分组情况：



**小组1**

核心代码：

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include "show/getCommand\_parse.h"

#include "show/getCommand\_readKey.h"

#include "show/showPrompt.h"

#include "process/fork.h"

#include "process/index\_of\_pipe.h"

#include "process/parent\_process.h"

#define MAXSIZE 4096

int main(int argc, char const \*argv[], char \*env[])

{

int ppipe[2];

int pid1, pid2;

char buff[MAXSIZE];

char \*buff\_2;

char \*args\_1[64];

char \*args\_2[64];

int arg\_nums = 0;

int pipe\_flag = 0;

printf("**\n\n\t** Welcom to Google Future Leaders very Power Shell System!**\n\n**");

while(1){

showPrompt();

readKey(buff);

if(!strcmp(buff, "**\n**")){

continue;

}

if((pipe\_flag = index\_of\_pipe(buff)) != -1){

pipe(ppipe);

buff[pipe\_flag] = 0; // instead '|' whit '\0'

buff\_2 = &buff[pipe\_flag + 1];

parse(buff\_2, args\_2);

}

parse(buff, args\_1);

if(!strcmp(args\_1[0],"exit")){

exit(0);

}

if((pid1 = fork()) == 0){

fork\_1(args\_1[0], args\_1, pipe\_flag, ppipe);

}

if(pipe\_flag != -1){

if((pid2 = fork()) == 0){

fork\_2(args\_2[0], args\_2, ppipe);

}

}

parent\_process(pid1, pid2, pipe\_flag, ppipe);

}

return 0;

}

编译过程：

cd show

gcc -fPIC -c getCommand\_parse.c

gcc -fPIC -c getCommand\_readKey.c

gcc -fPIC -c showPrompt.c

gcc -shared -o libshow.so getCommand\_parse.o getCommand\_readKey.o showPrompt.o

cp libshow.so ../

cd ../process

gcc -fPIC -c fork.c

gcc -fPIC -c index\_of\_pipe.c

gcc -fPIC -c parent\_process.c

gcc -shared -o libprocess.so fork.o parent\_process.o index\_of\_pipe.o

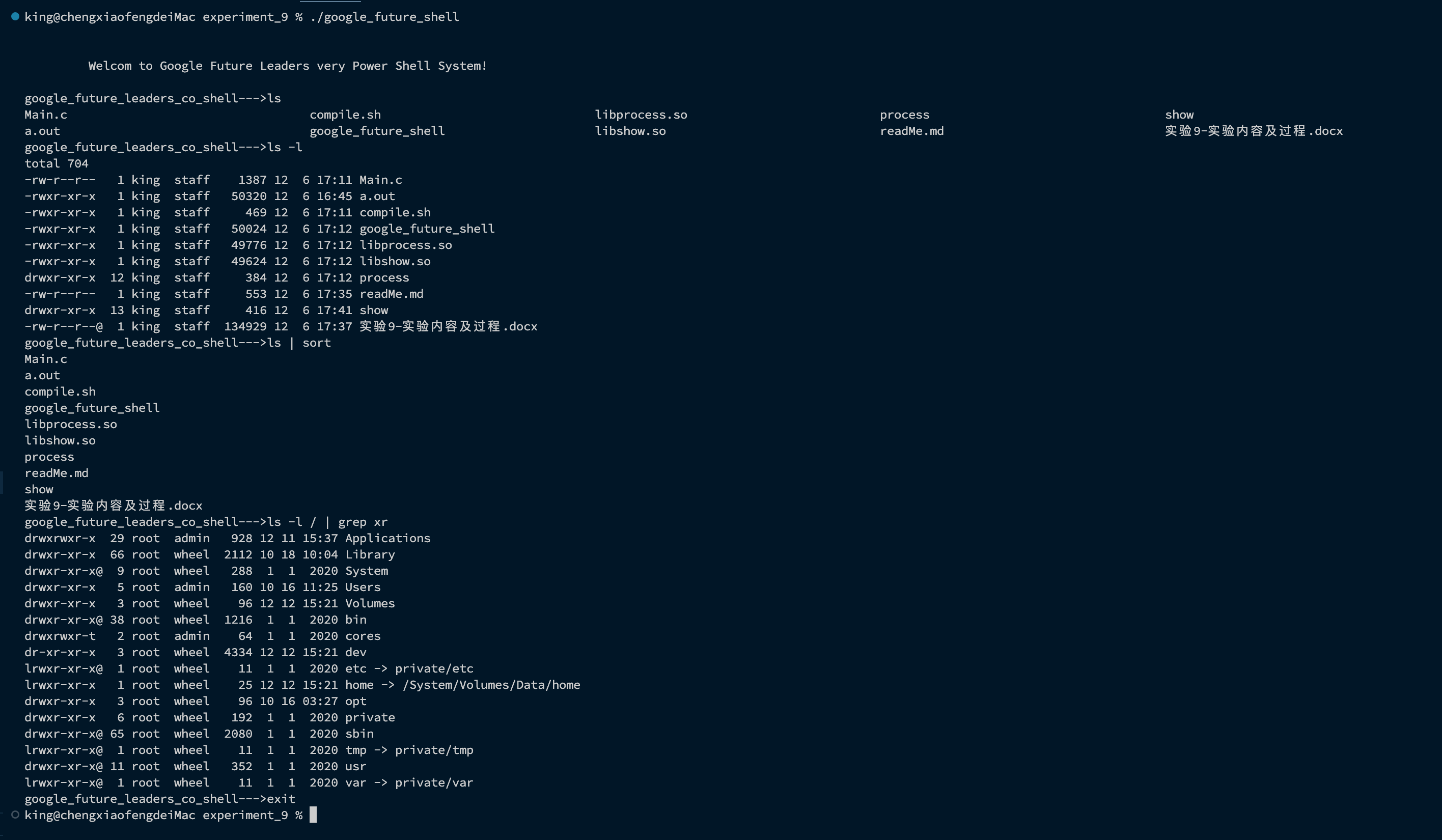
cp libprocess.so ../

cd ../

export LD\_LIBRARY\_PATH=.

gcc Main.c -o google\_future\_shell -L. -lprocess -lshow

测试：



**小组2**

接口：

void fork\_1(char\* command, char\*\* args, int pipe\_flag, int ppipe[]);

void fork\_2(char\* command, char\*\* args, int ppipe[]);

核心代码：

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include "fork.h"

void fork\_1(char\* command, char\*\* args, int pipe\_flag, int ppipe[]){

if(pipe\_flag != -1){

dup2(ppipe[1], 1);

close(ppipe[0]);

close(ppipe[1]);

}

execvp(command, args);

printf("could not execute %s\n", command);

exit(127);

}

void fork\_2(char\* command, char\*\* args, int ppipe[]){

dup2(ppipe[0], 0);

close(ppipe[0]);

close(ppipe[1]);

execvp(command, args);

printf("could not execute %s\n", command);

exit(127);

}

// int main(int argc, char const \*argv[])

// {

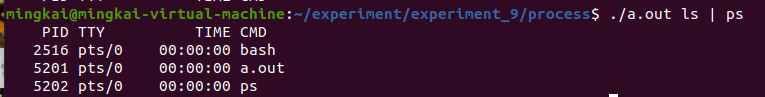
// char \*args[3] = {"ps"};

// fork\_1(\*args, args, -1);

// return 0;

// }

单元测试：



**小组3**

接口：

int index\_of\_pipe(char \*buff);

核心代码：

index\_of\_pipe.h

#ifndef \_INDEX\_PIPE\_

#define \_INDEX\_PIPE\_

int index\_of\_pipe(char \*buff);

#endif

index\_of\_pipe.c

#include "index\_of\_pipe.h"

int index\_of\_pipe(char \*buff){

int count = 0;

while(\*buff != '\0' && \*buff != '|'){

buff++;

count++;

}

if(\*buff == '\0'){

return -1;

}

return count;

}

**小组4**

接口：

void parent\_process(int pid1, int pid2, int pipe\_flag, int ppipe[]);

核心代码：

parent\_process.h

#ifndef \_PARENT\_PROCESS\_

#define \_PARENT\_PROCESS\_

void parent\_process(int pid1, int pid2, int pipe\_flag, int ppipe[]);

#endif

parent\_process.c

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include "parent\_process.h"

void wait\_child(int pid){

    if (pid = waitpid(pid, NULL, 0) < 0)

        printf(" pid error!");

}

void parent\_process(int pid1, int pid2, int pipe\_flag, int ppipe[]){

    if(pipe\_flag != -1){

        close(ppipe[0]);

        close(ppipe[1]);

    }

    wait\_child(pid1);

    if(pipe\_flag != -1){

        wait\_child(pid2);

    }

}

**小组5**

接口：

int parse(char \*buff, char\*\* args);

核心代码：

#include <stdio.h>

#include "getCommand\_parse.h"

int parse(char \*buff, char\*\* args){

int num = 0;

while(\*buff!='\0')

{

while((\*buff==' ')||(\*buff=='\t'||(\*buff=='\n')))

\*buff++='\0';

\*args++=buff;

++num;

while((\*buff!='\0')&&(\*buff!=' ')&&(\*buff!='\t')&&(\*buff!='\n'))

buff++;

}

\*(--args)='\0';

return num - 1;

}

// int main(int argc, char const \*argv[])

// {

// char buff[] = "ls -l /root/home/";

// char \*args[64];

// int argsnum = parse(buff, args);

// printf("the args num is %d\n", argsnum);

// for(int i = 0; i < argsnum; i++){

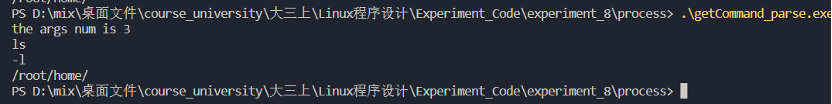
// printf("%s\n", args[i]);

// }

// return 0;

// }

单元测试：



**小组6**

接口：

void readKey(char\* buff);

核心代码：

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include "getCommand\_readKey.h"

#define MAXSIZE 4096

void readKey(char\* buff){

char\* ret;

ret = fgets(buff, MAXSIZE, stdin);

if(ret == NULL){

printf("error occurs!\n");

exit(1);

}

if(buff[strlen(buff) - 1] == "\n"){

buff[strlen(buff) - 1] = 0;

}

}

// int main(int argc, char const \*argv[])

// {

// char buff[MAXSIZE];

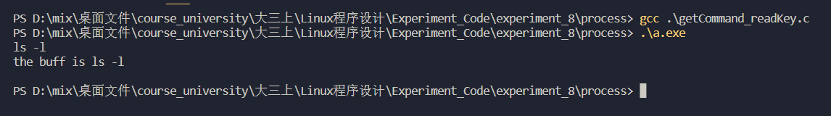
// readKey(buff);

// printf("the buff is %s\n", buff);

// return 0;

// }

单元测试：



**小组7**

接口：

void showPrompt();

核心代码：

#include <stdio.h>

#include "showPrompt.h"

void showPrompt(){

printf("google\_future\_leaders\_co\_shell--->");

}

// int main(int argc, char const \*argv[])

// {

// showPrompt();

// return 0;

// }

单元测试：

