《Linux 编程环境》期末考试样题

一. (共 20 分，每小题 2 分)给出完成下列各项功能所需要命令。陈述利用了该命令的哪个功能或选项。

1. 如何得知命令 netstat 众多选项中哪个选项可以打印出 IP 路由表？

   答：使用 man 命令，man 命令可以在线查阅命令的使用手册，使用命令 man netstat 查出 netstat 命令的使用手册，看看哪个选项和什么命令格式用来打印出 ip 的路由表。

2. 删除文件名为-f 的文件。

   答：使用 rm 命令，由于命令会默认以减号开始的命令行参数为命令处理的选项而不是处理对象，所以需要使用特殊的选项--以显式的方式说明选项的结束。命令为　rm -- -i

3. 在你完成上机作业过程中，使用什么命令将源程序文件 mylist.c 编译、链接生成可执行文件？可执行文件是什么名字？如何运行？

   答：使用 gcc 命令或者 make，实现编译和链接

   gcc　mylist.c –o mylist

   或者：make mylist

   生成的可执行文件名按照惯例为 mylist，一般情况下当前环境变量 PATH 不包括当前目录，所以，运行这个可执行程序应当加上当前路径，命令为：./mylist

4. 去掉文件 list.txt 中的所有空行(所谓空行指：行内不含有任何除空格之外的字符)，存为新文件 list-new.txt。

   答：使用 grep 命令可以用正则表达式对文本文件过滤，-v 选项用于筛选掉能匹配指定正则表达式的行，描述一个空行的正则表达式为^ *$，即：从行首开始(^)，有零个到多个空格(*)，然后是行尾($)，命令为：

   grep –v '^ *$' list.txt > list-new.txt

5. 检索目录 src 以及其子孙目录中的所有文件名后缀为.c 和.h 文件，查找哪些文件中含有字符串 TPDU，并列出在这文件中的行号。

   答：使用 find 命令和 grep 命令。find 命令可以在指定的目录树中查找满足某个条件的文件或目录，并对查找到的满足条件的对象执行一个动作。指定查找条件为"文件名后缀为.c 和.h"，动作为"查找哪些文件中含有字符串 TPDU，并列出在这文件中的行号"，分别是 find 的 -name 和-exec 选项。完整的命令为：

   find src –name "*.[ch]" –exec grep –n TPDU {} /dev/null \;

6. 文本文件 fexc.c 有几千行，保留了备份 fexc0.c 后另个程序员在 fexc.c 中作了多处修改，如何找出他在原先程序的基础上作了哪些修改？

答：diff 命令可以对文本文件进行比较，找出两个文件的差别。由于 C 语言源程序文件为文本文件，可以用 diff 完成上述功能。完整的命令为：

diff fexc0.c fexc.c

7. 用户使用低速网络从另台 Linux 中下载了一个 2.2GB 数据文件，他怀疑下载完该文件后远端文件又做了更新。如何用简便的方法验证本地和远端的两个文件的内容是否完全相同？

答：使用 md5sum 命令或 sha1sum，分别利用本地和远端文件的 16 字节“消息摘要”，通过比较“消息摘要”来判断两文件内容是否完全相同。

8. 目录 work.d 上周的备份目录为 work.bak，仅有一小部分文件作了修改，如何仅仅将被修改过的文件和新创建的文件拷贝到备份目录？

答：cp 命令实现了增量拷贝的功能，增量拷贝的基本功能就是检查源目录下的文件和目的目录下的同名文件的最后一次修改时间，当源文件的最后一次修改时间晚于目的文件的最后一次修改时间，或者同名的目的文件不存在时，才执行复制操作。完整的命令为：

cp –ur work.d work.bak

9. 程序 xftpd 正在运行，随时间的变化，系统的内存资源越来越紧张。如何判断该程序在运行过程中是否存在内存泄漏？

答：ps 命令的-l 选项，可以打印出进程当前的 SZ 属性，即：进程的虚拟内存空间大小。当进程在运行过程中存在内存泄漏时，进程的虚拟内存空间大小会逐渐随时间增大。使用 ps –l | grep xftpd 可以看出进程 xftpd 属性。

10. 如何让系统每 10 秒一次周期性地列出正在下载的文件 a.dat 的大小？

答：自动地重复执行一个命令，利用 shell 的循环机制。让 shell 睡眠 10 秒钟的命令为 sleep 10。整个命令为：

while true; do ls –l a.dat; sleep 10;done

二．(共 20 分，每小题 2 分)下列说法是否正确，简要阐述理由。

1. Linux 中普通用户忘记登录口令，可以通过超级用户 root 查出当前口令，然后重新登录。

答：错误。root 也不能查出用户的登录口令，但是可以重设登录口令，让用户使用新口令登录。

2. Linux 中命令 ls -l 与命令 ls -l *的执行结果相同，都是列出当前目录下

的所有文件。

答：错误。ls -l 列出当前目录下的所有文件和目录。*代表当下目录下的所有目录及文件，ls -l *列出当前目录下的所有文件（不包括目录）以及当前目录下所有目录中的内容。

3. 命令 ln 仅允许对普通文件实现硬连接，对目录不许硬连接操作。所以，所有目录的 link 数总为 1。

答：错误。尽管命令 ln 仅允许对普通文件实现硬连接，对目录不许硬连接操作，但是，目录的硬连接由系统自动实现，当前目录下的.文件和子目录中的..文件，均是指向当前目录的硬连接，在创建新目录时系统自动实现。所以，目录的 link 数会大于 1，一般情况下目录的 link 数=直属子目录数+2。

4. Linux 文件权限设计为简单的三级控制，用户 liu 对用户 sun 的文件 data.txt 要么具有全部的读权限，要么不可以读。因此，没有办法限制 liu 只对文件的指定部分读。

答：错误。可以利用 SUID 权限，用户 sun 将文件 data.txt 的读写权限设置为 rw-------，由文件所有者 sun 自己编写程序以实现对文件的访问，程序中的访问当然可以限制只对文件的指定部分读，但是该程序文件的属性应当为 rws--x--x，用户 liu 只有执行这个可程序程序文件才能实现对文件 data.txt 的访问。

5. Linux 的进程调度程序能保证大部分进程处于运行状态，只有少数进程处于阻塞状态，否则，系统的性能将大大下降。

答：错误。事实上，恰恰相反，大部分进程处于“阻塞”状态，只有所等待的条件满足后才能转换为“运行状态”。进程调度程序仅调度那些运行状态的进程。将进程的状态从“阻塞”转换为“运行”不是调度程序的任务，仅取决于外部条件的变化。

6. 程序从启动运行到运行结束总共持续了 t 秒，占用用户时间为 t1 秒，系统时间为 t2 秒，所以有 t=t1+t2 成立。

答：错误。除了 t1,t2，还有进程阻塞的时间。一般来说，会有 t>t1+t2.

7. fork 后得到两个几乎完全相同的进程，但是对内存的占用不会成倍增加。

答：正确。fork 后，从逻辑上可以得到两个几乎完全相同的进程，父子进程有独立的数据段、堆栈段和指令段，但是，操作系统可以通过使用 copy-on-write 技术等方式，使得父子进程共享同一段物理内存，除了代码段可以共享之外，也可以共享数据段和堆栈段。

8. 在 bash 中，圆括号和花括号都可以括起一组命令，两者没什么区别。

答：错误。首先，语法不同：（）是元字符，而｛｝不是，语法分别

是（list）和{ list;}。其次,语义不同：（）执行需要重启一个子 shell 进程，在子进程中执行圆括号内的各个命令，而｛｝在当前 shell 进程中执行。

9. 使用 socket 利用 TCP 协议编写通信程序，bind 调用只应该在服务端使用，客户端使用这一调用没有意义。

答：错误。bind 调用的目的是指定一个网络连接的本地端点名。客户端程序一般不使用 bind 调用，操作系统自动为 socket 分配本地 IP 地址和本地 TCP 端口号。如果本计算机有多个 IP 地址，客户要求必须使用其中的某个地址,或者,客户需要指定本地端口号,那么就需要用 bind 调用，然后再执行 connect 调用。

10. 程序用 socket 机制通过系统调用 ret=send(sock,buf,1024,0)向远端 TCP 主机发送 1024 字节数据，系统调用执行成功的标志是返回一个非负数的整数值，ret 的返回值为 1024，由于 TCP 提供了可靠的传输层服务，所以可以断定这 1024 字节数据已经成功地传输到了接收端进程。

答：错误。send 成功返回只能说明 1024 字节的数据已经拷贝到发送缓冲区中，但不能确定已经成功地传输到了接收端进程。

三．简答题（每题 4 分，共 40 分）

1. 在字符终端上执行 cat /usr/bin/*之后屏幕开始显示乱码，按下 Ctrl-C 中止后，无论从键盘输入什么内容，屏幕回显的内容都是各种线条等制表符号。解释产生这一现象的原因。

答：由于这个目录下是一些可执行文件，不是文本文件。不可正常打印的字节序列顺序发往终端，当终端收到的字节序列中凑巧有正好是一个终端控制转义序列，这个转义序列导致终端字符集发生了改变。

2. 用户在字符终端上使用 vi 命令编辑文件，当按下 Ctrl-S 执行存盘(Save)操作后，终端就进入了冻结状态，随后从键盘输入了多个命令，终端显示不发生任何改变。解释产生这一现象的原因。

答：尽管许多 Windows 编辑器 Ctrl-S 是存盘热键，但是在 Linux 终端里，Ctrl-S 是流控字符 XOFF（对应的 ASCII 码为 19）。当主机收到 XOFF 后,主机认为终端停止数据接收,主机就停止输出,直到收到 XON 字符（对应的 ASCII 码为 17），流控状态解除。XON 字符对应的热键是 Ctrl-Q。所以，在按下 Ctrl-Q 之前，终端的输出会一直处于“冻结”状态。

3. 写出能满足下列要求的正则表达式:

(1) 第一个字符必须是字母，其余字符必须是字母或数字或下划线。

(2) 匹配 C 语言算式 a[i]*b[j]，允许星号两侧有多余的空格。

写出 vi 中能实现下列替换要求的命令：

(3) 将格式为“日-月-年”的日期数据，如：27-06-2017，替换为

“年.月.日”格式，如：2017.06.27。

(4) 将 HTML 文件中所有以尖括号括起来的标签部分替换为空格。

答：

(1) [A-Za-z][a-zA-Z0-9_]*

(2) a\[i] ** *b\[j]

(3) s/\([0-9][0-9]*\)-\([0-9][0-9]*\)-\([0-9][0-9]*\)/\3.\2.\1/

(4) s/<[^<>]*>/ /

4. Linux 中某个普通磁盘文件对所有用户赋予了读写权限，但删除该文件失败。导致可修改文件无法删除的原因是什么？

答：应该是文件所处的目录没有写权限。因为要删除这个文件，只需要能对文件所在的目录可写，对文件自身的权限没有任何要求。

5. 在当前目录**/home/sun** 下执行 **chmod 000 .** 命令后， 随后执行 **chmod 777 .** 命令失败。为什么会失败？这样导致当前目录无法访问，如何解除这一困境？

答：. 代表当前目录，**chmod 000 .** 命令修改了当前目录的读写可执行权限，使得无法对当前目录中名称为“.”的文件进行检索访问和修改目录结构。因而 **chmod 777 .**命令会失败。

使用 **chmod 777 /home/sun** 可以解除这一困境，因为这个命令访问的是**/home** 目录，只要**/home** 目录有 **x** 属性，就可以修改这个目录下的 **sun** 的属性。这个过程中对当前目录的属性没有任何要求。

6. 在 Windows 下输入命令名，系统首先检索当前目录下是否有这个可执行文件，检索不到，会逐个检查 **PATH** 环境变量列出的各目录。Linux 不自动检索当前目录,而且环境变量 **PATH** 中一般不含当前目录**./**,为什么？

答：出于安全方面的考虑，假如用户 **a** 在**/tmp** 目录下创建名为 **ls** 的恶意程序，并对所用用户赋予执行权限 **x**。当用户 **b** 在**/tmp** 目录下执行命令 **ls**，由于 **PATH** 中含有**./**分量，将以用户 **b** 的身份执行**/tmp/ls**，因而造成安全问题。

7. 当前目录下具有可执行属性的 bash 脚本文件 **a.sh**，在 bash 提示符下执行命令**./a.sh** 与**. a.sh** (将前个命令的斜线换成空格)，有什么区别？

答：./a.sh 将创建子进程，在子 shell 中执行当前目录下的 a.sh 脚本里的程序；. a.sh 将在当前的 Shell 中执行 a.sh 脚本。

8. 命令 **gcc myap.c -o myap** 因源程序有大量问题，错误信息很多，为什么用 **gcc myap.c -o myap | more** 不能做到每显示一屏后暂停等待按下空格键再继续显示？要达到这个目的，应怎样使用这些命令？

答：gcc 将编译链接过程产生的错误信息输出到标准错误文件 stderr 中，而管道只是将前一个命令的标准输出 stdout 管道到下一个

命令的标准输入 stdin，并不影响标准错误 stderr 的输出。正确的用法是

gcc myap.c -o myap 2>&1 | more

其中，2>&1 将标准错误输出 **stderr** 输出到与标准输出 **stdout** 相同的地方。**stderr** 和 **stdout** 对应的文件描述符分别是 2 和 1。

9. 在同一台 Linux 上的两个进程之间交换数据可以通过下列两种方式。一，使用 TCP Socket 完成；二，两进程共享内存，并使用信号量(semaphore)的 P/V 操作操作完成。两种方式各有那些利弊？

答：使用 TCP socket 对程序员来说操作简便，但是数据需要在内核与用户进程之间进行两次拷贝。共享内存对于程序员操作复杂，但是不需要拷贝数据，效率较高。

10. 在 bash 命令提示符下输入下列内容后按下回车，会导致 Linux 操作系统崩溃吗？如果系统不崩溃的话，会出现什么现象？为什么？

a(){a; };a

答：这个命令首先定义了一个函数 a，函数中的内容就是调用自己（递归）；其次，调用函数 a。

这种递归死循环不会导致系统崩溃，但是会导致当前的 shell 进程的堆栈段不停生长，当超过极限之后，当前 shell 进程崩溃。

四．程序分析题（共 10 分）

某 Linux 系统在 TCP 端口 12345 提供一种专用的文件传输服务。下面的服务器端程序采用“多进程并发”的方法同时为多个客户端服务。每到达一个 TCP 连接就创建一个子进程运行可执行文件 **xftpd** 负责与这个客户端之间的文件传输。源程序省略了相关头文件和异常情况处理。

```
1  int main(void)
2  {
3      int admin_sock, data_sock;
4      char str[16];
5      static struct sockaddr_in name;

6      admin_sock = socket(AF_INET, SOCK_STREAM, 0);

7      name.sin_family = AF_INET;
8      name.sin_addr.s_addr = INADDR_ANY;
9      name.sin_port = htons(12345);
10     bind(admin_sock, &name, sizeof(name));

11     listen(admin_sock, 5);

12     signal(SIGCLD, SIG_IGN);
```

```
13    for (;;) {

14        data_sock = accept(admin_sock, 0, 0);

15        if (fork() == 0) {
16            close(admin_sock);
17            sprintf(fd_str, "%d", data_sock);
18            execlp("xftpd", "xftpd", str, 0);
19            printf("run xftpd\n");
20            exit(1);
21        }
22        close(data_sock);
23    }
24 }
```

1. 上述源程序中，有哪些数据（包括变量和常量）被存放在数据段？有哪些数据被存放在堆栈段？
2. 上述服务器进程大部分时间处于阻塞状态，程序中的哪条语句导致进程进入阻塞状态？阻塞状态在什么时机被解除？
3. 如果删除上述程序中的第 12 行，会有什么问题？
4. 如果删除上述程序中的第 22 行，会有什么问题？
5. 客户完成文件传输操作后断开 TCP 连接，第 19 行的语句未执行打印动作，为什么？

答：

1. 第 17,18,19 行都有用引号包起来的字符串数据，这些常量数据放在数据段；第 5 行的变量 name 声明为 static，它也被安排在数据段。其他的三个变量 admin_sock，data_sock，str，都是在函数内部定义的 auto 型变量，在堆栈段分配存储空间。第 7,8，9 行，11 行等等的 int 型常数以及其他行常数 0，存放在指令段，属于“立即寻址型”指令操作数。
2. 第 14 条语句的 accept 调用导致进程进入阻塞状态。当有新的 TCP 连接到来时，从阻塞状态中解除。
3. 第 12 行宣称忽略掉 SIGCLD 信号，当子进程中止时，将自动丢弃子进程的僵尸。如果缺少这条语句，随着结束了服务的子进程产生的僵尸越来越多，会导致操作系统内核中的进程槽耗光，无法创建新进程。
4. 第 22 行的 close 有两个重要功能。第一，父进程将不再使用的文件描述符 close，否则，父进程打开的文件描述符越来越多，当超过了单进程的最大打开文件数之后，以后再来 TCP 连接就无法成功 accept。第二，只有父进程 close 之后并且子进程也 close 时，客户端才能得到 TCP 被关闭的信息，否则，即使子进程 xftpd 崩溃，客户端 TCP 也得不到连接断开的消息。
5. 只要第 18 行的 exec 调用执行成功，子进程全部代码段被毁掉，被替换成文件 xftpd 中的代码段，自然第 19 行的程序也被毁掉，所以，只要第 18 行 exec 执行成功，第 19 行的 printf 就不会有机会执行。这个 printf 有机会执行的前提条件是 exec 执行失败。

五．编程题（共 10 分）

1. (4 分)编写一个完整的 C 语言程序，打印所有的命令行参数和选项。

```
int main(int argc, char **argv)
{
    int i;
    for (i = 0; i < argc; i++)
        printf("%d: [%s]\n", i, argv[i]);
}
```

2. (6 分)使用 Linux 与进程管理有关的系统调用，编写 C 语言程序，使用与下列 shell 命令相同的机制完成同样的功能。

ps -ef > proc.list; grep root < proc.list

为简化编程，假设系统调用执行均能成功，不考虑失败的情况。

提示：相关的系统调用如下

- 创建新进程用 `fork()`
- exec 系统调用可以用 `execlp(file，argv0, argv1, …, 0);`
- 等待子进程结束 `wait(&sv);` 通过整数 sv 获得子进程的返回码
- 创建匿名管道 `pipe(int fd[2]);` 其中 `fd[0]`为读端，`fd[1]`为写端
- dup2 系统调用用法为 `dup2(srcfd, dstfd);` srcfd 覆盖掉 dstfd
- 打开文件读 `fd = open(filename, O_RDONLY);`
- 打开文件写 `fd=open(filename, O_CREAT | O_WRONLY, 0666);`

答：

```
main()
{
    int fd, sv;
    if (fork() == 0) {
        fd = open("proc.list", O_CREAT | O_WRONLY, 0666);
        dup2(fd, 1);
        close(fd);
        execlp("ps", "ps", "-ef", 0);
    } else {
        wait(&sv);
        fd = open("proc.list", O_RDONLY);
        dup2(fd, 0);
        close(fd);
        execlp("grep", "grep", "root", 0);
    }
}
```