#include <stdio.h>

#include <stdlib.h>

#include<string.h>

#include <unistd.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#define MAX\_WORD 10

#define MAX\_CHAR 100

#define DEL " "

int input\_redirection\_flag;

int output\_redirection\_flag;

int piping\_flag;

char\* input\_file= NULL;

char\* output\_file= NULL;

///////////////////////////////////////

void remove\_endofline(char line[])

{

int i=0;

while(line[i]!='\n')

i++;

line[i]='\0';

}

//////////////////////////////

void read\_line(char line[])

{

char\* ret =fgets(line, MAX\_CHAR,stdin);

//printf("%s\n",line);

remove\_endofline(line);

if(strcmp(line,"exit")==0||ret==NULL)

exit(0);

}

////////////////////////////////////////

int process\_line(char\* temp[],char line[])

{

int i =0;

temp[i]=strtok(line,DEL);

while(temp[i]!= NULL){

i++;

temp[i]= strtok(NULL,DEL);

}

return 1;

}

//////////////////////////////////////////////////////////

int pipe\_and\_redirection\_check(char\* temp[]){

int i=0;

while(temp[i]!= NULL)

{

if(strcmp(temp[i],">")==0){

output\_redirection\_flag=1;

output\_file = temp[i+1];

return i;

}

else if(strcmp(temp[i],"<")==0){

input\_redirection\_flag =1;

input\_file = temp[i+1];

return i;

}

else if(strcmp(temp[i],"|")==0){

piping\_flag=1;

return i;

}

i++;

}

return i;

}

//////////////////////////////////////////////////////

void check\_line(char\* temp[])

{

int i =0;

int pipe\_count=0;

int output\_redirection\_count =0;

int input\_redirection\_count =0;

int total\_count;

if(temp[i]== NULL)

{

printf("no command\n");

return ;

}

while(temp[i]!=NULL)

{

if(strcmp(temp[i],">")==0)

output\_redirection\_count ++;

else if(strcmp(temp[i],"<")==0)

input\_redirection\_count++;

else if(strcmp(temp[i],"|")==0)

pipe\_count++;

i++;

}

total\_count=input\_redirection\_count+output\_redirection\_count+pipe\_count;

if(total\_count > 1){

printf("SORRY MY SHELL DOESN'T HANDLE THIS CASE\n");

temp[0]=NULL;

}

}

//////////////////////////////////////////////////////////////////////////

int read\_parse\_line(char\* args[],char\* piping\_args[],char line[])

{

char\* temp[MAX\_WORD];

int i=0;

int pos;

read\_line(line);

process\_line(temp,line);

check\_line(temp);

pos =pipe\_and\_redirection\_check(temp);

while(i<pos){

args[i]=temp[i];

i++;

}

args[i]=NULL;

if(piping\_flag==1)

{

int y=0;

while(temp[i]!=NULL)

{

piping\_args[y]=temp[i];

i++;

y++;

}

}

return 1;

}

/////////////////////////////////////////

/\*void piping\_handle( char\* args[] ,char\* piping\_args[],int pipefd[])

{

int pid;

int i;

pid=fork();

if(pid==0)

{

dup2(pipefd[1],1);

close(pipefd[0]);

close(pipefd[1]);

execvp(args[0],args);

error("failed to exec command 1");

}

else

{

dup2(pipefd[0],0);

close(pipefd[1]);

close(pipefd[0]);

execvp(piping\_args[0],piping\_args);

error("failed to exec command 2");

}

}

\*/

/////////////////////////////////////////////////////////////

int main()

{

char\* args[MAX\_WORD];

char line[MAX\_CHAR];

char\* piping\_args[MAX\_WORD];

int pipefd[2];

pipe(pipefd);

while(read\_parse\_line(args,piping\_args,line))

{

pid\_t pid = fork();

if(pid==0)

{

//////////////////////////////////////////////////

if(input\_redirection\_flag== 1&& input\_file!=NULL)

dup2(open(input\_file,O\_RDWR|O\_CREAT,0777),0);

////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////

else if(output\_redirection\_flag== 1&& output\_file!=NULL)

dup2(open(output\_file,O\_RDWR|O\_CREAT,0777),1);

///////////////////////////////////////////////////////////

else if(piping\_flag==1);

{

//iping\_handle(args,piping\_args,pipefd);

//exit(0);

}

execvp(args[0],args);

}

else

{

waitpid(pid, 0);

input\_redirection\_flag=0;

output\_redirection\_flag=0;

output\_file= NULL;

input\_file =NULL;

piping\_flag =0;

}

}

return 0;

}