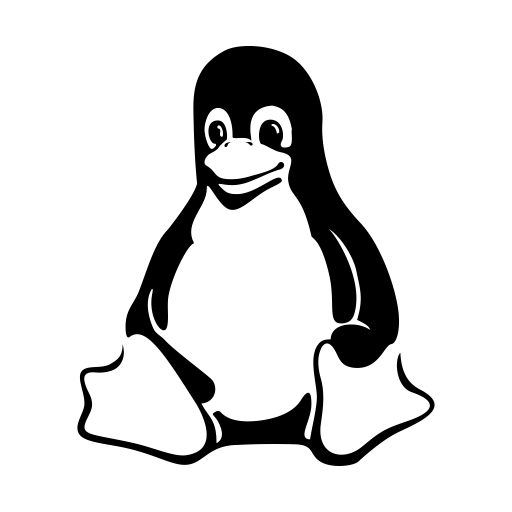
Linux高级程序设计

实验作业7



学号：0x091395

姓名：uknowho

年级：2014

班级：软件工程（3）班

# 思考与实验：

1. 设计一个程序，要求程序运行后进入一个无限循环，当用户按下中断键(Ctrl+Z)时，进入程序的自定义信号处理函数，当用户再次按下中断键(Ctrl+Z)后，结束程序运行。

#include<stdio.h> //printf

#include<stdlib.h> //exit

#include<signal.h> //signal

#include<unistd.h> //sleep

void myfunc()

{

printf("已进入自定义信号处理.\n");

printf("自定义信号处理结束...\n");

printf("请再次按下CRTL-Z退出程序...\n");

(void) signal(SIGTSTP, SIG\_DFL);

}

int main()

{

printf("正在每隔1秒打印一个\"hello\"中，等待CRTL-Z中断..\n");

(void) signal(SIGTSTP, myfunc); //使用自定义函数myfunc

while(1)

{

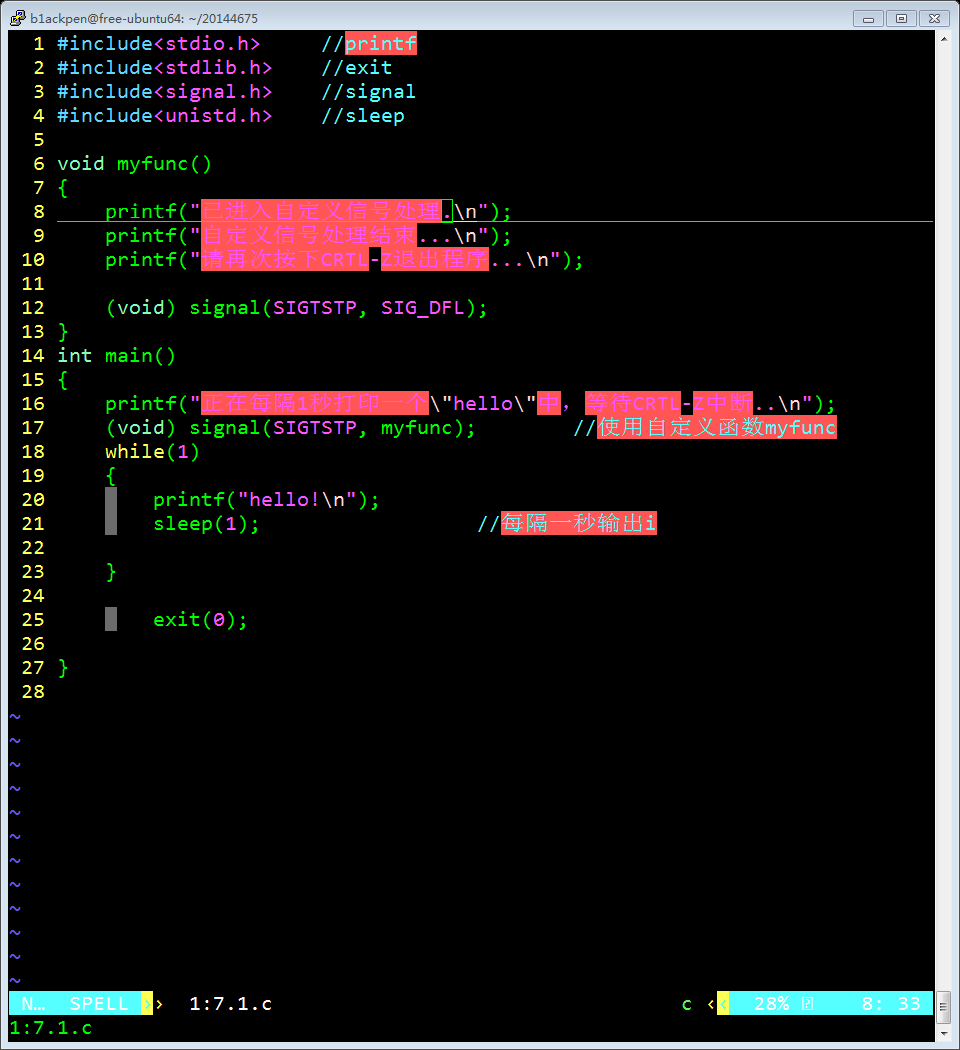
printf("hello!\n");

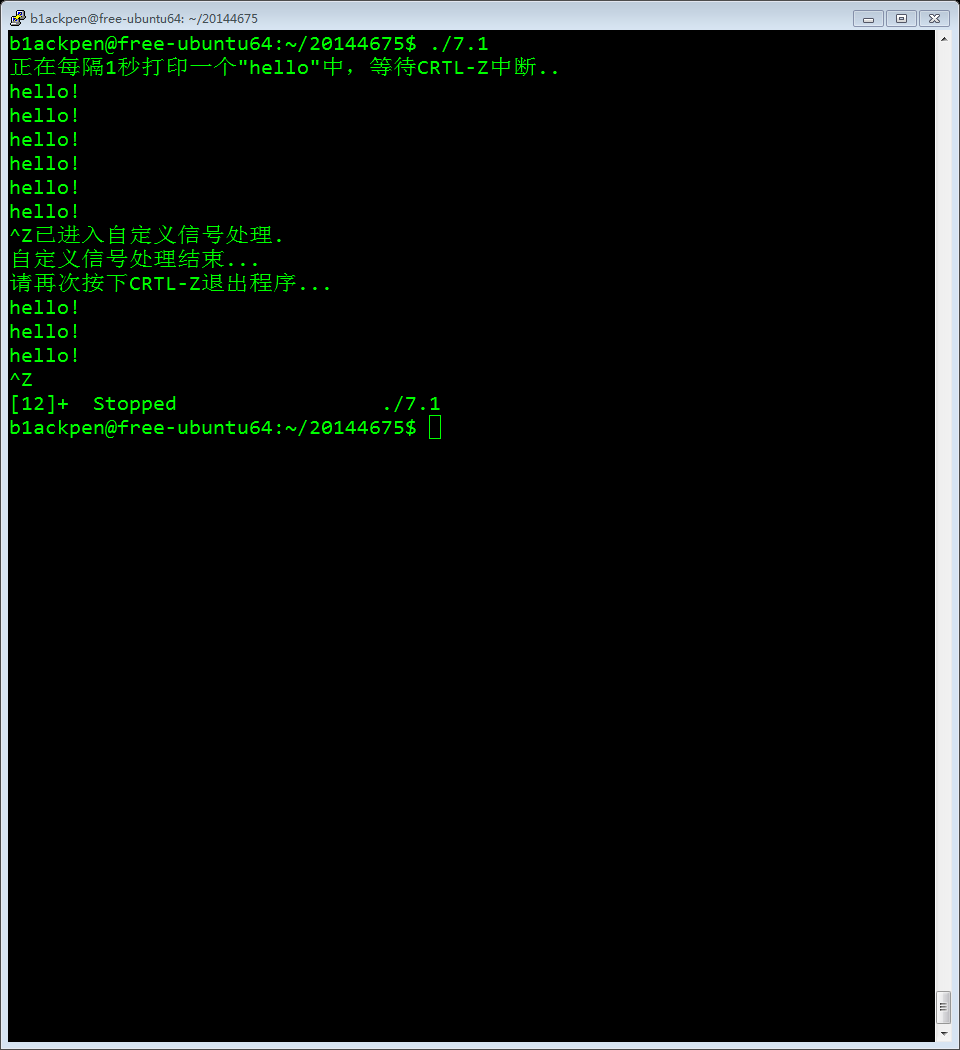
sleep(1); //每隔一秒输出i

}

exit(0);

}





1. 设计一个程序，要求程序主体运行时，即使用户按下的中断键(Ctrl+C)，也不能影响正在运行的程序，等程序主体运行完毕后才进入自定义信号处理函数。

#include<signal.h>

#include<stdlib.h>

#include<stdio.h>

#include<unistd.h>

void myfunc();

int main()

{

int i;

sigset\_t set, pendset;

struct sigaction action;

(void) signal(SIGINT, myfunc);

if(sigemptyset(&set)<0)

perror("Initial signal set failed.");

if(sigaddset(&set, SIGINT)<0)

perror("Add signal set failed");

if(sigprocmask(SIG\_BLOCK, &set, NULL)<0)

perror("Adding signal into signal set failed");

else

{

for(i=0; i<5; i++)

{

printf("Doing something....\n");

sleep(2);

}

}

if(sigprocmask(SIG\_UNBLOCK, &set, NULL)<0)

perror("Unlock signal from signal set failed");

return 0;

}

void myfunc()

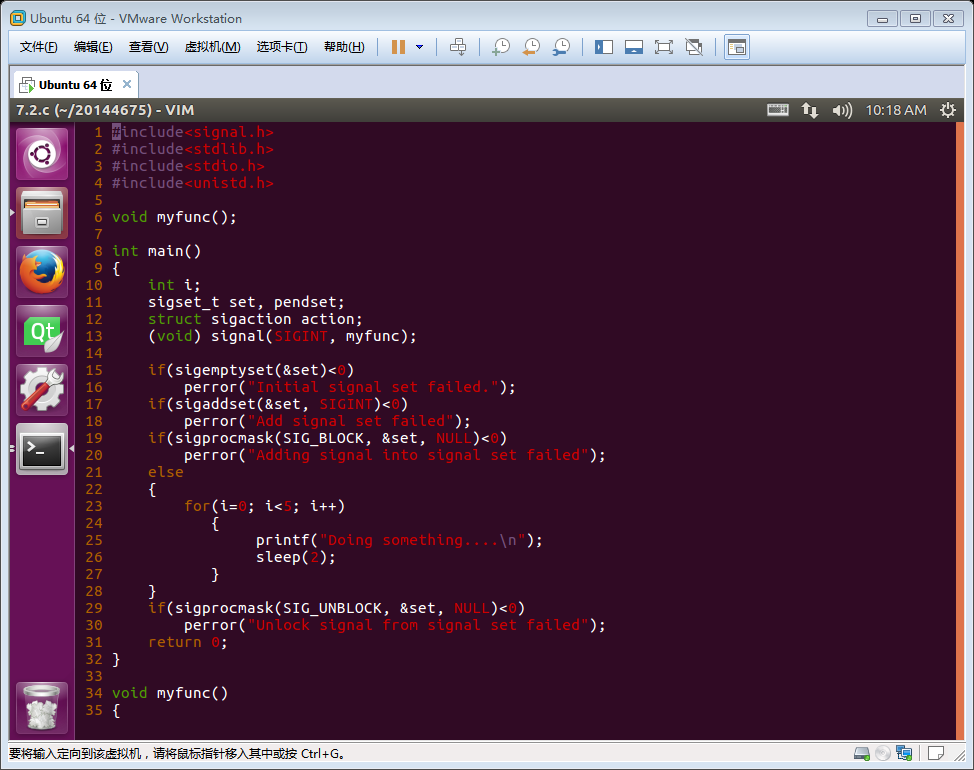
{

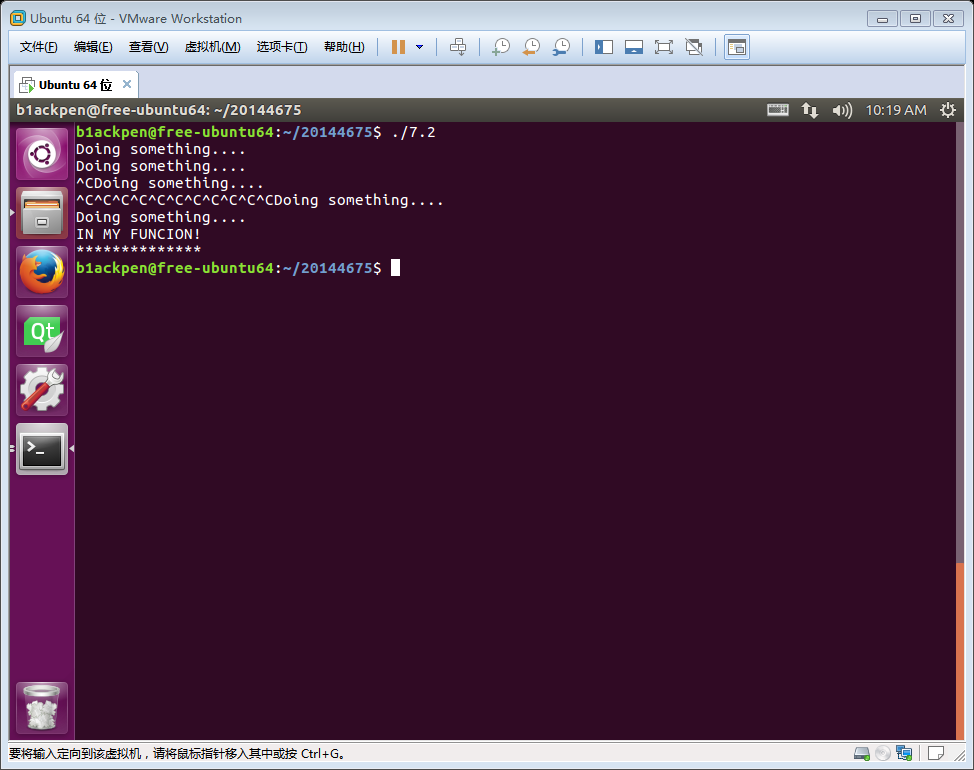
printf("IN MY FUNCION!\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

(void) signal(SIGINT, SIG\_DFL);

}





1. 设计一个程序，要求创建一个管道PIPE，复制进程，父进程运行命令“ls -l”，把运行结果写入管道，子进程从管道中读取“ls -l”的结果，把读出的作为输入接着运行“grep 7-5”。

#include<stdio.h>

#include<unistd.h>

#include<memory.h>

int main()

{

FILE \*fp;

int num;

char buf[500];

memset(buf, 0, sizeof(buf));

printf("建立管道...");

fp=popen("ls -l", "r");

if(fp!=NULL)

{

num=fread(buf, sizeof(char), 500, fp);

if(num>0)

{

printf("ls -l 结果如下：...\n");

printf("%s\n", buf);

}

pclose(fp);

}

else

{

printf("popen error!");

return 1;

}

fp=popen("grep 7-3.c", "w");

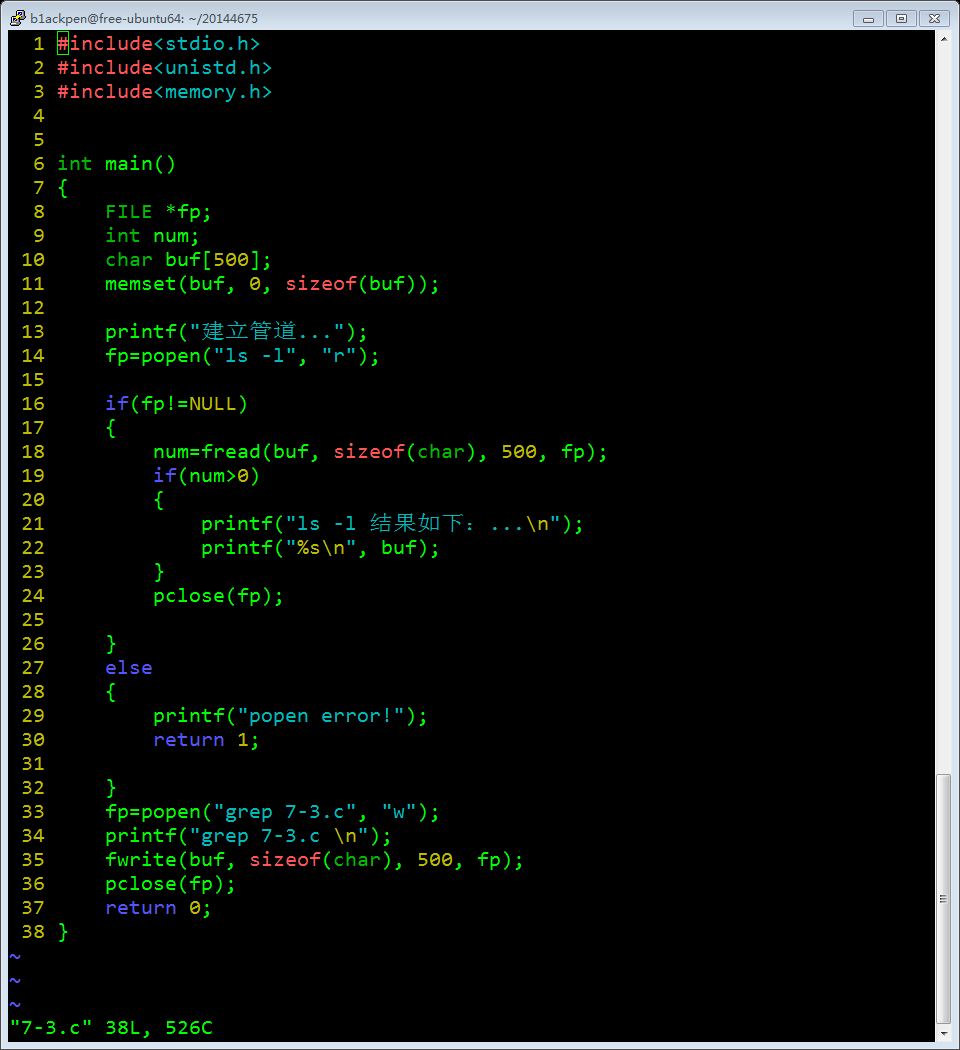
printf("grep 7-3.c \n");

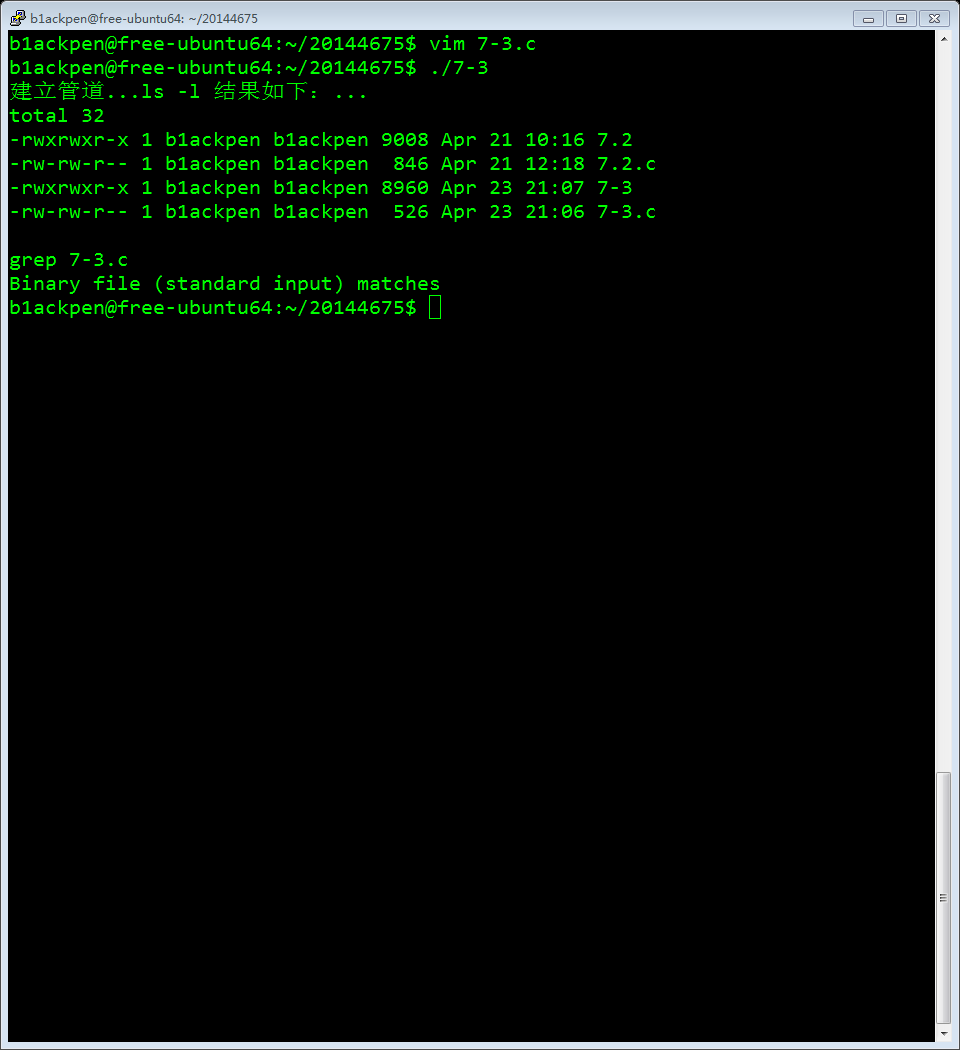
fwrite(buf, sizeof(char), 500, fp);

pclose(fp);

return 0;

}





1. 设计两个程序，要求用命名管道FIFO，实现简单的文本文件或图片文件的传输功能。

//文本传输 rc\_text.c 和sd\_text.c内容一样。

#include<stdio.h>

#include<string.h>

#include<sys/type.h>

#include<sys/stat.h>

int main()

{

itn i, rfd, wfd, len=0, fd\_in;

char str[32];

int flag, stdinflag;

fd\_set write\_fd, read\_fd;

struct timeval net\_timer;

mkfifo("fifo1", S\_IWUSR|S\_IRUSR|S\_IRGRP|S\_IROTH);

mkfifo("fifo2", S\_IWUSR|S\_IRUSR|S\_IRGRP|S\_IROTH);

wfd=open("fifo1", O\_WRONLY);

rdf=open("fifo2", O\_WRONLY);

if(rfd<=0 || wrf<=0)

{

return 0;

}

printf("MESSAGE SENDING! HELLO!\n");

while(1)

{

FD\_ZERO(&read\_fd);

FD\_SET(rfd, &read\_fd);

FD\_SET(fileno(stdin), &read\_fd);

net\_timer.tv\_sec=5;

net.timer.tv\_usec=0;

memset(str, 0, sizeof(str));

if(i=select(rfd+1, &read\_fd, NULL, NULL, &net\_timer)<=0)

{

continue;

}

if(FD\_ISSET(rfd, &read\_fd))

{

read(rfd, str, sizeof(str));

printf("--------------------\n");

printf("TERMINAL 1:%s\n", str);

}

if(FD\_ISSETfileno(stdin), &read\_fd)

{

printf("--------------------\n");

fgets(str, sizeof(str), stdin);

len=write(wfd, str, strlen(str));

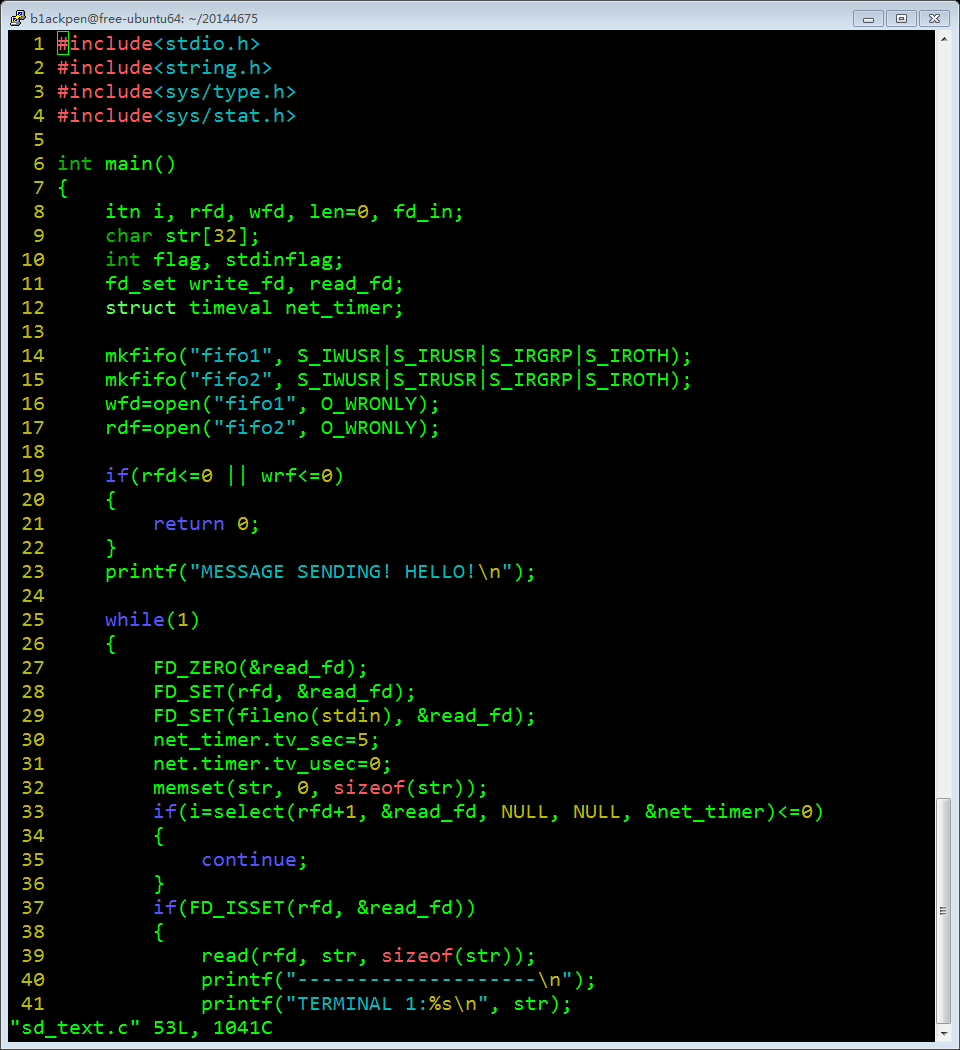
}

}

close(rfd);

close(wfd);

}



1. 设计两个程序，要求用消息队列，实现聊天程序，每次发言后自动在后面增加当前系统时间。增加结束字符，比如最后输入“88”后结束进程。

**//a.c**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

#include <time.h>

struct msgmbuf /\*定义消息的结构体\*/

{

long msg\_type; /\*消息类型\*/

int msg\_date; /\*添加系统时间\*/

char msg\_text[1024]; /\*消息内容\*/

};

int main()

{

int ret;

int qid;

key\_t key;

struct msgmbuf msg;

msg.msg\_type=100; /\*定义写入消息类型为100\*/

key=ftok(".",'a'); /\*产生标准key\*/

if(key==-1)

{

perror("产生标准key错误!\n");

exit(1);

}

qid=msgget(key,IPC\_CREAT|0666); /\*创建消息队列返回值为qid\*/

if(qid==-1)

{

perror("创建消息队列失败!");

exit(1);

}

while(1)

{

/\*从键盘获取发送内容，并利用msgsnd添加消息，ret判断添加消息是否成功\*/

printf("请输入发送内容:\n");

scanf("%s",&msg.msg\_text);

msg.msg\_date=system("date|cut -d' ' -f 5"); /\*添加系统时间\*/

ret=msgsnd(qid,&msg,sizeof(msg.msg\_text),msg.msg\_type);

if(ret<0)

{

perror("添加消息失败!");

exit(1);

}

/\*定义读取消息类型为200利用msgrcv读取消息，并添加到标准输出\*/

msg.msg\_type=200;

msgrcv(qid,&msg,sizeof(msg),msg.msg\_type,0);

if(ret<0)

{

perror("读取消息失败!");

exit(1);

}

printf("读取的内容为:\n%s\n",msg.msg\_text);

}

msgctl(qid,IPC\_RMID,NULL); /\*关闭消息队列\*/

return 0;

}

**//b.c**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

struct msgmbuf /\*定义消息结构体\*/

{

int msg\_type; /\*消息类型\*/

int msg\_date; /\*消息发送时间\*/

char msg\_text[1024]; /\*消息内容\*/

};

int main()

{

int ret;

int qid;

key\_t key;

struct msgmbuf msg;

msg.msg\_type=100; /\*读取消息类型为100\*/

key=ftok(".",'a'); /\*产生标准key\*/

if(key==-1)

{

perror("产生标准key错误!\n");

exit(1);

}

qid=msgget(key,IPC\_CREAT|0666); /\*创建消息队列\*/

if(qid==-1)

{

perror("创建消息队列失败!");

exit(1);

}

while(1)

{

/\*先利用msgrcv读取demo5-1发送的消息，并添加到标准输出\*/

ret=msgrcv(qid,&msg,sizeof(msg),msg.msg\_type,0);

if(ret==-1)

{

perror("读取消息失败!");

exit(1);

}

printf("读取的内容为:\n%s\n",msg.msg\_text);

/\*定义写入消息类型为200，并从键盘获取发送内容，利用msgsnd添加消息\*/

msg.msg\_type=200;

printf("请输入发送内容:\n");

scanf("%s",&msg.msg\_text);

msg.msg\_date=system("date|cut -d' ' -f 5");

ret=msgsnd(qid,&msg,sizeof(msg.msg\_text),msg.msg\_type);

if(ret==-1)

{

perror("添加消息失败!");

exit(1);

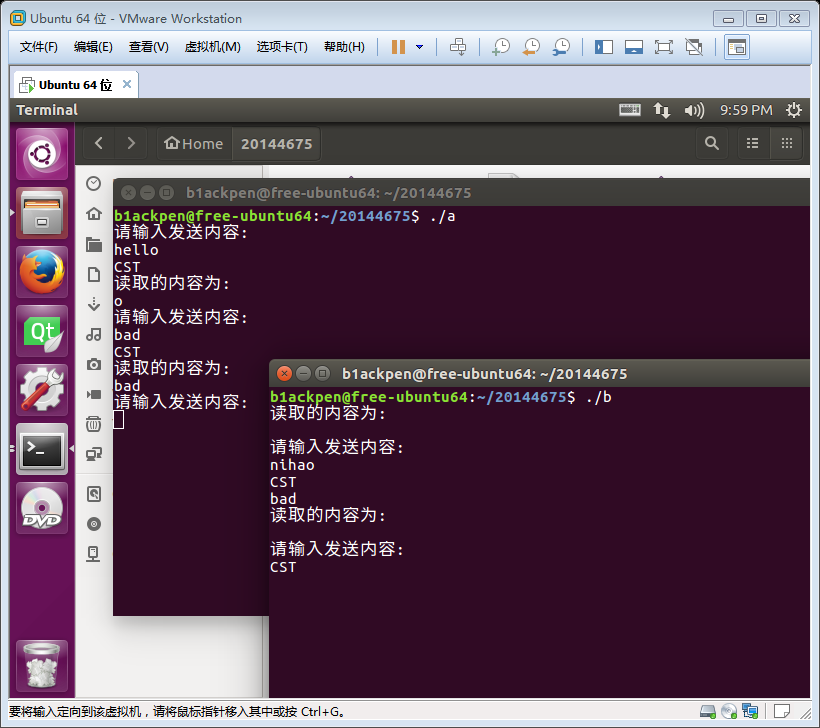
}

}

msgctl(qid,IPC\_RMID,NULL); /\*关闭消息队列\*/

return 0;

}



1. 设计两个程序，要求用mmap系统，实现简单的聊天程序。

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/mman.h>

#define PARENTMSG 0

#define SONMSG 1

#define INITMSG 2

typedef struct

{

char msg[2048];

int type;

}people;

int main()

{

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);

p\_map->type = INITMSG;

result = fork();

if(result < 0)

{

perror("Fork failed");

exit(0);

}

else if(result == 0)

{

printf("In child\*\*\*\*\*Sending a message--:\n");

scanf("%s",p\_map->msg);

p\_map->type = PARENTMSG;

while(1)

{

if(p\_map->type == SONMSG)

{

printf("Child Reciving a message--:\t%s\n",p\_map->msg);

printf("Child Sending a message--:\n");

scanf("%s",p\_map->msg);

if(strcmp(p\_map->msg,"ByeBye") == 0)

{

p\_map->type = PARENTMSG;

break;

}

p\_map->type = PARENTMSG;

}

}

munmap(p\_map, sizeof(people)\*10);

exit(0);

}

else

{

while(1)

{

if(p\_map->type == PARENTMSG)

{

printf("Father Receiving a message:\t%s\n",p\_map->msg);

printf("Father Sending a message:\n");

scanf("%s",p\_map->msg);

if(strcmp(p\_map->msg,"ByeBye") == 0)

{

break;

}

p\_map->type = SONMSG;

}

}

munmap(p\_map,sizeof(people)\*10);

printf("解除内存映射\n");

}

return 0;

}

