



微算機系統實習

MICROPROCESSOR SYSTEMS LAB.

SPRING, 2019

INSTRUCTOR : YEN-LIN CHEN(陳彥霖), PH.D.
PROFESSOR
DEPT. COMPUTER SCIENCE AND INFORMATION
ENGINEERING
NATIONAL TAIPEI UNIVERSITY OF TECHNOLOGY

作業評分方式

- 實驗部分佔該次實驗的總比例70%
報告總分佔該次實驗的總比例30%
* 當次實驗會因為難易度不同佔學期總成績的實驗分數比例也不同
- 實驗報告上傳格式，公告於社團
請同學上傳報告時依照上面的格式上傳，
 - 一、組別與組員名單
 - 二、實驗步驟截圖與說明
 - 三、組員貢獻比例 (組員%數加總必須等於100%)
 - 四、心得 (最少300字)
 - 實驗報告單獨分數為100分，以上第一、三、四點每缺少一項報告分數扣20分。第二點缺少報告扣40分。另外第二、四點不完整會依照狀況扣分，最高扣到該項目的上限分數。
 - 貢獻比例分配方式為: 報告總分*2*組員貢獻比例=組員報告得分

作業繳交

- 基本繳交時間

- 實驗：公布實驗後的隔週(3/4)上課結束前(18:00)

- 報告：公布實驗後的隔兩週(3/11)上課前(15:00)

- * 若有因為特殊原因繳交時間有變動助教會另外公布
超過時間遲交每隔一週（含一週內）分數打8折，採累計連乘方式，
實驗與報告打折是分開算的

- 舉例：

- 遲交三天 - 以遲交一週計算 $\text{<遲交的項目單獨分數>} * 0.8 = \text{該項目得到的分數}$

- 遲交九天 - 以遲交兩週計算 $\text{<遲交的項目單獨分數>} * 0.8 * 0.8 = \text{該項目得到的分數}$

- 以上配分與注意事項有問題請聯絡助教

說明

- 請依組別與助教領取TK1
 - 領取TK1時，須登記MAC
 - 該台TK1即為本學期組員共用機台
 - 每次實作機台為該組固定使用
 - 請好好愛惜機台
- 若板子有任何問題
 - 請通知助教
 - 不可自行燒錄TK1
 - 作業系統及Kernel

本次實驗目標

- 學習如何使用ssh與其它機台進行連線
- 學習使用跨平台編譯工具發展嵌入式系統程式
- 學習如何撰寫Makefile編譯程式

A decorative graphic on the left side of the slide, consisting of a network of white lines and circles on a dark blue background, resembling a circuit board or a neural network. The lines are vertical and horizontal, with some diagonal connections, and the circles are of varying sizes, some acting as nodes or endpoints.

跨平台連線

實驗部分

- JetPack安裝完畢以後請使用VM上的Ubuntu系統連到TK1
- 在VM上使用ssh的指令連到TK1
 - ssh -X 登入帳號@TK1板子IP
 - 範例使用192.168.1.3，請同學輸入自己TK1所分配的ip

```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ ssh -X ubuntu@192.168.1.3  
ubuntu@192.168.1.3's password:  
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.10.40-ga7da876 armv7l)  
  
* Documentation:  https://help.ubuntu.com/  
  
New release '16.04.2 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Wed Mar  1 15:08:57 2017 from 192.168.1.2  
ubuntu@tegra-ubuntu:~$
```




實驗零

LINUX程式設計-在UNUNTU下編 譯程式

LINUX系統程式設計

- 撰寫一個Hello World的程式，讓其可以運行在Linux系統上。

LINUX系統程式設計

- 建立helloworld.cpp
- 設計一個程式可以在Ubuntu顯示”Hello Ubuntu!!”。
- 先在任意一個地方創建資料夾(mkdir)。然後進入此資料夾。
- 用指令gedit helloworld.cpp 建立helloworld.cpp檔，並輸入以下程式碼：

```
#include <iostream>
using namespace std;
int main()
{
    cout << “Hello Ubuntu!!” << end;
    return 0;
}
```

LINUX系統程式設計

```
#include <iostream>
using namespace std;

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

在UNUNTU下編譯程式

- 在Terminal輸入以下指令
 - `g++ -o helloworld helloworld.cpp`
- 如果沒有g++編譯器

```
ubuntu@tegra-ubuntu: ~  
ubuntu@tegra-ubuntu:~$ g++ -v  
-bash: g++: command not found  
ubuntu@tegra-ubuntu:~$
```

- 輸入`sudo apt-get install build-essential -y` 安裝所需編譯器
- 產生名為helloworld的執行檔。
 - 可以使用`file`指令查看執行檔的屬性
`<ex> file helloworld`

在UBUNTU內執行程式

- 用指令執行此helloworld。
 - ./helloworld
- 執行的結果是會出現**Hello Ubuntu!!**。

```
[ubuntu@tegra-ubuntu:~/testhello$ ls
helloworld  helloworld.cpp  Makefile
[ubuntu@tegra-ubuntu:~/testhello$ ./helloworld
Hello Ubuntu!!
```



運用MAKEFILE編譯程式 專案

MAKEFILE檔建立

- 用指令 `gedit Makefile` 建立 Makefile 檔，並輸入以下文字：

- **helloworld:**

- # Make sure that you use a tab below

- `g++ -o helloworld helloworld.cpp`

- #開頭的為註解。

MAKEFILE檔建立

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

運用MAKEFILE編譯程式專案

- 對helloworld.cpp 作編譯的動作。使用的是make指令。
 - 首先確定在工作目錄裡有做好的helloworld.cpp和Makefile檔
 - 接著在Terminal用cd指令移動到工作目錄下。
 - 輸入make，便藉由Makefile開始編譯 helloworld.cpp。
 - 注意執行make時，必須有所在目錄或輸出目錄的使用權限
 - 最後產生名為helloworld的執行檔。
 - 可以使用file指令查看執行檔的屬性
- <ex> file helloworld

使用MAKEFILE執行編譯

```
[ubuntu@tegra-ubuntu:~/testhello$ make  
g++ -o helloworld helloworld.cpp  
[ubuntu@tegra-ubuntu:~/testhello$ ls  
helloworld helloworld.cpp Makefile  
[ubuntu@tegra-ubuntu:~/testhello$ file helloworld  
helloworld: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically  
linked (uses shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=938e8a8429e8036ce9  
d6ee3f13c1166c1b64cd21, not stripped
```



實驗一 跨平台嵌入式程式開發

實驗一程式要求

- 撰寫一支排序程式。
 - 以使用者自行輸入10個數字。
 - 至少要有main.cpp 與 sort.h & .cpp甚至更多。

```
ubuntu@tegra-ubuntu: ~/0223
ubuntu@tegra-ubuntu:~/0223$ ls
Makefile  main.cpp  main.o  sort.cpp  sort.h  sort.h~
Makefile~ main.cpp~ out      sort.cpp~ sort.h.gch  sort.o
ubuntu@tegra-ubuntu:~/0223$
```

- 要顯示排序前資料以及排序後資料。
- 注意事項
 - 使用快速排序法排序(Quick Sort)。

實驗一MAKEFILE要求

- 基本要求
 - 要有target:all
 - 編譯出可執行檔
 - target:clean
 - 刪除該執行檔與所有.o檔
 - 需使用\$()符號(變數)實作MakeFile。
- 進階要求(達成分數第二項才會滿分)
 - 在 Makefile 裡須加入 \$@ or \$< 。

實驗一預期執行結果

```
ubuntu@tegra-ubuntu: ~/0223
ubuntu@tegra-ubuntu:~/0223$ make run
g++ main.cpp -o out
./out
4
2
7
89
50
47
37
14
28
75
after sort
2
4
7
14
28
37
47
50
75
89
ubuntu@tegra-ubuntu:~/0223$
```


範例

- 輸入 `gedit helloworld.cpp` 產生檔案(名稱可變 Ex : main.cpp)

```
ubuntu@ubuntu: ~/program
ubuntu@ubuntu:~/program$ gedit helloworld.cpp
```

- 輸入印出Hello TK1 程式碼

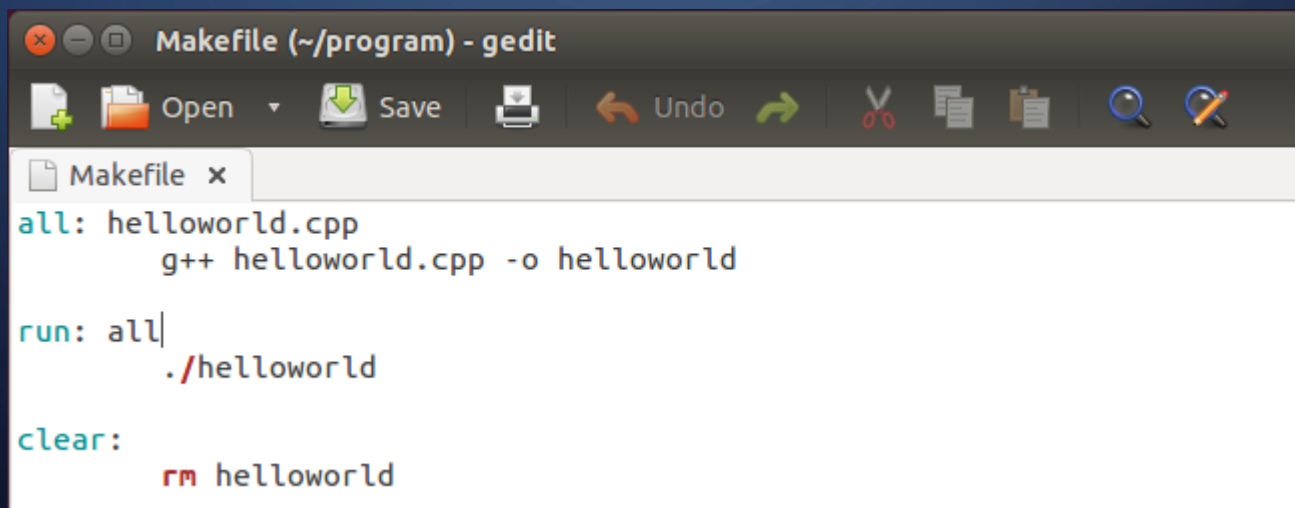
```
helloworld.cpp (~/program) - gedit
Open Save Undo
helloworld.cpp x
#include <iostream>
using namespace std;
int main(int argc, char** argv) {
    cout << "Hello TK1" << endl;
    return 0;
}
```

範例

- 輸入`gedit Makefile`產生檔案

```
ubuntu@ubuntu: ~/program
ubuntu@ubuntu:~/program$ gedit Makefile
```

- 輸入印出`Makefile`內容



The screenshot shows a gedit window titled "Makefile (~/.program) - gedit". The window has a menu bar with "Open", "Save", "Undo", and other standard editing icons. The main text area contains the following Makefile content:

```
all: helloworld.cpp
    g++ helloworld.cpp -o helloworld

run: all|
    ./helloworld

clear:
    rm helloworld
```

範例

- 輸入make run觀看執行結果

```
ubuntu@ubuntu: ~/program
ubuntu@ubuntu:~/program$ make run
g++ helloworld.cpp -o helloworld
./helloworld
Hello TK1
ubuntu@ubuntu:~/program$
```