后 将.c依赖文件 转为 .o目标文件

```
ALL:main.out
src = $(wildcard *.c)
obj = $(patsubst %.c,%.o,${src})
main.out:$(obj)
    gcc $^ -o $@
%.o:%.c
    gcc -c $< -o $@
.PHONY:clean
clean:
    rm *.o main.out
```

gdb

2024年4月18日　　11:05

八、gdb

gcc过程中用来调试代码

gcc -g main.c -o main.out（通过gcc -g 来生成可执行文件，才能使用gdb）

gdb ./main.out （通过gdb调试生成的可执行文件）

8.1 基本命令

list（l）：查看代码，一次显示10行
l 8 ：　　中间是第8行 上5下4（一共10行）

break 行号（b 行号）： 在某行加断点

info b 或 b ： 查看断点信息

run: 运行到断点通知

next(n): 执行下一条语句

step(s): 进入函数内部

print i: 查看当前 变量i 的值

ptype i: 查看 i 的变量类型

continue(c)： 跳到下一个（次）断点