# linux命令基础习惯

终端：一系列输入输出设备的集合

pwd 查看当前所处的目录

history 查看已经使用过的命令

cat /etc/shells 查看当前系统有哪些shell

cat $SHELL 查看当前系统正在使用的shell

**a.输入命令时的快捷键操作**

键入tab 补全命令

ctrl+a 回到正在输入命令的头部

ctrl+e 回到正在输入命令的尾部

ctrl+u 清空当前正在输入的命令

clear或ctrl+L 将当前终端的信息清空

# 类Unix系统目录

which [命令名称] 查看命令所在的路径位置

cat [文件名称] 查看文件里面的内容，输出到终端。如果cat后面没跟文件名，就读标准输入，遇到换行符之后，输出到标准输出。tac逆序输出文件内容。

**a.linux的主要目录**

bin:存放二进制可执行文件

boot:存放开机启动程序

dev:存放设备文件

home:存放用户

etc:用户信息和系统配置文件

lib:库文件

root:管理员宿主目录(家目录)

usr:用户资源管理目录

**b.linux系统文件类型**

普通文件 -

目录文件 d

字符设备文件 c

块设别文件 b

软连接 l

管道文件 p

套接字s

**c.软连接与硬连接**

1. 区别

在Unix世界里有两个'link'（连接）概念，一般称之为硬连接和软连接。一个硬连接仅仅是一个文件名。（一个文件可以有好几个文件名，只有将最后一个文件名从磁盘上删除，才能把这个文件删掉。文件名的个数是由ls(1)来确定的。所有的文件名都处于同一个状态，也就没有什么“源名字”之说。通常文件系统里的一个文件的所有名字包含着一样的数据信息，不过这样也不是必需的。）一个软连接（或符号连接）是完全不同的：它是一个包含了路径信息的小小的指定文件。因此，软连接可以指向不同文件系统里的文件（比如由NFS装载的不同机器文件系统上的文件），甚至可以指向一个不一定确实存在的文件。

1. 软连接

ln -s src\_file des\_file 建立软连接

软连接文件的大小含义是:执行的的源文件的路径大小

为保证软连接在任何目录下能够进行访问，在创建软连接的时候，源文件的路径要使用绝对路径。当软连接的源文件不存在时，软连接就会失效。

1. 硬连接

ln src\_file des\_file 建立硬连接

操作系统给每个文件赋予唯一的INode,多个硬连接文件具有相同INode节点,当有相同的INode文件存在时，彼此同步。

删除时，只将硬链接计数-1，当减为0时，源文件将被删除。

# 1.3创建修改用户和用户组

whoami 查看当前登陆用户

**a.修改文件属性**

chmod [u(用户) g(用户组) o(其他人)] [+(添加权限) -(减少权限)] [r(可读) w(可写) x(可执行)] 修改文件的属性

chmod [用户权限的值][用户组权限的值][其他人权限的值] 通过指定数值修改文件的属性

**b.修改用户属性**

sudo adduser [用户名称] 创建新的用户

sudo deluser [用户名称] 删除用户

sudo chown [用户名称] [文件名] 更改文件的所属用户

sudo chown [用户名称]:[用户组名称] 更改文件的所属用户及用户组

**c.修改用户组属性**

sudo addgroup [用户组名称] 创建新的用户组

sudo delgroup [用户组名称] 删除用户组

sudo chgrp [用户组名称] [文件名] 更改文件的所属用户组

# 1.4查找与检索

**a.find命令**

find [路径] -maxdepth [深度值] -type [‘文件类型符号’] 按类型查找文件，查找普通文件时使用的类型符号是f,而不是-

find [路径] -name [‘文件名过滤模式’] 按文件名查找文件

find [路径] -size [+(大于) -(小于)] [size值] 按文件大小查找文件 单位:k,M,G

-atime(访问时间) -mtime(属性修改时间) -ctime(内容改变时间) 时间都是以天为单位

-amin -mmin -cmin 时间是以分钟为单位

-exec ls -l {}\; 将搜索到的内容进行统一的执行处理

-ok ls -l{}\; 以询问的方式决定对每个find的结果进行处理

-xargs 将find搜索结果集执行某一命令。当结果集数量过大时，可以分片映射 e.g:find /usr/ -name “\*temp\*” | xargs ls -ld

**b.grep命令**

grep -r [搜索关键字] [搜索路径] -n 搜索文件内容 e.g:grep -r ‘copy’ ./ -n

ps aux | grep ‘cupsd’ 检索进程结果集

# 1.5软件安装与卸载

1. **软件安装**

联网->

更新软件资源列表到本地 sudo apt-get update ->

安装软件 sudo apt-get install 软件名

sudo dpkg -i 安装包名 使用软件包进行安装(.deb)

1. **软件卸载**

sudo apt-get remove 软件包名

sudo dpkge -r 软件包名

**1.6压缩与解压**

1. **gzip和bzip2**

gzip用来对单个文件进行压缩，不能对目录或多个文件进行压缩，解压时使用gunzip

bzip2和gzip同理，只是采用的压缩方式不同

1. **tar**

tar [-z(gzip)c(create)v(visual)f(file)] [压缩包名] [欲压缩的文件或目录] 进行多个文件或目录的压缩，使用gzip压缩方式

tar [-j(bzip2)c(create)v(visual)f(file)] [压缩包名] [欲压缩的文件或目录] 采用bzip2方式进行压缩

将压缩命令中的-c换成-x就编程对应的解压命令,

tar [-z(gzip)x(extract)v(visual)f(file)] [压缩包名] 使用gzip方式进行解压

tar [-j(bzip2)x(extract)v(visual)f(file)] [压缩包名] 使用bzip2方式进行解压

1. **rar**

rar a -r [压缩包名] [欲压缩的文件或目录] 压缩命令

unrar x [压缩包名] 解压命令

1. **zip**

zip -r [压缩包名] [欲压缩的文件或目录] 压缩命令

unzip [压缩包名] 解压命令

**1.7其他命令**

1. **man**

卷1:命令 卷2：系统调用 卷3：库函数 卷5：文件内容格式 卷9:内核例程

1. **clear**

清楚终端的显示内容，快捷键ctrl+l

1. **echo**

回显

1. **alias**

取别名，如:alias pg=’ps aus | grep’

1. **umask**

掩码+文件权限=777

新创建的文件不具备x权限

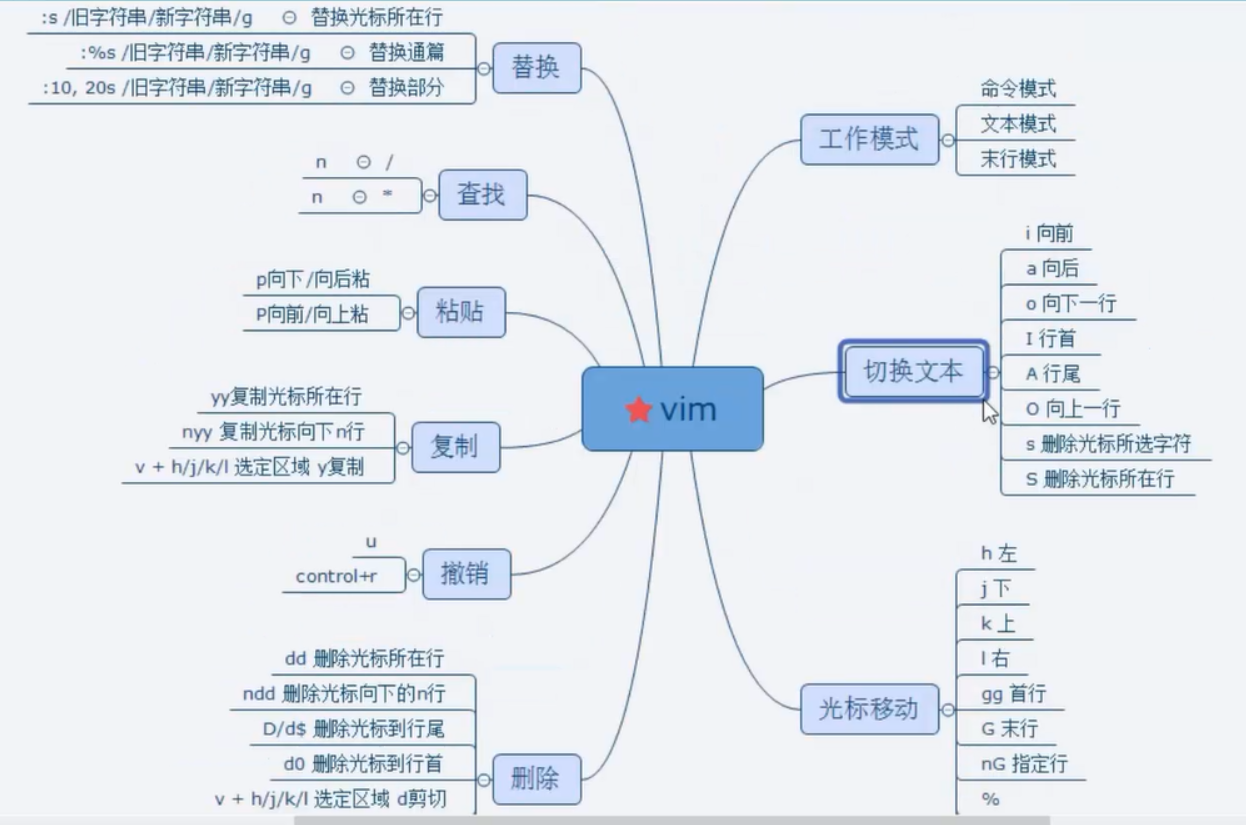
目录无x权限无法进入

1. **创建终端**

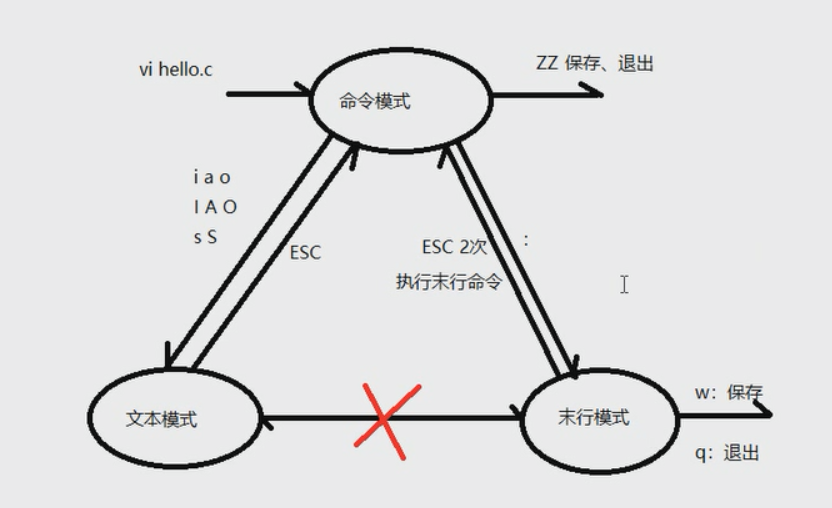
ctrl+alt+t 新建一个终端窗口

ctrl+shift+t 在一个终端窗口中新建分页

**2.1vim**



**a.vim的三种工作模式**



**b.vim基本操作**

1.进入文本模式

插入光标前一个字符 i

插入行首 I

插入光标后一个字符 a

插入行尾 A

向下新开一行，插入行首 o

向上新开一行，插入行首 O

删除光标所在的字符 s

删除光标所在的行 S

2.跳转

命令模式下:

跳转到指定行：行号+G

跳到文件头部：gg

跳到文件尾部：G

跳到行首：0

跳到行尾：$

末行模式下:

跳转到指定行：:+行号

1. 格式化代码

命令模式下：gg=G

1. 括号(包括小括号、中括号、大括号)跳转

命令模式下：%

1. 删除文本

命令模式下:

删除光标所在的单个字符： x

删除光标所在的单词，光标必须位于单词的首字母： d+w

删除光标至行尾的内容： D 或d+$

删除光标至行首的内容： d+0

删除光标所在的行： d+d

删除指定区域：按v切换为“可视模式”,使用hjkl挪移光标来选择待删除区域，之后按d删除

删除指定的多行：光标置于待删除多行的行首，行数+dd进行删除

1. 复制与粘贴

复制命令：y

复制一行：yy

前向粘贴：p

后向粘贴：P

1. 查找

自提供关键字：命令模式下，/+关键字+回车，下一个关键字的查找使用n,上一个关键字的查找使用N

已存在关键字：命令模式下，光标位于关键字任一个字符中，使用\*查找下一个关键字，使用#查找上一个关键字。

1. 替换

光标所在的字符进行替换：r

单行替换：光标置于待替换的行上，进入末行模式, :s /欲替换的内容/替换内容/g，/g是将该行的所有匹配内容进行替换，不加/g则只会替换该行首个匹配的关键字。

整篇替换：末行模式，:%s /欲替换的内容/替换内容/g

指定多行替换：末行模式, :开始行号,结束行号s /欲替换的内容/替换的内容/g

1. 其他命令

撤销：命令模式下，u

反撤销：命令模式下，ctrl+r

分屏:末行模式下，:sp 进行横分屏 :vsp 进行竖分屏

命令模式下, ctrl+w+w 进行不同的分屏之间的切换

跳转到帮助手册：命令模式下,光标置于待查看的关键字上，卷编号+K

查看宏定义：命令模式下，光标置于待查看的关键字上，[+d

末行模式下执行shell命令：:!+shell命令

1. 配置

默认vim特性的配置目录在/etc/vim/vimrc,用户可以自己创建~/.vimrc的隐藏文件，用户自己创建的配置文件优先级比较高。

**2.2 gcc**

**a.gcc编译4步骤**



**b.gcc编译常用参数**

查看gcc版本号: -v /--v/--version

指定头文件: -I目录

只进行编译，生成对应的.o文件，不进行链接:-c

包含调试信息:-g

编译优化，n越大优化得越多:-On n=0~3

提示更多警告信息:-Wall

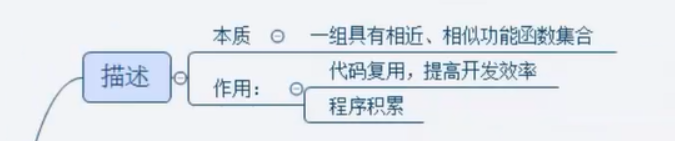
生成预处理文件:-E

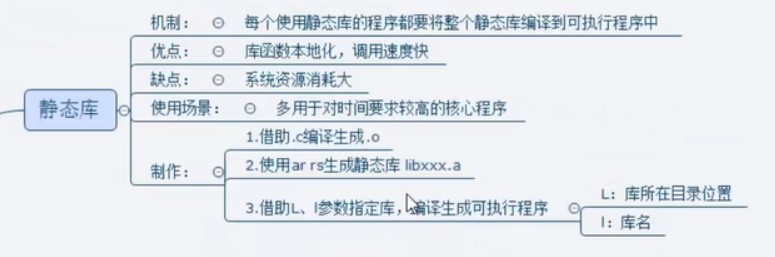
生成.c文件与头文件依赖关系用来Makefile,包括系统库的头文件:-M

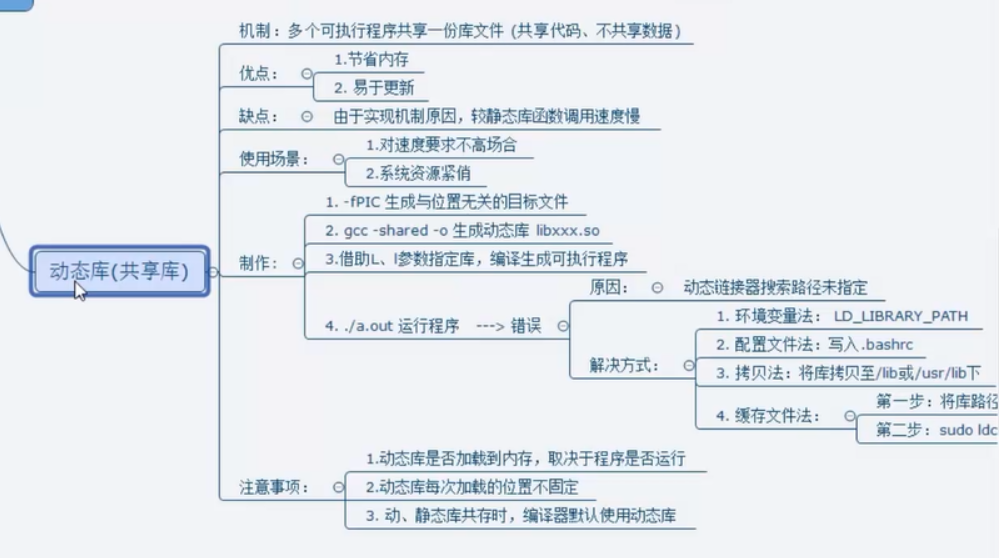
生成.c文件与头文件的依赖关系用来Makefile，不包括系统库的头文件:-MM

向程序中动态注册宏定义:-D

**2.3库文件**







1. **静态库**

1.创建静态库:先生成.o文件，之后利用ar rcs libname.a \*.o来生成

2.编译引用静态库:编译时和代码一起附加生成可执行文件，gcc hello.c libmath.a -o hello

1. **动态库**

1.创建动态库:

生成.o文件，由于动态库中的内容是在程序运行需要时才进行加载，内存地址进行延迟绑定，所以生成的代码需要与位置无关。gcc -c libsrc.c -o libsrc.o -fPIC

之后使用gcc -shared进行动态库的生成:gcc -shared -o libmymath.so add.o sub.o

1. 编译可执行文件

生成可执行文件时需要指定动态库，-l 指定动态库名(不包含前缀lib和后缀.so) -L指定动态库所在的路径

gcc hello.c -o hello -l mymath -L ./lib -I ./inc

1. 动态链接器库目录指定

直接运行可执行文件出错，

原因:

1. 链接器 工作于链接阶段，需要-l和-L指定动态库
2. 动态链接器 工作于程序运行阶段，工作时需要提供动态库所在的位置

解决:

1. 终端配置文件修改:在~/.bashrc终端配置文件中添加:export LD\_LIBRARY\_PATH=绝对库路径，之后通过. .bashrc或source .bashrc或重启终端来使.bashrc的配置修改生效。
2. 系统配置文件修改:sudo vi /etc/ld.so.conf-->写入动态库绝对路径-->sudo ldconfig -v 使配置文件生效-->ldd hello.out 可以查看引用的动态库信息，如果没有找到动态库，会出现 “not found”

**2.4 Makefile**

**文件名称**:Makefile或makefile

**规则**:

目标:依赖条件

（tab）命令

**原则**:1、若想生成目标，检查规则中的依赖条件是否存在，如不存在，则寻找是否有规则生成该依赖文件。

2、检查规则中的目标文件是否需要更新，必须先检查它的所有依赖，依赖中有任一个被更新，则目标必须更新。

**语义**:

1. 默认makefile中只完成第一个目标，ALL:target指定makefile文件中要完成的最终目标。
2. src=$(wildcard \*.c) 匹配当前工作目录下的所有.c文件，将文件名组成列表，赋值给src
3. obj=$(patsubst %.c %.o $(src)) 将src列表中的.c替换成.o
4. clean:(没有依赖)

-rm -rf $(obj) a.out rm前边的”-”表示删除不存在的文件时，不会产生错误，仍继续命令的执行。

1. 自动变量

$@:表示规则中的目标

$<:表示规则中的第一个条件

$^:表示规则中的所有条件，组成一个列表，以空格隔开，列表中有重复项则消除重复项。如果将该变量应用在模式规则中，它可将依赖条件列表中的依赖依次取出，套用模式规则。

1. 模式规则

%.c:%.o

gcc -c $< -o $@

1. 静态模式规则

表示以$(obj)为依赖条件的规则如何找到匹配的生成规则

$(obj):%.c:%.o

gcc -c $< -o %@

1. 伪目标

如果在文件目录中有同名文件，则make相应的操作会失效。使用伪目标表示不管是否有同名文件，规则中的命令条件都会进行执行。

.PHONY: clean ALL