**西安电子科技大学 网络与信息安全学院**

**操 作 系 统 实验报告**

**班 级： 1618011**

**学 号： 16180110065**

**姓 名： 王宇卿**

**Github： https://github.com/pitifulnoble**

**电子邮箱： pitifulnoble@outlook.com**

**2018年4月15日**

|  |
| --- |
| **实验题目：Unix系统shell实现和历史特性** |
| **实验摘要**  写一个c程序实现以下特性：  类似shell终端，可以支持基本Linux，Unix，Mac OS的基本命令  每个命令执行时，使用fork()启动一个新的进程  程序实现history特性可以查看历史命令，并通过!符号执行历史命令 |
| **二、相关原理与知识**  **了解C程序库unistd.h**  **对Linux基本知识的了解**  **熟悉进程相关知识** |
| **实验内容实验过程**  **基本命令实现：**    **History特性实现：** |
| **实验总结**  完成了实验要求，实现了Linux基本命令的支持，实现了history的特性。 |
| **五、问题总结**  **History的实现特性，本来实现怎么复用操作系统本身的history，但是遇到了问题**  **后期实现程序自定义保存历史命令** |
| **源代码**  **#include <stdio.h>**  **#include <unistd.h>**  **#include <stdlib.h>**  **#include <string.h>**  **#define MAX\_LINE 80 /\* The maximum length of a command \*/**  **#define BUFFER\_SIZE 50**  **#define buffer "\n\Shell Command History:\n"**  **//declarations**  **char history[10][BUFFER\_SIZE]; //history array to store history commands**  **int count = 0;**  **//function to display the history of commands**  **void displayHistory()**  **{**    **printf("Shell command history:\n");**    **int i;**  **int j = 0;**  **int histCount = count;**    **//loop for iterating through commands**  **for (i = 0; i<10;i++)**  **{**  **//command index**  **printf("%d. ", histCount);**  **while (history[i][j] != '\n' && history[i][j] != '\0')**  **{**  **//printing command**  **printf("%c", history[i][j]);**  **j++;**  **}**  **printf("\n");**  **j = 0;**  **histCount--;**  **if (histCount == 0)**  **break;**  **}**  **printf("\n");**  **}**  **//Fuction to get the command from shell, tokenize it and set the args parameter**  **int formatCommand(char inputBuffer[], char \*args[],int \*flag)**  **{**  **int length; // # of chars in command line**  **int i; // loop index for inputBuffer**  **int start; // index of beginning of next command**  **int ct = 0; // index of where to place the next parameter into args[]**  **int hist;**  **//read user input on command line and checking whether the command is !! or !n**  **length = read(STDIN\_FILENO, inputBuffer, MAX\_LINE);**      **start = -1;**  **if (length == 0)**  **exit(0); //end of command**  **if (length < 0)**  **{**  **printf("Command not read\n");**  **exit(-1); //terminate**  **}**    **//examine each character**  **for (i=0;i<length;i++)**  **{**  **switch (inputBuffer[i])**  **{**  **case ' ':**  **case '\t' : // to seperate arguments**  **if(start != -1)**  **{**  **args[ct] = &inputBuffer[start];**  **ct++;**  **}**  **inputBuffer[i] = '\0'; // add a null char at the end**  **start = -1;**  **break;**    **case '\n': //final char**  **if (start != -1)**  **{**  **args[ct] = &inputBuffer[start];**  **ct++;**  **}**  **inputBuffer[i] = '\0';**  **args[ct] = NULL; // no more args**  **break;**    **default :**  **if (start == -1)**  **start = i;**  **if (inputBuffer[i] == '&')**  **{**  **\*flag = 1; //this flag is the differentiate whether the child process is invoked in background**  **inputBuffer[i] = '\0';**  **}**  **}**  **}**    **args[ct] = NULL; //if the input line was > 80**  **if(strcmp(args[0],"history")==0)**  **{**  **if(count>0)**  **{**    **displayHistory();**  **}**  **else**  **{**  **printf("\nNo Commands in the history\n");**  **}**  **return -1;**  **}**  **else if (args[0][0]-'!' ==0)**  **{ int x = args[0][1]- '0';**  **int z = args[0][2]- '0';**    **if(x>count) //second letter check**  **{**  **printf("\nNo Such Command in the history\n");**  **strcpy(inputBuffer,"Wrong command");**  **}**  **else if (z!=-48) //third letter check**  **{**  **printf("\nNo Such Command in the history. Enter <=!9 (buffer size is 10 along with current command)\n");**  **strcpy(inputBuffer,"Wrong command");**  **}**  **else**  **{**  **if(x==-15)//Checking for '!!',ascii value of '!' is 33.**  **{ strcpy(inputBuffer,history[0]); // this will be your 10 th(last) command**  **}**  **else if(x==0) //Checking for '!0'**  **{ printf("Enter proper command");**  **strcpy(inputBuffer,"Wrong command");**  **}**    **else if(x>=1) //Checking for '!n', n >=1**  **{**  **strcpy(inputBuffer,history[count-x]);**  **}**    **}**  **}**  **for (i = 9;i>0; i--) //Moving the history elements one step higher**  **strcpy(history[i], history[i-1]);**    **strcpy(history[0],inputBuffer); //Updating the history array with input buffer**  **count++;**  **if(count>10)**  **{ count=10;**  **}**  **}**  **int main(void)**  **{**  **char \*args[MAX\_LINE/2 + 1];/\* max arguments \*/**  **int should\_run =1;**  **char inputBuffer[MAX\_LINE]; /\* buffer to hold the input command \*/**  **int flag; // equals 1 if a command is followed by "&"**      **pid\_t pid,tpid;**  **int i;**      **while (should\_run) //infinite loop for shell prompt**  **{**  **flag = 0; //flag =0 by default**  **printf("osh>");**  **fflush(stdout);**  **if(-1!=formatCommand(inputBuffer,args,&flag)) // get next command**  **{**  **pid = fork();**    **if (pid < 0)//if pid is less than 0, forking fails**  **{**    **printf("Fork failed.\n");**  **exit (1);**  **}**    **else if (pid == 0)//if pid ==0**  **{**    **//command not executed**  **if (execvp(args[0], args) == -1)**  **{**    **printf("Error executing command\n");**  **}**  **}**    **// if flag == 0, the parent will wait,**  **// otherwise returns to the formatCommand() function.**  **else**  **{**  **i++;**  **if (flag == 0)**  **{**  **i++;**  **wait(NULL);**  **}**  **}**  **}**  **}**  **}** |