《UNIX 程序设计》期末考试复习提纲

**（1）什么是操作系统？什么是 UNIX 操作系统？什么是 Linux 操作系统？它们之间的关系是怎样的？**

(1.1.1)操作系统（英语：operating system，缩写：OS）是管理计算机硬件与软件资源的计算机程序，同时也是计算机系统的内核与基石。操作系统需要处理如管理与配置内存、决定系统资源供需的优先次序、控制输入与输出设备、操作网络与管理文件系统等基本事务。操作系统也提供一个让用户与系统交互的操作界面。

(1.1.2)操作系统（Operating System，简称 OS）是管理和控制计算机硬件与软件资源的计算机程序，是直接运行在“裸机”上的最基本的系统软件，任何其他软件都必须在操作系统的支持下才能运行。

操作系统是用户和计算机的接口，同时也是计算机硬件和其他软件的接口。操作系统的功能包括管理计算机系统的硬件、软件及数据资源，控制程序运行，改善人机界面，为其它应用软件提供支持，让计算机系统所有资源最大限度地发挥作用，提供各种形式的用户界面，使用户有一个好的工作环境，为其它软件的开发提供必要的服务和相应的接口等。实际上，用户是不用接触操作系统的,操作系统管理着计算机硬件资源,同时按照应用程序的资源请求，分配资源，如：划分 CPU 时间，内存空间的开辟，调用打印机等。

UNIX 操作系统

Linux 是类 UNIX(UNIX alike)的操作系统(OS)，在源代码级上兼容绝大部分 UNIX 标准(IEEEPOSIX, System V, BSD)，是一个支持多用户， 多进程，多线程，实时性较好的功能强大而稳定的操作系统

**（2）UNIX 操作系统有哪些典型的分支？**

Linux,FreeBSD,Mac OS X(Darwin),Solaris,GNU 等分支。

**（3）什么是 Shell？内置变量$、$*、$@、$?、$# 的具体含义是什么？**

Shell 是一个命令行解释器，它读取用户输入，然后执行命令。Shell 的用户输入通常来自于终端（Terminal），有时也来自于 shell 脚本。

- $# 传递到脚本的参数个数；
- $* 以一个单字符串显示所有向脚本传递的参数；
- $$ 脚本运行的当前进程 ID 号；
- $! 后台运行的后一个进程的进程 ID 号；
- $@ 与$*相同，但是以多个字符串显示所有向脚本 传递的参数，每个字符串为一个参数；
- $- 显示 shell 使用的当前选项，与 set 命令功能相同；
- $? 显示最后命令的退出状态。0 表示没有错误，其 他任何值表明有错误；

**（4）常用的 Shell 命令，如 ls、cd、dd、pwd、ps、cp、chown、chmod、mkdir 等的具体用法。**

**ls [参数] [目录或文件]**

[说明]： ls 命令列出指定目录下的文件，缺省目录为当前目录./，缺省输出顺序为纵向按字符顺序排列。

[参数]： -a 列出所有文件，包括第一个字符为‘.’的隐藏文件； -l 长列表输出，显示文件详细信息，每行一个文件； -R 递归地列出每个子目录的内容； -r 逆序排列； -t 按时间顺序排列而非按名字； -F 在目录文件后加‘/’，在可执行文件后加‘*’

**pwd**

[说明]：本命令用于显示当前的工作目录。

**cd ［目录］**

**[说明]**本命令用于改变当前的工作目录。无参数时使用环境变量 $HOME 作为其参数，$HOME 一般为注册时进入的路径。

**mkdir［-m 模式］[-p]** 目录名

[说明]：用于建立目录。

[参数]：-m 按指定存取模式建立目录； -p 建立目录时建立其所有不存在的父目录；

**rmdir［-s］[-p] 目录名**

[说明]：本命令用于删除目录。

[参数]： -p：删除所有已经为空的父目录； -s：当使用-p 选项时，出现错误不提示；

**head［-n］[文件]**

[说明]：将文件的头 n 行显示输出，缺省值为 10 行。

[参数]： -n 整数，显示文件的前 n 行内容；

**tail［-n］[文件]**

[说明]：将文件的末 n 行显示输出，缺省值为 10 行。

[参数]：-n 整数，显示文件的末 n 行内容；

**more[参数] ［文件]**

[说明]：将文件显示在终端上，每次一屏。按回车键则上滚一行，按空格键则上滚一屏

[参数]： -c 显示文件之前先清屏；-n 行数 指定每屏显示的行数； +行号 从指定行号开始显示；

**cp[参数] 文件 1 ［文件 2 …] 目标**

[说明]：将文件 1[文件 2 ...]拷贝到目标上。若目标是文件名，则拷贝的文件只能有一个，若目标是目录，则拷贝的文件可以有多个，若目标文件不存在，则建立这个文件，若存在，则覆盖其以前的内容，若目标是目录，则将文件拷贝到这个目录下。

[参数]：-r 若文件名为目录，则拷贝目录下所有文件及子目录和它们的文件，此时 目标必须为目录；

**mv[参数] 文件 1 ［文件 2 …] 目标**

[说明]：将文件移动至目标，若目标是文件名，则相当于文件改名。

[参数]：-i　在覆盖已存在文件时作提示，若回答 y 则覆盖，其他则中止。 -f 覆盖前不作任何提示；

**rm[参数] 文件**

[说明]：删除文件或目录。

[参数]：-r　递归地删除目录及其所有子目录； -f　删除文件时不作提示；

**chmod[参数] 模式 文件**

[说明]：改变文件的存取模式；

[参数]：-R　递归的改变所有子目录下所有文件的存取模式；

**tar[参数]　文件或目录**

[说明]：可以为文件和目录创建档案；

[参数]：-c　创建新的档案文件，即打包； -r 把要存档的文件追加到档案文件的未尾； -t 列出档案文件的内容，查看已经备份了哪些文件； -u 更新文件； -x　从档案文件中释放文件； -f 使用档案文件或设备，这个选项通常是必选的； -v 详细报告 tar 处理的文件信息； -z 用 gzip 来压缩/解压缩文件，加上该选项后可以将档案文件进 行压缩，但还原时也一定要使用该选项进行解压缩。

[例子]：tar-cf　source.tar　*.c 将所有的.c 文件打包成 source.tar；

tar -rf source.tar　*.h　　将所有.h 的文件增加到 source.tar 的包中；

tar -tf source.tar　　列出 source.tar 包中的所有文件；

tar -uf source.tar　main.c　　跟新 source.tar 包中的 main.c 文件；

tar -zcvf　source.tar.gz　*.c　　打包并压缩*.c 文件为 source.tar.gz；

tar -zxvf　source.tar.gz　解压缩 source.tar.gz 包；

**find　路径名 文件名**

[说明]：递归地遍历指定路径下的每个文件和子目录，看该文件是否能使表达式值为真；

[参数]：-name 模式 文件名与模式匹配则为真； -depth 深度优先搜索； -print 显 示 输出使表达式为真的文件名；

**grep[参数]　模式 文件**

[说明]：指定文件中搜索模式，并显示所有包含模式的行；

[参数]：-c　仅显示各指定文件中包含模式的总行数； -i 模式中字母不区分大小写； -l 仅显示包含模式的文件名； -n 显示模式所在行的行号；

[模式]：* 匹配任意一个字符； ^ 匹配行开头； $ 匹配行结尾

**（5）什么是内部命令？什么是外部命令？它们的区别是什么？**

内部命令(built-in command)常驻内存

外部命令是一个个单独的可执行文件，储存在磁盘中，在需要的时候可以调出使用。

区别:1.内部命令不需要装载,在开机初始化完成之后即可试用;外部命令需要在启动 Shell 时保证装载，通常在环境变量 PATH 中需要保存外部命令所在路径。

**（6）什么是 C 语言的头文件？有哪些是操作系统提供的头文件？哪些是 C 语言的标准头文件？**

C 语言头文件通常以*.h 格式命名，主要编写函数声明，以供程序员使用。

操作系统提供的头文件:<unistd.h>, <fcntl.h>

C 标准头文件<stdio.h>,<stdlib.h>,<algorithm>

**（7）C 语言程序的基本结构是怎样的？编译器怎样搜索头文件？**

对于<>中指示的头文件，编译器会在包含目录中寻找，而不会在源目录中寻找； 对于""指示的头文件而言，编译器会首先在当前目录搜索，若查找不到会进而到包含目录中寻找；

**（8）怎样利用 gcc 编译源文件？gcc 的-o、-e、-static、-Wall 等选项的具体含义是什么？怎样使用？**

在 shell 脚本中输入：gcc［文件名］即可编译源文件

-o ： output，指定目标文件名

-e ： entry，后接一个参数表示入口地址

-static ： 使用静态链接函数库的方法编译

-Wall ： 生成所有警告信息

**（9）什么是动态链接库？什么是静态链接库？怎样用静态链接的方式编译 C 程序？**

动态链接库实质是一些在程序运行前由系统加载到内存中的常用的可执行文件，当程序执行过程之中需要用到这些函数时直接从动态链接库中读取并加载，从而大大降低编译时程序的文件大小。

这和静态链接恰好相反：静态库在程序连接的时候会自动的连接到程序里，所以一但编译完成，静态库也就不需要了。这个时候程序中已经包含静态库中需要使用到的函数代码，在运行时便可以脱离（本来所需要的）动态库。相对来说具有更高的效率，但同时也使得文件更庞大。

gcc -o test test.c /usr/lib/libm.a #指定路径

gcc -static

**（10）什么是文件系统？UNIX 的文件系统有什么特点？有哪些具体的文件类型？**

文件系统是操作系统用于明确存储设备（常见的是磁盘，也有基于 NAND Flash 的固态硬盘）或分区上的文件的方法和数据结构；即在存储设备上组织文件的方法。

Unix 可以把一个能随机存取的存储介质（如：硬盘、软盘和光盘）上的存储空间划分成一致多个区域，每个区域都可以像独立的物理设备一样单独进行管理和数 据存取，这样的存储区域,即是逻辑设备。在逻辑设备上按照一定的格式进行划分,就构成了逻辑文件系统，简称文件系统。

1.普通文件 这种文件包含了某种形式的数据,这些数据无论是文件还是二进制对于 UNIX 内核而言都是一样的。对普通文件内容的解释有处理该文件的应用程序进行。

2.目录文件 目录文件包含了其他文件的名字以及指向与这些文件有关信息的指针。对一个目录文件具有读权限的任一进程都可以读取该目录的内容,但是只有内核才能直接写目录文件。

3.块特殊文件 这种文件类型提供对设备带缓冲的访问,每次访问以固定长度为单位进行。

4.字符特殊文件 这种文件类型提供对设备不带缓冲的访问,每次访问长度可变。系统中的所有设备要么是字符特殊文件,要么是块特殊文件。

5. FIFO 这种类型文件用于进程间通信。也称为命名管道(namedpipe)。

6. 套接字(socket) 这种文件类型用于进程间的网络通信。

7. 符号链接(symbolic link) 这种文件类型指向另一个文件。

**(11)什么是文件的访问权限?使用 chmod 命令怎样改变访问权限?chmod 0777 是什么意思?**

访问权限决定什么用户可以以什么方式访问这个文件。文件访问者有 owner,group,others 三种,访问方式有读写执行三种。

在 shell 中执行 chmod [权限类型] [文件名] ,在 effective user 为 root 或该文件的拥有者的状态下即可修改访问权限。

chmod 0777 即赋予所有人 everybody 对文件的读写执行权限。

**(12)怎样解读 ls -l 命令显示的文件信息?**

ls -l (long)将当前目录下的所有文件打印在 shell 中,同时将对应文件的静态属性也打印出来。

状态如下:

-rwxr-xr-x 1 MikeBookPRO staff 8432 6 23 22:34 a.out

权限,硬链接数,所有者,组名,大小(byte), mm dd hh:mm , 路径名

**(13)什么是管道?什么是文件重定向?dup()、dup2()函数怎样使用?**

管道 pipe line 链接前后进程的标准输出与标准输入,将执行完的进程的标准输出内容作为输入给下一个进程处理。在 shell 脚本中用> ,<实现文件重定向: 将进程的标准输出或标准输入换成文件流。>>, <<

dup(int fd):复制一个现有的文件描述符

dup2(int fd, int fd2):与 dup 功能相似,在打开 fd 之后自动关闭 fd 的文件描述符。

利用函数 dup,我们可以复制一个描述符。传给该函数一个既有的描述符,它就会返回一个新的描述符,这个新的描述符是传给它的描述符的拷贝。这意味着,这两个描述符共享同一个数据结构。

**(14)什么是文件的静态属性和动态属性(文件描述符属性)?在文件描述符属性中,哪些是由进程维护的?哪些是由内核维护的?**

静态属性:由 stat()函数显示的文件属性都是静态属性。包括.... ls -l 的结果都是

动态属性：在打开文件之后知道的文件的属性，就比如文件的 offset

进程维护：FD_CLOEXEC (file descriptor close on exec() )

**（15）什么是会话（Session）、进程组？它们之间有什么关系？**

会话：一个或多个进程租的集合，通常在打开一个 shell 窗口之后就创立了一个新的会话

进程组：多个进程的集合。

关系：会话包含进程租，在一个会话创建之初会先创建一个进程，使得这个进程成为会话的首进程，继而成为一个新进程租的组长进程

**（16）父进程和子进程之间是什么关系？怎样在父子进程之间共享文件描述符？**

继承关系，父进程的所有内容会先拷贝给子进程，很多其他属性也会被子进程继承。包括：uid,grpid,euid,egrpid,grp,grpid,….

子进程创建之初就已经共享文件描述符。父进程与子进程共享同一个文件表项。故若要共享，需要在 fork()之前就将文件打开，即可共享文件描述符。

**（17）在一个进程中，文件描述符的增长规律是怎样的？例如，如果已经有 0、1、2、6 这样几个文件描述符，那么用 open()返回的下一个文件描述符是什么？**

总是打开最小的未被使用的非负整数。3

**（18）什么是 process id？父进程和子进程的 pid 之间有什么关系？**

process id 即进程 id，是一个进程独有的编号；

父进程总是会比子进程先创建出来，故父亲的 pid<子进程 pid。

**（19）什么是 C 语言程序的入口函数？在 C Startup Routine（start.S）中接受的 main 函数原型是什么？**

C 语言程序的入口函数是 main 函数。C 程序总是从 main 函数开始执行。

在 c 语言编译过程中，从_start 开始执行。

而_start 会调用 libc_start_main 函数，libc_start_main 这个函数的第一个参数就是指向一个 main 函数的指针

main 函数的签名为 int main(int argc, char **argv, char** environ)三个参数分别代表参数的个数，参数和环境变量。

**（20）什么是系统调用？什么是 C 语言库函数？它们之间有什么区别和联系？**

system call 是系统的函数，指操作系统较为底层的函数调用。

系统调用（英语：system call），又称为系统呼叫，指运行在使用者空间的程序向操作系统内核请求需要更高权限运行的服务。系统调用提供用户程序与操作系统之间的接口。大多数系统交互式操作需求在内核态执行。如设备 IO 操作或者进程间通信。

C 语言库函数是由 C 语言自带的库中实现的函数。很多 C 语言库函数(unistd.h,fcntl,h)大都会调用 system call 来实现比较底层的功能。

C 标准函数库（C standard library）是在 C 语言程序设计中，所有匹配标准的头文件（head file）的集合，以及常用的函数库实现程序（如 I/O 输入输出和字符串控制）。

C 函数可以移植性高，系统调用无法移植。

系统调用的函数签名在不同版本中基本保持一致

**（21）什么是 inode？里面存放什么信息？文件的文件名存放在哪里？**

inode 是索引节点，是 UNIX 操作系统中包含文件系统重要信息的一种数据结构。

里面存放了包括文件所有者，文件长度，文件在磁盘上的位置的指针等等信息。

文件的文件名存放在目录项中

**（22） C 程序的内存布局是怎样的？从低地址到高地址依次存放哪些段？**

从低到高：代码段，初始化过的数据，未初始化过的数据，堆，栈，命令行参数和环境变量其中程序的静态变量都放在代码段中。

在栈中，后入的变量居于低地址，先入的变量居于高地址

**（23） 怎样利用 fork()、exec()、waitpid()来创建和控制进程？**

父进程调用 fork()，之后程序中应该有一个条件判断判断是否为子进程，如果是，则子进程调用 exec，并且会将 if 中其他代码全部替换掉。之后，父进程执行 waitpid，回收子进程，防止其变为僵尸进程

**（24） 什么是孤儿进程、什么是僵尸进程？它们有什么特点？怎样避免产生过多僵尸进程？**

孤儿进程就是指父进程早于子进程结束，从而没有父进程的子进程

孤儿进程会被 1 号进程托管，并且在结束的时候被回收

僵尸进程就是指由于种种原因，父进程迟迟不使用 wait 和 waitpid 回收的子进程

僵尸进程一直在进程表中占有一个进程表象，但并没有占用任何资源（因为进程结束的时候资源早已回收）

可以使用两次 fork 的方法避免产生过多的僵尸进程。调用两次 fork 然后直接杀死子进程，留下孙子进程被 1 号进程托管，待其结束被 1 号进程回收。

**（25） 什么是前台进程？什么是后台进程？一个会话有几个前台进程组和几个后台进程组？**

前台进程：用户使用的有控制终端的进程

后台进程：是运行在后台的一种特殊进程。但仍联系控制终端并且周期性地执行某种任务或者等待处理某些发生的事件。

attention: 后台进程和守护进程并不是一个概念。守护进程完全独立于控制终端而后台进程并没有

在控制终端终止时，后台进程会随着前台进程一起终止，但是守护进程不会

守护进程会常驻内存，防止关掉控制终端内存消失

后台进程不等于守护进程

一个会话中包含一个前台进程组，一个或多个后台进程组。

**（26）C 程序如何退出并返回操作系统？exit()函数和_exit()/_Exit()函数的差别在哪里？**

进程的终止方式有 8 种,其中 5 种为正常终止,它们是

1. 从 main 返回。

2. 调用 exit。

3. 调用_exit 或_Exit。

4. 最后一个线程从其启动例程返回。

5. 最后一个线程调用 pthread_exit。

另外三种为异常终止方式,它们是

1. 调用 abort。

2. 接到一个信号并终止。

3. 最后一个线程对取消请求做出响应。

这三个函数用于正常终止一个程序:_exit 和_Exit 立即进入内核,exit 则先执行一些清理处理(包括调用执行各终止处理程序,关闭所有标准 I/O 流等),然后进入内核。

**(26) exec 函数族包含哪些具体的函数?**

execl,execv,execle,execve,execlp,execvp,fexecve(关系看 P202) 其中 execve 是系统调用,其它都是普通函数。

l = 参数显示表示 : arg0 …., (char*)0

v = 参数用数组表示 : char *const argv[]

e = envp[] 不使用当前环境变量,将其他变量加入到环境之中

p = filename,在 PATH 中寻找这个 file

f = fileno

**(28)什么是信号?SIGINT、SIGSTOP、SIGHUP、SIGALARM、SIGQUIT 等信号是如何产生的?缺省的处理动作是什么?**

在计算机科学中,信号是 Unix、类 Unix 以及其他 POSIX 兼容的操作系统中进程间通讯的一种有限制的方式。它是一种异步的通知机制,用来提醒进程一个事件已经发生。当一个信号发送给一个进程,操作系统中断了进程正常的控制流程,此时,任何非原子操作都将被中断。如果进程定义了信号的处理函数,那么它将被执行,否则就执行默认的处理函数。

- SIGALRM 是闹钟信号,当由 alarm 函数设置的计时器超时后产生此信号。进程的信号屏蔽字复原为调用信号处理屏蔽字之前。

- SIGQUIT,当用户在终端上按退出键(一般是 Ctrl+\)时,中断驱动程序产生此信号,并发送给前台进程组中的所有进程。

- SIGSTOP,作业控制信号,用以停止一个进程,当用户在终端上按挂起键(一般是 Ctrl+Z)时,终端驱动程序产生此信号。

- SIGHUP,如果终端接口检测到一个连接断开,则将此信号送给与该终端相关的控制进程(会话首进程)。

- SIGINT,当用户按中断键(一般是 Delete 或 Ctrl+C)时,终端驱动程序产生此信号并发送至前台进程组中的每一个进程。

缺省的处理动作都是终止。

**（29）什么是硬链接和软链接（符号链接）？读取软连接的函数是什么？（readlink）**

硬链接：通过 i 节点链接使多个目录项指向同一个文件的这种链接类型。

符号链接：对一个文件的间接指针，一般用于将一个文件或整个目录结构移到文件系统中的另一个位置。

readlink 函数打开符号链接本身,并读取该链接中的内容(不是该链接所引用的文件的内容)。

#include <unistd.h>

ssize_t readlink(const char *restrict pathname, char *restrict buf, size_t bufsize);

返回值:若成功则返回读到的字节数,若出错则返回-1。

如果此函数成功执行,则返回读入 buf 的字节数。在 buf 中返回的符号链接的内容不以 null 字符终止。

**（30）函数 link()和 unlink()的作用是什么？什么时候文件占用的磁盘空间才会真正被释放掉？（两个条件）**

link：创建一个新目录，引用旧目录

unlink：删除一个现有的目录项，并将由 pathname 所引用的文件计数-1

真正释放的条件：1.硬连接=0 2.未被任何进程打开或已经处于关闭状态

**（31）什么是可重入函数？怎样判断一个函数是不是可重入函数？**

可重入函数主要用于多任务环境中,一个可重入的函数简单来说就是可以被中断的函数,也就是说，可以在这个函数执行的任何时刻中断它，转入 OS 调度下去执行另外一段代码，而返回控制时不会出现什么错误。

（也可以这样理解，重入即表示重复进入，首先它意味着这个函数可以被中断，其次意味着它除了使用自己栈上的变量以外不依赖于任何环境（包括 static），这样的函数就是 purecode（纯代码）可重入，可以允许有该函数的多个副本在运行，由于它们使用的是分离的栈，所以不会互相干。）

可以被多个任务调用的过程，任务在调用时数据不会出错，输入数据相同就应产生相同的输出。

**（32）什么是带缓冲的输出和不带缓冲的输出？当父进程的输出缓冲区还未清空时，调用 fork 创建子进程，会出现什么情况？**

不带缓冲的输出：在用户的进程中对这这类的函数不会自动缓冲，每次执行就要进行一次系统调用，针对文件描述符操作。

带缓冲的输出：在系统调用的上一层多加了一个缓冲区，针对流来操作。

如果标准输出连接到终端设备，则它是行缓冲的，否则它是全缓冲的。即以交互方式运行时，只得到父进程中 printf 输出的行一次，原因是标准输出缓冲区由换行符冲洗。但是

当将标准输出重定向到一个文件时，却得到 printf 输出两次，原因是在 fork 之前调用了一次 printf，但当调用 fork 时，该行数据仍在缓冲区中，然后在将父进程数据空间复制到子进程中时，该缓冲区数据也被复制到子进程中，此时父进程和子进程各自有了该行内容的缓冲区。

**（33）编程的题目在这次考试中占很大的比重，要求详细掌握用法的函数包括：**

fork、waitpid、execlp、open、read、write、lseek、close、malloc、free 等。