实验八 Shell编程实验（进程管理实验）

## 实验目的

通过编写shell程序，了解子进程的创建和父进程与子进程间的协同，获得多进程程序的编程经验。

## 实验内容

**实验内容1**

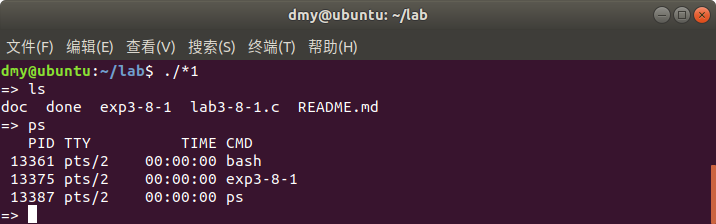
设计一个简单的shell解释程序，能实现基本的bash功能。

**实验内容2**

编写一个带有重定向和管道功能的Shell

## 实验步骤

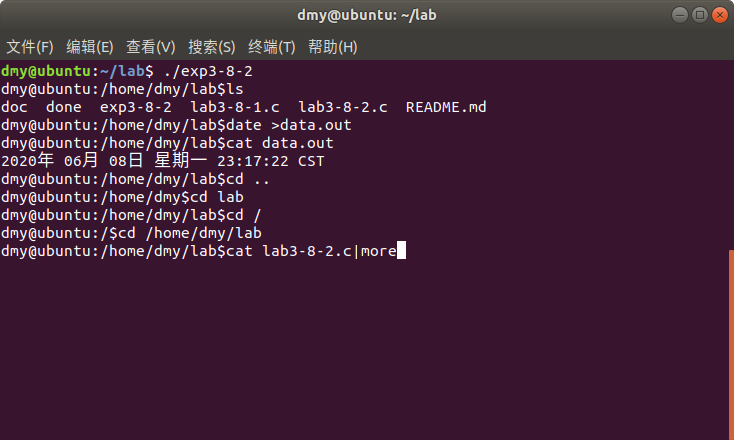
1. 直接运行实验手册上的代码，查看效果

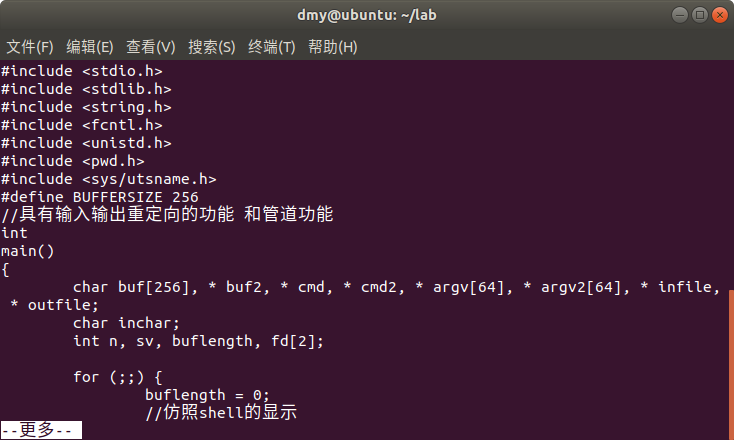


有一些指令和语法未实现比如execvp不能执行的cd指令

1. 完善第二个实验内容代码，模仿shell界面

管道与重定向功能测试





第二个实验的代码如下

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <fcntl.h>  #include <unistd.h>  #include <pwd.h>  #include <sys/utsname.h>  #define BUFFERSIZE 256  //具有输入输出重定向的功能 和管道功能  int  main()  {  char buf[256], \* buf2, \* cmd, \* cmd2, \* argv[64], \* argv2[64], \* infile, \* outfile;  char inchar;  int n, sv, buflength, fd[2];  for (;;) {  buflength = 0;  //仿照shell的显示  {  struct utsname t;  uname(&t);  printf("%s@%s:%s$", getpwuid(getuid())->pw\_name,t.nodename, getcwd(NULL, NULL));  }  inchar = getchar();  while (inchar != '\n' && buflength < BUFFERSIZE) {  buf[buflength++] = inchar;  inchar = getchar();  }  if (buflength > BUFFERSIZE) {  fprintf(stderr, "Command too long,please enter again!\n");  buflength = 0;  continue;  }  else  buf[buflength] = '\0';  //检查是否具有管道操作符  //strstr()在字符串中查找指定字符串的第一次出现，buf2指向管道符号前端的命令  buf2 = strstr(buf, "|");  if (buf2)  \*buf2++ = '\0';  else {  //否则查看是否具有重定向的操作符  infile = strstr(buf, "<");  outfile = strstr(buf, ">");  if (infile) {  \*infile = '\0';  infile = strtok(infile + 1, " \t\n");  }  if (outfile) {  \*outfile = '\0';  outfile = strtok(outfile + 1, " \t\n");  }  }  //解析命令行，分成一个个的标记  cmd = strtok(buf, " \t\n");  //执行管道命令  if (buf2) {  if (strcmp(cmd, "exit") == 0) exit(0);  if (!cmd) {  fprintf(stderr, "Command token error.\n");  exit(1);  }  n = 0;  //管道后端的命令  argv[n++] = cmd;  while (argv[n++] = strtok(NULL, " \t\n"));  //管道前端的命令  cmd2 = strtok(buf2, " \t\n");  if (!cmd2) {  fprintf(stderr, "Command token error.\n");  exit(1);  }  n = 0;  argv2[n++] = cmd2;  while (argv2[n++] = strtok(NULL, " \t\n"));  pipe(fd);  if (fork() == 0) {  dup2(fd[0], 0); //dup2 复制文件句柄，将fd[0]复制到描述符0。  close(fd[0]); close(fd[1]);  execvp(cmd2, argv2);  fprintf(stderr, "\*\* bad command\n"); exit(1);  }  else if (fork() == 0) {  dup2(fd[1], 1);  close(fd[0]); close(fd[1]);  execvp(cmd, argv);  fprintf(stderr, "\*\* bad command\n"); exit(1);  }  close(fd[0]);  close(fd[1]);  wait(&sv);  wait(&sv);  buflength = 0;  }  //如果没有管道命令,如果有重定向就执行重定向操作，如果没有重定向就当作普通shell命令执行  else {  if (cmd) {  if (strcmp(cmd, "exit") == 0) exit(0);  //实现 cd  else if (strcmp(cmd, "cd") == 0)  {  n = 0;  char\* p;  while (argv[n++] = strtok(NULL, "\n"))  sv = sv;  if (chdir(argv[0]) < 0)  puts("error try again");  buflength = 0;  }  else  {  n = 0;  argv[n++] = cmd;  while (argv[n++] = strtok(NULL, " \t\n"));  if (fork() == 0) {  int fd0 = -1, fd1 = -1;  if (infile) fd0 = open(infile, O\_RDONLY);  if (outfile) fd1 = open(outfile, O\_CREAT | O\_WRONLY, 0666);  if (fd0 != -1) dup2(fd0, 0);//dup2 复制文件句柄，将fd0复制到描述符0。  if (fd1 != -1) dup2(fd1, 1);//dup2 复制文件句柄，将fd1复制到描述符1。  close(fd0);  close(fd1);  execvp(cmd, argv);  fprintf(stderr, "\*\* Bad command\n");  exit(1);  }  wait(&sv);  buflength = 0;  }  }  }  }//for  } |