# Linux基础说明

## 常见Shell介绍：

/bin/sh     （被bash取代）

/bin/bash （linux默认shell）

/bin/ksh    ( kornshell 由AT&T bell lab 发展出来的）

/bin/tcsh  （整合C shell 提供更多功能）

/bin/csh    （已经被/bin/tcsh所取代）

/bin/zsh    （基于ksh开发，功能更强）

## shell文件功能

/etc/profile 对所有用户有效，用户登录时文件执行

/etc/bashrc 对所有bash shell有用

~/.bash\_profile 是交互式、login 方式进入 bash 运行的  
~/.bashrc 是交互式 non-login 方式进入 bash 运行的

所以通常前者会调用后者。

source /etc/profile 执行命令

**------当用户使用的时bash shell时**

~/.bash\_profile | ~/.bash\_login

Unix是.profile

登录的时候执行一次，之后会调用.bashrc，一般会在.bash\_profile文件中显式调用.bashrc。

~/.bashrc 当前用户的专用shell,每次打开shell都会调用

~/.bash\_logout 退出时执行

**------**

~/.zshrc zsh的配置文件

~/.xinitrc 桌面启动文件xorg

~/.Xdefault zsh样式文件

~/.Xres

对于特定用户，还可以在~/.bashrc、~/.xinitrc或~/.xprofile中设置自己的用户环境。不同之处在于：

.bashrc: 每次终端登录时读取并运用里面的设置。

.xinitrc: 每次startx启动X界面时读取并运用里面的设置

.xprofile: 每次使用gdm等图形登录时读取并运用里面的设

## 目录功能介绍

/             根目录

/bin        系统可执行程序

/boot      内核和启动程序相关文件

/lib          库文件，系统最基础的动态共享库

/media    挂载U盘光驱等文件

/mnt        给用户提供挂载其他文件系统

/usr         unix system res 庞大复杂的目录，很多程序会安装到这里

/sbin        root用户的可执行文件

/proc       系统内存映射，会有程序运行的一些信息

/etc          系统软件的配置文件目录

/opt 可选择目录，用户自己安装的软件

/var 常态变动文件

## 环境变量

$HOME 环境变量，当前用户的夹目录

## 重定向

dup(fd[1], STDOUT\_FILENO) //输出重定向到fd[1]

# Linux命令

## 目录相关命令

#切换目录 cd /home

显示上一次目录 cd -

显示当前目录 pwd

**mkdir命令**

mkdir -p xx/aa # 递归创建目录

mkdir aa bb # 创建2个目录(aa,bb)

--------------------------------------------------------------------------------------------

显示目录结构，需要安装 tree

**rmdir命令**

rmdir /etc/home # 删除/etc/home目录

rmdir -p /etc/home # 递归删除目录

-------------------------------------------------------------------------------------------

查看命令在哪个目录 which

## 文件相关命令

创建文件(如果文件已存在则更正文件创建时间) touch

**rm命令**

rm -r # 递归删除目录

rm -f # 强制删除

rm -rf # 递归强制删除

------------------------------------------------------------------------------------

复制 cp

移动 mv

读取文件（比较小的文件） Cat

**tail命令**

Tail -n 5 # 查文件最后5行内容

Tail -f # 实时显示显示文件添加内容

----------------------------------------------------------------------------------------

**head命令**

head /etc/passwd # 查看前几行 默认显示10行

Head -n 4 /etc/passwd # 显示4行

--------------------------------------------------------------------------------------

查看文件（回车一行，空格一页） more /etc/passwd

查看文件（分屏查看上下方向键） less /etc/passwd

## 统计

**wc命令**

wc -l # 显示有多少条命令

wc -w

wc -c

显示文件信息 du

显示系统空间 df -h

## 文件权限

- rw- r-- r-- 1

文件类型 归属用户 归属组 其他 硬连接数

改变文件权限 chmod

修改文件所属 chown a:b a.out # 用户a ; b组

chown group file 改变file文件的组为group

chown :user file 改变归属组

chown user: file 改变用户

Chgro 修改用户组权限

## 查找命令

**Find命令**

Find / -name # 按名字查找

Find / -type # 按类型查找

-f普通文件 -d目录文件 -l 连接文件 -b设备文件 -c字符设备 -p管道设备 -s本地套接字

Find / -size +1M -5M # 按大小查找，不写+-号，就是等于

-maxdepth 1 # 查找深度

## 用户命令

**useradd命令**

useradd -s # 指定shell

useradd -g # 指定组

useradd -d # 用户家目录

useradd -m # 家目录不存在自动创建

-----------------------------------------------------------------

添加用户组 groupadd Tree

修改密码 passwd Tree

## 其他命令

查看历史输入命令 history

查看命令说明 man

**ls命令**

ls -l # 查看文件详细信息

ls -a # 显示所有文件

-r

ls -lrt # 按照时间进行排序

ls \*.a # ls 允许输入正则

ls func.? #

--------------------------------------------------------------------------------------

**ln命令**

ln -s软连接

使用ls会看到箭头 可创建目录，

Ln 硬连接，会增加计数，硬连接计数为0，会删除文件夹。不能创建目录硬连接 引用同一块内存（磁盘快）

删除硬连接数 unlink /etc/passwd

-----------------------------------------------------------------------

返回当前登录的用户 whoami

-ok 安全模式

-exec

| xargs

| greg 内容过滤

Greg -rn

Greg -v 排除

# GCC编译

## 编译流程说明

初始文件 ：***hall.c***

gcc **-E** hall.c

|--> 生成 hallo.i文件

|--> 作用：**预定义、头文件展开、宏替换**；

gcc **-S**  hall.i

|-->生成 hallo.s文件

|-->作用：**生成汇编代码**

gcc **-c** hallo.s

|--> 生成hallo.s文件

|--> 作用**：生成可执行文件**

## gcc编译参数

**命令**

gcc add.c // 将直接生成add.out文件，可直接执行

gcc **-lstdc++**  // 编译c++程序

|  |  |
| --- | --- |
| -I ./include/www | 指定连头文件的目录./include/www |
| -o | 生成指定名称的文件 |
| -D 200 | .c 文件中**宏定义DEBUG**值为 200 |
| -g | 用于调试 |
| -wall | 警告 |
| -lxxxx | 连接库libxxxx.so，xxxx是库名与l连接在一起 |
| -L | 库文件目录 |

## 静态库文件(不全)

libxxxx.a--》lib.so

gcc -c add.c

gcc -I ../inculud/ed

ar res libxxxx.a \*.o

## 动态库（不全）

-c -o生成指定文件

-fpIc 与位置无关

Gcc -shared 打包成.so文件

## ldd 检查连接命令

**方法一：**将库文件拷贝到/lib /usr/lib

**方法二：**导入环境变量

export LD\_LIBRARY\_PATH=/lib:$LD\_LIBRARY\_PATH //新加环境变量

**方法三：修改ld.so.conf文文件**

vi /etc/ld.so.conf //打开文件,添加库文件目录

ldconfig //更新动态库

# Makefile

## makefile命名规则

makfile makefile

## 三要素

最简单：

app：main.c

gcc -o app -I ./include main.c

执行make命令

===

## 使用参数

**变量使用**

Objfile=main.o add.o sub.o div.o # 注释

app:$(objfile) # 使用变量

gcc -o app -I ./include main.o add.o sub.o

main.o : main.c

gcc -c main.c -I ./include

## 编译指定目标

SrcFiles=$(wildcard \*.c) //使用函数wildcard 获取当前所有.c文件

Test:

echo $(SrcFiles)

make tast //编译指定目标

objct=$(patsubst %.c,%.o,$(SrcFile)) //将srcfile文件变成.o文件

## 变量：

$@ 目标 $^全部依赖 $<第一个依赖 $?第一个变化的依赖

%.o: %.c

gcc -c $< -I ./indlcude/

## 定义一个可以删除生成文件功能

.PHONY:clean # 防止有歧义

clean:

rm \*.o

-rm \*.o //错误继续执行

@rm \*.o

# GDB调试

gdb app

run 运行

start 分步调试 next n step可以进到函数内部

quit q 退出

set args 10 20 设置传递参数10 20

list 显示 break 17设置17行断点 run会直接停到断点

b sum 停留在sum函数

b main.c:20

info b 查看断点

d 1 del 1 删除断点

c 跳到下一个断点

p argc 打印变量 /也可以打印结构体

ptype i //查看类型

display a //追踪a的值

undisplay

info display

p/o sb.st\_moe //转16进制

# linux系统函数

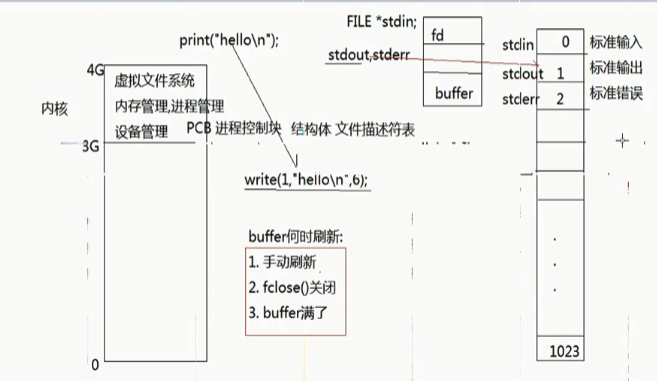
## IO操作

默认的三个宏定义，这3个默认打开

STDOUT\_FILENO 标准输出

STDIN\_FILENO 标准输入

STDERR\_FILENO 标准错误



#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

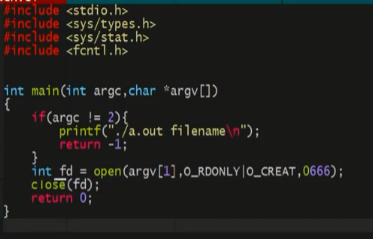
int open(const char \*filename, int flag, mode\_t mode)

filename: 路径+文件名

flag：O\_RDONLY只读，O\_WRONLY只写，O\_RDWR读写

O\_APPEND追加 O\_CREAT创建--O\_EXCL如果文件以存在，创建报错失败

mode 权限位



#include<unistd.h>

close()

#include<unistd.h>

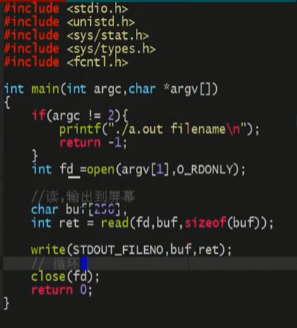
ssize\_t read(int fd, void \*buf, size\_t count)

fd:open打开的， buf：缓冲区， count：缓冲区大小 -1失败 0读到文件末尾

#include<unistd.h>

ssize\_t write(int fd, const void \*buf, size\_t count)

fd :open打开， buf：缓冲区， count：缓冲区大小 -1失败使用strerror（）， 0未写入



lseek移动文件的读写位置

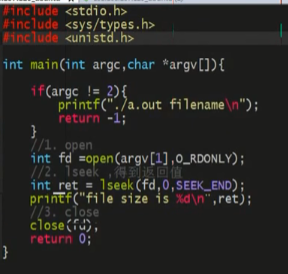
off\_t lseek(int fd, off\_t offset, int whence)

fd 文件描述符 offset：

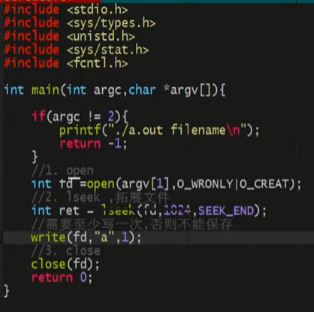
whence :SEEK\_SET 开始位置，SEEK\_CUR当前位置，SEEK\_END末尾；

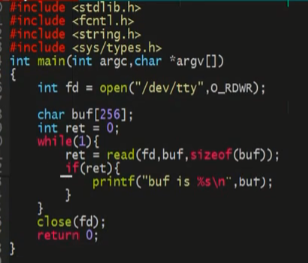
成功返回当前位置到开始的长度 失败-1

计算文件大小



扩展文件--先建文件，占空间







perror(“sssss”);这会直接打印出错误，ssss后添加

//设置非阻塞

**fcntl()函数设置阻塞**



#include<sys/type.h>

#include<sys/stat.h>

#include<unistd.h>

// 获取文件信息-文件属性

int stat(const char \*pathname, struct stat \*buf)

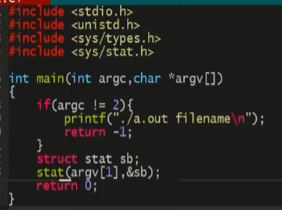
pathname 文件名，路径

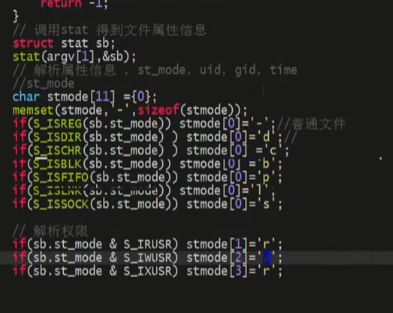
buf 传出参数

struct stat ssb ; &ssb;

成功返回0， 失败返回-1，设置errno

还有一个stat 命令





#inclue<pwd.h>

获得用户名

struct passwd \*getpwuid(uid\_t uid); 获取用户名需要uid

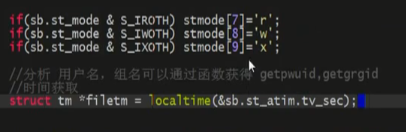
获取组

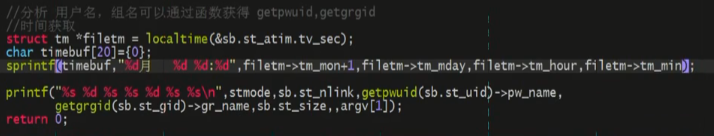
struct group \*getgroup(gid\_t gid);

#include<time.h>

struct tm \*localtime(const time\_t \*timp) //获得本地时间







lstat与stat区别，stat碰到链接会追塑源文件

判断用户是否有权限，文件是否存在，判断有效用户，当前用户能看到的权限

int access(const char\* pathname, int mode)

pathname:文件

mode:

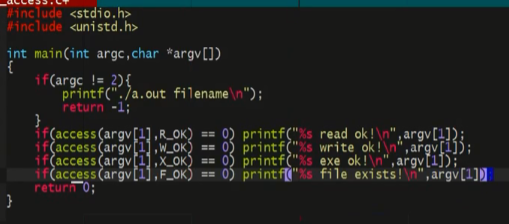
R\_OK 是否有读权限

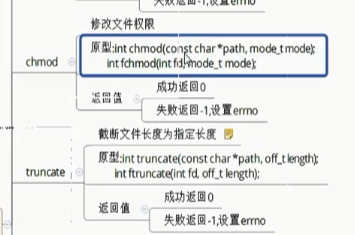
W\_OK 写权限

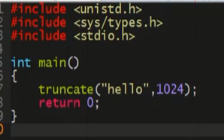
X\_OK 可执行权限

F\_OK 文件是否存在

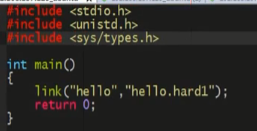
返回值有权限，文件存在 0，失败返回-1 设置error

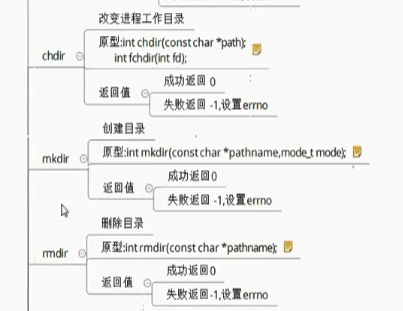




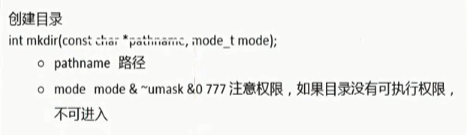


创建硬连接

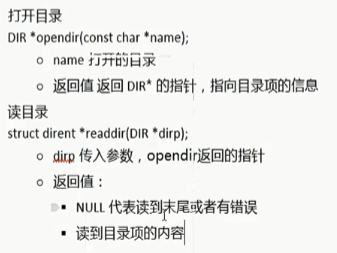


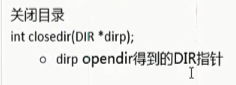






打开目录







## errno



## dup复制文件描述符

int dup(int oldfd)

新返回一个文件描述符

## dup2重定向文件描述符

int dup2(int oldfd, int newfd)

