1.3：UNIX基本概念

1.3.1：程序与进程

分为源程序和可执行程序。源程序要经过编译器编译后生成二进制代码文件称为可执行文件(程序)。UNIX还有另一种可执行程序，即shell脚本程序，例如用sh编写的脚本程序。脚本程序在执行过程中需要经过其他程序(如shell程序)的解释。

从用户的观点来看，进程是程序的一个执行实例。例如我们在终端键入执行某个程序的命令时便创建了该程序对应的进程。从UNIX系统内部来看，进程是运行程序并为程序提供执行环境的实体，是系统进行资源分配和调度的一个独立单位。

进程有一个控制点和自己独立地址空间。

进程具有生存期。当进程完成时，它会发出系统调用exit()而进入僵死状态，当系统释放了进程占用的所有资源的时候，该进程便消亡啦。

处于运行状态的进程有两种执行状态：用户态(如我)和核态(电脑，操作系统)，当进程发出系统调用的时候，便进入核态执行。进程在核态下执行的是操作系统的指令，这些指令完成用户的请求，如输入输出，存储分配等。

1.3.2：内核

实现管理计算机的硬件系统的软件叫做操作系统。UNIX中叫做内核，它是一个底层的，直接运行于硬件的程序，它控制着硬件，并创建，销毁和控制所有进程。在Linux中内核存放在/root/vmLinux的磁盘文件中。当系统启动时，一个称为字自举的特殊过程会将内核从磁盘中加载至内存并启动它运行。内核初始化整个系统，设置进程运行环境，创建几个初始进程，这些进程随后创建其他进程。

UNIX系统是层次模块结构，处在最内层的是UNIX内核，内核直接和硬件交互，向外提供UNIX系统调用接口。外层的程序，如shell以及who之类的实用程序，通过系统调用请求内核完成各种操作。

1.3.3：shell

UNIX不执行用户输入的命令，shell是UNIX系统的一个命令解释程序。用户输入的命令由shell来执行，起着协调用户和系统之间的一致性，在用户与系统之间进行交互的作用。用户一旦在UNIX中注册成功，系统就会为其创建一个进程来执行shell命令解释程序。这个shell称为注册shell，他负责读入并执行用户输入的命令。Shell常常可以接收两种形式的输入：来自终端的单条命令或者是来自一个文件的批处理命令(脚本文件),这种文件也称为叫做脚本文件或命令文件。

shell接收的命令有一些是内部命令，他们执行的是shell内部的使用程序。另外一些则是执行其他的实用程序，这些命令由shell将控制转交给对应的实用程序并启动它们执行。

1.3.4：用户名与用户ID，用户组与组ID

用户名用于系统的识别，用户ID用于检查是否有某项操作的权限

系统文件/etc/group记录了系统中存在的所有组和组ID，以及组内的用户

口令文件/etc/passwd记录了所有的注册用户，每一个用户有一个登记项

1.3.5：特权用户

及root用户，其用户ID为0

1.3.6：系统调用和库函数

我们通过调用库函数来完成程序，库函数通过调用一个或者多个系统调用来实现其功能，系统调用来对内核发出请求，系统调用一定会导致程序进入核态执行内核代码。

系统调用是UNIX内核提供的服务，而库函数处于内核之外，系统调用是我们无法更改的一种特性。

1.4：系统库

系统库给应用程序提供编译好的标准函数和系统调用函数的目标代码，这些代码在连接时与应用程序的目标代码装配在一块形成一个完整的可执行程序。如C标准库

如果程序中调用了其他库中的函数，则需要在编译时要指定连接相应的库，如使用了数学库，则需要在编译命令上加选项“-lm”指定连接数学库，调用了线程库中 的函数pthread\_create()，则要在编译命令中加上选项“-lpthread”指定连接Pthreads线程库。否则会报未定义的函数引用。

1.4.1：库文件

说明仅仅提供函数或者变量存在信息，定义才实际分配存储空间和确定函数的行为

1.4.2：保留字

是系统专门保留使用权并有特定含义的名字

1.4.3：特征测试宏

特征测试宏控制着编译一个源文件时头文件中有效符号名的确切集合

1.6：错误处理

头文件<errno.h>定义了变量errno以及它可以取值的错误码，每一个错误都是由大写E开头的宏名字。程序开始时，errno的初值为0，不应该用errno来检测一个调用是否失败，因为errno可能是前一次调用某个函数出错时的值。正确的做法是仅当函数的返回值指出这个变量被明显设置时才用errno来确定错误原因。

C标准函数strerror()可将错误码转换成可读的报错信息 char\* strerror(int errnum)返回与errnum错误码相对应的错误信息字符串，再次调用该函数，前一次调用得到的字符串将被覆盖。

如果想直接将errno当前值对应的错误信息输出到标准错误文件，则可用void perror(const char\* msg) perror首先打印msg指定的信息，后随一个冒号与空格，然后打印与errno对应的错误字符串，。如果msg是空指针或者指向指向空字符串，则将是和strerror打印完全相同的信息，但perror附加有换行符，它没有。

#include <errno.h>

#include <stdlib.h>

#define err\_exit(MESSAGE)

{

perror(MESSAGE)，exit(1)

}

1.7：系统信息

1.7.1：机器标识

每一台机器都有一个主机名。主机名由系统指定，函数int gethostname (char \*name, size\_t size); int sethostname(const char\* name, size\_t length)

gethostname返回以NULL字符结束的主机名于字符数组name之中，参数size给出此数组的字节大小，当size指定的空间不足的时候，返回-1. Sethostname设置主机的名字为name，那么是一个长度为length的字符串

1.7.2：硬件/软件类型识别

函数int uname(struct utsname \*info)可以用来查看运行该程序的计算机的操作系统版本和硬件信息 uname是一个系统调用，成功的时候返回非负整数，同时将操作系统和主机的信息保存在info所指的结构数据中，失败返回-1；utsname是一个结构

1.8：系统能力限制

学会使用联机帮助和百度就行啦

1.8.1：系统选项

这些宏一般定义在<unistd.h>中，则系统支持这些选项，否则系统可能支持也可能不支持，这时候就需要用sysconf()来查询

1.8.2：文件选项

<unistd.h>中定义了下面的宏名，则意味着选项对有些文件有效，对另外一些文件无效。为了查询一具体的文件是否有这些选项，要调用函数pathconf()或fpathconf()

sysconf用于获取与文件或目录无关的限制值，以及系统特征选项；pathconf和fpathconf用于获取与文件或者目录有关的限制值