POWERPOINT PRESENTATION TEMPLATE

My_Shell项目答辩

The Powerpoint template is a clean and simple template that can be used for personal and business use. Its clean approach makes it easy for the user to print. It comes in both standard and wide versions in ppt and pptx formats.

答辩人：杨雯茜 尹秋菊

01

项目当前开发状态

项目当前的开发状态

完成度：

· My Shell 项目已经完成了基本功能的实现，包括命令解析、执行、I/O 重定向、管道操作以及错误处理。

· 项目核心模块已通过初步测试，能够正确处理大多数常见命令和操作。

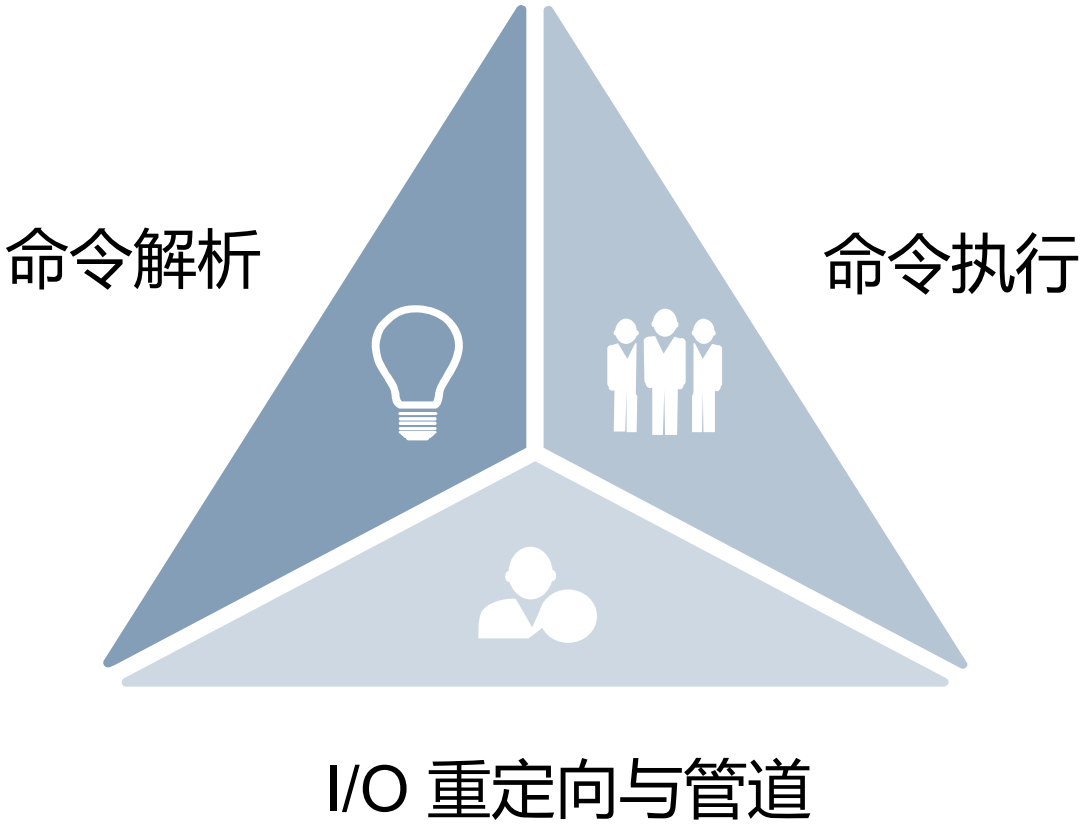
质量：

· 代码结构清晰，模块划分合理，具备良好的可读性和可维护性。

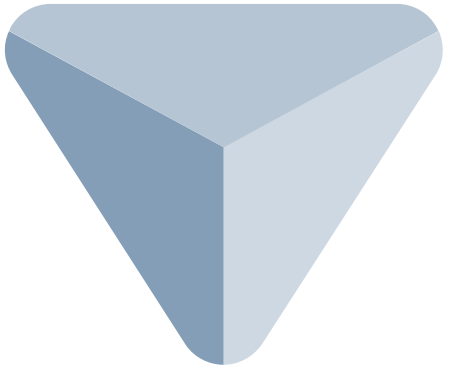
· 实现了基本的错误处理机制，能够处理常见的异常情况。

· 初步测试覆盖了常见的命令执行和文件操作，确保了功能的正确性。

核心内容



技术重难点

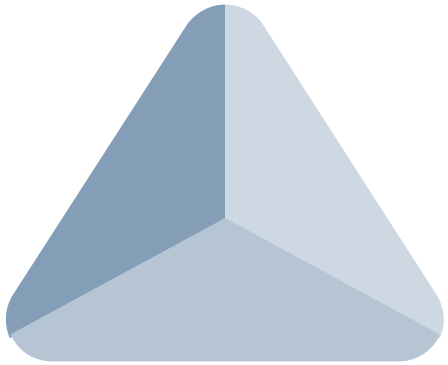


1

命令解析

· 准确解析用户输入的命令，并正确处理各种特殊字符和操作

· 考虑不同操作系统的特殊字符处理方式，以及用户输入的合法性检查。

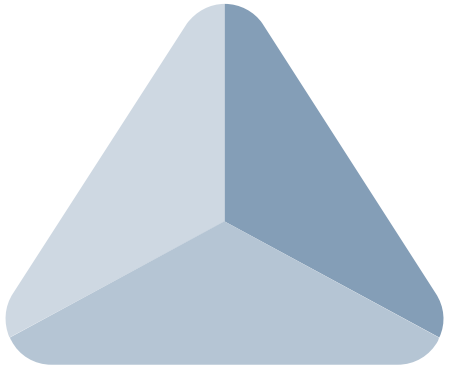


2

进程管理

· 高效地创建和管理子进程，避免资源泄漏和进程僵死。

· 确保父进程能够正确等待子进程完成，并处理子进程的退出状态。



3

I/O 重定向与管道

· 实现多层次的重定向和管道操作，确保命令的输入输出顺畅。

· 实现多层次的重定向和管道操作，确保命令的输入输出顺畅。

运行结果展示

```
yin@yin-VirtualBox: ~/My_Shell
yin@yin-VirtualBox:~/My_Shell$ g++ -o my_shell my_shell.cpp
yin@yin-VirtualBox:~/My_Shell$ ./my_shell
Welcome to The Shell!
/home/yin/My_Shell $ ls
my_shell  my_shell.cpp  test_script.sh
/home/yin/My_Shell $ pwd
/home/yin/My_Shell
/home/yin/My_Shell $ mkdir 6_29
/home/yin/My_Shell $ ls
6_29  my_shell  my_shell.cpp  test_script.sh
/home/yin/My_Shell $ mv 6_29 test
/home/yin/My_Shell $ ls
my_shell  my_shell.cpp  test  test_script.sh
/home/yin/My_Shell $ cd test
/home/yin/My_Shell/test $ cd ..
/home/yin/My_Shell $
```

1-文件操作命令

```
/home/yin/My_Shell $ touch hello.txt
/home/yin/My_Shell $ vi hello.txt
/home/yin/My_Shell $ cat hello.txt
hello
hello world
/home/yin/My_Shell $ head -n 1 hello.txt
hello
/home/yin/My_Shell $
```

1-文件查看命令

```
/home/yin/My_Shell $ df -h
文件系统        容量  已用  可用 已用% 挂载点
udev            1.4G     0  1.4G    0% /dev
tmpfs           280M  3.1M  277M    2% /run
/dev/sda5        78G  9.9G   64G   14% /
tmpfs           1.4G     0  1.4G    0% /dev/shm
tmpfs           5.0M  4.0K  5.0M    1% /run/lock
tmpfs           1.4G     0  1.4G    0% /sys/fs/cgroup
/dev/loop0      128K  128K     0  100% /snap/bare/5
/dev/loop1       62M   62M     0  100% /snap/core20/1328
/dev/loop3       55M   55M     0  100% /snap/snap-store/558
/dev/loop4      249M  249M     0  100% /snap/gnome-3-38-2004/99
/dev/loop5       66M   66M     0  100% /snap/gtk-common-themes/1519
/dev/sda1       511M  4.0K  511M    1% /boot/efi
tmpfs           280M  112K  280M    1% /run/user/1000
/dev/sr0         61M   61M     0  100% /media/yin/VBox_GAs_6.1.36
/dev/loop6       39M   39M     0  100% /snap/snapd/21759
/dev/loop7       64M   64M     0  100% /snap/core20/2318
/dev/loop8       92M   92M     0  100% /snap/gtk-common-themes/1535
/dev/loop9      350M  350M     0  100% /snap/gnome-3-38-2004/143
/home/yin/My_Shell $ free -h
               总计         已用        空闲      共享    缓冲/缓存    可用
内存:          2.7Gi       718Mi       634Mi        14Mi       1.4Gi       1.8Gi
交换:          2.0Gi       237Mi       1.8Gi
/home/yin/My_Shell $ uptime
 15:14:36 up 21:26,  1 user,  load average: 0.03, 0.08, 0.04
/home/yin/My_Shell $
```

1-系统信息命令

```
/home/yin/My_Shell $ top

top - 15:15:20 up 21:27,  1 user,  load average: 0.01, 0.06, 0.03
任务: 216 total,   1 running, 215 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.1 us,  4.2 sy,  0.0 ni, 93.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 s
MiB Mem :   2798.5 total,    634.8 free,    718.9 used,   1444.8 buff/cache
MiB Swap:   2048.0 total,   1810.0 free,    238.0 used.   1878.9 avail Mem

  进程号 USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    790 yin       20   0  285732  56204  20964 S   6.7   2.0   6:12.38 Xorg
   1106 yin       20   0 4800820 271752  71588 S   6.7   9.5  12:12.58 gnome-+
 202414 yin       20   0   14984   3940   3416 R   6.7   0.1   0:00.01 top
      1 root      20   0  169052  11420   7240 S   0.0   0.4   0:27.42 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.10 kthrea+
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.60 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_pa+
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworke+
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_per+
      9 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_ta+
     10 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_ta+
     11 root      20   0       0      0      0 S   0.0   0.0   0:02.10 ksofti+
```

1-进程管理命令

运行结果展示

```
/home/yin/My_Shell $ cat test_script.sh
echo " Hello from script!"
echo " Current Directory: $(pwd)"
ls -l
/home/yin/My_Shell $ ./my_shell test_script.sh
Welcome to The Shell!
Hello from script!
Current Directory: /home/yin/My_Shell
总用量 120
-rw-rw-r-- 1 yin yin    18 6月  29 15:13 hello.txt
-rwxrwxr-x 1 yin yin 95928 6月  29 15:09 my_shell
-rwx------ 1 yin yin  9149 6月  29 15:09 my_shell.cpp
drwxrwxr-x 2 yin yin  4096 6月  29 15:10 test
-rw-rw-r-- 1 yin yin    67 6月  29 14:24 test_script.sh
/home/yin/My_Shell $
```

2-shell 编程的功能

```
yin@yin-VirtualBox: ~/My_Shell
yin@yin-VirtualBox:~/My_Shell$ ./my_shell
Welcome to The Shell!
/home/yin/My_Shell $ cat hello.txt | wc -w
3
/home/yin/My_Shell $ grep -o 'hello' hello.txt | wc -l
2
/home/yin/My_Shell $ cat hello.txt | grep 'world'
hello world
/home/yin/My_Shell $ cat hello.txt | tr ' ' '\n' | sort | uniq -c
      2 hello
      1 world
/home/yin/My_Shell $
```

3-管道管理

```
/home/yin/My_Shell $ ls > output.txt
/home/yin/My_Shell $ cat output.txt
fruit.txt
hello.txt
my_shell
my_shell.cpp
output.txt
test
test_script.sh
/home/yin/My_Shell $ echo 'hello' > hello.c
/home/yin/My_Shell $ cat hello.c
hello
/home/yin/My_Shell $ cat fruit.txt
banana
orange
apple
/home/yin/My_Shell $ wc -l < fruit.txt
3
/home/yin/My_Shell $ sort < fruit.txt > sort_fruit.txt
/home/yin/My_Shell $ cat sort_fruit.txt
apple
banana
orange
/home/yin/My_Shell $ echo "lemon" >> fruit.txt
/home/yin/My_Shell $ cat fruit.txt
banana
orange
apple
lemon
/home/yin/My_Shell $
```

3-重定向

02

分工及情况汇报

Research objectives

成员:杨雯茜的收获&经验教训

宏定义和头文件引入

Command 结构体定义

fetchFileName 函数实现

命令解析函数（上半部分）

代码注释和文档编写

· 深入理解宏定义和头文件的作用和引入方式
· 掌握了结构体的定义与成员变量的初始化
· 错误处理能力得到提升，尤其是在命令解析过程中的异常处理。

· 处理复杂命令解析时，错误处理的完备性对于代码稳定性至关重要。
· 通过测试用例的设计与执行，学习到了有效验证代码功能性的方法。

成员:尹秋菊的收获&经验教训

命令解析函数（下半部分）

追加输出文件处理

命令执行函数

主函数实现

代码测试和调试

· 掌握了命令解析的大体流程，包括特殊字符的处理逻辑。
· 加深对于 I/O 重定向和进程管理的理解。
· 在测试与调试过程中，掌握了快速修复代码的能力。

· 在处理复杂逻辑时，代码的模块化和函数分割能够显著提高代码的可维护性。
· 通过测试用例的设计与执行，学习到了有效验证代码功能性的方法。

POWERPOINT PRESENTATION TEMPLATE

My_Shell项目答辩

感谢您的垂听