Lab1文档

2019年4月9日　　20:02

16302010026 陈涛

实验环境：Ubuntu 16.04 LTS

实验过程及结果：

```
chen@chen:~/chen/syetempd/lab/lab1$ gcc -o shell main.c
chen@chen:~/chen/syetempd/lab/lab1$ ./shell
myShell pwd
/home/chen/chen/syetempd/lab/lab1
myShell ls > a
myShell cat a
a
lab1Code
main.c
main.o
shell
myShell ls | sort | uniq | wc
      5       5      31
myShell cat < a | sort | uniq | wc > b
myShell cat b
      5       5      31
myShell rm a
myShell rm b
myShell ls
lab1Code  main.c  main.o  shell
myShell ^C
chen@chen:~/chen/syetempd/lab/lab1$ pwd
/home/chen/chen/syetempd/lab/lab1
chen@chen:~/chen/syetempd/lab/lab1$ ls > a
chen@chen:~/chen/syetempd/lab/lab1$ cat a
a
lab1Code
main.c
main.o
shell
chen@chen:~/chen/syetempd/lab/lab1$ ls | sort | uniq | wc
      5       5      31
chen@chen:~/chen/syetempd/lab/lab1$ cat < a | sort | uniq | wc > b
chen@chen:~/chen/syetempd/lab/lab1$ cat b
      5       5      31
chen@chen:~/chen/syetempd/lab/lab1$ rm a
chen@chen:~/chen/syetempd/lab/lab1$ rm b
chen@chen:~/chen/syetempd/lab/lab1$ ls
lab1Code  main.c  main.o  shell
chen@chen:~/chen/syetempd/lab/lab1$
```

上图：前半部分为程序执行结果，后半部分为linux自带shell执行结果
结果显示，程序运行结果与shell的运行结果相同，说明程序功能已经实现

实验逻辑

```
54      switch (cmd->type) {
55          default:
56              fprintf(stderr, "unknown runcmd\n");
57              exit(-1);
58
59          case ' ':
60              ecmd = (struct execcmd *) cmd;
61              if (ecmd->argv[0] == 0)
62                  exit(0);
63              // Your code here ...
64              execvp(ecmd->argv[0], ecmd->argv);
65              break;
66
67          case '>':
68          case '<':
69              // Your code here ...
70              rcmd = (struct redircmd *) cmd;
71              int fd = open(rcmd->file, rcmd->mode, 00777);
72              close(rcmd->fd);
73              dup2(fd, rcmd->fd);
74              close(fd);
75              runcmd(rcmd->cmd);
76              break;
77
78          case '|':
79              // Your code here ...
80              pcmd = (struct pipecmd *) cmd;
81              pipe(p);
82              if (fork1() == 0) {
83                  close(1);
84                  dup2(p[1], 1);
85                  close(p[0]);
86                  close(p[1]);
87                  runcmd(pcmd->left);
88              } else {
89                  wait(&r);
90                  close(0);
91                  dup2(p[0], 0);
92                  close(p[0]);
93                  close(p[1]);
94                  runcmd(pcmd->right);
95              }
96              break;
97      }
98      exit(0);
```

如上，输入的命令在经过处理成cmd格式时，传入该函数，通过判断cmd的类型来判断是哪种命令，并执行想要的处理case。

exec命令： 是用来执行linux自带命令，所以只需使用execvp传入命令与参数（第64行）即可

> < 命令(重定向)： 该命令是将程序的输出重定向到文件或者将文件作为输入。所以在实现时需要打开cmd中的文件（第71行），并使用dup2 （第73行）将 标准输出（当命令为 < 时，标准输入）重定向到打开的文件描述符，之后递归执行命令（第75行）即可。由于实现过程已经将需要重定向的输入或者输出存入结构中的fd，所以实现时无需判断是哪种符号，直接采用相同代码即可。

| 命令(管道)： 管道命令是将管道左边的输出作为管道右边命令的输入，所以需要采用父子线程，

并采用pipe函数，让父子线程进行通信。让子线程运行左半部分命令（第83-87行），父线程运行右半部分命令（第89-94行），将子线程的标准输出与父线程的标准输入相关联（第84、91行），并且在父线程运行前保证子线程运行结束，这样就可以把子线程的输出传输到父线程的输入，以此实现管道命令

实验中碰到问题

实验中，遇到了一个问题，新创建的文件写入之后，无法再次打开。
后面发现是open函数参数传入有误，没有传入mode参数
之后将

```
  int fd = open(rcmd->file, rcmd->mode);
```

改为

```
 int fd = open(rcmd->file, rcmd->mode, 00777);
```

即可