

# 系統程式-期末作業

筆記

資工二 110710547 翁瑋泓

## 內容

前言.....	3
快捷鍵.....	4
第一週.....	5
上課內容: .....	5
補充: .....	11
問題.....	12
第二週-簡易編譯器 .....	15
上課內容: .....	15
問題.....	16
第三週-編譯器 .....	17
上課內容: .....	17
第四週- assembler 組譯器 .....	20
上課內容: .....	20
問題.....	23
第五週-休息 .....	27
第六週-虛擬機 .....	28
上課內容: .....	28
問題.....	30
第七週- C 語言轉組合語言 .....	32
上課內容: .....	32
補充.....	34
問題.....	35
第八週-期中作業+系統程式理論說明 .....	37
上課內容: .....	37
期中作業.....	38
補充.....	38
問題.....	38
第九週-目的檔、二進位工具、qemu 虛擬機 .....	39
上課內容: .....	39
補充.....	44
問題.....	46
第十週- mingw, msys2、, process 與 thread、thread 競爭情況與死結 .....	48
上課內容: .....	48
補充.....	55
問題.....	55

第十一週-改用 Google Meet .....	57
第十二週-mingw, msys2、, process 與 thread、thread 競爭情況與死結 .....	58
上課內容: .....	58
補充.....	59
問題.....	60
第十三週-msys2, glib2、msys2, PostgreSQL .....	62
上課內容: .....	62
補充.....	65
問題.....	65
第十四週-msys2, glib2、msys2, PostgreSQL .....	69
上課內容: .....	69
補充.....	71
問題.....	72
第十五週-程式與報告均要將原創與引用清除區隔、組合語言 (Linux)、網路 socket 程式 .....	73
上課內容: .....	73
補充.....	83
問題.....	85
第十六週-作業系統 (理論部分)、嵌入式系統 (slide)、作業系統與嵌入式範例 .....	86
上課內容: .....	86
補充.....	87
第十七週- Http + Crawler、Crawler、mmap、rust 程式語言 .....	89
上課內容: .....	89
補充.....	92

## 前言

雖然直播都有看，大部分也有跟著操作但不懂的地方還是很多，也有很多內容聽了但還是不夠了解，我非常清楚是我基礎不夠高，不過我相信多理解這門課對我以後的幫助會非常大，所以我會繼續努力學習，也感謝老師上課補充的很完整，還經常會回答同學的問題。雖然前幾周的進度還算看得懂但後面比較深入的問題就比較不懂了，所以筆記會比較多是同學有提問的跟老師有回答過的。

## 快捷鍵:

Ctrl+s = 儲存

上傳作業指令:

git add -A

git commit -m “檔名” (or 檔名的一部份+tab 鍵)

git Push origin master

master -> master => 成功

拷貝:git Pull origin master 如果不行就先 git stash · 成功就有結果

# 第一週

## 上課內容:

解說:1.表格大小假設是 5，餘數應該是 0-4 之間的值。

解說:2.如果雜湊的格子很多，那麼碰撞再一起的機會就會比較少，那如果說雜湊格子很少，那就很容易碰撞再一起。

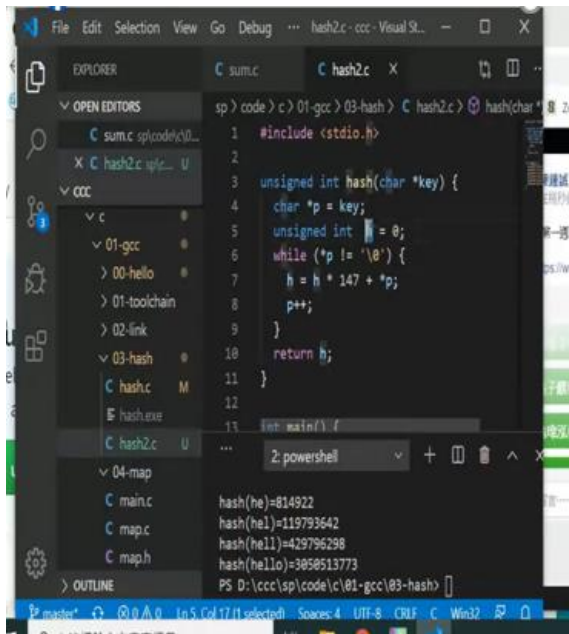
解說:3.表格大小為 5 的表格，如果要放入 6 筆資料，肯定放不下，兩個辦法，一個是連結串列，另外一個方法是你的表格永遠都要夠大，不夠大的時候，就會再從新打構，再擴充表格，為原本兩倍，再從新雜湊一次，這麼一來，永遠就夠。

```
sp > code > c > 01-gcc > 03-hash > C hash.c > hash(char *)
1  #include <stdio.h>
2
3  unsigned int hash(char *key) {
4      char *p = key;
5      unsigned int h = 37;
6      while (*p != '\0') {
7          h = h * 147 + *p;
8          p++;
9      }
10     return h;
11 }
12
13 int main() {
```

解說:1 這邊的 p++ 的目的是進到下一個字母，讓指標進到下一個位置。

解說:2 下一個字母還不是 0，所以還會繼續執行。

解說:3 c 語言的字串都是以\0 為結尾。

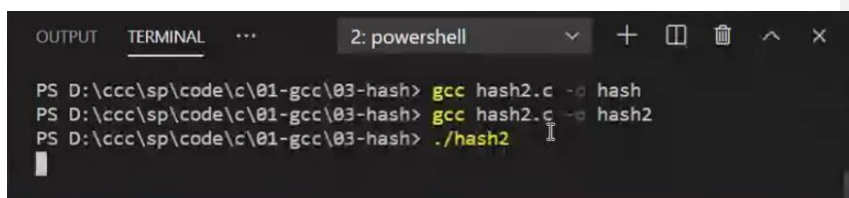


解說:1 首先先把雜湊值設為 0，把  $h = h * 147 + *p$  改為  $h += *p$ ，變得更簡單。

主程式就不用改，一樣在編譯一次。

解說:2 單純把每一個字元加起來，也算是雜湊，只不過不是一個非常好的雜湊程式。

解說:3 執行指令操作



解說:4 一開始的 hash 會一直變大，現在這個改編版本，感覺大小不會差很多，雖然還是持續變大，因為字越多，加起來的總值也會越多，只不過因為是單純的加總起來，因此範圍就會縮得比較小。

解說:5 執行結果

```
hash()=37
hash(h)=5543
hash(hell)=429796298
PS D:\ccc\sp\code\c\01-gcc\03-hash> gcc hash2.c -o hash
PS D:\ccc\sp\code\c\01-gcc\03-hash> gcc hash2.c -o hash2
PS D:\ccc\sp\code\c\01-gcc\03-hash> ./hash2
hash()=0
hash(h)=104
hash(he)=205
hash(hel)=313
hash(hell)=421
```

解說:6 雜湊函數，只要有一定的規則，就叫做雜湊，不一定要先\*再+，同樣的數字進去，同樣的數字出來，就叫雜湊。

```
1 #include "map.h"
2
3 Pair jList[] = {
4     {"ee", "000"}, {"JTr", "001"}, {"JEq", "010"}, {"JGE", "01"},
5     {"JLT", "100"}, {"JNE", "101"}, {"JLE", "110"}, {"JMP", "11"},
6 };
7
8 Map jMap;
9
10 int main() {
11     mapNew(&jMap, 17);
12     jMap.table = jList;
13     jMap.top = 8;
14     char *jCode = mapLookup(&jMap, "JLE");
15 }
```

解說:1 這一個陣列組合是 內土天舉的陣列組合有這 8 種



解說:2 INE 就是不相等 \JLE 就是小於等於 \JMP 不管怎麼樣都要跳

{ " ", " 000" } 不做跳躍動作

解說:3 mapNew(&jMap,17); 意思是說我要建立起一個大小為 17 的表格

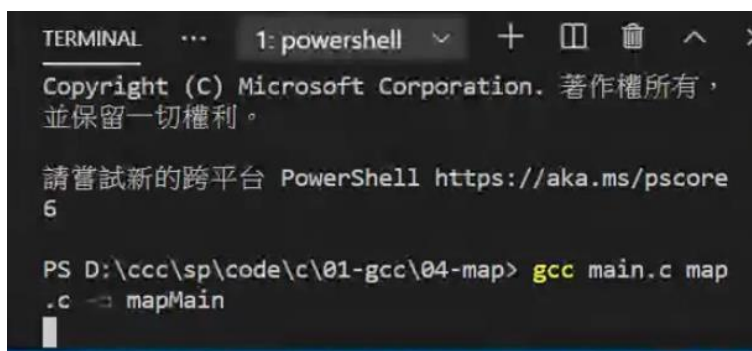
然後這個表格變數名稱叫 jMap;

```
8   Map jMap;
9
10  int main() {
11      mapNew(&jMap, 17);
12      jMap.table = jList;
13      jMap.top = 8;
14      char *jCode = mapLookup(&jMap, "JLE");
15      printf("jCode=%s\n", jCode);
16  }
17
18
```

解說:4 jMap.top = 8; 因為有 8 個元素 · 這裡 top 最高 8

解說:5 在 mapLookup · 等一下會搜尋( "JLE" ) 這個結構

解說:6 執行指令



```
TERMINAL ... 1: powershell
Copyright (C) Microsoft Corporation. 著作權所有，
並保留一切權利。

請嘗試新的跨平台 PowerShell https://aka.ms/pscore
6

PS D:\ccc\sp\code\c\01-gcc\04-map> gcc main.c map
.c -o mapMain
```

```
main.c  x  map.c
sp > code > c > 01-gcc > 04-map > C main.c > main()
11  mapNew(&jMap, 17);
12  jMap.table = jList;
13  jMap.top = 8;
14  char *jCode = mapLookup(&jMap, "JLE");
15  printf("jCode=%s\n", jCode);
16  }
17
18

TERMINAL  ...  1: powershell  +  -  ^
請嘗試新的跨平台 PowerShell https://aka.ms/pscore6
6
PS D:\ccc\sp\code\c\01-gcc\04-map> gcc main.c map
.c -o mapMain
PS D:\ccc\sp\code\c\01-gcc\04-map> ./mapMain
jCode=110
PS D:\ccc\sp\code\c\01-gcc\04-map> 
```

```
File Edit Selection View Go Debug ... map.c - ccc - Visual St...
main.c  x  map.c
sp > code > c > 01-gcc > 04-map > C map.c > mapNew(Map *, int)
1  #include "map.h"
2
3  Map* mapNew(Map *map, int size) {
4  map->table = NULL;
5  map->size = size;
6  map->top = 0;
7  return map;
8  }
9
10 Pair mapAdd(Map *map, char *key, void *value) {

OUTPUT  TERMINAL  ...  1: powershell  +  -  ^  x
請嘗試新的跨平台 PowerShell https://aka.ms/pscore6
PS D:\ccc\sp\code\c\01-gcc\04-map> gcc main.c map.c -o mapMain
PS D:\ccc\sp\code\c\01-gcc\04-map> ./mapMain
jCode=110
PS D:\ccc\sp\code\c\01-gcc\04-map> 
```

解說:7 一開始就只有設定大小 為 17 個，如果超過會爆掉。

解說:8 mapFind

```
17 ∨ int mapFind(Map *map, char *key) {  
18 ∨     for (int i=0; i<map->top; i++) {  
19 ∨         if (strcmp(map->table[i].key, key)==0)  
20             return i;  
21     }  
22     return -1;  
23 }  
24
```

解說:9 在這個 map 陣列裡面，一個一個找，從 0 開始，一直找到最後一個  
，然後一直去比較 key，看是不是我們要找的 key。

## 補充:

[資料結構] 雜湊 (Hash)

[https://ithelp.ithome.com.tw/articles/10208884?fbclid=IwAR2-4D2RLERqsY8\\_bLNb0cE9UEncDRkVhPh6WYBusYoc0OfZ9EJ4RuAGMko](https://ithelp.ithome.com.tw/articles/10208884?fbclid=IwAR2-4D2RLERqsY8_bLNb0cE9UEncDRkVhPh6WYBusYoc0OfZ9EJ4RuAGMko)  
<https://www.eecs.wsu.edu/~ananth/CptS223/Lectures/hashing.pdf?fbclid=IwAR2ncfTBMw1KM-sihZ6Uu5Xm5HhORijD4-K5n28pD3K3NySAH2EjZA6OiqU>

資料結構課程我建議應該實作三種資料結構

1. 連結串列
2. 二元樹
3. 雜湊表

<http://misavo.com/blog/%E9%99%B3%E9%8D%BE%E8%AA%A0/%E6%9B%B8%E7%B1%8D/C%E8%AA%9E%E8%A8%80/%E9%AB%98%E7%AD%89/structure>

以上是我網站的教材，請搭配你們的 C 語言課本一起看！

# 問題

## 同學問題:

不懂箭頭(->)跟 Pair 跟 Map

## 老師回答:

箭頭是用來《存取指標指向的結構內之欄位》

例如

```
typedef struct _Map {  
  
    Pair *table;  
  
    int size;  
  
    int top;  
  
} Map;  
  
Map* mapNew(Map *map, int size) {  
  
    map->table = NULL;  
  
    map->size = size;  
  
    map->top = 0;  
  
    return map;  
  
}
```

其中 map 是個結構指標，箭頭就可以用來存取其中的欄位 size, table,

top

### 同學問題:

老師，struct 是甚麼？

### 老師回答:

struct 是 structure 的短寫，也就是結構

struct 裡面可以有很多欄位，每個欄位都有名字和型態。

```
typedef struct _Pair {  
  
    char *key;  
  
    void *value;  
  
} Pair;  
  
typedef struct _Map {  
  
    Pair *table;  
  
    int size;  
  
    int top;  
  
} Map;
```

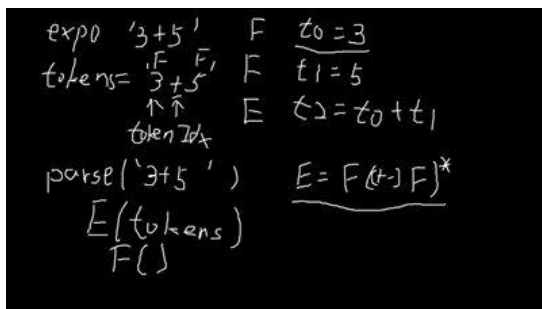
```
Pair jList[] = {
```

```
{ "", "000"}, {"JGT", "001"}, {"JEQ", "010"}, {"JGE", "011"},  
  
{"JLT", "100"}, {"JNE", "101"}, {"JLE", "110"}, {"JMP", "111"}  
  
};  
  
Map jMap;
```

## 第二週-簡易編譯器

### 上課內容:

exp0 '3+5' 的過程



#### 過程解說

```
parse('3+5') => tokens = "3+5", tokenIdx = 0
E()
  i1=F()      => i1=0
  3           => t0=3
  while () ...
    +         => op = +
    i2=F()    => i2=1, t1=5
    i=nextTemp() => i=2
    print()   => t2=t0+t1
```

#### 以 3+5 為例

```
...
// E = F ([+-] F)*
int E() {
  int i1 = F();           // t0=3, i1=0
  while (isNext("+")) {
    char op=next();       // op = +
    int i2 = F();         // t1=5, i2=1
    int i = nextTemp();   // i = 2
    printf("t%d=t%d%ct%d\n", i, i1, op, i2);
    // t2 = t0+t1
    i1 = i;
  }
  return i1;
}
...
```



# 問題

## 同學問題:

int F()中的 else if (c=='(') '(' 是哪裡來的

## 老師回答:

這是在處理有括號的情況，像是 `exp0 '3+(5-2)'`  
遇到 ( 後會進入處理 (5-2) 的程序

## 同學問題:

為甚麼@x 是 D=M?

## 老師回答:

因為 @x 是指定位 x 變數，這個變數在記憶體裡面。  
例如 x 的記憶體位址在 100，但是其內容是 3

@x

D=M

這樣就會把記憶體位址 100 內的 3 提取出來丟給 D，也就是  
@100

D=M

不能用 D=A, 因為這樣只會把 100 存入 D

## 同學問題:

```
int nextTemp() {  
    static int templdx = 0;  
    return templdx++;  
}
```

老師這個是先傳回值再加 1 嗎

## 老師回答:

對，先傳回值 templdx，然後在把 templdx 加 1

注意 `static int templdx = 0;` 只在程式一開始會執行一次，因為是 `static`.  
所以

第一次呼叫 `nextTemp()` 會傳回 0,

第二次會傳回 1,

以此類推

## 第三週-編譯器

### 上課內容:

越下層的運算優先序越高，越上層的運算優先序越低。

不管傳入的數字下一個字元是+\*/都會被先送到 TQ 檢查,如果是\*/就會被先 print。

exp1 '3+5\*8'

```
$ ./exp1 '3+5*8'
=== EBNF Grammar =====
E=T ([+-] T)*
T=F ([*/] F)*
F=Number | Id | '(' E ')'
```

E	=	3+5*8
		T+ T
		F*F
		5 8

exp1 : E=T ([+-] T)\* 比對

```
//      3   +   5*8
// E = T ([+-] T)*
int E() {
    int i1 = T();      // 3
    while (isNext("+-")) {
        char op=next(); // +
        int i2 = T();   // (5*8)
        int i = nextTemp();
        printf("t%d=t%d%ct%d\n", i, i1, op, i2);
        i1 = i;
    }
    return i1;
}
```

exp1: T = F ([\*/] F)\* 比對

```
//      5   *   8
// T = F ([*/] F)*
int T() {
    int f1 = F();      // 5
    while (isNext("*/")) {
        char op=next(); // *
        int f2 = F();   // 8
        int f = nextTemp();
        printf("t%d=t%d%ct%d\n", f, f1, op, f2);
        f1 = f;
    }
    return f1;
}
```

int len = p-start;//掃到的字串長度

char \*token = strTableEnd;

strncpy(strTableEnd, start, len);//strTableEnd 永遠指向 strTable 的結尾

//從起始字串開始複製一個 len 長度的字串到 strTableEnd

strTableEnd[len] = '\0';//把最後的字元設成'\0'

strTableEnd += (len+1);//把 str 指標往後移到剛剛的'\0'後面

types[tokenTop] = type;//設定好此字串之型態

```
tokens[tokenTop++] = token;
printf("token=%s\n", token); //印出詞彙
return p;
```

編譯器:

<https://github.com/ccccourse/sp/tree/master/code/c/02-compiler/03-compiler?fbclid=IwAR2QJz36aPBho12dQyx3ZI n-wXcz8- C63UvR-4T3t Yi23geJCSRz8HWk>

## 語法  
...

```
PROG = STMTS
BLOCK = { STMTS }
STMTS = STMT*
STMT = WHILE | BLOCK | ASSIGN
WHILE = while (E) STMT
ASSIGN = id '=' E;
E = F (op E)*
F = (E) | Number | Id
...
```

作業:

<https://github.com/weng0418/sp108b/blob/master/README.md>

作業參考寫法 (沒有產生中間碼的版本)

<https://github.com/ccckmit/sp108b/tree/master/homework/compiler-if>

這個參考版不完整，所以會無法編譯執行，請自行完成剩下的部分！

## 第四周- assembler 組譯器

### 上課內容:

組譯器:

```
PS D:\ccc\sp\code\c\03-asmVm\hack\c> mingw32-make  
gcc -std=c99 -O0 asm.c c6.c -o asm  
gcc -std=c99 -O0 vm.c -o vm  
PS D:\ccc\sp\code\c\03-asmVm\hack\c> ./asm ../test/Add
```

```
===== PASS2 =====  
00: @2          0000000000000010 0002  
01: D=A         1110110000010000 ec10  
02: @3          0000000000000011 0003  
03: D=D+A       1110000010010000 e090  
04: @0          0000000000000000 0000  
05: M=D         1110001100001000 e308
```

Symbolic syntax: `dest = comp ; jump`

Binary syntax: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

comp	c1	c2	c3	c4	c5	c6
0	1	0	1	0	1	0
1	1	1	1	1	1	1
-1	1	1	1	0	1	0
D	0	0	1	1	0	0
A	1	1	0	0	0	0
D	0	0	1	1	0	1
JA	1	1	0	0	0	1
-D	0	0	1	1	1	1
-A	1	1	0	0	1	1
D+1	0	1	1	1	1	1
A+1	1	1	0	1	1	1
D-1	0	0	1	1	1	0
A-1	1	1	0	0	1	0
D+A	0	0	0	0	1	0
D-A	0	1	0	0	1	1
A-D	0	0	0	1	1	1
D&A	0	0	0	0	0	0
D A	0	1	0	1	0	1
a==0						
a==1						

dest	d1	d2	d3	effect: the value is stored in:
null	0	0	0	The value is not stored
M	0	0	1	RAM[A]
D	0	1	0	D register
MD	0	1	1	RAM[A] and D register
A	1	0	0	A register
AM	1	0	1	A register and RAM[A]
AD	1	1	0	A register and D register
AMD	1	1	1	A register, RAM[A], and D register

jump	j1	j2	j3	effect:
null	0	0	0	no jump
JGT	0	0	1	if out > 0 jump
JEQ	0	1	0	if out = 0 jump
JGE	0	1	1	if out ≥ 0 jump
JLT	1	0	0	if out < 0 jump
JNE	1	0	1	if out ≠ 0 jump
JLE	1	1	0	if out ≤ 0 jump
JMP	1	1	1	Unconditional jump

Symbolic:

Examples: `MD=D+1`

Binary:

`1110011111011000`

pass2

```
===== PASS2 =====  
00: @2          0000000000000010 0002  
01: D=A         1110110000010000 ec10  
02: @3          0000000000000011 0003  
03: D=D+A       1110000010010000 e090  
04: @0          0000000000000000 0000  
05: M=D         1110001100001000 e308
```

./asm ../test/sum  
sum.asm 的組譯結果

```
00: @10          0000000000001010 000a
01: D=A          1110110000010000 ec10
02: @0           0000000000000000 0000
03: M=D          1110001100001000 e308
    p.key=i *p.value=16 top=26
04: @i           0000000000010000 0010
05: M=1          1110111111001000 efc8
    p.key=sum *p.value=17 top=27
06: @sum         0000000000010001 0011
07: M=0          1110101010001000 ea88
    (LOOP)
08: @i           0000000000010000 0010
09: D=M          1111110000010000 fc10
0A: @R0          0000000000000000 0000
0B: D=D-M        1111010011010000 f4d0
0C: @STOP        0000000000010110 0016
0D: D;JGT        1110001100000001 e301
0E: @i           0000000000010000 0010
0F: D=M          1111110000010000 fc10
10: @sum         0000000000010001 0011
11: M=D+M        1111000010001000 f088
12: @i           0000000000010000 0010
13: M=M+1        1111110111001000 fdc8
14: @LOOP        0000000000010000 0008
15: 0;JMP        1110101010000111 ea87
    (STOP)
16: @sum         0000000000010001 0011
17: D=M          1111110000010000 fc10
18: @R1          0000000000000001 0001
19: M=D          1110001100001000 e308
```

assembler pass1

```
void pass1(string inFile) {
    printf("===== PASS1 =====\n");
    char line[100]="";
    FILE *fp = fopen(inFile, "r");
    int address = 0;
    while (fgets(line, sizeof(line), fp)) {
        char *code = parse(line);
        if (strlen(code)==0) continue;
        printf("%02d:%s\n", address, code);
        if (code[0] == '(') {
            char label[100];
            sscanf(code, "%[^)]", label);
            symAdd(&symMap, label, address);
        } else {
            address ++;
        }
    }
    fclose(fp);
}
```

symAdd(&symMap, label, address); // 記住符號位址，給 pass2 編碼時使用

pass2 程式碼

```
void pass2(string inFile, string hackFile, string binFile) {
    printf("===== PASS2 =====\n");
    char line[100], binary[17];
    FILE *fp = fopen(inFile, "r");
    FILE *hfp = fopen(hackFile, "w");
    FILE *bfp = fopen(binFile, "wb");
    int address = 0;
    while (fgets(line, sizeof(line), fp)) {
        char *code = parse(line);
        if (strlen(code)==0) continue;
        if (line[0] == '(') {
            printf("%s\n", line);
        } else {
            code2binary(code, binary);
            uint16_t b = c6btoi(binary);
            printf("%02X: %-20s %s %04x\n", address, code, binary, b);
            fprintf(hfp, "%s\n", binary);
            fwrite(&b, sizeof(b), 1, bfp);
            address ++;
        }
    }
}
```

pass1

```
void pass1(string inFile) {
    printf("===== PASS1 =====\n");
    char line[100] = "";
    FILE *fp = fopen(inFile, "r");
    int address = 0;
    while (fgets(line, sizeof(line), fp)) {
        char *code = parse(line);
        if (strlen(code)==0) continue;
        printf("%02d:%s\n", address, code);
        if (code[0] == '(') {
            char label[100];
            sscanf(code, "(%[^)])", label);
            symAdd(&symMap, label, address);
        } else {
            address ++;
        }
    }
    fclose(fp);
}
```

作業:

<https://github.com/weng0418/sp108b/tree/master/%E4%BD%9C%E6%A5%AD0/%E7%B7%A8%E8%AD%AF%E5%99%A8-if2>

# 問題

## 同學問題:

無法 git pull

## 老師回答:

git pull 失敗的話，先打 git stash  
然後再 git pull

## 同學問題:

請問一下 irvm.c 裡面的 trace 是甚麼東西?

## 老師回答:

trace 就只是印出指令執行狀況  
#define trace printf

## 同學問題:

.c 跟 .h 檔的差異在哪

## 老師回答:

.c 檔通常放程式碼, .h 是表頭, 通常放 define 的那些東西

## 同學問題:

老師這段在做什麼 int \*varAdd(char \*name) {  
return mapAdd(&varMap, name, &t[0])->value;  
}

## 老師回答:

map.c 我們定義了一個可搜尋的結構 (雜湊表)，然後 varAdd 就呼叫  
mapAdd(&varMap, ....) 去新增一個變數到 varMap 雜湊表裡面。  
不直接用 mapAdd 是這樣模組化會比較好，呼叫時參數會少一點



### 同學問題:

執行 irvm.c 是輸入 ./compiler test/sum.c -irvm 嗎?沒有跑出結果..

### 老師回答:

./compiler test/sum.c -ir -run

### 同學問題:

為甚麼是 run?

### 老師回答:

-run 就是執行中間碼

```
#include "compiler.h"
int main(int argc, char * argv[]) {
    int isLexDump = 0, isIrDump = 0, isRun = 0;
    for (int i=0; i<argc; i++) {
        if (eq(argv[i], "-lex")) isLexDump = 1;
        if (eq(argv[i], "-ir")) isIrDump = 1;
        if (eq(argv[i], "-run")) isRun = 1;
    }
    readText(argv[1], code, TMAX);
    lex(code);
    if (isLexDump) lexDump();
    parse();
    irPass2();
    if (isIrDump) irDump();
    if (isRun) irRun();
}
```

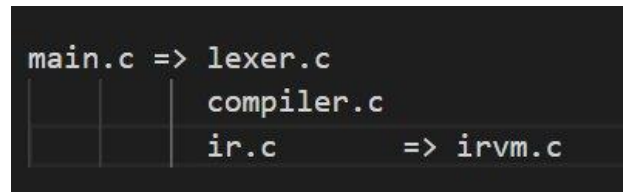
if (isRun) irRun();

這段會執行中間碼

同學問題:

可以畫一下這些檔案的關係圖嗎?

老師回答:



同學問題:

後四碼是甚麼?

老師回答:

後四碼 是 16 進位，也就是該二進位的對應 16 進位

同學問題:

可以說明一下 pass1 是如何讀取程式碼的嗎?是一行一行讀嗎?

老師回答:

是一行一行讀，先用 fopen 開檔，然後用 fgets 一行一行讀

同學問題:

```
int addr[SYM_SIZE] = {  
    0, 1, 2, 3,  
    4, 5, 6, 7,  
    8, 9, 10, 11,  
    12, 13, 14, 15,  
    16384, 24576,  
    0, 1, 2, 3, 4  
};
```

為甚麼後面又有 0, 1, 2, 3, 4?

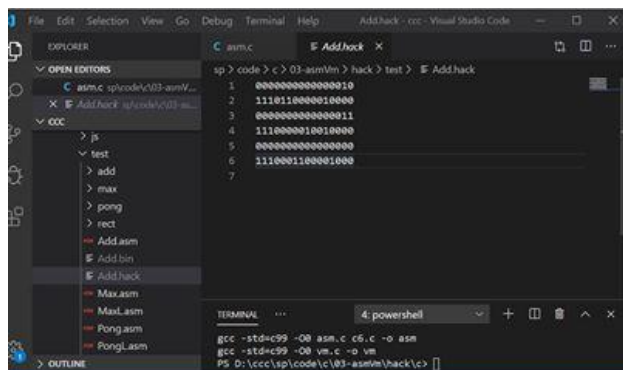
## 老師回答:

```
Pair symList[SYM_SIZE] = {
{"R0",&addr[0]}, {"R1",&addr[1]}, {"R2",&addr[2]}, {"R3",&addr[3]},
{"R4",&addr[4]}, {"R5",&addr[5]}, {"R6",&addr[6]}, {"R7",&addr[7]},
{"R8",&addr[8]}, {"R9",&addr[9]}, {"R10",&addr[10]}, {"R11",&addr[11]},
{"R12",&addr[12]}, {"R13",&addr[13]}, {"R14",&addr[14]}, {"R15",&addr[15]},
{"SCREEN",&addr[16]}, {"KBD",&addr[17]}, {"SP",&addr[18]}, {"LCL",&addr[19]},
{"ARG",&addr[20]}, {"THIS",&addr[21]}, {"THAT",&addr[22]}
};
R0 = 0, R1 = 1, ... R15 = 15
SCREEN = 16384, KBD = 24576
SP = 0, LCL=1, ARG = 2, THIS = 3, THAT = 4
其中的 SP = 0, LCL=1, ARG = 2, THIS = 3, THAT = 4 目前沒用到，但是 nand2tetris
習題規格要求 要寫進這些符號，以後會用到！
```

## 同學問題:

FILE \*hfp = fopen(hackFile, "w");老師你說這段是開啟.hack 的檔案那這個.hack 的內容可以去哪便看到?

## 老師回答:



第五週-休息

上課內容:

## vm 中的指令欄位提取

```
PS D:\ccc\sp\code\c\03-asmVm\hack>c> ./vm ../test/Add.bin
PC=0000 I=0002 A=0002 D=0000 m[A]=0000
PC=0001 I=EC10 A=0002 D=0002 m[A]=0000 a=0 c=30 d=2 j=0
PC=0002 I=0003 A=0003 D=0002 m[A]=0000
PC=0003 I=E090 A=0003 D=0005 m[A]=0000 a=0 c=02 d=2 j=0
PC=0004 I=0000 A=0000 D=0005 m[A]=0000
PC=0005 I=E308 A=0000 D=0005 m[A]=0005 a=0 c=0C d=1 j=0
exit program !
```

add 執行組譯:

```
PS D:\ccc\sp\code\c\03-asmVm\gcc\01-add> gcc main.c add.c -o add
```

```
PS D:\ccc\sp\code\c\03-asmVm\gcc\01-add> ./add
```

```
add(5, 8)=13
```

```
PS D:\ccc\sp\code\c\03-asmVm\gcc\01-add> gcc -S add.c -o add.s
```

foobar.s:

```
PS D:\ccc\sp\code\c\03-asmVm\gcc\01-add> gcc -fverbose-asm -S add.c -o add.s
```

```
PS D:\ccc\sp\code\c\03-asmVm\gcc\01-add> gcc -fverbose-asm -S main.c -o main.s
```

<https://eli.thegreenplace.net/2011/02/04/where-the-top-of-the-stack-is-on-x86>

```
PS D:\ccc\sp\code\c\03-asmVm\gcc\00-foobar> gcc -fverbose-asm -S foobar.c -o foobar.s
```

請大家看一下這個 foobar.s 的說明

<https://github.com/ccccourse/sp/tree/master/code/c/03-asmVm/gcc/00-foobar>

組合語言:

<https://github.com/ccccourse/sp/tree/master/code/c/03-asmVm/gcc/02-sum>

# 問題

同學問題:

老師你可以再講一次為什麼 要右移嗎那邊我不太懂

老師回答:

```
sp > code > c > 03-asmVm > hack > c > vm.md
1      a c d j
2      111a 1234 5612 3123
3      0x1000 = 0001 0000 0000 0000 >> 12 == 0000 0000 0000 000a
4      000a 0000 0000 0000 >> 12 == 0000 0000 0000 000a
5
6      0x0FC0 = 0000 1111 1100 0000 >> 6 == 0000 0000 00 c1..6
7      0000 c1..6 00 0000 >> 6 == 0000 0000 00 c1..6
8
9      0x0038 = 0000 0000 0011 1000 >> 3 == 0000 0000 0000 0 d1..3
10     0000 0000 00d1.3000 >> 3 == 0000 0000 0000 0 d1..3
11
12     0x0007 = 0000 0000 0000 0111 >> 0 == 0000 0000 0000 0 j1..3
13
```

同學問題:

是看得懂 windows 作業系統的組合語言嗎?

老師回答:

對, gcc 在 x86 上的組合語言

同學問題:

想問一下 x86 是甚麼...?

老師回答:

x86 是 Intel 一系列處理器的統稱

<https://zh.wikipedia.org/zh-tw/X86>

### 同學問題:

老師我的 foobar 產生不成功

```
PS C:\Users\user\Desktop\110710519計算機結構\co108a\sp\code\c\03-asm\gcc\00-foobar> gcc -fverbose-asm -S foobar.c -o foobar.s
gcc.exe: error: foobar.c: No such file or directory
gcc.exe: fatal error: no input files
compilation terminated.
```

### 老師回答:

你沒有把程式 foobar.c 創建存檔

請從 <https://eli.thegreenplace.net/2011/02/04/where-the-top-of-the-stack-is-on-x86> 剪貼進去!

### 同學問題:

老師你進入前置和離開的部分可以再說一次嗎?

### 老師回答:

這樣有聽懂嗎?

都是為了處理 esp, ebp 的定位



## 第七週- C 語言轉組合語言

### 上課內容:

#### **Fib 編譯:**

```
PS D:\ccc\sp\code\c\03-asmVm\gcc\03-fib> gcc main.c fib.c -o  
fib  
PS D:\ccc\sp\code\c\03-asmVm\gcc\03-fib> ./fib  
fib(10)=89
```

#### **inline.c:**

```
PS D:\ccc\sp\code\c\03-asmVm\gcc\04-inline> gcc inline.c -o inline  
PS D:\ccc\sp\code\c\03-asmVm\gcc\04-inline> ./inline  
sum = 30
```

展示了內嵌組合語言寫法

#### **globalCall.c:**

```
PS D:\ccc\sp\code\c\03-asmVm\gcc\05-globalcall> gcc -S globalCall.c -o  
globalCall.s  
PS D:\ccc\sp\code\c\03-asmVm\gcc\05-globalcall> gcc globalCall.c -o  
globalCall  
PS D:\ccc\sp\code\c\03-asmVm\gcc\05-globalcall> ./globalCall  
add(5, 8)=13
```

```
movl _a, %edx # a, D.1939 # edx = a  
movl _b, %eax # b, D.1939 # eax = b  
addl %edx, %eax # D.1939, D.1939 # eax = eax+edx = a+b  
movl %eax, _c # D.1939, c # c = eax
```

#### 工具鏈範例

<https://github.com/ccccourse/sp/tree/master/code/c/04-toolchain/gcc>

## 02-link:

```
PS D:\ccc\sp\code\c\04-toolchain\gcc\02-link> gcc main.c sum.c -o run
PS D:\ccc\sp\code\c\04-toolchain\gcc\02-link> ./run
sum(10)=55
```

```
$ gcc -S main.c -o main.s
$ gcc -S sum.c -o sum.s
```

```
$ gcc main.c sum.s -o run
$ ./run
sum(10)=55
```

```
$ gcc -c sum.c -o sum.o
$ gcc -c main.c -o main.o
```

```
$ gcc main.o sum.o -o run
$ ./run
```

## 04-make:

```
PS D:\ccc\sp\code\c\04-toolchain\gcc\04-make> mingw32-make
gcc -std=c99 -O0 sum.c main.c -o run
PS D:\ccc\sp\code\c\04-toolchain\gcc\04-make> ./run
sum(10)=55
https://github.com/ccccourse/sp/blob/master/code/c/04-toolchain/gcc/04-make/Makefile?fbclid=IwAR1NWOkDC4l8zntZeVkeW1g0Gr6oRSja9vRITAXcEKTrzb7BX2ER9II13A
```

## 05-makeLib

```
PS D:\ccc\sp\code\c\04-toolchain\gcc\05-makeLib> mingw32-make
ar -r libstat.a sum.o
```

```
ar: creating libstat.a
gcc -std=c99 -O0 -c main.c -o main.o
gcc -std=c99 -O0 libstat.a main.o -L ./ -lstat -o run
```

## 補充:

<https://gitbook.tw/chapters/fag/stash.html?fbclid=IwAR0KfjnVi08NfpYyrB9BUXeDeaGYto6X7czcbfbF-LFssiHUNQW7s8Mu5ig>  
<https://www.facebook.com/groups/ccccourse/permalink/294556158181369/>

x86 指令格式

<https://www.facebook.com/photo.php?fbid=1116743468377947&set=a.191713434214293&type=3&theater&ifg=1>

[https://www.sandpile.org/x86/opc\\_1.htm](https://www.sandpile.org/x86/opc_1.htm)

RISC-V

[https://zh.wikipedia.org/wiki/RISC-V?fbclid=IwAR2r8tGlo1s3kKEY1gqPmuhu\\_WXRKKQpukMgm0kQ7gT1MIUnYpxOSFQAiRU](https://zh.wikipedia.org/wiki/RISC-V?fbclid=IwAR2r8tGlo1s3kKEY1gqPmuhu_WXRKKQpukMgm0kQ7gT1MIUnYpxOSFQAiRU)

makefile:

<https://mropengate.blogspot.com/2018/01/makefile.html>

[https://www.sllabcd.github.io/blog/2016/10/03/how-to-write-make-file/?fbclid=IwAR1ReRqPlkY2rCq7RMCGd44SGayp3kvX7ZFk5QYx7PRVgzZvi\\_rhZqcD8C](https://www.sllabcd.github.io/blog/2016/10/03/how-to-write-make-file/?fbclid=IwAR1ReRqPlkY2rCq7RMCGd44SGayp3kvX7ZFk5QYx7PRVgzZvi_rhZqcD8C)

\$@：目前的目標項目名稱。

\$<：代表目前的相依性項目。

\$\*：代表目前的相依性項目，但不含副檔名。

\$?：代表需要重建（被修改）的相依性項目。

# 問題

同學問題:

老師我看不懂為甚麼會變 30

老師回答:

```
#include <stdint.h>

int main(int argc, char **argv)
{
    int32_t var1=10, var2=20, sum = 0;
    asm volatile ("addl %%ebx,%%eax;" /* eax = ebx + eax = 20 + 10 = 30 */
                  : "=a" (sum) /* output: sum = EAX = 30 */
                  : "a" (var1), "b" (var2) /* inputs: EAX = var1 = 10, EBX = var2 = 20 */
                  );
    printf("sum = %d\n", sum);
    return 0;
}
```

同學問題:

老師所以 addl 是加的意思嗎?

老師回答:

addl ebx, eax // -- eax = ebx + eax

l 是 long 常整數的意思

同學問題:

asm volatile {所以這個大括弧裡面的東西就是組合語言嗎?}

老師回答:

對!

同學問題:

老師 請問為什麼一開始要設 c=1

老師回答:

當我們用全域變數時，變數可直接存取，不需要透過 8(ebp) 這樣的框架站存器

沒設也 ok, 只是組合語言會稍有不同

## 同學問題:

老師我沒有顯示 ar 那段

```
PS C:\Users\Owner\Desktop\110710520-1\sp\sp\code\c\04-toolchain\gcc\05-makeLib
> mingw32-make
gcc -std=c99 -O0 libstat.a main.o -L ./ -lstat -o run
PS C:\Users\Owner\Desktop\110710520-1\sp\sp\code\c\04-toolchain\gcc\05-makeLib
> □
```

## 老師回答:

先把 libstat.a, \*.o 這類的檔刪掉，再重作就會有

## 第八週-期中作業+系統程式理論說明

### 上課內容:

#### 編譯執行 06-cpp/hello.cpp:

```
PS D:\ccc\sp\code\c\04-toolchain\gcc\06-cpp> g++ hello.cpp -o hello
PS D:\ccc\sp\code\c\04-toolchain\gcc\06-cpp> ./hello
hello!
```

#### **gdb** 的用法:

```
PS D:\ccc\sp\code\c\04-toolchain\gcc\07-gdb> gcc main.c add.c -o add -g
PS D:\ccc\sp\code\c\04-toolchain\gcc\07-gdb> gdb32 -q add
Reading symbols from add...done.
(gdb) break 6
Breakpoint 1 at 0x40135e: file main.c, line 6.
(gdb) r
Starting program: D:\ccc\sp\code\c\04-toolchain\gcc\07-gdb\add.exe
[New Thread 7152.0x1b6c]
Breakpoint 1, main () at main.c:6
6 int t = add(5, 8);
(gdb) n
7 printf("add(5, 8)=%d\n", t);
(gdb) n
add(5, 8)=13
8 return 0;
(gdb) n
9 }(gdb) n
0x004010fd in __mingw_CRTStartup ()
(gdb) n
Single stepping until exit from function __mingw_CRTStartup,
which has no line number information.
[New Thread 7152.0x2240]
[Inferior 1 (process 7152) exited normally]
```

(gdb) quit

## 系統程式投影片:

第一章-系統軟體:<https://www.slideshare.net/ccckmit/1-73472884>

第二章-電腦的硬體結構: <https://www.slideshare.net/ccckmit/2-73472886?fbclid=IwAR1AHeg0mafThpOvH95inMCYiFc-QDYbtjp5mx6hwU5pAwJIN67jkbH1Bpl>

第三章-組合語言:<https://www.slideshare.net/ccckmit/3-73472890>

第四章-組譯器:<https://www.slideshare.net/ccckmit/4-73472893>

第五章-連結與載入:<https://www.slideshare.net/ccckmit/5-73472900>

第六章-巨集處理器:<https://www.slideshare.net/ccckmit/6-73472903>

## 期中作業: [https://github.com/weng0418/-Interim-](https://github.com/weng0418/-Interim-report/blob/master/%E8%99%9B%E6%93%AC%E6%A9%9F%E7%A0%94%E7%A9%B6%E7%B3%BB%E7%B5%B1%E7%A8%8B%E5%BC%8F-%E6%9C%9F%E4%B8%AD%E5%A0%B1%E5%91%8A110710547%E7%BF%81%E7%91%8B%E6%B3%93%20(1).pdf)

[report/blob/master/%E8%99%9B%E6%93%AC%E6%A9%9F%E7%A0%94%E7%A9%B6%E7%B3%BB%E7%B5%B1%E7%A8%8B%E5%BC%8F-%E6%9C%9F%E4%B8%AD%E5%A0%B1%E5%91%8A110710547%E7%BF%81%E7%91%8B%E6%B3%93%20\(1\).pdf](https://github.com/weng0418/-Interim-report/blob/master/%E8%99%9B%E6%93%AC%E6%A9%9F%E7%A0%94%E7%A9%B6%E7%B3%BB%E7%B5%B1%E7%A8%8B%E5%BC%8F-%E6%9C%9F%E4%B8%AD%E5%A0%B1%E5%91%8A110710547%E7%BF%81%E7%91%8B%E6%B3%93%20(1).pdf)

## 補充

更多功能請參考

Gdb 調試利器: [https://linuxtools-rst.readthedocs.io/zh\\_CN/latest/tool/gdb.html](https://linuxtools-rst.readthedocs.io/zh_CN/latest/tool/gdb.html)

## 問題

### 同學問題:

旋轉是要做什麼的?

### 老師回答:

```
R1 = 10000000000000001
```

```
ROL R1, 4
```

```
R1 變成 000000000011000
```

就是像 SHL 一樣左移，但是移出部分會從右側補入。

# 第九週-目的檔、二進位工具、qemu 虛擬機

## 上課內容:

### 執行 01-bjdump

```
PS D:\ccc\sp\code\c\05-obj\01-objdump> gcc main.c add.c -o add
PS D:\ccc\sp\code\c\05-obj\01-objdump> ./add
add(5, 8)=13
PS D:\ccc\sp\code\c\05-obj\01-objdump> gcc -c add.c
PS D:\ccc\sp\code\c\05-obj\01-objdump> gcc -c main.c
PS D:\ccc\sp\code\c\05-obj\01-objdump> objdump -s add.o
add.o: file format pe-i386
Contents of section .text:
0000 5589e583 ec108b45 088945fc 8b450c89 U.....E..E..
0010 45f88b55 088b450c 01d0c9c3 E..U..E.....
Contents of section .rdata$zzz:
0000 4743433a 20287464 6d2d3129 20352e31 GCC: (tdm-1) 5.1
0010 2e300000 .0..
PS D:\ccc\sp\code\c\05-obj\01-objdump> objdump -d add.o
add.o: file format pe-i386
Disassembly of section .text:
00000000 <_add>:
0: 55 push %ebp
1: 89 e5 mov %esp,%ebp
3: 83 ec 10 sub $0x10,%esp
6: 8b 45 08 mov 0x8(%ebp),%eax
9: 89 45 fc mov %eax,-0x4(%ebp)
f: 89 45 f8 mov %eax,-0x8(%ebp)
12: 8b 55 08 mov 0x8(%ebp),%edx
15: 8b 45 0c mov 0xc(%ebp),%eax
18: 01 d0 add %edx,%eax
```



```

1a: c9 leave
1b: c3 ret
PS D:\ccc\sp\code\c\05-obj\01-objdump> objdump -h add.o
add.o: file format pe-i386
Sections:
Idx Name Size VMA LMA File off Algn
0 .text 0000001c 00000000 00000000 000000b4 2**2
CONTENTS, ALLOC, LOAD, READONLY, CODE
1 .data 00000000 00000000 00000000 00000000 2**2
ALLOC, LOAD, DATA
2 .bss 00000000 00000000 00000000 00000000 2**2
ALLOC
3 .rdata$zzz 00000014 00000000 00000000 000000d0 2**2
CONTENTS, ALLOC, LOAD, READONLY, DATA

```

## objdump 傾印執行檔

```

PS D:\ccc\sp\code\c\05-obj\01-objdump> objdump -d add.exe > add.dump
PS D:\ccc\sp\code\c\05-obj\01-objdump> objdump -h add.exe
add.exe: file format pei-i386
Sections:
Idx Name Size VMA LMA File off Algn
0 .text 00000c94 00401000 00401000 00000400 2**4
CONTENTS, ALLOC, LOAD, READONLY, CODE, DATA
1 .data 00000010 00402000 00402000 00001200 2**2
CONTENTS, ALLOC, LOAD, DATA
2 .rdata 00000148 00403000 00403000 00001400 2**2
CONTENTS, ALLOC, LOAD, READONLY, DATA
3 .eh_frame 000003a0 00404000 00404000 00001600 2**2
CONTENTS, ALLOC, LOAD, READONLY, DATA
4 .bss 00000060 00405000 00405000 00000000 2**2
ALLOC
5 .idata 0000037c 00406000 00406000 00001a00 2**2
CONTENTS, ALLOC, LOAD, DATA
CONTENTS, ALLOC, LOAD, DATA
7 .tls 00000020 00408000 00408000 00002000 2**2
CONTENTS, ALLOC, LOAD, DATA
8 .debug_aranges 00000018 00409000 00409000 00002200 2**0

```

```
CONTENTS, READONLY, DEBUGGING
9 .debug_info 00000dc5 0040a000 0040a000 00002400 2**0
CONTENTS, READONLY, DEBUGGING
10 .debug_abbrev 000000a9 0040b000 0040b000 00003200 2**0
CONTENTS, READONLY, DEBUGGING
11 .debug_line 000000d1 0040c000 0040c000 00003400 2**0
CONTENTS, READONLY, DEBUGGING
PS D:\ccc\sp\code\c\05-obj\01-objdump> objdump -d add.exe > add.dump
```

### 03-jitCall

```
PS D:\ccc\sp\code\c\05-obj\03-jitCall\win32> gcc jitCall.c -o jitCall
PS D:\ccc\sp\code\c\05-obj\03-jitCall\win32> ./jitCall
add(5, 8)=13
sum(10)=55
fib(10)=89
```

請注意看程式

<https://github.com/ccccourse/sp/blob/master/code/c/05-obj/03-jitCall/win32/jitCall.c>

特別是函數指標如何直接銜接到機器碼執行的部分。

如何做出機器碼的方法

<https://github.com/ccccourse/sp/tree/master/code/c/05-obj/03-jitCall/win32>

### 二進位工具 04-binutils

```
PS D:\ccc\sp\code\c\05-obj\04-binutils> gcc -c test.c
PS D:\ccc\sp\code\c\05-obj\04-binutils> nm test.o
00000000 b .bss
00000000 d .data
00000000 i .directve
00000000 r .rdata
00000000 r .rdata$zzz
```

```

00000000 r .rdata
00000000 r .rdata$zzz
00000000 D _a
00000004 C _b
00000004 C _c
U _printf
00000004 D _str
-aligncomm: "_b",2 -aligncomm: "_c",2
foo %d %s
GCC: (tdm-1) 5.1.0
PS D:\ccc\sp\code\c\05-obj\04-binutils> size test.o
text data bss dec hex filename
76 44 0 120 78 test.o
PS D:\ccc\sp\code\c\05-obj\04-binutils> nm test.o
00000000 b .bss
00000000 d .data
00000000 i .directve
00000000 r .rdata
00000000 r .rdata$zzz
00000004 C _b
00000004 C _c
00000000 T _foo
U _printf
00000004 D _str
PS D:\ccc\sp\code\c\05-obj\04-binutils> strip test.o
PS D:\ccc\sp\code\c\05-obj\04-binutils> nm test.o
D:\install\CodeBlocksPortable\App\CodeBlocks\MinGW\bin\nm.exe: test.o: no
symbols

```

## 05-link

```

PS D:\ccc\sp\code\c\05-obj\05-linker> gcc -c a.c b.c
PS D:\ccc\sp\code\c\05-obj\05-linker> objdump -h a.o
a.o: file format pe-i386
Sections:
Idx Name Size VMA LMA File off Algn
0 .text 00000034 00000000 00000000 000000b4 2**2
1 .data 00000000 00000000 00000000 00000000 2**2
ALLOC, LOAD, DATA

```

```

2 .bss 00000000 00000000 00000000 00000000 2**2
ALLOC
3 .rdata$zzz 00000014 00000000 00000000 000000e8 2**2
CONTENTS, ALLOC, LOAD, READONLY, DATA
PS D:\ccc\sp\code\c\05-obj\05-linker> objdump -h b.o
b.o: file format pe-i386
Sections:
Idx Name Size VMA LMA File off Algn
0 .text 00000024 00000000 00000000 000000b4 2**2
CONTENTS, ALLOC, LOAD, DATA
2 .bss 00000000 00000000 00000000 00000000 2**2
ALLOC
3 .rdata$zzz 00000014 00000000 00000000 000000dc 2**2
CONTENTS, ALLOC, LOAD, READONLY, DATA
PS D:\ccc\sp\code\c\05-obj\05-linker> ld -relocatable a.o b.o -o ab.o
PS D:\ccc\sp\code\c\05-obj\05-linker> objdump -h ab.o
ab.o: file format pe-i386
Sections:
Idx Name Size VMA LMA File off Algn
0 .text 00000058 00000000 00000000 000000b4 2**2
ALLOC
PS D:\ccc\sp\code\c\05-obj\05-linker> gcc a.o b.o -o ab
PS D:\ccc\sp\code\c\05-obj\05-linker> ./ab

```

a.o : text size = 34

b.o : text size = 24

ab.o : text size = 58

兩個程式段合併後 34 + 24 = 58

請仔細觀察段落大小

```

連結器會把段落合併：
a.o : text size = 34      data size = 0      .rdata$zzz size = 14
b.o : text size = 24      data size = 4      .rdata$zzz size = 14
ab.o : text size = 58      data size = 4      .rdata$zzz size = 28

兩個程式段合併後 34 + 24 = 58

```

## qemu 執行 xv6 作業系統

```
PS D:\ccc\course\sp\code\c\03-asmVm\qemu\xv6\img> qemu-system-i386 -  
nographic -drive file=fs.img,index=1,media=disk,format=raw -drive  
file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
```

## 下載 qemu

[https://www.qemu.org/download/?fbclid=IwAR01wM-C8dl1L9liv7DkjmYFrqEIBuSGFSBQ81DI\\_r1zkztVJy\\_xbDH3hb0#windows](https://www.qemu.org/download/?fbclid=IwAR01wM-C8dl1L9liv7DkjmYFrqEIBuSGFSBQ81DI_r1zkztVJy_xbDH3hb0#windows)

## QEMU 說明

<https://github.com/ccccourse/sp/tree/master/code/c/03-asmVm/qemu>

## 如何用 QEMU 執行 xv6 作業系統

<https://github.com/ccccourse/sp/tree/master/code/c/03-asmVm/qemu/xv6>

## 安裝 putty

<https://www.putty.org/?fbclid=IwAR0YsDLdNjBm8Jlv8-xkctNGVTHgohBtCU6iSHCeBIZxtuwKP18-e48060I>  
<https://the.earth.li/~sgtatham/putty/latest/w64/putty-64bit-0.73-installer.msi>

## 作業

請用這種方式算  $\text{power}(a, b)$

計算  $a^b$ ,  $a$  的  $b$  次方 !

<https://github.com/weng0418/sp108b/tree/master/%E4%BD%9C%E6%A5%AD0/%2003-jitCall>

## 補充

請確定 qemu 安裝好有設好路徑 PATH 了

C:\Program Files\qemu

然後打 `qemu-img` 應該看到下列結果

PS D:\ccc\sp\code\c\05-obj\05-linker> `qemu-img`

C:\Program Files\qemu\qemu-img.exe: Not enough arguments

Try '`qemu-img --help`' for more information

這樣才代表 PATH 設好了!

<https://bellard.org/?fbclid=IwAR0HairTJnJB-3icVjFduq1C7GT1O9Z7x79SNX1gXvwwp0Ouy-VWHypJ9Rw>

<http://archives.cse.iitd.ernet.in/~sbansal/csl862-virt/2010/readings/bellard.pdf>

**虛擬機:**

**Docker**

<https://zh.wikipedia.org/zh-tw/Docker?fbclid=IwAR3O3bJPD1YdJscCfzmen5hC2txzn0M3G8gCwO45b9OoCmEq03pOW60Ng1w>

**VMware**

<https://zh.m.wikipedia.org/zh-tw/VMware>

**VirtualBox**

<https://zh.wikipedia.org/wiki/VirtualBox>

**QEMU**

<https://zh.wikipedia.org/zh-tw/QEMU>

<http://sp1.wikidot.com/qemu>

<http://adaam-tw.blogspot.com/2010/06/kvm-qemu.html>

**Kubernetes 在夯什麼—K8S 基礎介紹**

<https://www.inwinstack.com/2018/05/08/what-is-kubernetes-part2/?fbclid=IwAR3ApIb1yp6x6EtulsywhHXX7HEd1KB6EujNez1pHdjzuJ4NLXdkwuDY68s>

# 問題

## 同學問題:

老師 請問這是要刪掉 code/c/05-obj/01-objdump/add.dump 才能 pull 嗎?

```
PS C:\Users\user\Desktop\110710519計算機結構\co108a\sp> git stash
No local changes to save
PS C:\Users\user\Desktop\110710519計算機結構\co108a\sp> git pull
error: The following untracked working tree files would be overwritten by merge:
code/c/05-obj/01-objdump/add.dump
Please move or remove them before you merge.
Aborting
Updating d49ab22..f4e3fe6
```

## 老師回答:

請刪除 add.dump 再 git pull 就可以了

## 同學問題:

請問要怎麼用 32 位元的 gcc 來執行 jitcall?

## 老師回答:

```
S D:\ccc\sp\code\c\05-obj\03-jitCall\win32> gcc jitCall.c -o jitCall
PS D:\ccc\sp\code\c\05-obj\03-jitCall\win32> ./jitCall
add(5, 8)=13
sum(10)=55
fib(10)=89
```

## 同學問題:

./jitCall 我這串打完之後沒有跑東西出來

## 老師回答:

```
gcc -m32 jitCall.c -o jitCall
```

### 同學問題:

請問 strip 跟 nm 指令是什麼意思?

### 老師回答:

nm: name mangling 名稱修飾

<https://stackoverflow.com/questions/6036066/strange-symbol-name-in-output-of-nm-command>

<https://sourceware.org/binutils/docs/binutils/nm.html>

Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler. See `c++filt`, for more information on demangling.

strip 中文:脫衣



## 第十週- mingw, msys2 、 , process 與 thread 、 thread 競爭情況與死結

上課內容:

下載 **msys2**:<https://www.msys2.org/>

測試 **forever.c**

```
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\01-forever> gcc forever.c -o forever
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\01-forever> ./forever
PS D:\ccc\course\sp\code\c\06-os1windows\02-forever> Start-Process forever.exe
// 此時會開出一個新視窗去跑 forever.exe
按 Ctrl-Alt-Del 可調出工作管理員，看行程狀態
```

```
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\01-forever> gcc forever.c -o forever
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\01-forever> ./forever
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\01-forever> Start-Process forever.exe
Start-Process forever.execc\sp\code\c\06-os1windows\01-stdc\01-forever>
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\01-forever> Get-Process -name
forever
50 5 616 2648 10.63 6616 1 forever
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\01-forever> Start-Process -
NoNewWindow forever.exe
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\01-forever> Get-Process -name
Handles NPM(K) PM(K) WS(K) CPU(s) Id SI ProcessName
-----
58 5 616 2624 19.64 3432 1 forever
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\01-forever> Stop-Process -name
forever
```

```
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\01-forever> Get-Process
Handles NPM(K) PM(K) WS(K) CPU(s) Id SI ProcessName
-----
539 31 18136 14104 0.64 2268 1 Applicatio...
334 21 3996 868 1.42 6816 1 AsusTPCenter
113 9 1400 196 0.17 7920 1 AsusTPHelper
...
```

## 執行 cat.c

```
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\02-cat> gcc cat.c -o cat
```

```
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\02-cat> ./cat cat.c
```

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

```
#define TEXT_SIZE 1000000
```

```
char text[TEXT_SIZE];
```

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

```
char *fileName = argv[1];
```

```
FILE *file = fopen(fileName, "rt");
```

```
int len = fread(text, 1, TEXT_SIZE-1, file);
```

```
text[len] = 0;
```

```
printf("%s\n", text);
```

```
}
```

```
PS D:\ccc\sp\code\c\06-os1windows\01-stdc\02-cat> ./cat README.md
```

```
# cat.c
```

```
## 蝨刻陌?瑁?
```

```
PS D:\ccc\course\sp\code\c\06-os1windows\01-stdc\02-cat> gcc cat.c -o cat
```

```
PS D:\ccc\course\sp\code\c\06-os1windows\01-stdc\02-cat> ./cat cat.c
```

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

```
#define TEXT_SIZE 1000000
```

```
char text[TEXT_SIZE];
```

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

```
char *fileName = argv[1];
```

```
FILE *file = fopen(fileName, "rt");
```

```
int len = fread(text, 1, TEXT_SIZE-1, file);
```

```
text[len] = 0;
```

```
printf("%s\n", text);
```

```
}
```

```
...
```

批注 [陳泓1]: 中文會是亂碼 ....

這是因為 C 語言預設不是用 UNICODE，而是 ASCII，所以中文會有問題。C 語言處理中文 Unicode 應該用寬字集，或者轉換後再輸出。

## 安裝 gcc

```
pacman -S gcc
```

## 安裝 make

```
pacman -S make
```

## 確認有無安裝成功

```
$ gcc
```

```
gcc: 嚴重錯誤：沒有輸入檔案  
編譯中斷。
```

```
$ make
```

```
make: *** 沒有指明目標並且找不到 makefile。 停止。  
這樣的回應代表你已經安裝好了 gcc 和 make
```

## 用 msys2 測試 task.c

```
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
```

```
$ gcc task.c -o task
```

```
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
```

```
$ ./task hello
```

```
0:hello
```

```
1:hello
```

```
2:hello
```

```
3:hello
```

```
4:hello
```

```
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
```

```
$ ./task A
```

```
0:A
```

```
1:A
```

```
2:A
```

```
3:A
```

```
4:A
```

```
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
```

```
$ ./task A & task B
```

```
[1] 1127
```

```
0:A
```

```
bash: task : 命令找不到
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ 1:A
2:A
3:A
4:A
^C
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ ./task A & ./task B
[2] 1129
0:A
0:B
1:A
1:B
2:A
2:B
3:A
3:B
4:A
4:B
[1] 已完成 ./task A
[2]+ 已完成 ./task A
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ ./task A & ./task B & ./task C
[1] 1131
0:A
[2] 1132
0:B
0:C
1:A
1:B
1:C
2:A
2:B
2:C
3:A
3:B
3:C
```

4:A

4:B

4:C

[1]- 已完成 ./task A

[2]+ 已完成 ./task B

### 在 **msys2** 中用 **gcc** 編譯 **forever.c** 並測試

```
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ gcc forever.c -o forever
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ ./forever
MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ ./forever &
[1] 1140
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ ps
PID PPID PGID WINPID TTY UID STIME COMMAND
976 1 976 7548 ? 197609 14:14:31 /usr/bin/mintty
1141 977 1141 1380 pty0 197609 14:36:22 /usr/bin/ps
1140 977 1140 3552 pty0 197609 14:36:20 /d/ccc/sp/code/c/06-os1windows/03-
msys2/forever
977 976 977 912 pty0 197609 14:14:32 /usr/bin/bash
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ kill 1140
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ ps
PID PPID PGID WINPID TTY UID STIME COMMAND
976 1 976 7548 ? 197609 14:14:31 /usr/bin/mintty
1142 977 1142 2612 pty0 197609 14:36:34 /usr/bin/ps
977 976 977 912 pty0 197609 14:14:32 /usr/bin/bash
[1]+ Terminated ./forever
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ ^C
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$
```

## msys2 測試 georgeMary.c

```
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ gcc georgeMary.c -o georgeMary
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/03-msys2
$ ./georgeMary
George
Mary
-----
George
-----
-----
...
```

## 執行 04-pthread/race.c

```
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/04-pthread
$ gcc race.c -o race
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/04-pthread
$ ./race
counter=5658163
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/04-pthread
$ ./race
counter=-1348781
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/04-pthread
$ ./race
counter=5007901
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/04-pthread
$ ./race
counter=5997929
```

```
PIKA@DESKTOP-QOC5V2F MINGW32 /d/ccc/sp/code/c/06-os1windows/04-pthread
$ gcc -S race.c -o race.s
```

## 作業系統與 Thread

[http://misavo.com/blog/%E9%99%B3%E9%8D%BE%E8%AA%A0/%E6%9B%B8%E7%B1%8D/C%EF%BC%83%E7%A8%8B%E5%BC%8F%E8%A8%AD%E8%A8%88/thread?fbclid=IwAR22nbvBM6zo1bezzgzIBHQA2nNKlx554n8ogeGpQ5z7\\_5rQa2bBi6tr0Ho](http://misavo.com/blog/%E9%99%B3%E9%8D%BE%E8%AA%A0/%E6%9B%B8%E7%B1%8D/C%EF%BC%83%E7%A8%8B%E5%BC%8F%E8%A8%AD%E8%A8%88/thread?fbclid=IwAR22nbvBM6zo1bezzgzIBHQA2nNKlx554n8ogeGpQ5z7_5rQa2bBi6tr0Ho)

## 看 race.s

```
inc:
....
movl counter(%rip), %eax
addl $1, %eax
movl %eax, counter(%rip)
....
dec:
...
movl counter(%rip), %eax
subl $1, %eax
movl %eax, counter(%rip)
...
```

很多 thread 對共用變數存取會產生 競爭情況

### race condition

#### Race\_condition:

[https://en.wikipedia.org/wiki/Race\\_condition](https://en.wikipedia.org/wiki/Race_condition)

### race condition

加了兩次，卻只得到 1，而不是 2.

這就是競爭情況造成的錯誤!

Thread 1	Thread 2		Integer value
			0
read value		←	0
	read value	←	0
increase value			0
	increase value		0
write back		→	1
	write back	→	1

## 補充

死結:

[https://zh.wikipedia.org/wiki/%E6%AD%BB%E9%94%81?fbclid=IwAR1\\_Vp8Y06OHpr-FvEo2a6bPdtYNO4\\_SNAxKegVgoFjzVrMbrSzXx6MkDc8](https://zh.wikipedia.org/wiki/%E6%AD%BB%E9%94%81?fbclid=IwAR1_Vp8Y06OHpr-FvEo2a6bPdtYNO4_SNAxKegVgoFjzVrMbrSzXx6MkDc8)

作業系統基本上很難處理好死結。

寫程式的人必須要自己注意，不要寫出死結的程式！

## 問題

同學問題:

老師我的沒有跑出來

```
PS C:\Users\User\Desktop\110710519計算機結構\co108a\sp\code\c\06-os1windows\01-stdc\01-forever> Get-Process -name
Get-Process : 遺失 'Name' 參數的引數。請指定一個 'System.String[]' 型別的參數，然後再試一次。
位於 線路:1 字元:13
+ ~~~~~
+ Get-Process -name
+ ~~~~~
+ CategoryInfo          : InvalidArgument: (:) [Get-Process], ParameterBindingException
+ FullyQualifiedErrorId : MissingArgument,Microsoft.PowerShell.Commands.GetProcessCommand
```

老師回答:

GetProcess 不要加 -name 或 Get-Process -name forever

同學問題:

老師我的有問題

```
user@DESKTOP-REPO5F3 MINGW32 ~
$ cd C:\Users\user\Desktop\110710519計算機結構\co108a\sp\code\c\06-os1windows\03
-msys2
bash: cd: C:\Users\user\Desktop\110710519計算機結構\co108a\sp\code\c\06-os1windows\03-msys2: No such file or directory
```

老師回答:

Linux /UNIX/mac/MSYS2 指令:

cd 切換資料夾

pwd 顯示目前資料夾路徑

ls 列出所有檔案



`ps : process status` 列出所有行程

`kill` : 刪除行程

不是 `C:\....`

應該用 `cd /c/.....`

UNIX 沒有 C, D 槽的觀念

另外中文資料夾名稱最好不要用

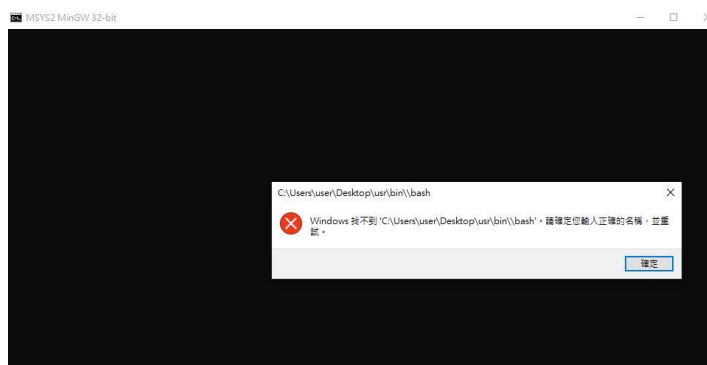
還有斜線不可用 `\`, 應該用 `/`

例如:

`cd /d/ccc/sp/code/c/06-os1windows/03-msys2`

## 同學問題:

老師 我把資料夾名字改掉之後 變這樣



## 老師回答:

請關掉 MSYS2 後重開一個新的

應該就可以解決了!

## 第十一週-改用 Google Meet

## 第十二週-mingw, msys2 、 , process 與 thread、thread 競爭情況與死結

### 上課內容:

在 vscode shell 裏執行

```
$ gcc erf.c -o erf -lm
```

```
PIKA@DESKTOP-QOC5V2F MINGW64 /d/ccc/sp/code/c/06-os1windows/03-  
msys2/03-math (master)
```

```
$ ./erf
```

```
normal(0,1) distribution between(-1.96, 1.96) is 0.950004
```

### 安裝 curl

```
pacman -S curl
```

### 安裝 wget

```
pacman -S wget
```

### pacman -S pkg-config

若沒有 make 請先 **pacman -S make**

```
PIKA@DESKTOP-QOC5V2F MINGW64 /d/ccc/sp/code/c/06-os1windows/03-  
msys2/04-pacman/03-glib
```

```
$ ls
```

```
glist.c glist.c htable.c Makefile README.md
```

```
PIKA@DESKTOP-QOC5V2F MINGW64 /d/ccc/sp/code/c/06-os1windows/03-  
msys2/04-pacman/03-glib
```

```
$ make
```

```
gcc glist.c -o glist `pkg-config --cflags glib-2.0` -g -Wall -std=gnu11 -O3 `pkg-config --  
libs glib-2.0`
```

```

gcc gclist.c -o gclist `pkg-config --cflags glib-2.0` -g -Wall -std=gnu11 -O3 `pkg-config --
libs glib-2.0`
gcc htable.c -o htable `pkg-config --cflags glib-2.0` -g -Wall -std=gnu11 -O3 `pkg-
config --libs glib-2.0`
PIKA@DESKTOP-QOC5V2F MINGW64 /d/ccc/sp/code/c/06-os1windows/03-
msys2/04-pacman/03-glib
$ ./glist
a
b
c
The list is now 0 items long
PIKA@DESKTOP-QOC5V2F MINGW64 /d/ccc/sp/code/c/06-os1windows/03-
msys2/04-pacman/03-glib
$ ./gclist
The list is now 0 items long
The list is now 2 items long
PIKA@DESKTOP-QOC5V2F MINGW64 /d/ccc/sp/code/c/06-os1windows/03-
msys2/04-pacman/03-glib
$ ./htable
There are 2 keys in the hash table
Jazzy likes Cheese
$ pkg-config --cflags glib-2.0
-mms-bitfields -IC:/msys64/mingw64/include/glib-2.0 -IC:/msys64/mingw64/lib/glib-
2.0/include -IC:/msys64/mingw64/include
$ pkg-config --libs glib-2.0
-LC:/msys64/mingw64/lib -lglib-2.0 -lintl

```

## 補充

請仔細看 **Makefile**

<https://github.com/ccccourse/sp/blob/master/code/c/06-os1windows/03-msys2/04-pacman/03-glib/Makefile>

特別是裡面的 **pkg-config** 那段

# 問題

## 同學問題:

老師我沒有 bash

## 老師回答:

這代表你灌 git 的時候沒有選用 bash  
沒關係，影響不大

## 同學問題:

老師請問要怎麼切到/d/ccc/sp/code/c/06-os1windows/03-msys2/03-math

## 老師回答:

msys2 的 /home 在  
C:\msys64\home  
cd /d/ccc/sp/code/c/06-os1windows/03-msys2/03-math

## 同學問題:

老師我用了 cd 但是沒有，是路徑不一樣嗎?

## 老師回答:

對，因為你的 D 巢沒有 ccc  
看你 clone 的 sp 在哪裡，就切到那裏去

## 同學問題:

老師這是哪裡出問題了

```
Owner@TC-Acerpu09grad MINGW64 ~
$ pacman -S curl
正在解決依賴關係...
正在檢查衝突的軟體包...

軟體包 (1) curl-7.70.0-1

總計安裝大小: 0.85 MiB
淨升級大小: 0.02 MiB

:: 進行安裝嗎? [Y/n] Y
(1/1) 正在檢查鑰匙圈中的鑰匙 [#####] 100%
(1/1) 正在檢查軟體包完整性 [#####] 100%
(1/1) 正在載入軟體包檔案 [#####] 100%
錯誤: 無法打開檔案 /var/cache/pacman/pkg/curl-7.70.0-1-x86_64.pkg.tar.zst: Child
process exited with status 127
錯誤: 無法提交處理 (無法開啟軟體包檔案)
發生錯誤，沒有軟體包被更新。
```

老師回答:

試著打 curl 看看，是否你已經有 curl 了

同學問題:

也在從新執行一遍了但還是沒辦法

```
$ curl
curl: try 'curl --help' or 'curl --manual' for more information
```

老師回答:

你已經有 curl 了，所以不用再下載

同學問題:

老師我的指令不成功

```
user@DESKTOP-REPO5F3 MINGW64 ~
$ pkg-config --cflags glib-2.0
Package glib-2.0 was not found in the pkg-config search path.
Perhaps you should add the directory containing 'glib-2.0.pc'
to the PKG_CONFIG_PATH environment variable
No package 'glib-2.0' found
```

老師回答:

請先裝 glib2

pacman -S glib2

同學問題:

老師我安裝了但還是出現不成功

老師回答:

今日 glib2 編譯失敗原因之補充更正 (影片)

<https://www.facebook.com/groups/ccccourse/permalink/333584860945165/>

## 第十三週-msys2, glib2 、 msys2, PostgreSQL

上課內容:

安裝並使用 **glib2**

```
pacman -Ss glib2
pacman -S mingw-w64-x86_64-glib2
make
./glist
./gslis
./htable
```

接著請安裝

```
$ pacman -S mingw-w64-x86_64-sqlite3
```

在安裝

```
$ pacman -S mingw-w64-x86_64-postgresql
```

```
$ make
$ ./sqlite_write
$ ls
$ ./sqlite_read
Id = 1
Name = Audi
Price = 52642
Id = 2
Name = Mercedes
Price = 57127
Id = 3
Name = Skoda
Price = 9000
```

```
Id = 4
Name = Volvo
Price = 29000
Id = 5
Name = Bentley
Price = 350000
Id = 6
Name = Citroen
Price = 21000
Id = 7
Name = Hummer
Price = 41400
Id = 8
Name = Volkswagen
Price = 21600
```

接著請閱讀 `sqlite_write.c`, `sqlite_read.c`

<https://sqlite.org/cli.html>

接著確認 `$ pacman -S mingw-w64-x86_64-postgresql`

```
## 創建資料庫
...
```

```
$ mkdir pgdata
$ initdb -D ./pgdata
$ pg_ctl -D ./pgdata -l logfile start
waiting for server to start.... done
server started
...
```

**## 執行 C 程式存取資料庫**

**\* 參考:** [用 C 語言透過 `libpq` 函式庫操作 PostgreSQL 資料庫](c/README.md)

**庫](c/README.md)**

```
$ createdb testdb
先切到 pgsql/c/ 的資料夾
```

**## 建置程式**

```
$ make
```



```
$ ./pghello
## make 專案
$ make
## pgcreate.c 創建 Cars 資料表
$ ./pgcreate
## 手動查詢 Cars 資料表
$ psql testdb
## 用 pqmultirows.c 程式列出資料表
$ ./pqmultirows
## 用 pqheader.c 列出 Cars 表格的欄位
$ ./pqheader
## 用 pglisttab.c 列出所有表格
$ ./pglisttab
## 用 pgtransact.c 進行原子交付
$ ./pgtransact
```

### 前置動作

```
$ pacman -S mingw-w64-x86_64-postgresql
$ mkdir pgdata
$ initdb -D ./pgdata
$ pg_ctl -D ./pgdata -l logfile start
waiting for server to start.... done
server started
## 先做 created
$ createdb testdb
```

請修改 **setting.h** 檔案的內容

```
#define connectStr "user=Tim dbname=testdb"
```

其中的 **Tim** 改成你的使用者名稱

你的 **msys2** 畫面中 **@** 前面的就是你的使用者名稱

**Tim@DESKTOP-QOC5V2F MINGW64 /d/ccc/sp2/database**

用它取代 **Tim**

改好之後切到 **pgsql/c** 然後打 **make**

```
$ make
```

**make** 成功，照這個執行一遍

<https://github.com/ccccourse/sp2/blob/master/database/pgsql/c/README.md>

接著改用交談操作

參考這個來做

<https://docs.postgresql.tw/tutorial/getting-started/creating-a-database>

```
$ createdb mydb
```

```
$ psql mydb
```

## 補充

請大家參考這篇操作

<https://github.com/ccccourse/sp2/tree/master/database/pgsql>

還有這篇

<https://github.com/ccccourse/sp2/blob/master/database/pgsql/c/README.md>

## 問題

同學問題:

老師我的沒有成功

```
正在處理軟體包變更-
(1/1) 正在重裝 mingw-w64-x86_64-glib2 [#####] 100%
找不到 schema 檔案：不做任何事。

user@DESKTOP-REPOSF3 MINGW64 /C:/Users/user/Desktop/110710519/col08a/sp2/msys2/03
-glib
$ make
bash: make: 命令找不到

user@DESKTOP-REPOSF3 MINGW64 /C:/Users/user/Desktop/110710519/col08a/sp2/msys2/03
-glib
$ ./glist
bash: ./glist: No such file or directory
```

老師回答:

請先裝 **make**

**pacman -S make**

同學問題:

```
Owner@TC-Acerpu09grad MINGW64 /C:/Users/Owner/Desktop/110710520-1/sp2/msys2/03-glib
$ make
make: *** 沒有指明目標並且找不到 makefile。 停止。
```

老師回答:

先切到 pgsql/c/ 的資料夾

同學問題:

老師我安裝好後打 make 出現這個

```
Owner@TC-Acerpu09grad MINGW32 /C:/Users/Owner/Desktop/110710520-1/sp2/msys2/03-glib
$ make
gcc glist.c -o glist `pkg-config --cflags glib-2.0` -g -Wall -std=gnu11 -O3 `pkg-config --libs glib-2.0`
Package glib-2.0 was not found in the pkg-config search path.
Perhaps you should add the directory containing 'glib-2.0.pc'
to the PKG_CONFIG_PATH environment variable
No package 'glib-2.0' found
Package glib-2.0 was not found in the pkg-config search path.
Perhaps you should add the directory containing 'glib-2.0.pc'
to the PKG_CONFIG_PATH environment variable
No package 'glib-2.0' found
glist.c:7:10: 嚴重錯誤: glib.h: No such file or directory
   7 | #include <glib.h>
     |          ~~~~~
編譯中斷。
make: *** [Makefile:10: glist] 錯誤 1
```

老師回答:

你沒裝好 glib2

```
$ pacman -S mingw-w64-x86_64-glib2
```

同學問題:

我在打一次後便這樣

```
10
$ pacman -S mingw-w64-x86_64-glib2
錯誤：無法初始化處理事務（無法鎖定資料庫）
錯誤：無法鎖定資料庫：File exists
如果你確認軟體包管理器沒有在運行，
你可以刪除 /var/lib/pacman/db.lck。
```

老師回答:

執行

```
rm /var/lib/pacman/db.lck
```

然後再執行 `pacman -S mingw-w64-x86_64-glib2`

同學問題:

老師我的不行

```
user@DESKTOP-REP05F3 MINGW64 /C:/Users/user/Desktop/110710519/co108a/sp2/msys2/03
~glib
$ make
gcc glist.c -o glist `pkg-config --cflags glib-2.0` -g -Wall -std=gnu11 -O3 `pkg-
-config --libs glib-2.0`
/bin/sh: pkg-config: 命令找不到
/bin/sh: pkg-config: 命令找不到
/bin/sh: gcc: 命令找不到
make: *** [Makefile:10: glist] Error 127

user@DESKTOP-REP05F3 MINGW64 /C:/Users/user/Desktop/110710519/co108a/sp2/msys2/03
~glib
$ ./glist
bash: ./glist: No such file or directory
```

老師回答:

pacman -S pkg-config

同學問題:

```
Raymond Huang
Raymond@LAPTOP-VIN8AR5H MSYS
/c/ccc/CCC/sp/sp2/msys2/03-glib
$ make
gcc glist.c -o glist `pkg-config --cflags glib-2.0` -g -
Wall -std=gnu11 -O3 `pkg-config --libs glib-2.0`
Package glib-2.0 was not found in the pkg-config
search path.
Perhaps you should add the directory containing
'glib-2.0.pc'
to the PKG_CONFIG_PATH environment variable
No package 'glib-2.0' found
Package glib-2.0 was not found in the pkg-config
search path.
Perhaps you should add the directory containing
'glib-2.0.pc'
to the PKG_CONFIG_PATH environment variable
No package 'glib-2.0' found
glist.c:7:10: 嚴重錯誤: glib.h: No such file or
directory
#include <glib.h>
~~~~~
編譯器斷。
make: *** [Makefile:10: glist] Error 1
```

老師回答:

請開 mingw64 版本的 msys2

不是 msys, 而是 MSYS2 MinGW 64-bit

同學問題:

我出現這個

```
gcc pghello.c -o pghello `pkg-config --cflags libpq` -g -Wall -std=gnu11 -O3 `pkg-config
--libs libpq`
```

```
pghello.c:2:10: 嚴重錯誤: libpq-fe.h: No such file or directory
```

```
2 | #include <libpq-fe.h>
```

```
| ^~~~~~
```

編譯中斷。

make: \*\*\* [Makefile:10 : pghello] 錯誤 1:

## 老師回答:

你的 `pkg-config` 有問題，可以自己把

`-I/mingw64/include` 取代 `makefile` 中的 ``pkg-config --cflags libpq``

也就是要改寫 `makefile`

然後

`$ pkg-config --libs libpq`

`-L/mingw64/lib -lpq -lintl -lssl -lcrypto -lm -lldap32 -lshell32 -lws2_32 -lsecur32`

也要手動取代掉

## 同學問題:

老師 我進不去交談操作

```
user@DESKTOP-REPO5F3 MINGW64 /C:/Users/user/Desktop/110710519/co108a/sp2/database
/pgsql/c
$ createdb mydb

user@DESKTOP-REPO5F3 MINGW64 /C:/Users/user/Desktop/110710519/co108a/sp2/database
/pgsql/c
$ psql mydb
psql: FATAL:  conversion between BIG5 and UTF8 is not supported
```

## 老師回答:

編碼轉換有問題，可能要參考這篇嘗試看看

<https://docs.postgresql.tw/server-administration/localization/character-set-support>

## 第十四週-msys2, glib2 、 msys2, PostgreSQL

### 上課內容:

請使用 **msys2/msys** 的終端機，不要用 **mingw32, mingw64** 的

```
$ gcc fork1.c -o fork1
$ ./fork1
$ ./fork2
```

```
$ gcc execvp1.c -o execvp1
$ ./execvp1
$ gcc system1.c -o system1
$ ./system1
$ gcc system1.c -o system1
$ ./system1
$ ./mysystem1
```

```
$ ./echo1
$ cat a.txt
$ gcc fecho1.c -o fecho1
$ ls
$ ./fecho1
$ ls
$ cat b.txt
```

### **Myshell**

```
$ ./myshell
```

**bug: cd** 無法正確運作

## myshell 第二版

```
$ gcc myshell.c -o myshell
$ ./myshell
myshell:/d/ccc/course/sp2/my/shell/v2 $ ls
myshell.c myshell.exe path.txt
myshell:/d/ccc/course/sp2/my/shell/v2 $ cd ..
myshell:/d/ccc/course/sp2/my/shell $ ls
my.h myshell.exe README.md v1 v2
myshell:/d/ccc/course/sp2/my/shell $ cd ..
myshell:/d/ccc/course/sp2/my $ ls
c5 shell telnet
myshell:/d/ccc/course/sp2/my $ cd ..
myshell:/d/ccc/course/sp2 $ ls
asm cpu embed hack msys2 net obj project tool
compiler database gcc linux my note.md os README.md vm
myshell:/d/ccc/course/sp2 $ echo hello world
hello world
myshell:/d/ccc/course/sp2 $ cd /
myshell:/ $ ls
asm.exe home mingw32 msys2.exe tmp
autorebase.bat InstallationLog.txt mingw32.exe msys2.ico usr
bin macro.exe mingw32.ini msys2.ini var
components.xml maintenancetool.dat mingw64 msys2_shell.cmd vm.exe
dev maintenancetool.exe mingw64.exe network.xml
etc maintenancetool.ini mingw64.ini proc
myshell:/ $
user@DESKTOP-96FRN6B MSYS /d/ccc/course/sp2/my/shell/v2
```myshell:/d/ccc/sp2/os/04-myshell/v2 $ ls
myshell.c myshell.exe path.txt README.md
myshell:/d/ccc/sp2/os/04-myshell/v2 $ cd ..
myshell:/d/ccc/sp2/os/04-myshell $ cd ..
myshell:/d/ccc/sp2/os $ cd /
myshell:/ $ ls
autorebase.bat maintenancetool.dat msys2.exe
bin maintenancetool.exe msys2.ico
components.xml maintenancetool.ini msys2.ini
dev mingw32 msys2_shell.cmd
```

```
etc mingw32.exe network.xml
home mingw32.ini proc
InstallationLog.txt mingw64 tmp
installer.dat mingw64.exe usr
installerResources mingw64.ini var
myshell:/ $ cd usr
myshell:/usr $ ls
bin include libexec share ssl x86_64-pc-msys
etc lib local src var
myshell:/usr $ exit
```

### myshell v2

<https://github.com/ccccourse/sp2/blob/master/os/04-myshell/v2/myshell.c>

注意這一行

```
sprintf(fullcmd, "cd %s;%s;pwd>%s", path, cmd, pathFile); // fullcmd = 切到 path;
使用者輸入的命令; 將路徑存入 pathFile 中。
```

傳統 Telnet 會話所傳輸的資料並未加密，帳號和密碼等敏感資料容易會被竊聽，因此很多伺服器都會封鎖 Telnet 服務，改用更安全的 SSH。

<https://zh.wikipedia.org/zh-tw/Telnet>

## 補充

<https://github.com/ccccourse/sp2/tree/master/os/02-fork/01-hello>

<https://github.com/ccccourse/sp2/blob/master/os/02-fork/01-hello/fork1.c>

<https://github.com/ccccourse/sp2/blob/master/os/02-fork/01-hello/fork2.c>

<https://github.com/ccccourse/sp2/blob/master/os/04-myshell/v1/myshell.c>



# 問題

同學問題:

這種分岔可以用在甚麼地方?

老師回答:

只要想產生子行程去執行某件事

同學問題:

```
open("a.txt", O_RDWR);
```

```
open("b.txt", O_CREAT|O_RDWR);
```

這兩行可以再說明一次嗎?

不太懂 O\_RDWR

老師回答:

<https://github.com/ccccourse/sp2/blob/master/os/03-fs/02-fecho/fecho1.c>

O\_RDWR 是可讀可寫的意思

同學問題:

那 O\_CREAT?

老師回答:

O\_CREAT 如果檔案不存在，就創建一個新檔

# 第十五週-程式與報告均要將原創與 引用清除區隔、組合語言 (Linux)、 網路 socket 程式

## 上課內容:

```
$ ssh gues@misavo.com
guest@misavo.com's password:
密碼為 csienqu (記得密碼打進去時不會回應是正常的)
guest@localhost:~$ ls
ccc selfie sp sp2 spMore
guest@localhost:~$ cd sp2
guest@localhost:~/sp2$ ls
asm database hack my note.md project vm
compiler embed linux nand2tetris obj README.md
cpu gcc msys2 net os tool
guest@localhost:~/sp2$ cd asm
guest@localhost:~/sp2/asm$ cd linux
guest@localhost:~/sp2/asm/linux$ ls
00-preface 03-fib 06-power 09-factorial README.md
01-hello 04-maxofthree 07-sum backup
02-hola 05-echo 08-average myMacro.s
guest@localhost:~/sp2/asm/linux$ cd 01-hello
guest@localhost:~/sp2/asm/linux/01-hello$ ls
hello helloMacro helloMacro.o helloMacro.s hello.o hello.s README.md
guest@localhost:~/sp2/asm/linux/01-hello$ gcc -c hello.s
guest@localhost:~/sp2/asm/linux/01-hello$ ld hello.o -o hello
guest@localhost:~/sp2/asm/linux/01-hello$ ls
hello helloMacro helloMacro.o helloMacro.s hello.o hello.s README.md
guest@localhost:~/sp2/asm/linux/01-hello$ ./hello
Hello, world
```

請先編譯執行 **hello.s**

```
# write(1, message, 13)
mov $1, %rax # system call 1 is write
mov $1, %rdi # file handle 1 is stdout
mov $message, %rsi # address of string to output
mov $13, %rdx # number of bytes
syscall # invoke operating system to do the write
```

這個是 **bios** 系統呼叫

<https://zh.wikipedia.org/wiki/BIOS%E4%B8%AD%E6%96%B7%E5%91%BC%E5%8F%A>  
[B](#)

目前的 **linux** 是用這份系統呼叫中斷表

[https://blog.rchapman.org/posts/Linux\\_System\\_Call\\_Table\\_for\\_x86\\_64/](https://blog.rchapman.org/posts/Linux_System_Call_Table_for_x86_64/)

**1 sys\_write** unsigned int fd const char \*buf size\_t count

**60 sys\_exit** int error\_code

<https://github.com/ccccourse/sp2/blob/master/asm/linux/01-hello/helloMacro.s>

**WRITES \$1, \$message, \$13**

<https://github.com/ccccourse/sp2/blob/master/asm/linux/myMacro.s>

```
guest@localhost:~$ cd sp2/asm/linux/01-hello/
guest@localhost:~/sp2/asm/linux/01-hello$ ls
hello helloMacro helloMacro.o helloMacro.s hello.o hello.s README.md
guest@localhost:~/sp2/asm/linux/01-hello$ gcc -c helloMacro.s
guest@localhost:~/sp2/asm/linux/01-hello$ ld helloMacro.o -o helloMacro
guest@localhost:~/sp2/asm/linux/01-hello$ ./helloMacro
Hello, world
```

這是有用巨集的版本

```
guest@localhost:~/sp2/asm/linux/02-hola$ gcc -no-pie hola.s -o hola
```

```
guest@localhost:~/sp2/asm/linux/02-hola$ ./hola
```

```
Hola, mundo
```

```
mov $message, %rdi
```

```
call puts
```

```
guest@localhost:~/sp2/asm/linux/02-hola$ gcc -no-pie holaMacro.s -o
```

```
holaMacroguest@localhost:~/sp2/asm/linux/02-hola$ ./holaMacro
```

```
Hola, mundo
```

```
uest@localhost:~/sp2/asm/linux/03-fib$ gcc -no-pie fib.s -o fib
```

```
guest@localhost:~/sp2/asm/linux/03-fib$ ./fib
```

0  
1  
1  
2  
3  
5  
8

```
guest@localhost:~/sp2/asm/linux/03-fib$ gcc -no-pie fibMacro.s -o fibMacro
guest@localhost:~/sp2/asm/linux/03-fib$ ./fibMacro
```

0  
1  
1  
2  
3  
5  
8  
13

```
guest@localhost:~/sp2/asm/linux/04-maxofthree$ gcc -std=c99 callmaxofthree.c
maxofthree.s && ./a.out
```

1  
2  
3  
4  
5  
6

```
guest@localhost:~/sp2/asm/linux/05-echo$ gcc -no-pie echo.s && ./a.out 25782 dog
huh '$$'
./a.out
25782
dog
huh
$$
```

```
guest@localhost:~/sp2/asm/linux/06-power$ gcc -no-pie power.s -o power
guest@localhost:~/sp2/asm/linux/06-power$ ./power
```

Requires exactly two arguments

```
guest@localhost:~/sp2/asm/linux/06-power$ ./power 2 3
8
```

```
guest@localhost:~/sp2/asm/linux/06-power$ ./power 3 4
81
```

```
guest@localhost:~/sp2/asm/linux/07-sum$ gcc callsum.c sum.s -o callsum
```

```
guest@localhost:~/sp2/asm/linux/07-sum$ ./callsum
```

26.7000000

67.2000000

0.0000000

89.1000000

```
uest@localhost:~/sp2/asm/linux/08-average$ gcc -no-pie average.s -o average
```

```
guest@localhost:~/sp2/asm/linux/08-average$ ./average 3 5 2 4 1
```

3

```
guest@localhost:~/sp2/asm/linux/08-average$ ./average 3 5 2 4
```

3.5

```
guest@localhost:~/sp2/asm/linux/08-average$ ./average 3 5
```

4

```
guest@localhost:~/sp2/asm/linux/09-factorial$ gcc -std=c99 callfactorial.c factorial.s
```

```
&& ./a.out
```

factorial( 0) = 1

factorial( 1) = 1

factorial( 2) = 2

factorial( 3) = 6

factorial( 4) = 24

factorial( 5) = 120

factorial( 6) = 720

factorial( 7) = 5040

factorial( 8) = 40320

factorial( 9) = 362880

factorial(10) = 3628800

factorial(11) = 39916800

factorial(12) = 479001600

factorial(13) = 6227020800

factorial(14) = 87178291200

factorial(15) = 1307674368000

```
factorial(16) = 20922789888000
factorial(17) = 355687428096000
factorial(18) = 6402373705728000
factorial(19) = 121645100408832000
```

<https://cs.lmu.edu/~ray/notes/gasexamples/>

### 01-timeTcp1

```
guest@localhost:~/sp2/net/01-timeTcp1$ ls
client.c Makefile README.md server.c
guest@localhost:~/sp2/net/01-timeTcp1$ make
gcc -std=c99 -O0 server.c -o server
gcc -std=c99 -O0 client.c -o client
guest@localhost:~/sp2/net/01-timeTcp1$ ./server&
[1] 25070
guest@localhost:~/sp2/net/01-timeTcp1$ ./client
Wed Jun 10 07:20:14 2020
guest@localhost:~/sp2/net/01-timeTcp1$ ./client
Wed Jun 10 07:20:16 2020
guest@localhost:~/sp2/net/01-timeTcp1$ ./client
Wed Jun 10 07:20:17 2020
guest@localhost:~/sp2/net/01-timeTcp1$ ./client
Wed Jun 10 07:20:18 2020
```

這些也可以在 **msys2 / msys** 裏編譯執行

### 02-timeTcp2

```
guest@localhost:~/sp2/net/02-timeTcp2$ make
gcc -std=c99 -O0 server.c ../net.c -o server
gcc -std=c99 -O0 client.c ../net.c -o client
guest@localhost:~/sp2/net/02-timeTcp2$ ps
PID TTY TIME CMD
24260 pts/7 00:00:00 bash
25070 pts/7 00:00:00 server
25123 pts/7 00:00:00 ps
guest@localhost:~/sp2/net/02-timeTcp2$ kill 25070
guest@localhost:~/sp2/net/02-timeTcp2$ ps
PID TTY TIME CMD
24260 pts/7 00:00:00 bash
25124 pts/7 00:00:00 ps
```

```

[1]+ Terminated ./server (wd: ~/sp2/net/01-timeTcp1)
(wd now: ~/sp2/net/02-timeTcp2)
guest@localhost:~/sp2/net/02-timeTcp2$ ps
PID TTY TIME CMD
24260 pts/7 00:00:00 bash
25127 pts/7 00:00:00 ps
guest@localhost:~/sp2/net/02-timeTcp2$ make
make: Nothing to be done for 'all'.
guest@localhost:~/sp2/net/02-timeTcp2$ ./server&
[1] 25131
guest@localhost:~/sp2/net/02-timeTcp2$ ./client
Wed Jun 10 07:22:37 2020
guest@localhost:~/sp2/net/02-timeTcp2$ ./client
Wed Jun 10 07:22:38 2020
guest@localhost:~/sp2/net/02-timeTcp2$
guest@localhost:~/sp2/net/02-timeTcp2$ ./client
Wed Jun 10 07:22:40 2020

```

### 03-telnet1

```

guest@localhost:~/sp2/net/03-telnet1$ make clean
rm -f server client *.exe
guest@localhost:~/sp2/net/03-telnet1$ make
gcc -Wall -std=gnu99 server.c ../net.c -o server
gcc -Wall -std=gnu99 client.c ../net.c -o client
guest@localhost:~/sp2/net/03-telnet1$ ./server&
[1] 25195
guest@localhost:~/sp2/net/03-telnet1$ ./client
connect to server 127.0.0.1 success!
127.0.0.1 $ ls
cmd=ls
client
client.c
Makefile
README.md
server
server.c
127.0.0.1 $ cat Makefile
cmd=cat Makefile

```

```

CC = gcc
CFLAGS = -Wall -std=gnu99
TARGET = server client
all: $(TARGET)
server: server.c ../net.c
$(CC) $(CFLAGS) $^ -o $@
client: client.c ../net.c
$(CC) $(CFLAGS) $^ -o $@
clean:
$(RM) $(TARGET) *.exe
127.0.0.1 $ ls
cmd=ls
client
client.c
Makefile
README.md
server
server.c
127.0.0.1 $ ls -all
cmd=ls -all
total 56
drwxrwxr-x 2 guest guest 4096 Jun 10 07:23 .
drwxrwxr-x 11 guest guest 4096 Jun 10 06:25 ..
-rwxrwxr-x 1 guest guest 14064 Jun 10 07:23 client
-rw-rw-r-- 1 guest guest 1154 Jun 10 06:25 client.c
-rw-rw-r-- 1 guest guest 210 Jun 3 07:34 Makefile
-rw-rw-r-- 1 guest guest 935 Jun 3 07:34 README.md
-rwxrwxr-x 1 guest guest 14008 Jun 10 07:23 server
-rw-rw-r-- 1 guest guest 1453 Jun 10 06:25 server.c
127.0.0.1 $ exit
缺陷 cd 不會換資料夾
guest@localhost:~/sp2/net/03-telnet1$ ./client
connect to server 127.0.0.1 success!
127.0.0.1 $ cd ..
cmd=cd ..
127.0.0.1 $ pwd
cmd=pwd
/home/guest/sp2/net/03-telnet1

```



```
127.0.0.1 $ cd ..
pwcmd=cd ..
127.0.0.1 $ d
cmd=pwd
/home/guest/sp2/net/03-telnet1
127.0.0.1 $ cd /
cmd=cd /
127.0.0.1 $ pwd
cmd=pwd
/home/guest/sp2/net/03-telnet1
127.0.0.1 $ exit
```

#### **04-telnet2**

```
& make
$ ./server&
[1] 25276
$ ./client
connect to server 127.0.0.1 success!
127.0.0.1
$ ls
$ pwd
$ cd ..
$ cd ..
$ pwd
$ ls
```

從 mac 透過 telnet2/client 連上 linux

```
$ make
$ ls
$ cd ../../../
$ s
$ cd ..
$ path=/home
```

## 重做

```
$ ./server&
```

```
[1] 25657
```

```
$ ps
```

```
PID TTY TIME CMD
```

```
$ ./client
```

大家在 linux 執行的時候只要執行 client 就好，否則很多 server 會衝到 ./client

會執行後，請看程式碼

<https://github.com/ccccourse/sp2/blob/master/net/01-timeTcp1/server.c>

<https://github.com/ccccourse/sp2/blob/master/net/01-timeTcp1/client.c>

## TimeTcp2

<https://github.com/ccccourse/sp2/blob/master/net/02-timeTcp2/server.c>

<https://github.com/ccccourse/sp2/blob/master/net/02-timeTcp2/client.c>

<https://github.com/ccccourse/sp2/blob/master/net/net.c>

```
guest@localhost:~/sp2/net/01-timeTcp1$ ping misavo.com
```

```
PING misavo.com (172.104.100.202) 56(84) bytes of data.
```

```
64 bytes from li1710-202.members.linode.com (172.104.100.202): icmp_seq=1
```

```
ttl=64 time=0.030 ms
```

```
64 bytes from li1710-202.members.linode.com (172.104.100.202): icmp_seq=2
```

```
ttl=64 time=0.050 ms
```

```
^C
```

```
--- misavo.com ping
```

```
char *host_to_ip(char *hostname, char *ip)
```

就是

把 misavo.com 這樣的 hostname 轉成

ip 172.104.100.202

```
net_init(net_t *net, int protocol, int side, int port, char *host)
```

初始化網路 socket

```
int net_connect(net_t *net)
```

client 端做的

```
int net_bind(net_t *net)
```

```
int net_listen(net_t *net)
```

```
int net_accept(net_t *net)
```

server 端做的

```
int net_close(net_t *net)
```

關閉連線，兩邊都可以做

<https://github.com/ccccourse/sp2/blob/master/net/net.c>

<https://github.com/ccccourse/sp2/blob/master/net/02-timeTcp2/server.c>

<https://github.com/ccccourse/sp2/blob/master/net/02-timeTcp2/client.c>

```
guest@localhost:~/sp2/net/01-timeTcp1$ ./client
```

```
Wed Jun 10 07:59:06 2020
```

```
guest@localhost:~/sp2/net/01-timeTcp1$ ./client
```

```
Wed Jun 10 07:59:08 2020
```

```
guest@localhost:~/sp2/net/01-timeTcp1$ ps
```

```
PID TTY TIME CMD
```

```
24260 pts/7 00:00:00 bash
```

```
25657 pts/7 00:00:00 server
```

```
26074 pts/7 00:00:00 ps
```

這是 **socket** 的 **timeTcp** 模組化後的範例程式

### telnet1

```
$ make
```

```
$ ./server&
```

```
[1] 26159
```

```
$ ps
```

```
PID TTY TIME CMD
```

```
1$ ./client
```

```
connect to server 127.0.0.1 success!
```

```
$ ls
```

```
$ pwd
```

```
$ cat Makefile
```

```
$ pwd
```

```
$ cd /
```

```
$ cd ../../
```

```
$ exit
```

### telnet2

```
$ make
```

```
$ ./server&
```

```
[1] 26250
```

```
$ ./client
```

```
connect to server 127.0.0.1 success!
```

```
$ ls
$ pwd
$ cat Makefile
$ pwd
$ cd /
$ exit
```

接著讀程式碼

<https://github.com/ccccourse/sp2/blob/master/net/03-telnet1/server.c>  
<https://github.com/ccccourse/sp2/blob/master/net/03-telnet1/client.c>

```
#include "../net.h"
int serv(int connfd) {
close(STDOUT_FILENO); // 關閉 stdout
dup2(connfd, STDOUT_FILENO); // 用 connfd 取代 stdout
dup2(connfd, STDERR_FILENO); // 用 connfd 取代 stderr
```

## 補充

《作業與報告》注意事項

[https://github.com/ccccourse/ccc109a/blob/master/00/md/submit.md?fbclid=IwAR2K\\_WS8r2YWxdkJJqRIWO782E7RITuT4lj\\_sL-yRMIGuprzdPW4V2wN4](https://github.com/ccccourse/ccc109a/blob/master/00/md/submit.md?fbclid=IwAR2K_WS8r2YWxdkJJqRIWO782E7RITuT4lj_sL-yRMIGuprzdPW4V2wN4)

著作權相關知識

[https://zh.wikipedia.org/wiki/Wikipedia:CC\\_BY-SA\\_3.0%E5%8D%8F%E8%AE%AE%E6%96%87%E6%9C%AC?fbclid=IwAR0CjLgWpG1ufht42qlvoPvsvU0cnxdI5PTBhL3p19wQaNCv4\\_j0MwO-UZQ](https://zh.wikipedia.org/wiki/Wikipedia:CC_BY-SA_3.0%E5%8D%8F%E8%AE%AE%E6%96%87%E6%9C%AC?fbclid=IwAR0CjLgWpG1ufht42qlvoPvsvU0cnxdI5PTBhL3p19wQaNCv4_j0MwO-UZQ)

<https://github.com/ccccourse/sp2/blob/master/asm/linux/README.md>

BIOS 中斷呼叫

[https://zh.wikipedia.org/wiki/BIOS%E4%B8%AD%E6%96%B7%E5%91%BC%E5%8F%A8?fbclid=IwAR2-6E0TY4w2s4AOeOlq5fYqT5v-EZ\\_pFvBKg6H-wn00Xd7WSkBbxUb8Ag](https://zh.wikipedia.org/wiki/BIOS%E4%B8%AD%E6%96%B7%E5%91%BC%E5%8F%A8?fbclid=IwAR2-6E0TY4w2s4AOeOlq5fYqT5v-EZ_pFvBKg6H-wn00Xd7WSkBbxUb8Ag)

從 近端的 mac 連到遠端 misavo.com

```
mac020:04-telnet2 mac020$ ./client 8080 misavo.com
```

```
connect to server 172.104.100.202 success!
```

```
172.104.100.202
```

```
$ ls
```

```
path=/home/guest/sp2/net/04-telnet2
```

```
cmd=ls
```

```
client
```

```
client.c
```

```
env.txt
```

```
Makefile
```

```
path.txt
```

```
server
```

```
server.c
```

```
172.104.100.202 /home/guest/sp2/net/04-telnet2
```

```
$ cd ../../..
```

```
path=/home/guest
```

```
cmd=cd /home/guest/sp2/net/04-telnet2;cd ../../..
```

```
172.104.100.202 /home/guest
```

```
$ ls
```

```
path=/home/guest
```

```
cmd=cd /home/guest;ls
```

```
ccc
```

```
selfie
```

```
sp
```

```
sp2
```

```
spMore
```

```
172.104.100.202 /home/guest
```

```
$ exit
```

這基本上就是 ssh 的功能，只是 ssh 有加密，我們的 telnet 沒加密。

# 問題

同學問題:

Windows 系統是連上 putty 之後 再執行下面的指令嗎？

老師回答:

可以用 ssh , putty 也行

先切到 :~/sp2/asm/linux/01-hello

# 第十六週-作業系統（理論部分）、嵌入式系統（slide）、作業系統與嵌入式範例

上課內容:

## nand2tetris os

<https://github.com/havivha/Nand2Tetris/blob/master/12/Keyboard.jack>

```
static Array keyboard;
```

```
let keyboard = 24576;
```

```
function char keyPressed() {
```

```
  return keyboard[0];
```

```
}
```

<https://github.com/havivha/Nand2Tetris/blob/master/12/Output.jack>

```
/**
```

```
 * Handles writing characters to the screen.
```

```
 * The text screen (256 columns and 512 rows) is divided into 23 text rows (0..22),
```

```
 * each containing 64 text columns (0..63).
```

```
 * Each row is 11 pixels high (including 1 space pixel), and 8 pixels wide
```

```
 * (including 2 space pixels).
```

```
 */
```

```
512*256
```

```
512/11=46.*, 256/8=32
```

```
let charMaps = Array.new(127);
```

```
// black square (used for non printable characters)
```

```
do Output.create(0,63,63,63,63,63,63,63,63,63,0,0);
```

```
// Assigns the bitmap for each character in the character set.
```

```
do Output.create(32,0,0,0,0,0,0,0,0,0,0,0); //
```

```
do Output.create(33,12,30,30,30,12,12,0,12,12,0,0); // !
```

```
do Output.create(34,54,54,20,0,0,0,0,0,0,0,0); // "
```

字體設定

# 補充

## 作業系統理論部分

<https://www.slideshare.net/ccckmit/10-73472927>

## 嵌入式系統

<https://www.slideshare.net/ccckmit/11-73472934>

### mini-arm-os

<https://github.com/jserv/mini-arm-os>

<https://github.com/jserv/mini-arm-os/blob/master/00-HelloWorld/reg.h>

<https://github.com/jserv/mini-arm-os/blob/master/00-HelloWorld/hello.c>

[https://github.com/jserv/mini-arm-os/blob/master/02-ContextSwitch-1/context\\_switch.S](https://github.com/jserv/mini-arm-os/blob/master/02-ContextSwitch-1/context_switch.S)

<https://github.com/jserv/mini-arm-os/blob/master/02-ContextSwitch-1/startup.c>

<https://github.com/jserv/mini-arm-os/blob/master/02-ContextSwitch-1/os.c>

[https://github.com/jserv/mini-arm-os/blob/master/06-Preemptive/context\\_switch.S](https://github.com/jserv/mini-arm-os/blob/master/06-Preemptive/context_switch.S)

<https://github.com/jserv/mini-arm-os/blob/master/06-Preemptive/syscall.S>

這個程式有三個 **thread**,

1. main
2. task1
3. task2

<https://github.com/jserv/mini-arm-os/blob/master/06-Preemptive/os.c>

<https://github.com/jserv/mini-arm-os/blob/master/07-Threads/os.c>

### riscv

<https://riscv.org/>

<http://www.eng.biu.ac.il/temanad/files/2019/11/RISC-V-Intro-for-Hackathon.pdf>

<https://riscv.org/specifications/>

### llvm

<https://llvm.org/docs/CompilerWriterInfo.html>

<https://github.com/ccccourse/sp2/blob/master/tool/llvm/02-hello/sum.ll>

<https://github.com/ccccourse/sp2/blob/master/tool/llvm/02-hello/sum.c>

<https://github.com/ccccourse/sp2/tree/master/tool/llvm>



**rust**

<https://doc.rust-lang.org/book/>

<http://askeing.github.io/rust-book/>

<http://askeing.github.io/rust-book/ownership.html>

<http://askeing.github.io/rust-book/references-and-borrowing.html>

<http://askeing.github.io/rust-book/lifetimes.html>

# 第十七週- Http + Crawler、Crawler、 mmap、rust 程式語言

上課內容:

## http server

<https://github.com/ccccourse/sp2/tree/master/net/05-http>

用 curl 抓自己的網站

```
mac020:05-http mac020$ curl http://misavo.com:9099/index.html
<html>
<body>
<ul>
<li><a href="hello.html">hello.html</a></li>
<li><a href="http://misavo.com">misavo.com</a></li>
</ul>
</body>
</html>mac020:05-http mac020$ curl http://misavo.com:9099/index.html
<html>
<body>
<ul>
<li><a href="hello.html">hello.html</a></li>
<li><a href="http://misavo.com">misavo.com</a></li>
</ul>
</body>
</html>mac020:05-http mac020$
</html>mac020:05-http mac020$ curl http://misavo.com:9099/hello.html
<html>
<body>
Hello!
<a href="https://tw.youtube.com/">YouTube</a>
</body>
</html>mac020:05-http mac020$
```

mac020:06-crawler mac020\$ ./crawSeq 9099 misavo.com

http://misavo.com:9099/index.html downloaded!

http://misavo.com:9099/hello.html downloaded!

http://misavo.com:9099/page1.html downloaded!

http://misavo.com:9099/page2.html downloaded!

http://misavo.com:9099/page3.html downloaded!

http://misavo.com:9099/page4.html downloaded!

http://misavo.com:9099/page5.html downloaded!

http://misavo.com:9099/page6.html downloaded!

http://misavo.com:9099/page7.html downloaded!

http://misavo.com:9099/page8.html downloaded!

http://misavo.com:9099/page9.html downloaded!

http://misavo.com:9099/page10.html downloaded!

http://misavo.com:9099/page11.html downloaded!

http://misavo.com:9099/page12.html downloaded!

http://misavo.com:9099/page13.html downloaded!

http://misavo.com:9099/page14.html downloaded!

http://misavo.com:9099/page15.html downloaded!

http://misavo.com:9099/page16.html downloaded!

http://misavo.com:9099/page17.html downloaded!

http://misavo.com:9099/page18.html downloaded!

http://misavo.com:9099/page19.html downloaded!

http://misavo.com:9099/page20.html downloaded!

## **Mmap**

<https://github.com/ccccourse/sp2/tree/master/os/05-mmap/time>

guest@localhost:~/sp2/os/05-mmap/time\$ ./timeProvider&./timeShower

[2] 352

07:05:49

07:05:50

07:05:51

07:05:52

07:05:53

07:05:54

07:05:55

07:05:56

07:05:57

07:05:58

07:05:59

07:06:00  
07:06:01  
07:06:02  
07:06:03  
07:06:04  
07:06:05  
07:06:06  
07:06:07  
07:06:08  
07:06:09  
07:06:10  
07:06:11  
07:06:12

### **rust**

<https://github.com/ccccourse/ccc109a/tree/master/00/rust>

```
mac020:01-hello mac020$ rustc hello.rs
mac020:01-hello mac020$ ./hello
Hello, world!
mac020:01-hello mac020$ pwd
/Users/mac020/Desktop/ccc/ccc109a/00/rust/00/01-hello
```

```
mac020:05-func mac020$ ./factorial_recursive
factorial(3)=6
factorial(5)=120
factorial(10)=3628800
```

### **cargo**

```
mac020:se mac020$ cargo new test1
Created binary (application) `test1` package
mac020:se mac020$ cd test1
mac020:test1 mac020$ cargo run
Compiling test1 v0.1.0 (/Users/mac020/Desktop/ccc/ccc109a/00/rust/se/test1)
Finished dev [unoptimized + debuginfo] target(s) in 6.74s
Running `target/debug/test1`
Hello, world!
mac020:test1 mac020$ pwd
/Users/mac020/Desktop/ccc/ccc109a/00/rust/se/test1
```

```
mac020:thread mac020$ cargo new thread1
Created binary (application) `thread1` package
mac020:thread mac020$ cd thread1
mac020:thread1 mac020$ ls
Cargo.toml src
mac020:thread1 mac020$ cargo build
Compiling thread1 v0.1.0
(/Users/mac020/Desktop/ccc/ccc109a/00/rust/sp/os/thread/thread1)
Finished dev [unoptimized + debuginfo] target(s) in 2.75s
mac020:thread1 mac020$ cargo run
Finished dev [unoptimized + debuginfo] target(s) in 0.09s
Running `target/debug/thread1`
Hello from thread number 0
Hello from thread number 1
Hello from thread number 2
Hello from thread number 3
Hello from thread number 4
Hello from thread number 5
Hello from thread number 6
Hello from thread number 7
Hello from thread number 8
Hello from thread number 9
```

## 補充

### IPC

<https://zh.wikipedia.org/zh-tw/%E8%A1%8C%E7%A8%8B%E9%96%93%E9%80%9A%E8%A8%8A>

### 套件

<https://crates.io/>

\$ cargo build

\$ cargo run

感謝  
觀看

