《Linux 操作系统与程序设计》

实验指导书

计算机与信息学院

二〇一八年九月

目　　录

实验 1——操作系统基本命令使用

一、实验目的

1．通过对 gedit、vi、vim 文本编辑器的使用，掌握在 Linux 环境下文本文件的编辑方法；

2．通过对常用命令 mkdir、cp、cd、ls、mv、chmod、rm 等文件命令的操作，掌握 Linux 操作系统中文件命令的用法。

二、实验任务与要求

1．vi 或 vim 的使用，要求能新建、编辑、保存一个文本文件

2．gedit 的使用，要求能新建、编辑、保存一个文本文件

3．掌握 mkdir、cd 命令的操作，要求能建立目录、进入与退出目录

4．掌握 cp、ls、mv、chmod、rm 命令的操作，要求能拷贝文件、新建文件、查看文件、文件重命名、删除文件等操作。

三、实验工具与准备

计算机 PC 机，Ubuntu or Centos 操作系统

四、实验步骤与操作指导

任务 1．vi 或 vim 的使用，要求能新建、编辑、保存一个文本文件

（1）点击”应用程序”→ “附件”→“终端”，或快捷方式“ctrl + T”打开终端，在终端输入命令：

```
[root@localhost root]#vi kk.c
```

按 i 键，进入插入状态。

（2）输入以下 C 程序

```
#include<stdio.h>
int main( )
{
 printf(“Hello world!\n”);
 return 0;
}
```

此时可以用→、←、↑、↓键编辑文本。

（3）保存文件为 kk.c

按 Esc 键，进入最后行状态，在最后行状态输入：wq 保存文件，退出 vi。

（4）用 vi 打开文件 kk.c，输入命令：

```
[root@localhost root]#vi kk.c
```

（5）修改程序为：

```
#include<stdio.h>
int main( )
{

  printf("  Hello world!\n");
  printf("*****************\n");
  return 0;
}
```

（6） 按 Esc 键，进入最后行状态，在最后行状态输入：wq aa.txt 保存文件，如图 1 所示，另存为文件 aa.txt 并退出 vi。。

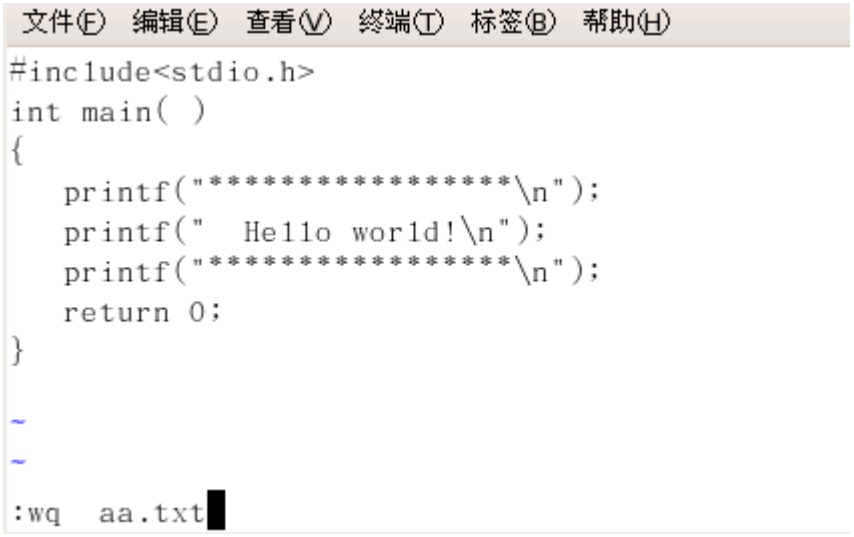

图 1 程序编辑环境

任务 2. gedit 的使用，要求能新建、编辑、保存一个文本文件

（1）启动 gedit，点击”应用程序”→ “附件”→“文本编辑器”，打开文本编辑器，或者终端输入 gedit。

（2）输入某个 C 程序

（3）保存文件为 kk.c

（4）用 gedit 打开文件 kk.c

（5）修改程序

（6）另存为文件 aa.txt 并退出。

任务 3. 掌握 mkdir、cd 命令的操作，要求能建立目录、进入与退出目录

（1）打开终端”应用程序”→ “附件”→“终端”，在终端用命令新建目录 kkk

```
[root@localhost root]#mkdir kkk
```

（2）进入目录 kkk，并在 kkk 目录下新建目录 kkka，进入 kkka 目录

```
[root@localhost root]#cd kkk
```

[root@localhost kkk]#mkdir kkka

[root@localhost kkk]#cd kkka

[root@localhost kkka]#

（3）执行命令 cd .. 命令，然后再进入 kkka 目录，输入命令 cd 、cd /etc，观察其结果。

[root@localhost kkka]#cd ..

[root@localhost kkk]#cd kkka

[root@localhost kkka]#cd

[root@localhost root]#cd /etc

[root@localhost etc]#

任务 4. 掌握 cp、ls、mv、chmod、rm 命令的基本操作，要求能拷贝文件、新建文件、查看文件的权限、修改文件以及删除文件。

（1）在 kkka 目录下建立文件 kk.c

[root@localhost root]#cd /root/kkk/kkka

[root@localhost kkka]#vi kk.c

（2）查看文件 kk.c 的属性

[root@localhost kkka]#ls -l kk.c

编辑 kk.c 文本，并用:wq 存盘。

（3）把 kk.c 更名为 aa.c

[root@localhost kkka]#mv kk.c aa.c

（4）修改文件 kk.c 的权限，使得文件所有者为可读、可写、可执行，对组内人及其他人可读、不可写、不可执行。

[root@localhost kkka]#cd ..

[root@localhost kkk]#chmod u=rwx,go=r kk.c

此时可用命令 ls 查看

[root@localhost kkk]#ls -l

（5）删除文件与文件夹

删除 kkka 文件夹下的文件 aa.c

[root@localhost kkk]#rm kkka/aa.c

查看文件夹 kkka 下否删除了文件 aa.c

[root@localhost kkk]ls -l kkka/aa.c

删除 kkka 文件夹下

[root@localhost kkk]#rmdir kkka

查看是否删除了文件夹

[root@localhost kkk]ls kkka -l

（6） 新建一个 linux_d 目录，并设置它的权限为 666。

（7）在指定的目录中搜索文件，利用 find 命令搜索含有通配符的文件*.c。

（8）设置当前的时间为 2017 年 10 月 01 日 10 点 23 分。

五、思考和意见

Linux 的 shell 命令让用户可以使用功能强大的命令，完成一些 windows 上无法完成或繁琐的操作，为以后程序的编写，系统管理提供了便利，我们应熟练掌握这些基础知识。

在 labs 目录下，用文本编辑器创建一个名字为 lab1 的文件，文件的内容为：“Use a text editor to create a file called lab1 under the labs directory in your directory hierarchy. The file should contain the text of this problem. ”。回答下列问题：

lab1 文件的类型，用 Linux 命令回答这个问题，给出会话过程，所有终端命令，并解释命令结果。

实验 2——SHELL 程序设计

一、实验目的

1．Shell 程序设计中变量的使用；

2．理解通道的概念并初步掌握它的使用方法；

3． 掌握算术操作、字符串操作、逻辑操作、文件操作；

4．掌握 if then fi、if then elif fi、case、while、for 等控制语句；

5．在 shell 脚本中使用函数；

二、实验任务与要求

1.观察变量$#,$0,$1,$2,$3,$@的含义

2.SHELL 程序设计中文件与文件夹的判断

3.顺序、分支、循环程序的设计

4.菜单程序的编写

三、实验工具与准备

计算机 PC 机，Ubuntu or Centos 操作系统

四、实验步骤与操作指导

任务 1．调试下列 shell 程序，写出变量$#,$0,$1,$2,$3,$@的含义。

```
#! /bin/bash
echo “程序名:$0”
echo “所有参数: $@”
echo “前三个参数:$1 $2 $3”
shift
echo “程序名:$0”
echo “所有参数: $@”
echo “前三个参数:$1 $2 $3”
shift 3
echo “程序名:$0”
echo “所有参数: $@”
echo “前三个参数:$1 $2 $3”
exit 0
```

修改程序，程序运行时从键盘输入文件名，判断文件是否存在，如果存在，显示文件内容。提示：

```
read  DORF
```

```
if [ -d $DORF ]
then
  ls $DORF
elif [ -f $DORF]
```

任务 2．调试下列 shell 程序，此程序的功能是：利用内部变量和位置参数编写一个名为 test2 的简单删除程序,如删除的文件名为 a，则在终端输入的命令为 test a。提示：除命令外至少还有一个位置参数，即$#不能为 0，删除的文件为$1。

(1)用 gedit 编辑程序

```
[root@localhost bin]#vi test2
#!/bin/sh
if test $# -eq 0
then
  echo "Please specify a file!"
else
  gzip $1 //先对文件进行压缩
  mv $1.gz $HOME/dustbin //移动到回收站
  echo "File $1 is deleted !"
fi
```

(2) 请修改程序，查看回收站中的文件，从键盘输入回收站中的某一文件，把此文件恢复到/home 目录下。

（3）删除垃圾箱中的所有文件。

任务 3．调试下列程序，程序的主要思想是用 while 循环求 1 到 100 的和。

(1)用 gedit 编辑脚本程序 test12

```
[root@localhost bin]#gedit test12
 total=0
 num=0
 while((num<=100));do
    total=`expr $total + $num
   ((num+=1))
 done
 echo "The result is $total"
```

（2）用 for 语句完成以上求和。

任务 4．调度下列程序，使用 shell 编写一个菜单，分别实现列出以下内容：（1）目录内容、（2）切换目录、（3）创建文件、（4）编辑文件、（5）删除文件的功能。在此例中将用

到循环语句 until、分支语句 case、输入输出语句 read 和 echo。

```
#! /bin/bash
until
   echo  "(1)List  you  selected  directory"
   echo  "(2)Change  to  you  selected  directory"
   echo  "(3)Creat  a  new  file"
   echo  "(4)Edit  you  selected  file"
   echo  "(5)Remove  you  selected  file"
   echo  "(6)Exit  Menu"
   read  input
   if test  $input  =  6  then
 exit 0
   fi
do
   case  $input  in
     1) ls;;
     2) echo -n "Enter  target  directory:"
        read  dir
        cd  $dir
        ;;
     3) echo -n "Enter  a  file  name:"
        read  file
        touch  $file
        ;;
     4) echo -n "Enter  a  file  name:"
        read  file
        vi  $file
        ;;
     5) echo -n  "Enter  a  file  name:"
        read  file
        rm  $file
        ;;
     *) echo  "Please  selected  1\2\3\4\5\6 " ;;
   esac
done
```

（2）修改以上程序，用菜单形式完成算术四则混合运算。

五、思考和意见

1、编写一个bash脚本程序，用for循环实现将当前目录下的所有.c文件移到指定的目录下，最后在显示器上显示指定目录下的文件和目录

实验 3——Linux 环境系统函数的应用

一、实验目的

1．掌握随机数函数的使用方法；

2． 掌握结构体 struct timeval 的成员 tv_sec 与 tv_usec 的应用；

3． 掌握时间函数 time、localtime、gettimeofday 的使用；

4. 掌握系统函数 system、tcgetattr 等的应用。

二、实验任务与要求

1. 随机数函数的使用；

2. 猜数游戏的程序；

3. 时间函数在简单记事本程序中的应用；

三、**实验工具与准备**

计算机 PC 机，Ubuntu or Centos 操作系统。

四、实验步骤与操作指导

任务 1.调试下列程序。产生 10 个介于 1 到 10 之间的随机数值。提示函数 rand( )会返回一个 0~ RAND_MAX（其值为 2147483647）之间的随机值。产生一个大于等于 0、小于 1 的数，此数可表示为：rand()/(RAND_MAX+1.0)。

```
[root@localhost root]#vim  5-1.c
```

程序代码如下：

```
#include <stdlib.h>
#include "stdio.h"
int main()
{
 int i,j;
 srand((int)time(0));
 for(i=0;i<10;i++)
 {
  j=1+(int)(10.0*rand()/(RAND_MAX+1.0));
  printf(" %d ",j);
 }
 printf(“\n”);
}
```

问题

（1）论述语句 srand((int)time(0));的功能；

（2）修改程序，产生 50 个 100-1000 之间的随机整数；

任务 2．程序设计。编写一个猜数游戏的程序，先产生一个随机数，要求被试输入一个数，计算机会提示猜大了，猜小了或恭喜您猜中了，直到猜中，退出程序。修改程序，限定猜数的次数作为难度系数，除了提示猜大了，猜小了或恭喜您猜中了外，还有次数已到，猜数失败。

任务3. 编写程序。编写一个简单的程序，其功能是实现输入任务及任务截止时间，通过系统时间和日期函数的使用，可计算输出任务的剩余时间。

提示：

printf("请输入任务截止时间，分别输入日期1~31，小时0~23、分、秒对应的数字\n");

scanf("%d%d%d%d", &i,&j,&k,&l);

为对应的输入时间，即余下时间可以从以下语句中得到。

```
time (&timep);
p=localtime(&timep);
a=(i-p->tm_mday)*24*3600+(      j-p->tm_hour)*3600+      (k-p->tm_min)*60+
l-p->tm_sec;
```

在余下时间为0时可以考虑响铃及显示任务。

五、思考和意见

1. 编写一个程序，求2～n间的素数，n由键盘输入，循环变量分别从2到n、2到(int)sqrt(n)，分别测出两个循环的所用时间。

实验 4——Linux 文件 I/O 操作

一、实验目的

1．掌握函数 stat 中文件属性的应用。

2．掌握系统函数 system、opendir、scandir 的使用。

3．掌握文件阻塞与非阻塞 I/O 的操作。

4. 掌握文件属性的判断。

二、实验任务与要求

1.测试文件S_IRUSR、S_IWUSR、S_IRGRP、S_IROTH属性。

2.应用readdir函数显示文件和子目录。

3.文件属性的判断。

4.阻塞 I/O 文件操作的程序设计。

5.掌握文件目录与文件的递归及深度遍历；

三、实验工具与准备

计算机 PC 机，Ubuntu or Centos 操作系统

四、实验步骤与操作指导

任务 1.程序设计。设计程序应用 system 函数在桌面建立文件夹 test，再新建一个文件 test，应用 chmod 函数使文件 test 具有 S_IRUSR、S_IWUSR、S_IRGRP、S_IROTH 属性，最后应用函数 stat 获取文件的大小与建立的时间。

任务 2.调试并分析下列程序的结果。程序的功能是用递归的方法列出某一目录下的全部文件的大小和文件夹及创建日期，包括子文件和子文件夹。程序代码如下：

```
#include<stdio.h>
#include<time.h>
#include<linux/types.h>
#include<dirent.h>
#include<sys/stat.h>
#include<unistd.h>
#include<string.h>
char *wday[]={"日","一","二","三","四","五","六"};
void list(char *name,int suojin)
{
   DIR *dirname;
   struct dirent *content;
```

```
    struct stat sb;
    struct tm *ctime;
    int i;
    if((dirname=opendir(name))==NULL)
    {
     printf("该目录不存在\n");
         return;
    }
    chdir(name);/*改换工作目录*/
    while((content=readdir(dirname))!=NULL)
    {


     for(i=0;i<suojin;i++)
         putchar('\t');
     if(content->d_type==4)
         printf("目录\t");
     else if(content->d_type==8)
         printf("文件\t");
     else
        printf("其他\t");
     stat(content->d_name,&sb);
     ctime=gmtime(&sb.st_mtime);
  printf("%d年%d月%d日 星期%s %d:%d:%d\t",ctime->tm_year+1900,
 1+ctime->tm_mon,ctime->tm_mday,wday[ctime->tm_wday],ctime->tm_hour,
ctime->tm_min,ctime->tm_sec);
 printf("%d\t",sb.st_size);
 printf("%s\n",content->d_name);/*列出目录或文件的相关信息*/
 if(content->d_type==4&&strcmp(content->d_name,"..")&&strcmp(content->d_na
me,"."))
{
     list(content->d_name,suojin+1);/*如果是目录，则递归列出目录里的内容*/
      }
}
    closedir(dirname);
```

```
    chdir("..");/*当该层目录中的文件列完后，返回父目录*/
}



int main(int argc,char *argv[])
{
  char name[256];
  printf("类型\t 最后修改时间\t\t\t 大小\t 文件名\n");
  printf("*****************************************************\n");
  if(argc==1) {
     printf("Enter directory name:");
     scanf("%s",name);
     list(name,0);
    }
  else
  {
     list(argv[1],0);
    }
}
```

五、思考和意见

利用函数 fopen(),fread(),fwrite(),fclose()来实现简单的文件备份（即将一个文件的内容拷贝到另一个文件中去）。被复制文件需要存在，并且含有内容。

分析提示：在 linux 系统中，文件和设备都被看作是数据流。进行操作之前，必须先将流打开。可以通过调用库函数 fopen()打开一个流,库函数 fopen()的返回值为一个 FILE 结构指针，此结构中包含对所打开流进行操作所需的全部信息。

当一个流被打开后，就可以对其执行 I/O 操作了。当一个流操作完成后，需要执行清空缓冲区、保存数据等操作。

将流关闭，可以通过调用函数 fclose()来完成。

实验 5——Linux 进程控制

一、实验目的

1. 掌握进程的常用终端命令；
2. 掌握用 system、exec 函数簇、fork 函数创建进程；
3. 掌握 waitpid 函数的应用；
4. 掌握守护进程过程；
5. 掌握守护进程在各种监控中的应用。

二、实验内容

1. 进程的常用终端命令；
2. 用 execl 函数创造进程；
3. 应用 fork 函数创建子进程；
4. 应用 fork 函数创建子进程,在父子进程中执行不同的任务；
5. waitpid 函数的应用；
6. 守护进程的编写与应用；

三、实验设备与实验准备

计算机 PC 机，Ubuntu or Centos 操作系统。

四、实验步骤与操作指导

任务 1

学习 at 指令的使用。写出 at 指令的使用格式。在 2018 年 12 月 8 日，通过 at 指令运行命令"ls -l"

任务 2：程序设计。用 execl 函数创造进程 ls -l，用 execvp 函数创造进程 ps -ef。提示：

显示当前目录下的文件信息可执行以下语句：

(1) execl("/bin/ls", "ls", "-al", NULL) ;

(2) char *arg[] = {"ps", "-ef", NULL};

execvp("ps", arg);

任务 3：程序设计。（1）在父子进程中分别编写循环程序，应用函数 sleep 的不同参数等，体现程序中父子进程的并发运行。（2）在父子进程中分别执行不同的任务，例如在子进程中执行查看当前目录详细信息文件信息的功能，在父进程中执行打印子进程 pid 号等，子进程退出后父进程才退出。

任务 4 用C语言设计一个闹钟程序，用户输入时间，格式为小时:分钟，例如 9:18 表示设定的时间为 9 时 18 分，到了设定时间后，发出蜂鸣声作为提示音，为保证检测时间的准确性，要求使用守护进程。

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <signal.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/stat.h>
void init_daemon(void);
int main()
{
   int hour,min;
 time_t timep;
   struct tm *p;
   time (&timep);
   p=localtime(&timep);/*获取系统时间*/
 printf("这是一个闹钟程序,输入你想要设定的时间:\n");
 scanf("%d:%d",&hour,&min);
 init_daemon(); /*调用守护进程*/
 while(1)
 {
     sleep(20);/*每隔 20 秒检查一下时间是否已到*/
     if(p->tm_hour==hour &&p->tm_min==min)
     {
         printf("时间到了!\n");
         printf("\7\7\7\7\7");/*到了发出 5 声蜂鸣，作为提示*/
     }

 }
}
void init_daemon(void)/*这是守护进程*/
{
 pid_t child1,child2;
```

```
int i;
child1=fork();
if(child1>0)
    exit(0);
else
    if(child1< 0)
    {
        perror("创建子进程失败");
        exit(1);
    }
    setsid();
    umask(0);
    for(i=0;i< NOFILE;++i)
        close(i);
    return;
}
```

五、思考和意见

改写以下代码，在子程序中用函数 system 启动一个较长时间运行的任务，而在父进程中执行完成任务后，应用 waitpid 函数等待子进程，子进程退出后父进程才退出。

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main ()
{
pid_t pid,wpid;
int status,i;
pid=fork();
if(pid==0)
{
 printf("这是子进程,进程号(pid)是:%d\n",getpid());
 sleep(5);            /*子进程等待 5 秒钟*/
 exit(6);
}
else
{
```

```
 printf("这是父进程,正在等待子进程……\n");
 wpid=wait(&status);  /*父进程调用 wait 函数，消除僵尸进程*/
 i=WEXITSTATUS(status);
 printf("等待的进程的进程号(pid)是:%d ,结束状态:%d\n",wpid,i);
   }
}
```

实验 6——Linux 进程通信

一、实验目的

1.掌握常用的几种硬中断方法；

2.了解常用的软中断;

3.掌握 signal 函数实现信号处理程序设计；

4.掌握应用管道实现信号处理的方法。

5．掌握无名管道与命名管道进行通信；

二、实验任务与要求

1.Ctrl+C 硬中断;

2.alarm 函数产生的 SIGALRM 信号;

3.应用 signal 函数实现信号处理程序编写;

4.应用管道实现信号处理的编写;

5.调用系统函数 kill 对进程的处理。

6. 管道读写程序的编写与应用；

7. 父子进程通过内存映射实现数据共享；

三、实验工具与准备

计算机 PC 机，Ubuntu or Centos 操作系统

四、实验步骤与操作指导

任务1：硬中断实例

运行下列程序 kk.c：

```
#include <unistd.h>
int main(void)
{
    while(1);
    return 0;
}
```

（1)程序运行过程中，请你使用硬中断 Ctrl+C 或 Ctrl-\中断程序的执行。

（2)可以使用信号 SIGSEGV 中断此程序，方法是先在后台运行此程序，得出程序进程号，然后用命令 kill 发送信号 SIGSEGV，如下形式：

```
$ ./kk   &   [1] 3940
$ kill -SIGSEGV 3940
```

$（再次回车）

任务 2：使用软件中断。利用 Linux 的软中断信号，编写一段 C 语言程序完成：显示数字 1-100，在程序运行中如果捕获到一个 SIGINT 信号，则转去执行输出“You enter CTRL+C”。

任务 3： 构建一个共享内存通信的实例，父进程将某一条消息写入共享内存，子进程从共享内存中读取消息。

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
#include<sys/types.h>   /*文件预处理，包含 waitpid、kill、raise 等函数库*/
#include<unistd.h>              /*文件预处理，包含进程控制函数库*/
#include <sys/mman.h>
#include <fcntl.h>
typedef struct        /*结构体，定义一个 people 数据结构*/
{
char name[4];
int   age;
}people;
main(int argc, char** argv)   /*C 程序的主函数，开始入口*/
{
pid_t result;
int i;
people *p_map;
char temp;
p_map=(people*)mmap(NULL,sizeof(people)*10,PROT_READ|PROT_WRITE,
MAP_SHARED|MAP_ANONYMOUS,-1,0);     /*调用 mmap 函数，匿名内存映射*/
result=fork();                /*调用 fork 函数，复制进程,返回值存在变量 result 中*/
if(result<0) /*通过 result 的值来判断 fork 函数的返回情况，这儿进行出错处理*/
{
perror("创建子进程失败");
exit(0);
}
else if (result==0)   /*返回值为 0 代表子进程*/
{
sleep(2);
```

```
for(i = 0;i<5;i++)
printf("子进程读取: 第 %d 个人的年龄是： %d\n",i+1,(*(p_map+i)).age);
(*p_map).age = 110;
munmap(p_map,sizeof(people)*10); /*解除内存映射关系*/
exit(0);
}
else                                    /*返回值大于 0 代表父进程*/
{
temp = 'a';
for(i = 0;i<5;i++)
{
temp += 1;
memcpy((*(p_map+i)).name, &temp,2);
(*(p_map+i)).age=20+i;
}
sleep(5);
printf( "父进程读取: 五个人的年龄和是： %d\n",(*p_map).age );
printf("解除内存映射……\n");
munmap(p_map,sizeof(people)*10);
printf("解除内存映射成功！\n");
}
}
```

任务 4 程序设计。有名管道程序设计：创建两个进程，在 A 进程中创建一个有名管道，并向其写入数据， 通过 B 进程从有名管道中读出数据。

五、思考和意见

设计一个程序，要求创建一个管道 PIPE，应用函数 fork 复制子进程，在父进程中运行命令”ls -l”，把某一文件名的信息写入管道，子进程从管道中读取这些文件信息。

实验 7——Linux 线程管理

一、实验目的

1. 了解 Linux 下多线程程序设计的基本原理；

2. 学习 pthread 库函数的使用。

二、实验任务与要求

1. 编写 thread 应用程序；

2. 创建两个线程分别完成不同的功能；

三、**实验工具与准备**

计算机 PC 机，Ubuntu or Centos 操作系统

四、实验步骤与操作指导

任务1：简单的多线程编程

Linux 系统下的多线程遵循 POSIX 线程接口，称为 pthread。编写 Linux 下的多线程程序，需要使用头文件 pthread.h，连接时需要使用库 libpthread.a。编写多线程程序 example.c，说明线程创建函数 pthread_create 和 pthread_join 的功能。

```
/* example.c*/
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
 void thread(void)
{
int i;
for(i=0;i<3;i++)
 printf("This is a pthread.\n");
}
int main(void)
 {
 pthread_t id;
 int i,ret;
ret=pthread_create(&id,NULL,(void *) thread,NULL);
if(ret!=0)
{
printf ("Create pthread error!\n");
```

```
exit (1);
}
for(i=0;i<3;i++)
  printf("This is the main process.\n");
pthread_join(id,NULL);
return (0);
}
```

编译此程序得到结果

任务2：创建两个线程，定义两个变量并赋值，一个线程计算两个变量的加法，另一个线程计算两个变量的减法。

任务3：创建两个子进程，一个子线程（生产者线程）依次向缓冲区写入整数0,1,2,...,19；另一个子线程（消费者线程）暂停3s 后，从缓冲区读数，每次读一个，并将读出的数字从缓冲区删除，然后将数字显示出来；父线程等待子线程2（消费者线程）的退出信息，待收到该信息后，父线程就返回。

五、思考和意见

有两个线程 T1 和 T2 动作描述如下，x、y、z 为两个线程共享变量。信号量 S1 和 S2 的初值均为 0，编写程序完成下面两个线程 T1、T2 并发执行。不断调整 sleep(n)的 n 值，多次运行程序后，观察 x、y、z 输出的值，各为多少？

```
线程 T1：                    线程 T2：
 y=1;                          x=1;
 y=y+2;                       x=x+1;
 sem_post(&S1);          sem_wait(&S1);
 z=y+1;              x=x+y;
sleep(n)                      sleep(n)
sem_wait(&S2);          sem_post(&S2);
y=z+y;                        z=x+z;
```

实验 8——Linux 网络通信

一、实验目的

1. 掌握基于“TCP 套接字”编程；
2. 掌握基于“UDP 套接字”编程;

二、实验任务与要求

1. 编写 TCP 客户端和服务器端程序
2. 编写 UDP 客户端和服务器程序（简单聊天程序）

三、实验工具与准备

计算机 PC 机，Ubuntu or Centos 操作系统

四、实验步骤与操作指导

任务 1：编写 TCP 客户端和服务器端程序

编写服务器可客户端程序，实现如下功能： 服务器等待接收客户的连接请求，一旦连接成功则显示客户地址，接着接收客户端的名称并显示；然后接收来自该客户的字符串，每当接收到一个字符串时，显示该字符串，并将字符串变换大小写进行加密，再将加密后的字符串发回客户端；之后，继续等待接收该客户的信息，直到客户端关闭连接。客户首先与相应等的服务器建立连接，接收接收用户输入的客户端名称，并将其发送给服务器；然后继续接收用户输入的字符串，再将字符串发送给服务器，同时接收服务器发回的加密后的字符串并显示。之后，继续等待用户输入字符串，直到用户输入 CTRL+D，客户关闭连接并退出。

```
/* 9-2-c.c */

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <netinet/in.h>

#define MAXLINE 80

#define SERV_PORT 8000
```

```
int main(int argc, char *argv[])
{
  struct sockaddr_in servaddr;
  char buf[MAXLINE];
  int sockfd, n;
  char *str;
  if (argc != 2) {
        fputs("usage: ./client message\n", stderr);
        exit(1);
  }
  str = argv[1];
  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  bzero(&servaddr, sizeof(servaddr));
  servaddr.sin_family = AF_INET;
  inet_pton(AF_INET, "127.0.0.1", &servaddr.sin_addr);
  servaddr.sin_port = htons(SERV_PORT);
  connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));
  write(sockfd, str, strlen(str));
  n = read(sockfd, buf, MAXLINE);
  printf("Response from server:\n");
  write(STDOUT_FILENO, buf, n);
  close(sockfd);
```

```
    return 0;
}
    /* 9-2-s.c */
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include <unistd.h>
    #include <sys/socket.h>
    #include <netinet/in.h>
    #define MAXLINE 80
    #define SERV_PORT 8000
    int main(void)
    {
      struct sockaddr_in servaddr, cliaddr;
      socklen_t cliaddr_len;
      int listenfd, connfd;
      char buf[MAXLINE];
      char str[INET_ADDRSTRLEN];
      int i, n;

      listenfd = socket(AF_INET, SOCK_STREAM, 0);
      bzero(&servaddr, sizeof(servaddr));
```

```
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(SERV_PORT);
        bind(listenfd, (struct sockaddr *)&servaddr, sizeof(servaddr));
        listen(listenfd, 20);
        printf("Accepting connections ...\n");
        while (1) {
            cliaddr_len = sizeof(cliaddr);
            connfd = accept(listenfd,
                    (struct sockaddr *)&cliaddr, &cliaddr_len);
            n = read(connfd, buf, MAXLINE);
            printf("received from %s at PORT %d\n",inet_ntop(AF_INET,
  &cliaddr.sin_addr,
        str, sizeof(str)),ntohs(cliaddr.sin_port));
              for (i = 0; i < n; i++)
                  buf[i] = toupper(buf[i]);
            write(connfd, buf, n);
            close(connfd);
        }
    }
```

任务 2：自行实现一个面向非连接的 UDP 编程，功能不限。

五、思考和意见

编写一个基于 TCP 协议的网络通信程序，要求服务器通过 socket 连接后，并要求输入用户，判断为 liu 时，才向客户端发送字符串“Hello，you are connected!”。在服务器上显示客户端的 IP 地址和端口号。