

## Homework 2 報告

### 報告前言:

- 使用語言:**c++**
- 使用平台:**Ubuntu**
- 需要安裝的套件: 跟老師作業上的規範一樣, 無須額外安裝套件。(↓所需套件)

libreadline-dev, libreadline6, libtinfo-dev, readline-doc (使用 GNU readline 需要),  
clang (編譯用, 必備)

- 編譯方式: 編譯程式的指令: **clang XXX.cpp -lreadline -o XXX.out**

```
xiaowang@xiaowang-VirtualBox:~/OS/hw2$ clang++ OShw2_s1043326.cpp -lreadline -o shell
```

↑使用 **clang++ OShw2\_s1043326.cpp -lreadline -o shell** 進行編譯

- 執行方式: 執行程式的指令: **./XXX.out**

```
xiaowang@xiaowang-VirtualBox:~/OS/hw2$ clang++ OShw2_s1043326.cpp -lreadline -o shell
xiaowang@xiaowang-VirtualBox:~/OS/hw2$ ./shell
```

(進入程式後會先進行清屏)

### 報告開始:

可以接收並執行任意不帶參數的指令。舉例:

icheyneh@/home/OS>**ls** << 使用者輸入 **ls** 並按下 Enter

接著你便透過你的 shell 來建立一個新的 ls 程序, 並將輸出導向標準輸出 (於文字介面上輸出)

1.

可以接收並任何帶有參數的指令。舉例:

icheyneh@/home/OS>**ls -al** << 使用者輸入 **ls** 並加上附加參數 **"-al"** 最後按下 Enter

接著你便透過你的 shell 來建立一個新的 ls 程序, 執行時給與 **-al** 這個參數, 並將最終輸出導向標準輸出

2.

在這兩個部分，我是寫在我的code的第98行，`void NormalCMD( const int argc, char** ( &argv ) );`這個function中，我們先將這個function拆成兩部份，第一部分為普通的狀況，code如下

```
110     pid_t pid = fork();
111     if( pid < 0 ){
112         cerr << "Something error!\n";
113         exit( 1 );
114     }
115     else if( pid == 0 ){
116         int tmpArgc = argc;
117         bool mark = false;
118
119         if( strcmp( argv[ 0 ], "ls" ) == 0 ||
120             ( strcmp( argv[ 0 ], "grep" ) == 0 && argc >= 2 ) ){
121             strcpy( argv[ tmpArgc++ ], "--color=auto" );
122             mark = true;
123         }
124
125         delete [] argv[ tmpArgc ];
126         argv[ tmpArgc ] = NULL;
127
128         if( execvp( argv[ 0 ], argv ) == -1 )
129             cerr << argv[ 0 ] << "：無此指令\n";
130
131         argv[ tmpArgc ] = new char[ typeMax ]();
132         if( mark ){
133             delete [] argv[ argc ];
134             argv[ argc ] = new char[ typeMax ]();
135         }
136
137         exit( 0 );
138     }
139     else waitpid( pid, NULL, 0 );
```

在這個狀況下我們會先宣告一個`pid_t pid`，`pid`是`process ID`的縮寫，然後我們就會利用`unistd.h`的`pid_t fork( void )`，來產生`parent`或`child process`當`pid == 0`時(`child process`)我們需要做的事情是利用同樣是`unistd.h`的`int execvp( const char *file, char *const argv[ ] );`來讓我們的`child process`轉變為`execvp`所呼叫的

file，也因為一轉變為execvp所呼叫的file後就無法回頭，所以才要先進行fork()，以避免parent process變成execvp所呼叫的file，execvp中argv[]所傳入的參數最後必須為NULL，所以line 125~126就是在將NULL放入尾端，而line 119~123則是當輸入指令為ls或者是grep我們加入—color=auto，就可以讓文字有顏色了，之後我們利用sys/type.h和sys/wait.h中的pid\_t waitpid( pid\_t pid, int \*status, int options );來等待特定的pid。而第二個部分則是我們必須去避免輸入是cd時我們還去利用child process來改變current working directory所以在這個狀況我們，要避免去進行fork()，也因此其實這段code是比上面的fork()先執行(line 99~108)。

```
99     if( strcmp( argv[ 0 ], "cd" ) == 0 ){
100         if( argc >= 2 ){
101             if( chdir( argv[ 1 ] ) != 0 )
102                 cerr << "沒有此一檔案或目錄\n";
103         }
104         else if( argc == 1 )
105             chdir( getenv( "HOME" ) );
106
107         return;
108     }
```

第1, 2題輸出:

(可以發現我的ls和grep抓取出來的字串是有顏色的)

```
xiaowang@xiaowang-VirtualBox: ~/OS/hw2
xiaowang@/home/xiaowang/OS/hw2> ls
input OShw2_s1043326.cpp OShw2_s1043326_easy.cpp ouput shell
xiaowang@/home/xiaowang/OS/hw2> ls -al
總計 60
drwxr-xr-x 2 xiaowang xiaowang 4096 5月 6 15:54 .
drwxr-xr-x 4 xiaowang xiaowang 4096 4月 18 00:33 ..
-rw-r--r-- 1 xiaowang xiaowang 454 5月 1 20:22 input
-rw-r--r-- 1 xiaowang xiaowang 10477 5月 6 13:28 OShw2_s1043326.cpp
-rw-r--r-- 1 xiaowang xiaowang 10174 4月 12 20:29 OShw2_s1043326_easy.cpp
-rw-r--r-- 1 xiaowang xiaowang 6 5月 6 15:54 ouput
-rwxr-xr-x 1 xiaowang xiaowang 18632 5月 6 15:53 shell
xiaowang@/home/xiaowang/OS/hw2> cat ouput
hello
xiaowang@/home/xiaowang/OS/hw2> grep include OShw2_s1043326.cpp
#include <iostream>
#include <stdlib.h>
#include <unistd.h>
#include <readline/readline.h>
#include <readline/history.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
xiaowang@/home/xiaowang/OS/hw2> 
```

第1, 2題心得:

在上課經過老師的講解後，對於fork()和exec()有了初步的認識，也因為有了這些知識，讓我順利的完成了第一部份。

3. 可以將多個不同的指令串接起來，提供 pipe '|' 的功能，讓使用者可以將一個程式的輸出 (standard output) 接到另一個程式的輸入，需要支援多個程式串接 (最多四個)。

在這個部份，我是寫在我的code的第142行，void

PipeCMD( const int argc, char\*\*( &argv ));

中，在這個function，我思考了很多種不同的方式，

最後採用的是只利用一組由pipe所產生的fd和一

個宣告的int fd來控制我的pipe，也就是我一共只

有三個fd而已，想要進行pipe，就會自然而然地想要將指令切成一組組的char\*\*，code如下↓

```
143 char*** argvList;
144 int pipeCounter = 0, *argcList, pFd[ 2 ], FdIn = STDIN_FILENO;
145
146 CreateArgvList( ptrMax, argcList, argvList );
147 for( int i = 0; i < argc; ++i ){
148     if( strchr( argv[ i ], '|' ) != NULL ){
149         if( argv[ i ][ 0 ] == '|' ) pipeCounter++;
150         char *tmpArgv = new char[ typeMax ];
151         strcpy( tmpArgv, argv[ i ] );
152         char *find = strtok( tmpArgv, "|" );
153         while( find != NULL ){
154             strcpy( argvList[ pipeCounter ][ argcList[ pipeCounter ]++ ], find );
155             find = strtok( NULL, "|" );
156             if( find != NULL ) pipeCounter++;
157         }
158         if( argv[ i ][ strlen( argv[ i ] ) - 1 ] == '|' && strlen( argv[ i ] ) > 1 )
159             pipeCounter++;
160         delete [] tmpArgv;
161         delete [] find;
162     }
163     else
164         strcpy( argvList[ pipeCounter ][ argcList[ pipeCounter ]++ ], argv[ i ] );
165 }pipeCounter++;
```

先宣告一個char\*\*\* argvList和int\* argcList來最為存放每一組argv和argc的地方，int pFd[ 2 ]和int FdIn則是pipe所需要的fd，其中line 146所出現的CreateArgvList為我自行撰寫的function，code在line 78~86，而有Create，自然也會有Destory，在line 88~96，Create負責跟記憶體要求argvList和argcList的空間，Destory則負責刪除，而line 149~162則為調整不同的pipe輸入方式我們一樣可以成功接收，而當我們成功分割出argvList時，我們也就可以開始進行我們的pipe工程了，code如下↓

```

167     pid_t pid[ pipeCounter ];
168
169     for( int i = 0; i < pipeCounter; ++i ){
170         pipe( pFd );
171         pid[ i ] = fork();
172         if( pid[ i ] < 0 ){
173             cerr << "Something error!\n";
174             exit( 1 );
175         }
176         else if( pid[ i ] == 0 ){
177             dup2( FdIn, STDIN_FILENO );
178             if( i != pipeCounter - 1 )
179                 dup2( pFd[ 1 ], STDOUT_FILENO );
180
181             ExecCMD( argcList[ i ], argvList[ i ] );
182
183             DestroyArgvList( ptrMax, argcList, argvList );
184             if( i != 0 ) close( FdIn );
185             close( pFd[ 0 ] );
186             close( pFd[ 1 ] );
187             exit( 0 );
188         }
189         else{
190             if( i != 0 ) close( FdIn );
191             close( pFd[ 1 ] );
192             if( i == pipeCounter - 1 ) close( pFd[ 0 ] );
193             else FdIn = pFd[ 0 ];
194         }
195     }
196     for( int i = 0; i < pipeCounter; ++i ) waitpid( pid[ i ], NULL, 0 );
197
198     DestroyArgvList( ptrMax, argcList, argvList );

```

一開始我們宣告了 `pid_t pid array`，這個的目的在於到時候我們的 `parent` 可以一組組的等待 `child` 完成然後我們利用 `for` 迴圈來進行 `int pipe( int pipefd[ 2 ] );` 及 `fork()`，然後判斷是 `child process` 還是 `parent process`，如果是 `child process`，我們要做的事情就是將 `stdin` 用前一個部份的 `pFd[ 0 ]` 取代(第一組直接用 `stdin` 即可)，並且將這一個部份的 `stdout` 用 `pfd[ 1 ]` 取代，這個部份可以用 `unistd.h` 的 `int dup2( int oldfd, int newfd );` 來執行，然後就是利用 `exec()` 來執行這個部份，然後 `child process` 所有的 `fd` 再利用

`unistd.h`的`int close( int fd )`關掉，就完成`child`的部分了，而`parent`只需要判斷什麼時候要關掉`pipe`中的`fd`並且讓`fdIn = pFd[ 0 ]`即可，等到整個結束之後再進行`waitpid`。

第3題輸出：

```
xiaowang@/home/xiaowang/OS/hw2> ls|grep s
0Shw2_s1043326.cpp
0Shw2_s1043326_easy.cpp
shell
xiaowang@/home/xiaowang/OS/hw2> ls|grep s|sort -r
shell
0Shw2_s1043326_easy.cpp
0Shw2_s1043326.cpp
xiaowang@/home/xiaowang/OS/hw2> ls|grep s|sort -r|grep 0
grep0: 無此指令
xiaowang@/home/xiaowang/OS/hw2> ls|grep s|sort -r|grep 0
0Shw2_s1043326_easy.cpp
0Shw2_s1043326.cpp
xiaowang@/home/xiaowang/OS/hw2> ls|grep s|sort -r|grep 0|grep .
grep.: 無此指令
xiaowang@/home/xiaowang/OS/hw2> ls|grep s|sort -r|grep 0|grep .
0Shw2_s1043326_easy.cpp
0Shw2_s1043326.cpp
xiaowang@/home/xiaowang/OS/hw2> ls|grep s|sort -r|grep 0|grep .|sort -r
0Shw2_s1043326_easy.cpp
0Shw2_s1043326.cpp
xiaowang@/home/xiaowang/OS/hw2> ls|grep s|sort -r|grep 0|grep .|sort -r|grep
0Shw2_s1043326_easy.cpp
0Shw2_s1043326.cpp
```

第3題心得：

在第三題卡了一段時間，從一開始的A | B無法執行，一步步的進步到可以執行很多次`pipe`，這個讓我的成就感很大，雖然剛開始遇到了一些挫折，但是在閱讀了文章和看一下別人的範例後，我還是成功克服了這一個部份。

提供 redirect '`<, >, 1>, 2> ...`' 的功能讓程式將標準輸出、標準錯誤輸出等重新導向到新目的地（作業中只測試從檔案導入或導出到檔案，但會跟 `pipe` 混合運用）

4.

在這個部份，我想將跟`pipe`混用的部分拆到第5個



part來講，而前面的部份，則是在我的code的第201行，`void RedirectionCMD( const int argc, char**(&argv ))`;在這一題中，我們將指令拆成兩個部份，一邊為檔案，另一部分則為前面的指令，code如下↓

```
202 char*** argvList;
203 int *argcList, fd, stdNum = -1, cNo = -1;
204
205 CreateArgvList( 2, argcList, argvList );
206 for( argcList[ 0 ] = argc - 1; argcList[ 0 ] >= 0; --argcList[ 0 ] ){
207     if( strcmp( argv[ argcList[ 0 ] ], ">" ) == 0 || strcmp( argv[ argcList[ 0 ] ], "<" ) == 0 ){
208         stdNum = ( strcmp( argv[ argcList[ 0 ] ], ">" ) == 0 ? STDOUT_FILENO : STDIN_FILENO );
209         break;
210     }
211     else if( strcmp( argv[ argcList[ 0 ] ], "1>" ) == 0 || strcmp( argv[ argcList[ 0 ] ], "2>" ) == 0 ){
212         stdNum = ( strcmp( argv[ argcList[ 0 ] ], "1>" ) == 0 ? STDOUT_FILENO : STDERR_FILENO );
213         break;
214     }
215     else if( strcmp( argv[ argcList[ 0 ] ], ">>" ) == 0 || strcmp( argv[ argcList[ 0 ] ], "2>>" ) == 0 ){
216         stdNum = ( strcmp( argv[ argcList[ 0 ] ], ">>" ) == 0 ? 3 : 4 );
217         break;
218     }
219     else if( strcmp( argv[ argcList[ 0 ] ], "1>&2" ) == 0 || strcmp( argv[ argcList[ 0 ] ], "2>&1" ) == 0 )
220         cNo = ( strcmp( argv[ argcList[ 0 ] ], "1>&2" ) == 0 ? STDOUT_FILENO : STDERR_FILENO );
221 }
222 for( int i = 0; i < argcList[ 0 ]; ++i ) strcpy( argvList[ 0 ][ i ], argv[ i ] );
223 for( int i = argcList[ 0 ] + 1, j = 0; i < argc; ++i ) strcpy( argvList[ 1 ][ j++ ], argv[ i ] );
```

`stdNum`為判斷等一下要轉變的是哪一個(stdin, stdout, stderr)，等到字串切好後，我們就可以進行redirection了，code如下↓



```

225     pid_t pid = fork();
226     if( pid < 0 ){
227         cerr << "Something error!\n";
228         exit( 1 );
229     }
230     else if( pid == 0 ){
231         if( stdNum == STDIN_FILENO )
232             fd = open( argvList[ 1 ][ 0 ], O_RDONLY, 0644 );
233         else if( stdNum == STDOUT_FILENO || stdNum == STDERR_FILENO )
234             fd = open( argvList[ 1 ][ 0 ], O_CREAT | O_TRUNC | O_WRONLY, 0644 );
235         else if( stdNum == 3 || stdNum == 4 ){
236             fd = open( argvList[ 1 ][ 0 ], O_APPEND | O_WRONLY, 0644 );
237             stdNum = ( stdNum == 3 ? STDOUT_FILENO : STDERR_FILENO );
238         }
239         if( fd < 0 ){
240             cerr << "File " << argvList[ 1 ][ 0 ] << " can't open!\n";
241             exit( 1 );
242         }
243         dup2( fd, stdNum );
244         if( cNo != -1 ) dup2( fd, cNo );
245         close( fd );
246
247         ExecCMD( argcList[ 0 ], argvList[ 0 ] );
248
249         DestroyArgvList( 2, argcList, argvList );
250
251         exit( 0 );
252     }
253     else waitpid( pid, NULL, 0 );
254
255     DestroyArgvList( 2, argcList, argvList );

```

首先，我們先判斷stdNum是0~4的哪種(3和4為Append)然後再用dup2進行duplicate，將stdNum指向fd，在進行execvp即可。

Dup2圖解：

0	STDIN_FILENO	
1	STDOUT_FILENO	
2	STDERR_FILENO	
3(fd)	OPEN()	

Dup2(fd, STDIN\_FILENO);

0	STDIN_FILENO	-----
1	STDOUT_FILENO	
2	STDERR_FILENO	

3(fd)	OPEN()	←---
-------	--------	------

Close(fd);

0	OPEN()	
1	STDOUT_FILENO	
2	STDERR_FILENO	

第4題輸出：

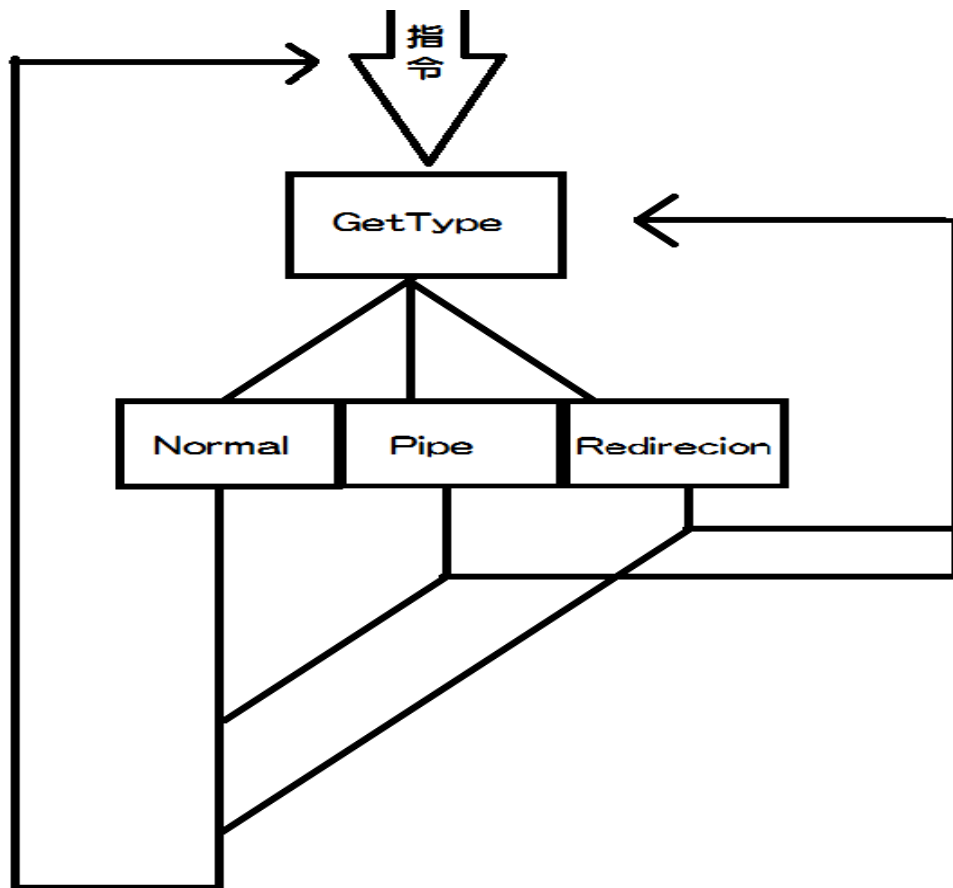
```
xiaowang@/home/xiaowang/OS/hw2> ls > ouput
xiaowang@/home/xiaowang/OS/hw2> cat < ouput
input
OShw2_s1043326.cpp
OShw2_s1043326_easy.cpp
ouput
shell
xiaowang@/home/xiaowang/OS/hw2> ls x
ls: 無法存取 'x': 沒有此一檔案或目錄
xiaowang@/home/xiaowang/OS/hw2> ls x 2> ouput
xiaowang@/home/xiaowang/OS/hw2> cat < ouput
ls: 無法存取 'x': 沒有此一檔案或目錄
xiaowang@/home/xiaowang/OS/hw2> □
```

第4題心得：

在做完第3題後，就會發現其實第4題用dup2來運行是十分容易的也因此，可以十分迅速的將第4題完成了呢。

## 5. 混合：

在這一個部份，我利用圖片來說明，



當我們輸入指令後，會先判斷是哪一種，並進到適合的function，然後pipe和redirection會利用call function的方式，在進入判斷，判斷是哪一個部份，一直到全部都是Normal後，就可以順利解決混合的部分了

混合的輸出：

```
xiaowang@/home/xiaowang/OS/hw2> ls -al | grep 0
-rw-r--r-- 1 xiaowang xiaowang 10509 5月 6 16:19 0Shw2_s1043326.cpp
-rw-r--r-- 1 xiaowang xiaowang 10174 4月 12 20:29 0Shw2_s1043326_easy.cpp
xiaowang@/home/xiaowang/OS/hw2> ls -al | grep 0|sort > ouput
xiaowang@/home/xiaowang/OS/hw2> cat ouput
-rw-r--r-- 1 xiaowang xiaowang 10174 4月 12 20:29 0Shw2_s1043326_easy.cpp
-rw-r--r-- 1 xiaowang xiaowang 10509 5月 6 16:19 0Shw2_s1043326.cpp
xiaowang@/home/xiaowang/OS/hw2> cat < ouput|grep cpp|grep 20
-rw-r--r-- 1 xiaowang xiaowang 10174 4月 12 20:29 0Shw2_s1043326_easy.cpp
xiaowang@/home/xiaowang/OS/hw2> cat < ouput|grep cpp|grep 20 > ouput2
xiaowang@/home/xiaowang/OS/hw2> cat ouput2
-rw-r--r-- 1 xiaowang xiaowang 10174 4月 12 20:29 0Shw2_s1043326_easy.cpp
xiaowang@/home/xiaowang/OS/hw2>
```

混合心得：

在一開始，思考了很久要如何解決這個問題，  
最後想到可以利用這種call function的方式解決，  
好險有想到。

## 6. 加分題 – 自動補字：

加分題的部分，用readline就可以快速地解決了，  
而上下補字的部分則用add\_history這個readline  
的函式就可以解決了，code如下↓

```
49     input = readline( user );  
50     add_history( input );
```

```
xiaowang@/home/xiaowang/OS/hw2> o
```

↓ 按下tab後自動補完字

```
xiaowang@/home/xiaowang/OS/hw2> ouput
```

↓ 當出現檔名分歧時選單Show出全部選項

```
xiaowang@/home/xiaowang/OS/hw2> ouput  
ouput    ouput2  
xiaowang@/home/xiaowang/OS/hw2> ouput
```

```
xiaowang@/home/xiaowang/OS/hw2> ls ~  
examples.desktop  OS  test  下載  公共  圖片  影片  文件  桌面  模板  音樂  
xiaowang@/home/xiaowang/OS/hw2> ls ~
```

↑ 案”↑”可以找尋歷史紀錄

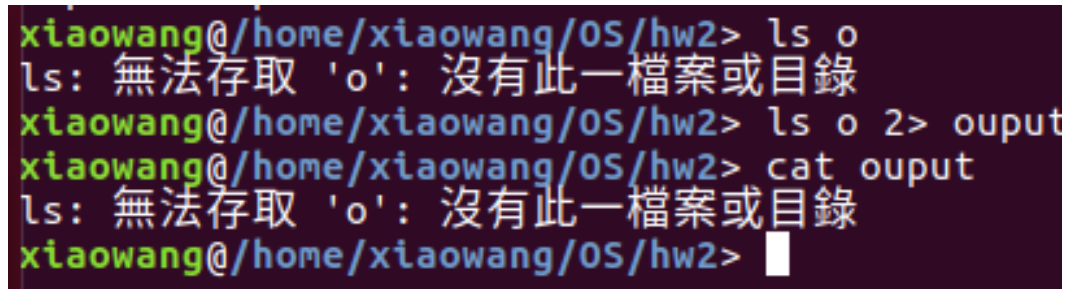
心得：

在上一次作業，其實我就做完了，也聽說因為助教readline的版本和我們得不相符，所以被扣分的事情呢。

加分題 - 2> (stderr導向)

在這個部份，其實也就只是把stderr導向檔案，所以我做完第4題也就做完了

輸出：



```
xiaowang@/home/xiaowang/OS/hw2> ls o
ls: 無法存取 'o': 沒有此一檔案或目錄
xiaowang@/home/xiaowang/OS/hw2> ls o 2> ouput
xiaowang@/home/xiaowang/OS/hw2> cat ouput
ls: 無法存取 'o': 沒有此一檔案或目錄
xiaowang@/home/xiaowang/OS/hw2> █
```

心得：

在做完第4題的時候，這個加分題也就完成了呢。

加分題 - ~

當使用者用到”~”的時候，外面的shell會幫我們把~轉成/home/user，然而在裡面，我們則要自己實作，code如下↓

```

31 void SetUserPath( char*( &target )){
32     char *tmpArgv = new char[ typeMax ](), *shrink = new char[ typeMax ]();
33     swap( target, tmpArgv );
34
35     strcat( target, getenv( "HOME" ));
36     shrink = strtok( tmpArgv, "~" );
37     if( shrink != NULL ) strcat( target, shrink );
38
39     tmpArgv = shrink = NULL;
40     delete [] tmpArgv;
41     delete [] shrink;
42 }

```

我們先將target和空的tmpArgv進行交換，然後再利用strcat將getenv( “HOME” )補進target，在將tmpArgv中沒~的地方加進來即可

輸出：

```

xiaowang@/home/xiaowang/OS/hw2> ls ~
examples.desktop OS 下載 公共 圖片 影片 文件 桌面 模板 音樂
xiaowang@/home/xiaowang/OS/hw2> cd ~
xiaowang@/home/xiaowang> cd OS/hw2
xiaowang@/home/xiaowang/OS/hw2> cat > ~/test
hello
xiaowang@/home/xiaowang/OS/hw2> cat < ~/test
hello
xiaowang@/home/xiaowang/OS/hw2> cat < ~/
.ICEauthority          .vboxclient-clipboard.pid
.Xauthority            .vboxclient-display.pid
.atom/                 .vboxclient-draganddrop.pid
.bash_history          .vboxclient-seamless.pid
.bash_logout           .viminfo
.bashrc                .xsession-errors
.cache/                .xsession-errors.old
.compiz/               OS/
.config/               examples.desktop
.dbus/                 test
.dmrc                  下載/
.gconf/                公共/
.local/                圖片/
.mozilla/              影片/
.pki/                  文件/

```

心得：

外面的shell有許多讓我意想不到的功能，也讓我覺得撰寫外面shell的人是真的非常非常得厲害。