**《计算机操作系统》实验报告1**

实验题目：Linux操作系统基本命令

姓名：高远 学号：19123348 实验日期：2021.09.09

**实验环境：**

Vmware kali GNU/Linux

**实验目的：**

1. 了解Linux运行系统环境，熟悉交互式分时系统、多用户环境的运行机制。
2. 联系Linux系统命令接口的使用，学会Linux基本命令、后台命令、管道命令等命令的操作要点。

**实验内容：**

通过终端或虚拟终端，在基于字符的交互界面中进行Shell的基本命令的操作。

**操作过程：**

|  |
| --- |
| ① 执行pwd查看当前目录。  ② 用who am i看看当前用户信息。  ③ 通过who看看有谁在系统中。  ④ 用vmstat显示系统状态。 |
| ① 执行cat > mytext.txt通过键盘输入一些信息，用ctrl+c结束，建立文件mytext.txt  ② 执行cat mytext.txt 显示文件内容。  ③ 执行ln mytext.txt mytext2.dat cat mytext2.dat  ④ 执行ls –l mytext?.\* 显示文件目录，注意i节点号，链接计数。 |
| ① 执行 ls –l看看当前目录的内容，请特别注意文件类型、文件的存取控制权限、i节点号、文件属主、文件属组、文件大小、建立日期等信息。  ② 执行cd /lib  ls –l|more  看看/lib目录的内容，这里都是系统函数。再看看/etc，这里都是系统配置用的数据  文件；/bin中是可执行程序；/home下包括了每个用户主目录。 |
| ① 执行chmod 751 mytext.txt  ls –l mytext.txt  ② 执行chown stud090 mytext.txt |
| ① 执行ps –ef 根据本《实验指导》第二部分介绍的进程管理命令选项，查看当前系统中各个进程的信息。特别注意进程号、父进程号、属主等内容。  ② 执行本《实验指导》第二部分介绍的wait和sleep命令。 |

**结果：**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**题目：**

|  |
| --- |
| 思考题1：  你的用户名、用户标识、组名、组标识是什么？当前你处在系统的哪个位置  中？现在有哪些用户和你一块儿共享系统？  答：通过id命令可知，用户名：gy；用户标识：1000、组名：gy、组标识：1000；  通过pwd命令可知当前处于 /home/gy 目录下；  通过who命令可知当前只有用户gy在系统中。 |
| 思考题2：  文件链接是什么意思？有什么作用？  答：文件链接是用多个（新）文件名与一个文件实体建立链接，即一个文件起了多个名字或多个文件名使用同一个i结点。 |
| 思考题3：  Linux文件类型有哪几种？文件的存取控制模式如何描述？  答：Linux文件类型有普通文件（~）、目录文件（d）、块设备文件（b）、字符字符设备特别文件（c）、命名管道文件（p）等；“存取控制模式”指对不同用户分配不同的操作权。 |
| 思考题4：  执行了上述操作(chmod和chown)后，若想再修该文件，看能不能执行。为什么？  答：不可以，因为执行完后命令后改变了文件的所有者和权限，同组的人可以读和执行，其他人只能执行，不能修改。 |
| 思考题5：  系统如何管理系统中的多个进程？进程的家族关系是怎样体现的？有什么用？  答：共享内存允许多个进程共享一块内存来交换信息，而通过信号量来控制多个进程对共享资源的访问。每个进程都有父进程，而所有的进程以init进程为根，由此可以从容易一个进程查找到其他进程或遍历整个家族进程树。 |
| 讨论题1：  Linux系统命令很多，在手头资料不全时，如何查看命令格式？  答：可以通过 man [命令] 来显示联机手册或 [命令] --help 来显示联机帮助。 |
| 讨论题2：  Linux系统用什么方式管理多个用户操作？如何管理用户文件，隔离用户空间？用命令及结果举例说明。  答：用户登录后，Linux 将用户置于登录用户的主目录中(通常在/home 中的)，启动  一个 Shell 进程与用户交互。通过存取控制模式来对不同用户分配不同的操作权。Linux 文件系统将用户分成三类，即文件主、同组人、其他人。每种人可以行使的操作有三种，即读（r）、 写（w）、执行（x）。 |
| 讨论题3：  用什么方式查看你的进程的管理参数？这些参数怎样体现父子关系？当结束一个父进程后其子进程如何处理？用命令及结果举例说明。  答：使用ps命令来查看。 |
| 讨论题4：  Linux 系统“文件”的含义是什么？它的文件有几种类型？如何标识的？  答：其含义是文件类型有5种：普通文件（~）、目录文件（d）、块设备文件（b）、字符字符设备特别文件（c）、命名管道文件（p）。 |
| 讨论题5：  Linux 系统的可执行命令主要放在什么地方？找出你的计算机中所有存放系统的可执行命令的目录位置。  答：/bin目录及/user/bin目录。 |
| 讨论题6：  Linux系统的设备是如何管理的？在什么地方可以找到描述设备的信息？  答：通过设备识别与设备挂载来管理。通过fdisk -l来查看磁盘分区情况，使用mount命令查看挂载信息。 |
| 讨论题7：  画出Linux根文件系统的框架结构。描述各目录的主要作用。你的用户主目录在哪里？  答：框架结构：  /bin 存放供所有用户使用的完成基本维护任务的命令。  /ets 存放系统和应用软件的配置文件。  /home 是普通用户的家目录。  /lib 存放系统最基本的共享链接库和内核模块。  /root 存放root用户的家目录。  /sbin 存放超级用户使用的可执行文件，里面多是系统管理命令。  /tmp 存放存储临时文件。  我的用户主目录在 /home/gy 中。 |
| 讨论题8：  Linux系统的Shell是什么？请查找这方面的资料，说明不同版本的Shell的特点。  答：Shell程序是利用文本编辑程序建立的一系列Linux命令和实用程序序列的文本文件。它可以像 Linux 的任何命令一样执行。在执行 Shell 程序时，Linux 一个接一个地解释并执行每个命令。Shell 是一种很成熟的程序设计语言，跟其他任何语言一样，有自己的语法、变量和控制语句。以下介绍的是 Linux 的 bash 的基本语法。   1. Ash，只包含24个内部命令，是linux中占用资源最少的shell。 2. Bash，是Linux默认的shell，可以使用上下方向键查阅和修改命令、查找自动匹配以某个字符串开头的命令以及使用help命令。 3. Ksh，即korn shell，有42条内部命令。 4. Esh，也即tcsh，是Linux比较大的内核，有52条内部命令，指向/bin/tcsh。 5. Zch，是Linux最大的内核，共有84条内部命令。 |
| 讨论题9：  下面每一项说明的是哪类文件。  答：   1. -rwxrw-r— 是普通文件。 2. /bin 是目录文件。 3. ttyx3 是命名管道文件。 4. brw-rw-rw- 是块文件。 5. /etc/passwd 是普通文件。 6. crw-rw-rw 是字符设备文件。 7. /usr/lib 是目录文件。 8. Linux 是目录文件。 |

**体会：**

通过本次实验，我对Linux操作系统及其运行环境有了初步的认识，也熟悉了Linux操作系统的交互式分时系统、多用户环境的运行机制，同时熟悉了Linux操作系统命令接口的使用，学会Linux基本命令、后台命令、管道命令等命令的操作要点。这对于今后深入学习Linux系统打下了基础，思考与讨论题也启发了我对Linux的了解。

**附录：（源程序）**

本次实验无源程序。

**《计算机操作系统》实验报告2**

实验题目：用户界面与Shell命令

姓名：高远 学号：19123348 实验日期：2021.09.16

**实验环境：**

Vmware kali GNU/Linux

**实验目的：**

（1）掌握图形化用户界面和字符界面下使用 Shell 命令的方法。

（2）掌握 ls、cd 等 Shell 命令的功能。

（3）掌握重定向、管道、通配符、历史记录等的使用方法。

（4）掌握手工启动图形化用户界面的设置方法。

**实验内容：**

图形化用户界面（GNOME 和 KDE）下用户操作非常简单而直观，但是到目前为止图形化用户界面还不能完成所有的操作任务。

字符界面占用资源少，启动迅速，对于有经验的管理员而言，字符界面下使用 Shell 命令更为直接高效。

Shell 命令是 Linux 操作系统的灵魂，灵活运用 Shell 命令可完成操作系统所有的工作。并且类 UNIX 的操作系统在 Shell 命令方面具有高度相似性。熟悉掌握 Shell 命令，不仅有助于掌握 RHEL Server 5，而且几乎有助于掌握各发行版本的 Linux，甚至 UNIX。

RHEL Server 5 中不仅可在字符界面下使用 Shell 命令，还可以借助于桌面环境下的终端工具使用 Shell 命令。桌面的终端工具中使用 Shell 命令时可显示中文，而字符界面下显示英文。

**操作过程：**

|  |
| --- |
| 1. **图形化用户界面下的Shell命令操作** |
| 【操作要求1】显示系统时间，并将系统时间修改为 2011 年 9 月 17 日零点。  【操作步骤1】  （1） 启动计算机，以超级用户身份登录图形化用户界面。  （2） 依次单击顶部面板的「应用程序」菜单=>「附件」=>「终端」，打开桌面环境下的终端工具。  （3） 输入命令“date”，显示系统的当前日期和时间。  （4） 输入命令“date 091700002011”，屏幕显示新修改的系统时间。在桌面环境的终端中执行时显示中文提示信息，如图 1-1 所示。  【操作要求2】切换为普通用户，查看 2011 年 9 月 17 日是星期几。  【操作步骤2】  （1） 前一操作是以超级用户身份进行的，但通常情况下只有在必须使用超级用户权限的时候，才以超级用户身份操作。为提高操作安全性，输入“su - gy”命令切换为普通用户 gy。  （2） 输入命令“cal 2011”，屏幕上显示出 2011 年的日历，由此可知 2011 年 9 月 17 日是星期日，参见图 1-2。  【操作要求3】查看 ls 命令的-s 选项的帮助信息。  【操作步骤3】  （1） 输入 “man ls” 命令，屏幕显示出手册页中 ls 命令相关帮助信息的第一页。（2） 使用 PgDn 键、PgUp 键以及上、下方向键找到-s 选项的说明信息。  （3） 由此可知，ls 命令的-s 选项等同于--size 选项，以文件块为单位显示文件和目录的大小。  （4） 在屏幕上的“：”后输入“q”，退 出 ls 命令的手册页帮助信息。  【操作要求4】查看/etc 目录下所有文件和子目录的详细信息  【操作步骤4】  （1） 输入命令“cd /etc”，切换到/etc 目录。  （2） 输入命令“ls -al”，显示/etc 目录下所有文件和子目录的详细信息。 |
| 1. **字符界面下的Shell命令操作** |
| 【操作要求1】查看当前目录。  【操作步骤1】  （1） 启动计算机后默认会启动图形化用户界面，按下 CTRL+ALT+F1 键切换到第 1 个虚拟终端。  （2） 输入一个普通用户的用户名（gy）和口令，登录系统。字符界面下输入口令时，屏幕上不会出现类似“\*”的信息，提高了口令的安全性。  （3） 输入命令“pwd”，显示当前目录。  【操作要求2】用 cat 命令在用户主目录下创建一名为 f1 的文本文件，内容为：  Linux is useful for us all.  You can never imgaine how great it is.  【操作步骤2】  （1） 输入命令“cat >f1”，屏幕上输入点光标闪烁，依次输入上述内容。  使用 cat 命令进行输入时，不能使用左右上下方向键，只能用退格键（Backspace）来删除光标前一位置的字符。并且一旦按下回车键，该行输入的字符就不可修改。  （2） 上述内容输入后，按 Enter 键，让光标处于输入内容的下一行，按 CTRL+D 键结束输入。  （3） 要查看文件是否生成，输入命令“ls”即可。  （4） 输入命令“cat f1”，查 看 f1 文件的内容。  【操作要求3】向 f1 文件增加以下内容：Why not have a try?  【操作步骤3】  （1） 输入命令“cat >>f1”，屏幕上输入点光标闪烁。  （2） 输入上述内容后，按 Enter 键，让光标处于输入内容的下一行，按 CTRL+D 键结束输入。  （3） 输入“cat f1”命令，查看 f1 文件的内容，会发现 f1 文件增加了一行。  【操作要求4】统计 f1 文件的行数，单词数和字符数，并将统计结果存放在 countf1 文件。  【操作步骤4】  （1） 输入命令“wc <f1> countf1”，屏幕上不显示任何信息。  （2） 输入命令“cat countf1”，查看 countf1 文件的内容，其内容是 f1 文件的行数、单词数和字符数信息，即 f1 文件共有 3 行，19 个词和 87 个字符。  【操作要求5】将 f1 和 countf1 文件的合并为 f 文件  【操作步骤5】  （1） 输入命令“cat f1 countf1 >f”，将两个文件合并为一个文件。  （2） 输入命令“cat f”，查看 f 文件的内容。  【操作要求6】分页显示/etc 目录中所有文件和子目录的信息  【操作步骤6】  （1） 输入命令“ls /etc|more”，屏幕显示出“ls /etc”命令输出结果的第一页，屏幕的最后一行上还出现“--More--”字样，按空格键可查看下一页信息，按 Enter 键可查看下一行信息。  （2） 浏览过程中按“q”键，可结束分页显示。  【操作要求7】仅显示/etc 目录中前 5 个文件和子目录。  【操作步骤7】  （1）输入命令“ls /etc |head -n 5”, 屏幕显示出“ls /etc”命令输出结果的前面 5 行。  【操作要求8】清除屏幕内容  【操作步骤8】  （1）输入命令“clear”，则屏幕内容完全被清除，命令提示符定位在屏幕左上角。 |
| 1. **通配符的使用** |
| 【操作要求1】显示/bin/目录中所有以 c 为首字母的文件和目录  【操作步骤1】  （1）输入命令“ls /bin/c\*”，屏幕将显示/bin 目录中以 c 开头的所有文件和目录。  【操作要求2】显示/bin/目录中所有以c为首字母,文件名只有3个字符的文件和目录。  【操作步骤2】  （1） 按向上方向键，Shell 命令提示符后出现上一步操作时输入的命令“ls /bin/c\*”。  （2） 将其修改为“ls /bin/c??”，按 下 Enter 键，屏幕显示/bin 目录中以 c 为首字母,文件名只有 3 个字符的文件和目录。  【操作要求3】显示/bin 目录中所有的首字母为 c 或 s 或 h 的文件和目录。  【操作步骤3】  （1）输入命令“ls /bin/[csh]\*”，屏幕显示/bin 目录中首字母为 c 或 s 或 h 的文件和目录。  【操作要求4】显示/bin/目录中所有的首字母是 v、w、x、y、z 的文件和目录。  【操作步骤4】  （1）输入命令“ls /bin/[v-z]\*”，屏幕显示/bin 目录中首字母是 v~z 的文件和目录。  【操作要求5】重复上一步操作。  【操作步骤5】输入命令“！！”，自动执行上一步操作中使用过的“ls /bin/[v-z]\*”命令。  【操作要求6】查看刚执行过的 5 个命令。  【操作步骤6】  （1）输入命令“history 5”，显示最近执行过的 5 个命令。  **4. 设置手工启动图形化界面** |
| 【操作要求1】设置开机不启动图形化用户界面。  【操作步骤1】  （1） 按下 ALT+F7 键，切换回到图形化用户界面，以超级用户身份登录。  （2） 依次单击「应用程序」菜单=>「附件」=>「文本编辑器」，打开 gedit 文本编辑器。  （3） 单击工具栏上的「打开」按钮，从「打开文件...」对话框中选择/etc 目录中的 inittab文件。  （4） 将文件中的“id：5：initdefault：”所在行的“5”修改为“3”。  【操作要求2】手工启动图形化用户界面。  【操作步骤2】  （1） 计算机重启后只有字符界面可用，输入用户名和相应的口令后，登录 Linux 系统。  （2） 输入命令“startx”，启动图形化用户界面。  （3） 单击「系统」菜单=>「注销」，弹出对话框，单击「注销」按钮，返回到字符界面。 |

**结果：**

|  |
| --- |
| 1. **图形化用户界面下的Shell命令操作** |
| **图1-1 设置时间**    **图1-2 查看日历**    **图1-3-1 查看帮助信息**    **图1-3-2 查看帮助信息**    **图1-4** |
| 1. **字符界面下的Shell命令操作** |
| **图2-1 查看当前目录**    **图2-2 创建文本文件**    **图2-3 增加文件内容**    **图2-4 统计文件结果**    **图2-5 文件合并**    **图2-6 /etc目录信息**    **图2-6 显示/etc 目录中前 5 个文件和子目录** |
| 1. **通配符的使用** |
| **图3-1显示/bin/目录中所有以 c 为首字母的文件和目录**    **图3-2显示/bin/目录中所有以 c 为首字母,文件名只有 3 个字符的文件和目录**    **图3-3显示/bin 目录中所有的首字母为 c 或 s 或 h 的文件和目录**    **图3-4 显示/bin/目录中所有的首字母是 v-z 的文件和目录**    **图3-5 执行上一步操作**    **图3-6 显示最近执行过的 5 个命令** |

**体会：**

通过本次实验，我对用户界面与Shell命令有了进一步的认识，掌握了图形化用户界面和字符界面下使用Shell命令的方法，掌握了ls、cd等Shell命令的功能，掌握了重定向、管道、通配符、历史记录等的使用方法，掌握了手工启动图形化用户界面的设置方法。同时也对Linux系统有了更深的理解。

**附录：（源程序）**

本次实验无源程序。

**《计算机操作系统》实验报告3**

实验题目：进程管理及进程通信

姓名：高远 学号：19123348 实验日期：2021.09.23

**实验环境：**

Vmware kali GNU/Linux

**实验目的：**

1. 利用Linux提供的系统调用设计程序，加深对进程概念的理解。
2. 体会系统进程调度的方法和效果。
3. 了解进程之间的通信方式以及各种通信方式的使用。

**实验内容：**

用vi 编写使用系统调用的C 语言程序。

**操作过程：**

|  |
| --- |
| 1. 编写程序。显示进程的有关标识（进程标识、组标识、用户标识等）。经过5 秒钟后，执行另一个程序，最后按用户指示（如：Y/N）结束操作。 |
| 1. 参考例程1，编写程序。实现父进程创建一个子进程。体会子进程与父进程分别获得 不同返回值，进而执行不同的程序段的方法. |
| 1. 参考例程2，编写程序。父进程通过循环语句创建若干子进程。探讨进程的家族树以 及子进程继承父进程的资源的关系。 |
| 1. 参考例程3，编写程序。使用fork()和exec()等系统调用创建三个子进程。子进程分别启动不同程序，并结束。反复执行该程序，观察运行结果，结束的先后，看是否有不同次序。 |
| 1. 参考例程4，编写程序。验证子进程继承父进程的程序、数据等资源。如用父、子进程修改公共变量和私有变量的处理结果；父、子进程的程序区和数据区的位置。 |
| 1. 参照《实验指导》第五部分中“管道操作的系统调用”。复习管道通信概念，参考例程5，编写一个程序。父进程创建两个子进程，父子进程之间利用管道进行通信。要求能显示父进程、子进程各自的信息，体现通信效果。 |
| 1. 编程验证：实现父子进程通过管道进行通信。进一步编程，验证子进程结束，由父进程执行撤消进程的操作。测试父进程先于子进程结束时，系统如何处理“孤儿进程”的。 |
| 1. 编写两个程序一个是服务者程序，一个是客户程序。执行两个进程之间通过消息机制通信。消息标识MSGKEY 可用常量定义，以便双方都可以利用。客户将自己的进程标识（pid）通过消息机制发送给服务者进程。服务者进程收到消息后，将自己的进程号和父进程号发送给客户，然后返回。客户收到后显示服务者的pid 和ppid，结束。以下例程6 基本实现以上功能。这部分内容涉及《实验指导》第五部分中“IPC系统调用”。先熟悉一下，再调试程序。 |
| 1. 这部分内容涉及《实验指导》第五部分中“有关信号处理的系统调用”。编程实现软中断信号通信。父进程设定软中断信号处理程序，向子进程发软中断信号。子进程收到信号后执行相应处理程序。 |
| 1. 怎么样，试一下吗？用信号量机制编写一个解决生产者—消费者问题的程序，这可是受益匪浅的事。本《实验指导》第五部分有关进程通信的系统调用中介绍了信号量机制的使用。 |

**结果：**

|  |
| --- |
| **图1 例程0：显示进程的有关标识** |
| **图1 例程1：利用fork()创建子进程**  思考题：  子进程是如何产生的？又是如何结束的？子进程被创建后它的运行环境是怎样建立的？  答：  通过fork()函数来创建子进程，调用格式为n=fork()，该系统调用返回值可以是 n=0、n>0、n=-1。如果执行成功，fork()返回给父进程的是子进程的标识符（pid）；返还  给子进程的是0；创建失败返回–1。n=0 表示获得此参数的进程将执行子进程的程序段（由系统调度确定）。n>0 表示获得此参数的进程将执行父进程的程序段（由系统调度确定），该参数即为子进程的标识符（pid）。n<0 表示创建失败。通过exit()函数来结束进程。在 Linux 中，有三种正常结束进程的方法，两种异常终止的方法。正常结束为：在main函数中调用return（相当于调用exit）、调用exit以及调用\_exit函数；异常终止为调用abort，产生一个SIGABRT信号或进程收到特定信号。这个信号可以是进程自己产生，也可以来自其它进程和内核。调用格式为void exit(status)、void \_exit(int status)，其中status为退出的状态，0为正常，非0为错误。创建时先从 PCB 集合 中申请一个空闲的 PCB，再为新进程分配内存等资源，并根据父进程提供的参数和分配到 的资源情况来对 PCB 进行初始化，最后将新进程插入就绪队列。 |
| **图2 例程2：循环调用fork()创建多个子进程**  思考题：   1. 画出进程的家族树。子进程的运行环境是怎样建立的？反复运行此程序看会有什么情况？解释一下。   答：进程的家族树如下图图3所示。子进程的运行环境由创建它的父进程建立的，反复运行此程序会发现pid一直在变化而父亲的pid一直未改变，因为父进程在一直创建新的子进程，所以得到子进程的pid在变化。    **图3 进程的家族树**  ② 修改程序，使运行结果呈单分支结构，即每个父进程只产生一个子进程。画出进程树，解释该程序。  答：若该进程为子进程，则返回其pid；若该进程为父进程，则创建子进程并退出。    **图4 进程的单分支家族树实验结果**    **图5进程的单分支家族树** |
| **图6 例程3:创建子进程并用execlp()系统调用执行程序的实验**  思考题：  子进程运行其它程序后，进程运行环境怎样变化的？反复运行此程序看会有什么情况？解释一下。  答：一个进程调用 exec 函数执行另一个程序后，这个进程就完全被新程序代替。由于并没有产生新进程所以进程标识号不改变，除此之外旧进程的其它信息，代码段、数据段、栈段等均被新程序的信息所代替。新程序从自己的 main()函数开始运行。反复运行此程序发现，每次的运行结果并不相同，是因为每个子进程运行其他程序时会随着其他程序的结束而结束，因此结束的次序不固定。 |
| **图7 例程4：观察父、子进程对变量处理的影响**  思考题：  子进程被创建后，对父进程的运行环境有影响吗？解释一下。  答：没有影响，因为父进程的程序段与数据段不受子进程的影响。 |
| **图8例程5：管道通信的实验**  思考题：  ① 什么是管道？进程如何利用它进行通信的？解释一下实现方法。  ② 修改睡眠时机、睡眠长度，看看会有什么变化。请解释。   1. 加锁、解锁起什么作用？不用它行吗？   答：   1. 管道通信是进程的一种通信方式。Linux 中两个进程可以通过管道来传递消息。管道在逻辑上被看作管道文件，在物理上则由文件系统的高速缓冲区构成。管道用 pipe()系统调用建立。发送信息的进程用 write()把信息写入管道，接收进程用 read()从管道中读取信息。 2. 修改睡眠世纪和睡眠长度会使得进程的唤醒时间改变，因为睡眠时机决定了进程睡眠的时间点，而睡眠长度决定了进程被唤醒的时间点。 3. 加锁与解锁可以解决共享临界资源的问题。若不使用，则会导致读入错误的数据。 |
| **图9 “孤儿进程”的实验**  思考题：  对此作何感想，自己动手试一试？解释一下你的实现方法。  答：在父进程创建子进程后使其睡眠五秒，而父进程睡眠一秒，则能保证父进程先于子进程结束。 |
| **图10 例程6：消息通信的实验**  思考：  想一下服务者程序和客户程序的通信还有什么方法可以实现？解释一下你的设想，有兴趣试一试吗。  答：还能通过信号量机制。用户进程可以通过使用操作系统提供的一对原语来对信号量进行操作，从而可以实现进程互斥、进程同步。用一个整型数的变量作为信号量，用来表示系统中某种资源的数量，由此控制多个进程访问共享的资源。 |
| **图11 例程7：软中断信号实验**  思考题：  思考：这就是软中断信号处理，有点儿明白了吧？讨论一下它与硬中断有什么区  别？看来还挺管用，好好利用它。  答：   1. 软中断是执行中断指令产生的，而硬中断是由外设引发的。 2. 硬中断的中断号是由中断控制器提供的，软中断的中断号由指令直接指出，无需使用中断控制器。 3. 硬中断是可屏蔽的，软中断不可屏蔽。 4. 硬中断处理程序要确保它能快速地完成任务，这样程序执行时才不会等待较长时间，称为上半部。 5. 软中断处理硬中断未完成的工作，是一种推后执行的机制，属于下半部。 |
| **图12 生产者—消费者问题实验** |
| 讨论题：  讨论Linux 系统进程运行的机制和特点，系统通过什么来管理进程？  答：在Linux系统中，对于每个资源和每个进程都设置了一个数据结构，用于表征其实体， 我们称之为资源信息表或进程信息表，其中包含了资源或进程的标识、描述、状态等信息以及一批指针。通过这些指针，可以将同类资源或进程的信息表，或者同一进程所占用的 资源信息表分类链接成不同的队列，便于操作系统进行查找。为了便于系统描述和管理进程的运行，在OS的核心为每个进程专门定义了一个数据结构—进程控制块 PCB(Process Control Block)。PCB作为进程实体的一部分，记录了操作系统所需的，用于描述进程的当前情况以及管理进程运行的全部信息，是操作系统中最重要的记录型数据结构。 |
| 讨论题：  C 语言中是如何使用Linux 提供的功能的？用程序及运行结果举例说明。  答：通过在文件中添加头文件来调用进程管理的库函数。例如，若将jincheng0.c中的头文件部分注释掉，则程序会报错。    **图13 程序报错** |
| 讨论题：  什么是进程？如何产生的？举例说明。  答：进程是进程实体的运行过程，是系统进行资源分配和调度的一个独立单位。在系统中每当出现了创建新进程的请求后，OS便调用进程创建原语Creat按下述步骤创建一个新进程：申请空白 PCB、为新进程分配其运行所需的资源、初始化进程控制块(PCB)以及如果进程就绪队列能够接纳新进程，便将新进程插入就绪队列。；例如，用户登录、进程调度、提供服务与创建请求都会创建进程。 |
| 讨论题：  进程控制如何实现的？举例说明。  答：通过OS内核的原语来实现的。例如进程通过调用阻塞原语block将自己阻塞或是通过wakeup原语被唤醒。 |
| 讨论题：  进程通信方式各有什么特点？用程序及运行结果举例说明。  答：   1. Socket通信方式：实现简单；自带同步机制；随进程持续。 2. 共享内存方式：速度快、效率高；自身不具备同步机制；随内核持续，生命力更强。 3. 管道通信：只支持单向数据流；自身具备同步机制；随进程持续。 4. 有名管道方式：支持无亲缘关系的进程通信；自身具备同步机制；随进程持续。 5. 信号方式：可携带的信息极少；不具备同步机制。 6. 消息队列方式：支持无亲缘关系的进程通信；提供有格式的字节流；消息具有类型或优先级；具备同步机制；随内核持续。   例子为jincheng5.c，父子进程通过管道进程通信。 |
| 讨论题：  管道通信如何实现？该通信方式可以用在何处？  答：管道在逻辑上被看作管道文件，在物理上则由文件系统的高速缓冲区构成。管道用 pipe()系统调用建立。发送信息的进程用 write()把信息写入管道，接收进程用 read()从管道中读取信息。该通信方式可以用在数据传输、资源共享以及事件通知。 |
| 讨论题：  什么是软中断？软中断信号通信如何实现？  答：软中断是利用[硬件中断](https://baike.baidu.com/item/%E7%A1%AC%E4%BB%B6%E4%B8%AD%E6%96%AD)的概念，用软件方式进行模拟，实现宏观上的异步执行效果。通过kill与signal来实现。 |

**体会：**

本次实验中，我接触到了大量关于Linux系统进程的知识，利用Linux提供的系统调用设计程序，加深对进程概念的理解，体会了系统进程调度的方法和效果。并了解进程之间的通信方式以及各种通信方式的使用。同时也通过本实验，对OS中进程的创建、控制与父进程的关系等等概念有了更深刻的认识。

**附录：（源程序）**

|  |
| --- |
| **/\* jincheng0.c \*/**  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int main()  {  int i,child\_pid,pid,status;  char yes\_or\_no;  printf("进程标识是 %d, 组标识是 %d，用户标识是%d\n",getpid(),getgid(),getuid());  sleep(5);  child\_pid=fork(); /\*创建子进程3\*/  if(child\_pid==0)  {  execlp("./jincheng1.o","./jincheng1.o",(char \*)0); /\*子进程 启动其它程序\*/  perror("exec3 error.\n ");  exit(3);  }  while((pid=wait(&status))!=-1) /\*等待子进程结束\*/  {  if(child\_pid==pid) /\*若子进程 结束\*/  printf("child process terminated with status %d\n",(status>>8));  }  puts("Child processes terminated.");  printf("是否结束程序？");  scanf("%c", &yes\_or\_no);  if (yes\_or\_no == 'Y')  {  puts("Parent processes terminated.");  exit(0);  }  else if (yes\_or\_no == 'N')  {  puts("Parent processes is not terminated.");  }  } |
| **/\* jincheng1.c \*/**  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  /\*例程1：利用fork()创建子进程,用fork()系统调用创建子进程的例子\*/  int main()  {  int i;  if (fork()) {/\*父进程执行的程序段\*/  i = wait(0); /\* 等待子进程结束\*/  printf("It is parent process.\n");  printf("The child process,ID number %d, is finished.\n", i);  }  else  {  printf("It is child process.\n");  sleep(2);  /\*子进程执行的程序段\*/  exit(0); /\*向父进程发出结束信号\*/  }  exit(0);  } |
| **/\* jincheng2.c \*/**  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  /\*例程2：循环调用fork()创建多个子进程。建立进程树\*/  int main()  {  int i,j;  printf("My pid is %d, my father’s pid is %d\n",getpid(),getppid());  for(i=0; i<3; i++)  {  if(fork()==0)  {  printf("%d pid=%d ppid=%d\n",i,getpid(),getppid());  }  else  {  j=wait(0);  printf("%d:The child %d is finished.\n",getpid(),j);  }  }  exit(0);  } |
| **/\* jincheng2\_modi.c \*/**  /\*例程2\_修改版：运行结果呈单分支结构。  建立进程树\*/  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int main()  {  int i;  printf("My pid is %d, my father’s pid is %d\n",getpid(),getppid());  for(i=0; i<3; i++)  {  if(fork()!=0)  {  printf("%d pid=%d ppid=%d\n",i,getpid(),getppid());  }  else  {  break;  }  }  exit(0);  } |
| **/\* jincheng3.c \*/**  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  /\*例程3:创建子进程并用execlp()系统调用执行程序的实验  创建子进程，子进程启动其它程序\*/  int main()  {  int child\_pid1,child\_pid2,child\_pid3;  int pid,status;  setbuf(stdout,NULL);  child\_pid1=fork(); /\*创建子进程1\*/  if(child\_pid1==0)  {  execlp("echo","echo","child process 1",(char \*)0); /\*子进程1 启动其它程序\*/  perror("exec1 error.\n ");  exit(1);  }  child\_pid2=fork(); /\*创建子进程2\*/  if(child\_pid2==0)  {  execlp("date","date",(char \*)0); /\*子进程2 启动其它程序\*/  perror("exec2 error.\n ");  exit(2);  }  child\_pid3=fork(); /\*创建子进程3\*/  if(child\_pid3==0)  {  execlp("ls","ls",(char \*)0); /\*子进程3 启动其它程序\*/  perror("exec3 error.\n ");  exit(3);  }  puts("Parent process is waiting for chile process return!");  while((pid=wait(&status))!=-1) /\*等待子进程结束\*/  {  if(child\_pid1==pid) /\*若子进程1 结束\*/  printf("child process 1 terminated with status %d\n",(status>>8));  else  {  if(child\_pid2==pid) /\*若子进程2 结束\*/  printf("child process 2 terminated with status %d\n",(status>>8));  else  {  if(child\_pid3==pid) /\*若子进程3 结束\*/  printf("child process 3 terminated with status %d\n" ,(status>>8));  }  }  }  puts("All child processes terminated.");  puts("Parent process terminated.");  exit(0);  } |
| **/\* jincheng4.c \*/**  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  /\*例程4：观察父、子进程对变量处理的影响  创建子进程的实验。子进程继承父进程的资源，修改了公共变量globa 和私有变  量vari。观察变化情况。\*/  int globa=4;  int main()  {  pid\_t pid;  int vari=5;  printf("before fork.\n");  if ((pid=fork())<0) /\*创建失败处理\*/  {  printf("fork error.\n");  exit(0);  }  else  if(pid==0)  { /\*子进程执行\*/  globa++;  vari--;  printf("Child %d changed the vari and globa.\n",getpid());  }  else /\*父进程执行\*/  printf("Parent %d did not changed the vari and globa.\n",getpid());  printf("pid=%d, globa=%d, vari=%d\n",getpid(),globa,vari); /\*都执行\*/  exit(0);  } |
| **/\* jincheng5.c \*/**  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  /\*例程5：管道通信的实验\*/  /\*程序建立一个管道fd\*/  /\*父进程创建两个子进程P1、P2 \*/  /\*子进程P1、P2 分别向管道写入信息\*/  /\*父进程等待子进程结束，并读出管道中的信息\*/  int main()  {  int i,r,j,k,l,p1,p2,fd[2];  char buf[50],s[50];  pipe(fd);/\*建立一个管道fd\*/  while((p1=fork())==-1);/\*创建子进程1\*/  if(p1==0)  {  lockf(fd[1],1,0); /\*子进程1 执行\*/  /\*管道写入端加锁\*/  sprintf(buf,"Child process P1 is sending messages! \n");  printf("Child process P1! \n");  write(fd[1],buf,50);  lockf(fd[1],0,0);  /\*信息写入管道\*/  /\*管道写入端解锁\*/  sleep(5);  j=getpid();  k=getppid();  printf("P1 %d is weakup. My parent process ID is %d.\n",j,k);  exit(0);  }  else  {while((p2=fork())==-1);/\*创建子进程2\*/  if(p2==0)  {/\*子进程2 执行\*/  lockf(fd[1],1,0);/\*管道写入端加锁\*/  sprintf(buf,"Child process P2 is sending messages! \n");  printf("Child process P2! \n");  write(fd[1],buf,50);/\*信息写入管道\*/  lockf(fd[1],0,0);/\*管道写入端解锁\*/  sleep(5);  j=getpid();  k=getppid();  printf("P2 %d is weakup. My parent process ID is %d.\n",j,k);  exit(0);  }  else  {l=getpid();  wait(0);  if(r=read(fd[0],s,50)==-1)  printf("Can't read pipe. \n");  else  printf("Parent %d: %s \n",l,s);  wait(0);  if(r=read(fd[0],s,50)==-1)  printf("Can't read pipe. \n");  else  printf("Parent %d: %s \n",l,s);  exit(0);  }  }  } |
| **/\* jincheng5\_modi.c \*/**  #include<stdio.h>  #include<stdlib.h>  #include<errno.h>  #include<unistd.h>  int main()  {  pid\_t pid;  pid = fork();  if (pid == 0)  {  printf("It is child process.\n");  printf("pid: %d ppid: %d\n",getpid(),getppid());  sleep(5);  printf("pid: %d ppid: %d\n",getpid(),getppid());  printf("child processes terminated.\n");  }  if (pid < 0)  {  perror("fork error.\n");  exit(1);  }  else  {  printf("It is father process.\n");  sleep(1);  printf("father process terminated.\n");  }  return 0;  } |
| **/\* jincheng6\_client.c \*/**  /\*客户程序\*/  /\*The client send a message to server,and receives another message from  server\*/  #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  #include<sys/types.h>  #include<sys/ipc.h>  #include<sys/msg.h>  #define MSGKEY 75  struct msgform /\*定义消息结构\*/  {  long mtype;  char mtext[256];  };  int main()  {  struct msgform msg;  int msgqid,pid,\*pint;  msgqid=msgget(MSGKEY,0777);/\*建立消息队列\*/  pid=getpid();  pint=(int \*)msg.mtext;  \*pint=pid;  msg.mtype=1;/\*定义消息类型\*/  msgsnd(msgqid,&msg,sizeof(int),0);/\*发送消息\*/  msgrcv(msgqid,&msg,256,pid,0);/\*接受从服务者发来的消息\*/  printf("Clint : receive from pid %d\n",\* pint);  exit(0);  } |
| **/\* jincheng6\_server.c \*/**  /\*例程6：消息通信的实验\*/  /\*客户进程向服务器进程发出信号，服务器进程接收作出应答，并再向客户返回消  息。\*/  /\*===============================\*/  /\*服务者程序\*/  /\*The server receives the message from client,and answer a message\*/  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  #include<sys/types.h>  #include<sys/ipc.h>  #include<sys/msg.h>  #define MSGKEY 75  struct msgform  { /\*定义消息结构\*/  long mtype;  char mtext[256];  }msg;  int msgqid;  int main()  {  int i,pid,\* pint;  void cleanup();  for(i=0;i<20;i++) /\*设置软中断信号的处理程序\*/  signal(i,&cleanup);  msgqid=msgget(MSGKEY,0777|IPC\_CREAT); /\*建立消息队列\*/  for(;;) /\*等待接受消息\*/  {  msgrcv(msgqid,&msg,256,1,0); /\* 接受消息\*/  pint=(int \*)msg.mtext;  pid=\*pint;  printf("Server :receive from pid %d\n",pid);/\*显示消息来源\*/  msg.mtype=pid;  \*pint=getpid(); /\*加入自己的进程标识\*/  msgsnd(msgqid,&msg,sizeof(int),0); /\*发送消息\*/  }  exit(0);  }  void cleanup()  {  msgctl(msgqid,IPC\_RMID,0);  exit(0);  } |
| **/\* jincheng7.c \*/**  /\* 例程7：软中断信号实验  父进程向子进程发送18 号软中断信号后等待。子进程收到信号，执行指定的程  序，再将父进程唤醒。\*/  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int main()  {  int i,j,k;  void func();  signal(18,&func); /\*设置18 号信号的处理程序\*/  if(i=fork())/\*创建子进程\*/  {/\*父进程执行\*/  j=kill(i,18);/\*向子进程发送信号\*/  printf("Parent: signal 18 has been sent to child %d,returned %d.\n",i,j);  k=wait(0); /\*父进程被唤醒\*/  printf("After wait %d,Parent %d: finished.\n",k,getpid());  }  else  { /\*子进程执行\*/  sleep(10);  printf("Child %d: A signal from my parent is recived.\n",getpid());  } /\*子进程结束，向父进程发子进程结束信号\*/  //exit(0);  }  void func() /\*处理程序\*/  {  int m;  m=getpid();  printf("I am Process %d: It is signal 18 processing function.\n",m);  } |
| **/\* jinchen8.c \*/**  #include<stdio.h>  #include<stdlib.c>  #include<unistd.h>  #include<pthread.h>  #include<semaphore.h>  #define NUM 5  int queue[NUM];  sem\_t empty\_num, commodity;  void\* producer(void\* arg)  {  int i = 0;  while (1) {  sem\_wait(&empty\_num); // 空格数--， 为0阻塞  queue[i] = 1;  printf("The producer put a commodity in number %d.\n%d %d %d %d %d\n", i, queue[0], queue[1], queue[2], queue[3], queue[4]);  sem\_post(&commodity); // 产品数++  i = (i + 1) % NUM;  sleep(rand() % 3);  }  return NULL;  }  void\* consumer(void\* arg)  {  int i = 0;  while (1) {  sem\_wait(&commodity); // 产品数--，为0的话阻塞  queue[i] = 0;  printf("The customer purchased the commodity in number %d./n%d %d %d %d\n", i, queue[0], queue[1], queue[2], queue[3], queue[4]);  sem\_post(&empty\_num); // 空格数++  i = (i + 1) % NUM;  sleep(rand() % 6);  }  return NULL;  }  int main()  {  pthread\_t pid, cid;  sem\_init(&empty\_num, 0, NUM);  sem\_init(&commodity, 0, 0);  pthread\_create(&pid, NULL, producer, NULL);  pthread\_create(&cid, NULL, consumer, NULL);  pthread\_join(pid, NULL);  pthread\_join(cid, NULL);  sem\_destroy(&empty\_num);  sem\_destroy(&commodity);  return 0;  } |

**《计算机操作系统》实验报告4**

实验题目：Linux进程调度与系统监视

姓名：高远 学号：19123348 实验日期：2021.10.09

**实验环境：**

Vmware kali GNU/Linux

**实验目的：**

（1）熟练掌握手工启动前后台作业的方法。

（2）熟练掌握进程与作业管理的相关 Shell 命令。

（3）掌握 at 调度和 cron 调度的设置方法。

（4）了解进行系统性能监视的基本方法。

**实验内容：**

1. 作业和进程的基本管理。
2. at 进程调度。所谓进程调度就是设定某个指定的作业在固定的时间、或者固定的频率，或者系统空闲时自动执行的操作。根据作业要求执行的条件不同，可选中不同的调度方式。at 调度：在指定的时间执行一次特定的作业。batch 调度：在系统空闲时执行一次特定的作业。cron 调度：每到指定的时间就执行特定的作业，可执行多次。
3. cron 进程调度。cron 调度的内容共有 6 个字段，从左到右依次为分钟、小时、日期、月份、星期和命令。如表格1所示。在设置 cron 调度时，所有的字段都不能为空，字段之间用空格分开，如果不指定字段内容，则使用“\*”符号。使用“-”符号表示一段时间。如果在日期栏中输入“1-5”则表示每个月前 5 天每天都要执行该命令。使用“,”符号表示指定的时间。如果在日期栏中输入“5,15,25”则表示每个月的 5 日、15 日和 25 日都要执行该命令。使用“/”符号表示间隔频率，如果在小时栏中输入“\*/2”，表示 某 2 小时执行一次该命令。

**表格1 crontab 文件的格式**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段 | 分钟 | 小时 | 日期 | 月份 | 星期 | 命令 |
| 取值范围 | 0~59 | 0~23 | 01~31 | 01~02 | 0~6,0为星期天 |  |

1. 系统性能监视。

**操作过程：**

|  |
| --- |
| 1. **作业和进程的基本管理** |
| 【操作要求 1】  先在前台启动 vi 编辑器并打开f4文件，然后挂起，最后在后台启动一个查找 inittab 文件的 find 作业，find的查找结果保存到f5。  【操作步骤1】  （1）以超级用户（root）身份登录到 RHEL Server 5 字符界面。  （2）输入命令“vi f4”，在前台启动 vi 文本编辑器并打开 f4 文件。  （3）按下 Ctrl+Z 组合键，暂时挂起“vi f4”作业，屏幕显示该作业的作业号。  （4）输入命令“find / -name inittab > f5 &”，启动一个后台作业。在显示作业号的同时还显示进程号。  【操作要求 2】  查看当前作业、进程和用户信息，并对作业进行前后台切换。  【操作步骤2】  （1）输入命令“jobs”，查看当前系统中的所有作业。  （2）输入命令“fg 2”，将“find / -name inittab > f5 &”作业切换到前台。屏幕显示出“find / -name inittab > f5”命令，并执行此命令。稍等片刻，作业完成后屏幕再次出现命令提示符。  （3）输入命令“cat f5”，查看“find / -name inittab > f5”命令的执行结果。  （4）再次输入命令“jobs”，可发现当前系统中的只有一个已停止的作业“vi f4”。  （5）输入命令“kill -9 %1”，终止“vi f4”作业。  （6）稍等片刻，输入命令“jobs”，查看到当前没有任何作业。  （7）输入命令“ps –l”，查看进程的相关信息，显示出的信息类似如下信息。  （8）输入命令“who –H”，查看用户信息。 |
| 1. **at 进程调度** |
| 【操作要求1】  设置一个调度，要求在 2008 年 1 月 1 日 0 时，向所有用户发送新年快乐的问候。  【操作步骤1】  （1）超级用户输入命令“at 00:00 01012008”，设置 2008 年 1 月 1 日 0 时执行的 at 调度的内容。  （2）屏幕出现 at 调度的命令提示符“at>”，输入“wall Happy New Year!”，向所有用户发送消息。  （3）光标移动到“at>”提示符的第三行，按下 Ctrl+D 组合键结束输入。根据调度设置的时间，最后显示出作业号和将要运行的时间。  【操作要求2】  设置一个调度，要求 5 分钟后向所有用户发送系统即将重启的消息，并在 2分钟后重新启动计算机。  【操作步骤2】  （1）超级用户输入命令“at now +5 minutes”，设置5分钟后执行的at调度的内容。  （2）屏幕出现 at 调度的命令提示符“at>”，输入“wall please logout; the computer will restart.”，向所有用户发送消息。  （3）在“at>”提示符的第二行输入“shutdown –r +2”，系统2分钟后将重新启动。“shutdown –r +2”命令与“reboot +2”命令效果相同，都是在 2 分钟后重新启动。  （4）光标移动到“at>”提示符的第三行，按下 Ctrl+D 组合键结束输入。最后显示作业号和运行时间。  【操作要求3】  【操作步骤3】  （1） 输入“atq”命令，查看所有的 at 调度，显示出作业号、将在何时运行以及 at 调度的设定者。  （2） 输入“atrm 1”命令删除作业号为 1 的 at 调度，并再次输入“atq”命令查看剩余的所有 at 调度内容。  （3） 5 分钟后系统将自动运行作业号为 2 的 at 调度内容。先向所有用户发送消息，然后再等 2 分钟重新启动。 |
| 1. **cron 进程调度** |
| 【操作要求1】  helen 用户设置 crontab 调度，要求每天上午 8 点 30 份查看系统的进程状态，并将查看结果保存于 ps.log 文件。  【操作步骤1】  （1）以普通用户helen登录，并输入命令“crontab –e”，新建一个crontab配置文件。  （2）屏幕出现vi编辑器，按下“i”，进入输入模式，输入“30 8 \* \* \* ps >ps.log ”。  （3）按下 Esc 键退出 vi 的文本输入模式，并按下“：”键切换到最后行模式，输入“wq”，保存并退出编辑器，显示“crontab: installing new crontab”信息。  （4）输入命令“crontab -l”，查 看 helen 用户的 cron 调度内容。  （5） 为立即查看到 crontab 调度的结果，切换为超级用户，并适当修改系统时间，如修改为 8 点 29 分。最后退回到 helen 用户。  （6） 等待 1 分钟后，查看 ps.log 文件的内容，如果显示出正确的内容，那么说明 crontab调度设置成功。  【操作要求2】  helen用户添加设置crontab调度，要求每三个月的1号零时查看正在使用的用户列表。  【操作步骤2】  （1）再次输入命令“crontab –e”，出 现 vi 编辑器，按下“i”，屏幕进入文本输入模式。  （2）在原有内容之后，另起一行，输入“0 0 \* \*/3 \* who >who.log ”。  （3）最后保存并退出 vi 编辑器。  （4）为立即查看到 crontab 调度的结果，切换为超级用户，并适当修改系统时间，如修改为 3 月 31 日 23 点 59 分。最后退回到 helen 用户。  （5）等待 1 分钟后，查看 who.log 文件的内容，如果显示出正确的内容，那么说明新增加的 crontab 调度设置成功。  【操作要求3】  查看 cron 调度内容，最后删除此调度。  【操作步骤3】  （1）输入命令“crontab -l”，查看 cron 调度内容。  （2）输入命令“crontab -r”，删除 cron 调度内容。  （3）再次输入命令“crontab -l”，此时无 cron 调度内容。 |
| 1. **系统性能监视** |
| 【操作要求1】  利用 Shell 命令监视系统性能.  【操作步骤1】  （1）输入命令“top”，屏幕动态显示CPU利用率、内存利用率和进程状态等相关信息。  （2）按下 M 键，所有进程按照内存使用率排列。  （3）按下 T 键，所有进程按照执行时间排列。  （4）最后按下 P 键，恢复按照 CPU 使用率排列所有进程。  （5）按下 CTRL+C 组合键结束 top 命令。  【操作要求2】  利用「系统监视器」工具监视 CPU 使用情况。  【操作步骤2】  （1）启动 GNOME 桌面环境，依次单击「系统」菜单=>「管理」=>「系统监视器」，打  开「系统监视器」窗口。  （2）自动显示「资源」选项卡，查看当前 CPU、内存和交换分区、网络历史的使用情况。  【操作要求3】  利用「系统监视器」查看当前所有的进程，要求显示出启动进程的用户。  【操作步骤3】  （1）在「系统监视器」窗口单击「进程列表」选项卡，默认显示当前用户启动的所有进程。单击「查看」菜单，选中「所有的进程」单选按钮，并选中「依赖关系」复选框，则显示系统中所有的进程。  （2）单击「编辑」菜单中的「首选项」，弹出「系统监视器首选项」对话框。在「进程」选项卡，选中「进程域」栏的「用户」复选框，要求显示出启动进程的用户。单击「关闭」按钮，显示进程的各种信息。  【操作要求4】  利用「系统监视器」查看所有的文件系统。  【操作步骤4】  （1）在「系统监视器」窗口单击「文件系统」选项卡，显示当前 RHEL Server 5 系统中主要的文件系统。  （2）单击「编辑」菜单中的「首选项」，弹出「系统监视器首选项」对话框。在「文件系统」选项卡，选中「显示全部文件系统」复选框，要求显示出全部的文件系统。最后单击「关闭」按钮。  （3）「文件系统」选项卡显示全部的文件系统的信息。  【操作要求5】  利用「系统日志」工具查看系统日志。  系统日志文件都保存于/var/log 目录中，包括以下重要的日志文件：  boot.log 记录系统引导的相关信息  cron 记录 cron 调度的执行情况  dmesg 记录内核启动时的信息，主要包括硬件和文件系统的启动信息  maillog 记录邮件服务器的相关信息  messages 记录系统运行过程的相关信息，包括 I/O、网络等  rpmpkgs 记录已安装的 RPM 软件包信息  secure 记录系统安全信息  Xorg.0.log 记录图形化用户界面的 Xorg 服务器的相关信息  【操作步骤5】  超级用户依次单击「系统」菜单=>「管理」=>「系统日志」，打开「系统日志」窗口。  可分别查看各类系统日志。 |

**结果：**

|  |
| --- |
| 1. **作业和进程的基本管理** |
| **图1 作业操作**    **图2 查看进行信息和用户信息** |
| 1. **at进程调度** |
| **图2-1 设置调度**    **图2-2 查看ata调度并删除** |
| 1. **cron进程调度** |
| **图3-1 用户设置crontab调度**    **图3-2 用户添加设置crontab调度**    **图3-3 查看crontab调度并删除** |
| 1. **系统性能监视** |
| **图4-1 按cpu使用率显示进程信息**    **图4-2 按内存使用率显示进程信息**    **图4-3 按执行时间显示进程信息**    **图4-4 查看资源信息**    **图4-4 设置并查看所有进程**    **图4-5 设置进程显示首选项**    **图4-6 设置显示所有文件系统**    **图4-7 显示文件系统信息**    **图4-8 查看系统日志** |

**体会：**

本次实验中，我掌握手工启动前后台作业的方法、进程与作业管理的相关 Shell 命令以及at 调度和 cron 调度的设置方法，并了解进行系统性能监视的基本方法。

**附录：（源程序）**

本实验无源程序。

**《计算机操作系统》实验报告5**

实验题目：用户与组群管理

姓名：高远 学号：19123348 实验日期：2021.10.21

**实验环境：**

Vmware Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-38-generic x86\_64)

**实验目的：**

（1）理解/etc/passwd 和/etc/group 文件的含义。

（2）掌握桌面环境下管理用户与组群的方法。

（3）掌握利用 Shell 命令管理用户与组群的方法。

（4）掌握批量新建用户帐号的步骤和方法。

**实验内容：**

（1）桌面环境下管理用户与组群

（2）编辑用户配置文件

（3）利用Shell命令管理用户与组群

（4）批量新建多个用户账号

**操作过程：**

|  |
| --- |
| 1. **桌面环境下管理用户与组群** |
| 【操作要求1】  新建两个用户帐号，其用户名为 xuser1 和 xuser2，口令为“e12ut59er”和“wfu1t28er”。  【操作步骤】  （1） 以超级用户身份登录 X Window 图形化用户界面，依次单击「系统」菜单=>「管理」=>「用户和组群」，启动「用户管理者」窗口。  （2） 单击工具栏上的「添加用户」按钮，出现「创建新用户」窗口。在「用户名」文本框中输入用户名“xuser1”，在「口令」文本框中输入口令“e12ut59er”，在「确认口令」文本框中再次输入口令“e12ut59er”，如图 5-2 所示，然后单击「确定」按钮，返回「用户管理者」窗口。  （3） 用同样的方法新建用户 xuser2。  （4） 依次单击顶部面板的「应用程序」=>「附件」=>「文本编辑器」，启动 gedit 文本编  辑器，打开/etc/passwd 和/etc/shadow 文件将发现文件的末尾出现表示 xuser1 和 xuser2户帐号的信息。打开/etc/group 和/etc/gshadow 文件将发现文件末尾出现表示 xuser1和 xuser2 私人组群的信息。  （5） 按下 CTRL+ALT+F2 组合键切换到第 2 个虚拟终端，输入用户名 xuser2 和相应的口令可登录 Linux 系统，说明新建用户操作已成功。  （6） 输入“pwd”命令，屏幕显示用户登录后进入用户主目录“/home/xuser2”。  （7） 输入“exit”命令，xuser2 用户退出登录。  （8） 按下 ALT+F7 组合键返回 GNOME 桌面环境。  【操作要求2】  锁定xuser2用户账号  【操作步骤】  （1） 在「用户管理者」窗口选中 xuser2 用户帐号，单击工具栏上的「属性」按钮，打开「用户属性」窗口。  （2） 选中「帐号信息」选项卡让「本地口令被锁」复选框被选中。单击「确定」按钮，返回「用户管理者」窗口。  （3） 按下 CTRL+ALT+F2 组合键，再次切换到第 2 个虚拟终端，输入用户名 xuser2 和相应的口令，发现 xuser2 用户无法登录 Linux 系统，说明 xuser2 用户账号的确已被锁定。  【操作要求3】  删除xuser2用户  【操作步骤】  （1） 在「用户管理者」窗口，单击「编辑」菜单的「首选项」，弹出「首选项」对话框，不选中「隐藏系统用户和组」复选框，如图 5-5 所示，最后单击「关闭」按钮。此时「用户」选项卡中显示包括超级用户和系统用户在内的所有用户。  （2） 在「搜索过滤器」文本框中输入“x\*”并按下 Enter 键，则仅显示以 x 为首字母的用户。  （3） 选中 xuser2 用户，单击工具栏上的「删除」按钮，弹出对话框，如图 5-8 所示，单击「是」按钮，返回「用户管理者」窗口，发现 xuser2 用户已被删除。  （4） 在「搜索过滤器」文本框中输入“\*”并按下 Enter 键，则显示所有用户。  【操作要求4】  新建两个组群，分别是 myusers 和 temp。  【操作步骤】  （1） 在「用户管理者」窗口选中「组群」选项卡，当前显示出所有组群。  （2） 单击工具栏上的「添加组群」按钮，出现「创建新组群」对话框。在「组群名」文本框中输入“myusers”，单击「确定」按钮，返回「用户管理者」窗口。  （3） 用相同的方法新建 temp 组群。  【操作要求5】  修改 myusers 组群属性，将 xuser1 和 helen 用户加入 myusers 组群。  【操作步骤】  （1） 从「组群」选项卡中选择 myusers 组群，单击工具栏上的「属性」按钮，弹出「组群属性」窗口。  （2） 选择「组群用户」选项卡，选中 helen 和 xuser1 前的复选框，设置 helen 用户和 xuser1用户的 myusers 组群的成员，如图 5-11 所示。单击「确定」按钮，返回「用户管理者」窗口。  【操作要求6】  删除 temp 组群  【操作步骤】  （1）从「组群」选项卡中选择 temp 组群，单击工具栏上的「删除」按钮，出现确认对话框，单击「是」按钮即可。 |
| 1. **编辑用户配置文件** |
| 【操作要求1】  新建用户配置文件 myusers-profile  【操作步骤】  （1） 依次单击「系统」菜单=>「管理」=>「用户配置文件编辑器」，打开「User Profile Editor」窗口。  （2） 单击「添加」按钮，弹出「Add Profile」窗口，在「Profile name」文本框中输入用户配置文件名“myusers-profile”。单击「添加」按钮，回到「User Profile Editor」窗口。  【操作要求2】  设置 myusers-profile 用户配置文件的内容：应用程序的默认字体为中易宋体18030，桌面背景为花园。  【操作步骤】  （1） 在「User Profile Editor」窗口选中“myusers-profile”文件，单击「编辑」按钮，出现「编辑配置文件 myusers-profile」窗口。  （2） 在「编辑配置用户」窗口中依次单击「系统」菜单=>「首选项」=>「字体」，打开「字体首选项」对话框，如图 5-16 所示。单击应用程序字体的字体列表，出现「拾取字体」对话框，从 「字体族」选择“中易宋体 18030”，并单击「确定」按钮。  （3） 回到「字体首选项」对话框，如图 5-18 所示，此时窗口中的字体发生变化，单击窗口右上角的关闭按钮，关闭此对话框。  （4） 在「编辑配置用户」窗口中依次单击「系统」菜单=>「首选项」=>「桌面背景」，打开「桌面背景首选项」对话框，选择“花园”。此时「编辑配置用户」窗口的桌面也发生变化。最后单击「关闭」按钮。  （5） 单击「编辑配置用户」窗口「配置文件」菜单的「保存」项，保存用户配置文件的修改内容。最后单击「编辑配置用户」窗口右上角的关闭按钮，回到「User Profile Editor」窗口。  【操作要求3】  设置 xuser1 的用户配置文件为 myusers-profile  【操作步骤】  （1） 在「User Profile Editor」窗口选中 myusers-profile 文件，单击「Users」按钮，出现「配置文件 myusers-profile 的用户」对话框。  （2） 选中 xuser1 用户的复选框，如图 5-20 所示，最后单击「关闭」按钮。  （3） 单击「系统」菜单的「注销」项，超级用户退出 GNOME 桌面环境。  （4） 以 xuser1 用户登录，并启动 GNOME 桌面环境，查看应用程序的字体和桌面环境。 |
| 1. **利用Shell命令管理用户与组群** |
| 【操作要求1】  新建一名为 duser 的用户，其口令是“tdd63u2”，主要组群为 myusers。  【操作步骤】  （1） 按下 CTRL+ALT+F3 组合键，切换到第 3 个虚拟终端，以超级用户身份登录。  （2） 输入命令“useradd -g myusers duser”，建立新用户 duser，其主要组群是 myusers。  （3） 为新用户设置口令，输入命令“passwd duser”，根据屏幕提示输入两次口令，最后屏幕提示口令成功设置信息。  （4） 输入命令“cat /etc/passwd”，查看 /etc/passwd 文件的内容，发现文件的末尾增加 duser用户的信息。  （5） 输入命令“cat /etc/ group”，查 看/etc/ group 文件的内容，发现文件内容未增加。  （6） 按下 ALT+F4 组合键，切换到第 4 个虚拟终端，输入 dusr 用户名和口令可登录 Linux系统。  （7） 输入“exit”命令，duser 用户退出登录。  【操作要求2】  将 duser 用户设置为不需口令就能登录。  【操作步骤】  （1） 按下 ALT+F3 组合键，切换到正被超级用户使用的第 3 个虚拟终端。  （2） 输入命令“passwd -d duser”。  （3） 按下 ALT+F3 组合键，再次切换到第 3 个虚拟终端，在“Login：”后输入用户名“duser”，按下 Enter 键就直接出现 Shell 命令提示符，说明 duser 用户不需口令即可登录。  【操作要求3】  查看 duser 用户的相关信息。  【操作步骤】   1. 在第3个虚拟终端输入命令“id duser”，显示 duser 用户的用户 ID（UID）、主要组群的名称和 ID（GID）。   【操作要求4】  从普通用户 duser 切换为超级用户。  【操作步骤】  （1） 第4个虚拟终端当前的 Shell 命令提示符为“$”，表明当前用户是普通用户。  （2） 输入命令“ls /root”，屏幕上没有出现/root 目录中文件和子目录的信息，而是出现提示信息，提示当前用户没有查看/root 目录的权限。  （3） 输入命令“su -”或者是“su - root”，屏幕提示输入口令，此时输入超级用户的口令，验证成功后 Shell 提示符从“$”变为“#”，说明已从普通用户转换为超级用户。  （4） 再次输入命令“ls /root”，可查看/root 目录中文件和子目录的信息。  （5） 输入“exit”命令，回到普通用户的工作状态。  （6） 输入“exit”命令，duser 用户退出登录。  【操作要求5】  一次性删除 duser 用户及其工作目录。  【操作步骤】  （1） 按下 ALT+F3 组合键，切换到正被超级用户使用的第 3 个虚拟终端。  （2） 输入命令“userdel –r duser”，删除 duser 用户。  处于登录状态的用户不能删除。如果在新建这个用户时还创建了私人组群，而该私人组  群当前又没有其他用户，那么在删除用户的同时也将一并删除这一私人组群。  （3） 输入命令“cat /etc/passwd”，查 看/etc/passwd 文件的内容，发现 duser 的相关信息已消失。  （4） 输入命令“ls /home”，发 现 duser 的主目录/home/duser 也不复存在。  【操作要求 6】  新建组群 mygroup。  【操作步骤】  （1） 在超级用户的Shell提示符后输入命令“groupadd mygroup”，建立mygroup组群。  （2） 输入命令“cat /etc/group”，发现group文件的末尾出现 mygroup 组群的信息。  （3） 输入命令“cat /etc/gshadow”，发现gshadow文件的末尾也出现 mygroup 组群的信息。  【操作要求7】  将 mygroup 组群改名为 newgroup。  【操作步骤】  （1） 输入命令“groupmod –n newgroup mygroup”，其中–n 选项表示更改组群的名称。  （2） 输入命令“cat /etc/group”，查 看组群信息，发现原来 mygroup 所在行的第一项变为“newgroup”。  【操作要求 8】  删除 newgroup 组群。  【操作步骤】  （1）超级用户输入“groupdel newgroup” 命令，删除 newgroup 组群。 |
| 1. **批量新建多个用户账号** |
| 【操作要求】  为全班同学 20 位同学创建用户帐号，用户名为“s”+学号的组合，其中班级名册中第一位同学的学号为 080101。所有同学都属于 class0801 组群。所有同学的初始口令为 111111。  【操作步骤】  （1） 以超级用户身份登录，输入命令“groupadd -g 600 class0801”（假设值为 600的 GID 未被使用），新建全班同学的组群 class0801。  （2） 输入命令“vi student”，新建用户信息文件。  （3） 按下“i”键，切换为 vi 的文本编辑模式，输入第一行信息： s080101:x:601:600::/home/s080101:/bin/bash”。  （4） 按下 ESC 键，切换到命令行模式，拖动鼠标，将整行选中，然后按下字母键 y 两次。也就是将当前选中的行放到 vi 的暂存区域（类似于 Windows 的剪贴板）。  （5） 然后按下字母键 p，就复制一行信息，重复此操作 19 次，然后部分修改每位同学用户信息不同的地方。  （6） 最后编辑完成的文件，为节约篇幅仅显示前 10 位同学信息。最后  保存并退出 vi。  （7） 输入命令“vi stu-passwd”，新建用户口令文件。  （8） 按下“i”键，切换为 vi 的文本编辑模式，输入第一行信息：“s080101:111111”，即所有同学的初始口令为 111111。按下 ESC 键，切换到命令行模式，拖动鼠标，将整行选中，然后按下字母键 y 两次，复制行。  （9） 连续按 p 键 19 次，就可复制出 19 行信息，然后修改成正确的用户名。  （10） 输入命令“newusers < students”，批量新建用户帐号。  （11） 输入命令“pwunconv”，暂时取消 shadow 加密。  （12） 输入命令“chpasswd <stu-passwd”，批量新建用户的口令。  （13） 输入命令“pwconv”，进行 shadow 加密，完成批量创建用户帐号工作。  （14） 输入命令“cat /etc/passwd”，查看/etc/passwd 文件将发现所有的用户帐号均已建立。  （15） 可尝试以新建的用户名登录，并应该及时修改用户的口令。 |

**结果：**

|  |
| --- |
| 1. **桌面环境下管理用户与组群** |
| **图1-1 打开用户管理者窗口**    **图1-2 添加 xuer1 用户**    **图1-3 完成用户添加**  **图1-4 /etc/passwd文件**    **图1-5 /etc/group文件**    **图1-6 登录xuser2用户**    **图1-7 锁定xuser2用户账号**    **图1-8 查看xuser2状态**    **图1-9 切换用户界面**    **图1-10 删除xuser2用户**    **图1-11 删除后界面**    **图1-12 创建两个组**    **图1-13 添加用户到组**    **图1-14 删除“temp” 组群** |
| 1. **编辑用户配置文件** |
| **图2-1 更换壁纸**    **图2-2 安装gnome-tweak桌面配置工具**    **图2-3 执行gnome-tweaks命令** |
| 1. **利用Shell命令管理用户与组群** |
| **图3-1 新建用户并设置密码**    **图3-2 查看/etc/passwd命令**    **图3-3 查看/etc/group命令**    **图3-4 将duser设置为无需密码登录**    **图3-5 验证duser登录不需要密码**    **图3-6 查看duser用户的相关信息**    **图3-7 duser用户查看/root目录**    **图3-8 root用户查看/root目录**    **图3-9 删除duser用户**    **图3-10 查看/etc/passwd文件**    **图3-11 查看/home目录**    **图3-12 root用户新建“mygroup”组群**    **图3-13 查看/etc/group文件**    **图3-14 查看/etc/gshadow文件**    **图3-15 将mygroup改名为newgroup**    **图3-16 查看/etc/gshadow文件**    **图3-17 删除newgroup组群** |
| 1. **批量新建多个用户账号** |
| **图4-1 新建用户文件**    **图4-2 新建用户口令文件**    **图4-3 批量新建用户及其口令**    **图4-4 查看/etc/passwd文件** |

**体会：**

在本实验中，我理解了/etc/passwd 和/etc/group 文件的含义、掌握桌面环境下管理用户与组群的方法、掌握利用 Shell 命令管理用户与组群的方法并掌握批量新建用户帐号的步骤和方法。同时在使用vi命令时熟悉了文本编辑器的使用。

**附录：（源程序）**

本实验无源程序。

**《计算机操作系统》实验报告6**

实验题目：SHELL编程

姓名：高远 学号：19123348 实验日期：2021.11.04

**实验环境：**

Vmware Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-38-generic x86\_64)

**实验目的：**

1. 掌握vi的三种工作方式，熟悉vi编辑程序的使用。
2. 学习Shell程序设计方法。掌握编程要领。

**实验内容：**

1. 学习使用vi编辑程序。
2. 编写Shell程序。
3. 将程序文件设置为可执行文件（用chmod命令）。
4. 在命令行方式中运行Shell程序。

**操作过程：**

|  |
| --- |
| 1. 按本《实验指导》第三部分的内容。熟悉vi 的三种工作方式。熟悉使用各种编辑功能。 |
| 1. 创建和执行Shell 程序   用前面介绍的Vi 或其他文本编辑器编写Shell 程序，并将文件以文本文件方式保存在相应的目录中。  用chmod 将文件的权限设置为可执行模式，如若文件名为shdemo.h,则命令如下：  $ chmod 755 shdemo.h(文件主可读、写、执行，同组人和其他人可读和执行)  在提示符后执行Shell 程序：  $ shdemo.h （直接键入程序文件名执行）  或 $ sh shdemo.h （执行Shell 程序）  或 $ .shdemo.h （没有设置权限时可用点号引导） |
| 1. 用vi 编写《实验指导》“第四部分Shell 程序设计”中的例1（假设文件名为prog1.h），练习内部变量和位置参数的用法。   用chmod将文件的权限设置为可执行模式，并在提示符后键入命令行：  $./prog1.  或 $sh prog1.  屏幕显示:  Name not provided  在提示符后键入命令行：  $./prog1.h Theodore #有一个参数  屏幕显示:  Your name is Theodore #引用$1 参数的效果 |
| 1. 进一步修改程序prog1.h，要求显示参数个数、程序名字，并逐个显示参数。 |
| 1. 修改例1程序（即上面的 prog1.h），用read命令接受键盘输入。若没有输入显示第一种提示，否则第二种提示。 |
| 1. 用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例2、例3，练习字符串比较运算符、数据比较运算符和文件运算符的用法，观察运行结果。 |
| 1. 修改例2程序，使在程序运行中能随机输入字符串，然后进行字符串比较。 |
| 1. 修改例3程序，使在程序运行中能随机输入文件名，然后进行文件属性判断。 |
| 1. 用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例4、例5、例6、例7，掌握控制语句的用法，观察运行结果。 |
| 1. 用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例8 及例9掌握条件语句的用法，函数的用法，观察运行结果。 |
| 1. 编程，在屏幕上显示用户主目录名（HOME）、命令搜索路径（PATH），并显示由位置参数指定的文件的类型和操作权限。 |

**结果：**

|  |
| --- |
| 思考题：试一试vi 的三种工作方式各用在何时？用什么命令进入插入方式？怎样退出插入方式？文件怎样存盘？注意存盘后的提示信息。  答：  一、三种工作方式   * 1. 插入方式：   插入方式可用插入命令（见表 2）的任何一个进入，屏幕下方有一行“------Insert------”字样表示。插入时用户输入的任何字符都认为是文本的内容，且可以用退格键来纠正错误。在插入方式中按一下<ESC>，即退出插入方式，进入转义命令方式。   * 1. 转义命令方式：   刚进入 vi 或退出插入方式，即为转义命令方式。这时键入的任何字符转义为特殊功能，如：移动、删除、替换等。大多数转义命令由一个或两个字母组成，操作时没有提示符，而且输入命令不需要按<ENTER>。   * 1. 末行命令方式：   在转义命令方式中，按冒号“：”就进入末行命令方式。屏幕最末一行的行首显示冒号作为命令提示。命令行输入后按<ENTER>开始执行。此时用户可进行文件的全局操作，如：全局查找、替换、文件读、写等。  二、进入插入方式  主要的文本插入：  i text <ESC> 在光标前插入新文本，ENTER 可重起一行  I text <ESC> 在当前行起始处插入新文本  a text <ESC> 在光标后输入新的文本  A text <ESC> 在当前行末尾输入新的文本  o text <ESC> 在当前行下产生新的一行并输入文本  O text <ESC> 在当前行上产生新的一行并输入文本  三、退出插入方式  用<ESC>键可退出输入模式。  四、存盘：  键入命令：  ：w  命令 w 将文件以当前名字存入磁盘，并覆盖了文件先前的副本。但 w 命令不影响缓冲区的内容。  如果要将文件以不同的名字保存，例如 newfile，需键入命令：  ：w newfile  如果不存在名为 newfile 的文件，vi 编辑器将存入文件并给出文件的大小。如果这个文件已经存在，那么 vi 会通知你，但并不刷新文件的内容。如果想刷新已存在的文件的内容，则可键入命令：  ：w！newfile  同样，感叹号“！”放弃了标准的 vi 防止覆盖文件的保护功能，强制刷新。 |
| **图1 用vim编写shell程序**    **图2 执行shell程序** |
| **图3 编写prog1.h**    **图4 执行程序** |
| **图5 修改程序prog1.h**    **图6 执行程序** |
| **图7 修改程序prog1.h**    **图 8 执行程序** |
| **图9 编写程序prog2.h**    **图10 执行程序**    **图11 编写程序prog3.h**    **图12 创建目录filea 以及子目录cppdir并查看权限**    **图13 执行程序** |
| **图14 修改程序prog2.h**    **图15 执行程序** |
| **图16 修改程序prog3.h**    **图17 执行程序** |
| **图18 编写程序prog4.h**    **图19 执行程序**    **图20 编写程序prog5.h**    **图21 执行程序**    **图22 编写程序prog6.h**    **图23 执行程序**    **图24 编写程序prog7.h**    **图25 执行程序** |
| **图26 编写程序prog8.h**    **图27 执行程序**    **图28 编写程序prog9.h**    **图30 执行程序** |
| **图31 编写程序prog10.h**    **图32 执行程序**  思考题：到此为止你对Shell 有所认识了吧？怎么样？自己再编两个程序：   * 1. 做个批处理程序，体会一下批处理概念。   2. 做个菜单，显示系统环境参数。将此程序设置为人人可用。     **图33 编写批处理程序prog11.h实现**    **图34 执行程序**    **图35 编写菜单程序prog12.h**    **图36 执行程序** |
| 讨论题： |
| 1. **Linux的Shell有什么特点？**   （1）把已有命令进行适当组合构成新的命令。  （2）提供了文件名扩展字符（通配符，如\* 、 ?、 [ ]），使得用单一的字符串可以匹配多个文件名，省去键入一长串文件名的麻烦。  （3）可以直接使用Shell的内置命令,而不需创建新的进程，如Shell中提供的cd、echo、exit、pwd、kill等命令。为防止因某些Shell不支持这类命令而出现麻烦，许多命令都提供了对应的二进制代码，从而也可以在新进程中运行。  （4）Shell允许灵活地使用数据流，提供通配符、输入/输出重定向、管道线等机制，方便了模式匹配、I/O处理和数据传输。  （5）结构化的程序模块，提供了顺序流程控制、条件控制、循环控制等。  （6）Shell提供了在后台执行命令的能力。  （7）Shell提供了可配置的环境，允许创建和修改命令、命令提示符和其它的系统行为。  （8）Shell提供了一个高级的命令语言，能够创建从简单到复杂的程序。这些Shell程序称为Shell脚本，利用Shell脚本，可把用户编写的可执行程序与Unix命令结合在一起，当作新的命令使用，从而便于用户开发新的命令。 |
| 1. **怎样进行Shell编程？如何运行？有什么条件？**   shell 程序是利用文本编辑程序建立的一系列 Linux 命令和实用程序序列的文本文件。它可以像 Linux 的任何命令一样执行。在执行 Shell 程序时，Linux 一个接一个地解释并执行每个命令。Shell 是一种很成熟的程序设计语言，跟其他任何语言一样，有自己的语法、变量和控制语句。可以使用vi或vim等文本编辑器编写程序文件。先写入shell命令行或linux命令行后保存，再在终端使用chmod命令将文件的权限设置为可执行模式。如文件名为 shdemo.h,则命令如下：chmod 755 shdemo.h(文件主可读、写、执行，同组人和其他人可读和执行)，最后在 Shell 提示符后直接键入程序文件名执行:$ shdemo.h或$ sh shdemo.h或$ .shdemo.h(没有设置权限时可用点号引导)。 |
| 1. **vi编辑程序有几种工作方式？查找有关的详细资料，熟练掌握屏幕编辑方式、转移命令方式以及末行命令的操作。学习搜索、替换字符、字和行，行的复制、移动，以及在vi中执行Shell命令的方式。**   vi编辑程序有三种工作方式：插入方式、转移命令方式与末行命令方式。  ①插入方式  插入方式可用插入命令（见表 2）的任何一个进入，屏幕下方有一行“------Insert------”字样表示。插入时用户输入的任何字符都认为是文本的内容，且可以用退格键来纠正错误。在插入方式中按一下<ESC>，即退出插入方式，进入转义命令方式。  ②转义命令方式  刚进入 vi 或退出插入方式，即为转义命令方式。这时键入的任何字符转义为特殊功能，如：移动、删除、替换等。大多数转义命令由一个或两个字母组成，操作时没有提示符，而且输入命令不需要按<ENTER>。  ③末行命令方式  在转义命令方式中，按冒号“：”就进入末行命令方式。屏幕最末一行的行首显示冒号作为命令提示。命令行输入后按<ENTER>开始执行。此时用户可进行文件的全局操作，如：全局查找、替换、文件读、写等。 |
| 1. **编写一个具有以下功能的Shell程序。**   **(1) 把当前目录下的文件目录信息输出到文件 filedir.txt 中；**  **(2) 在当前目录下建立一个子目录，目录名为 testdir2 ；**  **(3) 把当前目录下的所有扩展名为 c 的文件以原文件名复制到子目录testdir2中；**  **(4) 把子目录中的所有文件的存取权限改为不可读。（提示：用 for 循环控制语句实现，循环的控制列表用’ls’ 产生。）**  **(5) 在把子目录 testdir2 中所有文件的目录信息追加到文件 filedir.txt 中；**  **(6) 把你的用户信息追加到文件 filedir.txt 中；**  **(7) 分屏显示文件 filedir.txt**    **图37 编写程序prog13.h**    **图38 执行程序** |

**体会：**

通过此次实验，我学习使用了vim编写Shell程序、将程序文件设置为可执行文件（用chmod命令）并在命令行方式中运行Shell程序。过程中也遇到了许多语句上的小错误，但是通过寻找资料和不断试错，最终也都得到了结果。这次的实验让我对于使用vi编程的理解变得更加深刻，也更加熟悉如何使用linux的文本编辑器，同时也了解了变量、参数、操作符和重复语句等的概念和操作。

**附录：（源程序）**

|  |
| --- |
| 1. **程序prog1.h**   #Name display program  read name  if [ $name ]  then  echo "Your name is " $name  else  echo "Name not provided "  fi |
| 1. **程序prog2.h**   read string1  read string2  if [ $string1 = $string2 ]  then  echo "string1 equal to string2"  else  echo "string1 not equal to string2"  fi  if [ string1 ]  then  echo "string1 is not empty"  else  echo "string1 is empty"  fi  if [ -n string2 ]  then  echo "string2 has a length greater than zero"  else  echo "string2 has a length equal to zero"  fi |
| 1. **程序prog3.h**   read filename  if [ -d $filename ]  then  echo $filename "is a directory"  else  echo $filename "is not a directory"  fi  if [ -f $filename ]  then  echo $filename "is a regular file"  else  echo $filename "is not a regular file"  fi  if [ -r $filename ]  then  echo $filename "has read permission"  else  echo $filename "filea does not have read permission"  fi  if [ -w $filename ]  then  echo $filename "has write permission"  else  echo $filename "does not have write permission"  fi  if [ -x $filename ]  then  echo $filename "has execute permission"  else  echo $filename "does not have execute permission"  fi |
| 1. **程序prog4.h**   for filename in 'ls'  do  cp $filename backup/$filename  if [ $? -ne 0 ]  then  echo " copy $filename failed"  fi  done |
| 1. **程序prog5.h**   loopcount=0  result=0  while [ $loopcount -lt 10 ]  do  let loopcount=$loopcount+1  let result=$result+$loopcount\*2  done  echo "result is" $result |
| 1. **程序prog6.h**   result=0  until [ $loopcount -ge 10 ]  do  loopcount=`expr $loopcount + 1`  result=`expr $result + $loopcount \\* 2`  done  echo "result is" $result |
| 1. **程序prog7.h**   select item in continue finish  do  if [ $item = "finish" ]  then  break  fi  done |
| 1. **程序prog8.h**   case $1 in  01|1) echo "Month is January" ;;  02|2) echo "Month is February" ;;  03|3) echo "Month is March" ;;  04|4) echo "Month is April" ;;  05|5) echo "Month is May" ;;  06|6) echo "Month is June" ;;  07|7) echo "Month is July" ;;  08|8) echo "Month is August" ;;  09|9) echo "Month is September" ;;  010|10) echo "Month is October" ;;  011|11) echo "Month is November" ;;  012|12) echo "Month is December" ;;  esac |
| 1. **程序prog9.h**   displaymonth()  {  case $1 in  01|1) echo "Month is January" ;;  02|2) echo "Month is February" ;;  03|3) echo "Month is March" ;;  04|4) echo "Month is April" ;;  05|5) echo "Month is May" ;;  06|6) echo "Month is June" ;;  07|7) echo "Month is July" ;;  08|8) echo "Month is August" ;;  09|9) echo "Month is September" ;;  010|10) echo "Month is October" ;;  011|11) echo "Month is November" ;;  012|12) echo "Month is December" ;;  esac  }  displaymonth 4  displaymonth 8  displaymonth 12 |
| 1. **程序prog10.h**   echo "home :" $HOME  echo "path :" $PATH  if [ $# -ne 0 ]  then  echo " `ls -ld $1` "  fi |
| 1. **程序prog11.h**   ./prog8.h>prog8.txt  vim prog8.txt |
| 1. **程序prog12.h**   select cho in env finish  do  if [ $cho = "finish" ]  then  echo "It is finished"  break  else  $cho  fi  done |
| 1. **程序prog13.h**   ls -o>filedir.txt  mkdir testdir5  chmod 755 testdir5  cp $( find \*.c) testdir5  cd testdir5  for filename in `ls`  do  chmod 311 $filename  done  cd  cd /home/gy  ls -o testdir5>>filedir.txt  id>>filedir.txt  less filedir.txt |