/\*

\* run\_command.c : do the fork, exec stuff, call other functions

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <signal.h>

#include <errno.h>

#include <sys/wait.h>

#include "shell.h"

#include <unistd.h>

void run\_command(char \*\*myArgv) {

pid\_t pid;

int stat;

int number;

/\* Create a new child process.

\* Fill in code.

\*/

pid =fork();

switch (pid) {

/\* Error. \*/

case -1 :

perror("fork");

exit(errno);

/\* Parent. \*/

default :

/\* Wait for child to terminate.

\* Fill in code.

\*/

if(!is\_background(myArgv)){

if (waitpid(pid, &stat, 0) == -1) {

perror("waitpid");

exit(1);

}

}

/\* Optional: display exit status. (See wstat(5).)

\* Fill in code.

\*/

if (WIFSIGNALED(stat)) {

printf("child process terminated by signal %d.\n",WTERMSIG(stat));

} else if (!WIFEXITED(stat)) {

printf("child process exited with status %d.\n",WEXITSTATUS(stat));

} else if (WIFSTOPPED(stat)) {

printf("child process stopped by signal %d.\n",WSTOPSIG(stat));

}

return;

/\*

WIFEXITED(status)如果子進程正常結束則為非0值。

WEXITSTATUS(status)取得子進程exit()返回的結束代碼，一般會先用WIFEXITED來判斷是否正常結束才能使用此宏。

WIFSIGNALED(status)如果子進程是因為信號而結束則此宏值為真。

WTERMSIG(status)取得子進程因信號而中止的信號代碼，一般會先用WIFSIGNALED來判斷後才使用此宏。

WIFSTOPPED(status)如果子進程處於暫停執行情況則此宏值為真。一般只有使用WUNTRACED時才會有此情況。

WSTOPSIG(status)取得引發子進程暫停的信號代碼。

\*/

/\* Child. \*/

case 0 :

/\* Run command in child process.

\* Fill in code.

\*/

if(is\_background(myArgv)){

int count = 0;

while( myArgv[count] != NULL ){

count++;

}

myArgv[count-1] ='\0';//NULL

execvp(myArgv[0],myArgv);

}else{

execvp(myArgv[0],myArgv);

/\*if(execvp(myArgv[0],myArgv)<0){ //當execvp回傳一個int代表執行成功+上if防止出錯

perror("execvp");

exit(1);

} \*/

/\* Handle error return from exec \*/

exit(errno);

}

}

}