

# Code reviews

Chih-Hsuan Yang(SCC)

[zxc25077667@pm.me](mailto:zxc25077667@pm.me)

November 30, 2021

# whoami

1. The [Linux Kernel Module Programming Guide contributor](#).
2. The [first contributor of Debian Reference traditional Chinese](#).
3. The TA of OS.
4. The annoying student (hacker).
5. MacacaHub Team's Linux maintainer.
6. My special topic: [Docker container \(Linux kernel\)](#)

# Annoying SP

## 細節等等再說

Dear TA,

Oh, 我現在才看到網大上的 QA 更新，謝謝助教回覆。

有兩個問題，我想一併討論（就是「的樣子」跟「超重要」）：

硬碟與記憶體中 bit 等級的排列 (bitwise permutation)，稱作「的樣子」還是有語意模糊的地方(即便都稱作 Link time)，不過我想，用黃色小鴨除錯法：

1. 解釋一遍程式「如何存放」於硬碟(sector 對齊, inode table, 作業系統從來沒有保證硬碟中的 endian)；以及 Linker 「如何 static link」(Hint: symbol table)
2. 解釋一遍動態函式庫「如何載入」記憶體，如何透過「GOT/PLT」完成函式呼叫。[同時可以參考責投影片第 10 頁最後一行，這是怎麼做到的？或是可以下 `strace` 看看 glibc 給了 kernel 什麼東西。](#)

就會知道 bitwise permutation 是有改善空間的。

喔對，「超重要」這條呢，你是設定 chmod 444 對吧！我們有以下三條資訊：

1. 你是 root (你的 prompt 是用 # 字號)
2. 你可以「讀取當前」test.c 檔案，並「寫入當前」test.o 檔案
3. umask 幾乎所有發行版預設都是 022

這樣重新解釋一遍上面的黃色小鴨第二條，就會知道了。

最後，cc1 這稱呼我是第一次知道，謝謝助教資訊，至於 preprocessor 或是說 preprocessing 稱作 cpp 我是沒有在你附的參考資料中找到。

有一點比較神奇的是 GNU as(assembler), GNU ld(linker) 都看得懂 # 之類的 preprocessor symbol，若簡稱它做 cpp 好像哪邊怪怪的。

以上，很高興能夠再討論：)

Best,  
SCC

我知道當時被我吵的助教在現場

## Re: About the computer network midterm

From: zxc25077667 <zxc25077667@protonmail.com>

To: m093040067@nsysu.edu.tw <m093040067@nsysu.edu.tw>

CC: 閻浩文 <chov1234@gmail.com>

Date: Sunday, May 30th, 2021 at 9:03 PM

Dear TA,

先感謝助教的用心回覆

猶豫這麼久，相信你也是這篇回信我斷斷續續打了很久，關於需要 3-way handshake 的原因，我也想了一陣子。  
考慮點是誤區上說的 2-way handshake 有什麼錯誤，而是，到底需不需要 handshake？

TCP 的 checksum 只有 error detection 沒有 error correction。

他的算法大致這樣：（這是在寫作業，去參考 RFC793 以實物的部分式碼）

```
130     for (size_t i = 0; i < pkt.size(); i += 2)
131         cu += ~short(pkt[i]) << 8 | pkt[i + 1];
132     cu = (cu >> 16) + (cu & 65535);
133     cu += (cu >> 16);
134     cu += (int(htons(short(cu)))) << 16;
135     cu += 1000; // and get urgent pointer (FINME)
136     memcpy(pkt.data() + i * sizeof(int), &cu, sizeof(cu));
137 }
```

這樣的 checksum 算法只有 error detection 沒有 error correction 這也就表明：即使 TCP receiver 收到包檢查 checksum 後，發現 checksum 不對，也是利用重傳機制。

（實際上，checksum 無法更正封包錯誤無異，所以從節省開銷考量，可以選擇丟棄。）

所以，在考卷上，我只有說道：重傳 (retransmission) 並無錯誤，因為 checksum 也只是幫助重傳的內部機制。

以下是我過去對現在的理解：

1. TCP 的 retransmission 是不是構成 reliable 的唯一因素？

是，不過是 checksum, (duplicated) ack 這是 timeout... 都是提醒 sender 要進行「重傳」的動作。  
所以 checksum, (duplicated) ack 這是 timeout... 等等都應算在 retransmission 裡。

下詳第二點。

3. 亂序 retransmission 的條件？

只有兩種

1. Duplicated Ack (out-of-order [higher than current seq])

2. Timeout (不保證掉在中間的一段，或是沒到，或是 checksum 不正確掉)

4. 判斷 4-way handshake 是不是 reliable 的基石？

我想這問題跟前一樣的敘述：

> A transmission is reliable only if have a 3-way handshake.

這是錯的：

單單見的例子：DHCP 就是 4-way handshake 他仍然保持 reliable for IP automatically assigning.

那如果改成這樣：

> A transmission is reliable only if have at least 3-way handshake.

這也是錯的，考慮上第三點的敘述：

4.a) Duplicated Ack: 這個判斷只適用於 out of order 的規範，要提醒重送。

4.b) Timeout: 該機制只適用於 timeout 呀，要提醒重送。

重傳的機制，都讓 handshake 沒有關係。

5. 不能構成一個不用 handshake 就可以 reliable 的方式？

根據 reliable 定義，收的對且正確。

那，我們所知的 QUIC 就是以這個例子。

（這點我考慮很久，因為過去我們一直認為 TCP 可靠，QUIC 不那麼可靠，但是我實在找不出 TCP 和 QUIC 的可靠性差異）

（換句說，我認為 TCP 和 QUIC 是很像可靠的）

（如果助教有疑惑，不妨也想想看，這樣讓我想起過往的問題：「建立連線的目的是什麼？」）

以上，謝謝助教！

----- Original Message -----

On Tuesday, May 25th, 2021 at 4:59 PM, <m093040067@nsysu.edu.tw> wrote:

Dear 楊同學：

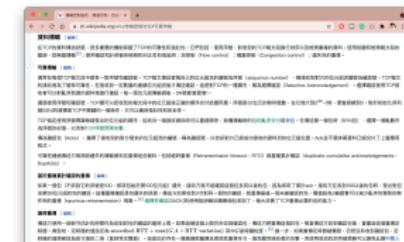
8. 通常而言，談TCP的可靠度，是相對於UDP而來。

所以要抓穩「TCP跟别的protocol有什麼不同」

在wiki上：

在TCP的資料傳送狀態，很多重要的機制保證了TCP的可靠性和強壯性。它們包括：使用序號，對收到的TCP報文段進行排序以及檢測重複的字節；使用校驗和檢測報文段的錯誤，即<sup>無錯傳輸</sup>；使用確認和計時器來檢測和糾正丟包或延遲；流控制 (Flow control)；擁塞控制 (Congestion control)；遺失包的重傳。

<https://zh.wikipedia.org/w/index.php?title=%E4%BC%A0%E8%93%86%E6%8E%A7%E5%8B%8B%E5%8D%8F%E8%AE%AE%E9%80%8A>



可靠傳輸不光是只有重傳這件事，

還有序號的正確性, ack, timer等等。

而這些的基礎，就是因為TCP跟UDP不一樣，是有先建立連線，而後才能有後面這些mechanisms，因此建立連線確實是可靠性的根基，甚至是基石。

至於如何通過這些發達，來達成可靠性，同學的說明跟別人比起來，確實是有不足的。

助教這邊沒辦法再給你超過3以上的分數。

否則對其他同學也不公平。

非常感謝楊同學再次的來信。

助教 歐陽安婕

寄件者: "楊志瑋" <zxc25077667@protonmail.com>

收件者: m093040067@nsysu.edu.tw

副本: "閻浩文" <chov1234@gmail.com>

寄件備份: 2021 5 月 21 星期五 上午 12:35:41

主旨: Re: About the computer network midterm

Dear TA,

# Outline

1. Linux?

2. Codes

Linux?

# Linux

1. Linux is not UNIX. [Click me](#)
2. Debian? [Click me](#)
3. Not only CLI. [Click me](#)

對，很抱歉，在場所有使用 Debian 生態系的人  
你們的電腦裡面**都有我**的貢獻。

# Codes

## Array indexing

```
9 FILE *direction(const char *filename, int option)
10 {
11     const char *_OPT_ [2] = {"w+", "a+"};
12     FILE *f = fopen(filename, _OPT_[option]);
13
14 #ifdef __DEBUG__
15     printf("filename: %s, opt: %d, FILE HEX: %p\n",
16         filename, option, f);
17 #endif
18     return f;
19 }
```

## hw2 myshell must in Ubuntu?

```
$ docker run -rm -it zxc25077667/sp_hw2
```

```
1 FROM gcc as builder
2
3 COPY . /src
4 WORKDIR /src
5 RUN make && mkdir /app && cp ./myshell /app/
6
7 FROM ubuntu:20.04
8 RUN apt-get update && \
9     apt-get install -y valgrind
10
11 COPY --from=builder /app /app
12 WORKDIR /app
```

# hw4 part1

## The busy box demo.

```
1 FLAGS= -Wall -Wextra -std=gn
2
3 all: compile
4     ln busybox hostinfo
5     ln busybox mydate
6     ln busybox printdir
7     ln busybox mycat
8     ln busybox pipe_ls
9
10 compile:
11     gcc ${FLAGS} -Ofast --st
12         busybox
13             strip busybox
14
15     dbg:
16         gcc -g lib.[ch] main.c -
17
18     clean:
19         rm busybox hostinfo mydate printdir mycat pipe_ls
```

### [2021 Network System Programming Homework 4]

This homework focuses on system programming and pipe.

#### Part 1:

1. Write, compile, and run a program named **hostinfo** that prints out system information in the following format.

Sample output:

```
ubuntu@ubuntu:~/Desktop/D083040001/part1$ ./hostinfo
hostname: ubuntu
5.0.0-23-generic
hostid: 8323329
```

2. Write, compile, and run a program named **mydate** that prints out the day and time in the following format.

Sample output:

```
ubuntu@ubuntu:~/Desktop/D083040001/part1$ ./mydate
Oct 8(Tue), 2019 1:22 PM
```

3. Write a program called **printdir** that prints the current directory. Determine what size buffer to pass to `getcwd()` for dynamic allocation.(Do not use `pwd()`.)

Sample output:

```
ubuntu@ubuntu:~/Desktop/D083040001/part1$ ./printdir
/home/ubuntu/Desktop/D083040001/part1
```

4. Write a program called **mycat** that is a simple version of the program cat. The program takes exactly one file name as argument; you should open it for reading and display its contents to the screen. Check that there is exactly one argument (`argc == 2`) and if not, display the usage message "**Usage: mycat filename**".

Sample output:

```
ubuntu@ubuntu:~/Desktop/D083040001/part1$ cat 123
123456
ABCDE
***
ubuntu@ubuntu:~/Desktop/D083040001/part1$ ./mycat 123
123456
ABCDE
***
```

## hw4 part2 built in shell

```
33  /***************************************************************************/  
34  /* lookup table */  
35  /***************************************************************************/  
36 #define __DECLARE.bi_quit(func)          \  
37     static inline void bi_##func(const char **argv) \  
38     {  
39         FREE_UNTIL_NULL((char **) argv);  
40         exit(0);  
41     }  
42 __DECLARE.bi_quit(logout);  
43 __DECLARE.bi_quit(bye);  
44 __DECLARE.bi_quit(exit);  
45 __DECLARE.bi_quit(quit);  
46  
47 #define COMMAND(cmd)    \  
48     {  
49     #cmd, bi_##cmd \  
50     }  
51  
52 static struct cmd {  
53     char *keyword;           /* When this field is argv[0] ... */  
54     void (*do_it)(const char **); /* ... this function is executed. */  
55 } inbuiltls[] = {  
56     COMMAND(builtin), /* List of (argv[0], function) pairs. */  
57     COMMAND(echo),    COMMAND(quit), COMMAND(exit), COMMAND(bye),  
58     COMMAND(logout),  COMMAND(cd),   COMMAND(pwd),  COMMAND(hostname),  
59     COMMAND(id),     {NULL, NULL} /* NULL terminated. */  
60 };  
61
```

## hw5 macros review

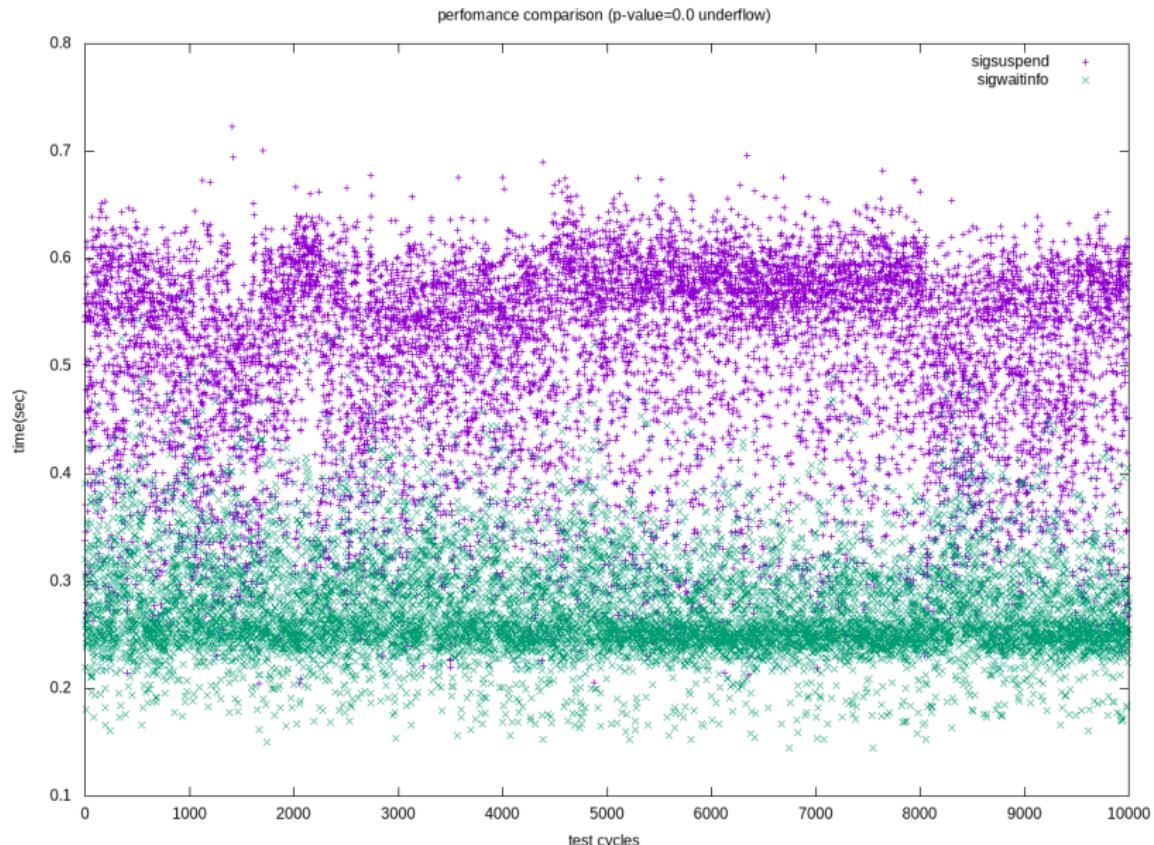
\$ code /media/d/git/nsysu/sp/hw5/2  
Compare my popen and [Android's popen](#)

# hw7 hooking

```
1 #ifndef __HOOK_H__
2 #define __HOOK_H__
3 #include <signal.h>
4
5 #define sigsuspend my_sigwaitinfo
6
7 int my_sigwaitinfo(const sigset_t * __set);
8
9#endif
```

```
79     case 0: /* child */
80         for (int scnt = 0; scnt < numSigs; scnt++) {
81             if (kill(getppid(), TESTSIG) == -1)
82                 errExit("kill");
83             if (sigsuspend(&emptyMask) == -1 && errno != EINTR)
84                 errExit("sigsuspend");
85         }
86         exit(EXIT_SUCCESS);
87
88     default: /* parent */
89         for (int scnt = 0; scnt < numSigs; scnt++) {
90             if (sigsuspend(&emptyMask) == -1 && errno != EINTR)
91                 errExit("sigsuspend");
92             if (kill(childPid, TESTSIG) == -1)
93                 errExit("kill");
94         }
95         exit(EXIT_SUCCESS);
96     }
```

# hw7 statistics



[https://man7.org/tlpi/code/online/dist/svmsg/svmsg\\_file.h.html](https://man7.org/tlpi/code/online/dist/svmsg/svmsg_file.h.html)

1. The client different.
2. The defined response message type *RESP\_MT\_FAILURE*.
3. serverId resource leakage.
4. **The kernel bug.**



**Talk is cheap.  
Show me the code.**

Linus Torvalds

quotefancy

[https://quotefancy.com/quote/1445782/  
Linus-Torvalds-Talk-is-cheap-Show-me-the-code](https://quotefancy.com/quote/1445782/Linus-Torvalds-Talk-is-cheap-Show-me-the-code)

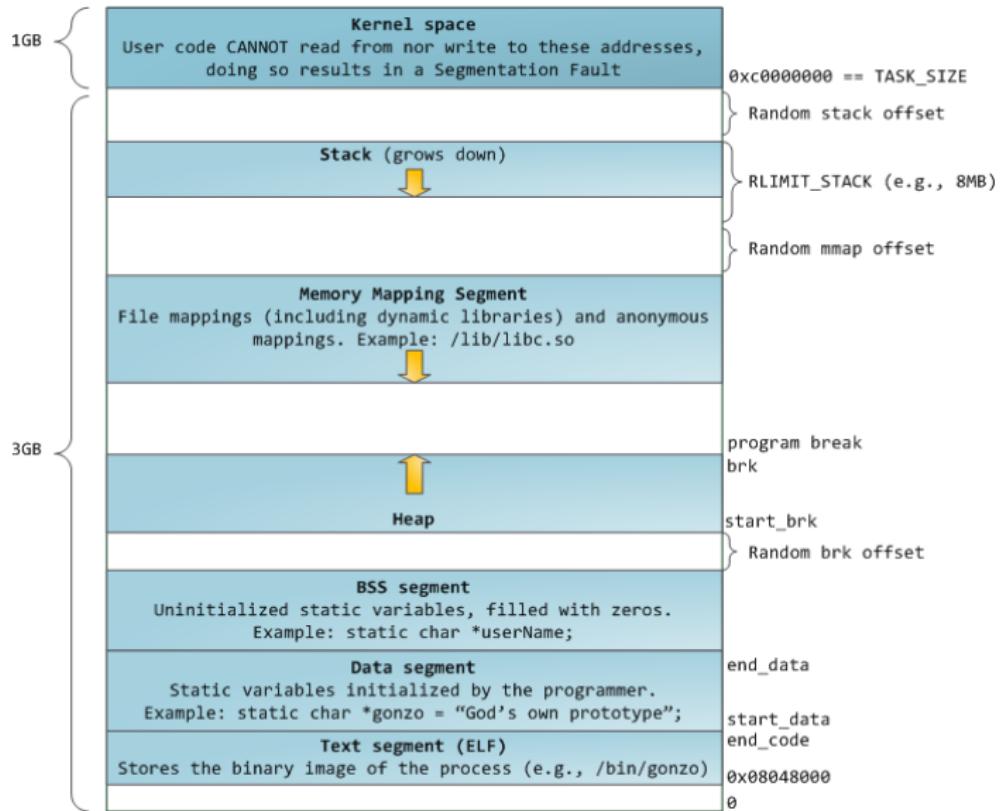
# High scalability VPN server

TCP, WireGuard, prof Lai.

<https://scc-net.tw/nextcloud/s/D5JSGsRgEayTYkE>

Open source in GPLv3

# Process



Go back to the origin.



我是楊志瓈，今年有修 林俊宏老師 的網際系統程式設計的學生。

有幾點希望可以小小討論（跟老師或跟 TA 都好）：

1. Vim is not vi （網大公告版之檔案名稱有誤），我相信網路上有許多 vi 與 Vim 的介紹，但是將兩者混用是不精確的。

2. GNU := GNU is Not Unix && Linux is Unix-like rather than UNIX. 詳細可以看 Wikipedia 上面的族譜，變可以大致瞭解其中族系關係。過去我有看過一部紀錄片《Revolution OS》（中譯：作業系統革命），如果您有興趣想要瞭解更多細節的話，可以參考裡面 RMS, Torvalds, Raymond... 的說法。

3. 《貓也會的 gcc》 簡報中的：

第 4 頁，第二行，cpp, cc1 What？沒有看懂在表達什麼，或是有什麼我不知道的區塊，懇請賜教。（我知道 gcc -E, -S, -c 等參數）

第 6 頁，「躺」應修正為「存放」，另外該句「的樣子」語意模糊 (ambiguous)，建議修正為嚴謹用語，例如：檔案路徑、mmap 映射分頁 等等。

第 12 頁下方，「軟件」應改使用為「軟體」

第 14 頁下方，ln -s libtest.so.1 libtest.so 應改為 ln -s libtest.so.1 /usr/lib/libtest.so，另外，超重要那行不必要，因為 default 權限是 755，已包含讀、執行權限。

第 15 頁，少了一行 extern int nine\_nine();

其他主要是課堂上老師的口誤：

當下我沒有紀錄下來，就憑印象大致跟老師討論：

1. apt package manager 不是源自於 Ubuntu，而是源自於 Debian 的 dpkg，相較於 centOS(很抱歉，該發行版即將停止支援) 的 dnf/yum (rpm 生態系) 實有所不同。

(另外在我使用的作業系統 Archlinux 5.14.8 所使用的 AUR 檔案格式、套件管理方式是有區別的) (對，我知道 Arch 的 Linux 版本不是很重要，但是就是說清楚而已)

我對 rpm 生態系不熟，對 apt 比較瞭解，據我所知：

'apt update' 會去搜尋 /etc/apt/ 底下的 source list 確定 fetch 套件的來源，以及相關憑證。不是 99% 都在裡面，而是有「常用」指令套件包在裡面

2. WSL2 已經是虛擬機，Windows 在 NT-kernel 下方實做一層 hypervisor (因未公開原始碼，我也沒有去瞭解，所以不知道細節)，但是確實 WSL2 已經是 VM。

這個 WSL2 可以開啟「GUI 界面」，如果老師手動安裝過 Archlinux 應該就會知道，並且知道怎麼把 X forward 出來：)

(但是 Windows 有沒有支援 X11 client 我不知道，我沒用 Windows 很久了。) (你可以用 VNC 等 server-client 把 DE 傳出來是沒問題的)

(WSL2 的 PID = 1 是 shell，如果要切換為 Linux 環境常用的 systemd 背後該怎麼做，老師應該知道中間會有一些 hack)

3. 關於過去學生都沒有這些問題：

我想，我跟過去學生不同。我有嘗試貢獻 Debain 官方文件不良部份，但是不確定原因得註冊帳號被拒絕。

我有改過 AOSP 的 Linux kernel，在我的 Android 上成功運行 Docker，同時也有 SELinux, microG.

我有閱讀並貢獻過，The Linux Kernel Module Programming Guide 5.x 一書。

我想說的是，我有信心我對 Linux distro. and the Linux kernel 有一定程度的掌握度。

以上，樂意與老師、助教討論，謝謝

Best,  
SCC

## 課程公告板

[回列表](#)

M093040064 (莊凱崴) 2021-09-28 17:00

0

SP電子書

Q&A

Q1. 第 4 頁，第二行，cpp, cc1 What ? 沒有看懂在表達什麼，或是有什麼我不知道的區塊，懇請賜教。（我知道 gcc -E, -S, -c 等參數）

A1. cpp, cc1都是gcc底下的組件。cpp會先對你的C code做一些預處理，cc1是整個編譯過程中實際把預處理後的C code轉成組語的組件。

請詳閱第3頁，或者官方doc：

[http://gcc.gnu.org/wiki/FAQ#include\\_search\\_path](http://gcc.gnu.org/wiki/FAQ#include_search_path)

Q2. 第 6 頁，「躺」應修正為「存放」，另外該句「的樣子」語意模糊 (ambiguous)，建議修正為嚴謹用語，例如：檔案路徑、mmap 映射分頁等等。

A2. 我覺得用「躺」比較平易近人，所以我不打算修。另外「的樣子」既不是檔案路徑也不是分頁檔，是那些0跟1的排列方式 aka 0跟1的長相。

Q3. 第 12 頁下方，「軟件」應改使用為「軟體」

A3. 改了，感謝提醒。

Q4. 第 14 頁下方，ln -s libtest.so.1 libtest.so 應改為 ln -s libtest.so.1 /usr/lib/libtest.so,

A5. 改了，感謝提醒。

Q5. 另外，超重要那行不必要，因為 default 權限是 755，已包含讀、執行權限。

那是你的系統default 權限是 755，別的系統未必是，umask去瞭解一下。

Q6. 第 15 頁，少了一行 extern int nine\_nine();

A6. 改了，感謝提醒。

附檔

[讀也會的gcc.pdf \(20MB\)](#)

[vi清義.pdf \(925KB\)](#)

# Annoying SP

Dear TA,

Oh, 我現在才看到網大上的 QA 更新，謝謝助教回覆。

有兩個問題，我想一併討論（就是「的樣子」跟「超重要」）：

硬碟與記憶體中 bit 等級的排列 (bitwise permutation)，稱作「的樣子」還是有語意模糊的地方(即便都稱作 Link time)，不過我想，用黃色小鴨除錯法：

1. 解釋一遍程式「如何存放」於硬碟(sector 對齊, inode table, 作業系統從來沒有保證硬碟中的 endian)；以及 Linker 「如何 static link」(Hint: symbol table)
  2. 解釋一遍動態函式庫「如何載入」記憶體，如何透過「GOT/PLT」完成函式呼叫。[同時可以參考責投影片第 10 頁最後一行，這是怎麼做到的？或是可以下 strace 看看 glibc 給了 kernel 什麼東西。](#)
- 就會知道 bitwise permutation 是有改善空間的。

喔對，「超重要」這條呢，你是設定 chmod 444 對吧！我們有以下三條資訊：

1. 你是 root (你的 prompt 是用 # 字號)
2. 你可以「讀取當前」test.c 檔案，並「寫入當前」test.o 檔案
3. umask 幾乎所有發行版預設都是 022

這樣重新解釋一遍上面的黃色小鴨第二條，就會知道了。

最後，cc1 這稱呼我是第一次知道，謝謝助教資訊，至於 preprocessor 或是說 preprocessing 稱作 cpp 我是沒有在你附的參考資料中找到。有一點比較神奇的是 GNU as(assembler), GNU ld(linker) 都看得懂 # 之類的 preprocessor symbol，若簡稱它做 cpp 好像哪邊怪怪的。

以上，很高興能夠再討論：)

Best,  
SCC

# Let's make a test

Trace the /proc/<pid>/maps

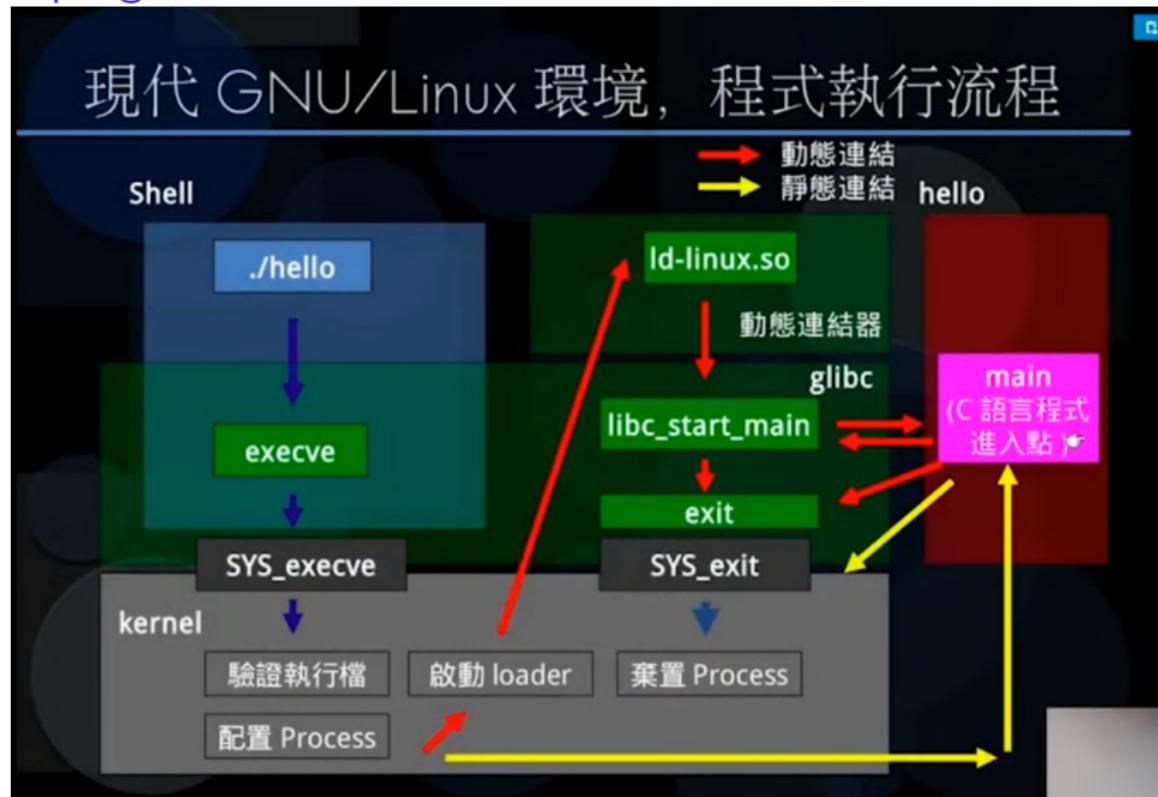
```
1 #include <unistd.h>
2 int main(void)
3 {
4     return sleep(9999999);
5 }
```

## Let's make a test

Trace the /proc/<pid>/maps

```
1 #include <unistd.h>
2 #include <stdlib.h>
3 int main(void)
4 {
5     void *p = malloc(100);
6     sleep(9999999);
7     free(p);
8     return 0;
9 }
```

# How programs are executed?



[https://www.slideshare.net/jserv/  
how-a-compiler-works-gnu-toolchain](https://www.slideshare.net/jserv/how-a-compiler-works-gnu-toolchain)