linux复习大纲

**第1章**

1. 在Linux/UNIX系统中，通常将**命令解释程序称为shell**
2. 红旗版本Linux系统的默认shell是**bash;**
3. **系统调用**是操作系统内核与用户程序、应用程序之间的接口
4. Linux有两种版本，一个是核心（Kernel）版本，另一个是发行（Distribution）版本。

核心版本：主要是Linux的内核，官方版本由Linus本人维护

序号由三部分数字构成，其形式为: major.minor.patchlevel

例如：2.2.11表示对核心2.2版本的第11次修订。

约定：minor为奇数时，表示该版本加入新内容，但不一定很稳定，相当于测试版；minor为偶数时，表示这是一个可以使用的稳定版本。

1. Linux与其他操作系统相比，显著的特点
2. Linux系统必备分区及类型

1硬盘分区

**（1）分区的命名设计**

Linux通过字母和数字的组合来标识硬盘分区。具体如下：

|  |  |  |
| --- | --- | --- |
| 前两个字母 | 分区所在设备的类型 | hd：IDE硬盘 sd：SCSI硬盘。 |
| 第三个字母 | 分区在哪个设备上 | hda：第一块IDE硬盘 hdb：第二块IDE硬盘 sdc：第三块SCSI硬盘 |
| 数字 | 分区的次序 | 数字1-4表示主分区或扩展分区，逻辑分区从5开始 |
| 例如：/dev/hda3是指第一个IDE硬盘上的第三个主分区或扩展分区；/dev/sdb6是第二个SCSI硬盘上的第二个逻辑分区。  注意：如果硬盘上没有分区，则一律不加数字，代表整块硬盘。 | | |

（**2）分区前的考虑**

安装Linux需要一个**根文件系统分区**（类型为 ext3、ext2 或 reiserfs）和一个**交换分区**（类型为swap），这种分区方案适用于大多数用户。

**根分区（/）**：Linux 根文件系统驻留的地方。

**交换分区**（swap）：用来支持虚拟内存的交换空间，当没有足够的内存来处理系统数据时，就要使用交换分区的空间。交换分区的大小建议设置为计算机内存的1~2倍之间。

主分区最多四个：编号为1、2、3、4。

当需要将硬盘划分为更多分区时，可将其中一个主分区改为扩展分区。扩展分区必须划分为逻辑分区，才能使用。逻辑分区的编号从5开始。扩展分区最多1个。

硬盘分区表为64字节，每一个分区的信息占用16字节。因此主分区最多四个。

**第2章 常见命令**

1.命令补齐是指：当键入的字符足以确定目录中一个唯一的文件时，只须按 Tab 键就可以自动补齐该文件名的剩下部分，例如：

要把目录 /freesoft 下的文件 gcc-2.8.1.tar.gz 解包，

当键入到 tar xvfz /freesoft/g 时，如果此文件是该目录下唯一以 g开头的文件，这时就可以按下 Tab 键，这时命令会被自动补齐为：

tar xvfz /freesoft/gcc-2.8.1.tar.gz

2.bash命令的一般格式是：命令名 [选项] [参数1] [参数2] …

例如：将file1.c复制到myfile.c，并强制覆盖后者原有内容

cp –f file1.c myfile.c

**命令名必须是小写的英文字母**

1. 掌握常见命令的常见用法：who，cal,echo,date,which,clear,more,less,tail,head,find,grep
2. cat命令

一般格式：cat [选项] 文件

功能：在标准输出上显示文件的内容；连接两个或多个文件

例： $ cat f1 f2>f3 //把文件f1和f2的内容合并起来，放在f3中

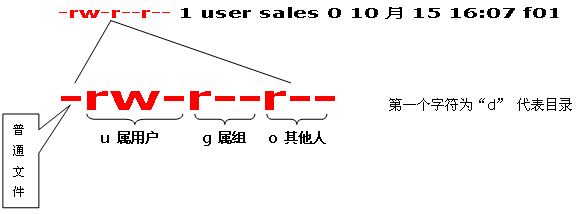
常用选项：

-b，--number-noblank 从1开始对所有非空输出行进行编号。

-n，--number 从1开始对所有输出行编号。

-s，--squeeze-blank 将多个相邻的空行合并成一个空行。

1. ls -aliR cp -afrR file1 file2
2. rm rmdir mkdir mv pwd ps which **软硬链接** ln file1 ln -s file2
3. cd / cd .. cd ../.. cd ./
4. **chmod** 更改文件权限



例：rwx　rw-　r--

r=读取属性　　//值＝4

w=写入属性　　//值＝2

x=执行属性　　//值＝1

chmod u+x,g+w f01　　//为文件f01设置自己可以执行，组员可以写入的权限

chmod u=rwx,g=rw,o=r f01

chmod 777 f01

chmod a+x f01　　//对文件f01的u,g,o都设置可执行属性

1. tar 压缩与解压

**压缩**

tar -cvf test.tar test/\*.\* //仅打包，不压缩

tar -zcvf test.tar.gz test/\*.\* //打包后，以gzip压缩 在参数f后面的压缩文件名是自己取的，习惯上用tar来做，如果加z参数，

**解压**

tar -zxvf test.tar.gz

**主要选项：**

-c: 建立压缩档案

-x：解压

-z：有gzip属性的

-v：显示解压或压缩所有过程

-f: 压缩文件名

**第3章 Vi**

1. vi编辑器有三种工作方式命令方式、输入方式和ex转义方式
2. 通过输入vi的插入命令( i )、附加命令( a )、打开命令(o )、替换命令( s )、修改命令( c )或取代命令( r )  **可以从命令方式进入到输入方式**
3. 输入方式回到命令方式的办法是按下<Esc>键
4. :wq 把编辑缓冲区的内容写到你编辑的文件中，退出编辑器，回到 shell下。

:ZZ 仅当作过修改时才将缓冲区内容写到文件上。

:x 与 :ZZ相同。

:q! 强行退出vi。感叹号（! )告诉vi，无条件退出，丢弃缓冲区内容

5.通常用户不能直接执行由文本编辑器建立的shell脚本，

vi直接编辑生成的脚本文件没有“执行”权限

**第4章 shell 程序编程**

1.输入定向到shell脚本 用输入重定向方式让shell从给定文件中读入命令行，并进行相应处理；如： $ bash < ex1

2.以脚本名作为bash参数 如：$bash ex2 /user/meng /usr/zhang；特点：可以在脚本名后面带有参数，从而将参数值传递给程序中的命令。

3.将shell脚本的权限设置为可执行， 然后在提示符下直接执行它。要用chmod命令将其设置为有“执行”权限。**如： #chmod a+x ex2;./ex2**

4.用户可以通过重新为环境变量赋值，改变存放历史命令的文件

$ HISTFILE="/home/mengqc/.myhistory"

还可以重新设定能够保留的命令个数 $ HISTSIZE=600

5.定义别名 $ alias ll=’ls -l’

$ my=/home/mengqc

（定义变量并赋值）

$ ll $my

$ alias

alias ..=’cd..’

alias cp=’cp -i’

注意：等号两边不能有空格

如果想取消先前定义的别名，可使用如下命令：unalias name…

可以一次将所有的别名都从别名表中删除，使用如下命令：unalias -a

6.一般通配符

\*（星号），它匹配任意字符的0次或多次出现

?（问号），它匹配任意一个字符

7.双引号:由双引号括起来的字符（**除$、倒引号（`）和反斜线（\）外**）均作为普通字符对待。

单引号 :由单引号括起来的字符**都**作为普通字符出现

例：echo “current directory is `pwd`”

echo “home directory is $HOME”

echo “file\*.?”

echo “directory ‘$HOME’ ”

$ str=‘echo “directory is $HOME” ’

$ echo $str

echo “directory is $HOME”

$echo ‘ The time is `date`, the file is $HOME/abc’

The time is `date`, the file is $HOME/abc

$echo current directory is `pwd`

current directory is /home/zhang

$ today=`date`

$ echo Today is $today

Today is 一 6月 26 21：40：18 CST 2006

**$ usrnum=`echo The number of users is \` who | wc -l\``**

**$ echo $usernum**

**$file=”m.c”**

**$ usrnum=`cat $file | wc -l`**

**$ echo $usernum**

8.输入出重定向符“<”“>”

作用：把命令的标准输出重新定向到指定的文件

一般形式是：命令 > 文件名

**如：$ who > abc**

**$ cat abc**

who命令的输出重定向到abc文件中，

屏幕上看不到执行who命令的结果，

可以查看abc文件的内容即可看到。

**输入和输出重新定向可以连在一起使用。例如：**

**$ wc -l < infile > outfile**

功能是：

命令wc从文件infile中输入信息，

按“行”统计后的结果送到另一个文件outfile中，不在屏幕上显示。

系统为每个进程自动打开三个标准文件

标准输入、标准输出、错误输出，

其文件描述字分别为0，1和2。

标准错误输出也可重定向到一个文件中，其一般形式是：

命令 2> 文件名

命令 2>> 文件名

**$gcc m1.c 2> errfile**

**功能是：**

**对C源文件m1.c进行编译，并把编译过程中产生的错误信息重定向到文件errfile中，**

**其中数字2表示标准错误输出的文件描述字。**

10.注释：以#开头的正文行表示注释

如果shell脚本中第一行是以“#!”开头，则后面所跟的字符串就是所使用的shell的绝对路径名。

对于bash脚本，第一行通常是： #!/bin/bash

这一行说明，该脚本是用哪一种shell编写的，从而调用相应的解释程序予以执行。

11.一个管道线中可以包括多条命令：

ls -l $HOME | wc –l

ls | grep m\*.c | wc –l //显示当前目录中文件名是以m开头的C文件的数目

12.后台命令 如：调用C编译器对C程序进行编译，如果在此同时想做别的事情，则要输入：

$ gcc m1.c&

在一条命令后面输入“&”，告诉shell该命令后台运行，

而shell马上显示提示符状态，可以输入新命令。

13.顺序执行

在执行时，以分号隔开的各条命令从左到右依次执行

pwd ; who | wc -l ; cd /usr/bin

逻辑与 &&

一般格式：命令1 && 命令2

其功能是：**先执行命令1，如果执行成功，才执行命令2；**

**否则，若命令1执行不成功，则不执行命令2。**

例： $cp ex1 ex10 && rm ex1

逻辑或||

一般格式为：命令1 || 命令2

**其功能是：先执行命令1，如果执行不成功，则执行命令2；**

**否则，若命令1执行成功，则不执行命令2。**

例： $cat abc || pwd

如果不能将文件abc的内容列出来，

则显示当前的工作目录的绝对路径。

1. **成组命令**

{ echo “User Report for ` date ` ”; who ; }

使用花括号时在格式上应注意:

左括号 “{ ”后面应有一个空格;

右括号“}”之前应有一个分号（ ;）

(echo "Current directory is ` pwd ` "; cd /home/mengqc ; ls -l ;

cp m1 em1 && rm m1; cat em1)

左括号 “( ”后面不需空格;右括号“)”之前也不需加一个分号

二者存在重要区别：

用花括号括起来的成组命令只是在本shell内执行，不产生新的进程；

而用圆括号括起来的成组命令要建立新的子进程,

因此不会改变父shell的变量值和工作目录等。

|  |  |
| --- | --- |
| 例：$ a=“current value”; export a  $ echo $a  current value  $ (a=“new value-1”; echo $a)  new value-1  $ echo $a  current value | $ a=“current value”; export a  $ echo $a  current value  $ { a=“new value-2”; echo $a; }  new value-2  $ echo $a  new value-2 |
| $ pwd  /home/mengqc  $ (cd /bin; pwd)  /bin  $ pwd  /home/mengqc | $ pwd  /home/mengqc  $ { cd /bin; pwd;}  /bin  $ pwd  /bin |

15. 变量赋值

定义变量并赋值的一般形式是： 变量名=字符串

例如： myfile=/usr/meng/m1.c

引用变量值： 在变量名前面加上一个符号“$”

如果变量值须出现在长字符串的开头或者中间，为了使变量名与其后的字符区分开，避免shell把它与其它字符混在一起视为一个新变量，则应该用花括号将该变量名括起来。

例如，

**$ dir=/usr/meng**

**$ cat ${dir}qc/m1.c**

**若输入：$ cat $dirqc/m1.c**

则系统会提示出错，因为它认为dirqc是个新变量，在前面没有对它赋值，其值为空串，所以无法找到m1.c

16.命令替换：将一个命令的执行结果赋值给变量

有两种形式的命令替换：一种是使用倒引号引用命令，其一般形式是：

**`命令`，如：$ dir=`pwd`**

另一种形式是：

$(用分号隔开的命令)

如：$ dir=$(pwd)

$ echo $(pwd ; cd /home/mengqc ; ls)

17.读取数组元素值的一般格式是：

${数组名[下标]} 例如：$ echo ${city[0]}

定义一个数组并为其赋初值的一般形式是：

数组名=(值1 值2 … 值n)

其中，各个值之间以空格分开。

例如：$ A=(This is an example of shell script)

$echo ${A[0]} ${A[2]} ${A[3]} ${A[6]}

若没有给出数组元素的下标，则数组名表示下标为0的数组元素

使用\*或@当作下标，则会以数组中所有元素取代[\*]或[@]

$echo ${A[\*]}

This is an example of shell script

表达式${#@}和${#\*}

它们的值分别是由$@和$\*返回的参数的个数。

表达式${#name[i]}

该表达式的值是数组name第i个元素值的长度(字符个数)。

表达式${#name[\*]}和${#name[@]}

它们的值都是数组name中已经设置的元素的个数。

18.read命令

可以利用read命令从键盘上读取数据，然后赋给指定的变量。read命令的一般格式是：

read 变量1 [ 变量2 …]

变量个数与给定数据个数相同，则依次对应赋值

变量个数少于数据个数，则从左至右对应赋值，但最后一个变量被赋予剩余的所有数据。

变量个数多于给定数据个数，则依次对应赋值，而没有数据与之对应的变量取空串

19.命令行实参与脚本中位置变量的对应关系如下所示：

**exam1 m1 m2 m3 m4**

**$0 $1 $2 $3 $4**

**例：set m1.c m2.c m3.c**

把字符串m1.c赋值给$1；m2.c赋值给$2；m3.c赋值给$3

但是$0不能用set命令赋值，它的值总是命令名

**每执行一次shift命令，就把命令行上的实参向左移一位，即相当于位置参数向右移动一个位置。**

命令行： ex7 A B C D E F

原位置参数： $0 $1 $2 $3 $4 $5 $6

移位后： $0 $1 $2 $3 $4 $5

注意：shift命令不能将$0移走，所以经shift右移位置参数后，$0的值不会发生变化。

shift命令可以带有一个整数作为参数

【例】 $ cat ex8

#!/bin/bash

#ex8: shell script to demonstrate the shift comand

echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9

shift

echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9

shift 4

echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9

#end

$ex8 A B C D E F G H I J K

输出：

ex8 A B C D E F G H I

ex8 B C D E F G H I J

ex8 F G H I J K

1. **export语句**

$ major=“computer science”

$ export major

$ cat exam12

major=“English”

echo “His major is $major”

exam13

$ cat exam13

echo “her major is $major”

$ ./exam12

His major is?

Her major is ?

20.只有使用 $((算术表达式))的形式才能返回表达式的值

$ echo “((12\*9))”// ((12\*9))

$ echo “$((12\*9))” // 108

**21.控制结构（详见书上例子） if,while,case(脚本编程必考)**

if test -f "$1"

then echo "$1 is an ordinary file . "

else echo "$1 is not an ordinary file . "

Fi

测试条件常见参数：

-e            该档名是否存在  
-f             该档名是否为普通文件（file)  
-d            该文件名是否为目录（directory)

**for语句**

格式是：

for 变量 [ in 值表 ]；do 命令表；done

根据循环变量的取值方式，其使用格式可分为三种：

格式一：

for 变量 in 值表

do

命令表

done

for day in Monday Wednesday Friday

do

echo $day

done

for 变量 in 文件正则表达式

do

命令表

done

**for file in m\*.c**

**do**

**Num=’cat $file | wc -l’**

**done**

**格式三：值表可以全部是位置参数，此时格式如下：**

for i in $\* 等价于 for 变量

do do

命令表 命令表

done done

示例：

dir=$1; shift

if [-d $dir]

then

cd $dir

for name

do

if [ -f $name]

then cat $name

echo “End of ${dir}/$name”

else echo “error filename: ${dir}/$name”

done

else echo “error directory name: $dir”

fi

算术表达式

for ((e1;e2;e3))

do

命令表

done

例： for ((i=1;i<=$1;i++))

do

for ((j=1;j<=i;j++))

do

echo –n “\*”

done

echo “”

done

**22.break命令和continue命令，见书上例子**

23.在shell脚本中可以定义并使用函数。其定义格式为：

[function]函数名( )

{

命令表

}

函数应先定义，后使用。调用函数时，直接利用函数名，如showfile，不必带圆括号

shell脚本与函数间的参数传递可利**用位置参数和变量直接传递**

$ cat ex17

valtest()

{

echo “Begin to test how to pass values”

echo $a $b $c

echo $1 $2 $3 $4

echo “The end of function valtest”

}

a=“current working directory”

b=“is”

c=`pwd`

**valtest Let’s begin to use Linux**

echo “Today is `date`”

$ ex17

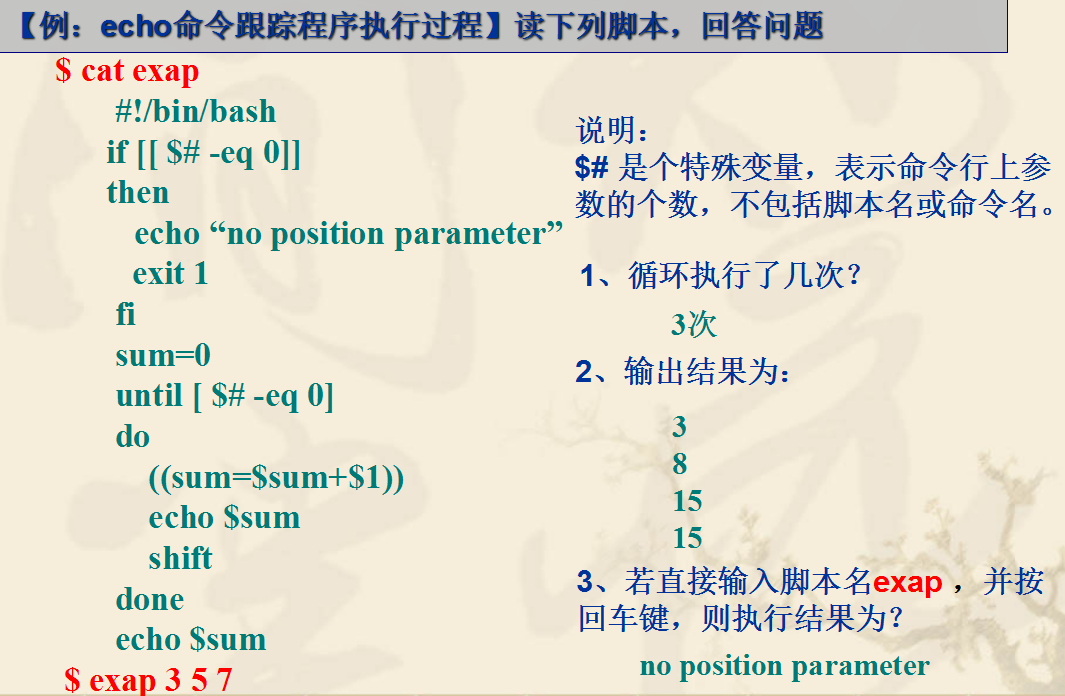
Begin to test how to pass values

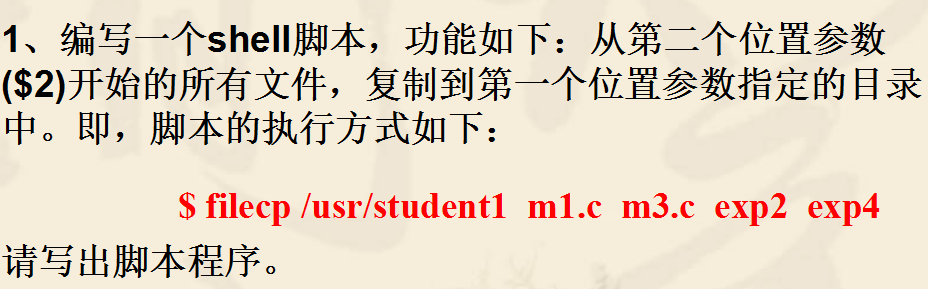
current working directory is /home/mengqc

Let’s begin to use

The end of function valtest

Today is 一 3月 24 10：30 CST 2008





**比如：n个C文件编写及统计行数的脚本**

**第6章**

当使用gcc时，gcc会完成**预处理、编译、汇编和连接**；前三步生成目标文件，

连接时把生成的目标文件链接成可执行文件；



-c 只生成目标文件，不进行连接。用于对源文件的分别编译

-S 只进行编译，不做汇编，生成汇编代码文件格式，其名与源文件相同，但扩展名为.s

-o file 将输出放在文件file中。如果未使用该选项，则可执行文件放在a.out中

-g 指示编译程序在目标代码中加入供调试程序gdb使用的附加信息

gdb常见命令：list,print

**@ 运算符：**

**print array@10 从基地址array开始的10个数组元素值**

**print array[3]@5 从array第三个元素开始的，5个数组元素值**

**设置断点 break**

break linenum

break linenum if condition

break function

break file:linenum

break file:function

break \*address

break

显示断点：显示程序中设置了哪些断点

info breakpoints [num]

info break [num]

维护停止点：清除和停用停止点

delete clear disable enable

**运行程序：设置断点后，用run命令运行程序**

程序的单步跟踪

设置断点后，可以让程序一步步地向下执行，用户可以仔细检查运行过程，实行单步跟踪的命令是step和next，其格式是：

step [N] 其中N为步长

next [N]

**两者区别是：**

**后者遇到函数调用时，执行整个函数，即将其作为一条指令对待；**

**前者进入函数内执行，每次仍然是执行N行语句。**

用make命令，必须编写一个叫做**makefile**的文件，这个文件描述了软件包中文件之间的关系，提供更新每个文件的命令；

一般在一个软件包里，通常是可执行文件靠目标文件（.o后缀）来更新，目标文件靠编译源文件来更新；

makefile写好之后，每次改变了某些源文件，只要执行make命令，所有必要的重新编译将执行。

make程序利用makefile中的数据、每个文件的最后修改时间来确定那个文件需要更新，对于需要更新的文件，根据makefile数据中定义的命令来更新。

makefile涉及三方面内容:

**目标文件、相依文件和操作命令**

test：prog.o code.o

gcc –o test prog.o code.o

 prog.o：prog.c prog.h code.h

gcc –c prog.c

 code.o：code.c code.h

gcc –c code.c

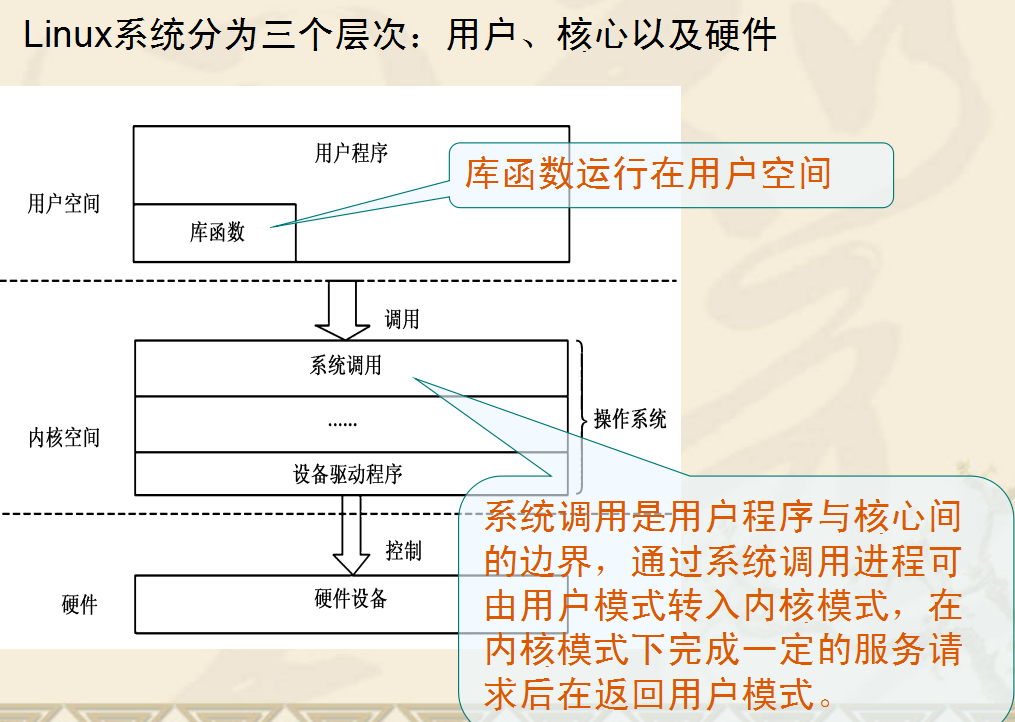
 clean：

rm –f \*.o

1. **环境编程（重点之重）**

1.一般的函数调用不能把进程状态从**用户态**变为**内核态**，而**系统调用**却可以，并将进程上下文从用户堆栈切换到系统堆栈。

**系统调用和库函概念及区别**



2.常用的有关文件操作的系统调用有：

creat，open，close，read，write，mkdir，rmdir，chdir

**熟练掌握open，close，read，write，见ppt上的例子**

1. 进 程 控 制：f**ork**、exec、clone的进程创建方法**，sleep的进程睡眠和exit的**进程退出调用；另外Linux还提供了父进程等待子进程结束的系统调用**wait，waitpid**。
2. **进程创建及通信（管道通信，信号通信，消息队列），详见书中及ppt上的例子，编程的结构（相关函数的的顺序问题）及输出结果的分析**
3. getpid(),getppid(),父进程与子进程哪个先结束，输出结果不一样
4. int filedes[2]，int pipe( filedes);创建管道，其中参数filedes存放两个整数，分别代表管道的读filedes[0]和写端口filedes[1]。即读和写对应的文件描述符

先write(filedes[1],mesg,n))，后read(filedes[0],buf,BUFSIZ))

7.发送信号kill(pid,sig)(**详见书中及ppt上的例子**)

将指定信号sig（不一定是杀死子进程，可以是其他信号！）发送给指定进程pid，其中参数为pid进程号，pid与sig均为整数值。

接收信号signal(sig,func)

接收到其它进程发送给它的信号后，完成指定的功能func，func一般是函数。

8.创建消息队列：(**详见书中及ppt上的例子**)

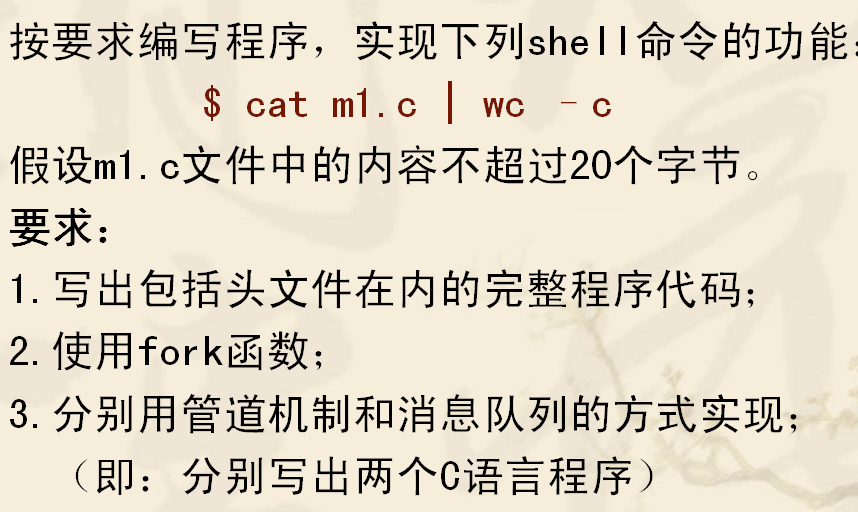
int msgget(key\_t key, int flags); 若执行失败，则返回-1(需在fork()之前)

发送消息：

int msgsnd(int msqid, struct msgbuf \*ptr, size\_t nbytes, int flags);

接收消息：

int msgrcv(int msqid, struct msgbuf \*ptr, size\_t nbytes, long type, int flags);



**第8章**

**用户账号的文件**

系统设置了多个文件，存放有关信息，最重要的3个为：

/etc/passwd文件

/etc/shadow文件

group文件

UID 用户号，唯一表示某用户的数字

GID 用户所属的私有组号，该数字对应group文件中的GID