Chapter 1: Start C-Language How To

Speaker: Lung-Sheng Chien

OutLine

- Course skeleton
- Introduction of programming language
- How to use Visual C++
- MSDN library
- Linux machine

Schedule: July

課程網頁：http://www.oz.nthu.edu.tw/~d947207/

日	一	二	三	四	五	六
		1	2	3 chapter 1	4 Chapter 2	5 Chapter 3
6 Chapter 4	7	8 Chapter 5	9 Chapter 6	10 Chapter 7	11 vim	12
13	14 數學營	15 數學營	16 數學營	17 數學營	18 數學營	19 數學營
20 數學營	21 數學營	22 數學營	23 數學營	24	25	26
27	28 Chapter 8	29 Chapter 9	30 Chapter 10	31		

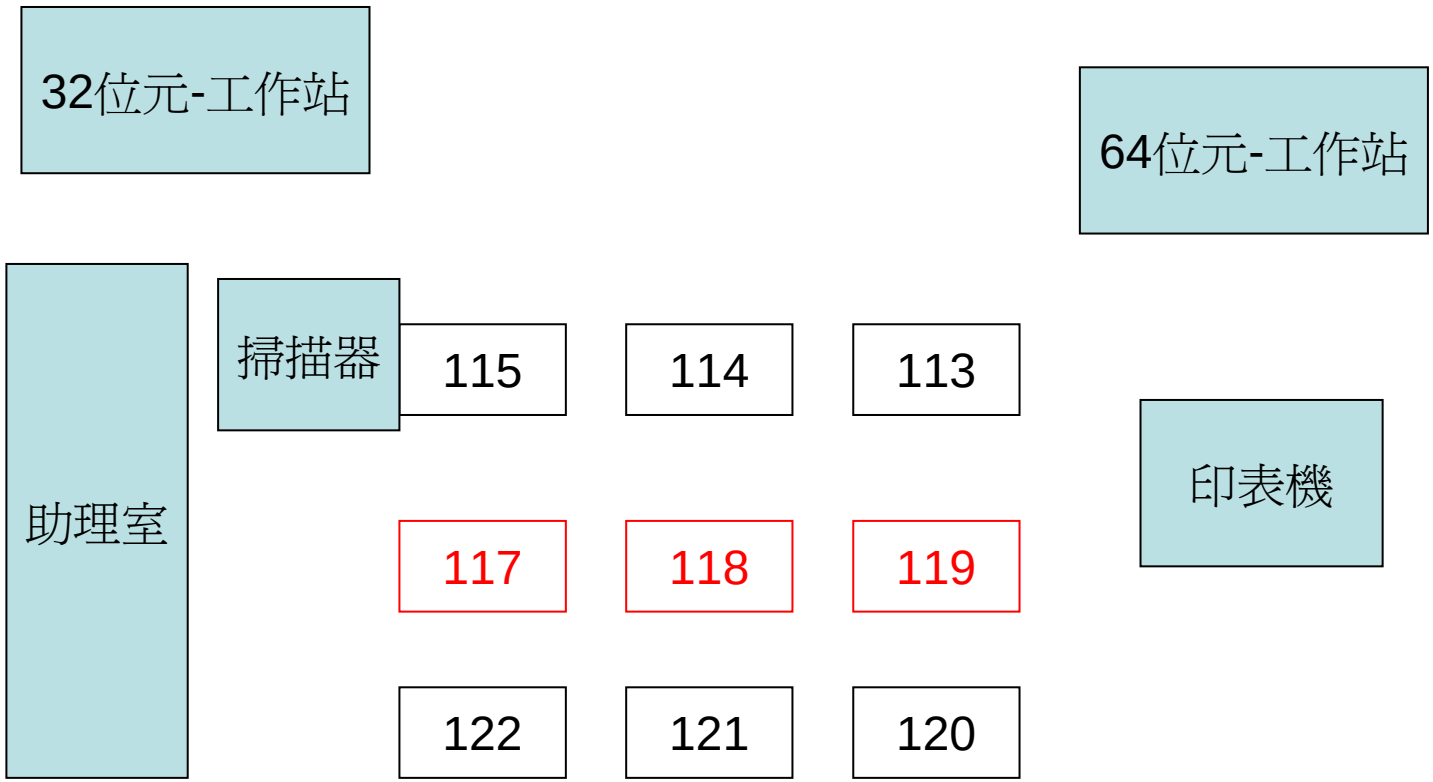
Workstations we have

IP	地點	OS	cpu	memory
140.114.34.1	R705	Fedora 7 64-bit	Intel(R) Xeon(R) CPU X5365 @ 3.00GHz , 2 cpu	64 GB
140.114.34.11	R705	Fedora 7 64-bit	Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz	8 GB
140.114.34.12	R705	Fedora 7 64-bit	Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz	8 GB
140.114.34.13	R705	Fedora 7 64-bit	Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz	8 GB
140.114.34.201	R705	RedHat 9 32-bit	Intel(R) XEON(TM) CPU 2.20GHz, 2 cpu	4 GB
140.114.34.214	R705	RedHat 9 32-bit	Intel(R) Pentium(R) 4 CPU 3.00GHz	2 GB
140.114.34.216	R705	RedHat 9 32-bit	Intel(R) Pentium(R) 4 CPU 3.00GHz	2 GB

Platform and resource

platform	Linux	Windows
compiler	gcc, g++ icpc (Intel C compiler)	vc 6.0 (Microsoft Visual Studio)
editor	vi	vc IDE interface
C++ document	?	MSDN Library 2008
GUI support	Qt	Qt
Makefile generator	qmake	qmake + vc

R705 (電腦室) floorplanning

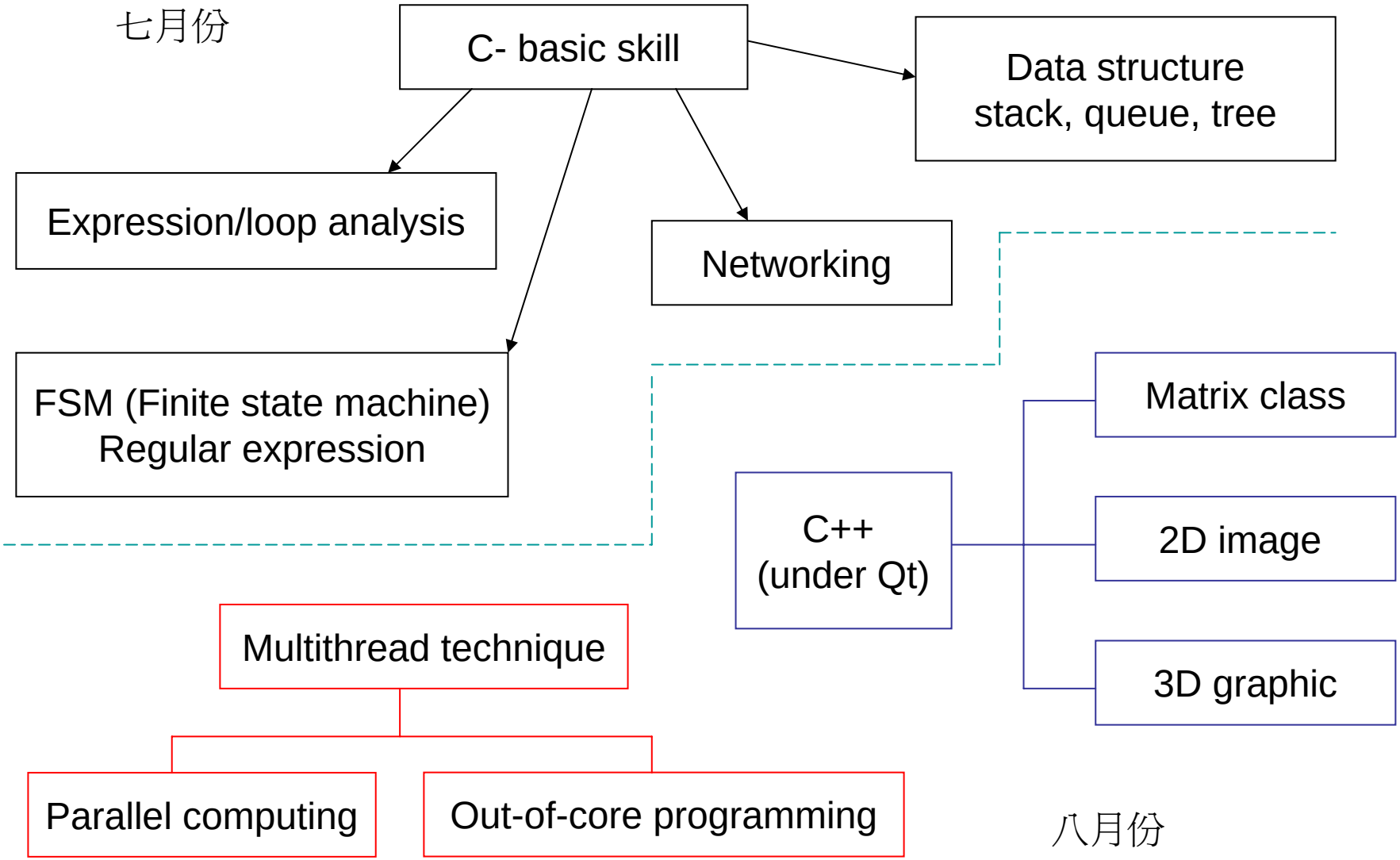


IP 140.114.34.117 ~119 are experimental computers in this course

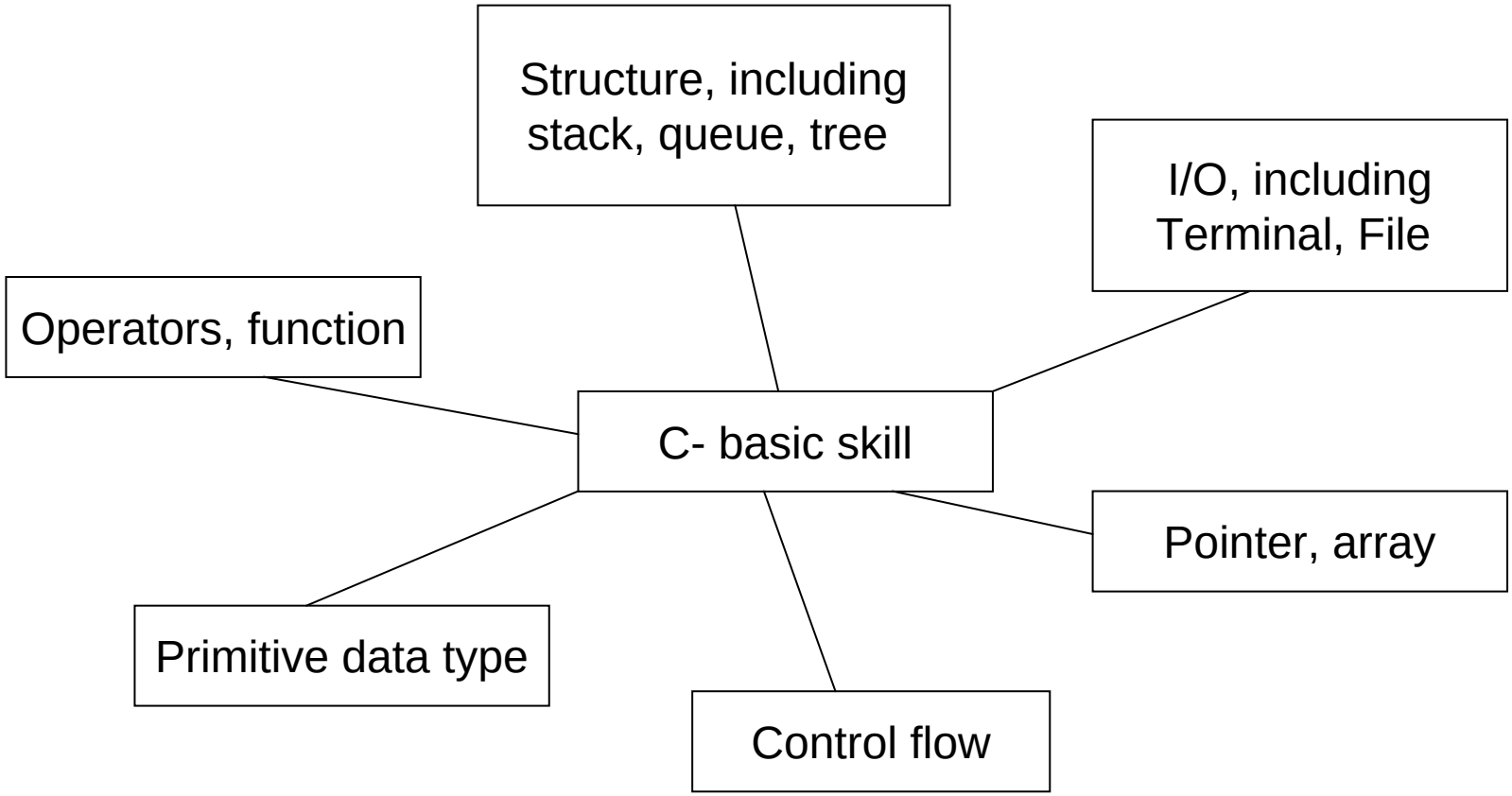
Software list in experimental computer

software	description
Visual studio 6.0	Write C/C++ source code, compile and link
MSDN library 2008	C++ document
Qt library	GUI programming and Makefile generator
ssh secure shell	login workstations, it use MD5-encryption for connection

What we must learn



Topics in C-language



TextBook: The C Programming Language, Kernighan

Delivery after this course

- MATLAB (interpreter), symbolic toolbox
- 2-Elevators system
- Out-of-core programming
- 2D image (image processing), GIS, GRASS
- 3D graphic (mesh generator), finite element
- Maze (老鼠走迷宮)
- Prime number
- Compiler issue, debugger
- Issue about economy

OutLine

- Course skeleton
- Introduction of programming language
- How to use Visual C++
- MSDN Library
- Linux machine

Sorts of Programming Language

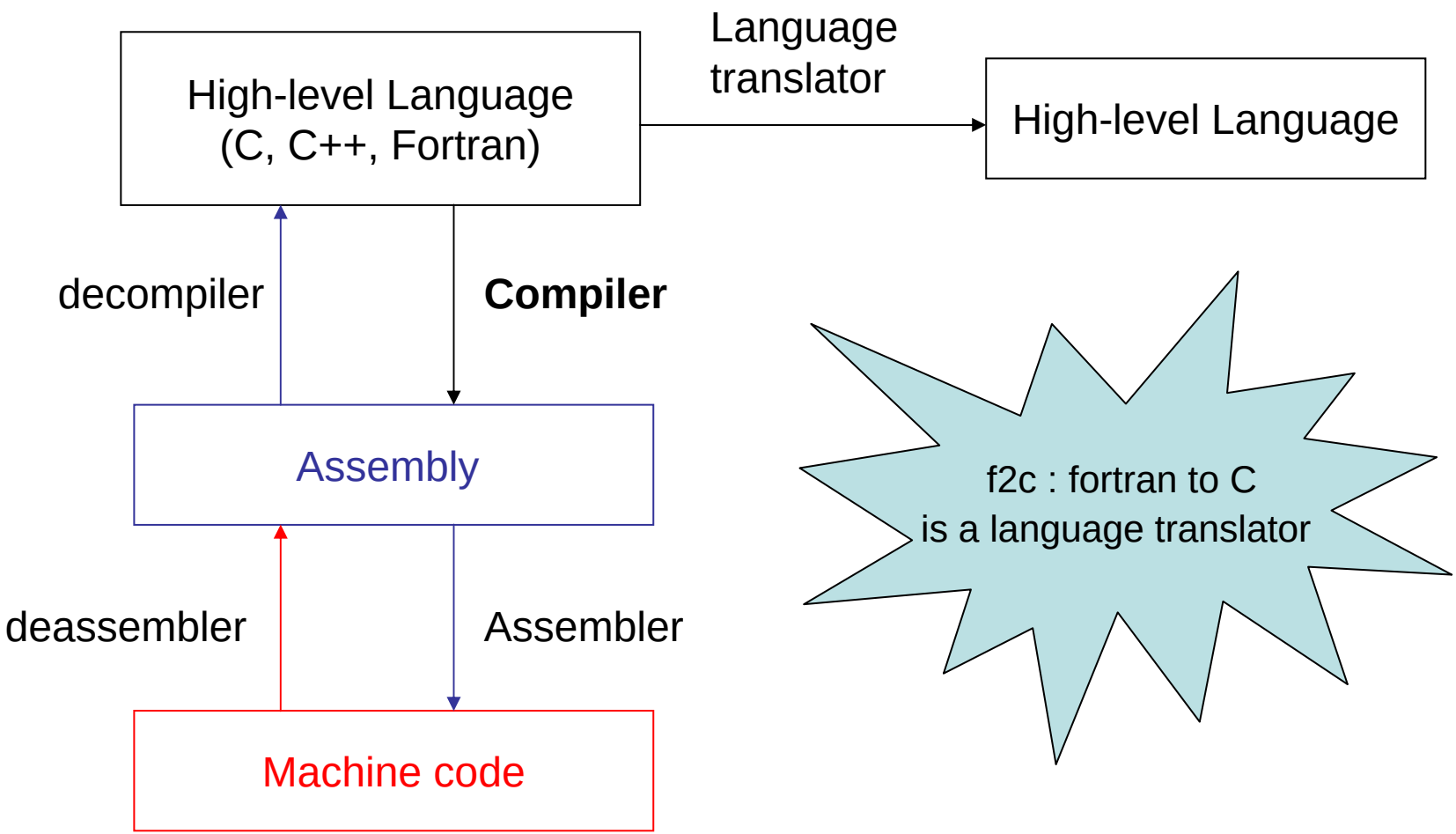
- High-level language (C, C++, Java, Fortran, Verilog, VHDL, COBOL, …), independent of Machine
- Low-level language, assembly, depends on machine’s architecture.
- Machine code, can be executed in cpu.Of course it depends on machine architecture.

Why not Assembly or Machine code?

- awkward (笨拙) and low readability
- Operation is atomic, we need more abstract-like programming style
- Performance is Human-tuning, time-consuming

Address	Label	Instruction (AT&T syntax)	Object code[9]
		.begin	
		.org 2048	
	a_start	.equ 3000	
2048		ld length,%	
2064		be done	00000010 10000000 00000000 00000110
2068		addcc %r1,-4,%r1	10000010 10000000 01111111 11111100
2072		addcc %r1,%r2,%r4	10001000 10000000 01000000 00000010
2076		ld %r4,%r5	11001010 00000001 00000000 00000000
2080		ba loop	00010000 10111111 11111111 11111011
2084		addcc %r3,%r5,%r3	10000110 10000000 11000000 00000101
2088	done:	jmpl %r15+4,%r0	10000001 11000011 11100000 00000100
2092	length:	20	00000000 00000000 00000000 00010100
2096	address:	a_start	00000000 00000000 00001011 10111000
		.org a_start	
3000	a:		

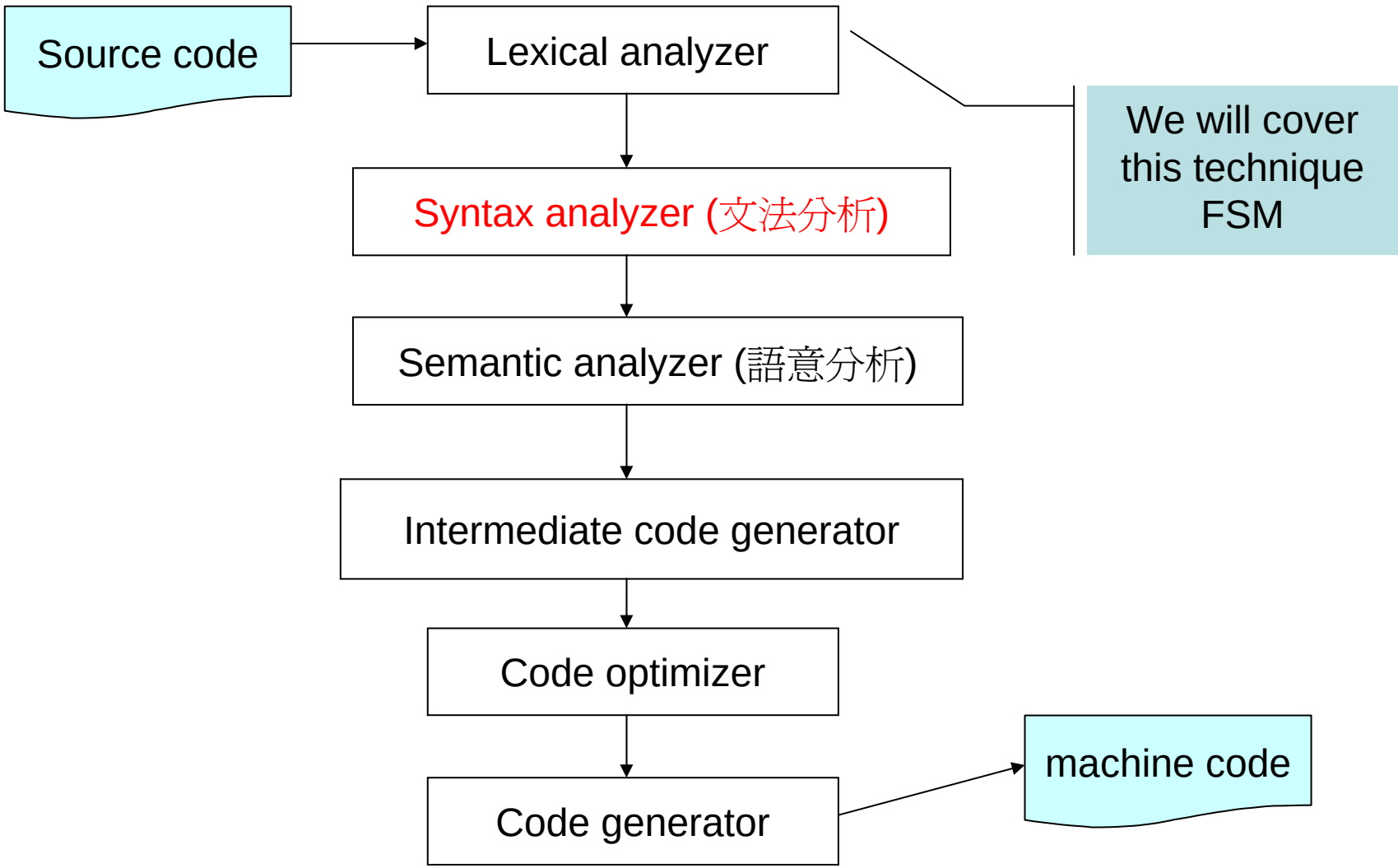
Hierarchical view of Language



C++ compiler we will use

Compiler	Author	Windows	Linux
gcc, g++	GNU	yes, with Cygwin	yes
Intel C++	Intel	yes	yes
Visual Studio	Microsoft	yes	no

Phases of a Compiler



Role of Compiler

- Shorten cycle of development
  - find the bugs
  - help programmers to write efficient and economic codes (relate to what language you use)

- Optimization, speed, low-power, …

- Code generation

Standard of C/C++

- ANSI C is current C-language standard, proposed by ANSI (American National Standard Institute 美國國家標準局)

- Microsoft visual C has additional keywords not in C-standard, however this is *o.k.* if we write C codes under standard rule.

OutLine

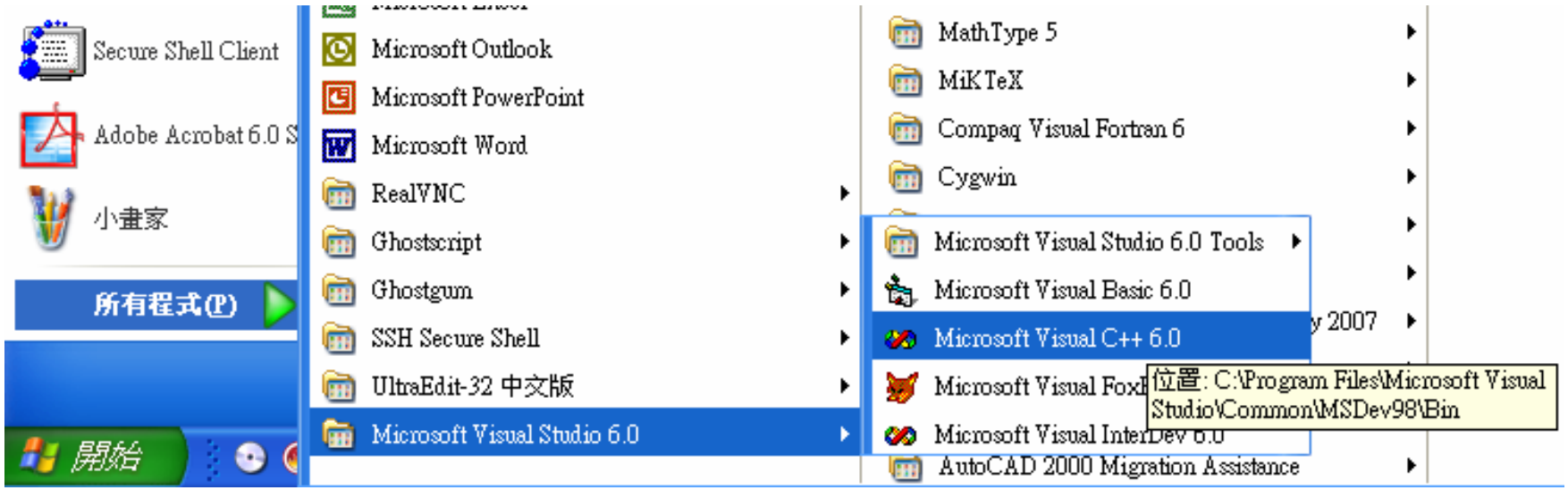
- Course skeleton
- Introduction of programming language
- How to use Visual C++
  - write “Hello World” program
- MSDN library
- Linux machine

Why visual studio (microsoft)

- IDE (Integrated Development Environment)介面
- Code editor
- Project management
- Debugger : weapon for learning C-language

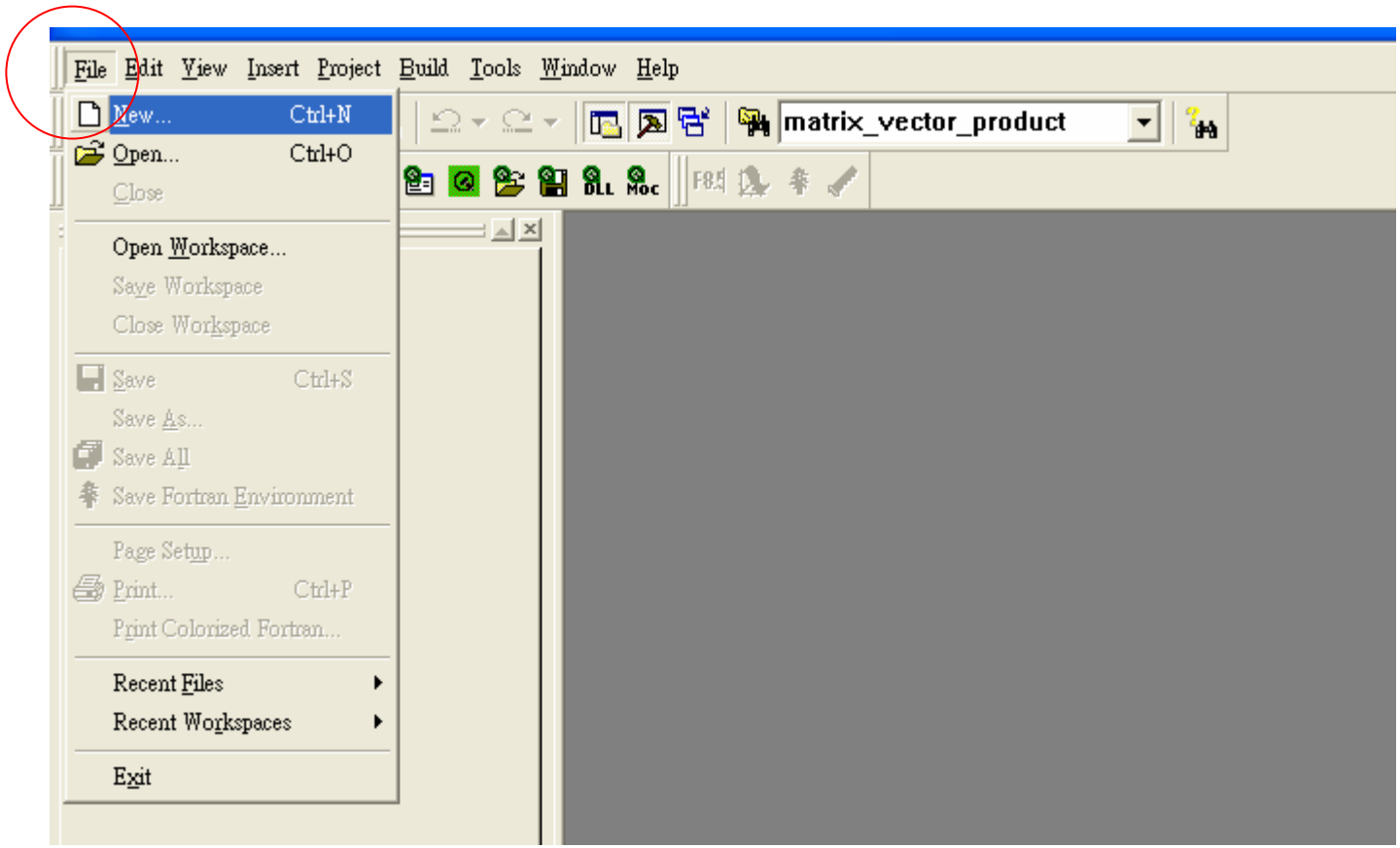
開啓Visual Studio IDE 介面

開始 → 程式集 → Microsoft Visual Studio 6.0 → Microsoft Visual C++ 6.0



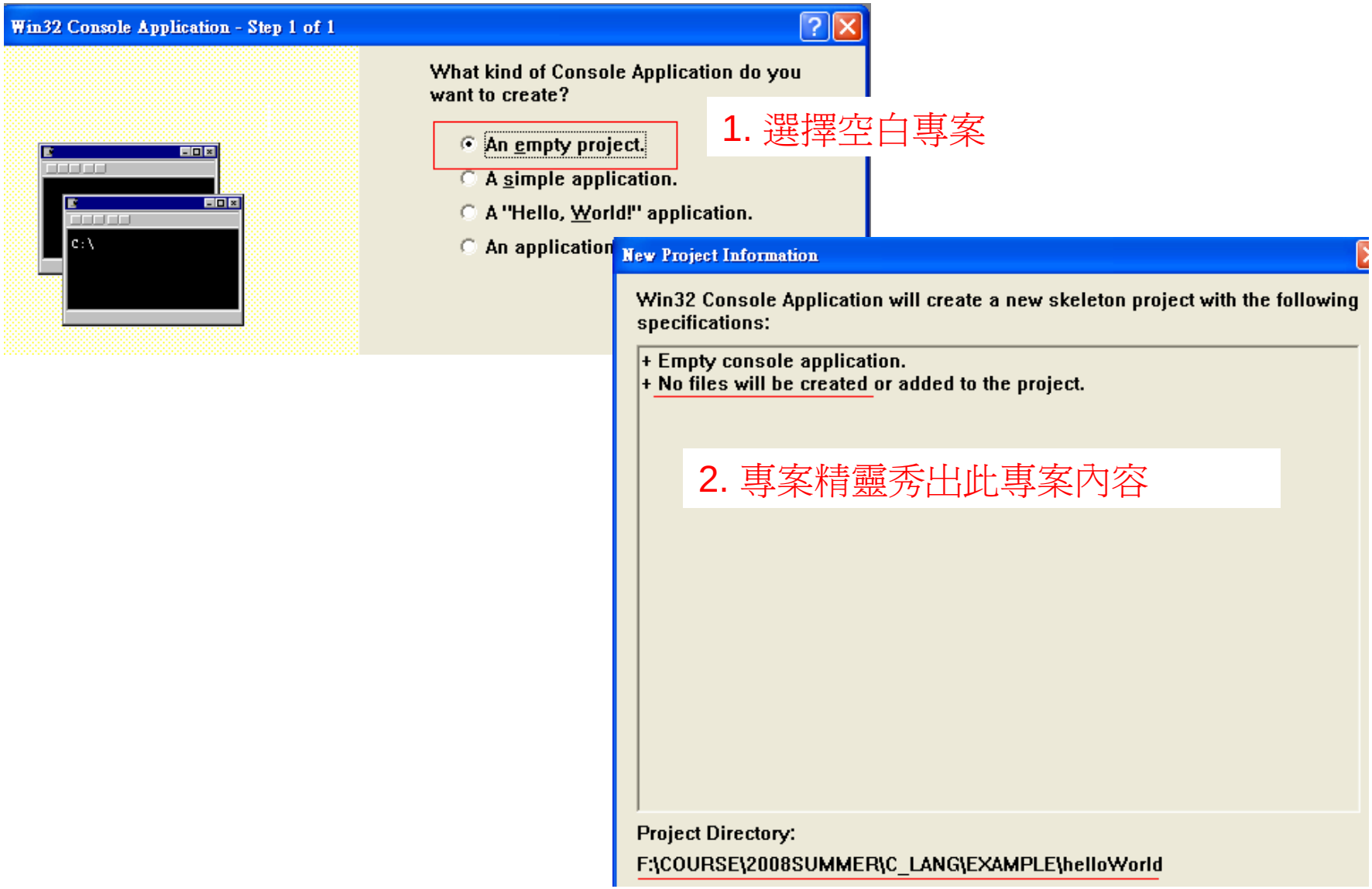
Step 1: 如何開啓新專案 (new project)

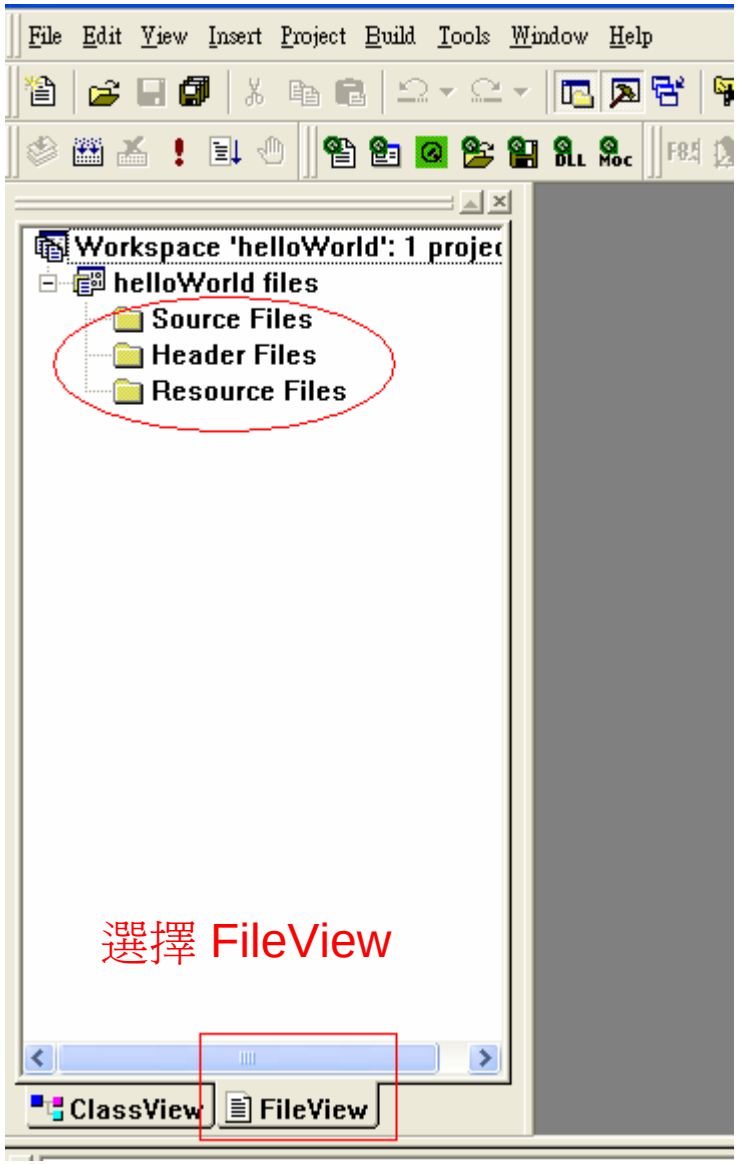
[File] → [New]



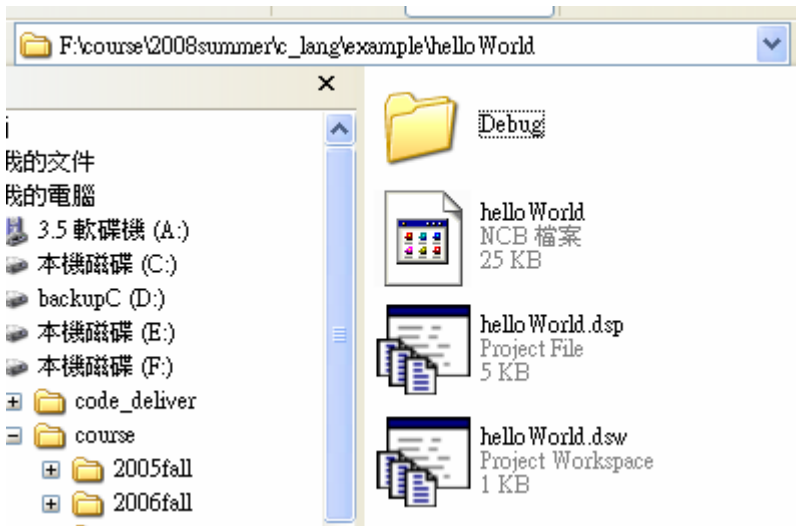
Step 2: 選擇 console應用程式 (非視窗型)

Step 3: 選取空白專案 (有專案骨架但無程式碼)



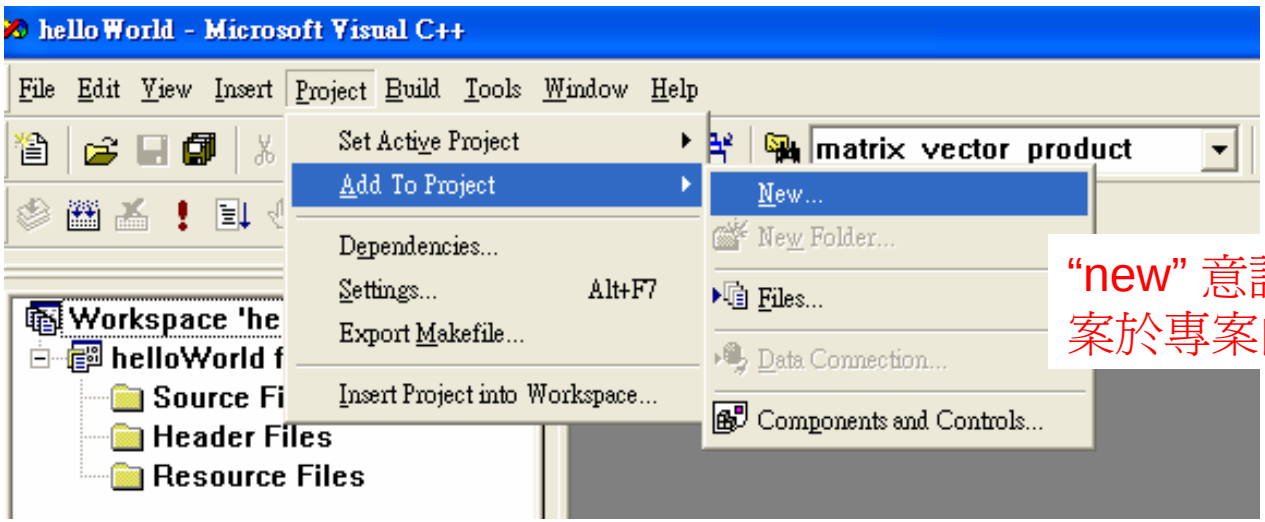


Click “FileView”, no files are in this project. In directory “helloWorld”, only project related files exist.



Step 4: 加入程式碼於此空白專案

[project] → [Add to Project] → [New]

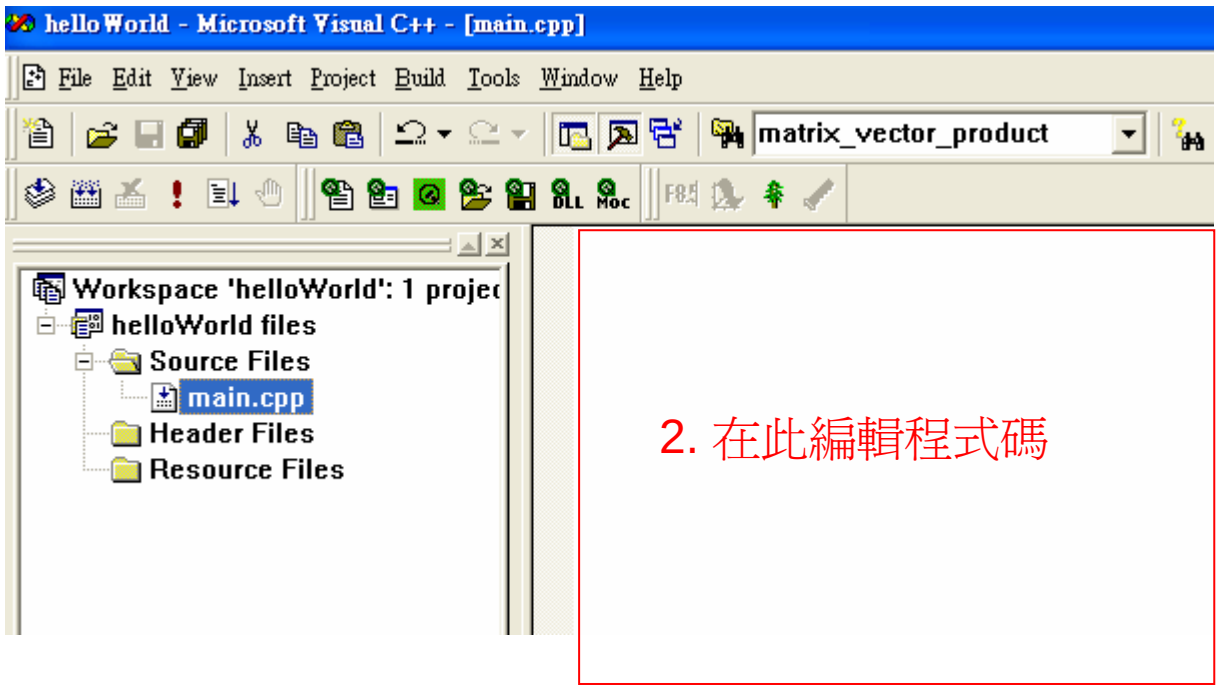


“new” 意謂加入一個新檔案於專案內

加入C++ 檔名 main.cpp 於專案內

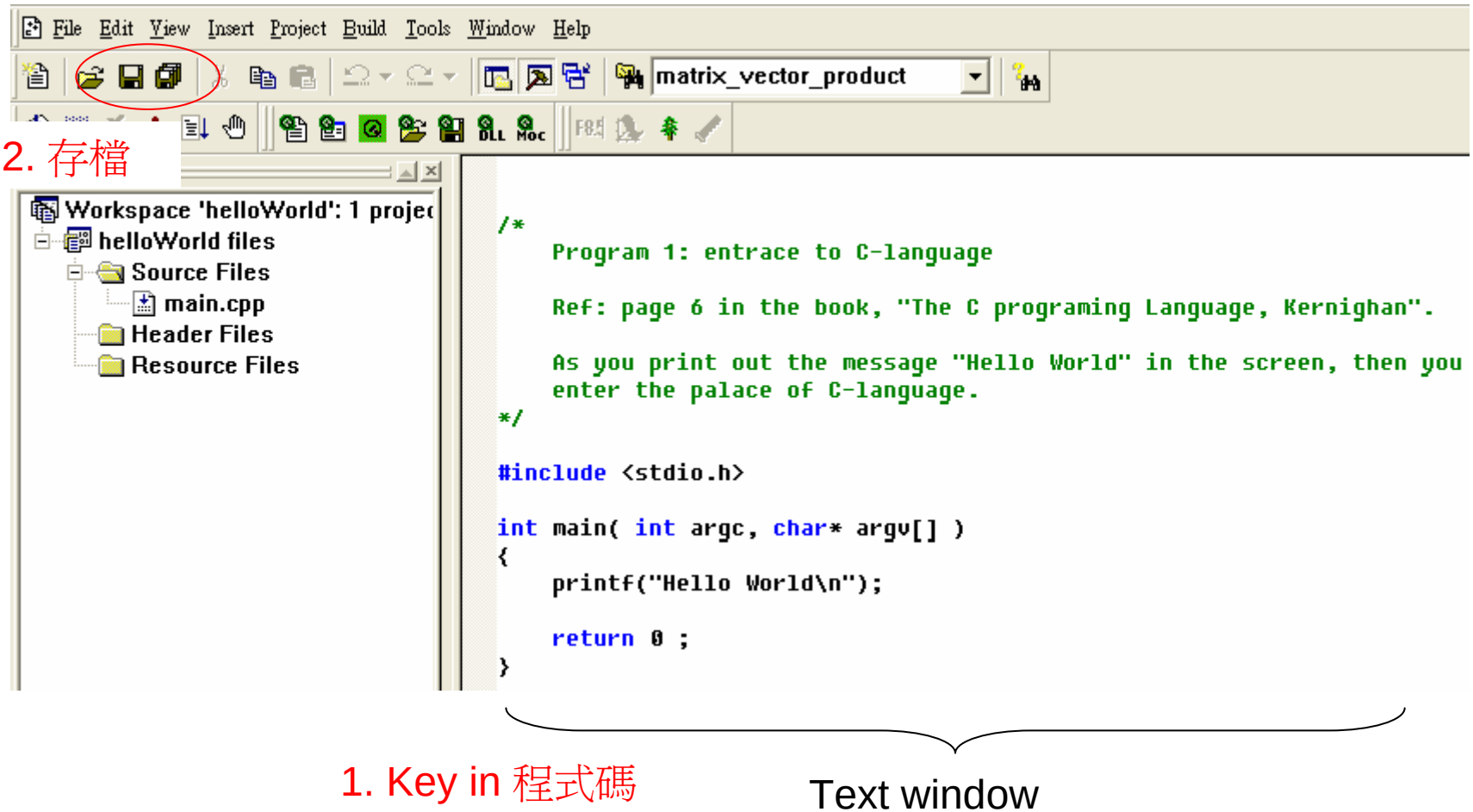
在 “Source Files” 的目錄下出現 main.cpp

1. 滑鼠點擊 main.cpp



2. 在此編輯程式碼

Step 5: 鍵入程式碼並存檔



1. Key in 程式碼

Text window

Step 6: Compilation (編譯) : translate source code to object code

1. [Build] → [Compile main.cpp]

File Edit View Insert Project Build Tools Window Help

Compile main.cpp Ctrl+F7
Build helloWorld.exe F7
Rebuild All
Batch Build...
Clean
Update All Dependencies...
Start Debug
Debugger Remote Connection...
Execute helloWorld.exe Ctrl+F5
Set Active Configuration...
Configurations...
Profile...

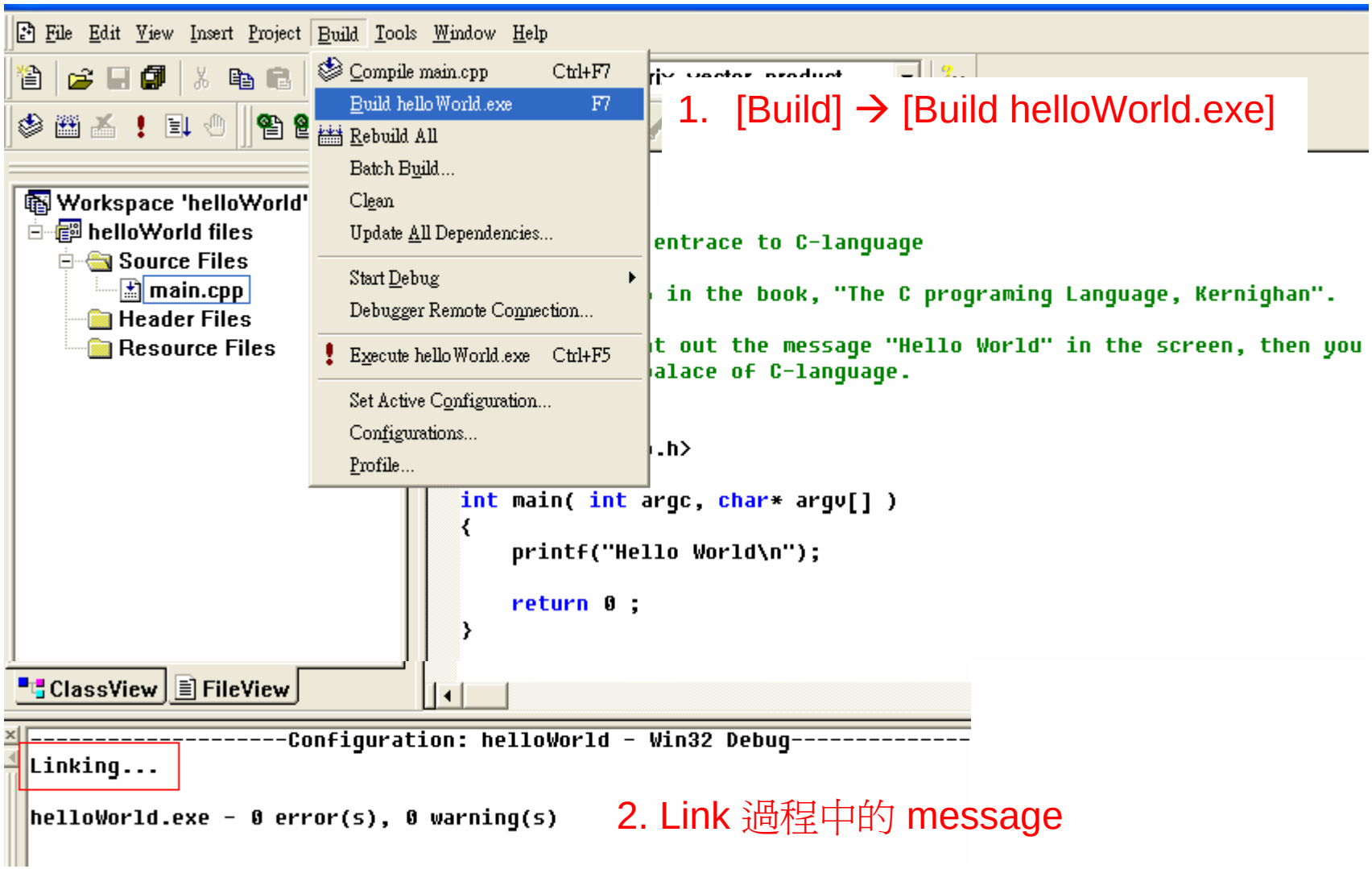
Workspace 'helloWorld'
helloWorld files
Source Files
main.cpp
Header Files
Resource Files

```
entrace to C-language

in the book, "The C programing Language, Kernighan".

t out the message "Hello World" in the screen, then you
alace of C-language.

.h>

int main( int argc, char* argv[] )
{
    printf("Hello World\n");

    return 0 ;
}
```

```
--------------------Configuration: helloWorld - Win32 Debug---------
Compiling...
main.cpp

main.obj - 0 error(s), 0 warning(s)
```

2. 編譯時的資訊

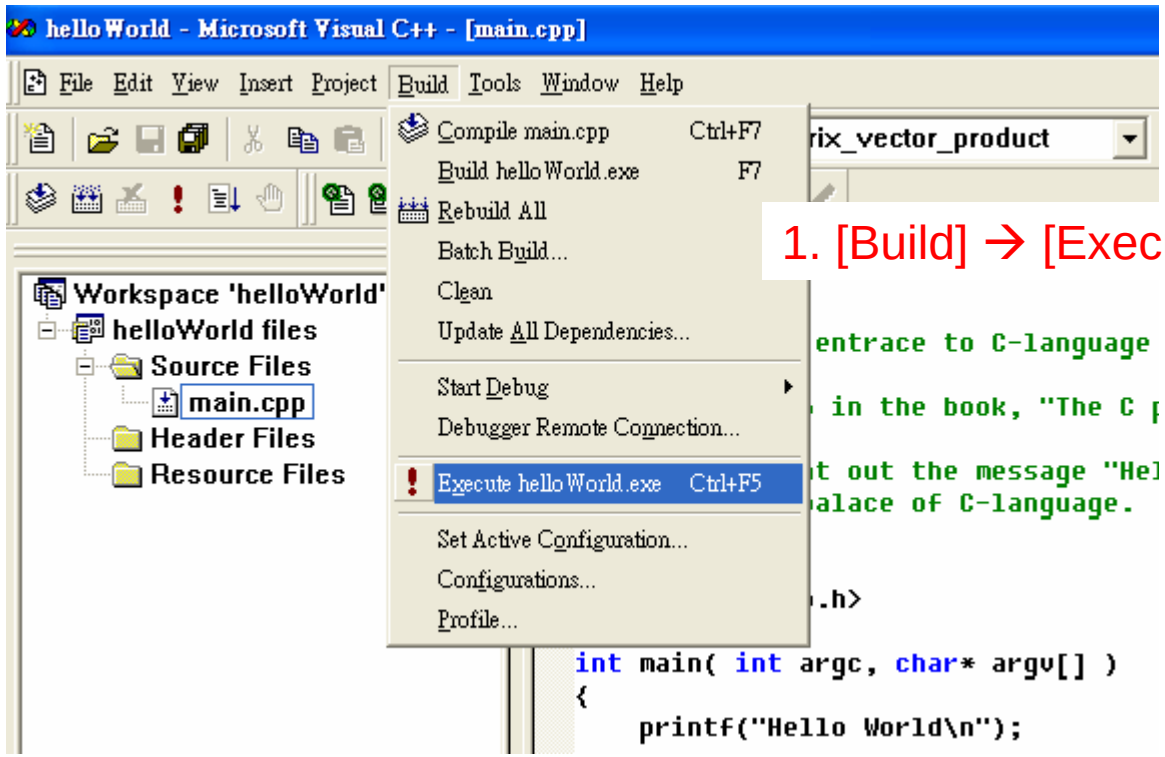
Step 7: Build (Linking phase, 鏈結) : combine object code into a executable file



1. [Build] → [Build helloWorld.exe]

2. Link 過程中的 message

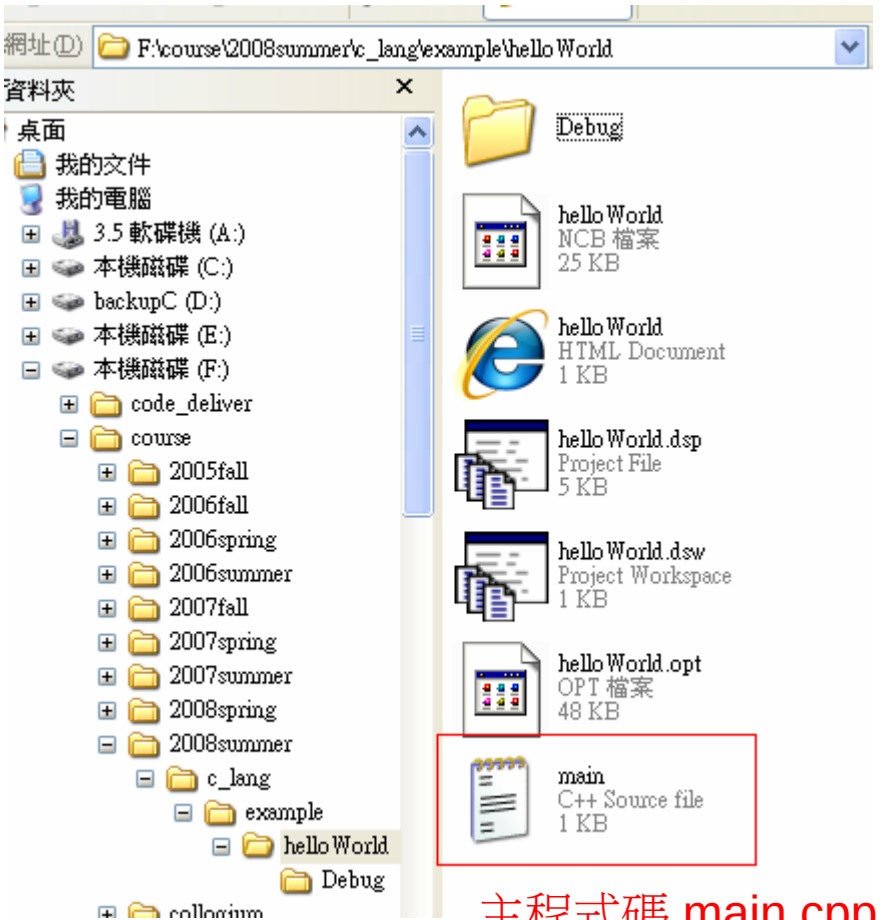
Step 8: Execution (執行helloWorld.exe)



1. [Build] → [Execute helloWorld.exe]

2. 執行結果

What we have done!



主程式碼 main.cpp

執行檔

Object code by compilation

Source code's interpretation

comments

A compiler regards characters between /* and */ as comments and ignores them.

```
/*
    Program 1: entrace to C-language

    Ref: page 6 in the book, "The C programing Language, Kernighan".

    As you print out the message "Hello World" in the screen, then you
    enter the palace of C-language.
*/

#include <stdio.h>

int main( int argc, char* argv[] )
{
    printf("Hello World\n");

    return 0 ;
}
```

Function body

Include information about standard library

Std: standard, io: Input/Output

main is a function with 2 arguments and return integer (int)

main calls library function **printf** to print string constant "Hello world" into screen

"return 0" corresponds to return type "int" of main

Key sentences

標頭檔

```
#include <stdio.h>
```

回傳型別

引數 (argument)

```
int main( int argc, char* argv[] )
```

函數原型 (prototype)

函數名字

參數 (parameter)

```
printf("Hello World\n");
```

呼叫函數 printf

```
return 0 ;
```

回傳整數 0

函數定義 (definition)

```
int main( int argc, char* argv[] )
{
    • • •
    return 0 ;
}
```

Purpose of #include <stdio.h>

When compiler read " printf("Hello World\n"); ", it would recognize it is a function with function name printf, then compiler would do "type checking", say one must declare prototype of function printf first such that compiler can do type checking.

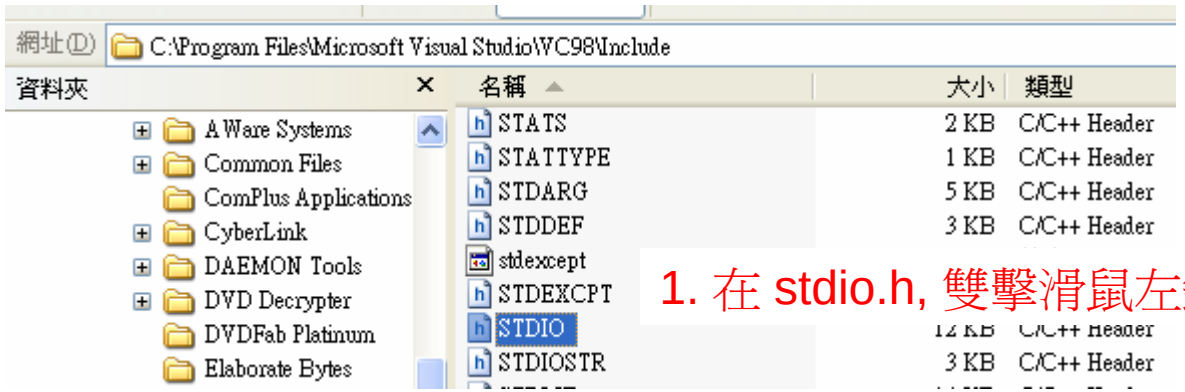
Example: comment #include <stdio.h>, then compile again, error occurs

```
//#include <stdio.h>

int main( int argc, char* argv[] )
{
    printf("Hello World\n");

    return 0 ;
}
```

ClassView FileView

```
main.cpp
f:\course\2008summer\c_lang\example\helloworld\main.cpp(16) : error C2065: 'printf' : undeclared identifier
Error executing cl.exe.

main.obj - 1 error(s), 0 warning(s)
```

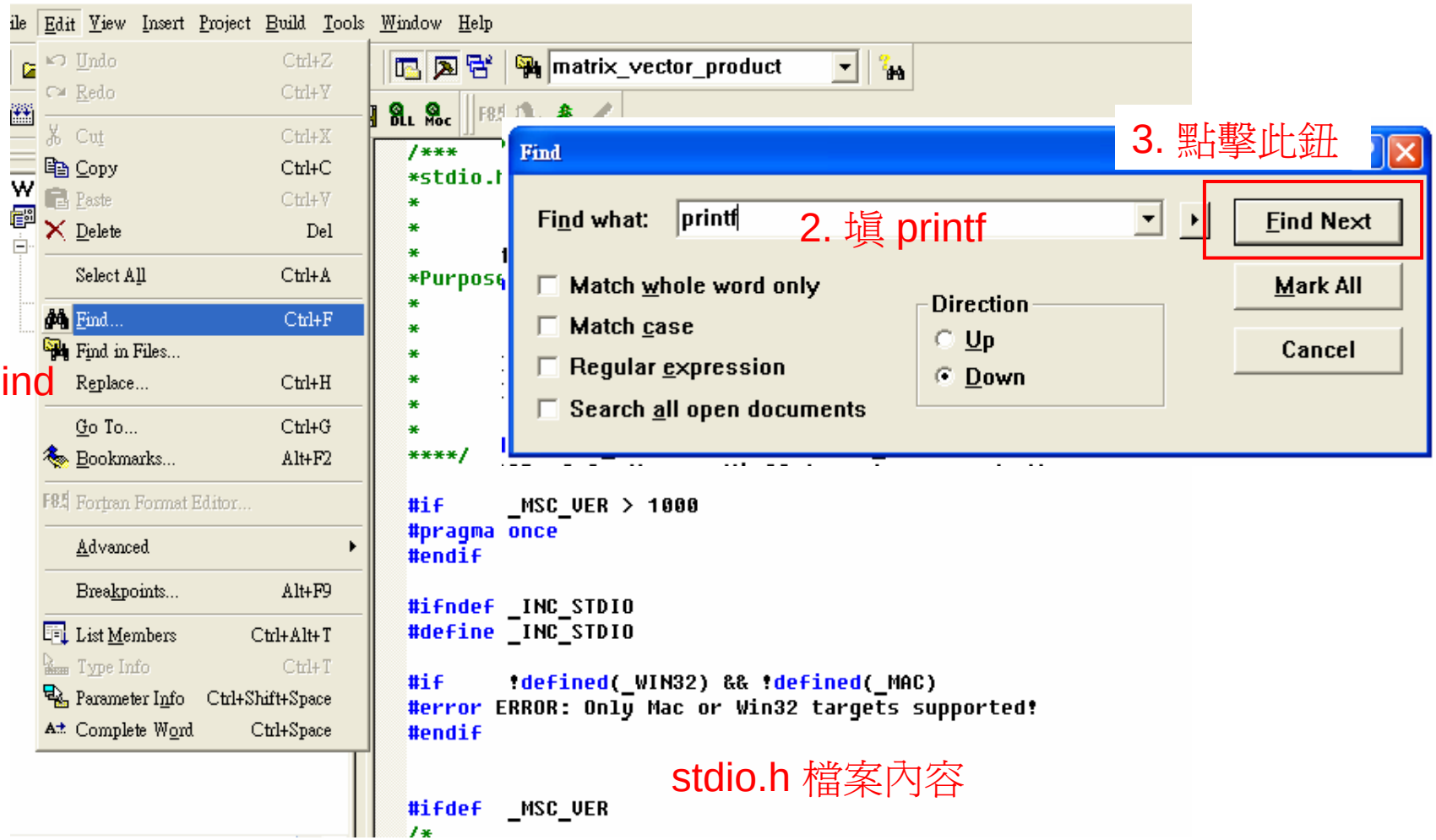
在目錄C:\Program Files\Microsoft Visual Studio\VC98\Include中打開檔案stdio.h



1. 在 stdio.h, 雙擊滑鼠左鍵 或

2. 在 stdio.h, 點擊滑鼠右鍵, 選擇由 Visual Studio開啓此檔

[Edit] → [Find] → 填入printf → 點擊按鈕 “Find Next”



Find prototype of printf in file stdio.h

```
_CRTIMP int __cdecl _flushall(void);
_CRTIMP FILE * __cdecl fopen(const char *, const char *);
_CRTIMP int __cdecl fprintf(FILE *, const char *, ...);
_CRTIMP int __cdecl fputc(int, FILE *);
_CRTIMP int __cdecl _fputchar(int);
_CRTIMP int __cdecl fputs(const char *, FILE *);
```

Not “printf”, 按 F3 尋找下一個

```
_CRTIMP int __cdecl _getw(FILE *);
_CRTIMP void __cdecl perror(const char *);
_CRTIMP int __cdecl _pclose(FILE *);
_CRTIMP FILE * __cdecl _popen(const char *, const char *);
_CRTIMP int __cdecl printf(const char *, ...);
_CRTIMP int __cdecl putc(int, FILE *);
_CRTIMP int __cdecl putchar(int);
_CRTIMP int __cdecl puts(const char *);
```

This is prototype of function printf

```
int  printf(const char *, ...);
```

字串

printf("Hello World\n");

“Hello World” 是字串,
type checking 成功

Declare prototype of printf before using it

```
//#include <stdio.h>
#ifdef  __cplusplus
extern "C" {
    int printf(const char *, ...);
}
#endif

int main( int argc, char* argv[] )
{
    printf("Hello World\n");

    return 0 ;
}
```

File stdio.h is not included

1. Declare prototype of printf

2. Call function printf

Keywords, “ifdef”, “extern”, “__cplusplus”, are explained later

Error: use printf before declaring its prototype, why?

```
//#include <stdio.h>


int main( int argc, char* argv[] )
{
    printf("Hello World\n");

    return 0 ;
}

#ifdef  __cplusplus
extern "C" {
    int printf(const char *, ...);
}
#endif
```

Compiler does not see any prototype

```
Compiling...
main.cpp
f:\course\2008summer\c_lang\example\helloworld\main.cpp(17) : error C2065: 'printf' : undeclared identifier
f:\course\2008summer\c_lang\example\helloworld\main.cpp(24) : error C2373: 'printf' : redefinition; different type modifiers
Error executing cl.exe.

main.obj - 2 error(s), 0 warning(s)
```

Why function “main” has no prototype?

- “main” is an entry point of program, it is unique, say only one main can appear.
- “main” has the definition, which is enclosed by a pair of brace.

```
#include <stdio.h>

int main( int argc, char* argv[] ) ;

int main( int argc, char* argv[] )
{
    printf("Hello World\n");

    return 0 ;
}
```

Declare prototype of function main

Definition of “main”

Mismatch between prototype and definition

```
#include <stdio.h>

int main( ) ;

int main( int argc, char* argv[] )
{
    printf("Hello World\n");

    return 0 ;
}
```

Prototype of “main”

Definition of “main”

```
--------------------Configuration: helloWorld - Win32 Debug--------------------
Compiling...
main.cpp
F:\course\2008summer\c_lang\example\helloWorld\main.cpp(17) : error C2731: 'main' : function cannot be overloaded
        F:\course\2008summer\c_lang\example\helloWorld\main.cpp(16) : see declaration of 'main'
Error executing cl.exe.

main.obj - 1 error(s), 0 warning(s)
```

Overloaded is a C++ keyword, we will interpret later

OutLine

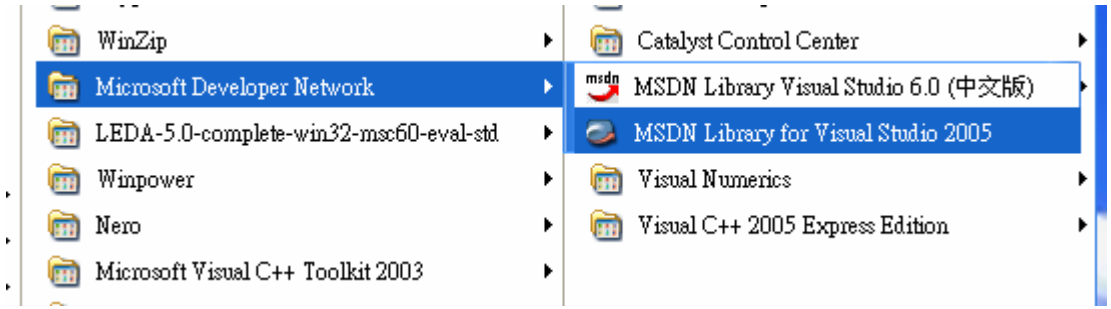
- Course skeleton
- Introduction of programming language
- How to use Visual C++
- MSDN library
- Linux machine

MSDN Library

- MicroSoft Developer Network
- Includes sample code, technical articles, and C/C++ standard description
- It is free, you can download it from microsoft’s download center

開啓 MSDN Library

開始 → 所有程式 → Microsoft Developer Network → MSDN Library

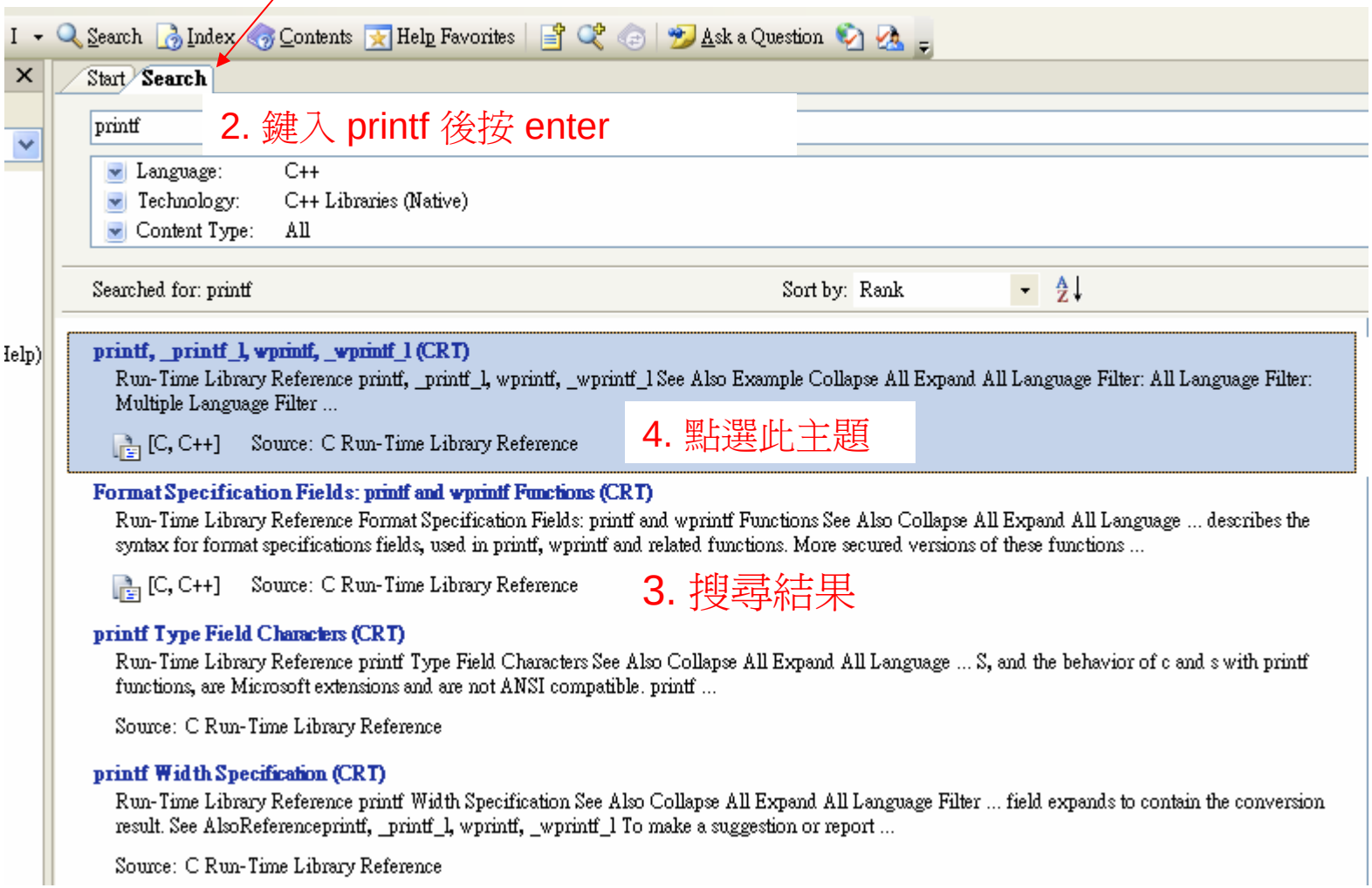


1. 啓動MSDN Library

2. MSDN Library 首頁

搜尋 printf 相關文章 [1]

1. 點選 search



搜尋 printf 相關文章 [2]

printf, _printf_l, wprintf, _wprintf_l | Search

URL: ms-help://MS.MSDNQTR.v80.en/MS.MSDN.v80/MS.VisualStudio.v80.en/dv_vccrt/html

Run-Time Library Reference

printf, _printf_l, wprintf, _wprintf_l

See Also Example

Collapse All Language Filter: Multiple

Print formatted output to the standard output stream. More secure versions a

```
int printf(
   const char *format [,
      argument]...
);
int _printf_l(
   const char *format,
   locale_t locale [,
      argument]...
);
int wprintf(
   const wchar_t *format [,
      argument]...
);
int _wprintf_l(
   const wchar_t *format,
   locale_t locale [,
      argument]...
);
```

Prototype of printf

Example code, you can copy it and test it

Example

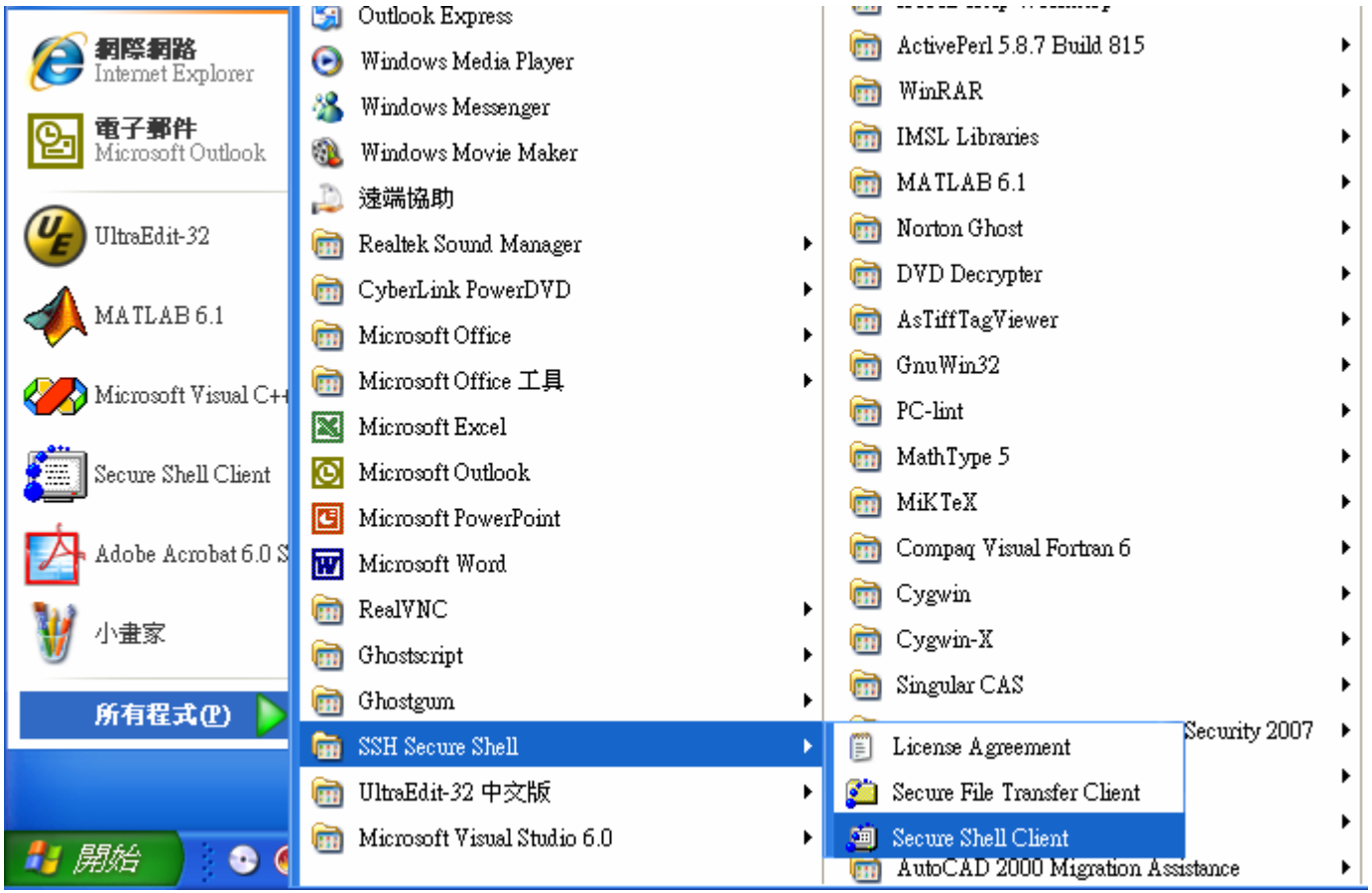
```
// crt_printf.c
// This program uses the printf and wprintf func
// to produce formatted output.

#include <stdio.h>

int main( void )
{
   char     ch = 'h',
            *string = "computer";
   wchar_t  wch = L'w',
            *wstring = L"Unicode";
   int      count = -9234;
   double   fp = 251.7366;

   // Display integers
   printf( "Integer formats:\n"
           "   Decimal: %d  Justified: %.6d  "
           "Unsigned: %u\n",
           count, count, count, count );
```

OutLine

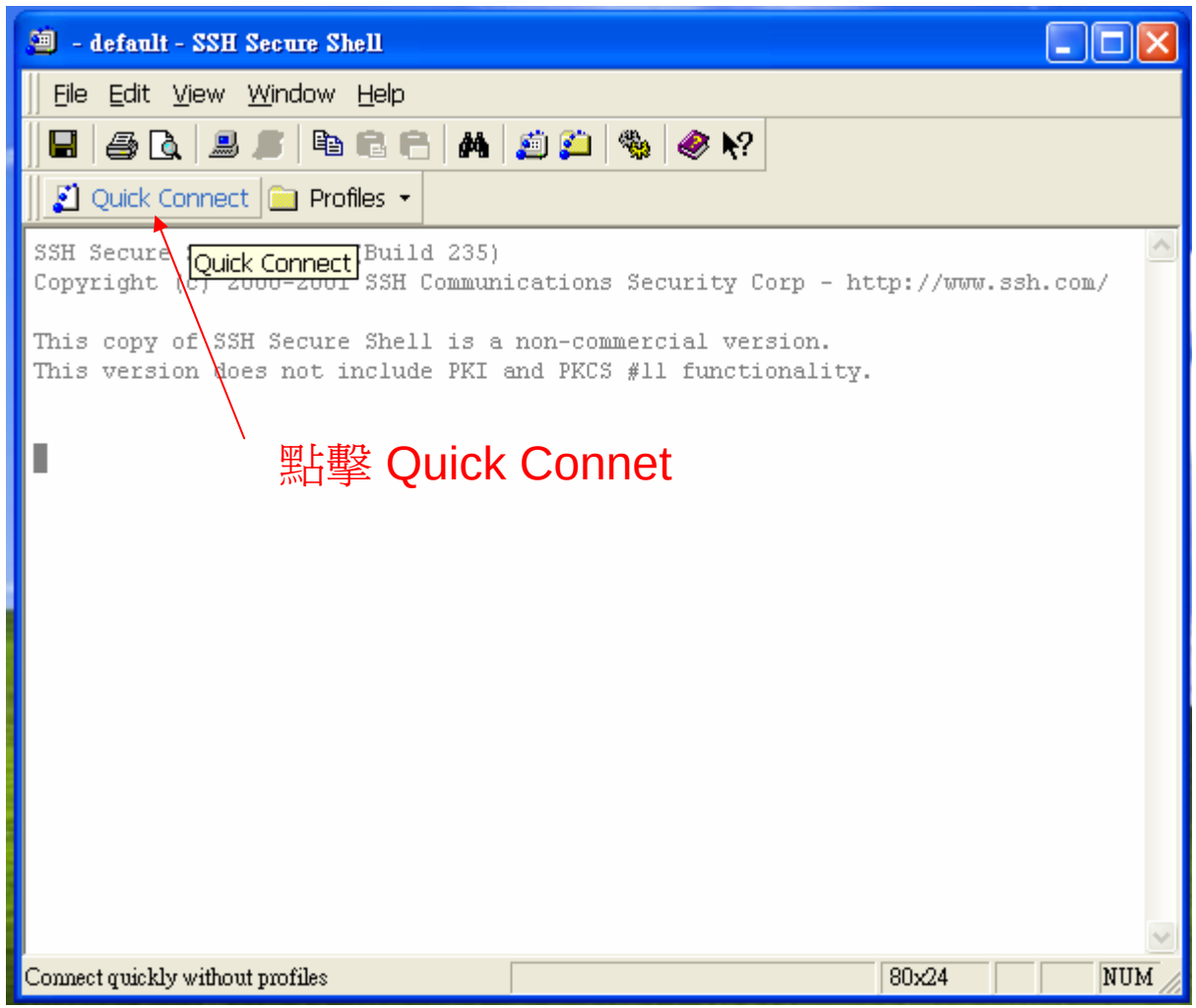
- Course skeleton
- Introduction of programming language
- How to use Visual C++
- MSDN library
- Linux machine
  - use ssh to login remote machine
  - commands in Linux machine
  - How to compile

開啓 ssh 通訊程式 (MD5 加密)

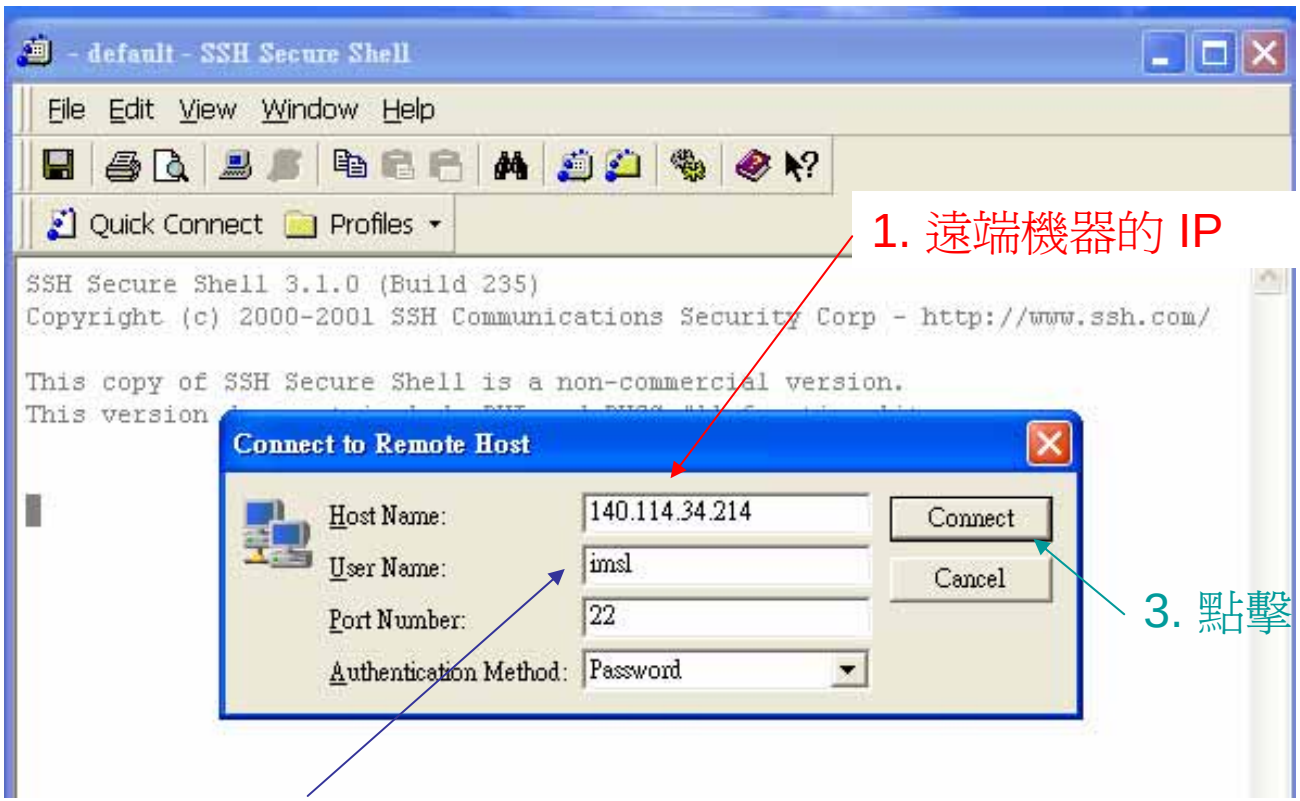
程式集 → SSH Secure Shell → Secure Shell Client



建立新連線 [1]


- default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
Quick Connect
SSH Secure (Build 235)
Copyright (c) 2000-2001 SSH Communications Security Corp - http://www.ssh.com/
This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.
點擊 Quick Connet
Connect quickly without profiles
80x24
NUM

建立新連線 [2]

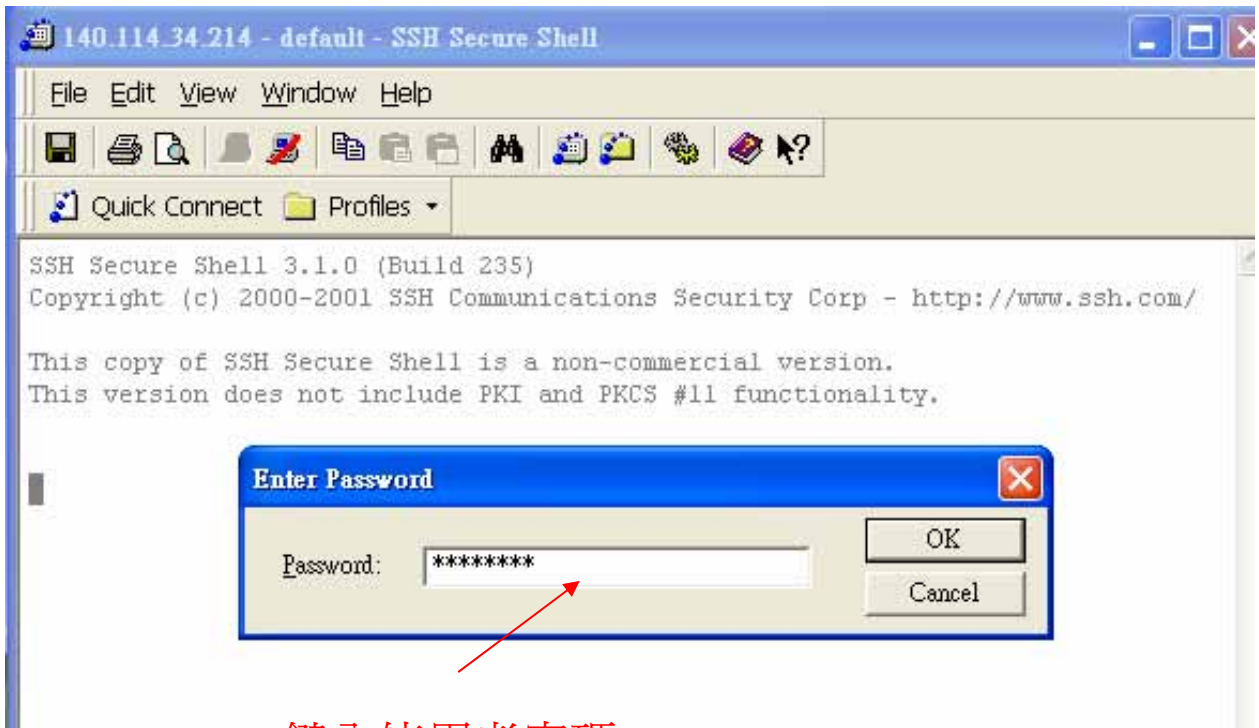


1. 遠端機器的 IP

2. 使用者帳號

3. 點擊 Connect 按紐

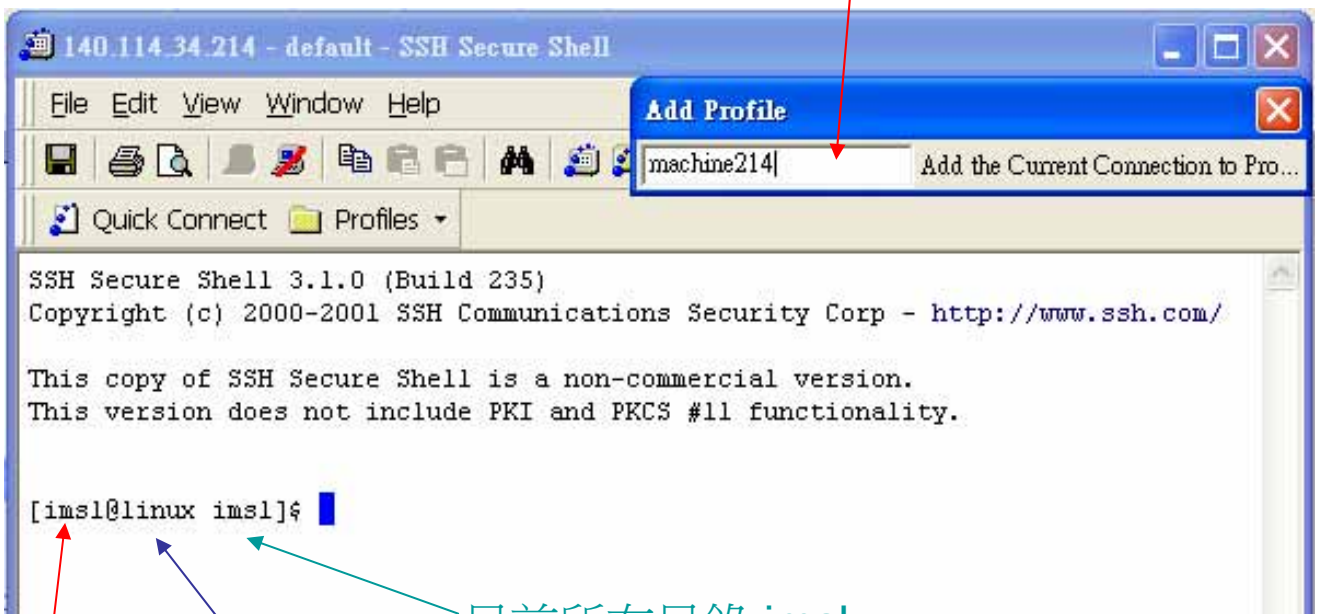
建立新連線 [3]



鍵入使用者密碼

建立新連線 [4]

1. 鍵入此機器代碼



目前所在目錄 imsl

機器名稱 linux

使用者帳號 imsl

Commands in common use

command	Description
passwd	Change password
pwd	Current working directory
ls	List all files and subdirectory in current directory
cd	Change directory
mkdir	Make a new directory
rm	Remove a file/directory
top	Show process information
cat /proc/cpuinfo	Show cpu’s information
cat /proc/meminfo	Show memory’s information
uname -a	Show machine’s information
man	Look up manual of commands
icc, icpc	Intel C/C++ compiler
gcc, g++	GNU C/C++ compiler
which	Show full path of commands

uname -a

```
[imsl@linux imsl]$
[imsl@linux imsl]$ uname -a
Linux linux.am.nthu.edu.tw 2.4.20-8smp #1 SMP Thu Mar 13 17:45:54 EST 2003 i686
i686 i386 GNU/Linux
[imsl@linux imsl]$
```

linux.am.nthu.edu.tw : domain name

2.4.20-8smp : 作業系統 RedHat9 核心版本

I686: 32位元機器，x86_64: 64位元機器

```
[macrold@quartetl ~]$ uname -a
Linux quartetl.am.nthu.edu.tw 2.6.23.15-80.fc7 #1 SMP Sun Feb 10 16:52:18 EST 20
08 x86_64 x86_64 x86_64 GNU/Linux
```

cat /proc/cpuinfo

```
[imsl@linux imsl]$
[imsl@linux imsl]$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 3
model name      : Intel(R) Pentium(R) 4 CPU 3.00GHz
stepping        : 3
cpu MHz         : 3014.560
cache size      : 1024 KB
physical id     : 0
siblings        : 2
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 3
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips        : 6016.20
```

1. CPU 是 Pentium 4

2. Actual running clock rate

cat /proc/meminfo

所有記憶體大小 爲 2GB

虛擬記憶體爲 2GB

```
[imsl@linux imsl]$
[imsl@linux imsl]$ cat /proc/meminfo
        total:    used:    free:  shared: buffers:  cached:
Mem:  2113986560 429600768 1684385792        0 105287680 120418304
Swap: 2097434624        0 2097434624
MemTotal:      2064440 kB
MemFree:       1644908 kB
MemShared:           0 kB
Buffers:        102820 kB
Cached:         117596 kB
SwapCached:          0 kB
Active:         236416 kB
ActiveAnon:      23168 kB
ActiveCache:    213248 kB
Inact_dirty:      5112 kB
Inact_laundry:       0 kB
Inact_clean:      2056 kB
Inact_target:    48716 kB
HighTotal:     1179584 kB
HighFree:      1032636 kB
LowTotal:       884856 kB
LowFree:        612272 kB
SwapTotal:     2048276 kB
SwapFree:      2048276 kB
[imsl@linux imsl]$
```

top

```
[imsl@linux imsl]$
[imsl@linux imsl]$ top
```

所有記憶體大小 爲 2GB

虛擬記憶體爲 2GB

```
 14:53:41  up 11 days, 23:18,  1 user,  load average: 0.00, 0.00, 0.00
48 processes: 47 sleeping, 1 running, 0 zombie, 0 stopped
CPU0 states:   0.4% user   0.0% system    0.0% nice   0.0% iowait  99.1% idle
CPU1 states:   0.1% user   0.0% system    0.0% nice   0.0% iowait  99.4% idle
Mem:  2064440k av,  419528k used, 1644912k free,       0k shrd,  102820k buff
                    236676k actv,    5112k in_d,    2056k in_c
Swap: 2048276k av,       0k used, 2048276k free                  117596k cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME CPU COMMAND
 4048 root      15   0 57676 8024  2052 S     0.5  0.3   4:15   0 X
 4057 gdm       15   0 17636  17M  6868 S     0.1  0.8   5:29   0 gdmgreeter
29384 imsl      15   0  1252 1252   944 R     0.1  0.0   0:00   1 top
    1 root      15   0   464  464   420 S     0.0  0.0   0:38   1 init
    2 root      RT   0     0    0     0 SW    0.0  0.0   0:00   0 migration/0
    3 root      RT   0     0    0     0 SW    0.0  0.0   0:00   1 migration/1
    4 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 keventd
    5 root      34  19     0    0     0 SWN   0.0  0.0   0:00   0 ksoftirqd_CPU
    6 root      34  19     0    0     0 SWN   0.0  0.0   0:00   1 ksoftirqd_CPU
   11 root      25   0     0    0     0 SW    0.0  0.0   0:00   0 bdflush
    7 root      15   0     0    0     0 SW    0.0  0.0   0:02   0 kswapd
    8 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 kscand/DMA
    9 root      15   0     0    0     0 SW    0.0  0.0   2:40   1 kscand/Normal
   10 root      15   0     0    0     0 SW    0.0  0.0   5:23   1 kscand/HighMe
   12 root      15   0     0    0     0 SW    0.0  0.0   0:39   0 kupdated
   13 root      25   0     0    0     0 SW    0.0  0.0   0:00   0 mdrecoveryd
   17 root      15   0     0    0     0 SW    0.0  0.0   0:45   1 kjournald
   75 root      25   0     0    0     0 SW    0.0  0.0   0:00   0 khubd
 2642 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 kjournald
 3381 root      25   0     0    0     0 SW    0.0  0.0   0:00   0 knodemgrd
```

pwd and ls

1. 目前所在目錄

2. 列出所有檔案和子目錄

```
[imsl@linux imsl]$
[imsl@linux imsl]$ pwd
/home/imsl
[imsl@linux imsl]$ ls -al
total 48
drwx------    4 imsl     imsl        4096 Jun 16 11:45 .
drwxr-xr-x   13 root     root        4096 Jun 16 11:32 ..
-rw-------    1 imsl     imsl        2899 Jun 16 11:54 .bash_history
-rw-r--r--    1 imsl     imsl          24 Aug  1  2007 .bash_logout
-rw-r--r--    1 imsl     imsl        1563 Aug  1  2007 .bash_profile
-rw-r--r--    1 imsl     imsl         124 Aug  1  2007 .bashrc
-rw-r--r--    1 imsl     imsl         847 Aug  1  2007 .emacs
-rwxrwxr-x    1 imsl     imsl          52 Aug  1  2007 .flexlmrc
-rw-r--r--    1 imsl     imsl         120 Aug  1  2007 .gtkrc
drwx------    2 imsl     imsl        4096 Aug  1  2007 .ssh
drwxr-xr-x    2 imsl     imsl        4096 Aug  1  2007 test
-rw-------    1 imsl     imsl         614 Jun 16 11:45 .viminfo
[imsl@linux imsl]$
```

3. 個人設定檔

4. 查詢 pwd 用法

```
[imsl@linux imsl]$ man pwd
PWD(1)                                  FSF                                  PWD(1)

NAME
       pwd - print name of current/working directory

SYNOPSIS
       pwd [OPTION]
```

mkdir

```
[imsl@linux imsl]$ mkdir course
[imsl@linux imsl]$ ls
course  test
[imsl@linux imsl]$ ls -al
total 52
drwx------    5 imsl     imsl         4096 Jun 16  2008 .
drwxr-xr-x   13 root     root         4096 Jun 16 11:32 ..
-rw-------    1 imsl     imsl         2899 Jun 16 11:54 .bash_history
-rw-r--r--    1 imsl     imsl           24 Aug  1  2007 .bash_logout
-rw-r--r--    1 imsl     imsl         1563 Aug  1  2007 .bash_profile
-rw-r--r--    1 imsl     imsl          124 Aug  1  2007 .bashrc
drwxrwxr-x    2 imsl     imsl         4096 Jun 16  2008 course
-rw-r--r--    1 imsl     imsl          847 Aug  1  2007 .emacs
-rwxrwxr-x    1 imsl     imsl           52 Aug  1  2007 .flexlmrc
-rw-r--r--    1 imsl     imsl          120 Aug  1  2007 .gtkrc
drwx------    2 imsl     imsl         4096 Aug  1  2007 .ssh
drwxr-xr-x    2 imsl     imsl         4096 Aug  1  2007 test
-rw-------    1 imsl     imsl          614 Jun 16 11:45 .viminfo
[imsl@linux imsl]$
```

1. 產生新的子目錄 course

檔案日期

cd

```
[imsl@linux imsl]$
[imsl@linux imsl]$ cd course/
[imsl@linux course]$ ls
[imsl@linux course]$ ls -al
total 8
drwxrwxr-x    2 imsl     imsl         4096 Jun 16  2008 .
drwx------    5 imsl     imsl         4096 Jun 16  2008 ..
[imsl@linux course]$
```

1. 進入子目錄 course

2. 目前所在目錄

3. 目錄 course 是空的

```
[imsl@linux course]$
[imsl@linux course]$ cd ..
[imsl@linux imsl]$
```

2. cd .. 離開目前目錄, 回到上層目錄

Compiler icpc, gcc

```
[imsl@linux imsl]$
[imsl@linux imsl]$ man icpc
ICC(1)                    Intel(R) C++ Compiler Options                    ICC(1)

NAME
       icc - invokes the Intel(R) C++ compiler

SYNOPSIS
       icc [  options ] file1 [ file2 ...]  where:

       options
              represents zero or more compiler options.

       filen  is  a  C/C++  source (.C .c .cc .cp .cpp .cxx .c++ .i), assembly
              (.s), object (.o), static library (.a), or other linkable  file.

       Note:  The  icpc command uses the same compiler options as the icc com-
       mand.  Invoking the compiler using icpc compiles .c, and  .i  files  as
       C++.   Invoking  the  compiler using icc compile
       Using icpc always links in C++ libraries.  Using
       libraries if C++ source is provided on the comma
```

```
[imsl@linux imsl]$ icpc -v
Version 10.0
```

版本編號

```
[imsl@linux imsl]$
[imsl@linux imsl]$ man gcc
GCC(1)                                GNU                                GCC(1)

NAME
       gcc - GNU project C and C++ compiler

SYNOPSIS
       gcc [-câ-Sâ-E] [-std=standard]
           [-g] [-pg] [-Olevel]
           [-Wwarn...] [-pedantic]
           [-Idir...] [-Ldir...]
           [-Dmacro[=defn]...] [-Umacro]
           [-foption...] [-mmachine-option...]
           [-o outfile] infile...

       Only the most useful options are listed here; see below for the remain-
       der.  g++ accepts mostly the same options as gcc.
```

```
[imsl@linux imsl]$ gcc -v
Reading specs from /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/specs
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/u
sr/share/info --enable-shared --enable-threads=posix --disable-checking --with-s
ystem-zlib --enable-__cxa_atexit --host=i386-redhat-linux
Thread model: posix
gcc version 3.2.2 20030222 (Red Hat Linux 3.2.2-5)
[imsl@linux imsl]$
```

版本編號

which

```
[imsl@linux imsl]$
[imsl@linux imsl]$
[imsl@linux imsl]$ which icpc
/opt/intel/cc/10.0.023/bin/icpc
[imsl@linux imsl]$
[imsl@linux imsl]$
[imsl@linux imsl]$ which gcc
/usr/bin/gcc
[imsl@linux imsl]$
```

1. Full path of command icpc

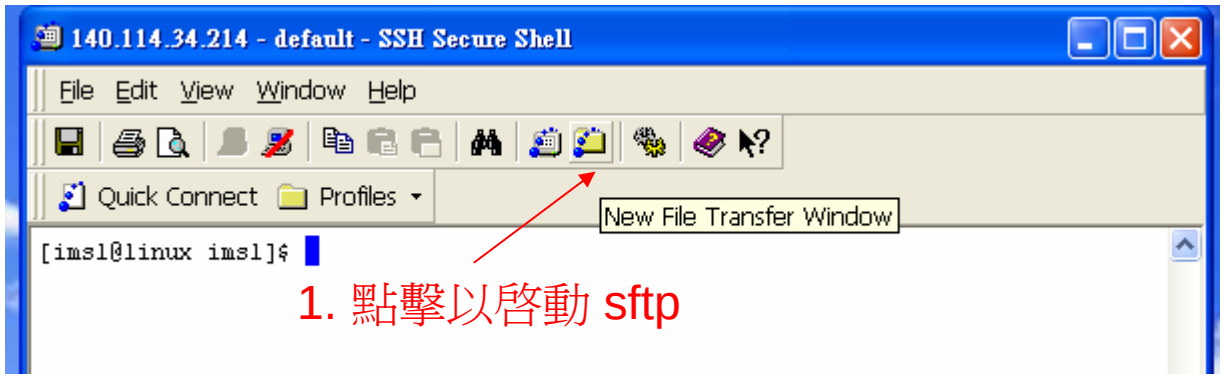
2. Full path of gcc

```
[imsl@linux imsl]$ cd /opt/intel/cc/10.0.023/bin/
[imsl@linux bin]$ ls
codecov   iccvars.csh   icpc.cfg   profdcg     profrun.bin   xiar
icc       iccvars.sh    map_opts   profmerge   pronto_tool   xild
iccbin    icpc          mcpcom     proforder   tselect
icc.cfg   icpcbin       prelink    profrun     uninstall.sh
[imsl@linux bin]$
```

3. 到 icpc 所在目錄

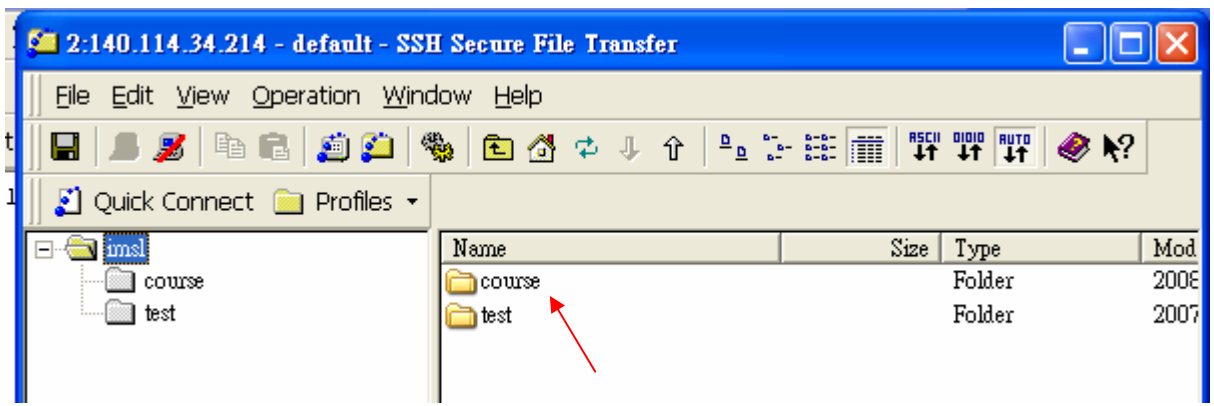
4. icpc 存在此目錄

使用 sftp 傳輸檔案 [1]



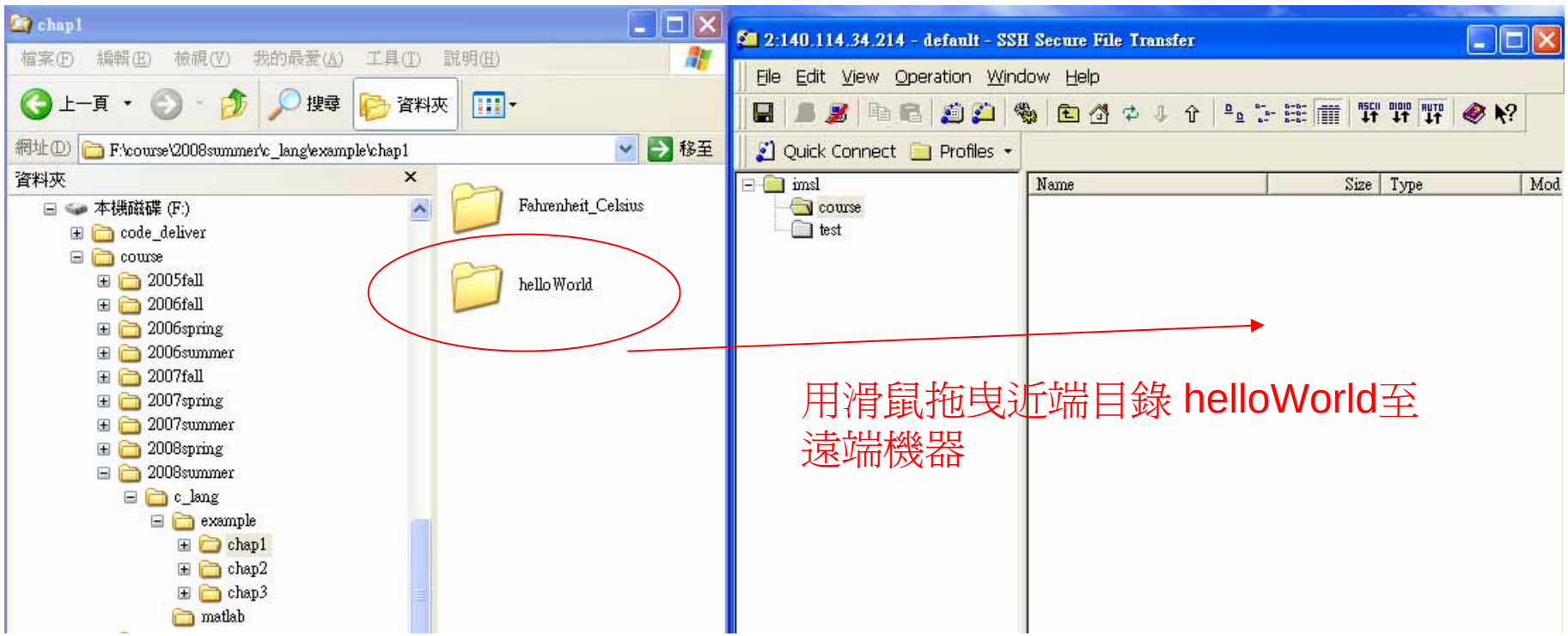
1. 點擊以啓動 sftp

sftp (secure ftp) window

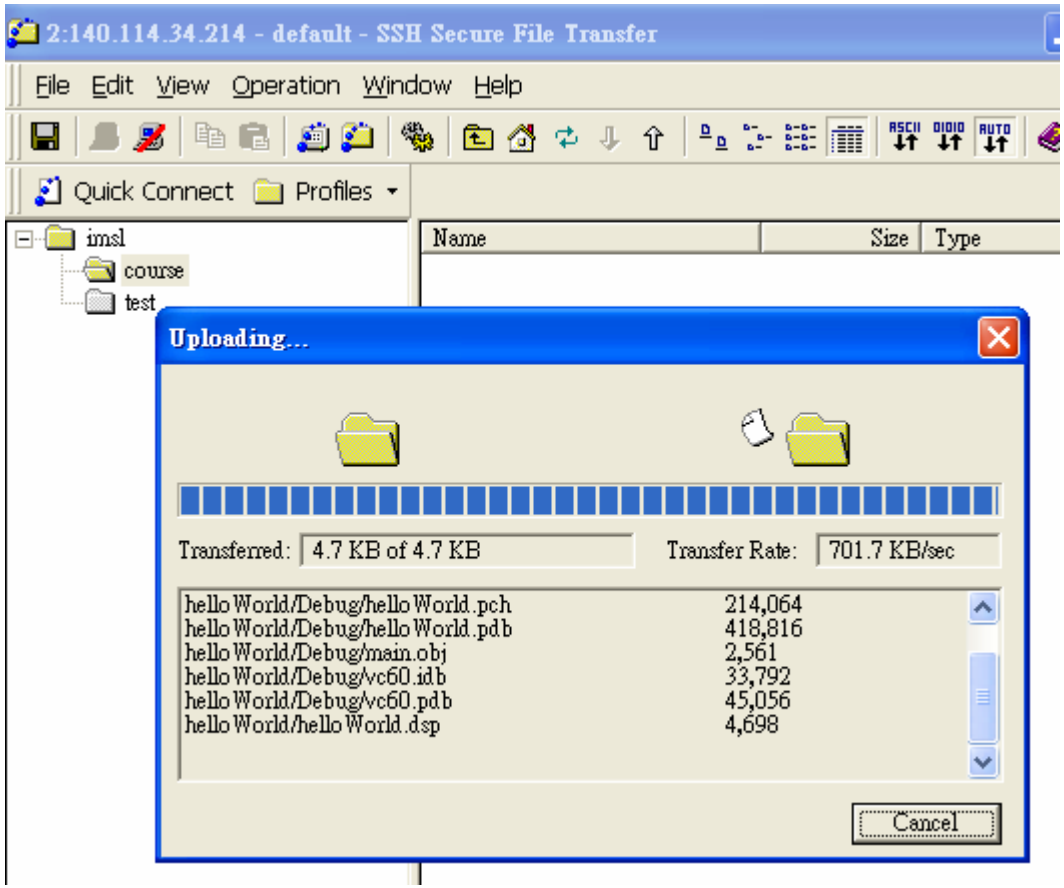


2. 點擊目錄 course並進入

使用 sftp 傳輸檔案 [2]



使用 sftp 傳輸檔案 [3]



編譯程式 [1]

```
[imsl@linux imsl]$ ls
course  test
[imsl@linux imsl]$ cd course/
[imsl@linux course]$ ls
helloWorld
[imsl@linux course]$ cd helloWorld/
[imsl@linux helloWorld]$ ls
Debug            helloWorld.dsw  helloWorld.opt  main.cpp
helloWorld.dsp   helloWorld.ncb  helloWorld.plg
[imsl@linux helloWorld]$ icpc main.cpp
[imsl@linux helloWorld]$ ls
a.out  helloWorld.dsp  helloWorld.ncb  helloWorld.plg
Debug  helloWorld.dsw  helloWorld.opt  main.cpp
[imsl@linux helloWorld]$ ./a.out
hello, world
[imsl@linux helloWorld]$
```

1. 進入目錄 course

2. 進入目錄 helloWorld

3. 編譯 main.cpp 產生執行檔 a.out

4. 執行 a.out

編譯程式 [2]

```
[imsl@linux helloWorld]$ ls
a.out  helloWorld.dsp  helloWorld.ncb  helloWorld.plg
Debug  helloWorld.dsw  helloWorld.opt  main.cpp
[imsl@linux helloWorld]$ rm a.out
[imsl@linux helloWorld]$ ls
Debug          helloWorld.dsw  helloWorld.opt  main.cpp
helloWorld.dsp  helloWorld.ncb  helloWorld.plg
[imsl@linux helloWorld]$ g++ main.cpp
[imsl@linux helloWorld]$ ls
a.out  helloWorld.dsp  helloWorld.ncb  helloWorld.plg
Debug  helloWorld.dsw  helloWorld.opt  main.cpp
[imsl@linux helloWorld]$ ./a.out
hello, world
[imsl@linux helloWorld]$
```

1. 移除 a.out

2. 用 g++ 編譯

3. 產生 a.out

4. 執行結果

passwd：換密碼

```
[imsl@linux imsl]$
[imsl@linux imsl]$ passwd
Changing password for user imsl.
Changing password for imsl
(current) UNIX password:
New password:
Retype new password:

The password has been changed on qdot.am.nthu.edu.tw.
passwd: all authentication tokens updated successfully.
[imsl@linux imsl]$
```

1. 輸入目前的密碼, 按entrer

2. 輸入新密碼, 按enter

3. 再輸入一次新密碼, 按enter