分组情况：



**小组1**

接口：

void readKey(char\* buff){

printf("读取键盘输入一行到buff\n");

}

核心代码：

#include <stdio.h>

#include <unistd.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);

}

}

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

{

char buff[MAXSIZE];

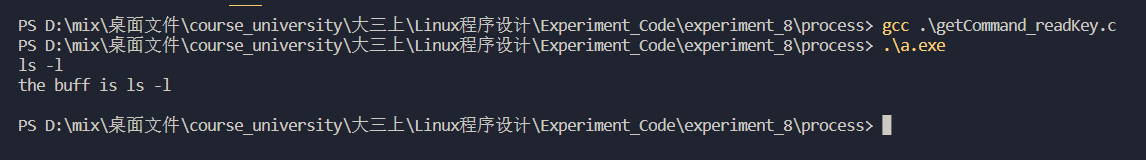
readKey(buff);

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

return 0;

}

单元测试：



**小组2**

接口：

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

printf("将buff解析为字符串数组\n");

}

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;

}

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++){

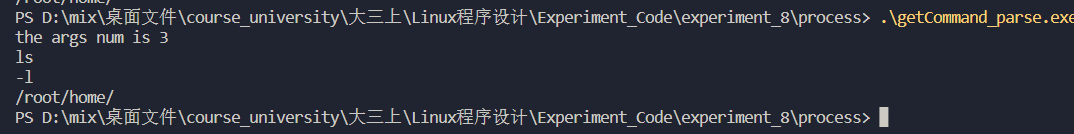
printf("%s**\n**", args[i]);

}

return 0;

}

单元测试：



**小组3**

接口：

void showPrompt(){

printf("显示提示信息！\n");

}

核心代码：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "showPrompt.h"

int main(void showPrompt())

{

char \*rt;

char buf[MAXLINE];

while(1)

  {

printf("@");//显示命令提示符

rt = fgets(buf,MAXLINE,stdin);

if(rt == NULL)

{

printf("fgets error\n");//创建失败

exit(1);

}

if(!strcmp(buf,"\n"))

{

printf("@");

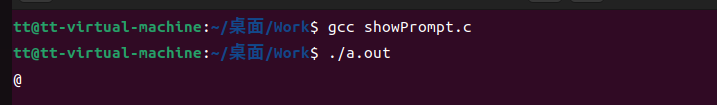
continue;//循环结束

}

  }

}

单元测试：



**小组4**

接口：

void childDo(char\* command, char\*\* args){

printf("使用exec系列函数运行命令\n");

}

核心代码：

#include <stdio.h>

#include <unistd.h>

#include "childDo.h"

void childDo(char\* command, char\*\* args){

    execvp(command, args);

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

    exit(127);

}

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

{

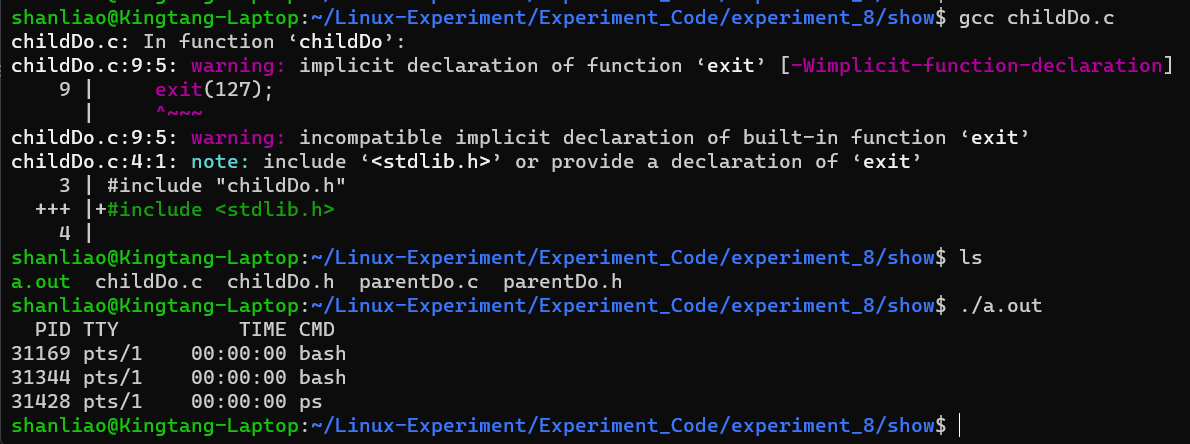
    char \*args[3] = {"ps", "-l", "./"};

    childDo(\*args, args);

    return 0;

}

单元测试：



**小组5**

接口：

void parentDo(int pid, int status){

printf("父进程做的事\n");

}

核心代码：

void parentDo(int pid, int status)

{

    printf("父进程做的事情");

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

        printf(" pid error!");

}

Main.c

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include "process/getCommand\_parse.h"

#include "process/getCommand\_readKey.h"

#include "process/showPrompt.h"

#include "show/childDo.h"

#include "show/parentDo.h"

#define MAXSIZE 4096

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

{

int pid;

int status;

char buff[MAXSIZE];

char \*args[64];

int arg\_nums = 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;

}

arg\_nums = parse(buff, args);

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

exit(0);

}

pid = fork();

if(pid < 0){

printf("fork error!**\n**");

continue;

}else if (pid == 0)

{

childDo(\*args, args);

}

parentDo(pid, &status);

}

return 0;

}

测试：

