

微算機系統實習

第 13 組專案報告

LAB 01

組別： 13

班級、姓名與學號：

醫工三 B812110004 葉芸茜

醫工三 B812110011 湯青秀

日期：2024.03.12

Lab 00

一、實驗內容：

在 Linux 環境上，撰寫一個 Hello World 程式，並使用 g++ 進行編譯並執行，讓其可以運行在 Linux 系統上。

二、實驗過程及結果：

1. 實驗過程

(1) 在 Lab0 目錄下建立 helloworld.cpp

```
nvidia@ubuntu:~/Desktop$ cd Lab0
nvidia@ubuntu:~/Desktop/Lab0$ gedit helloworld.cpp
```

(2) 設計 .cpp 檔程式碼

```
helloworld.cpp
~/Desktop/Lab0

#include <iostream>

using namespace std;

int main(){
    cout << "Hello Ubuntu!!" << endl;
    return 0;
}
```

(3) 建立 MAKEFILE 檔

```
nvidia@ubuntu:~/Desktop/Lab0$ gedit Makefile
```

(4) 設計 MAKEFILE

```
helloworld:
# Make sure that you use a tab below
    g++ -o helloworld helloworld.cpp
```

(5) 執行 MAKEFILE。指令為 make

2. 實驗結果

使用指令直接編譯

```
nvidia@ubuntu:~/Desktop/Lab0$ g++ -o helloworld helloworld.cpp
nvidia@ubuntu:~/Desktop/Lab0$ ./helloworld
Hello Ubuntu!!
nvidia@ubuntu:~/Desktop/Lab0$
```

透過 MAKEFILE 編譯

```
nvidia@ubuntu:~/Desktop/Lab0$ gedit Makefile
nvidia@ubuntu:~/Desktop/Lab0$ make
make: 'helloworld' is up to date.
nvidia@ubuntu:~/Desktop/Lab0$ ls
helloworld helloworld.cpp Makefile
```

3. 遇到的問題&問題怎麼解決

引號打成中文的，使他編譯失敗

Lab 01

一、實驗內容：

1. 目標 1：使用 SSH 與其它機台進行連線
2. 目標 2：撰寫 Makefile 編譯程式
 - (1) Makefile 要有 target: all
 - (2) 編譯出跨平台可執行檔
 - (3) Makefile 要有 target: clean
 - (4) 刪除該執行檔與所有.o 檔
 - (5) 在 Makefile 中使用變數
 - (6) 在 Makefile 裡須加入 \$@, \$< or \$?
3. 目標 3：使用跨平台編譯開發嵌入式系統程式將虛擬機 CROSS_COMPILE 後的執行檔傳至 TX2，且以 ssh 連到 TX2 執行

二、實驗過程及結果：

1. 實驗過程

(1) 目標一

- (a) 使用 microUSB 接口連接電腦，並確認成功辨認裝置
- (b) 打開 README-usb-dev-mode.txt 來找尋 ip，或是以 arp -a 看 ip (對應板上的 MAC 編號)。附圖可找到 ip 為 192.168.137.212

```
C:\ 命令提示字元

網際網路網址      實體位址      類型
192.168.242.254    00-50-56-f8-5e-34    動態
192.168.242.255    ff-ff-ff-ff-ff-ff    靜態
224.0.0.22         01-00-5e-00-00-16    靜態
224.0.0.251        01-00-5e-00-00-fb    靜態
224.0.0.252        01-00-5e-00-00-fc    靜態
230.0.0.1          01-00-5e-00-00-01    靜態
239.255.255.250    01-00-5e-7f-ff-fa    靜態
255.255.255.255    ff-ff-ff-ff-ff-ff    靜態

介面: 192.168.137.1 --- 0xc
網際網路網址      實體位址      類型
192.168.137.212    00-04-4b-f8-35-84    靜態
192.168.137.255    ff-ff-ff-ff-ff-ff    靜態
224.0.0.22         01-00-5e-00-00-16    靜態
224.0.0.251        01-00-5e-00-00-fb    靜態
230.0.0.1          01-00-5e-00-00-01    靜態
239.255.255.250    01-00-5e-7f-ff-fa    靜態
255.255.255.255    ff-ff-ff-ff-ff-ff    靜態

C:\Users\LAB 1222>
```

(c)使用 SSH 連接 TX2 板與其他機台。指令為 ssh -Y 登入帳號@TX2 板子 ip，對應本機台及 ip 號碼可寫為 ssh -Y nvidia@192.168.137.212

```
Ubuntu 64-bit 1804 tx2 - VMware Workstation 16 Player (Non-commercial use only)
Player | [Icons] [Window] [Close]
Sun 23:30
nvidia@nvidia-desktop: ~
File Edit View Search Terminal Help
nvidia@ubuntu:~$ ssh -Y nvidia@192.168.137.212
The authenticity of host '192.168.137.212 (192.168.137.212)' can't be establish
ed.
ECDSA key fingerprint is SHA256:4PEmOmXk0hHDe5iNmKUHDfddhQC1hiLQ76f1zZp4Zuc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.137.212' (ECDSA) to the list of known hosts
.
nvidia@192.168.137.212's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.9.201-tegra aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

447 packages can be updated.
361 updates are security updates.

Last login: Mon Feb 26 17:09:00 2024 from 192.168.137.212
nvidia@nvidia-desktop:~$
```

(2)目標二

(a)在 MAKEFILE 中設計自動化變數及變數

- (i) CROSS_COMPILE = aarch64-linux-gnu-，因為 TX2 是使用 ARMv8 指令集
- (ii) CXX 是指 C++編譯器的名字
- (iii) CXXFLAGS 是 C++編譯器的選填，本次實驗使用 C++11 的 library
- (iv) \$@表示工作目標檔名
- (v) \$^表示所有被依賴文件的檔名，並以空格隔開這些檔名

(b)達成實驗要求

目標	達成
有 target: all	V
編譯出跨平台可執行檔	V
有 target: clean	V
刪除該執行檔與所有.o 檔	V
使用變數	V
加入 \$@ 、 \$< 或 \$?	V

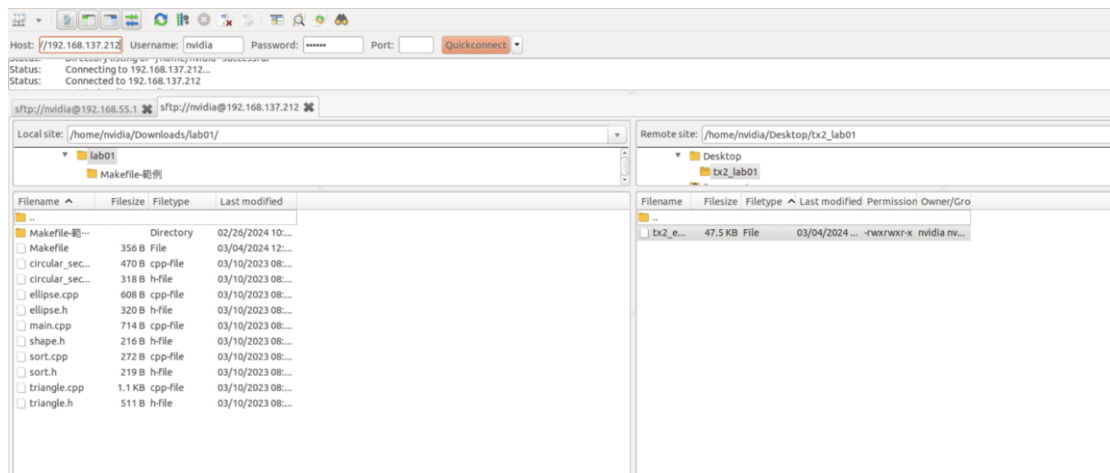
```
Makefile
CROSS_COMPILE=aarch64-linux-gnu-
CXX=$(CROSS_COMPILE)g++
LD=$(CROSS_COMPILE)ld
CXXFLAGS=-std=c++11

OBJ=ellipse.o sort.o triangle.o main.o circular_sector.o
SRC=ellipse.cpp sort.cpp triangle.cpp main.cpp circular_sector.cpp
EXE=tx2_exe
all:$(EXE)
$(EXE):$(OBJ)
    $(CXX) -o $@ $^
$(OBJ):$(SRC)
    $(CXX) $(CXXFLAGS) -c $^
clean:
    rm -f $(EXE)
    rm -f $(OBJ)
```

(3)目標三

- (a)在本機台目錄下執行 MAKEFILE，指令為 make，可使用 ls 指令確認是否成功編譯(成功會出現執行檔 tx2_exe 及其他.o 檔)

(b)將編譯的跨平台執行檔 tx2_exe 使用 filezilla 傳輸至 TX2



(c)在終端機(利用 SSH 遠端進 TX2 介面)輸入 ls，來查看目前目錄下的所有檔案，並確認是否有上個步驟傳輸過來的 tx2_exe(若有出現，則代表傳送成功)



(d)在 TX2 板執行該執行檔，指令為 ./tx2.exe

2. 預期實驗結果

(1)目標一

成功連接 tx2 後，terminal 中 nvidia@ 的後綴從 ubuntu 轉變為 nvidia-desktop

(2)目標二

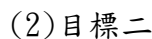
(a)進入檔案存放的資料夾後輸入 make 即會出現編譯命令。編譯成功後再輸入 ls 即可看到 tx2_exe 檔案成功建置。

(b)輸入 make clean 後即會出現 rm -f 命令。刪除成功後再輸入 ls 即可看到資料夾中 exe 及 .o 檔案都被成功清除。

(3)目標三

透過 filezilla 將 exe 檔案傳輸到 tx2 的目標資料夾後，使用成功連接到板子的終端，進入該資料夾並輸入 ./tx2_exe 後，即可看到程式中的輸出成功顯示。

(1)目標一

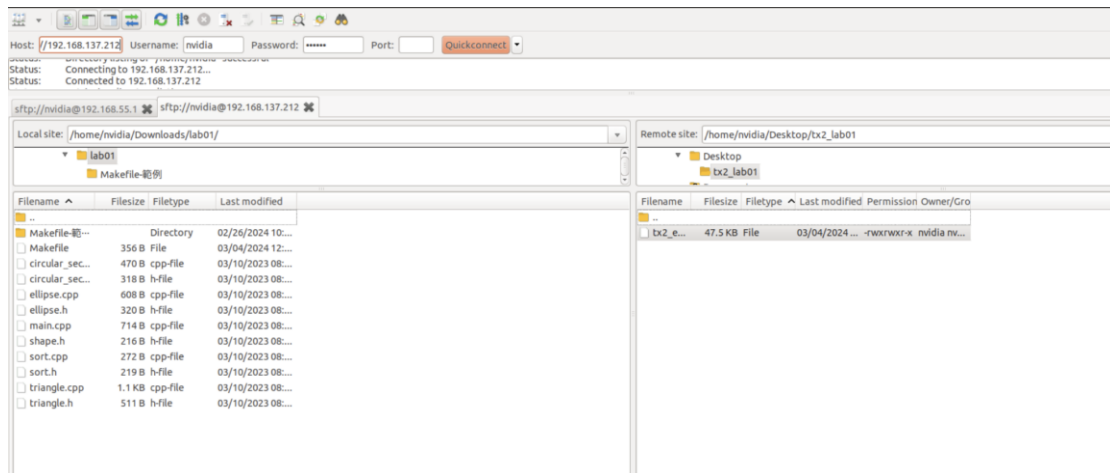


```
makefile clear
```

```
nvidia@ubuntu:~/Downloads/lab01$ ls
circular_sector.cpp  circular_sector.h  ellipse.cpp  ellipse.h  main.cpp  Makefile-範例  shape.h  sort.cpp  sort.h  triangle.cpp  triangle.h
nvidia@ubuntu:~/Downloads/lab01$ make
aarch64-linux-gnu-g++ -std=c++11 -c ellipse.cpp sort.cpp triangle.cpp main.cpp circular_sector.cpp
aarch64-linux-gnu-g++ -o tx2_exe ellipse.o sort.o triangle.o main.o circular_sector.o
nvidia@ubuntu:~/Downloads/lab01$ ls
circular_sector.cpp  circular_sector.o  ellipse.h  main.cpp  Makefile  shape.h  sort.h  triangle.cpp  triangle.o
circular_sector.h  ellipse.cpp  ellipse.o  main.o  Makefile-範例  sort.cpp  sort.h  triangle.h  tx2_exe
nvidia@ubuntu:~/Downloads/lab01$ make clean
rm -f tx2_exe
rm -f ellipse.o sort.o triangle.o main.o circular_sector.o
nvidia@ubuntu:~/Downloads/lab01$ ls
circular_sector.cpp  circular_sector.h  ellipse.cpp  ellipse.h  main.cpp  Makefile  Makefile-範例  shape.h  sort.cpp  sort.h  triangle.cpp  triangle.h
nvidia@ubuntu:~/Downloads/lab01$
```

(3)目標三

filezilla



執行結果

```
nvidia@nvidia-desktop:~/Desktop$ cd tx2_lab01
nvidia@nvidia-desktop:~/Desktop/tx2_lab01$ ls
tx2_exe
nvidia@nvidia-desktop:~/Desktop/tx2_lab01$ ./tx2_exe
原本的: 62.8319 78.5398 52.3599 50
排序後: 50 52.3599 62.8319 78.5398
作業一完成
nvidia@nvidia-desktop:~/Desktop/tx2_lab01$
```

4. 遇到的問題&問題怎麼解決

- (1)網路接口接不到板子，後來發現是板子本身沒辦法接到(雖然助教測試說沒問題，但是microUSB也接不到，可能要用自己筆電測試看看，但後來拿到了另一個板子所以該結論未知)，後來換一個板子就都很順利連接到了
- (2)make的問題，一直說沒辦法編譯，後來發現是虛擬機裡沒有安裝aarch64-linux-gnu-g++，用sudo apt-get install g++-aarch64-linux-gnu指令就可以成功安裝了
- (3)makefile設計很爛，一直沒辦法編譯(把一堆檔案丟給變數那裏)一直讀不到，後來把後面程式碼改寫成\$^就可以順利跑了
- (4)忘記實驗目標要將執行檔丟到跨平台執行，所以我們一直在本機台執行，後來有傳送到TX2的環境執行
- (5)寫了make clean卻不會用，後來才知道要在終端執行指令，在終端輸入指令make clean就可以順利刪除所有.o檔及.exe檔

三、程式碼

目標 1: 使用 SSH 與其它機台進行連線
ssh -Y nvidia@192.168.137.212

目標 2: 撰寫 Makefile 編譯程式
CROSS_COMPILE=aarch64-linux-gnu- CXX=\$(CROSS_COMPILE)g++ LD=\$(CROSS_COMPILE)ld CXXFLAGS=-std=c++11 OBJ=ellipse.o sort.o triangle.o main.o circular_sector.o SRC=ellipse.cpp sort.cpp triangle.cpp main.cpp circular_sector.cpp EXE=tx2_exe all:\$(EXE) \$(EXE):\$(OBJ) \$(CXX) -o \$@ \$^ \$(OBJ):\$(SRC) \$(CXX) \$(CXXFLAGS) -c \$^ clean: rm -f \$(EXE) rm -f \$(OBJ)

目標 3: 使用跨平台編譯開發嵌入式系統程式
./tx2_exe

四、本次實驗過程說明與解決方法:

本次實驗逐步依據 1. 實驗過程的步驟執行，過程中遇到的問題可參照 4. 遇到的問題及解決方法。

五、分工:

學號、組員	貢獻比例	工作內容
B812110004 葉芸茜	50%	文書處理、實驗設計與實作、程式規劃、測試與除錯
B812110011 湯青秀	50%	文書處理、實驗設計與實作、程式規劃、測試與除錯