孟华儒 21917075 作业第二题 模拟bash

实现的功能：

run [path] (-C cpu) ：运行一个外部程序。Cpu调度因为函数不识别并未实现。

list ：查看进程信息，包括进程名，pid，ppid。

kill [pid] ：杀死某个进程，参数为进程pid。

quit ：退出程序。

提交内容：

一个c文件，一个录屏演示视频，一份说明（本文档）。不需要make文件。

流程：

1. 打印基础信息，提醒用户输入指令。
2. 获取指令信息以及参数。
3. 分析指令，执行相应的分支。
4. 结束程序时打印感谢信息。

主要函数：

1. void runnnn(char \*p[10])

当指令为run时会调用本函数，函数会提取参数中的软件路径，并执行目标程序。

1. void printinfo()

当指令为list时会调用本函数，函数会通过访问/proc/文件夹中的进程文件夹下的status文件中的部分信息，并进行打印。

详细代码：

#include <unistd.h>

#include <sched.h>

#include <stdlib.h>

#include <dirent.h>

#include <sys/types.h>

#include <stdio.h>

#include "string.h"

#include "math.h"

#include <signal.h>

#define \_\_USE\_GNU

#define \_GNU\_SOURCE

void runnnn(char \*p[10]); //运行程序函数声明

char ppath[30] ={0}; //存放当前路径

char hostname[30] ={0}; //存放主机名称

int isprocdir(char dir[]) //查看目录是否是纯数字（进程名）

{

int len = strlen(dir);

if(len==0)

return 0;

for(int i=0;i<len;i++)

{

if(dir[i]<'0'||dir[i]>'9')

return 0;

}

return 1;

}

void printinfo() //list功能实现

{

printf("PNAME\t\t\t\t\tPID\tPPID\n");

DIR \*procdir = opendir("/proc/");

struct dirent \*dirstr;

while((dirstr = readdir(procdir))!=NULL)//遍历进程文件夹下面的纯数字文件夹

{

if(isprocdir(dirstr->d\_name)==1)

{

char stafile[500];

sprintf(stafile,"/proc/%s/status",dirstr->d\_name);

FILE\* fp = fopen(stafile,"r");

//按行读取，截取有用的内容并打印：进程名，pid。ppid

char msg[20];

fgets(msg,99,fp);

char \*p = strtok(msg,"\t");

p = strtok(NULL,"\t");

p[strlen(p)-1]='\0';

printf("%-30s",p);

fgets(msg,99,fp);

fgets(msg,99,fp);

fgets(msg,99,fp);

fgets(msg,99,fp);

fgets(msg,99,fp);

p = strtok(msg,"\t");

p = strtok(NULL,"\t");

p[strlen(p)-1]='\0';

printf("%13s",p);

fgets(msg,99,fp);

p = strtok(msg,"\t");

p = strtok(NULL,"\t");

p[strlen(p)-1]='\0';

printf("%9s\n",p);

}

}

}

int main ()

{

printf("Welcome to the MHR's shell\n");

//获取主机名和路径

gethostname(hostname,sizeof(hostname));

getcwd(ppath,sizeof(ppath));

while(1)

{

printf("MHR@%s:%s ",hostname,ppath);

char cmd[50] = {0};

fgets(cmd,49,stdin);

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

if(strlen(cmd)==0)

continue;

char \*cmd\_set[10] = {0,0,0,0,0,0,0,0,0,0};

char \*p = strtok(cmd," ");//截取命令

int pos = 0;

while(p!=NULL && pos < 10)//获取参数

{

cmd\_set[pos] = p;

p = strtok(NULL," ");

pos++;

}

if(strcmp(cmd,"run")==0)//run

{

runnnn(cmd\_set);

}

else if(strcmp(cmd,"list")==0)//list

{

printinfo();

}

else if(strcmp(cmd,"kill")==0)//kill

{

kill(atoi(cmd\_set[1]),SIGKILL);//杀死参数进程

}

else if(strcmp(cmd,"quit")==0)//quit

{

printf("Thank you for using!\n");

return 1;

}

else

{

printf("UNKNOWN CMD!\n");

}

}

return 0;

}

void runnnn(char \*p[10])//参数结构：run 文件名 (setCPU) (CPU)

{

int pid=0;

pid=fork();//fork

if (pid < 0)

printf("error in fork!");

else if (pid == 0) {

if(p[2]!=NULL)

{

if(strcmp("-C",p[2])==0)//设定CPU

{

printf("setCPU %s\n",p[3]);

//改变CPU亲和度

//cpu\_set\_t mask;

//CPU\_ZERO(&mask);

//CPU\_SET(atoi(p[3]),&mask);

//sched\_setaffinity(0,sizeof(cpu\_set\_t),&mask);

//函数不识别，功能暂时注释

}

else

{

printf("UNKNOWN CMD!\n");

return;

}

}

char \*argv[]={p[1]};

char \*envp[]={0,NULL};

execve(p[1],argv,envp);//运行

}

return;

}