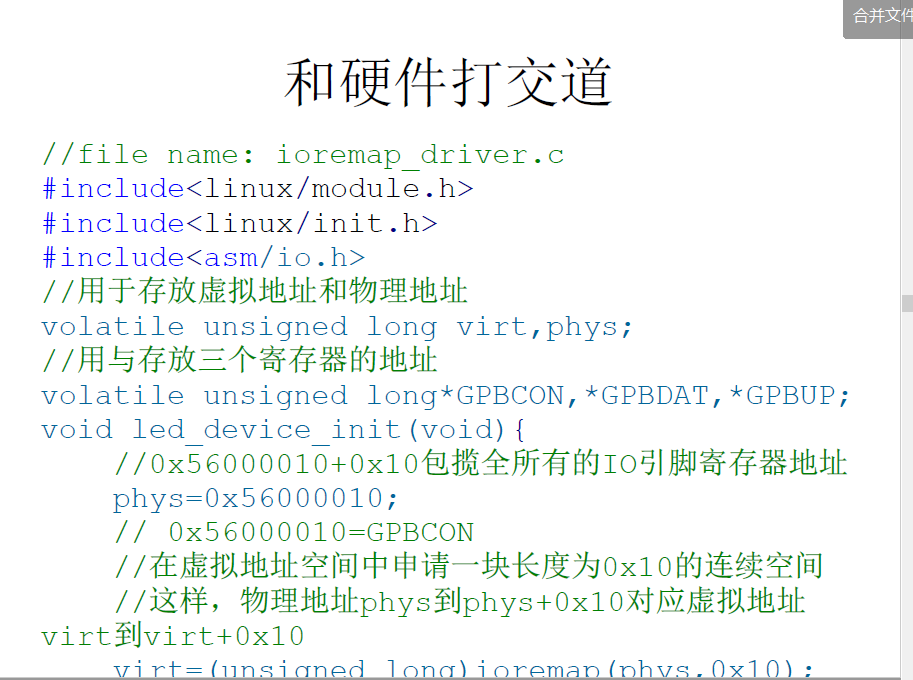
# Linux重点整理

阿汤哥称重点占80-85左右，重点和出题均凭印象，可能会出超范围题目

2021更新：

阿汤哥非常仗义，说85分在重点内就是85分，一道15分的大题，被他无情地扔出了重点外。

“这个有什么好考的，划掉划掉”



目录

[Linux重点整理 1](#_Toc48169654)

[Linux Basics 3](#_Toc48169655)

[什么是Linux 3](#_Toc48169656)

[Linux的历史 3](#_Toc48169657)

[Linux的作者是谁？ 3](#_Toc48169658)

[Linux的发行版有哪些？ 3](#_Toc48169659)

[Linux是由什么构成的 3](#_Toc48169660)

[Linux的协议 3](#_Toc48169661)

[MBR和GPT是什么？（不考） 3](#_Toc48169662)

[常见的文件系统概念 3](#_Toc48169663)

[Linux启动流程（占分不大） 4](#_Toc48169664)

[内核和GRUB关系 4](#_Toc48169665)

[多用户和多任务 4](#_Toc48169666)

[VT（选择） 4](#_Toc48169667)

[分区（简单了解） 4](#_Toc48169668)

[基本命令 4](#_Toc48169669)

[ls-l的各字段意义 4](#_Toc48169670)

[进程、线程的概念 5](#_Toc48169671)

[用于寻求帮助的命令（不考） 5](#_Toc48169672)

[UNIX Overview 5](#_Toc48169673)

[层次图 5](#_Toc48169674)

[Linux Basics(2) 6](#_Toc48169675)

[基本命令 6](#_Toc48169676)

[重定向(重要) 6](#_Toc48169677)

[管道(重要) 6](#_Toc48169678)

[基本的环境变量（Home、Path、PS1……） 7](#_Toc48169679)

[Find和grep 7](#_Toc48169680)

[Shell Programming（没讲，不考编程题，靠概念） 7](#_Toc48169681)

[什么是shell 7](#_Toc48169682)

[Shell不止bash一种，有哪些其他的shell？（不考） 7](#_Toc48169683)

[什么功能是Shell进程本身提供的？哪些是Shell下的别的程序提供的？ 8](#_Toc48169684)

[执行脚本文件的方式和区别（脚本不要求写，要求读懂） 8](#_Toc48169685)

[read的基本功能，引号的基本用法（单引号，双引号的区别）（读懂） 8](#_Toc48169686)

[参数变量和内部变量 8](#_Toc48169687)

[条件分支（While，Until掌握一个即可） 8](#_Toc48169688)

[命令组合 8](#_Toc48169689)

[$(())用作计算 9](#_Toc48169690)

[即时文档 9](#_Toc48169691)

[Programming Prerequisite 9](#_Toc48169692)

[源代码是怎样编译成可执行码的？ 9](#_Toc48169693)

[编译的流程，动态库和静态库的区别 10](#_Toc48169694)

[gcc，g++的常见参数（ppt上的11个） 10](#_Toc48169695)

[.o .a .so 11](#_Toc48169696)

[make&makefile：只要求能读得懂，不要求写 11](#_Toc48169697)

[makefile的流程要知道（怎么实现增量编译的？执行次序是怎样的？只考课上讲过的） 11](#_Toc48169698)

[System Programming 12](#_Toc48169699)

[文件，文件系统的概念、七种文件类型 12](#_Toc48169700)

[VFS的作用 13](#_Toc48169701)

[VFS的四种结构，分别表示什么意思？ 13](#_Toc48169702)

[硬链接和软链接的特点 13](#_Toc48169703)

[系统调用和库函数有什么差别？ 14](#_Toc48169704)

[库函数的缓存机制要求掌握例子 14](#_Toc48169705)

[系统调用要求写程序，不会单独地问（打开一个文件可读/可写/可读可写） 14](#_Toc48169706)

[文件的权限 15](#_Toc48169707)

[关闭文件 15](#_Toc48169708)

[fcntl设置标志位不考 ioctl指向驱动 16](#_Toc48169709)

[哪些调用是库函数/系统调用？不要搞混 16](#_Toc48169710)

[缓存：理解库函数的缓存机制带来的影响 16](#_Toc48169711)

[文件锁 16](#_Toc48169712)

[Kernel Driver 16](#_Toc48169713)

[内核的定义 16](#_Toc48169714)

[内核编译步骤 16](#_Toc48169715)

[内核有哪些基本功能 16](#_Toc48169716)

[驱动 17](#_Toc48169717)

[加载和释放模块的命令 17](#_Toc48169718)

[理解模块之间的依赖关系 17](#_Toc48169719)

[Linux内核模块与应用程序区别（重点）（用户态程序和内核态程序的区别） 17](#_Toc48169720)

[开发驱动程序有哪些注意点 17](#_Toc48169721)

[设备驱动（不考） 17](#_Toc48169722)

[设备驱动类型 18](#_Toc48169723)

[内核模块操作/proc文件（选择判断） 18](#_Toc48169724)

[导出符号表（仅看画圈，不看实现） 18](#_Toc48169725)

[模块参数传递 18](#_Toc48169726)

[编译内核模块 19](#_Toc48169727)

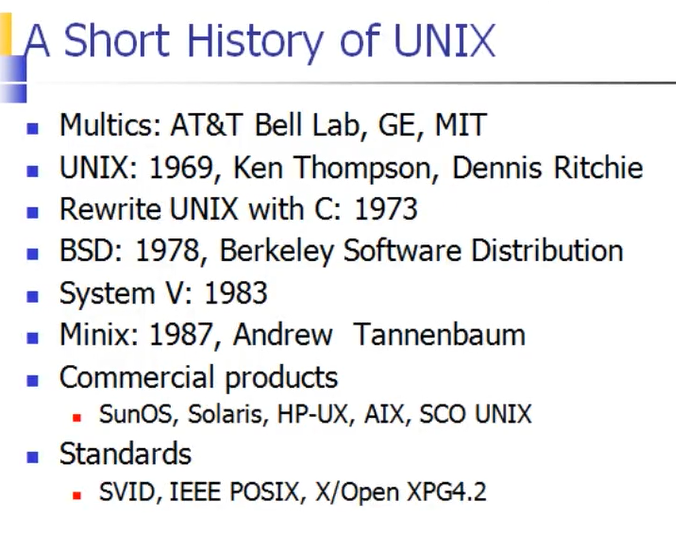
[内核模块操作/proc文件（选择判断） 19](#_Toc48169728)

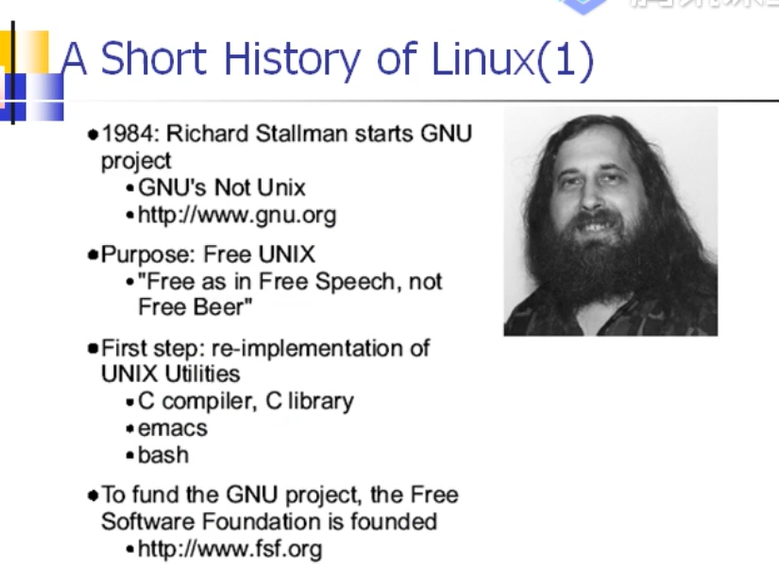
## Linux Basics

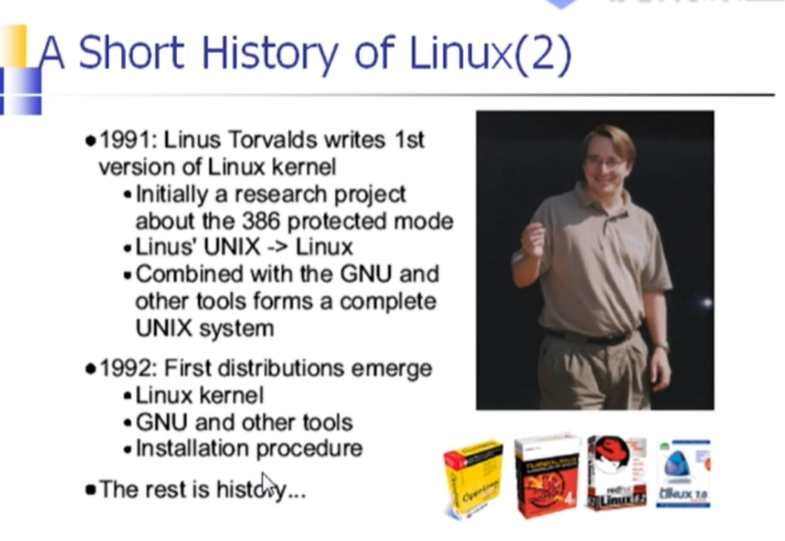
###### 什么是Linux

全称GNU/Linux，是一套GNU通用公共许可证下免费使用和自由传播的类UNIX操作系统，开源，流行，支持绝大多数可用平台

###### Linux的历史（大概看看）







###### Linux的作者是谁？

Linus

###### Linux的发行版有哪些？

Redhat -> Fedora

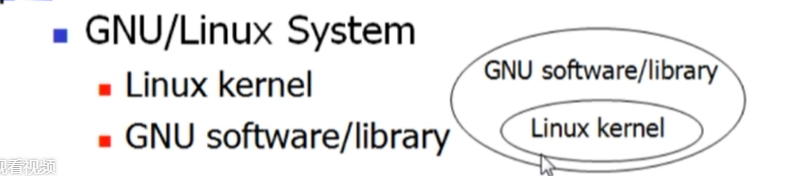
Debian

SuSe

Mandrake

Ubuntu

###### Linux是由什么构成的



###### Linux的协议

GPL协议，

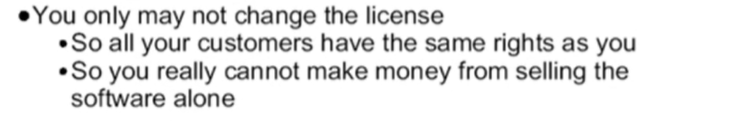
GPL协议最主要的几个原则：

1、确保软件自始至终都以开放源代码形式发布，保护开发成果不被窃取用作商业发售。任何一套软件，只要其中使用了受 GPL 协议保护的第三方软件的源程序，并向非开发人员发布时，软件本身也就自动成为受 GPL 保护并且约束的实体。也就是说，此时它必须开放源代码。

2、GPL 大致就是一个左侧版权（Copyleft，或译为“反版权”、“版权属左”、“版权所无”、“版责”等）的体现。你可以去掉所有原作的版权 信息，只要你保持开源，并且随源代码、二进制版附上 GPL 的许可证就行，让后人可以很明确地得知此软件的授权信息。GPL 精髓就是，只要使软件在完整开源的情况下，尽可能使使用者得到自由发挥的空间，使软件得到更快更好的发展。

3、无论软件以何种形式发布，都必须同时附上源代码。例如在 Web 上提供下载，就必须在二进制版本（如果有的话）下载的同一个页面，清楚地提供源代码下载的链接。如果以光盘形式发布，就必须同时附上源文件的光盘。

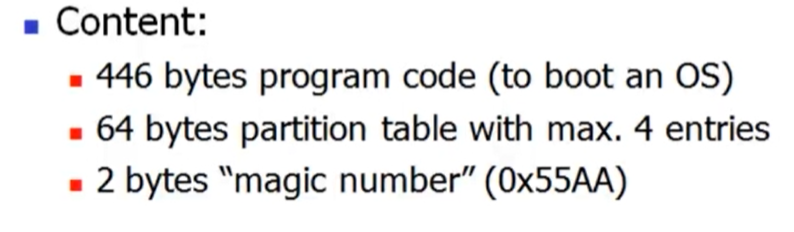
4、开发或维护遵循 GPL 协议开发的软件的公司或个人，可以对使用者收取一定的服务费用。但还是一句老话——必须无偿提供软件的完整源代码，不得将源代码与服务做捆绑或任何变相捆绑销售。



###### MBR和GPT是什么？（不考）

MBR：Master Boot Record，主引导记录。

硬盘的0磁道的第1个扇区称为MBR，共512字节，由BIOS定位



GPT：GUID Partition Table Scheme，GUID磁盘分区表，及“全局唯一标识磁盘分区表”，是一个实体硬盘的分区表的结构布局的标准。

###### 常见的文件系统概念

文件系统定义：操作系统中负责存取和管理文件的部分

常见Linux文件系统：VFS、EXT2、EXT3、FAT32

###### Linux启动流程（占分不大）

BIOS

检查内存和硬件，从非易失性内存中加载选项，检查启动设备，加载启动设备的MBR并执行它

MBR

包含“引导加载程序”和分区表

传统上由LILO（Linux Loader） / GRUB（Grand Unified Boot Loader）设置，一般在/dev/hda

引导程序

 将压缩的内核映像加载到内存

 内核解压缩并启动...

初始化过程

 配置文件/ etc / inittab

 运行级别

###### 内核和GRUB关系

GRUB保存在MBR或/boot/grub 1.5-2扇区

理解文件系统结构，不需要和LILO一样启动配置

配置文件在/boot/grub/grub.conf ，告知了标题，root，内核和init文件

用grub-install在MBR安装

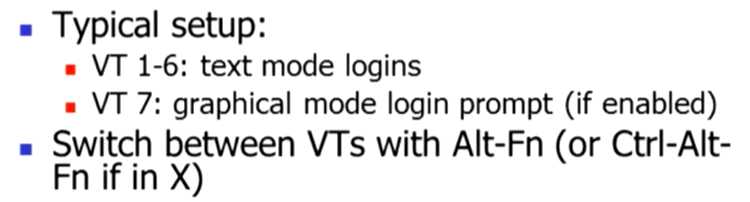
###### 多用户和多任务

Linux是多用户多任务的操作系统，多用户可以模拟运行多任务，相互独立

使用系统前需要登陆，通过用户名密码来确认身份

登录有多种方式：控制台、串行终端，网络连接

###### VT（选择）



###### 分区（简单了解）

/boot：开机文件

/bin：普通命令，如文件操作

/lib：函数库

/dev：设备

/etc：配置文件等

/usr：自动安装的软件目录

/var：软件产生的数据存放目录，如日志、数据库文件、缓存文件

/root：管理员的home目录

/sys：系统运行中的硬件驱动

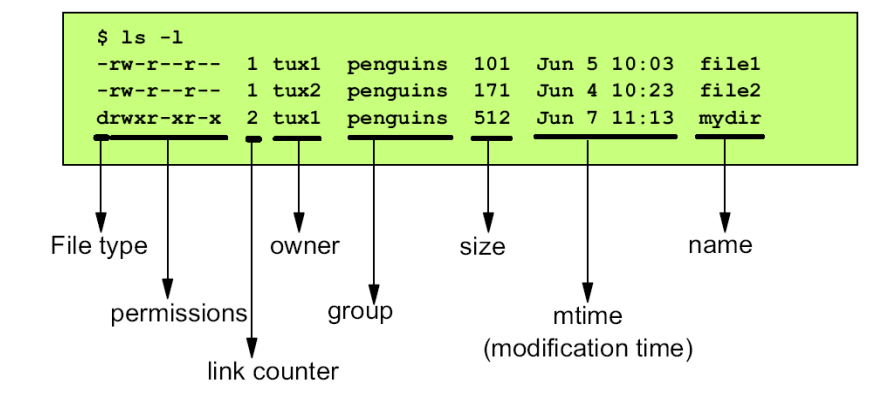
/proc：正在运行的内核信息

/sbin：sbin下大多是跟系统管理有关的可执行程序，一般需要root权限

/opt：主机额外安装软件所摆放的目录。默认是空的。 一般安装软件的时候，可以自己指定安装到这个目录下，便于查找和管理

/srv：主要用来存储本机或本服务器提供的服务或数据

###### ls-l的各字段意义



###### 进程、线程的概念

进程是任务。

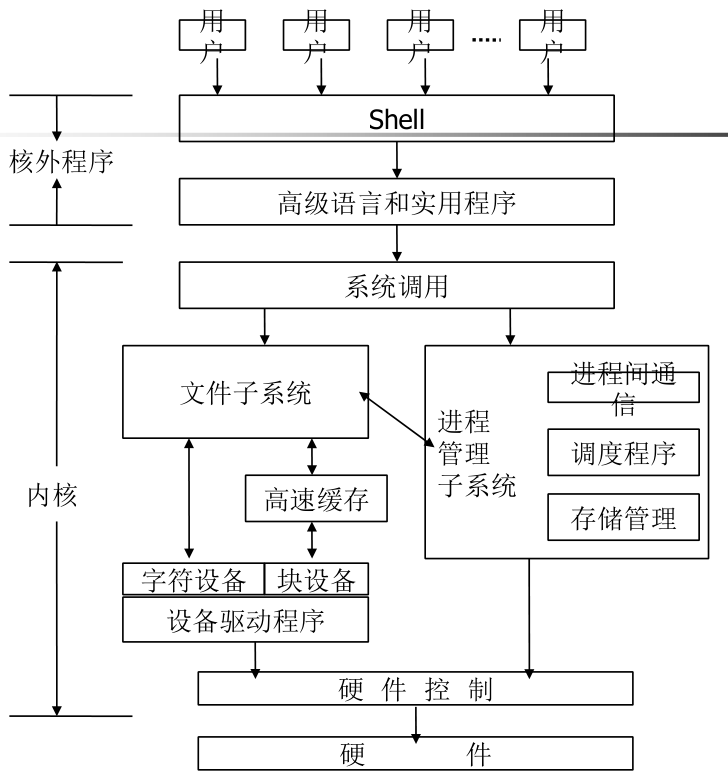
进程是一个正在执行的程序实例。由执行程序、它的当前值、状态信息以及通过操作系统管理此进程执行情况的资源组成。

###### 用于寻求帮助的命令（不考）

man info

###### UNIX Overview

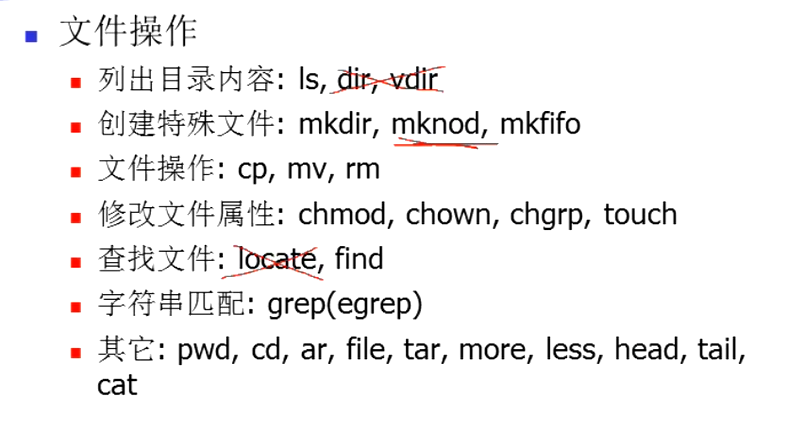
###### 层次图

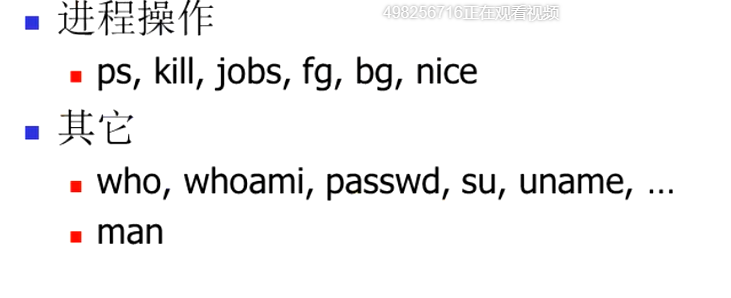


系统调用位置、用户态和内核态程序

## Linux Basics(2)

###### 基本命令





Ls：看文件信息

passwd：改密码

mkpasswd

date：显示或设置系统时间(主要是通过百分号加一系列字母显示不同格式的时间)

cal：显示日历

who、finger：显示目前登陆系统的用户信息

write：传讯息给其他使用者，wall是向系统当前所有打开的终端上输入信息

talk：允许多个终端通信

mesg：设置y时，其他用户可利用write指令将信息直接显示在您的屏幕上（n不可以）

clear：清屏

echo（-n不换行；-e输出转移字符的意义；echo `command`输出command执行后的结果）

pwd：显示工作目录

mkdir：创建目录（mkdir -p AAA/test，如果不加-p且AAA目录不存在则产生错误）

rmdir：删除空目录（-p 是当子目录被删除后使它也成为空目录的话，则顺便一并删除）

touch：修改文件/目录的时间属性（touch linuxidcA -r sampleFileB，A改为B的时间，-c用于如果输入的文件名不存在则不创建文件，一般是和其他参数联合使用，-d/-t分别是不同的日期设置格式）cp：复制（-i/f：是否提示覆盖已存在的目标文件；-r：递归复制目录）

mv：移动

ps：列出所有进程，常用参数是列出所有及显示命令

-A/-e：显示所有进程 -f：显示进程父子关系（PID-PPID） -x：显示所有进程（一般与au连用，-A不能与-u连 用） -u：显示进程拥有者、状态、资源占用等详细信息 -a：显示同一终端下的所有进程

pstree：树状结构列出进程

job、fg、bg、ctrl+Z：任务前后台执行（jobs查看当前控制台下的任务） ctrl+z将任务挂起，挂起的任务可以 通过fg/bg恢复执行 jobs命令中输出有加号和减号，带加号的作业被当做默认作业，带减号的为下一个默认作 业。 bg 将进程搬到后台运行（Background） fg 将进程搬到前台运行（Foreground）

kill+信号+进程号，常用的九种信号

nice、renice：改进程优先级

top：监控 Linux 系统状况，比如cpu、内存的使用

dir、vdir不要求掌握（windows下命令）

mknod创建设备文件、mkfifo创建管道

chown：更改拥有者；chgrp：更改用户组

locate（去/var/lib/slocate资料库找，更快）、find：搜文件，建议掌握find

ar：写c库打包 head -c/-n：显示文件开头的前n个字节/n行(tail显示结尾)

file显示文件类型

whoami：显示自身用户名称

uname：显示系统信息

###### 重定向(重要)

012文件描述符

标准输入（0）：<

标准输出（1）：> 或1> 覆盖原文件，>> 追加源文件

标准错误（2）：2> 标准错误覆盖原文件

2>&1：把标准错误输出重定向到标准输出，加&符号是因为后面是文件描述符而不是文件名 &>file：把标准输出和标准错误输出都重定向到文件file中

###### 管道(重要)

一个进程的输出作为另一个进程的输入（前指令 | 后指令）

###### 基本的环境变量（Home、Path、PS1……）

$HOME：当前用户登陆目录

$PATH：以冒号分隔的用来搜索命令的目录清单

$PS1：命令行提示符，通常是”$”字符

$PS2：辅助提示符，用来提示后续输入，通常是”>”字符

$IFS：输入区分隔符。当shell读取输入数据时会把一组字符看成是单词之间的分隔符，通常是空格、制表符、换行符等。

###### Find和grep

find：指定目录下查找文件

常见用法：

find . –name “\*.c”：当前目录所有拓展名是c的文件

find . –type f：当前目录所有普通文件

find . –ctime -20：20天内更新过的文件

grep：查找内容

常见用法：

-c：统计符合样式的列数

-i：忽视大小写

-n：显示行号

-v：反转显示

-A n：显示符合样式内容以及以后n行内容（after）

-B n：显示符合样式内容以及以前n行内容（before）

-C n：显示符合样式内容以及前后n行内容（context）

-r：递归查找

举例：

grep “test” /etc/passwd：在/etc/passwd中查找所有文件名带有test的文件

## Shell Programming（没讲，不考编程题，考概念）

###### 什么是shell

###### Shell不止bash一种，有哪些其他的shell？（不考）

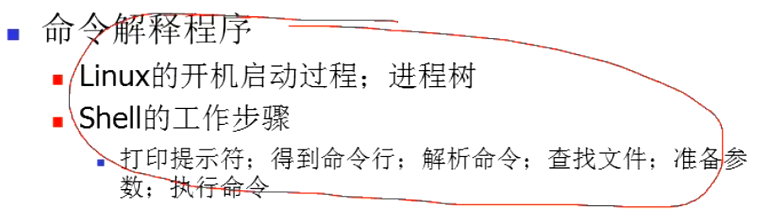
ash、ash.static、bsh：小的shell

bash、sh：来自GNU项目

csh、tcsh：C shell

ksh：Korn Shell

###### 什么功能是Shell进程本身提供的？哪些是Shell下的别的程序提供的？

重定向、管道是shell提供

###### 执行脚本文件的方式和区别（脚本不要求写，要求读懂）

命令行直接输入：只会执行一次

编写脚本文件，每次调用执行一次，执行方法如下：

 $sh [file]

 chmod +x [file]; ./[file]

 source [file]; . [file]

###### read的基本功能，引号的基本用法（单引号，双引号的区别）（读懂）

read target

单引号内的所有字符都保持它本身字符的意思，而不会被bash进行解释，例如，$就是$本身而不再是bash的变量引用符；\就是\本身而不再是bash的转义字符。

除了$、``（不是单引号）和\外，双引号内的所有字符将保持字符本身的含义而不被bash解释。

###### 参数变量和内部变量

参数变量：

$#: 传递到脚本的参数个数

$0: 脚本程序的名字

$1,$2,…: 脚本程序的参数

$\*: 一个全体参数组成的清单，各个参数之间用环境变量IFS中的第一个字符分隔

$@: “$\*”的一种变体，它不使用IFS环境变量

内部变量：用户在shell脚本里定义的变量

var=value

echo $var

###### 条件分支（While，Until掌握一个即可）

字符串比较：等于=、不等于!=、为空-z、不为空-n

算术比较：等于-eq、不等于-ne、大于-gt、大于等于-ge

逻辑比较：求反!、与-a、或-o

没有 == 只有 =

0是true 1是false

* If语句：

if [expression]

then

statements

elif [expression]

then

statements

else

statements

fi

注：可以把then写在上一行，用分号分隔。

if [expression]; then

* Case语句：

case str in

str1 | str2 | … ) statements;;

str3 | str4 | … ) statements;;

\*) statements;;

esac

注：str用变量表示时需要加上双引号

case “$answer” in

yes | y ) echo “Yes”;;

no | n ) echo “No”;;

\* ) echo “Sorry”;;

Esac

* For语句：

for var in list

do

statements

done

* While语句：

while [condition]

do

statements

done

* Until语句

until [condition]

do

statements

done

* Select语句

select item in itemlist

do

statements

done

###### 命令组合

分号串联：command1；command2

Add命令：&&

Or命令：||

###### $(())用作计算

x=$(($x+1))

###### 即时文档

定义：在shell脚本中向一条命令传送输入数据

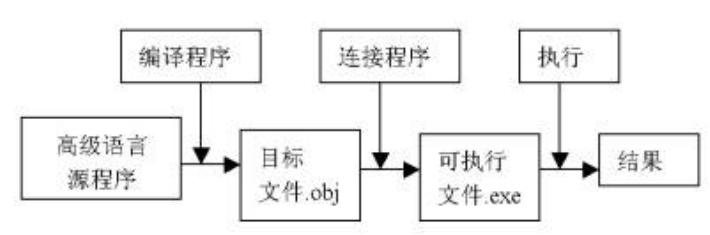
举例：

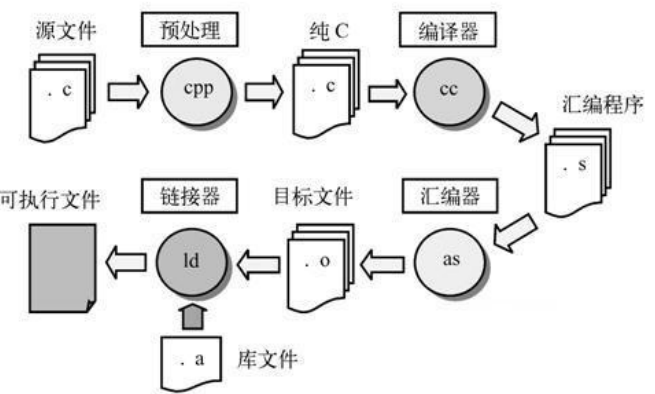
已知输入cat >> file.txt << World!，file.txt文件内容为Hello,

输出为Hello, World!

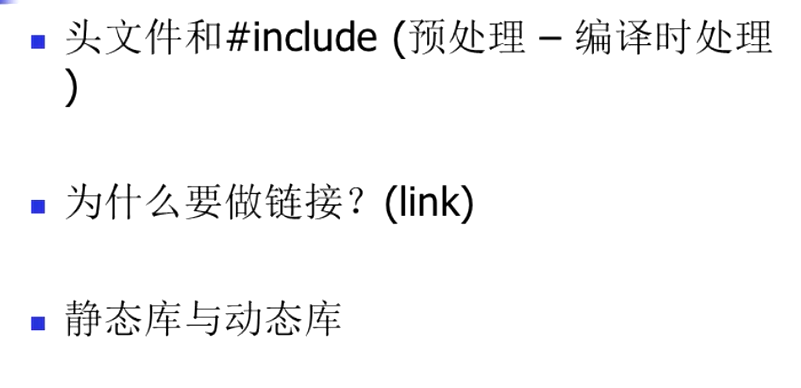
## Programming Prerequisite

###### 源代码是怎样编译成可执行码的？





###### 编译的流程，动态库和静态库的区别



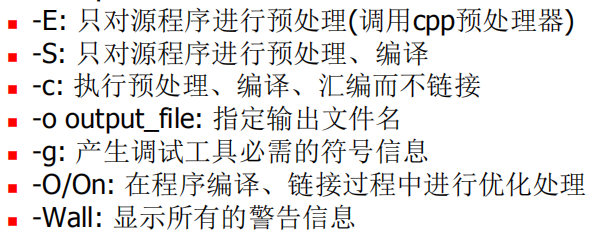
#include< > 引用的是编译器的类库路径里面的头文件。#include" " 引用的是你程序目录的相对路径中的头文件。

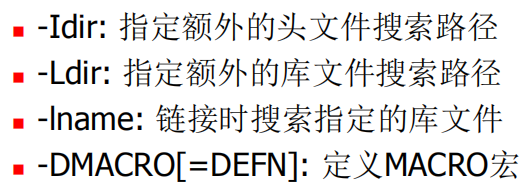
编译过程：

静态库：在链接阶段，会将汇编生成的目标文件.o与引用到的库一起链接打包到可执行文件中，对应的连接方式是静态链接。（特点：编译时期完成静态库与函数库的链接，后续操作与函数库无瓜葛，移植方便，浪费空间和资源）

动态库：链接时不会载入库，程序运行时才会载入（特点：增量更新，便于升级程序，进程间资源共享，减少空间浪费，链接载入时的显示调用）

###### gcc，g++的常见参数（ppt上的11个）





###### .o .a .so

.o文件：object file

.a文件：static library file (archive file)

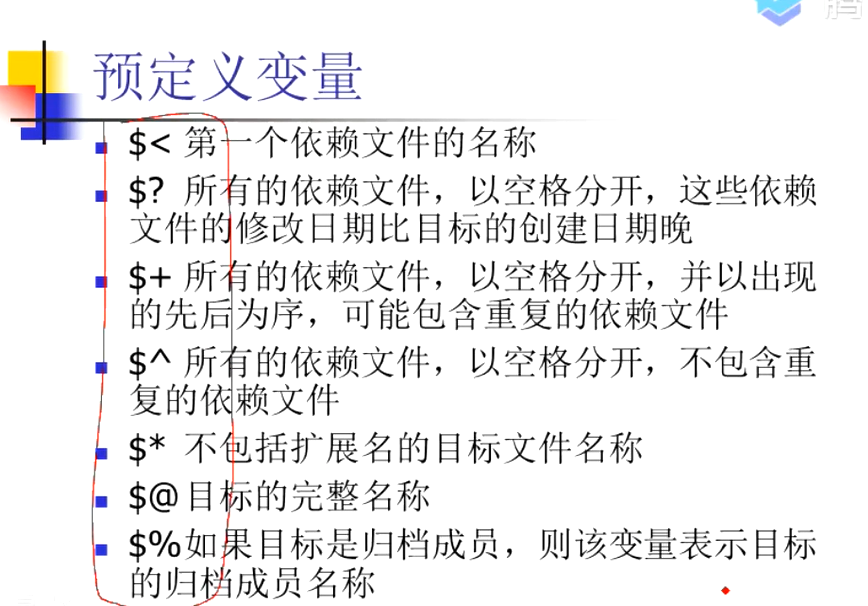
.so文件： dynamic library file(shared file)

###### make&makefile：只要求能读得懂，不要求写

###### makefile的流程要知道（怎么实现增量编译的？执行次序是怎样的？只考课上讲过的）

增量的实现：

Makefile中使用“依赖dependency”来实现增量编译



target:dependencies

<TAB>system command1

<TAB>system command..

依赖是一个文件列表，当有文件更新时，执行这条规则

注意：根据文件的修改时间来判断是否要更新，

比如某个依赖文件的时间比target的时间要新就编译

1. make会在当前目录下找makefile(Makefile)文件

2. 查找文件中第一个目标如A

3. 如果A不存在，或A的依赖文件最近更新的时间比A要新，则执行指定的命令来生成A文件

4. 如果A的依赖文件B不存在，那么在此文件中寻找目标为B的依赖文件，找到则根据命令生成B[是一个递归的过程]

5. Make根据B生成A

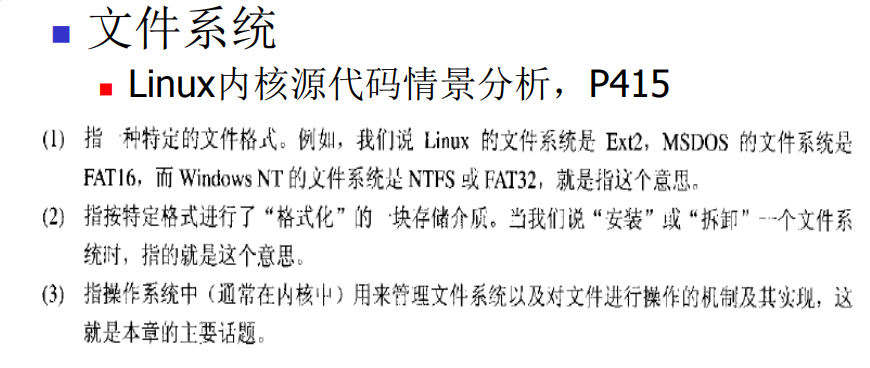


## System Programming

###### 文件，文件系统的概念、七种文件类型

文件：一个可以被读/写/可读可写的对象。文件有确定的属性，包括访问权限和类型等。

文件系统：文件和它们某些属性的集合。文件系统为指向这些文件的文件序列号提供了一个命名空间。



七种文件类型：

regular file -

character special file c

block special file b

fifo p

socket s

symbolic link l

directory d

文件结构：

Byte stream; no particular internal structure

###### VFS的作用

VFS（virtual file system）：为各类文件系统提供了一个统一的操作界面和应用编程接口。VFS可以让open()、read()、write()等系统调用不用关心底层的存储介质和文件系统类型就可以工作。

###### VFS的四种结构，分别表示什么意思？

super block（超级块对象）：存放系统中已安装文件系统的信息

i-node object（索引节点对象）：存放关于具体文件的一般信息

file object（文件对象）： 存放目录项与对应文件进行连接的信息

dentry object（目录项对象）：存放打开文件于与进程之间进行交互的有关信息

###### 硬链接和软链接的特点

硬链接：不同文件对应同一个inode；不能跨越文件系统，对应系统调用为link

ln

软链接：存储被链接文件的文件名来实现链接（而不是inode）；可跨越文件系统；对应系统调用为symlink

ln -s

###### 系统调用和库函数有什么差别？

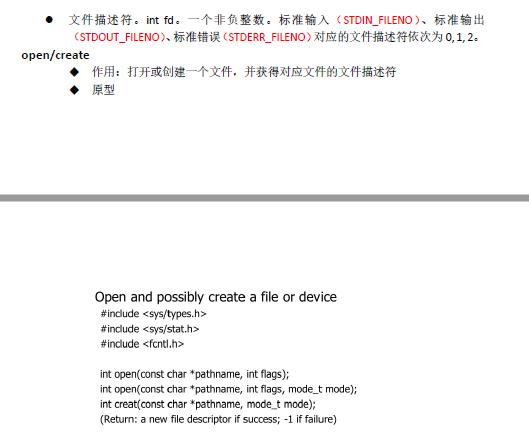
它们都以C函数的形式出现

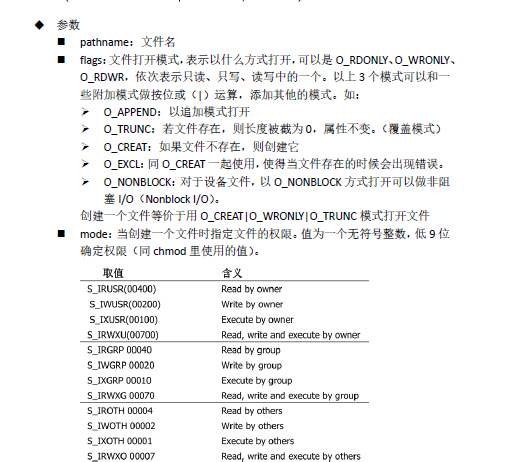
系统调用是Linux内核的对外接口，提供最小接口，需要切换到内核进行相关操作。编译运行速度快。

库函数依赖于系统调用，提供复杂的功能，可移植性好。

###### 库函数的缓存机制要求掌握例子

###### 系统调用要求写程序，不会单独地问（打开一个文件可读/可写/可读可写）





###### 文件的权限

r4 w2 x1

文件默认666，目录默认777

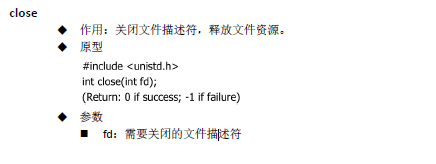
Umask是补码，用以保证不同用户的不同读写权限

umask = 默认 – umask

如drwxr-xr-x=（drwxrwxrwx）-（d----w—w-）=777-022=755

Rwx=7

###### 关闭文件



###### fcntl设置标志位不考 ioctl指向驱动

ioctl：控制io设备，request由对应设备的驱动程序实现，linux只是提供了一个统一的接口

###### 哪些调用是库函数/系统调用？不要搞混

###### 缓存：理解库函数的缓存机制带来的影响

###### 文件锁

文件锁的作用：几个进程同时操作一个文件时，用锁来保证数据的完整性。

记录锁：锁定文件的一部分，需要主动告知锁定的范围

劝告锁：只用于检测文件是否加锁，由应用程序自己控制是否进行读写（只建议）

强制锁：检查，加锁由内核控制 影响[open() read() write()]等

共享锁：多个进程可以使用同一把锁，常被用作读共享锁

排他锁：同时只允许一个进程使用，常被用作写锁

特殊的锁：

共享模式强制锁：某个文件被加上了共享模式强制锁，那么其他进程打开该文件的时候不能与该文件的共享模式强制锁所设置的访问模式相冲突

租借锁：当进程尝试打开一个被租借锁保护的文件时，该进程会被阻塞，同时，在一定时间内拥有该文件租借锁的进程会收到一个信号。收到信号之后，拥有该文件租借锁的进程会首先更新文件，从而保证了文件内容的一致性，接着，该进程释放这个租借锁。如果拥有租借锁的进程在一定的时间间隔内没有完成工作，内核就会自动删除这个租借锁或者将该锁进行降级，从而允许被阻塞的进程继续工作。

## Kernel Driver

###### 内核的定义

操作系统是一系列程序的集合，其中最重要的部分构成了内核

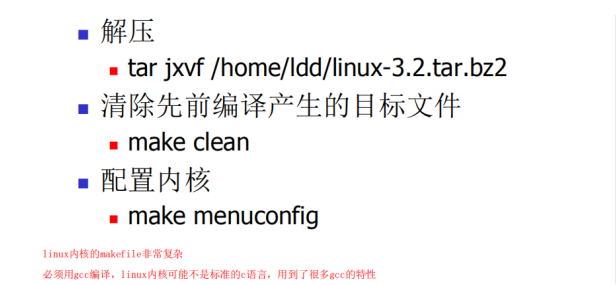
（来自网络）：内核是操作系统的内部核心程序,它向外部提供了对计算机设备的核心管理调用。

单内核/微内核

单内核是一个很大的进程，内部可以分为若干模块，运行时是一个独立的二进制文件，模块间通讯通过直接调用函数实现

微内核中大部分内核作为独立的进程在特权下运行，通过消息传递进行通讯

###### 内核编译步骤



使用内核的makefile，并加入make menuconfig的配置内核命令

###### 内核有哪些基本功能

内存管理，文件系统，进程管理，多线程支持，抢占式，多处理支持

###### 驱动

许多常见驱动的源代码集成在内核源码里

第三方开发的驱动可以单独编译成模块.ko文件

编译需要内核头文件的支持

内核态的程序

直接和硬件打交道，决定给硬件发什么信号

###### 加载和释放模块的命令

底层：insmod / rmmod

高层：modprobe / modprobe -r

###### 理解模块之间的依赖关系

自动按需加载，卸载

moddep

lsmod

modinfo

一个模块A引用另一个模块B所导出的符号，我们就说模块B被模块A引用

如果要装载模块A，必须先要装载模块B。否则，模块B所导出的那些符号的引用就不可能被链接到模块A中。这种模块间的相互关系就叫做模块依赖

###### Linux内核模块与应用程序区别（重点）（用户态程序和内核态程序的区别）

C语言程序（用户态程序）在用户空间运行，内核态程序在内核空间运行；

C语言程序（用户态程序）入口是main()，内核态程序入口由module\_init()指定；

C语言程序（用户态程序）没有出口，内核态程序出口由module\_exit()指定；

C语言程序（用户态程序）可以直接运行，内核态程序由insmod命令载入；

C语言程序（用户态程序）用gdb调试，内核态程序用kdbug，kdb，kgdb等调试；

###### 开发驱动程序有哪些注意点

不能使用C库来开发驱动程序

没有内存保护机制

小内核栈

并发上的考虑

###### 设备驱动（不考）

除了



###### 设备驱动类型

Linux系统将设备分为3种类型：

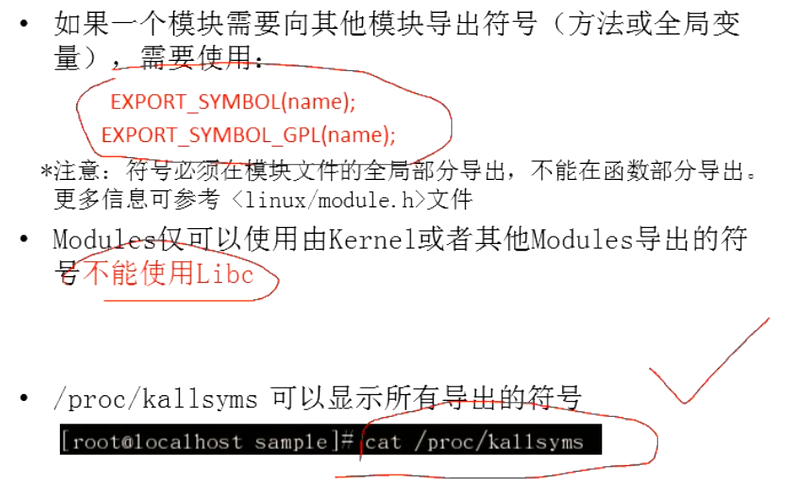
字符设备Character Driver->字符型文件

块设备Block Driver->块文件

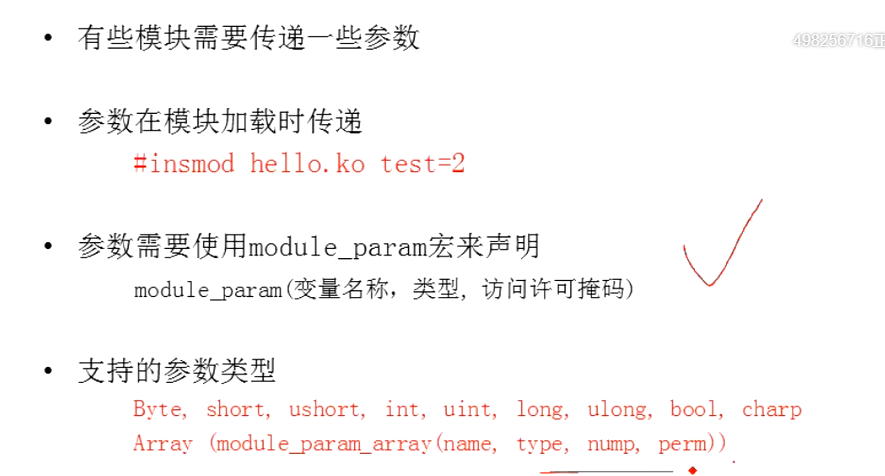
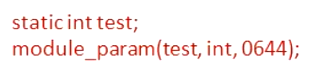
网络接口设备Network Driver->Socket

###### 内核模块操作/proc文件（选择判断）

###### 导出符号表（仅看画圈，不看实现）



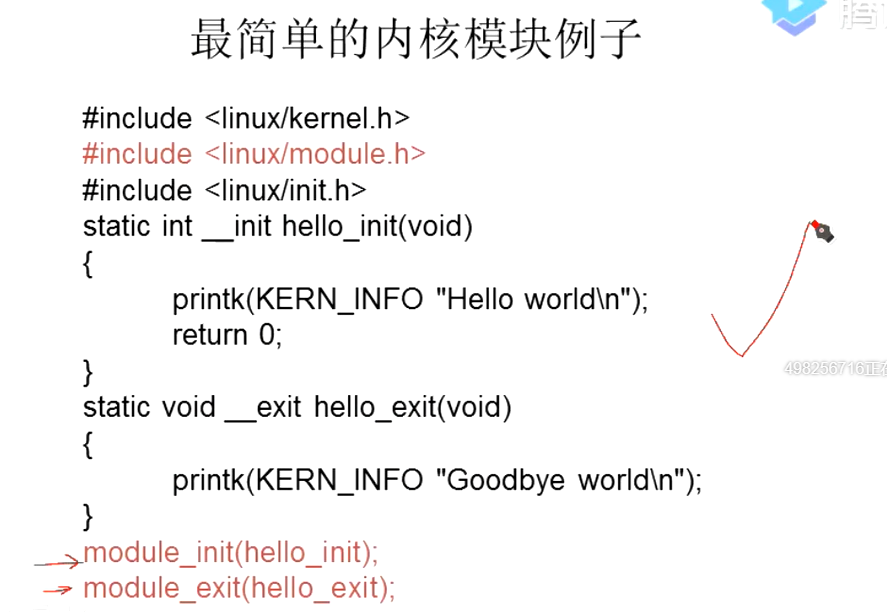
###### 模块参数传递



没有浮点类型

###### 编译内核模块

知道要用到内核源码的makefile



###### 内核模块操作/proc文件（选择判断）