广东工业大学

计算机科学与技术学院

《Linux技术》

实验报告



|  |  |
| --- | --- |
| 课程名称： | Linux技术 |
| 实验名称： | Linux环境编程、系统管理 |
| 指导教师： | 赵靖亮 |
| 学生姓名： | 王宁 |
| 班级学号： | 4-3120005084 |
| 实验日期： | 2023-05-29 |
| 实验时间： | 4学时 |

实验报告撰写要求

实验操作是教学过程中理论联系实际的重要环节，而实验报告的撰写又是知识系统化的吸收和升华过程，因此，实验报告应该体现完整性、规范性、正确性、有效性。现将实验报告撰写的有关内容说明如下：

1、 实验报告模板为电子版。

2、 下载统一的实验报告模板，学生自行完成撰写和打印。报告的首页包含本次实验的一般信息：

*  班级学号：例如：2-5 表示第二班学号为5。
*  实验日期：例如：2023-05-09 表示本次实验日期。
*  实验时间：例如：4学时 表示本次实验所用的时间。

实验报告正文部分，从五个方面（目的、内容、步骤等）反映本次实验的要点、要求以及完成过程等情况。模板已为实验报告正文设定统一格式，一、二已填好，学生只需填三、四、五。三和四计分，五不计分。

3、 实验过程中可灵活参考教材、PPT和网络教程以解决问题。报告应独立完成，若出现雷同报告，均按零分处理。

4、 实验报告正文部分具体要求如下：

一、实验目的

本次实验所涉及并要求掌握的知识点。

二、实验内容

实验内容、原理分析及具体实验要求。

三、主要实验步骤

根据具体实验，记录、整理相应操作、命令、运行结果等，包括截图和文字说明。

若显示内容过长，可仅截取关键部分图。

具体项目已给出，学生在项目下面添加实验记录。

四、相关知识测试

完成思考题目。

五、实验心得体会

若在实验过程中出现故障和问题，应详细记录并进行故障分析，说明故障排除的过程及方法。

一、实验目的

*本次实验所涉及并要求掌握的知识点。*

**1. Linux环境编程**

1.1理解系统调用和库函数的异同。

1.2学会用系统调用和库函数进行编程。

1.3掌握一些常用的系统调用和库函数的功能及应用。

**2.** **系统管理**

2.1 理解系统管理的内涵和作用。

2.2学会对用户和组进行一般管理。

二、实验内容

**1. Linux环境编程**

1.1使用系统调用对文件进行操作。

2.2使用系统调用对进程进行控制。

3.3使用管道机制进行进程通信。

4.4使用消息队列进行进程通信。

**2. 系统管理**

2.1为新用户建立账号和工作组，删除本地用户和组。

2.2在 U 盘上建立文件系统，并进行安装。

三、主要实验步骤

**1. Linux环境编程**

1.1 编写一个程序，把一个文件的内容复制到另一个文件上，即实现简单的copy功能。要求：只用open()，read()，write()和close()系统调用，程序的第一个参数是源文件，第二个参数是目的文件。

**代码：**

文本

描述已自动生成

**编译执行：**

gcc demo1.c -o demo1.o

# 把src\_file.txt文件内容复制到dest\_file.txt文件中

./demo1.o src\_file.txt dest\_file.txt

# 其余参数错误运行请看代码中的错误处理

图形用户界面, 文本, 应用程序

描述已自动生成1.2 编写一个程序，它首先打开一个文件，然后利用fork()创建一个子进程；随后，当父进程运行时先执行wait()；父子进程都打印自己和其父进程的ID号；并且，二者都向该文件写入（利用write）一条信息，表明是在哪个进程中。

**代码：**

图形用户界面, 文本

描述已自动生成

图片包含 文本

描述已自动生成

**编译执行：**

gcc demo2.c -o demo2.o

./demo2.o

# 首先打开文件process.txt，fork()创建子进程

# 当父进程运行时先执行wait()等待子进程执行结束

# 父子进程都打印自己和其父进程的ID号

# 并都向该文件写入（利用write）一条信息，表明是在哪个进程中

图形用户界面, 文本, 应用程序

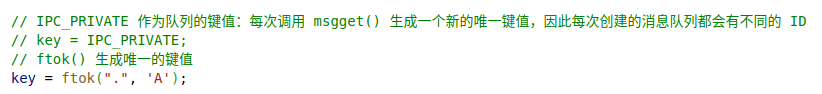
描述已自动生成

1.3 调用msgget()创建一个消息队列，输出其队列ID。然后再次调用msgget()打开已有的队列。如果两次得到的ID相同，则显示打开队列的ID。

**生成消息队列键值：**

每次调用msgget()生成不同的键值则导致消息队列ID不同；

每次调用msgget()生成相同键值则导致消息队列ID相同。



**代码：**

在编译执行时分别启用第15、17行代码查看消息队列键值对消息队列ID的影响。

文本

描述已自动生成

图形用户界面, 文本

描述已自动生成

**编译执行：**

1. **IPC\_PRIVATE作为消息队列键值**

gcc demo3.c -o demo3.o

./demo3.o

# 每次调用msgget()生成不同的键值则导致消息队列ID不同

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

1. **ftok()生成唯一的键值**

gcc demo3.c -o demo3.o

./demo3.o

# 每次调用msgget()生成相同键值则导致消息队列ID相同

文本

中度可信度描述已自动生成

**2. 系统管理**

2.1 分别以普通用户和 root 身份登录，看能否建立新用户账号。

1. **以普通用户登录，建立新用户账号**

文本

描述已自动生成

1. **以root身份登录，建立新用户账号**

文本

描述已自动生成

2.2 为新用户(如 Zhang San)建立账号和工作组，并进行相应配置；以该用户身份登录，修改密码等；最后删除该用户。

1. **为新用户ZhangSan建立账号和工作组**

文本

描述已自动生成

1. **配置用户的相关设置，如用户权限、主目录等**

# 将用户的默认shell设置为Bash，

# 并将主目录设置为"/home/ZhangSan"

usermod --shell /bin/bash --home /home/ZhangSan ZhangSan

1. **以该用户身份登录**

文本

描述已自动生成

1. **修改密码**

文本

描述已自动生成

1. **删除该用户**

# 删除用户及其主目录

# -r选项将同时删除用户的主目录

sudo userdel -r ZhangSan

文本

描述已自动生成

ZhangSan该用户及其所在的用户组都被删除

文本

描述已自动生成

2.3 在U盘上建立一个文件系统(类型为ext3或ext4)，然后安装到根文件系统上。然后，将根文件系统上的某个目录或文件复制到子文件系统中。最后卸下该子文件系统。

1. **插入U盘并确定其设备名称**

# 查看已连接的存储设备列表

sudo fdisk -l

# 根据设备的大小和类型，找到对应的U盘设备名称

# 假设U盘设备名为/dev/sdb

1. **对U盘进行分区和格式化**

# 使用工具如fdisk或parted创建一个ext4文件系统的分区

sudo fdisk /dev/sdb # 进入fdisk工具

n # 创建新分区

<Enter> # 默认选择主分区

<Enter> # 默认选择下一个可用分区编号

<Enter> # 默认选择起始扇区

<Enter> # 默认选择结束扇区

t # 设置分区类型

83 # 选择Linux文件系统类型

# 83:Linux native file systems(ext2,ext3,ext4)

w # 保存并退出

# 使用mkfs命令创建相应的文件系统

sudo mkfs.ext4 /dev/sdb1

# 注意将/dev/sdb1替换为实际的U盘分区设备

1. **挂载U盘分区到一个临时目录**

# 创建一个空目录作为挂载点，假设为/mnt/usb

sudo mkdir /mnt/usb

sudo mount /dev/sdb1 /mnt/usb

1. **复制根文件系统上的目录或文件到U盘分区中**

# 将目标文件或目录复制到挂载的U盘分区

sudo cp -a /path/to/source /mnt/usb

# 将/path/to/source替换为要复制的源文件或目录的路径

1. **完成复制后，卸载U盘分区**

# 卸载U盘分区

sudo umount /mnt/usb

# 确保没有任何进程在使用U盘分区后再执行卸载操作

四、相关知识测试

* 1. 什么是系统调用？什么是库函数？二者有何异同？

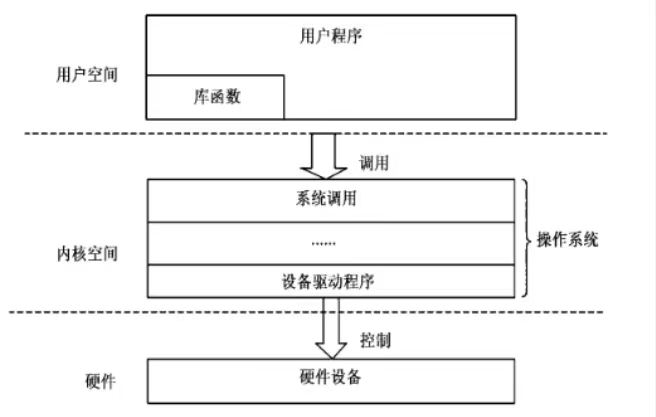
系统调用是操作系统提供给应用程序的接口，用于访问底层操作系统的功能和资源。应用程序通过系统调用向操作系统发出请求，以执行特定的操作，如文件操作、进程管理、网络通信等。系统调用提供了一个受控的方式，允许应用程序与操作系统进行交互，并利用操作系统提供的功能来完成各种任务。

库函数是由编程语言或操作系统提供的一组预定义函数，用来解决共性问题，并为程序的开发和执行提供更方便的环境。这些函数被封装在库文件中，供应用程序调用。库函数通常是在用户空间中执行的，使用高级语言编写，并提供了一系列的接口，隐藏了底层系统调用的复杂性。

二者异同点如下：

1. 目的：系统调用用于访问操作系统的底层功能和资源，而库函数提供了高级的功能和算法，以供应用程序调用。
2. 实现：系统调用是由操作系统内核提供的接口，涉及内核态和用户态之间的切换；而库函数是由编程语言或操作系统提供的预定义函数，在用户态下执行。
3. 功能：系统调用提供了对底层操作系统功能的直接访问，如文件系统、进程管理、网络通信等；库函数提供了高级功能和算法，如字符串处理、数学计算、图形操作等。
4. 复杂性：系统调用通常比较底层，需要处理更多的细节和资源管理；而库函数更加抽象和简化，提供了更高级的接口，隐藏了底层实现的复杂性。
5. 开销：系统调用的开销相对较高，涉及到用户态和内核态之间的切换；而库函数的开销相对较低，通常在用户态下执行，不涉及切换。
6. 调用方式：系统调用使用特定的指令或中断来触发操作系统的功能；库函数通过在程序中调用函数名来执行相应的操作。

综上所述，它们提供了不同层次的接口和功能。系统调用用于底层的操作系统访问，而库函数提供了更高级的功能和算法，使开发人员能够更方便地编写应用程序。系统调用与库函数的关系：



* 1. 使用系统调用的一般方式是什么？

在Unix和Linux系统中，系统调用是以C函数形式提供给用户的，有类型、名称、参数，并且要表明相应的文件包含。

不同的系统调用所需要的头文件不同，这些头文件包含了相应程序代码中用到的宏定义、类型定义、全局变量、函数说明等。

对C语言来说，这些头文件一般放在：/usr/include/sys或/usr/include/linux。

1.3 c语言程序pipe1.3.c实现了类似如下shell脚本的功能：

$ echo good morning|sed s/good/hi/g

结合源文件，回答以下问题：

1. 该程序的实现过程？

该程序实现了一个简单的管道通信，创建了一个管道，通过左右两个子进程来实现数据的传递和处理。

1. 通过pipe()创建了一个管道，pipefd数组保存了管道的文件描述符
2. 通过fork()创建了左侧子进程（leftpid）和右侧子进程（rightpid）
3. 左侧子进程负责将数据写入管道。它关闭了管道的读取端（pipefd[0]），将标准输出重定向到管道的写入端（pipefd[1]），然后使用execvp()执行了echo命令，将字符串"good morning"输出到标准输出
4. 右侧子进程负责从管道读取数据并进行处理。它关闭了管道的写入端（pipefd[1]），将标准输入重定向到管道的读取端（pipefd[0]），然后使用execvp()执行了sed命令，将输入中的"good"替换为"hi"并输出到标准输出
5. 父进程关闭了管道的两端，然后使用wait()等待两个子进程的结束

整个过程实现了将左侧子进程输出的数据通过管道传递给右侧子进程进行处理，并等待两个子进程都结束后才退出。

1. 解释函数 dup()、execvp() 的用法。

dup()是一个系统调用，用于复制文件描述符：

int dup(int oldfd);

dup()会复制参数oldfd所指向的文件描述符，并返回一个新的文件描述符。新的文件描述符与原文件描述符指向相同的文件表项，这意味着它们可以被用来进行相同的读写操作。

通常情况下，dup()会返回当前可用的最小文件描述符作为新的文件描述符，如果复制成功则返回新的文件描述符，否则返回-1，并设置相应的错误码。

dup()常用于在文件描述符重定向时，复制一个文件描述符到另一个可用的文件描述符上。

execvp()是一个库函数，用于执行指定的程序：

int execvp(const char \*file, char \*const argv[]);

execvp()会根据指定的文件名file来执行相应的程序，并使用argv参数传递命令行参数给被执行的程序。该函数会搜索PATH环境变量指定的路径，找到第一个匹配的可执行文件并执行。

argv参数是一个以空指针结尾的字符串数组，其中每个字符串代表一个命令行参数。数组的第一个元素通常是被执行的程序的名称，后续元素是传递给程序的命令行参数。

execvp()会取代当前的进程映像，将指定的程序加载到当前进程中，开始执行新的程序。如果函数执行成功，它不会返回，除非出现错误。

该函数的返回值为-1，如果执行失败，则设置相应的错误码。

execvp()常用于在程序中启动其他可执行文件，实现进程的替换。它允许在运行时指定程序名称和参数，并动态地执行其他程序。

1. 为代码红色部分添加注释。（直接在下面代码里添加。将注释刷为与本行一样的格式）

#include <unistd.h>

#include <stdio.h>

#include <errno.h>

#include <sys/types.h>

#include <stdlib.h>

int main(int argc, char\* argv[])

{

int pipefd[2]; // 管道文件描述符数组

pid\_t leftpid, rightpid; // 左子进程和右子进程的进程ID

char \*arg1[]={"echo","good morning",0}; // 左子进程的命令及参数

char \*arg2[]={"sed","s/good/hi/g",0}; // 右子进程的命令及参数

if(pipe(pipefd)){ // 创建管道

fprintf(stderr,"%s:pipe failed:%s\n",argv[0],strerror(errno));

exit(1);

};

// 创建左子进程

if((leftpid=fork())<0){ // 创建左子进程失败

fprintf(stderr,"%s:fork of left child failed:%s\n",argv[0],strerror(errno));

exit(1);

};

if(leftpid==0){ // 左子进程

close(pipefd[0]);

close(1);

dup(pipefd[1]); // 将管道的写端复制到文件描述符1（标准输出）

close(pipefd[1]);

execvp("echo",arg1 ); // 执行echo命令

}

// 创建右子进程

if((rightpid=fork())<0){ // 创建右子进程失败

fprintf(stderr,"%s:fork of right child failed:%s\n",argv[0],strerror(errno));

exit(1);

};

if(rightpid==0){ // 右子进程

close(pipefd[1]);

close(0);

dup(pipefd[0]);

close(pipefd[0]);

execvp("sed",arg2 ); // 执行sed命令

}

else{

close(pipefd[0]);

close(pipefd[1]);

wait(NULL); // 等待子进程结束

wait(NULL);

}

}

文本

中度可信度描述已自动生成

类似于：

图示

描述已自动生成

2.1 系统管理员的职责主要有哪些方面？

系统管理员主要工作：

1. 设置整个计算机系统，包括硬件和软件，如安装硬件设备安装操作系统和软件包，为用户建立账号等。
2. 做适当的备份和需要时的恢复。
3. 处理由于可供使用的计算机资源有限而遇到的问题。
4. 排除由于连接问题而造成的网络阻塞。
5. 进行操作系统的升级和维护。
6. 为用户提供常规支持。

2.2 如何查封一个用户账号？要使一个用户账号失效，应该怎么做？

查封一个用户账号：

1. 将用户记录从/etc/passwd文件中去掉，但是保留该用户的主目录和其他文件。
2. 或者在/etc/passwd（或/etc/shadow）文件中，在相关用户记录的passwd字段的首字符前加上符号“\*“
3. 将用户账号的shell设置成一个特定的信息的程序，以避免引起不必要的误会。

要使一个用户账号失效，可以选择删除或查封用户账号。查封用户账号步骤如上所示，删除一个用户账号如下所示：

使用userdel命令删除用户账号

userdel [-r] login # login表示用户账户

# 参数-r，可以将用户主目录及该目录下的文件删除，也会将该用户在系统中的其他一些文件删除，如该用户在/var/spool/mall的邮箱。

2.3 如何统计系统中磁盘空间的使用情况和空闲情况？

# 使用df命令：df命令可以显示文件系统的磁盘空间使用情况

df -h

# 使用du命令：du命令用于估算文件或目录的磁盘空间使用量

du -sh <目录路径>

2.4 在系统初启过程中，如何让系统自动启动某些程序，以及设置环境变量？

1. 自启动程序

对于Linux系统，可以将启动脚本或命令添加到/etc/rc.local文件中。在该文件中添加要启动的程序的命令，确保其具有适当的权限和路径

1. 设置环境变量

对于Linux系统，可以编辑用户的.bashrc或系统的/etc/environment文件来设置环境变量。将所需的环境变量添加到文件中，并在终端或系统启动时自动加载。

2.5 让一个用户拥有对某个组的资源的访问权限，应该怎么做？

# 如果该组尚未存在，可以使用以下命令创建组

sudo groupadd <group\_name>

# 添加用户到组：将目标用户添加到该组中

sudo usermod -a -G <group\_name> <username>

# 设置资源的访问权限：对于目标资源（文件、目录等）

## 使用chown命令将资源的所有者设置为目标组：

sudo chown :<group\_name> <resource\_path>

## 使用chmod命令设置资源的访问权限：

sudo chmod g+<permission> <resource\_path>

2.6 如何手工设置磁盘限额？

要手动设置磁盘限额，可以使用Linux中的Quota工具。Quota工具允许为特定用户或组设置磁盘配额，以限制他们使用的磁盘空间。

# 检查文件系统是否已启用磁盘配额：

sudo quotacheck -avug

# 启用磁盘配额

sudo quotaon -avug

# 为特定用户或组设置磁盘限额

## 设置用户限额：

sudo edquota -u <username>

## 设置组限额：

sudo edquota -g <groupname>

2.7 请参照fdisk命令输出的结果来计算所指定的硬盘容量请参照fdisk命令输出的结果来计算所指定的硬盘容量：

Disk /dev/hda: 255 heads, 63 sectors, 2480 cylinders

Units = cylinders of 16065 \* 512 bytes

磁头数（heads）= 255

扇区数（sectors）= 63

柱面数（cylinders）= 2480

每个扇区的字节数 = 512

=>

总扇区数 = 255 × 63 × 2480

总字节数 = 总扇区数 × 512 = 2,016,404,480,000 字节≈1.88 TB

=>

硬盘容量大概为1.88TB。

五、实验心得体会

在本次实验中，我了解了如何管理多用户多任务Linux系统，以及简单的制作文件系统。

要习惯性备份数据和系统，在出现问题时能挽回很大的损失，关键数据的权限不能乱给。在日常使用云服务器或本地虚拟机中的Linux系统时，可以用带有管理员权限的普通用户替代root用户使用，root用户权限很高，新手很容易出现差错。