# 一、基于虚拟技术的Linux安装

## 1. 新建虚拟机

1. 安装位置剩余空间最少20G
2. 虚拟内存选择小于等于2G
3. 网络连接-----桥接
4. 虚拟磁盘类型IDE
5. 虚拟磁盘拆分成多文件、适中

## 2. 修改虚拟机设置

1. 移除 autoinst.iso光驱、floppy（软盘）、打印机等
2. 添加串口Serial Port
3. 创建共享文件夹

## 3. 安装设置

【注】：（命令）linux rescue #用于系统修复

1. 检测安装介质、安装号码 -------跳过
2. 创建分区

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型 | 大小 | 作用 | 挂载点 |  |
| swap | 内存2倍 | 虚拟内存 |  | 交换分区 |
| ext3 | 100M | 引导分区 | /boot |  |
| ext3 | 剩余全部 | 根 | / |  |

1. 软件选择“软件开发”“网络服务”“现在定制”
2. 防火墙-----禁用
3. SELinux-----允许

键盘组合键：<Ctrl +Alt+Backspace> ------- 作用：重启X-windows

# 二、系统的优化与基本设置

## 1. 安装虚拟机工具（VMware Tools）

1. 挂载安装光盘
2. 进入光盘目录 [root@localhost ~]# cd /media/
3. 将VMware Tools 解包到 /usr/local/src

tar –zxvf VMwaretools[tab][tab] –C /usr/local/src

d.执行安装脚本 \*.pl

## 2. 修改终端下显示模式

a．打开/etc/grub.conf

b. 在kernel行尾加入 vga=791

## 3. 修改系统默认运行级别

a．打开/etc/inittab

b. 将initdefault行中的5改为3

【注】修改完2、3文件后重启，系统将默认进入文本状态

**Linux的运行级别**

***0*** *关机*

**1** *单用户无需登录直接获得root权限（系统维护）*

**2** *多用户、无网络（排除网络故障）*

**3** *多用户、文本模式（系统、网络管理）*

**4** *自定义*

**5** *图形（桌面用户）*

**6** *重启*

## 4. 关闭冗余服务

在终端下执行 ntsysv

|  |  |  |
| --- | --- | --- |
| 保留服务 | haldaemon | 图形界面支持 |
| kudzu | 即插既用设备 |
| messagebus | 图形界面支持 |
| network |  |
| nfs | nfs服务（网络文件系统可以将服务器上的目录共享给远程的计算机，并在远程计算机上挂载） |
| protmap |
| vmware-tools |  |
| vmware-tal。。。。 |  |
| xfs | 超级服务器---可以管理很多不能独立运行的网络服务 |
| xineted |

# 三、文件系统

|  |  |  |  |
| --- | --- | --- | --- |
|  | 目录 | 作用 | 备注 |
| / | boot | Linux内核、引导程序（grub、lilo） | 内核的作用：  文件管理、内存管理、进程管理、设备管理 |
| bin | 普通用户可以执行的命令 |  |
| sbin | 只有超级用户可以执行的命令 |  |
| etc | 配置文件目录 |  |
| /etc/rc\* | rc表示Runlevel Changes（运行级别） |
| proc | 内核启动映像 | 观察内核的运行状态、微调内核的一些参数 |
| sys | 系统总线映射 |  |
| usr | 第三方软件目录 | /usr/include C头文件 |
| root | 超级用户专属目录 |  |
| home | 普通用户家目录 |  |
| media | 光盘挂载目录 |  |
| mnt | 外部存储设备挂载目录 |  |
| tmp | 临时文件目录 |  |
| var | 日志（网站）等需要频繁访问的文件 |  |
| lib | C库目录 |  |
| dev | 设备文件目录 | 每个文件对应一个设备，而设备可能存在，也可能不存在 |
| /dev/ttyn | 物理终端 |
| /dev/ptsn | 虚拟终端 |
| /dev/zero | 只读 ——0 |
| /dev/null | 只写 ——丢弃 |
| /dev/random | 只读 ——随机数 |
| /dev/consde | 操作系统所必须，关系到系统登录 |

# 四、常用操作

1. <Alt+Fn> n=1~7 ——物理终端间切换
2. 文本下启动图形界面 startx
3. 退出图形界面 <Ctrl+Alt+Backspace>
4. 图形界面下运行文本<Ctrl+Alt+Fn> n=1~6
5. 文本进入图形 <Alt+F7>
6. 终止程序运行 <Ctrl+C>
7. 执行程序运行命令时，在末尾家&——表示让程序在后台运行
8. 作业调度

jobs #查看作业号

fg 作业号 #将后台作业调入前台

<Ctrl+Z> 表示挂起前台作业

bg 作业号 #将挂起作业调入后台

1. <Ctrl+L> 清屏
2. history #显示历史命令 默认最多1500条
3. ! 历史命令编号 #重新执行该命令
4. 历史相似命令
5. <Ctrl+r> 注：在宿主机即虚拟机外将重启虚拟机
6. 输入相似部分
7. 回车执行
8. Tab用法
9. 补全命令
10. 显示相似命令
11. 补全目录或文件

## 附：C语言输入输出

### ****printf****

只能写标准输出中输出的字符串

### ****fprintf****

可以写任意文件中输出的字符串

【注】在Linux中一切皆文件

1. stdin ——标准输入
2. stdout ——标准输出

将数据存入缓冲区，待缓冲区满或遇到换行字符或程序结束时，才向输出设备上输出内容

1. stderr ——标准错误输出（不经过缓冲区直接输出）

例：fprintf(stdout, “Hello”);

### ****返回值return****

1. return EXIT\_SUCCESS;

EXIT\_SUCCESS宏定义在stdlib.h文件中，宏值为0，表示程序运行成功并退出

1. return EXIT\_FAILURE;

EXIT\_FAILURE宏定义在stdlib.h文件中，宏值为1，表示程序运行失败，并退出

注：返回值使用宏定义目的在于提高程序的可读性，在main中使用也可以在调用exit函数中使用。

void exit（int status）结束当前进程，status形参的值用于程序退出的值，相当于main函数中的return值

### **异常处理**

assert(0) //如果形参的值为真时，不做任何处理；否则退出程序，并报错

例：

运行结果：

#include <stdio.h>

#include <stdlib.h>

void fun(void)

{

int n = 0;

assert(n!=0);

fprintf(stdout, "%d\n", 5/n);

}

int main(int argc, char\* argv[])

{

fun();

return EXIT\_SUCCESS;

}

[root@localhost ~]# gcc a.c

/tmp/cc37WP4H.o: In function `fun':

a.c:(.text+0x1b): undefined reference to `assert'

collect2: ld 返回 1

例2：

1 #include <stdio.h>

2 #include <stdlib.h>

3

4 int main(int argc, char\* argv[])

5 {

6 int n = 0;

7

8 while(1)

9 {

10 sleep(1);

11 fprintf(stderr, "\r%02d", n++);

12 }

13

14 return EXIT\_SUCCESS;

15 }

# 五、常用命令

## a. 开关机命令

1. reboot
2. shutdown -r now 重启
3. init 6
4. shutdown -h 0
5. poweroff 关机
6. init 3
7. logout
8. exit 退出登录（终端下）

## b. 文件操作命令

### pwd

功能：显示当前工作目录的绝对路径

### ls

功能：显示文件和目录列表

语法：ls [参数列表] [文件或目录名称列表]

参数： -a 显示所有文件包括隐藏文件（文件名以.开始的文件）

-l 以长格式显示文件完整信息

-h 以适当单位显示文件大小（必须与-l配合）

-R 递归显示目录中的所有文件

-i 显示文件的i节点的值

**节点：每个文件系统会对磁盘上的文件进行编号，这个号码在当前文件系统中是唯一的**

【注】中括号——可选项

尖括号——必选

列表用空格分隔，不特殊说明，次序不限

【附】提示符：$PS1 用于定义shell提示符，显示内容与颜色

例：echo $PS1

\u : 当前用户

\h : 当前主机

\w: 当前工作目录 # 超级用户

\$ : 提示符类型

$ 普通用户

\A：24小时格式显示HH:MM

我的设置：

PS1="\[\e[36m\]\A \[\e[32m\][\u\[\e[31m\]@\[\e[33m\]\h \[\e[32m\]\W]\ \[\e[31m\]\$\[\e[0m\]"

命令过长时用于换行

显示效果：



**ls长格式显示说明：**

第一列：文件权限及类型说明

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 3 4 | 5 6 7 | 8 9 10 |

文件类型 所有者权限 所属者权限 其他人权限

[ - 普通文件

d 目录

l 符号链接（软链接）

p 管道文件

c 字符设备文件——顺序读写

b 块设备文件——随机读写（意义：统一设备访问接口，除特殊设备（网卡））

s 套接字文件

**文件颜色**

**黑色**——普通文件 **蓝**——目录 **黄**——设备  **浅蓝**——符号链接

**红**——压缩文件 **绿**——可执行文件 **红闪**——丢失目标的符号链接

**紫**——套接字 **棕**——管道文件

第二列：硬连接数（一个文件有几个文件名，而它的i节点号相同）

第三、四列：文件大小（空目录大小4096）

**注**：设备文件表示主次设备号，设备号相同，表示使用相同的驱动程序，次设备号，表示不同的子设备

第五列：文件时间 ctime 创建时间

atime 最后访问

mtime 最后修改

**注**：alias ll=’ls -l’ (临时修改命令别名)

### cd

功能：改变工作目录

语法：cd [目标目录]

示例：cd - （回到上一次所在工作目录）

### mkdir

功能：创建目录

语法：mkdir [参数] [目录列表]

参数：-p 递归创建

### touch

功能：修改文件时间或创建一个空文件

语法：touch [文件列表]

注：如果文件存在用来修改或目录的时间（当前系统时间），否则创建一

空的普通文件

### rm

功能：删除文件或目录

语法：rm [参数] [文件或目录列表]

参数： -r 删除目录时加此参数

-i 每删除一个文件或目录时提醒

-f 删除前不提醒

注：rm命令默认情况下，不提示，直接删除，所以在很多操作系统上位rm

命令起别名：alias rm=’rm -i’

### mv

功能：文件目录移动或更目

语法：mv [参数] 源文件或目录 目标文件或目录

参数：-f 覆盖前不提醒（强制删除force）

注：目标文件或目录，存在——移动；不存在——更名

### cp

功能：复制文件或目录

语法：cp [参数] 源文件或目录列表 目标文件或目录

参数： -r 复制目录，默认只复制文件不复制目录

-f 覆盖前不提醒

-a 不改变文件的权限和属性

注：如果文件已存在，覆盖时保持原有权限属性不变

### ln

功能：创建链接文件

语法：ln [参数] 源文件或目录 链接文件名

参数：-s 创建软连接（符号链接）文件

注：默认情况下。ln创建硬连接文件

### cat

功能：查看文件

语法：cat [参数] 文件名

参数： -n 显示行号

-b 空行不显示行号

-s 连续多个空行，只显示一个空行

### more

功能：分页显示文件

用空格翻页，q退出

### less

功能：同more

用<pageup/down>翻页，q退出

### tail

功能：显示文件尾部内容

语法：tail [-n 行号] 文件名

### head

功能：显示文件首部内容，使用同tail

## c. 查找相关

### find

功能：文件查找

语法：find 起始路径 -name 文件名 [-ls]

说明：基于文件名查找，[-ls]用于长格式显示查找结果

注：文件名中可以使用通配符

\*可以通配任意个任意字符

？可以通配一个任意字符

### grep

功能：基于文件内容查找

语法：grep [参数] 查找内容 起始路径或文件名

参数： -n 显示行号

-r 递归查找

-i 忽略大小写

### updatedb与locate

功能：基于数据库进行查找

语法： 创建数据库（整个磁盘文件名数据库） updated

查找 locate 部分或全部文件名

注：在更新数据库后的文件改变无法查找

### which

功能：命令或命令别名查找

语法：which 命令或命令别名

## d. 用户相关

### group

功能：添加组

语法：group 组名 [-g 组ID]

注：系统会为用户和组进行编号，小于500的为系统用户或系统组，默认情

况下新添加的组和用户的ID由500开始

### useradd 或adduser

功能：添加用户

语法：useradd [-g 组名] [-d 家目录] 用户名

参数： -g——省略

1. 先创建与用户名相同的组
2. 创建用户并添加到此组

-d——省略 创建家目录 /home/用户名

### userdel

功能：删除用户

语法：userdel [-d] 用户名

参数：-d 删除用户的同时删除其家目录

### group

功能：删除组

语法：group 组名

注：必须是空组才能删除

### passwd

功能：修改用户密码

语法：passwd [用户名]

注：不加用户名，修改当前用户密码；否则，修改指定用户密码，但仅限超级用户

### su与exit

功能：用于切换用户

语法：切换 su [-] [目标用户]

参数： - ——不写，只切换用户不改变环境设置

省略目标用户——相当于root

示例： su linfeng ——$PATH (root)

root su - linfeng ——$PATH(linfeng)

返回原用户 exit

## e. 文件权限相关

### chgrp

功能：改变文件所属组

语法：chgrp 组名 文件或目录名列表 [参数]

参数：-R 递归修改

### chown

功能：改变文件或目录的所有者和所属组

语法：chown [参数] 所有者.所属组 文件或目录列表

参数：-R

说明： 所有者 —— 只修改所有者

所有者.所属组 —— 两者都修改

.所属组 —— 只修改所属组

示例：chown -R linfeng.embedded a.c

### chmod

功能：改变文件权限

语法：chmod [参数] 权限表达式 文件或目录列表

参数：-R

权限表达式：

1. 八进制
2. 组合表达

|  |  |  |  |
| --- | --- | --- | --- |
| 所有者 | u | +  -  = | r |
| 所属组 | g | w |
| 其他人 | r | x |
| 以上三组 | a |  |

示例： chmod u+x test #给test文件所有者执行权限

chmod ug+rw test

附：

etc/passwd #该文件存储用户的配置信息

etc/group #该文件存储用户组配置信息

## 磁盘管理相关

### fdisk

功能： （1）分区管理

fdisk 磁盘设备文件名

示例：fdisk /dev/sda #对/dev/sda磁盘进行分区

（2）磁盘设备查看

fdisk -l

### mount

功能：磁盘设备挂载

语法：mount [-t 分类类型] [-o 挂载参数列表] 设备文件名 挂载点

分区类型：

|  |  |  |
| --- | --- | --- |
| 类型 | 用途 | 备注 |
| vfat | Win下fat和fat32 | 通常可省略参数 -t |
| ext3 | Linux |
| iso9660 | 光盘 |
| smbfs | Win下网络邻居共享目录，Linux对应samba文件系统 |  |
| nfs | Network file system |  |
| yaffs | 嵌入式设备，可读写文件系统 |  |
| ntfs | Win下ntfs |  |

挂载参数： ro 只读

rw 读写

gid 指定挂载点组ID

uid 指定挂载点用户ID

mode=xxx 指定挂载点权限

iocharset=cp936 （字符设备乱码问题）

**注**：在UNIX族操作系统上，将存储设备与目录进行关联的操作称为**挂载**，被关联的目录称为**挂载点。**

### umount

功能：卸载

语法：umount 挂载点或设备文件名

注：device a busy 的原因 —— 此设备正在使用

### df

功能：文件系统查看

语法：df [参数] [分区设备文件名或挂载点]

参数：-h 以适当单位显示大小

示例：df -h /dev/sda1

### du

功能：查看目录使用情况

语法：du [参数] [目录名]

参数： -s 只显示总计使用情况

-h 同上

示例：du -sh /mnt/usb

## 系统管理

### env

功能：查看系统环境变量

变量格式： 变量=值

常见变量：

|  |  |
| --- | --- |
| HOSTNAME | 当前主机名 |
| SHELL | 当前shell |
| HISTSIZE | 历史命令记录数量 |
| USER | 当前用户名 |
| LS\_COLORS | 指定ls显示不同文件的颜色 |
| PATH | 命令路径 |
| PWD | 当前路径 |
| LANG | 当前语言类型 |
| HOME | 家目录 |
| LOGNAME | 登录用户名 |
| OLDPWD | 上一次所在工作目录 |

### date

功能：查看和修改系统时间

语法： （1）查看 date

（2）修改 date MMDDHHmm[[cc]YY][.SS]

月 日 时 分 年 秒

### hwclock

功能：查看和修改rtc时钟

语法： （1）查看 hwclock

（2）修改 hwclock -s （以rtc时钟同步系统时钟）

hwclock -h （以系统时钟修改rtc时钟）

### ifconfig

功能：IP查看与设置

语法： （1）查看 ifconfig [网卡设备名]

（2）配置 ifconfig 网卡设备名 IP地址 [netmask(子网掩码)]

注：网卡名 eth m[n]

m ——物理网卡ID

n ——虚拟网卡编号（一个网卡最多虚拟四个网卡）

## 包管理命令

### 1.tar

功能：打包或解包

语法： 1）打包 tar zcvfj 包文件名 目录

参数： z gzip压缩

c 创建包（必选）

v 显示打包过程（可选）

f 打包成文件（必选）

j bizip压缩

压缩类型比较：

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 压缩比 | 速度 | 扩展名 |
| bizip | 大 | 慢 | .tar .bz2 |
| gzip | 小 | 快 | .tar .gt或.tgz . |

2）解包 tar zxvf 包文件名 [-C 目标目录]

参数： x 解包

-C 指定解包路径

3）查看包 tar ztvf 包文件名

### 2. rpm

功能：红帽包管理

用法： 1）安装包

rpm -ivhU --nodeps rpm包文件名

参数： -i 安装（install）

-v 显示安装过程

-h 显示安装进度

-U 若包已安装则更新

--nodeps 不检查依赖关系直接安装

2）查看系统已安装了哪些rpm包

rpm -qa

3）卸载rpm包

rpm -e 包名

4）查看已安装文件

rpm -ql 包名

：“|”管道符

cmd1 | cmd2 —— cmd1的标准输出将为cmd2的标准输入

## 附：

创建以下目录结构满足学习需要

/work/

|-- C

|-- C++

|-- database

|-- embedded

| |-- bootloader

| |-- filesystem

| |-- kernel

| |-- qt

| `-- toolchains

|-- process

|-- project

|-- qt

|-- shell

|-- signall

|-- socket

| |-- tcp

| | |-- client

| | `-- server

| `-- udp

|-- software

`-- systemcall

|-- base

|-- jpegdisplay

`-- waveplay

遇到问题根据提示解决进行分析

1. 在哪里产生
2. 为什么产生
3. 解决方案
4. 解决问题（能备份一定要备份，数据无价）

## I. 进程相关

### 1. ps

功能：查看进程

语法：ps [参数]

参数： a 显示所有终端进程

u 显示进程详细信息

x 显示系统进程

示例：ps aux

每列意义： USER 进程拥有者

PID 进程号

**注**：进程号范围1~65535，每个进程加1，循环递增使用

进程ID为1的进程为init进程是系统第一个进程，又称为初始进程，是所有进程的“父”进程，在系统运行过程中其一直驻留内存

%CPU

%MEM 内存占用率

VSZ 虚拟内存大小

RSS 物理内存大小 ？ 不依赖任何终端

TTY 进程所依赖的终端 ttyn 占用终端

ptsn 虚拟终端

**注**：在Linux中所有设备个数标号几乎都从0开始，因此tty2将表示第三个终端

S 休眠

STAT 进程运行状态—— R 运行中

START 启动时间 Z 僵尸进程

TIME 运行时间（实际占用处理器的时间）

COMMAND 启动命令

### top

功能：动态进程查看

语法：top

命令： m 显示内存统计信息

q 退出

### free

功能：显示内存信息

语法：free [参数]

参数： -g

-m

-k

-b

注：基本单位，默认以k为单位

### kill

功能：给指定进程发送指定信号

语法1：查看信号列表

kill -l

|  |  |
| --- | --- |
| SIGUP | 挂起 |
| SIGINT | 终端中断信号 |
| SIGAPRT | 程序异常终止 |
| SIGFPE | 浮点数例外 |
| SIGKILL | 进程终止 |
| SIGSEGV | 段错误（内存非法访问，win下蓝屏） |

语法2：为进程发送信号

kill -信号值 进程ID

注：信号——在Linux操作系统上用于实现进程间的简单通信

### killall

功能：结束进程（基于进程启动命令）

语法：killall 进程启动命令

示例：killall httpd #daemon进程守护

注：内部实际发送的信号为SIGKILL

注：httpd网站服务器程序，IIS，apache（阿帕奇）

# 六、VI编辑器

## a. 进入与退出

### 1. 进入

Vi [文件名称] [+行号]

### 2. 退出

:wq 保存并退出

:q 退出

:q! 不保存退出（强制退出）

:w 保存

:w 文件名 另存为

:w! 强制保存

## b. 工作模式

编辑模式 ：功能等同于记事本

Ii,Aa,Oo,Ss

<ESC>

命令模式 ：对vi下达简单编辑命令（默认）

<ESC>

:或/

末行模式 ：下达文件操作命令或查找、另存为等相对复杂的操作命令

插入 I 。。。。。行首

I 插入到当前光标前

追加 A 。。。。。行尾

a 追加到当前光标后

新行 O 。。。。。。。上。。。。。

o 在当前行下新建一行

替换 S 。。。。。。。行

s 替换当前字符

## c．编辑命令

复制 [n]yy 复制n行，复制一行时可省略n

剪切 [n]dd 剪切n行，剪切一行时可省略n

粘贴 p 将剪切板内容粘贴到当前行下（可视情况除外）

撤销 u 撤销编辑操作

恢复 :redo 恢复已经撤销的操作

## d. 光标移动

<pageUp> <pageDown>

h j k l <home> <end>

W 下一字符首（白空格——空格、制表符、换行）

w 下一单词首（白空格，字符类型不同划分单词）

E 下一字符串尾

e 下一单词尾

gg 文件首

G 文件尾

:行号 调到第几行

## e. 查找与替换

#### 1. 完全匹配查找

1. 将光标移动到要查找的单词上
2. 按“#”
3. 大写“N”下一处

小写“n”上一处

#### 2. 部分匹配查找

1. /字符串
2. 大写“N”下一处

小写“n”上一处

#### 3. 替换

:s/原串/新串[/g] #g表示全局替换

## f．分屏操作

水平分屏 :split 文件名

垂直分屏 :vs 文件名

<Ctrl+w> . W 下一屏

<Ctrl+w> . W 上一屏

## g．可视

主要用于非整行的复制、粘贴、剪切操作

1. 将光标移动到可视的开始或结束处，按“v”
2. 通过光标移动命令将光标移动到可视的结束或开始处
3. 通过“d”剪切、“y”复制
4. 通过“p”可将剪切或复制的可视内容粘贴到光标后

## f．Vi配置

行号 :set nu

:set nonu

语法加亮 :syntax on

:syntax off

去除查找内容加亮 :nohlsearch

设置水平制表符缩进 :set ts=n #n为缩进量

永久配置

1. 修改~/.bashrc #加入vi=’vim’的别名命令
2. 编辑vi配置脚本

~/.vimrc

# 七、Linux主要配置文件

加载引导程序

系统上电

作用：加载操作系统内核，负责向内核传递启动参数

种类：pc grub 嵌入式设备 vivi 小

lilo u-boot

配置文件：

加载内核

/etc/grub.conf(/boot/grub/grub.conf)

启动初始进程

—— /etc/inittab

挂载存储设备

—— /etc/fstab

/etc/sysconfig/network

/etc/sysconfig/network-scripts/ifcfg-xxx

网络配置

/etc/resolv.conf

/etc/hosts

/etc/profile #用户环境（与bash无关）

~/.bashrc #用户环境设置

/etc/bashrc #用户环境（针对bash shell）

用户配置

/etc/passwd #用户设置

/etc/group #组设置

/etc/shadow #密码设置

## /etc/grub.conf

1. 全局变量（顶格）

default=值 #用于指定默认引导的操作系统（由title次序决定其值）

timeoutout=值 #默认引导超时时间

splashimage=(hd0,0)/grub/splash.xpm.gz #背景图

#个性制作——320\*480 索引图——保存xpm格式——gzip压缩

#(hd0,0)表示第一块硬盘的第一分区，相当于/boot/

Hiddenmenu #隐藏操作系统菜单选项

title #菜单项文件

password=xxxxx #MD5密码（128位）

1. 局部变量（一个制表符）

root 用于指定引导分区（内核所在分区，即/boot文件系统）

kernel /vmlinuz-2.6.18-308.el5 ro root=LABEL=/ rhgb quiet vga=791

#用于指定内核引导参数

1.内核所在文件2.以只读方式挂载根分区

3.用于指定根文件系统的位置[标签]（等同于root=/dev/hda3）

4.启动时不打印任何信息

5.用于指定终端显示模式

vga——frame buffer 模式（以内存作显存，可不用显卡）

791——十六进制表示1024\*768\*16bpps（16表示十六位色，屏幕上一点，占两个字节用来放三原色，rgb565）

|  |  |  |
| --- | --- | --- |
| 5r | 6g | 5b |

initrd #系统启动镜像文件

chainloader +1 #工具链

**注：**

**注：**生成MD5密码命令

[root@linfeng ~]# grub-md5-crypt

sudo apt-get install hwinfo  
安装完成后  
sudo hwinfo --framebuffer   
结果就可以看到：  
Mode 0x0300: 640x400 (+640), 8 bits  
Mode 0x0301: 640x480 (+640), 8 bits  
Mode 0x0303: 800x600 (+800), 8 bits  
Mode 0x0305: 1024x768 (+1024), 8 bits  
Mode 0x0307: 1280x1024 (+1280), 8 bits  
Mode 0x0311: 640x480 (+1280), 16 bits  
Mode 0x0312: 640x480 (+2560), 24 bits

<http://forum.ubuntu.org.cn/viewtopic.php?t=236915>

## /etc/inittab

格式： **标号:运行级别:动作:命令**

1. 自定义（2~3个字符）
2. 0~6之间，0项或多项，表示此配置项在哪个初始化级别下生效，不写表示所有级别均生效

initdefault #指定系统默认初始级别

1. 默认 sysinit #指定系统初始化脚本(系统参数、环境变量）

wait #等待该项执行完毕后，再执行其他配置项

## /etc/fstab

格式：

**设备文件名 挂载点 文件系统 挂载参数 开机是否磁盘检测 挂载次序**

前四项参考mount 1——检测

0——不检查

由1~n逐次挂载。0最后挂载，

**注**：如果在文件中出现的存储设备，挂载时仅需指定设备文件或挂载点即可。

例如：fatab中有一行

/dev/cdrom /media iso9660 defualts 0 0

挂载此设备时，仅需执行 mount /dev/cdrom

mount /media

## /etc/sysconfig/network

NETWORKING=yes/no #系统启动时是否开启IPV4网络支持

HOSTNAME=xx.xx.com #当前主机名称（全名）

GATEWAY=xx.xx.xx.xx #网关IP

**注**：网络是否启动与网关地址配置，重启网络后生效

主机名配置重启系统后生效

重启网络方法：

service network restart

Linux下服务管理命令 start

服务名 stop

（见ntsysv） status #状态

## etc/sysconfig/network-scripts/ifcfg-xxx(ethm[:n]/lo(回环))

DEVICE=xxx #设备名等同于（完全相同）

BOOTPROTO=static/dhcp #静态设置或动态获取

BROADCAST=xxxx #广播地址

HWADDR=xxxx #MAC地址一般不能改动（可以没有，但不能错）

IPADDR=xxxx #IP地址

NETMASK=xxxx #子网掩码

NETWORK=xxxx #网络

ONBOOT=yes/no #网卡是否随网络启动（no——禁用网卡）

注：网卡重启生效

重启网卡 ifdown 网卡设备

Ifup 网卡设备

## /etc/resolv.conf

DNS客户端设置——（作用是主机名转为IP地址）

search xxxxx #没用可以删除

nameserver DNS服务器IP #（最多三个）

## /etc/hosts

主机表文件——主要负责本机IP与主机名解析与反解析

（不添加主机表，可能导致图形界面起不来或启动缓慢）

## /etc/profile

用户环境——所有用户均执行此文件，重新登录生效（Bash无关）

## ~/.bashrc

用户环境设置——只针对当前用户有效，重新登录后生效

## /etc/bashrc

用户环境设置——针对所有用户，重新登录后生效（针对Bash shell设置）

## /etc/passwd

用户设置——格式

**用户名：是否需要登录密码：用户ID：组ID：用户说明：家目录：用户shell**

X-需要 空-不需要

**注**：修改后立即生效

## /etc/group

组设置——格式：

**组名：组密码：组ID：组扩展用户列表用“，”分隔**

有无无所谓

## /etc/shadow

密码设置——格式：

**用户名：MD5密码：。。。。。。。。。。**

## 附：

开机后进入单用户系统

1. 按“e”进入编辑模式
2. 修改内核参数，在最后加1（即可进入单用户）

<Ctrl+d> 结束单用户

取消Tab键报警声

进入/etc/inputrc

删除set bell-style none 行前面的注释符#

彻底消除报警声

修改/etc/rc.d/rc.local文件（脚本文件），在最后面加上一句rmmod pcspkr命令（目的是让系统每次在开机的时候删除pcspkr内核模块）

重启

**rmmod**

**功能说明**：删除模块。  
**语　　法**：[rmmod](http://www.linuxso.com/command/rmmod.html) [-as][模块名称...]  
**补充说明**：执行rmmod指令，可删除不需要的模块。Linux操作系统的核心具有模块化的特性，应此在编译核心时，务须把全部的功能都放如核心。你可以将这些功能编译成一个个单独的模块，待有需要时再分别载入它们。  
**参　　数：**  
  -a 　删除所有目前不需要的模块。   
  -s 　把信息输出至syslog常驻服务，而非终端机界面。

与内核模块操作相关的[命令](http://www.linuxso.com/command/)还有

:[ls](http://www.linuxso.com/command/ls.html)mod     [modinfo](http://www.linuxso.com/command/modinfo.html)   [depmod](http://www.linuxso.com/command/depmod.html)    rmmod    inmod    [modprobe](http://www.linuxso.com/command/modprobe.html)

pcspkr内核模块

# 八、shell编程

Shell：用于用户与内核进行交互的一段程序

分类： 1）GUI 用户图形接口

2）CUI 终端用户接口

[ ash —— 嵌入式设备（小）

[ bash —— pc机（全）

[ csh —— 网络设备（网）

## shell脚本的构成

1. 命令
2. 变量
3. 注释（只有单行注释，以#号开始，至行尾结束）
4. 解释器说明

#!解释器路径

#!/bin/sh——符号链接，用于指定当前系统使用的默认shell

注：嵌入式系统中的应用

1. 系统的环境设置
2. 启动或配置应用
3. 设备的加载与参数设置

Shell脚本默认扩展名为.sh

echo #显示字符串、变量

参数：-n 不换行显示

## Shell脚本的执行方法

1. 使用当前shell解释执行

语法： source 脚本文件名（可包含相对或绝对路径）

. 脚本文件名（可包含相对或绝对路径）

1. 使用指定shell解释执行

语法： shell命令 脚本文件名

示例： sh test.sh

1. 直接运行（脚本中指定shell执行）

语法： 路径/脚本文件名

示例： ./test.sh

## 变量的分类和使用

三种形式变量：

1. 系统变量：由操作系统设置（env）
2. 预定义变量：

$? —— 上一条命令的执行结果（任何命令执行成功返回0，执行失败返回非0，而非0值一般为错误号）

$0~$9 —— shell脚本执行时的命令行参数

例：./test.sh [参数列表(空格分隔)] #实际使用$1~$9,$0代表自身

1. 自定义变量：
2. 变量无类型
3. 无需声明

语法：

1. 赋值 变量名大写，单词下划线连接

规定： 等号两端不能有空格

值中包含空格要使用引号括起来

1. 引用：$A
2. 释放：unset A B C （变量列表空格分隔）

注： ‘单引号’不支持变量，”双引号”支持变量

`反引号`用于括起来一条shell命令，当执行到包含反引号的语句时，当先执行反引号中的命令

## 测试语句

1. 文件测试

语法：test 测试符 文件名/目录

[ 测试符 文件名/目录 ]

测试符： -r 读

-w 写

-x 执行

-d 目录

-f 文件

-L 符号链接

-e 存在

1. 数值测试

语法：test 数值1 测试符 数值2

[ 数值1 测试符 数值2 ]

测试符： e 等于

n 不等于

l 小于

g 大于

t、q 无意义

示例： -lt -le -nl -eq -ge

1. 字符串测试

语法1：test 串1 测试符 串2

[ 串1 测试符 串2 ]

测试符： == !=

注：测试中一旦出现变量要用双引号将其括起来，否则可能出错

语法2：test 测试符 串

[ 测试符 串 ]

测试符： -n 是否非空

-z

1. 逻辑测试

-a and 与

-o or 或

! 非

1. 算数运算

语法：expr 操作数1 运算符 操作数2

运算符：+ - \* /

(运算符两边加空格，乘法需要使用转义\\*)

1. 特殊符号

? \* 通配符

` ` 反引号

\ 转义符

|  |  |  |  |
| --- | --- | --- | --- |
|  | 模式 | 符号 | 说明 |
| 输入重定向 | 新建 | < | 将符号后内容定向到符号前命令的stdin |
| 追加 | << |  |
| 输出重定向 | 新建 | > | 将符号前命令的stdout定向到符号后的文件 |
| 追加 | >> | （不清空原有内容） |

语法： 输入重定向 cmd < 串

输出重定向 cmd > 文件

**注**：标准错误输出，是将stderr定向到stdout然后输出(2>&1)

stdin 0

stdout 1

stderr 2

例: ls adsadas >>test 2>&1

ls adsadsa >>/dev/null 2>&1

从命令行读取数据

语法：read 变量名

## 流程控制

1. 分支（if）

if 测试1

then

命令1

elif 测试2

then 可重复，0~任意次

命令2

else 可选

命令3

fi

1. 分支（case）

case $变量 in 数值) 5)

模式1) #模式用于设置匹配方式 字符串) stop)

命令 正则表达式) 9[0-9])

;;

模式2)

命令

;;

模式n)

命令

;;

esac

1. 循环（while）

while 测试 #真值时循环

do

命令

done

until 测试 #假值时循环

do

命令

done

1. 循环（for）

for 变量名 in 值列表（空格分隔）

do

命令

done

## 函数

用法：

定义： function 函数名

{

#函数体

}

调用： 函数名 参数列表

**注**：参数列表与命令行参数用法一样，$0~$9

函数在shell当中相当于一条命令

## 附：

1. 命名法：

匈牙利法：字母全小写，单词以下划线分隔

骆驼法：函数——每个单词首字母大写

变量——第二个单词开始首字母大写

1. exit 1 #退出当前shell
2. 目录的可执行权限

表示是否可以进入这个目录，软连接文件的权限都是777

# 九、C开发工具

## 1. Gcc —— GNU（GUN is Not Unix）

可执行文件

目标文件

.o

汇编

.s

源文件

预处理 编译 组译 链接

静态库

共享库

预处理

1. #include 头文件包含
2. #define PI 3.14 宏替换
3. #ifdef 条件编译

#if 0

等预处理

**静态库**：编译时加载，运行时无关

**共享库**：编译时检测，运行时加载

## 2.语法：

gcc [参数] [源文件列表]

## 3.参数：

### 过程控制

Gcc 默认编译到链接阶段，生成可执行文件

-E 仅编译到预处理阶段，将预处理后的代码作为stdout

-S 进行到编译阶段，生成同名的.s汇编文件

-c 进行到组译阶段，生成同名的.o目标文件

-o 文件名 用于指定编译器输出文件时的文件名

注：编译器自带宏 \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_

表示 当前文件 所在函数 当前行号

### 预处理阶段

-I路径 用于为当前预编译添加默认头文件（需要用到的头文件不在默认目录时）

例：gcc main.c -o main -I. #包含头文件在当前目录

-include 头文件 用于指定当前预处理包含的头文件（需要调用的头文件不在该文件中时）

例：gcc main.c -include def.h #文件中需要引用def.h头文件

-D宏名 用于向当前预处理添加宏定义

例：gcc main.c -DSUMMER #给文件一个宏参数

**注**：

1. 宏定义时，

宏体中出现运算符，必须将宏体用括号括起来；如果是带参

宏，宏体中的宏参，必须用括号括起来。

如：#define DIV(a, b) ((a)+(b))

1. 头文件

< > 在默认头文件目录查找

“ ” 现在当前目录下查找，再到默认头文件目录下查找

1. 条件编译

#ifndef DEF\_H //如果没有定义这个宏（该头文件之前的代码中）

#define DEF\_H //定义一个DEF\_H宏

#define VALUE 8

#endif //DEF\_H 判断对应哪个if，防止缺省

防止头文件重复，宏名起发为当前文件名称大写

### 链接阶段

-l库名 指定链接库

例：./sin -lm #math.h头文件需要调用链接库 (见man sin)

注：C库名称规范

libxxx.so.版本号 —— 共享库

libxxx.a.版本号 —— 静态库

lib.xxx.la.版本号

前缀 库名 类型

-L路径 添加链接库默认路径

-static 使用静态链接库

-s 去除冗余 —— 如标识符（见nm）

### 编译阶段

-On n为0~4的整数，用于指定编译器对代码的优化级别，数值越大优化级别越高（-O0 为默认）

**注**：优化是编译器修改变量的存储位置与流程控制

**建议**：优化级别为2或3

-Wall 显示所有警告信息

建议编译参数：-Wall -O3 -o 文件名

## 4.制作共享库

语法： gcc -fpic -shared 源文件列表 -o 库文件名

地址可重定位 共享库

示例： gcc -Wall -O3 -fpic -shared \*.c -o libxxx.so

使用： gcc -Wall -O3 -s main.c -o main -L. –I.

共享库运行时不能加载解决方法：

方法1：添加到共享库默认路径为/lib/（针对于自己制作的共享库不提倡）

方法2：通过环境变量指定共享库位置

LD\_LIBARAY\_PATH变量末尾加上自己的“：库路径”

方法3：

1. 修改Linux配置文件/etc/ld.so.conf，在此文件中新起一行写上库路径
2. 执行命令ldconfig，更新系统共享库的hash表

## 5.制作静态库

1. 将要制作的静态库的源文件编译成目标文件
2. 归档（archive）

ar -r 静态库名 目标文件列表

1. 编译 略（参照共享库）
2. 运行 略

## 6.相关文件命令

1. ldd ——查看可执行文件运行时所需要的共享库

用法：ldd 可执行文件名

1. file ——查看文件类型、架构等信息

用法：file 文件名

1. stat ——查看文件属性、权限、时间等信息

用法：stat 文件名

1. nm ——查看可执行文件中的标识符（包括库文件）

用法：nm 可执行文件名 #strip的文件无效（gcc参数-s）

1. strip ——去除可执行文件中的冗余信息（包括库文件）

用法：strip 可执行文件

1. cproto ——由.c源文件生成.h头文件

用法：cproto 源文件名 #重定向到所需的.h文件中

## 7.编译错误

1. 错误提示中由行号、文件、函数名是编译时出错，语法错误
2. 不显示行号等信息，链接错误

## 8.ftp使用

1. ftp 服务器IP或主机名 #进入ftp登录会话
2. 输入用户名、密码

匿名用户，用户名：ftp 密码：无（直接回车）

成功后进入ftp shell

操作命令：

ls 查看服务器文件列表

cd 切换在服务器中的目录

get 服务器上的文件名 下载一个文件

put 本地文件名 上传一个文件

mget 服务器上的文件名（可带通配符） 下载多个文件

mput 本地文件名（可带通配符） 上传多个文件

by 退出ftp shell

1. 保存位置为当前打开ftp的目录

## 9.源码文件的安装过程

1. 解包到指定目录（/usr/local/src/）
2. 进入源码目录
3. 对源码软件进行配置

配置方法：（执行配置脚本）

./configure [--help] #不加参数默认安装

**注**：为什么要配置？

1. 检测系统架构
2. 检查库是否满足条件
3. 检测编译器版本
4. 生成编译规则文件
5. 编译 make
6. 安装 make install

**注**：默认情况下，手动源码安装的库文件，在/usr/local/lib/目录下

## 附：规范

1. 多个源文件和头文件构成的项目，要为其创建一个目录，项目目录。该项目名应对应项目的可执行文件名
2. #include < >（中间加空格）
3. 运算符两侧必须加空格，一元运算符除外。
4. 逗号后加空格。
5. 声明语句和可执行语句之间加空格
6. return 之前加空格
7. 一条语句过长时，要分多行书写
8. 用水平制表符进行缩进，缩进要按层次，制表符宽度为4（set ts=4）
9. 文件末尾加空行

# 十、C语言

## 1.数据类型

### 基本数据类型

主要针对内存，为了节省内存

|  |  |  |  |
| --- | --- | --- | --- |
| 整型 | char | 1 | Unsigned值域0~255 |
| Signed值域-128~127 |
| short | 2 | Unsigned值域0~65535 |
| Signed值域-32768~32767 |
| int | 2 | 字节数-16位系统以下 |
| 4 | 字节数-32位系统以上 |
| long | 4 | 以1开头的10位整数 |
| long long | 8 | （4+4 = 8） |
| short long | 3 | （（2+4）\ 2 =3） |
| 实型 | float | 4 | （精度）小数点后6位 |
| double | 8 | （精度）小数点后16位 |

值域范围： n = 字节数 \* 8

有符号 0 ~ 2^n – 1

无符号 -2^(n-1) ~ 2^(n-1) – 1

常量表示法：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 5L | 5.0 | | 5.0f | | .5 | 075 | | 0x123f | 0UL |
| Int | Long | Double | | float | | double | 八进制int | | 十六进制int | 0 unsigned long |
| 2E3 | -2E-3 | 0.8E-5 | | .8E-5 | | 8E0.5 |  | |  |  |
| 2\*10^3 | -2\*10^-3 | 都double | | | | | | | | |
| ‘a’ | ‘\n’ | ‘\m’ | ‘\0’ | | ‘\123’ | | | ‘\x1234’ | | |
| 字符a | 换行 | m | 0 | | 字符的八进制表示法（转义字符开始不多于3位的八进制数） | | | 字符的十六进制表示法（转义字符+x开始后不多于4位的十六进制数） | | |

### 衍生数据类型

* + 1. 数组

声明：类型 数组名 [行长度][列长度]={{…}, {…}, ….{…}}

在内存中按行存储

* + 1. 指针

声明：类型 \*变量名

存储内存地址的变量，为了存储所有地址，所占字节数与该系统的地址总线有关。一般与int类型的字节数相同。

指针的运算：

指针与整型 p + n p – n

如：p + 1 —— 指向下一个元素（元素表示给指针的类型单元）

Int \*P = (int \*)0x1000 0000;

P + 2;

所以此时P为0x1000 0008

指针与指针 p – q

指针与指针进行减法（两指针类型必须相同），只有两指针指向同一数组时，才有意义，其运算结果为两指针间有多少个元素。

* + 1. 枚举

enum 枚举类型 {枚举列表}

例：enum WEEK{Monday = 5， Tuesday， 。。。。}

### 构造数据类型

* + - 1. 结构体

先定义类型

通过定义的类型声明变量

结构体取成员的运算符有两个，分别是 . 和 ->

. 是通过结构体变量取成员 如：s.a;

->是通过结构体地址取成员 如：ps->a;

注：内存对齐

1. 4字节对齐
2. 如果结构体中的所有成员的数据宽度均未达到4字节，按最大成员数据宽度进行对齐。

示例：

struct std1{

int a; //4

char b; //1

//占空3

float c; //4

char d; //1

//占空3

double e; //8

}

sizeof(std1) = 24

struct std2{

char c; //1

//占空1

short s; //2

};

sizeof(std2) = 4

* + - 1. 共同体

所有成员使用同一块内存

union st{

char a[2];

short b;

};

int main(int argc, char \*argv[])

{

union st s;

s.b = 0x1122;

fprintf(stdout, “%x, %x\n”, s.a[0], s.a[1]);

return EXIT\_SUCCESS;

}

结果：22,11

判断当前系统是大端系统还是小端系统：

在上例基础上修改

s.b = 0x0001;

fprintf(stdout, "这是%s端系统\n", s.a[1] ? “大” ：”小”);

* + - 1. 位段结构

struct st{

unsigned int a : 5; //5表示位数

unsigned int b : 8;

unsigned int c : 9;

unsigned int d : 1;

};

sizeof(struct st) = 4

对齐方式和结构体相同，位数不能超过前边类型的宽度

## 2.运算符

C语言中有42个运算符

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 赋值 | = | += | -= | /= | %= | \*= |  | 6 |
| 算术 | + | - | \* | / | % | ++ | -- | 7 |
| 关系 | > | < | <= | => | == | != |  | 6 |
| 逻辑 | && | || | ! |  |  |  |  | 3 |
|  | \*(取值) | &(取址) |  |  |  |  |  | 2 |
| 位运算 | & | | | ^ | ~ | << | >> |  | 5 |
|  | &= | |= | ^= |  | <<= | =>> |  | 5 |
| 三元 | ? : |  |  |  |  |  |  |  |
| 逗号 | ， |  | sizeof |  |  |  |  |  |

**注**：a << n 相当于 a\*2^n

~1 为 -2 ~2 为 -3 ~3 为 -4

1 原码 0000 0001 取反后 1111 1110

-2原码 1000 0010 反码 1111 1101 补码 1111 1110

在计算机中负数都以补码形式存在，所以~1为-2

## 附：rgb565

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Rgb565

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 0 | 0 | 0 |

R （r & 0xF8）<< 8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 0 | 0 |

G (g & 0xFC) << 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 0 | 0 | 0 |

B (b >> 3)

Rgb565 = ( (r & 0xF8) << 8) | ((g & 0xFC) << 3) | ( b >> 3);

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

int main(int argc, char \*argv[])

{

uint8\_t r = 0xFF;

uint8\_t g = 0xCC;

uint8\_t b = 0x99;

uint16\_t rgb = 0;

rgb = ((r & 0xF8) << 8) |

((g & 0xFC) << 3) |

(b >> 3);

fprintf(stdout, "%4X\n", rgb);

return EXIT\_SUCCESS;

}

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

typedef union{

uint16\_t rgb;

struct {

uint16\_t b : 5;

uint16\_t g : 6;

uint16\_t r : 5;

}colors;

}Color;

int main(int argc, char \*argv[])

{

uint8\_t r = 0xFF;

uint8\_t g = 0xCC;

uint8\_t b = 0x99;

Color color;

color.colors.r = r >> 3;

color.colors.g = g >> 2;

color.colors.b = b >> 3;

fprintf(stdout, "%04X\n", color.rgb);

return EXIT\_SUCCESS;

}

# 十一、MAKE工具

## 1.编译规则文件

GUNMakefile 高

规则文件 makefile 优先级

Makefile（一般） 低

一般编写使用Makefile命名，其他两个多用于在一个项目中存在检测

## 2.规则文件由规则构成

目标 ：[依赖列表]

tab 命令

。 多行或

。 0行

。

语法：

规则：

1. 先递归、 后迭代

（检测依赖关系）（根据目标与依赖的时间关系，决定是否重新执行规则里的命令）

1. 在规则文件中，第一个目标为默认目标，如果在下达make命令时，不指定目标，那就使用默认目标，为递归结点。

也可以在make时，指定一个目标

1. 没有依赖的目标为伪目标，最好在此目标之上先做说明。

如：PHONY：clean

clean：

rm –f circle.o area.o main.o

## 3.变量

1. 自定义变量

赋值：变量名=值列表（空格分隔） #多行加转义符

引用：$(变量名)

注：

= 引用时才展开变量值

:= 赋值时直接展开变量（替换后不变）

+= 连接形成列表

?= 如果原来变量没有赋值时，才有效

1. 预定义变量

|  |  |  |
| --- | --- | --- |
| $@ | 本条规则的目标 |  |
| $^ | 本条规则所有依赖 |  |
| $< | 本条规则第一个依赖 |  |
| CC | 默认的C编译器 |  |
| CFLANGS | 指定编译阶段的参数 |  |

1. Make的推断

推断简单的编译命令

多目标生成时，可将多个目标作为依赖关系，重新设定一个目标

## 4.自动生成Makefile

a． qmake -project #使用QT自带工具生成一个round.pro项目文件

b. qmake #通过xxx.pro项目文件生成Makefile编译规则文件

c. vi Makefile文件，对其根据需要修改 编译参数

d. make #编译 指定编译器

头文件指定

库名指定