**今天的工作**

**修改pipe**

**遇到的问题**

1. pipe在shell中怎么进行。
2. 命令和参数放在哪里，怎么放，怎么老是段错误！！！
3. 运行完指令直接退出shell了！怎么解决execvp毙掉父进程问题！！
4. waitpid为什么一直在等待？？？

**解决的问题**

1. 指令序列存放在commandCompose，遍历输入的指令，如果发现管道符号|就返回\_switch=1,如果\_switch=1就调用pipe();如果调用成功应当返回值\_switch=1
2. <https://www.cnblogs.com/luosongchao/p/3381564.html>

一开始是想要把管道前指令放到cmd1[]，管道后指令放到cmd2[]中，两个命令的参数分别放到param1[],param2[]但是一直有段错误。。。搞了半天最后决定用两个char \* cmd1[]和char \* cmd2[]来存放管道前的参数和管道后的参数。cmd1[0]和cmd2[0]分别存放管道前后的命令。

1. <https://wenku.baidu.com/view/1c01004cfe00bed5b9f3f90f76c66137ef064f61.html>

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

执行管道前的命令比如ls

}

else{

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

执行管道后的命令比如grep

}

else{

父进程部分

sleep(1);

return 1;

}

}

1. <https://blog.csdn.net/yangbodong22011/article/details/49741027>

不知道为什么。。。反正改成了sleep(1)

**待进一步解决**

关于字符型数组char \* a和char a[]和char \* a[]和char 的初始化和赋值方法存取方式等strcpy（）

**关键代码**

**1.pipe实现**

/\*\*\*\*\* Info \*\*\*\*\*/

/\* Author: ZHJ \*/

/\* Function: 实现管道命令 \*/

int pipe\_zhj(){

/\* Note：如下是测试用例 \*/

/\* cmd中存放指令序列,cmd[0]为pipe前的指令，cmd为pipe后的指令 \*/

/\* param1存放cmd[0]指令的参数 \*/

/\* param2存放cmd[1]指令的参数 \*/

/\* 测试用例 \*/

//printf("entering pipe!\n");

int \_\_switch=1;

char \*cmd1[MAX\_CMD\_LEN];

char \*cmd2[MAX\_CMD\_LEN];

int i=0;

int j=0;

/\*initialize\*/

while(strcmp(commandCompose[i],"|")!=0){

cmd1[i]=(char \*)malloc(MAX\_CMD\_LEN \* sizeof(char));

strcpy(cmd1[i],commandCompose[i]);

i++;

}

cmd1[i]=0;

i++;

j=0;

while(commandCompose[i]!=NULL){

cmd2[j]=(char \*)malloc(MAX\_CMD\_LEN \* sizeof(char));

strcpy(cmd2[j],commandCompose[i]);

//param1[j]=commandCompose[i--];

i++;

j++;

}

cmd2[j]=0;

//创建管道

int pid1;

int pid2;

int fd[2];//fd[0]读端，fd[1]写端

pipe(fd);

int flag;

//创建子进程

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

//子进程1，默认为pipe前的指令ls

dup2(fd[1],1);//把标准输出流重定向到管道写端

close(fd[0]);//关闭管道的文件描述符

close(fd[1]);

if((flag=execvp(cmd1[0],cmd1))<0){

printf("son:no such command %d",flag);

}

exit(EXIT\_SUCCESS);

}

else{

//父进程，默认为pipe后的指令

//waitpid(pid1,NULL,0);

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

dup2(fd[0],0);//把标准输入流重定向到管道读端

close(fd[0]);

close(fd[1]);

if((flag=execvp(cmd2[0],cmd2))<0){

printf("father:no such command %d",flag);

}

exit(EXIT\_SUCCESS);

}

//printf("succesee");

//exit(1);

else{

//waitpid(pid1,NULL,0);

//waitpid(pid2,NULL,0);

sleep(1);

return \_\_switch;

}

}

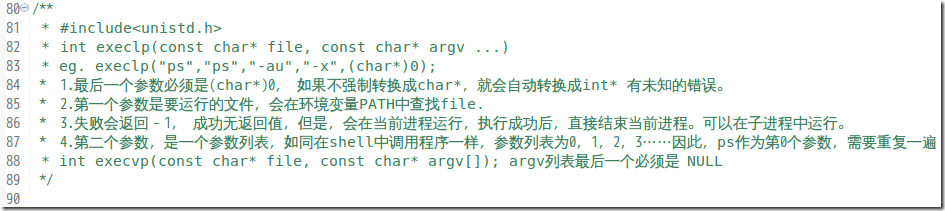
}

**其他想法**

今天一直在修改这个，因为一直报错段错误

搞错了几个很重要的地方：

1. execvp(要执行的文件char\* file,命令的参数char \* argv[])



* 1. argv[]数组中第一个元素应该是要执行的指令，即要重复一遍第一个参数
  2. argv[]数组最后一个元素必须是NULL

1. 父子进程的执行，一开始把管道前的命令放在子进程执行，管道后的命令放在父进程执行，因为两个都用了execvp()，所以会把父进程都毙掉，于是命令运行完之后直接退出了shell，所以要把管道前后的命令都放到子进程中执行，这样父进程就不会被毙掉了。
2. 父进程不能使用waitpid(pid1,NULL,0)等待子进程执行完，会导致无限等待下去，原因未知！所以就用sleep(1)等待一下。。。有待改进

**明天的计划**

听组长安排