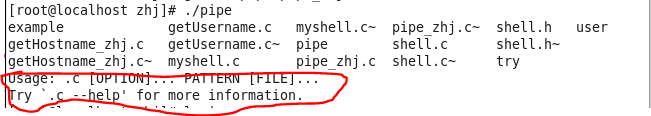
**今天的工作**

继续完成功能，从完成管道和文件重定向功能开始

管道：比如ls | grep main.c（在当前目录下找到main.c后缀的文件）

**遇到的问题**

1. 怎么实现pipe
   1. 使用什么exec()族函数
   2. 建立测试用例
2. git时遇到无法push提示远程仓库版本和当前本地仓库版本冲突
3. 第一次测试显示no such command，表示exec执行不成功
4. expected ‘char \* const\*’ but argument is of type ‘char \*’
5. ls运行成功，grep没有运行成功



6.readline库的安装直接yum install readline-devel不成功，显示，没有readline这个依赖包

**解决的问题**

1. 使用execvp函数
   1. 如果成功执行没有返回值，发生异常返回-1
   2. 第一个参数是要执行的指令，第二个参数是参数列表

比如我要执行的是cmd[0]中存放的文件(从环境变量PATH开始查找)，参数列表就是要传入的参数；列表。

execvp(cmd[0],cmd);

1. git fetch origin master→git merge origin master→shift zz保存，然后再尝试git push
2. execvp的第二个参数argv数组最后一位必须是0！！<https://blog.csdn.net/u011857683/article/details/81160059>

char \* argv[] = {"wensen.sh", 0};

1. 类型不匹配
2. 变量赋值出错，应该是：
   1. 子进程：execvp(pipe符”|”前面的指令比如ls给出输出到管道写端，ls的命令参数)
   2. 父进程：execvp(pipe符”|”后面的指令比如grep从管道读端读输入，grep的参数如”.c”)
   3. 在之前cd挂载的地方安装readline-devel依赖包，显示需要再安装另一个依赖包，依次安装之后成功，再yum install readline-devel，然后/usr/include/readline中有readline.h用#include<readline/readline.h>包含。

<https://blog.csdn.net/CTO_51/article/details/12857375>

**待进一步解决**

1.多重pipe

**关键代码**

**1.pipe\_zhj**

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

/\* Author: zhj \*/

/\* Function: 实现管道\*/

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

#include<stdio.h>

#include <unistd.h>

#include<stdlib.h>

void 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 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

char \* cmd[10];

char \* param1[]={"",0};

char \* param2[]={"-l",".c",0};

cmd[0]="ls";

cmd[1]="grep";

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*1.创建管道\*/

int pid;

int fd[2];//fd[0]读端，fd[1]写端

pipe(fd);

int flag;

/\*2.创建子进程\*/

if((pid=fork())==0){

//子进程，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(cmd[0],param1))<0){

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

}

exit(EXIT\_SUCCESS);

}

else{

//父进程，默认为pipe后的指令

waitpid(pid,NULL,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(cmd[1],param2))<0){

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

}

exit(EXIT\_SUCCESS);

}

}

//

//int main(){

//

// pipe\_zhj();

//

// return 0;

//

//}

**2.getUsername**

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

/\* Author: zhj \*/

/\* Function: 得到用户名\*/

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

#include <stdio.h>

#include <string.h>

#include <pwd.h>

void getUsername\_zhj(){

/\* Note： max\_name\_len是用户名最大的长度\*/

/\*这里定义为40\*/

const max\_name\_len=40;

char username[max\_name\_len];

struct passwd\* pwd = getpwuid(getuid());

strcpy(username, pwd->pw\_name);

printf("%s\n",username);

}

int main()

{

getUsername\_zhj();

return 0;

}

**其他想法**

无

**明天的计划**

继续完成功能，我打算从完成管道和文件重定向功能开始

管道：比如ls | grep main.c（在当前目录下找到main.c后缀的文件）