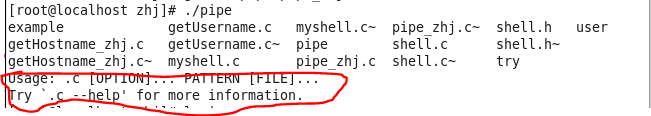
**今天的工作**

1. 继续完成功能
   1. 管道：比如ls | grep main.c（在当前目录下找到main.c后缀的文件）
   2. 文件重定向：比如指令里遇到“<”、“>”分别进行重定向
2. 了解linux的命令，总结下要实现的功能和支持的命令

**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



1. readline库的安装直接yum install readline-devel不成功，显示，没有readline这个依赖包
2. 怎么获得当前工作路径

**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>

1. 要实现的功能
   1. 命令解释执行
   2. 命令补全的功能
   3. 文件名替换
   4. I/O重定向
   5. 通信管道建立

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **命令** | **类型** | **用法** | **功能** | **备注** |
| ls | 外部命令 | ls | 列出目录中的内容 | 常用命令 |
| cat | 外部命令 | cat 文件名 | 显示文件内容 | 常用命令 |
| echo | 内建命令 | echo abc | 显示一行文本或者字符串 | 常用命令 |
| cd | 内建命令 | cd 目录 | 进入指定目录 | 常用命令 |
| type | 内建命令 | type 命令名 | 显示命令的类型（外部/内建） | 常用命令 |
| tab键 | 特殊键盘事件 | more pa<tab> echo fi<tab> | ①按下Tab键一次，自动补足完整的命令变成了more pasaage ②按下Tab键两次，系统会显示当前目录下所有具有相同前缀的文件名称，供用户选择 | 命令补全功能 |
| history | 内建命令 | ①history ②history n | ①查看所有历史执行的命令 ②查看最近执行的n条命令 | 查阅历史记录 |
| ！ | 特殊关键字 | ①！n ②！！ | ①运行第n条历史记录 ②运行上一条历史记录 | 查阅历史记录 |
| alias | 内建命令 | alias dir='ls -l' | 创建别名 | 别名功能 |
| unalias | 内建命令 | unalias dir | 取消别名 | 别名功能 |
| & | 特殊关键字 | ls -l & | 后台的方式执行该命令，显示后台运行程序的进程PID，shell返回命令提示符状态 | 后台功能 |
| > | 特殊元字符 | ls -l > File1 | 将ls –l命令的结果送至File1文件中 | I/O重定向 |
| < | 特殊元字符 | cat < File1 | cat从File1获得输入,察看File1文件的内容 | I/O重定向 |
| | | 特殊元字符 | ls | grep .c | 建立一个ls进程和grep进程之间的管道 | 通信管道建立 |
| rename | 外部命令 | rename a.c b.c a.c | 将文件a.c的名字改成b.c | 文件名替换 |
| grep | 外部命令 | 较少单独使用，一般配合管道使用 | 使用正则表达式搜索文本，并把匹配的行打印出来 | 常用命令 |

1. 使用getcwd(buffer,max\_len)函数来获得当前工作目录的绝对目录，并将该绝对路径复制到buffer,max\_len是buffer的空间大小

**待进一步解决**

1.多重pipe

**关键代码**

**1.pipe管道**

/\*\*\*\*\* 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.redirect文件重定向**

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

/\* Author: zhj \*/

/\* Function: 实现文件重定向\*/

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

#include<stdio.h>

#define max\_path\_len 20

void redirect\_zhj(){

char filepath[max\_path\_len]="/root/zhj/a.txt";

char result[max\_path\_len];

freopen(filepath,"r",stdin);//输入流重定向到filepath指定的文件

scanf("%s",&result);

freopen(filepath,"w",stdout);//输出流重定向到filepath指定的文件

printf("right!%s",result);

}

int main(){

redirect\_zhj();

return 0;

printf("this is a test");

}

**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）