广东工业大学

先进制造学院

《Linux 技术》

期末作业

课 程 名 称 ：	Linux 技术
学 生 姓 名 ：	陈煜祺
班 级 学 号 ：	22 计科 8 班 - 3122008883
填 写 日 期 ：	2024 年 12 月 10 日

目录

一、shell demo 程序

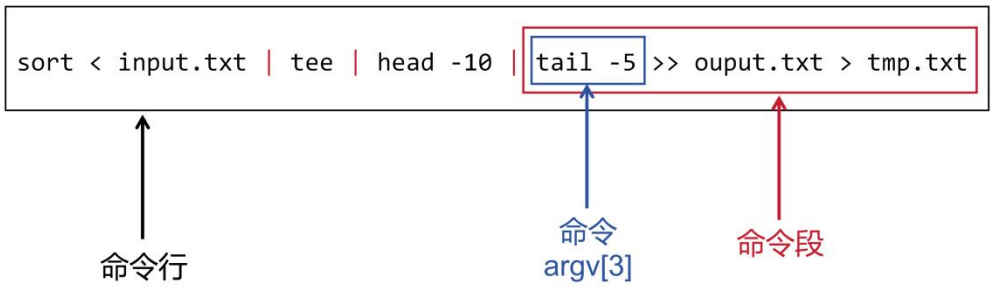
1. 设计思路说明

（1）总体思路

本 shell demo 程序需要实现内置命令、外部命令、重定向文件流、管道命令、非法命令处理共 5 个基本需求。为此，需要先正确地输入用户的命令，并对其进行合理的语义分割，程序才能正确识别。

在此，我们定义“命令行”为用户输入的一整行字符串；“命令段”为将命令行按管道符进行分割后得到的数个子字符串；而“命令”则是命令段中去除了重定向文件流部分后剩余的，真正用于执行的命令；其中一个命令又包含若干个“单词”。需要特别说明的是，其实在执行命令段时，多余的输入重定向文件会被省略。但参照真实 shell 的处理方案，应该允许用户输入，只是在后续执行时将其无效化。

综上，部分关键全局变量的定义如下图所示。



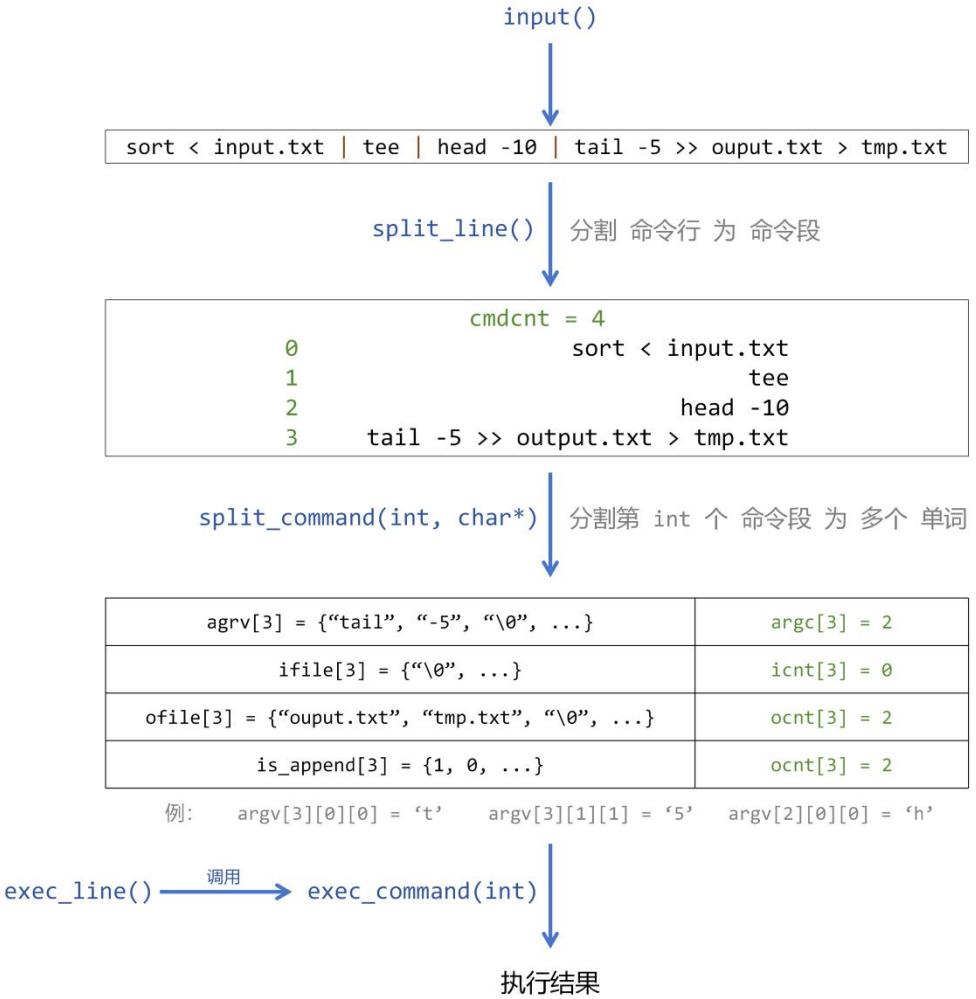
内容	内容存储变量	数量存储变量
命令行	char line[SIZE]	为 1 ，无需存储
命令段	不单独存储	int cmdcnt
命令	char argv[MAXNUM][MAXNUM][SIZE]	int argc[MAXNUM]
输入文件	char ifile[MAXNUM][MAXNUM][SIZE]	int icnt[MAXNUM]
输出文件	char ofile[MAXNUM][MAXNUM][SIZE]	int ocnt[MAXNUM]

变量	类型	含义
argv[i][j][k]	char	第 i 个命令段 的 第 j 个单词 的 第 k 个字符
ifile[i][j][k]	char	第 i 个命令段 的 第 j 个输入文件 的 第 k 个字符
argc[i]	int	第 i 个命令段 所拥有的 单词数

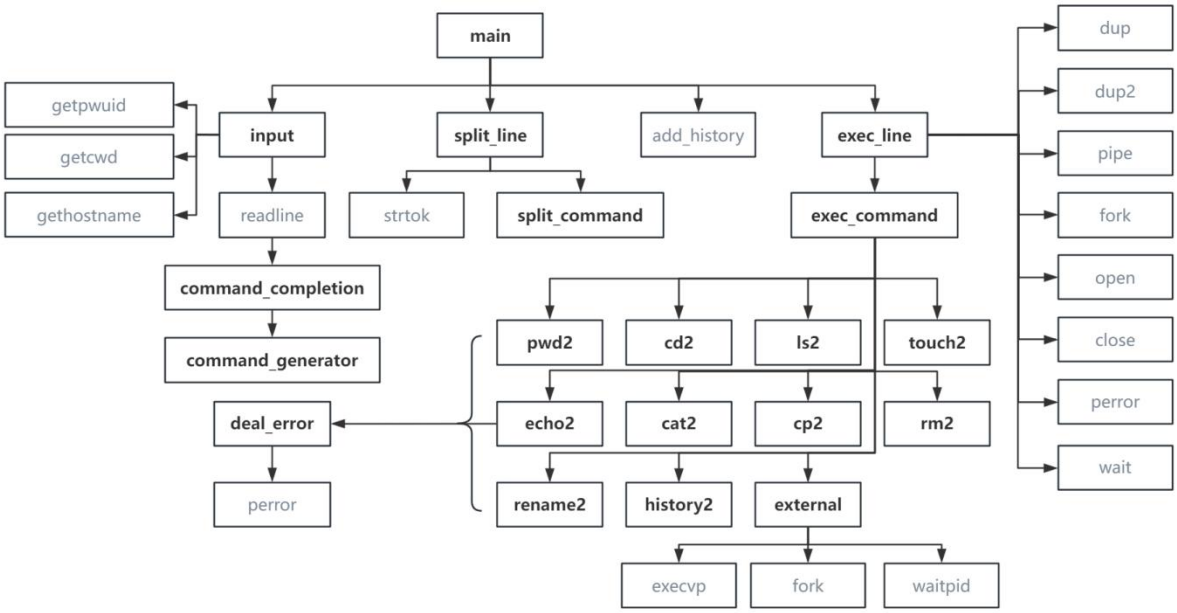
因此，本程序对用户输入的命令行的总体处理流程如下图所示。

先通过 `input()` 函数将用户命令行输入，然后使用 `split_line()` 将其分割为数个命令段。接着对每个命令段，调用 `split_command()` 函数提取其命令单词、重定向输入文件名、重定向输出文件名，并将其内容和数量存储到对应的全局变量中。至此，语义分割部分完成。

接下来，程序调用 `exec_line()` 函数执行用户输入的命令行。该函数将创建多个子进程，每个子进程将调用 `exec_command()` 函数执行对应的命令段，子进程间通过管道通信，并通过 `dup2()` 实现文件重定向。此外还需实现提示词输出、记录历史命令等零碎功能。



为了更清晰地说明本程序的执行流程，现将各函数之间的调用关系图展示在下页。其中黑色字体标识的是本程序中编写的函数，灰色字体标识的是调用的外部库函数。另外省略了部分简单的字符串处理函数如 `strcpy`、`strcmp` 等。其中 `command_completion` 等函数用于特色功能 TAB 命令补全。



由于本程序模块众多、调用关系复杂、实现细节多，文字描述难以言尽。报告未尽之处，敬请老师参阅对应部分的源代码，或运行程序进行试验。另外，本程序及报告有七大特色，具体请参见特色说明文档。

（2）语义分割

本部分将详细说明语义分割模块的实现思路。

首先通过 `input()` 函数将用户输入的命令行输入。该函数同时还实现了特色提示词输出、全局变量清空、将命令行录入历史记录的功能。具体请参见代码 `split.c`。

接着调用 `split_line()` 函数将命令行分割成若干个命令段。该函数的核心是调用了 `strtok()` 函数，按管道符 `|` 将字符串分割。

对分割后得到的每个命令段，送入 `split_command()` 函数进行进一步的分割。该函数的核心是使用了双指针算法（又称滑动窗口算法，可以参见力扣），将对应的字符串子串取出，并复制到对应的全局变量中存储。其规则如下：

1. 从前往后遍历当前处理的命令段
2. 跳过所有的空白符（如空格、TAB 等）
3. 对于遇到的每一个重定向符（包括 `<` 、 `>` 和 `>>`），使用双指针算法，将其后紧跟的一个单词提取出来，复制到对应的全局变量数组中。

（3）内置命令

本部分将详细说明各内置命令的实现思路。由于内置命令较多，现用表格展示如下。

内置命令	调用格式	实现功能	实现思路
pwd2	pwd2	显示当前工作路径	调用 getcwd 函数实现
cd2	cd2 + 路径 path	将工作路径转移到 path 下； 若 path 为空字符串，则返回到当前主目录下； 若 path 为“-”，则后退到上一工作目录。	比对 path，若为空，则直接返回；若为 “-” ，则修改为 “..”。然后调用 chdir 函数实现。
ls2	ls2 + 路径 path	显示路径 path 下（path 为空时，则指当前工作目录）的所有文件和文件夹。	调用 opendir 函数打开 path 路径，将其记录在 dirent 结构体下。接着遍历结构体链表，利用其结点中记录的信息，结合 S_ISDIR 宏函数判断其属性后，按题目要求进行输出。
touch2	touch2 + 文件 filename	若文件 filename 不存在，则新建文件 filename；否则什么都不做	调用 fopen 函数实现
echo2	echo2 + 字符串 str	输出字符串 str	对 argv 数组中存储的字符串，用 printf 输出。
cat2	cat2 + 文件 filename	显示文件 filename 内容，并在每一行前添加行号。	调用 fopen 函数打开文件，并用 fgetc 函数逐个读入字符，读到换行符时，将行号变量自增，然后输出。注意在读取前也需先输出一次第一行的行号。最后调用 fclose 关闭文件。
cp2	cp2 + 文件 filename1 + 文件 filename2	新建文件 filename2，将 filename1 的内容拷贝到 filename2 中	调用 fopen 函数打开两个文件，并用 fgetc 逐个读取 filename1 中的字符，用 fputc 逐个输出到 filename2 中。最后用 fclose 关闭两个文件。
rm2	rm2 + 文件 filename rm2 -r + 文件夹 foldername	删除文件 filename 删除文件夹 foldername	调用 remove 函数实现，注意若 rm2 的第一个参数为-r 时，remove 函数应传入的是 rm2 的第二个参数。
rename2	rename2 +文件 filename1 + 文件 filename2	将文件 filename1 改名为 filename2	调用 rename 函数实现
history2	history2	显示使用过的所有历史命令	历史记录用循环队列存储，根据队首指针来遍历队列，逐个输出。
quit	quit	退出 shelldemo	用 break 退出 main 函数中的 while(1) 循环。

（4）外部命令

外部命令通过 external 函数实现，其实现思路是先用 fork 函数创建子进程，再在子进程中调用 execvp 函数执行外部命令。

（5）重定向文件流

重定向的输入输出文件已经在语义分割时分割出来，并存储在了 ifile 和 ofile 数组中，其中 is_append 数组中记录的值决定了是以清空文件内容方式，还是以附加到文件末尾的方式进行输出。

输入重定向使用 open(ifile[i][icnt[i]-1], O_RDONLY)的方式打开文件，并通过 dup2(ifid, STDIN_FILENO)实现重定向。

输入重定向使用 open(ofile[i][ocnt[i]-1], O_WRONLY | O_CREAT | O_APPEND) 或者 open(ofile[i][ocnt[i]-1], O_WRONLY | O_CREAT | O_TRUNC)的方式打开文件，取决于 is_append 的值。然后通过 dup2(ofid, STDOUT_FILENO)实现重定向。

（6）管道命令

由管道符分割的命令段已经在语义分割时分割出来，并存储在了 agrv 数组中。

每个命令段都用独立的子进程实现，子进程之间通过管道进行通信。简而言之，第 i 个子进程的输出是第 i+1 个子进程的输入。当然，第 1 个子进程的输入与第 cmdcnt 个子进程的输出要特别处理，不作改动，前者为标准输入，后者为标准输出。

具体实现时，首先调用 pipe 函数生成多个管道，将其文件描述符存储在 pfid[MAXNUM][2]数组中。其中 pfid[i][0]为第 i 个进程的读端，pfid[i][1]为第 i 个进程的写端。然后通过 dup2 函数实现重定向，并用 close 关闭无用的管道。在调用 exec_command 执行完命令段后，还应调用 close 函数关闭对应的读写端，避免阻塞。

（7）非法命令处理

在调用各个库函数时，对各种异常情况通过 if 语句进行判断，若其返回值异常，则可以认为其在执行时发生了错误。此时首先调用 perror(line)函数将该命令和错误信息输出到标准错误流 stderr（此处 line 存储了命令行），

然后再用 fopen 和 fprintf 的方式将 line 和错误信息 strerror(errno)输出到.shelldemo_err.log 文件中。

2. 编译、调试、执行结果展示

（1）编译

使用 make 命令编译的结果如下页图所示。可见运行 make 前无 shelldemo 程序，运行后编译出了可执行程序，并且试验执行成功，说明编译成功。makefile 中的内容见下代码块。

```
shelldemo: main.o input.o split.o execute.o built-in.o external.o
error.o complete.o
    gcc main.o input.o split.o execute.o built-in.o external.o error.o
complete.o -o shelldemo -lreadline

main.o: main.c shelldemo.h
    gcc -c main.c

input.o: input.c shelldemo.h
    gcc -c input.c

split.o: split.c shelldemo.h
    gcc -c split.c

execute.o: execute.c shelldemo.h
    gcc -c execute.c

built-in.o: built-in.c shelldemo.h
    gcc -c built-in.c

external.o: external.c shelldemo.h
    gcc -c external.c

error.o: error.c shelldemo.h
    gcc -c error.c

complete.o: complete.c shelldemo.h
    gcc -c complete.c -lreadline

clean_o:
    rm *.o

clean_all:
    rm shelldemo *.o
```

```
cyq@cyq-virtual-machine:~/proj$ ls
built-in.c  complete.c  error.c  execute.c  external.c  input.c  main.c  makefile  shelldemo.h  split.c  test.txt
cyq@cyq-virtual-machine:~/proj$ make
gcc -c main.c
gcc -c input.c
gcc -c split.c
gcc -c execute.c
gcc -c built-in.c
gcc -c external.c
gcc -c error.c
gcc -c complete.c -lreadline
gcc main.o input.o split.o execute.o built-in.o external.o error.o complete.o -o shelldemo -lreadline
cyq@cyq-virtual-machine:~/proj$ ./shelldemo

=========== Welcome to 3122008883_cyq's shelldemo! ===========

cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ quit

================= The shelldemo has quitted. =================
```

（2）调试

首先使用 gcc -g main.o input.o split.o execute.o built-in.o external.o error.o complete.o -o shelldemo -lreadline 命令编译生成可供 gdb 调试的可执行程序。然后使用 gdb shelldemo 命令进入调试，如下图所示，出现了“(gdb)”提示词，进入成功。

```
cyq@cyq-virtual-machine:~/proj$ gcc -g main.o input.o split.o execute.o built-in.o external.o error.o complete.o -o shelldemo -lreadline
cyq@cyq-virtual-machine:~/proj$ gdb shelldemo
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from shelldemo...
(No debugging symbols found in shelldemo)
(gdb)
```

然后利用 break 命令在 exec_line 函数入口处设置断点，并用 display 命令显示变量 cmdcnt 的值（即命令段的个数）。用 info 命令可见断点设置成功。设置完毕后，使用 run 命令开始程序的运行。

```
(gdb) break exec_line
Breakpoint 1 at 0x35f9
(gdb) display /d (int)cmdcnt
1: /d (int)cmdcnt = 0
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000000035f9 <exec_line+8>
(gdb) run
Starting program: /home/cyq/proj/shelldemo
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

=========== Welcome to 3122008883_cyq's shelldemo! ===========

cyq@cyq-virtual-machine:[/home/cyq/proj]>>$
```

输入一行命令 cat test.txt | sort | tee | head -3 后回车，可见程序按预期停止在了 exec_line() 函数的入口处，此时显示 cmdcnt = 4，等于命令段的数量，符合预期。

```
=========== Welcome to 3122008883_cyq's shelldemo! ===========

cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cat test.txt | sort | tee | head -3

Breakpoint 1, 0x00005555555575f9 in exec_line ()
1: /d (int)cmdcnt = 4
(gdb)
```

然后使用 continue 命令使程序继续运行。输出了正确的结果。

```
(gdb) continue
Continuing.
[Detaching after fork from child process 3599]
[Detaching after fork from child process 3601]
[Detaching after fork from child process 3602]
[Detaching after fork from child process 3605]
1
2
3
```

接下来输入一行命令 echo2 ... 后回车，可见程序按预期停止在了 exec_line() 函数的入口处，此时显示 cmdcnt = 1，等于命令段的数量，符合预期。输入 continue 后，程序继续执行，也输出了预期的结果。

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ echo2 3122008883 cyq hello world

Breakpoint 1, 0x00005555555575f9 in exec_line ()
1: /d (int)cmdcnt = 1
(gdb) continue
Continuing.
3122008883 cyq hello world
```

接下来输入 quit 退出程序，可见 gdb 提示了程序正常退出。最后再在 gdb 中输入 quit 退出 gdb。至此，使用 gdb 调试的展示完毕。

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ quit

================= The shelldemo has quitted. =================

[Inferior 1 (process 3596) exited normally]
(gdb) quit
cyq@cyq-virtual-machine:~/proj$
```

（3）执行

1）内置命令

①.pwd2

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ pwd2
/home/cyq/proj
```

如图，pwd2 显示了当前工作路径。

②.cd2

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cd2
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cd2 ..
cyq@cyq-virtual-machine:[/home/cyq]>>$ cd2 proj
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cd2 -
cyq@cyq-virtual-machine:[/home/cyq]>>$ cd2 proj
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cd2 .
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$
```

如图，若 cd2 的参数为空或.，则返回到当前目录；若参数为..或- ，则后退到上级目录，符合作业要求。

③.ls2

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls2
File:   shelldemo
File:   makefile
File:   complete.c
File:   shelldemo.h
File:   main.c
File:   built-in.c
File:   external.c
Dir:    ./
File:   split.c
File:   execute.c
File:   error.c
Dir:    ../
File:   input.c
File:   test.txt
File:   .shelldemo_err.log
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls2 ../proj
File:   shelldemo
File:   makefile
File:   complete.c
File:   shelldemo.h
File:   main.c
File:   built-in.c
File:   external.c
Dir:    ./
File:   split.c
File:   execute.c
File:   error.c
Dir:    ../
File:   input.c
File:   test.txt
File:   .shelldemo_err.log
```

如图，显示了参数路径下的所有文件和文件夹，以"File"和"Dir"字符串区分显示。

④.touch2

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ touch2 touched_file.txt
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls
built-in.c  error.c    external.c  main.c    shelldemo    split.c   touched_file.txt
complete.c  execute.c  input.c     makefile  shelldemo.h  test.txt
```

如图，执行 touch2 touched_file.txt 后，通过 ls 命令可以发现，目录下多出了 file.txt 文件，符合预期。

⑤.echo2

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ echo2 hello,world!
hello,world!
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ echo2 c y q 1 2 3 4
c y q 1 2 3 4
```

如图，通过 echo2 命令可以实现输出一个或多个字符串。

⑥.cat2

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cat2 shelldemo.h
1       #include <stdio.h>
2       #include <stdlib.h>
3       #include <string.h>
4       #include <errno.h>
5       #include <unistd.h>
6       #include <dirent.h>
7       #include <fcntl.h>
8       #include <pwd.h>
9       #include <unistd.h>
10      #include <sys/stat.h>
11      #include <sys/types.h>
12      #include <sys/wait.h>
13      #include <readline/readline.h>
14      #include <readline/history.h>
15
16      #define MAXNUM 32
17      #define SIZE 512
18      #define NONE_COLOR      "\033[0m"
19      #define LIGHT_GREEN     "\033[1;32m"
20      #define LIGHT_BLUE      "\033[1;34m"
```

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cat2 test.txt
1       1
2       2
3       3
4       4
5       5
```

如图，通过 cat2 命令展示了 shelldemo.h 的部分内容（限于篇幅没有截图完整）和 test.txt。按作业要求，在文件内容前添加了行号。test.txt 内容为升序排序的五个数字，后文测试其他命令时会再用到。

⑦.cp2

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cp2 non_exist_file new_file
cp2 non_exist_file new_file: No such file or directory
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cp2 test.txt new_file
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cat new_file
1
2
3
4
5
```

如图，通过 cp2 将 test.txt 复制为了 new_file，并通过 cat 验证了其正确性。若命令参数所代表的文件不存在，如 not_exist_file，则会提示错误，错误信息在标准错误流中进行了输出。

⑧.rm2

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls
built-in.c  error.c    external.c  input.c  makefile  shelldemo    split.c   touched_file.txt
complete.c  execute.c  folder      main.c   new_file  shelldemo.h  test.txt
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ rm2 new_file
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ rm2 -r folder
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls
built-in.c  error.c    external.c  main.c    shelldemo    split.c   touched_file.txt
complete.c  execute.c  input.c     makefile  shelldemo.h  test.txt
```

如图是尝试通过 rm2 new_file 和 rm2 -f folder 删除文件和文件夹的截图。在删除前后都用 ls 命令展示了当前目录下的文件列表，对比可见删除是成功的。

⑨.rename2

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls
built-in.c  error.c    external.c  main.c    shelldemo    split.c   touched_file.txt
complete.c  execute.c  input.c     makefile  shelldemo.h  test.txt
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ rename2 touched_file.txt new_name.txt
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls
built-in.c  error.c    external.c  main.c    new_name.txt  shelldemo.h  test.txt
complete.c  execute.c  input.c     makefile  shelldemo     split.c
```

如图是使用 rename2 将 touched_file.txt 重命名为 new_name.txt 的截图。通过 ls 命令列出了前后的文件，从红框标记处可见，重命名成功，符合预期。

⑩.history2

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ history2
1       ls
2       echo2 hello,world!
3       echo2 c y q 1 2 3 4
4       cat test.txt
5       sort -r < test.txt > test.txt
6       cat test.txt
7       cat test.txt
8       tee > test.txt
9       cat test.txt
10      cat2 touched_file.txt
11      cat2 touched_file.txt
12      rm touched_file.txt
13      touch touched_file.txt
14      cat2 touched_file.txt
15      clear
16      cat2 makefile
17      cat2 shelldemo.h
18      cat2 test.txt
19      cp2 non_exist_file new_file
20      cp2 test.txt new_file
21      cat new_file
22      clear
23      ls
24      rm2 new_file
25      rm2 -r folder
26      ls
27      rename2 touched_file.txt new_name.txt
28      ls
29      non_exist_command
30      history2
```

如图，通过 history2 展示了本次运行中所有输入过的历史命令，包括非法命令也会录入其中。

⑪.quit

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ quit

================= The shelldemo has quitted. =================

cyq@cyq-virtual-machine:~/proj$
```

2）外部命令

```
=========== Welcome to 3122008883_cyq's shelldemo! ===========

cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls -l /home
总计 4
drwxr-x--- 20 cyq cyq 4096 12月 10 01:25 cyq
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls -la
总计 160
drwxrwxr-x  2 cyq cyq  4096 12月 10 02:15 .
drwxr-x--- 20 cyq cyq  4096 12月 10 01:25 ..
-rw-rw-r--  1 cyq cyq  2682 12月 10 01:13 built-in.c
-rw-rw-r--  1 cyq cyq   592 12月  9 12:20 complete.c
-rw-rw-r--  1 cyq cyq   229 12月  9 12:11 error.c
-rw-rw-r--  1 cyq cyq  4296 12月  9 12:19 execute.c
-rw-rw-r--  1 cyq cyq   405 12月  9 12:14 external.c
-rw-rw-r--  1 cyq cyq   861 12月  9 12:18 input.c
-rw-rw-r--  1 cyq cyq  1025 12月  9 12:38 main.c
-rw-rw-r--  1 cyq cyq   648 12月  9 12:40 makefile
-rw-rw-r--  1 cyq cyq     0 12月 10 02:09 new_name.txt
-rwxrwxr-x  1 cyq cyq 31680 12月 10 01:33 shelldemo
-rw-rw-r--  1 cyq cyq 66326 12月 10 02:16 .shelldemo_err.log
-rw-rw-r--  1 cyq cyq  1182 12月  9 12:38 shelldemo.h
-rw-rw-r--  1 cyq cyq  2576 12月  9 12:17 split.c
-rwxrwxrwx  1 cyq cyq    10 12月 10 02:08 test.txt
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$
```

如图是 `ls -l /home` 和 `ls -la` 的执行结果。

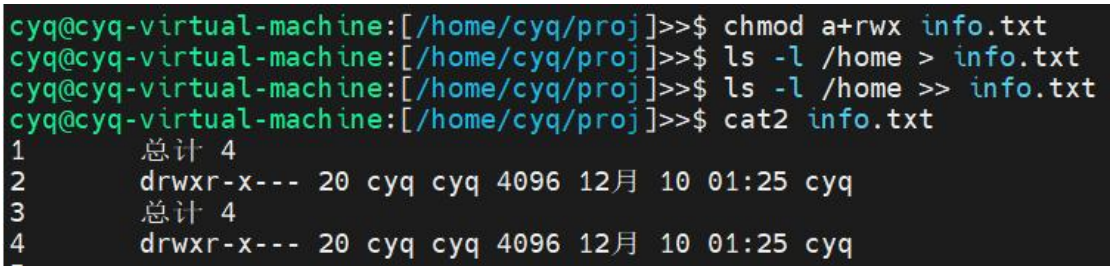
```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ head -10 non_exist
head: 无法以读模式打开 'non_exist': 没有那个文件或目录
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ head -10 built-in.c
#include "shelldemo.h"

void pwd2()
{
    char cwd[SIZE];
    if(getcwd(cwd, sizeof(cwd)) != NULL)
        printf("%s\n", cwd);
    else
        deal_error();
}
```

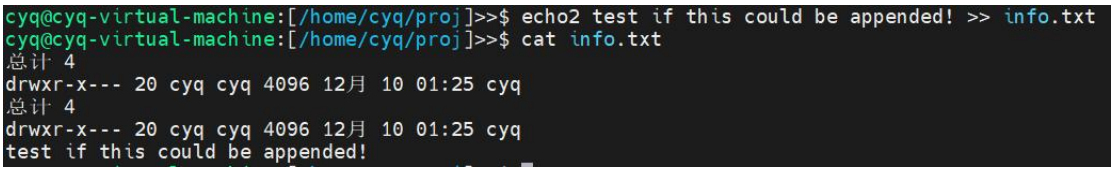
如图是 `head -10` 的执行结果。并且由图可见，若执行时出现文件不存在等错误，也会在标准错误流中输出对应的报错信息。

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ mkdir folder
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls
built-in.c  error.c    external.c  input.c  makefile      shelldemo    split.c
complete.c  execute.c  folder      main.c   new_name.txt  shelldemo.h  test.txt
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ rm folder
rm: 无法删除 'folder': 是一个目录
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ rm -r folder
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ rm new_name.txt
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ ls
built-in.c  error.c    external.c  main.c    shelldemo    split.c
complete.c  execute.c  input.c     makefile  shelldemo.h  test.txt
```

如图是 mkdir、ls、rm -f 等命令的组合展示。先创建了 folder 文件夹，再尝试通过 rm 命令删除它，可见在没有添加-f 选项时出现了报错，添加后成功删除。在删除前后用 ls 进行了展示，验证了其正确性。

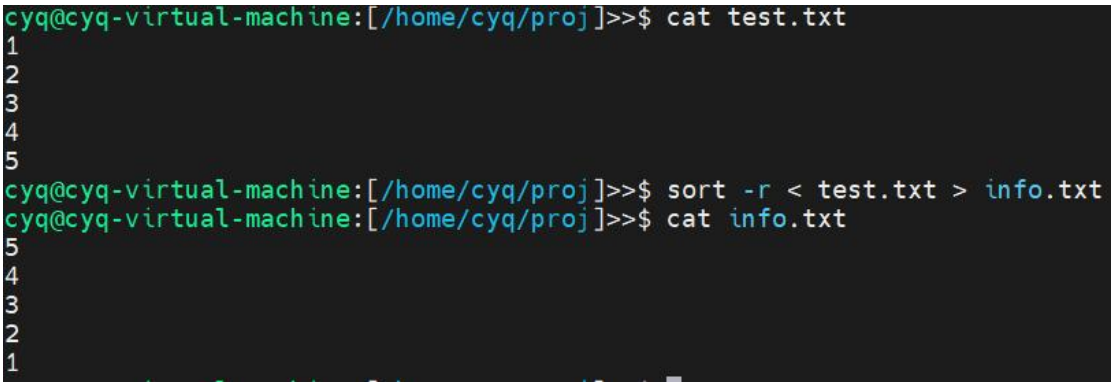
3）重定向文件流命令





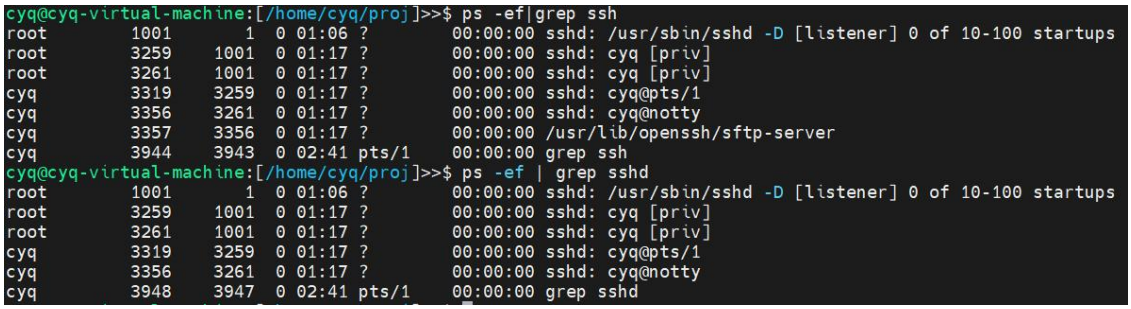
由于某些文件由子进程进行创建，可能会出现没有进一步操作的权限的情况。这时需要用 chmod a+rwx info.txt 等命令将其权限打开后才能进一步测试。这一权限操作比较繁琐，所以这也是后续程序改进的方向之一。

如图所示，是仿照大作业要求文档进行的测试。首先通过重定向符>将 ls -l /home 的结果输出到 info.txt 文件中。然后通过重定向符>>将外部命令 ls -l /home 的结果，再次写入到 info.txt 文件中，并以文末续写的方式写入。通过 cat2 info.txt 可以看到实现了>>应有的效果。然后再通过>>将内置命令 echo2 的结果也写入了 info.txt 的文末，可见也符合预期。



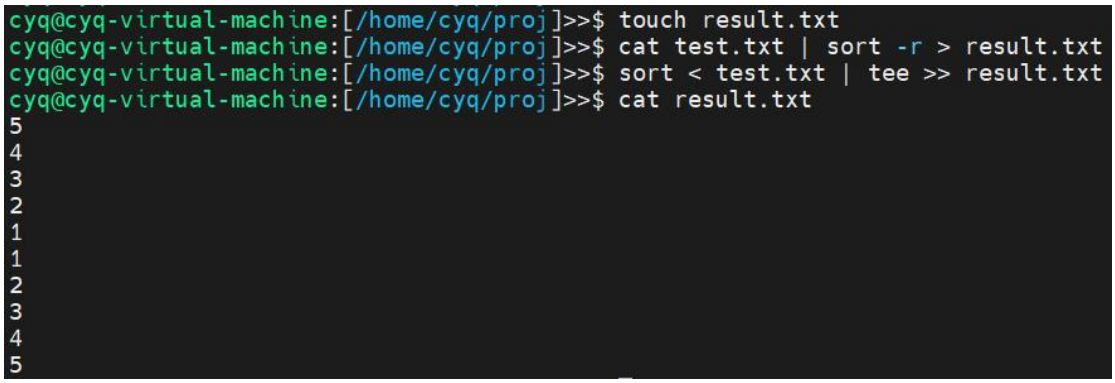
最后测试了在同一命令段中同时出现输入重定向和输出重定向的情况，可见 sort -r < test.txt 的结果正确写入到了 info.txt 文件中。

4）管道命令



如图，是 shelldemo 大作业要求文档的示例命令，可见 ps -ef 命令段的输出被当作了 grep ssh 和 grep sshd 命令段的输入，输出结果符合预期。并且无论管道符|两侧是否有空格，都能正确分割识别。

下面是本程序特色之一，管道命令和重定向文件流命令混合使用的截图。实现了诸如 cat test.txt | sort >> result.txt、sort < test.txt | tee > result.txt 等功能。特色说明文档中会有更详细的介绍。



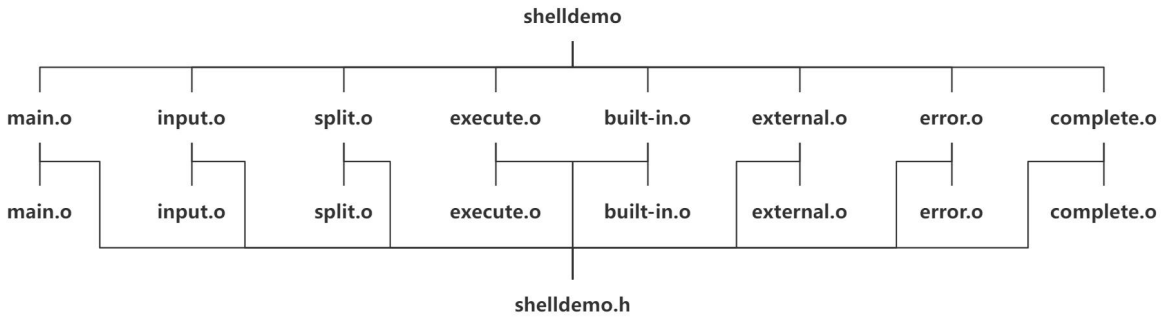
下面是本程序特色之二，多管道命令符嵌套连续使用的截图。实现了诸如 cat2 shelldemo.h | tail -20 | head -5 | sort -r | tee 等多管道连续使用的功能。

5）非法信息处理

```
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ chmod a-rwx info.txt
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cat2 info.txt
cat2 info.txt: Permission denied
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cp2 non_exist_file 1.txt
cp2 non_exist_file 1.txt: No such file or directory
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ non_command
non_command: No such file or directory
cyq@cyq-virtual-machine:[/home/cyq/proj]>>$ cat .shelldemo_err.log
cat2 info.txt: Permission denied
cp2 non_exist_file 1.txt: No such file or directory
non_command: No such file or directory
```

如图所示，不合法信息处理在标准错误流中进行了输出。同时也被记录到了 .shelldemo_err.log 文件中，通过 cat 命令展示发现符合预期。

3．文件关系依赖图



二、shell 脚本程序

1. 设计思路说明

该脚本程序较第一题简单许多，用 bash 语法中的 while 循环和 case 选择即可实现。考虑到提示词较复杂，还需要显示日期，所以没有选用 select 语句来实现。

总而言之，总体设计思路是通过 while true 的死循环不断显示日期和菜单，直到用户输入 6 退出。在死循环中，通过 echo 显示日期和菜单，其中日期用 %y/%m/%d 的方式得到。然后用 read -p 实现带提示词的输入，接着将输入的 chioce 变量通过 case 进行分支操作，执行对应的命令，详见代码。

2. 源代码

```
#!/bin/bash
while true
```

```
do
    # 显示日期和菜单
    echo "--------------------------------"
    echo " 3122008883  Linux " `date +%y/%m/%d`
    echo "--------------------------------"
    echo "1) show all threads"
    echo "2) show current users"
    echo "3) show files in current dir"
    echo "4) show the computer's name"
    echo "5) show the kernel version"
    echo "6) quit"
    echo "--------------------------------"

    # 显示提示词，输入选项
    read -p "Please select: " choice

    # 根据不同的选项执行
    case $choice in
        1) ps -eLf;;  # 显示所有线程
        2) whoami;;   # 显示当前用户
        3) ls -la;;   # 展示当前目录下的文件
        4) hostname;; # 显示主机名
        5) uname -a;; # 显示内核版本
        6) echo "The program has exited." # 退出程序
           break;;
        *) echo "Invalid input!";;         # 非法输入
    esac
done
```

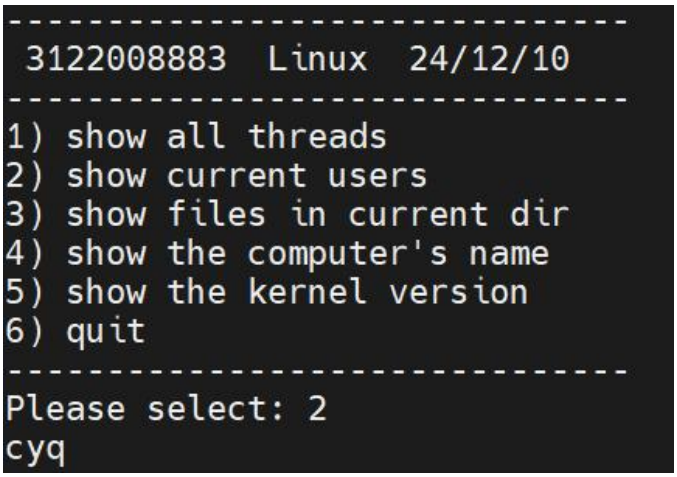
3．运行结果截图

（1）显示所有线程

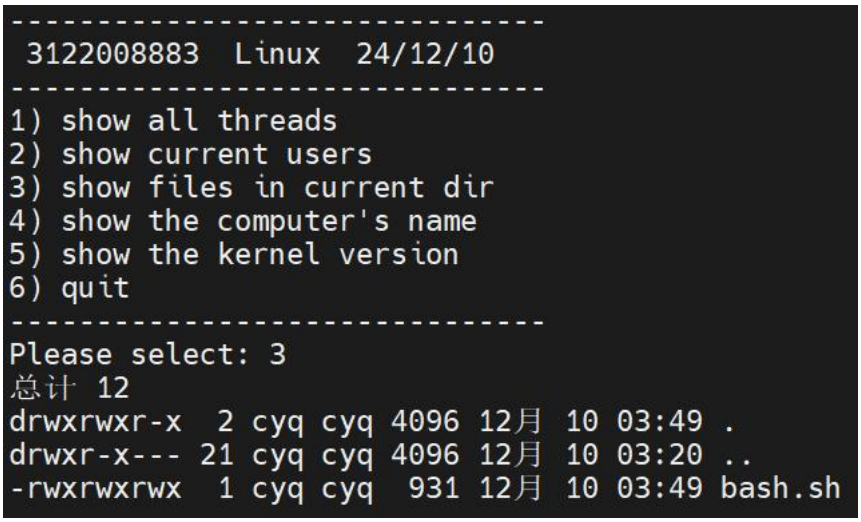
```
cyq@cyq-virtual-machine:~/bash$ ./bash.sh
--------------------------------
 3122008883  Linux  24/12/10
--------------------------------
1) show all threads
2) show current users
3) show files in current dir
4) show the computer's name
5) show the kernel version
6) quit
--------------------------------
Please select: 1
UID          PID    PPID     LWP  C NLWP STIME TTY          TIME CMD
root           1       0       1  0    1 01:06 ?        00:00:03 /sbin/init auto noprompt splash
root           2       0       2  0    1 01:06 ?        00:00:00 [kthreadd]
root           3       2       3  0    1 01:06 ?        00:00:00 [pool_workqueue_release]
root           4       2       4  0    1 01:06 ?        00:00:00 [kworker/R-rcu_g]
root           5       2       5  0    1 01:06 ?        00:00:00 [kworker/R-rcu_p]
root           6       2       6  0    1 01:06 ?        00:00:00 [kworker/R-slub_]
root           7       2       7  0    1 01:06 ?        00:00:00 [kworker/R-netns]
```

由于线程太多，这里只展示前面几条。

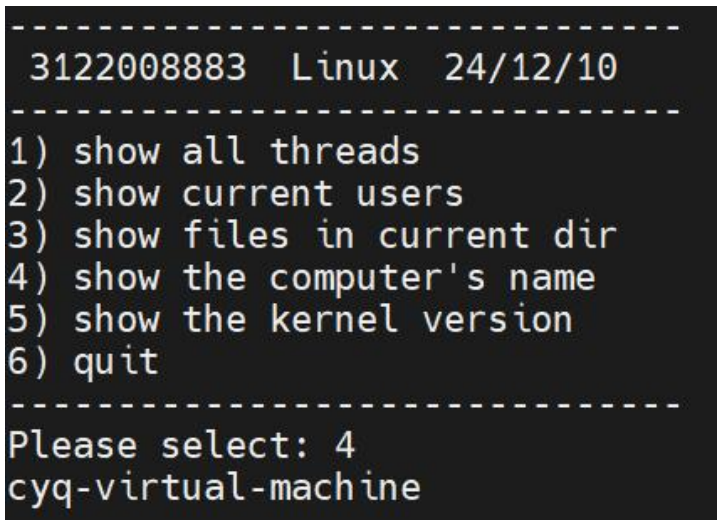
（2）显示当前用户



（3）显示当前目录下的文件



（4）显示主机名



（5）显示内核版本

```
------------------------------
 3122008883  Linux  24/12/10
------------------------------
1) show all threads
2) show current users
3) show files in current dir
4) show the computer's name
5) show the kernel version
6) quit
------------------------------
Please select: 5
Linux cyq-virtual-machine 6.8.0-49-generic #49~22.0
4.1-Ubuntu SMP PREEMPT_DYNAMIC Wed Nov  6 17:42:15
UTC 2 x86_64 x86_64 x86_64 GNU/Linux
```

（6）退出程序

```
------------------------------
 3122008883  Linux  24/12/10
------------------------------
1) show all threads
2) show current users
3) show files in current dir
4) show the computer's name
5) show the kernel version
6) quit
------------------------------
Please select: 6
The program has exited.
cyq@cyq-virtual-machine:~/bash$
```

（7）非法输入处理

```
------------------------------
 3122008883  Linux  24/12/10
------------------------------
1) show all threads
2) show current users
3) show files in current dir
4) show the computer's name
5) show the kernel version
6) quit
------------------------------
Please select: hello
Invalid input!
```