

《操作系统原理》实验报告

学院: 信息科学与工程学院

专业班级: 电子信息工程

学号:

学生姓名: Henry Lee

指导教师:

2023 年 10 月

实验一：熟悉系统命令

实验目的：

练习和操作一些基本的 Linux 命令，包括操作命令、管理命令、帮助命令和文本编辑命令，并通过使用文本编辑程序完成程序的录入，达到能基本操作和使用 Linux 操作系统的目的。

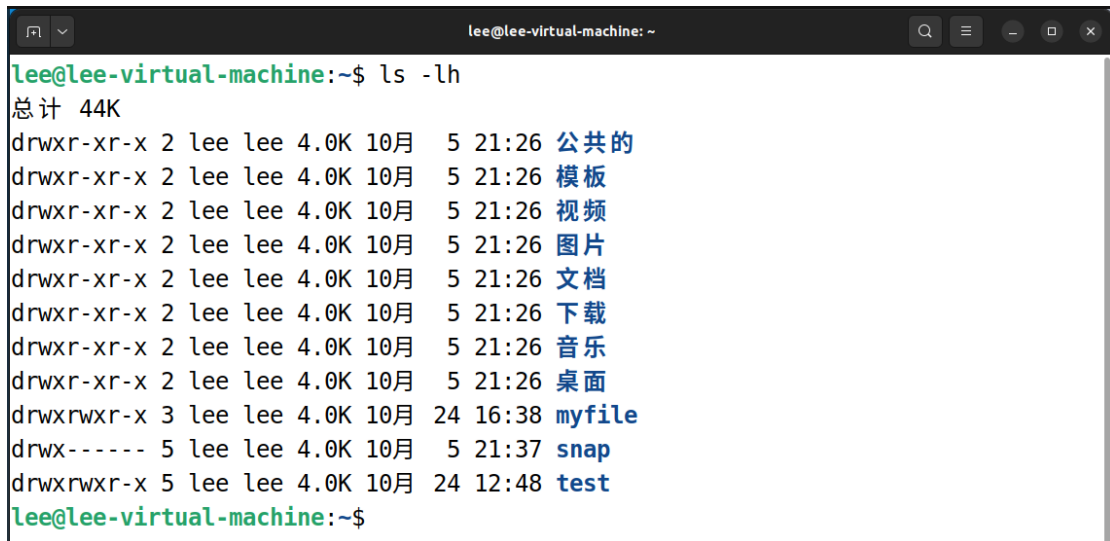
实验要求：

写出一些基本的 Linux 命令的操作步骤，选取 5 个主要命令写入报告并给出例句。

实验步骤及内容：

1. 使用 ls 命令列出文件目录

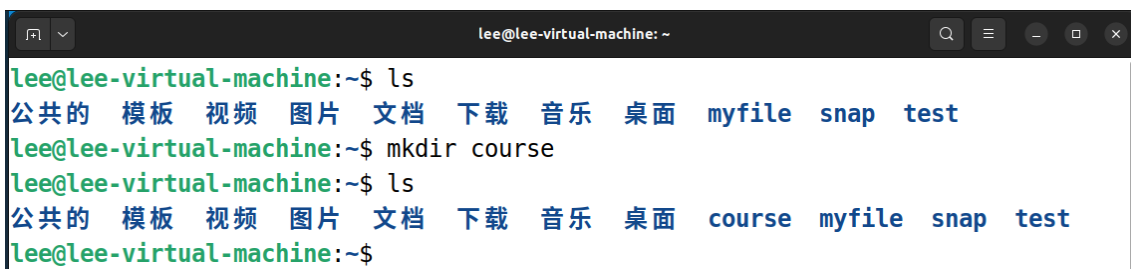
新建终端后，输入命令 `ls -lh`，按长格式列出文件名，包括文件类型标识、权限、链接数、文件主名、文件组名、文件大小和日期。



```
lee@lee-virtual-machine: ~  
lee@lee-virtual-machine:~$ ls -lh  
总计 44K  
drwxr-xr-x 2 lee lee 4.0K 10月 5 21:26 公共的  
drwxr-xr-x 2 lee lee 4.0K 10月 5 21:26 模板  
drwxr-xr-x 2 lee lee 4.0K 10月 5 21:26 视频  
drwxr-xr-x 2 lee lee 4.0K 10月 5 21:26 图片  
drwxr-xr-x 2 lee lee 4.0K 10月 5 21:26 文档  
drwxr-xr-x 2 lee lee 4.0K 10月 5 21:26 下载  
drwxr-xr-x 2 lee lee 4.0K 10月 5 21:26 音乐  
drwxr-xr-x 2 lee lee 4.0K 10月 5 21:26 桌面  
drwxrwxr-x 3 lee lee 4.0K 10月 24 16:38 myfile  
drwx----- 5 lee lee 4.0K 10月 5 21:37 snap  
drwxrwxr-x 5 lee lee 4.0K 10月 24 12:48 test  
lee@lee-virtual-machine:~$
```

2. 使用 mkdir 命令建立目录

在终端上输入命令 `mkdir course`，即在当前目录下建立一个新的目录 `course`。



```
lee@lee-virtual-machine: ~  
lee@lee-virtual-machine:~$ ls  
公共的 模板 视频 图片 文档 下载 音乐 桌面 myfile snap test  
lee@lee-virtual-machine:~$ mkdir course  
lee@lee-virtual-machine:~$ ls  
公共的 模板 视频 图片 文档 下载 音乐 桌面 course myfile snap test  
lee@lee-virtual-machine:~$
```

3. 使用 mv 命令移动文件

将已有的 myfile 目录下的 hello.c 文件,利用 mv [filename] [direction]命令,移动 hello.c 文件到新建好的空目录 course 下面。

```
lee@lee-virtual-machine: ~  
lee@lee-virtual-machine:~$ ls  
公共的 模板 视频 图片 文档 下载 音乐 桌面 course myfile snap test  
lee@lee-virtual-machine:~$ ls course  
lee@lee-virtual-machine:~$ ls myfile  
hello.c  
lee@lee-virtual-machine:~$ cd myfile  
lee@lee-virtual-machine:~/myfile$ mv hello.c ~/course  
lee@lee-virtual-machine:~/myfile$ cd  
lee@lee-virtual-machine:~$ ls course  
hello.c  
lee@lee-virtual-machine:~$
```

4. 使用 cat 命令显示文件内容

将刚才移动到 course 目录下面的 hello.c 文件里的内容,利用 cat hello.c 命令显示在终端上面。

```
lee@lee-virtual-machine: ~/course  
lee@lee-virtual-machine:~/course$ ls  
hello.c  
lee@lee-virtual-machine:~/course$ cat hello.c  
#include <stdio.h>  
  
int main()  
{  
    printf("Hello,this is Lee's OS course homework!\n");  
}  
lee@lee-virtual-machine:~/course$
```

5. 使用 chmod 命令改变文件读取权限

利用 chmod u-r hello.c 命令,取消当前用户对 hello.c 文件的读权限,使其不可读。

```
lee@lee-virtual-machine: ~/course  
lee@lee-virtual-machine:~/course$ ls -l  
总计 4  
-rw-rw-r-- 1 lee lee 92 11月  3 13:00 hello.c  
lee@lee-virtual-machine:~/course$ chmod u-r hello.c  
lee@lee-virtual-machine:~/course$ ls -l  
总计 4  
--w-rw-r-- 1 lee lee 92 11月  3 13:00 hello.c  
lee@lee-virtual-machine:~/course$ cat hello.c  
cat: hello.c: 权限不够  
lee@lee-virtual-machine:~/course$
```

实验二：运用 C 语言函数实现操作系统中文件的读写操作

实验目的：

熟悉运用 C 语言函数实现操作系统中文件的读写操作；参考教材 P31 例题，学习进程控制算法设计。

实验要求：

综合使用 Linux 基本文件命令、编辑器的使用。练习在 Linux 的字符模式下，编辑、编译及调试一个 C 程序的基本方法。

在上述基础上，自己练习调试 C 语言程序，如杨辉三角等。

实验步骤及内容：

（一）写一个 C 语言程序，利用系统调用，实现文件的复制功能

1.1 在~/course 下，新建一个 test 目录，并在该目录下面创建一个 filecopy.c 的源文件，接着在终端中输入 vi filecopy.c 命令：

```
lee@lee-virtual-machine: ~/course/test
lee@lee-virtual-machine:~/course$ mkdir test
lee@lee-virtual-machine:~/course$ cd test
lee@lee-virtual-machine:~/course/test$ touch filecopy.c
lee@lee-virtual-machine:~/course/test$ ls
filecopy.c
lee@lee-virtual-machine:~/course/test$ vi filecopy.c
```

1.2 按下回车后，即使用 vi 编辑器打开了我们刚创建好的 filecopy.c 文件，此时由终端进入到 vi 全屏幕编辑画面：

```
lee@lee-virtual-machine: ~/course/test
~
~
~
~
~
~
~
~
~
~
"filecopy.c" 0 lines, 0 bytes
```

1.3 进入 vi 之后，处于命令行模式，因此需要先按下字母 i 键进入插入模式，然后输入教材示例给出的 C 语言文件复制程序的代码：

```
lee@lee-virtual-machine: ~/course/test
#include <fcntl.h>
#include <sys/stat.h>
#include <stdio.h>
#include <unistd.h>
#define SIZE 1

void filecopy(char *Infile,char *Outfile)
{
    char Buffer[SIZE];
    int In_fh,Out_fh,Count;
    if((In_fh=open(Infile,O_RDONLY))==-1)
        printf("Open the Infile!");
    if((Out_fh=open(Outfile,(O_WRONLY|O_CREAT|O_TRUNC),(S_IRUSR|S_IWUSR))==-1)
        printf("Open the Outfile!");
    while((Count=read(In_fh,Buffer,sizeof(Buffer)))>0)
    {
        if(write(Out_fh,Buffer,Count)!=Count)
            printf("Writing date");
        if(Count==-1)
            printf("Reading date");
    }
    close(In_fh);
    close(Out_fh);
}
```

1.4 由于教材所给出的代码仅为 filecopy 的函数定义，而为了使程序能够在终端中运行，还需要在后面补充 main 函数部分来调用 filecopy 这个函数。并且，在这里我给它传入两个文件名（下图中的 1.txt 和 2.txt）作为 filecopy 函数的参数。

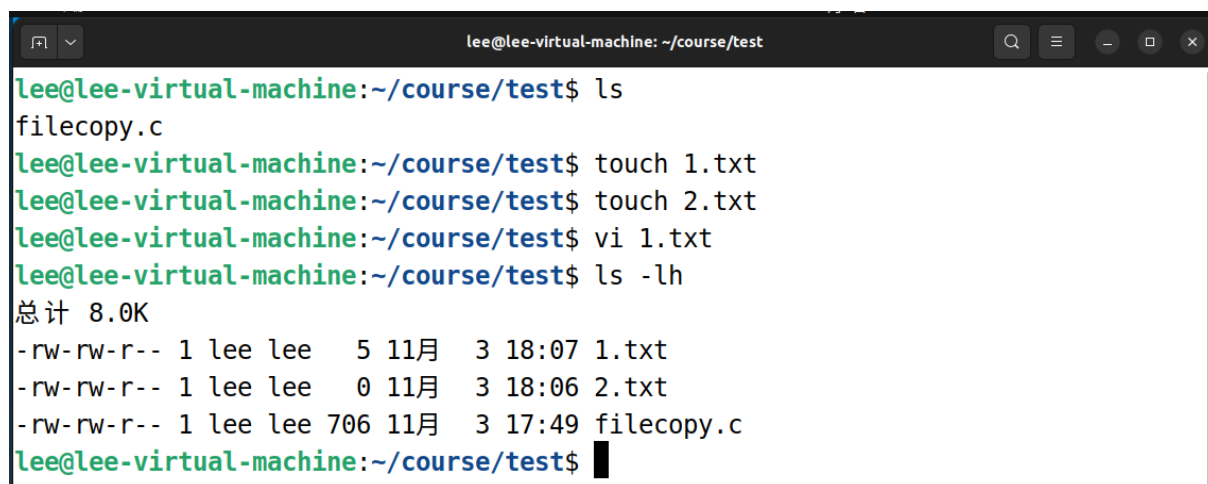
```
int main()
{
    char *infile="1.txt";
    char *outfile="2.txt";
    filecopy(infile,outfile);
    return 0;
}
~
```

1.5 至此，已经完成好了 filecopy.c 文件的编辑工作。接着，按下 ESC 键由插入模式切换到命令模式，然后键入 :wq!

```
|:wq!|
```

回车后，即可保存和退出 vi 编辑器，并返回到终端中。

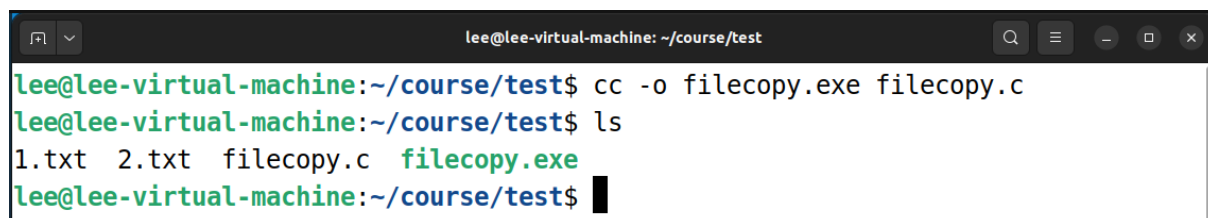
1.6 然后，仿照上述步骤，在该目录下创建好两个文本文件，分别命名为“1.txt”和“2.txt”。其中，使用 vi 在 1.txt 里面写入内容 GOOD，而 2.txt 里则什么都不写，操作如下：



```
lee@lee-virtual-machine: ~/course/test
lee@lee-virtual-machine:~/course/test$ ls
filecopy.c
lee@lee-virtual-machine:~/course/test$ touch 1.txt
lee@lee-virtual-machine:~/course/test$ touch 2.txt
lee@lee-virtual-machine:~/course/test$ vi 1.txt
lee@lee-virtual-machine:~/course/test$ ls -lh
总计 8.0K
-rw-rw-r-- 1 lee lee  5 11月  3 18:07 1.txt
-rw-rw-r-- 1 lee lee  0 11月  3 18:06 2.txt
-rw-rw-r-- 1 lee lee 706 11月  3 17:49 filecopy.c
lee@lee-virtual-machine:~/course/test$
```

1.7 接着利用 cc 命令编译写好的 filecopy.c 文件，使其生成 filecopy.exe 的可执行文件。

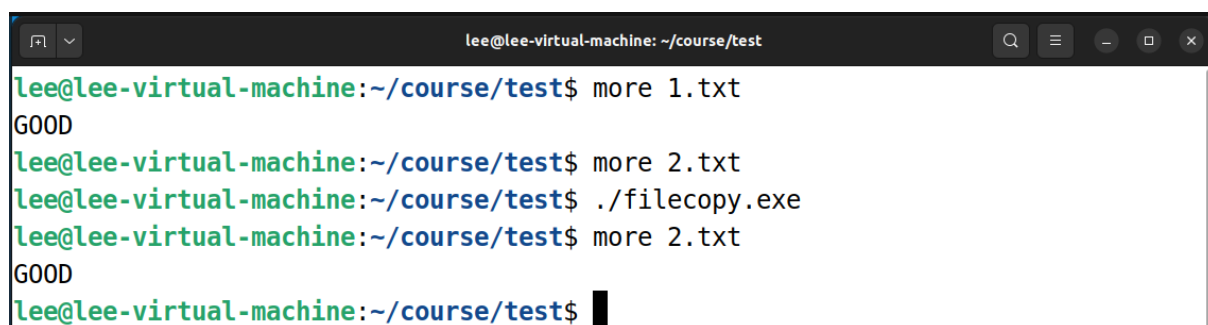
在终端上输入：cc -o filecopy.exe filecopy.c 即可：



```
lee@lee-virtual-machine: ~/course/test
lee@lee-virtual-machine:~/course/test$ cc -o filecopy.exe filecopy.c
lee@lee-virtual-machine:~/course/test$ ls
1.txt 2.txt filecopy.c filecopy.exe
lee@lee-virtual-machine:~/course/test$
```

1.8 然后利用 ./filecopy.exe 命令，运行生成的可执行文件。

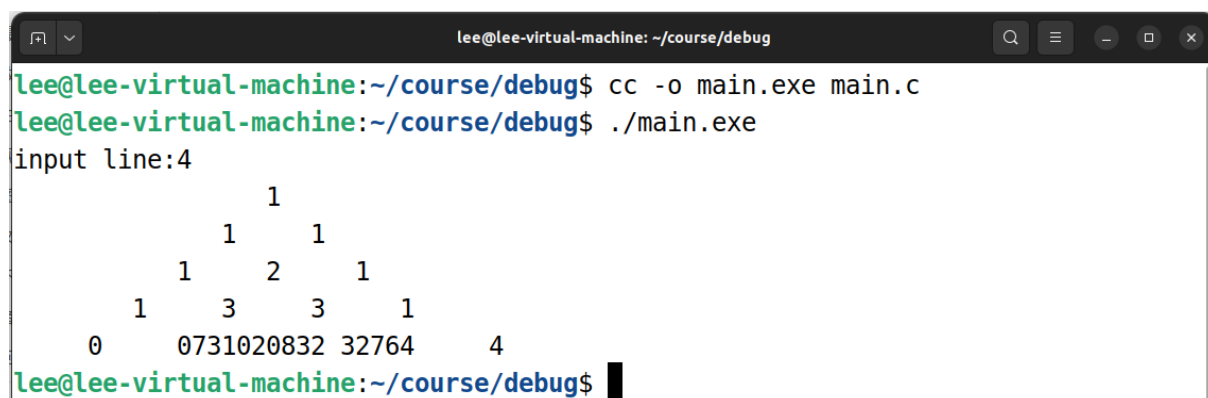
如下图所示，空白文本文件 2.txt 中被拷贝了 1.txt 的内容，即实现了文件的复制功能：



```
lee@lee-virtual-machine: ~/course/test
lee@lee-virtual-machine:~/course/test$ more 1.txt
GOOD
lee@lee-virtual-machine:~/course/test$ more 2.txt
lee@lee-virtual-machine:~/course/test$ ./filecopy.exe
lee@lee-virtual-machine:~/course/test$ more 2.txt
GOOD
lee@lee-virtual-machine:~/course/test$
```

(二) 练习调试 C 语言程序，打印杨辉三角

这里，我们仿照前面的步骤，在~/course/debug 目录下，已经写好了一个打印杨辉三角的 C 语言程序，但在编译运行后，输出结果显示有误：

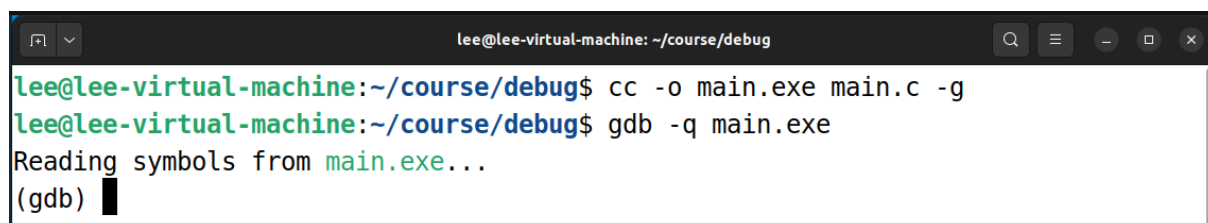


```
lee@lee-virtual-machine: ~/course/debug
lee@lee-virtual-machine:~/course/debug$ cc -o main.exe main.c
lee@lee-virtual-machine:~/course/debug$ ./main.exe
input line:4
      1
     1 1
    1 2 1
   1 3 3 1
  0 0731020832 32764 4
lee@lee-virtual-machine:~/course/debug$
```

由上面的结果，初步可判断出可能存在数组访问越界的情况。现在利用 **GDB 工具** 来对这个运行有误的 C 语言程序进行调试，以定位到具体出错的地方。

2.1 调试准备

先使用 `cc -o main.exe main.c -g` 产生用于符号测试的执行文件，方便后续的调试；接着，输入 `gdb -q main.exe`，进入 GDB 调试界面：



```
lee@lee-virtual-machine: ~/course/debug
lee@lee-virtual-machine:~/course/debug$ cc -o main.exe main.c -g
lee@lee-virtual-machine:~/course/debug$ gdb -q main.exe
Reading symbols from main.exe...
(gdb)
```

2.2 进行调试

在 GDB 环境下，可输入以下命令进行调试工作：

`list` 显示代码

`break [行号]` 设置断点

`delete break [断点编号]` 删除断点

`run` 运行程序

`step` 单步运行，`continue` 继续运行

`print [变量名]` 查看变量的值

本次调试过程中的部分画面，如下图所示：

```
lee@lee-virtual-machine: ~/course/debug
lee@lee-virtual-machine:~/course/debug$ cc -o main.exe main.c -g
lee@lee-virtual-machine:~/course/debug$ gdb -q main.exe
Reading symbols from main.exe...
(gdb) list
1      #include<stdio.h>
2      //Yang Hui Triangle
3
4      int main()
5      {
6          int n;
7          printf("input line:");
8          scanf("%d",&n);
9
10         int arr[n][n];
(gdb) break 27
Breakpoint 1 at 0x1462: file main.c, line 27.
(gdb) run
Starting program: /home/lee/course/debug/main.exe
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
input line:4

Breakpoint 1, main () at main.c:27
27         for(int i=0;i<=n;i++)
(gdb) █
```

2.3 调试完毕

经过多次调试，最终定位到上方源代码 27 行 for 循环中的循环判断条件设置有误。修改好相关代码后，重新编译和运行程序，最后在屏幕上正确显示打印出了杨辉三角：

```
lee@lee-virtual-machine: ~/course/debug
lee@lee-virtual-machine:~/course/debug$ cc -o main.exe main.c
lee@lee-virtual-machine:~/course/debug$ ./main.exe
input line:5
          1
         1 1
        1 2 1
       1 3 3 1
      1 4 6 4 1
lee@lee-virtual-machine:~/course/debug$ █
```


实验三：熟悉操作系统中进程通信的实例——管道的建立

实验目的：

了解和熟悉 Linux 支持的消息通信机制。通过编写 C 语言程序了解系统管道通信的建立，并熟悉管道传送信息的方式及步骤。

实验要求：

参考教材 P66-67 例 1、例 2，利用文件的读写操作进行编程，建立一个 C 语言的管道程序。

实验步骤及内容：

● 实验原理

用 C 语言编写一个程序，建立一个管道。同时父进程生成一个子进程，子进程向管道中写入一个字符串，接着父进程从管道中读出该字符串。

● 源代码

```
lee@lee-virtual-machine: ~/course
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

int main()
{
    int x, fd[2];
    char buf[30], s[30];
    pipe(fd);                /*创建管道*/
    while((x=fork())!=-1);   /*创建子进程失败时，循环*/
    if(x==0)
    {
        sprintf(buf,"this is an example\n");
        write(fd[1],buf,30); /*把buf中的字符写入管道*/
        sleep(5);           /*睡眠5秒，让父进程读*/
        exit(0);            /*关闭x，子进程自我中止*/
    }
    else
    {
        wait(0);            /*父进程读管道中的字符*/
        read(fd[0],s,30);
        printf("%s",s);
    }
}

"pipe.c" 25 lines, 744 bytes
```

● 调试结果

子进程写入字符串到管道后，会执行 `sleep(5)`，此时终端中光标闪烁五秒，如下：

```
lee@lee-virtual-machine: ~/course
lee@lee-virtual-machine:~/course$ cc -o pipe.exe pipe.c
lee@lee-virtual-machine:~/course$ ./pipe.exe
█
```

五秒过后返回到了父进程，此时父进程会读取并打印管道中子进程写入的字符串：

```
lee@lee-virtual-machine: ~/course
lee@lee-virtual-machine:~/course$ cc -o pipe.exe pipe.c
lee@lee-virtual-machine:~/course$ ./pipe.exe
this is an example
lee@lee-virtual-machine:~/course$ █
```

思考题:

体会操作系统如何通过系统调用 `fork`, `exec`, `wait` 等实现 `shell` 功能。改写此程序，加入自己的特点（如学号+姓名+课程等），并输出父子程序的说明。

示例代码:

```
#include <stdio.h>
int main ()
{
    char  command[32];
    char  * prompt = "$";
    while (printf ("%s", prompt) , gets (command) != NULL)
    {
        if (fork () == 0)                /*子进程段*/
            execlp (command , command , (char *)0);
        else
            wait (0);                    /*父进程段*/
    }
}
```

● 解题思考

首先，对于示例代码，我将 `gets()` 函数变更为 `fgets()` 函数，使得现在程序可以从输入流 `stdin` 中获取 `sizeof(command)` 个字符并存入 `command` 中，从而防止堆栈溢出；

但是，由于 `fgets()` 函数会把换行符存入缓冲区，因此需要手动将换行符转换为字符串的结束符 `'\0'`，即添加语句：`command[strcspn(command, "\n")] = 0;`

最后，在此代码中，我加入了自己的特点（如学号、姓名、课程名称）。

● 最终代码



```
lee@lee-virtual-machine: ~/test/3
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
#include<string.h>

int main(){
    char command[32];
    char *prompt=" $ ";
    char *studentID="202108405111";
    char *name="李振荣";
    char *course="操作系统原理";

    while(printf("%s-%s-%s", studentID, name, course, prompt) , fgets(command, sizeof(command), stdin) != NULL){
        command[strcspn(command, "\n")] = 0;
        if(fork() == 0){
            printf("Son pid:\n");           //输出子进程段说明
            execlp(command, command, (char*)0);
        }
        else{
            wait(0);
            printf("Father pid:\n");       //输出父进程段说明
        }
    }

    return 0;
}
"shell.c" 26 lines, 597 bytes
```

● 调试结果

创建好上述的 shell.c 的源文件，编译后运行生成的 shell.exe 文件。

输入 ls，此时父进程会先创建（fork）一个子进程。在执行子程序时，去调用函数 execlp，传进函数的参数就为‘ls’。效果为向终端输入命令 ls，返回结果为显示当前目录的所有文件和子目录，最后返回到父程序：

```
lee@lee-virtual-machine: ~/course
lee@lee-virtual-machine:~/course$ cc -o shell.exe shell.c
lee@lee-virtual-machine:~/course$ ./shell.exe
202108405111-李振荣-操作系统原理 $ ls
Son pid:
debug hello.c hello.exe pipe.c pipe.exe shell.c shell.exe test test1
Father pid:
202108405111-李振荣-操作系统原理 $ █
```

同理，还可以在里面输入一些其他的命令，如 who、date、pwd 等：

```
lee@lee-virtual-machine: ~/course
lee@lee-virtual-machine:~/course$ cc -o shell.exe shell.c
lee@lee-virtual-machine:~/course$ ./shell.exe
202108405111-李振荣-操作系统原理 $ ls
Son pid:
debug hello.c hello.exe pipe.c pipe.exe shell.c shell.exe test test1
Father pid:
202108405111-李振荣-操作系统原理 $ who
Son pid:
lee      tty2      2023-11-12 14:37 (tty2)
Father pid:
202108405111-李振荣-操作系统原理 $ pwd
Son pid:
/home/lee/course
Father pid:
202108405111-李振荣-操作系统原理 $ date
Son pid:
2023年 11月 12日 星期日 17:01:23 CST
Father pid:
202108405111-李振荣-操作系统原理 $ █
```

个人小结

每一次的实验操作都会让我学到在理论课堂上难以学到的东西，所以我对每一次实验操作的机会都非常珍惜。不一定我最终的课程实验报告会完成得多么完美，但我总是在每次实验的过程中投入到了百分之百的精力去努力完成它！那么，经过这三次 Linux 操作系统的上机实验过后，我也有以下的总结和体会：

首先，对于实验一，让我们自己亲身尝试了许多 Linux 的基本命令，算是对 Linux 操作系统进行了初步的入门。相较于我们日常使用到的图形操作界面的 Windows 系统，而现在却让我们在终端中只靠敲命令去执行一些操作，我也感到很不适应。但慢慢地，随着练习次数的增加，我对命令的掌握也更加熟练，也愈发体会到命令操作所带来的高效！比如，对于一个文件的拷贝工作，在图形界面下，你也许需要用鼠标从一个文件夹复制粘贴到另一个文件夹下，需要好几个步骤；而在 Linux 命令模式下，**只需要一行命令即可实现**，可见命令操作的高效和强大！

其次，对于实验二，让我们练习了在 Linux 字符模式下，编辑、编译和调试一个 C 语言程序。在刚开始编辑程序时，我们使用到了 Linux 下的 **vi 编辑器**，对于初学者来说这个的确非常难用。其命令模式、插入模式以及底行模式间的转换，让我们在编写代码过程中遇到了不小的麻烦。不过我们也很快适应了这种“敲命令”式的风格，并掌握到了光标的使用技巧。像“一行命令即可实现删除”这样的操作，让我们再次感受到了 vi 编辑器开发程序时的高效，真不愧为很多程序员大佬所推崇的开发工具。除此之外，我还学习了如何使用 **GDB 工具**来调试 C 语言程序，也是**依靠命令来设置断点、单步运行以及显示变量值**等等，真的也很有意思！

最后，在实验三中，我们练习了用 C 语言程序来建立一个管道的例子，从而加深了我们对 Linux 进程通信机制的理解，并增强了我们的操作系统知识。除此之外，在思考题中，我们还结合系统调用，建立了一个具有我们自己个人特色的 Shell 终端，其本质也是用到了进程通信。**但本次实验中我们建立的 Shell 不包含路径检索功能，也没有包括参数处理功能**，所以在此 Shell 终端下仅能执行那些包含默认路径的命令如 ls、pwd、date 等，对于输入 cd 这种命令是暂时无法执行的。

综上，通过这几次实验，让我不仅对 Linux 操作系统的基本使用，以及 Linux 下编写开发和运行调试 C 语言程序，有了一定的了解；而且还让我对 Linux 操作系统的进程通信机制和系统调用方法，有了更深的认识。**最后我想，这些知识也会为我今后无论是学习还是工作上，都将带来重要的帮助和指导！**