程式設計(二)-HW05

Due to 06/16 PM 11:59／授課老師：紀博文

一、基本資料

姓名：林育辰
系級：資工 111
學號：40771131H

二、檔案有哪些？

- **hw0501.c**
- **hw0502.c**
- **hw0503.c**
- **hw0503_header.h**
- **hw0504.c**
- **hw0505.c**
- **助教的 ifconfig (格式參考用)**
- **curl 7.70**
- **Makefile**
- **README.pdf**

◎每個.c 檔皆有註解！**請格外注意 README 中的格式要求**

三、如何執行？

請輸入 make
→編譯 hw0501.c ~ hw0505.c 檔
→產生 hw0501~hw0505

- hw0503 若要使用 debug_messenge，編譯時請加入 “-D”的 DEFINE 如：
  $ gcc -D [DEBUG_LEVEL] hw0503.c -o hw0503
- hw0504 請特別注意必須安裝 **curl-7.70.0** 版本

指令如下：
$ make
$ ./hw0501 [option] [filename] [-o] [filename]
$ ./hw0502
$ ./hw0503
$ ./hw0504 [option]
$ ./hw0505

※注意

- 請安裝 **Base64**，以便第四題的使用 (signature 要使用)
- 請安裝 **curl7.70** 版本，此外務必安裝 **libssl-dev**，否則無法使用 https 的網址(第四題)，**請勿安裝 7.63 版本，否則 Makefile 無法編譯**
- 請安裝 grep(版本 3.4), awk(版本 5.0.1)
- 請安裝 **ifconfig**，**只接受** wiki 上的格式，以及最新的格式：
- 請配備 **dirent.h**

(1)最新格式[ifconfig] (版本：net-tools 2.10-alpha)

```
yuchen0515@mathlin:~/Programming/hw05$ ifconfig
enp2s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 30:65:ec:bc:2c:23  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2104  bytes 185516 (185.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2104  bytes 185516 (185.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.20  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::897e:c9d0:cd3f:8049  prefixlen 64  scopeid 0x20<link>
        ether 3c:a0:67:98:b6:2d  txqueuelen 1000  (Ethernet)
        RX packets 35091  bytes 42232473 (42.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 25653  bytes 4915676 (4.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

(2)

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:92:c7:3b
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::8039:3a0d:2406:8667/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1878 errors:0 dropped:0 overruns:0 frame:0
          TX packets:914 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1520198 (1.5 MB)  TX bytes:65083 (65.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:60 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4984 (4.9 KB)  TX bytes:4984 (4.9 KB)
```

四、索引

說明

1 Base64 (25 pts)

In computer science, **Base64** is a group of binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation. The term Base64 originates from a specific MIME content transfer encoding. Each Base64 digit represents exactly 6 bits of data. Three 8-bit bytes (i.e., a total of 24 bits) can therefore be represented by four 6-bit Base64 digits.

The Base64 index table can be found in the following link.

`https://en.wikipedia.org/wiki/Base64`

Because Base64 is a six-bit encoding, and because the decoded values are divided into 8-bit octets on a modern computer, every four characters of Base64-encoded text represents three octets of unencoded text or data. This means that when the length of the unencoded input is not a multiple of three, the encoded output must have padding added so that its length is a multiple of four. The padding character is =, which indicates that no further bits are needed to fully encode the input. The padding character is not essential for decoding, since the number of missing bytes can be inferred from the length of the encoded text.

Now I want you to develop the following program.

```
./hw0501 [options]
    -e, --enc         Encode a file to a text file.
    -d, --dec         Decode a text file to a file.
    -o, --output      Output file name.
```

Note that all options' arguments are mandatory.

◎題意說明

→ 簡單來說，此題即為撰寫出同 base64 的功能，其中包含 encode, decode，而 base64 的原理為 讀入 3 bytes→共 24bits，接著每 6bits 區隔成四份，並依照 A-Z, a-z, 0-9, +\，此外若讀入的字元不足 24bits 則補 0 進來，並且補充的部份若 6bits 皆為 0 則 encode “=”。

※注意

1) 課堂上說過，**輸入不會刻意刁難**，參數必須符合我的格式

2) -e, -d 兩個參數只能輸入其中一個，根據題目所敘述，參數後方必定跟著一個參考(file name)

3)除了第 2 點的條件外，**務必要有-o** (即 output)，其後必須跟著一個檔名

◎輸出&輸出格式

編譯後，依據需求填寫參數

例如我想將 test.txt 的資料用 base64 處理，輸出為 test4.txt 則可輸入：

$ ./hw0501 -e test.txt -o test4.txt

若想從 test4.txt 中解密，並輸出到 test5.txt 中，則可輸入

$./hw0501 -d test4.txt -o test5.txt

→e, d 兩者只能選擇其一，而且必須要有，-o 務必要有，每個參數後面務必跟隨一個檔名，此外前者的檔名必須存在(沒有內容我要怎麼解密、加密呢！？)

→當然，你也可以將 e 用--enc 代替，-d 用--dec 代替，-o 用--output 代替(同題目敘述)

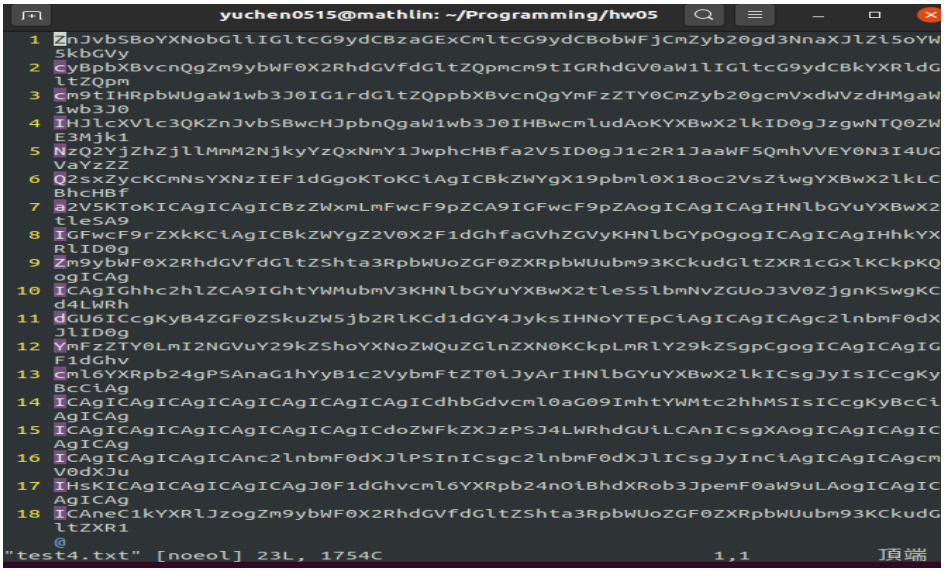
◎執行結果

#1

```
yuchen0515@mathlin:~/Programming/hw05$ ./hw0501 -e hw0503.c -o test4.txt
```

將 hw0503.c 內容 encode，輸出至 test4.txt 中

test4.txt 部份內容如下：

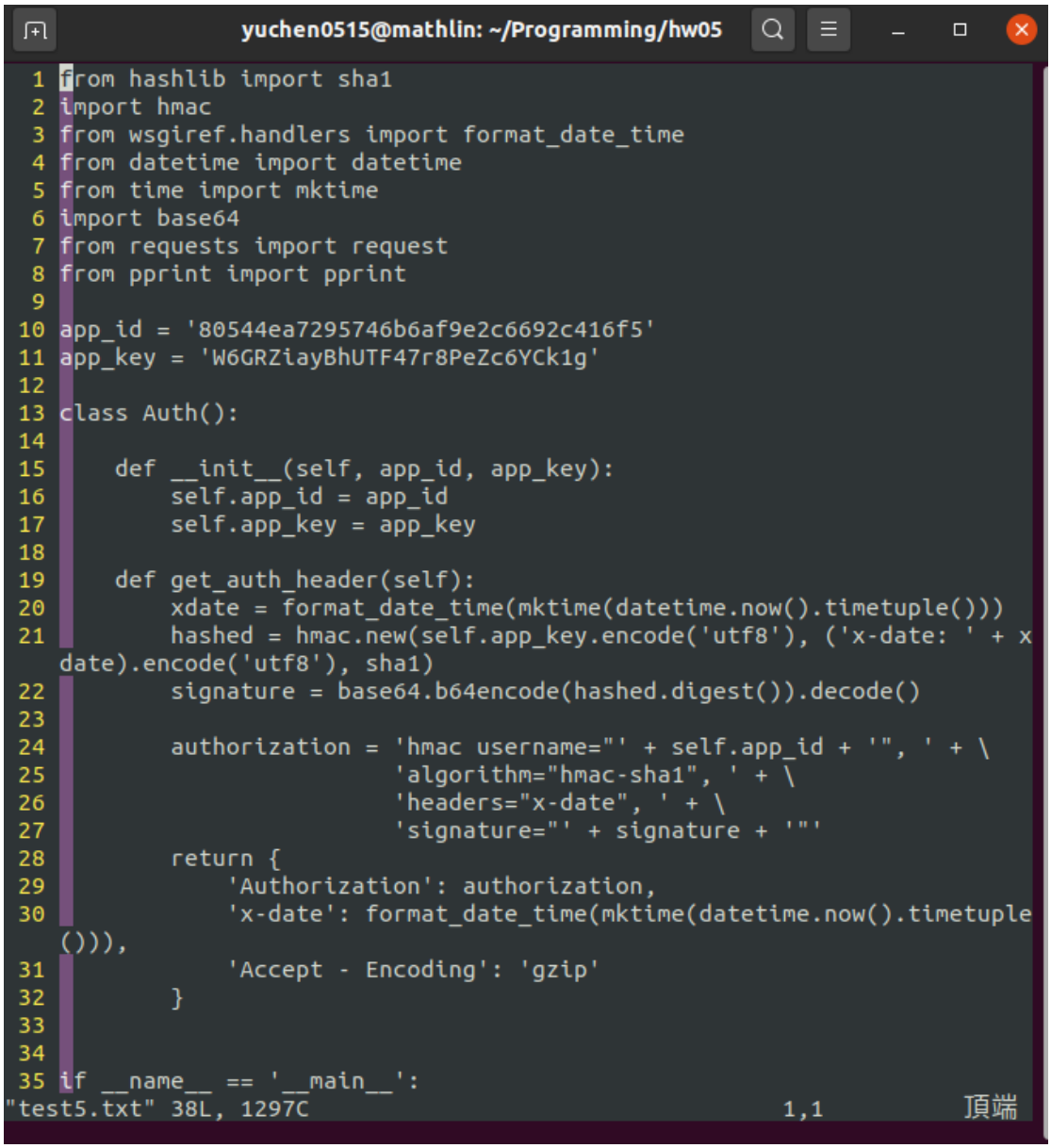


#2

```
yuchen0515@mathlin:~/Programming/hw05$ ./hw0501 -d test4.txt -o test5.txt
```

將剛才的 test4.txt 檔案解密至 test5.txt

test5.txt 內容如下(部份內容)：



(其實此為 第四題公開平台 python 版本的範例程式，在第四題中是參考此檔案寫出 curl 的 header，使用 c 語言取得資料的)

◎程式設計思路

1) 加密時，將檔案每 byte 的存入，存入滿 3bytes 後，接著以 6bits 分割，並對應到相對應的字元，接著再寫入

2) 倘若不足時，就直接補 0 即可

3) 解密時，將檔案每 byte 存入(即存入 char)，先將其對應表上相對位置取得後，再 6bits 拼裝，拼成 24bits 接著輸出即可

說明

2 Traffic Monitor (25 pts)

In this class, I have shown you how to use **popen**. This time, I want you to use this to develop a program that can continuously monitor the network traffic throughput. How to calculate the network throughput?

$$\frac{\text{TX/RX Bytes Difference}}{\text{Time Difference}} \text{ Mbps/Kbps/bps.}$$

Note that we use **bps**, which is bit per second, instead of **Bps**, which is Byte per second here. So you need to use popen to get the output of **ifconfig** and calculate the throughput. The program should be with an infinite loop. The user needs to use Ctrl+C to terminate it.

◎題意說明

→此題便是運用 ifconfig 指令，取得其中的總流量，並根據時間變化去計算出他的每秒流量，為一個無限迴圈，要終止必須使用 Ctrl+C

※注意

(1) 必須安裝 grep (版本 3.4), awk(版本 5.0.1)

(2) 務必安裝 **ifconfig**，**只接受** wiki 上的版本格式，以及最新的版本格式(版本：net-tools 2.10-alpha)，此外因應助教 6/13(六)23:45 公告，我是針對 20.04(ubuntu)進行製作，不過我的程式 20.04, 18.04, 16.04 三種版本都通用，為保險起見，以**版本越高的 ifconfig** 為主

(3) 題目提到必須從 Bps 改為 bps(Byte→bit)，雖然題目上使用 Kbps，也提供三種單位 Mbps/Kbps/bps，本想直接根據數值大小去調整單位，但會造成整個程式運行過程，每次執行單位會不同，反而造成不容易觀看、比較，因此在我的**程式中一律使用 bps**

(4) 本程式為 **2 秒執行一次**，輸出**仍為平均**(按照公式)，可放心

(5) 本題是計算網路流量，若一直為 0 代表你沒有連網，可試著開網頁、通訊軟體等網際網路工具，便會有正常數字了(0 也是正常數字，只是你沒使用到網路)

◎輸入&輸出格式

編譯後，執行”$ ./hw0502”

```
yuchen0515@mathlin:~/Programming/hw05$ ./hw0502
1)
wlp3s0: (192.168.43.20): TX: 0bps; RX: 0bps
lo: (127.0.0.1): TX: 3360bps; RX: 3360bps
2)
wlp3s0: (192.168.43.20): TX: 0bps; RX: 0bps
lo: (127.0.0.1): TX: 5216bps; RX: 5216bps
3)
wlp3s0: (192.168.43.20): TX: 0bps; RX: 0bps
lo: (127.0.0.1): TX: 3360bps; RX: 3360bps
4)
wlp3s0: (192.168.43.20): TX: 0bps; RX: 0bps
lo: (127.0.0.1): TX: 5216bps; RX: 5216bps
```

◎ifconfig 格式參考

(最新版本)

```
yuchen0515@mathlin:~/Programming/hw05$ ifconfig
enp2s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 30:65:ec:bc:2c:23  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2104  bytes 185516 (185.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2104  bytes 185516 (185.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.20  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::897e:c9d0:cd3f:8049  prefixlen 64  scopeid 0x20<link>
        ether 3c:a0:67:98:b6:2d  txqueuelen 1000  (Ethernet)
        RX packets 35091  bytes 42232473 (42.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 25653  bytes 4915676 (4.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

(助教版本)

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:92:c7:3b
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::8039:3a0d:2406:8667/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1878 errors:0 dropped:0 overruns:0 frame:0
          TX packets:914 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1520198 (1.5 MB)  TX bytes:65083 (65.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:60 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4984 (4.9 KB)  TX bytes:4984 (4.9 KB)
```

除了以上兩種格式以外，其餘一概不接受 (即 20.04, 18.04, 16.04)！
儘量使用越高版本的

◎程式設計思路

1) 單純使用 popen，並且用 grep 和 awk 取得我所需要的資料存入陣列內
2) 一個先前讀入的陣列，和 2 秒後讀入的陣列，接著兩個相減除以二即可

說明

3 Debug Messages (25 pts)

In this class, I have told you that a professional program developer should has his/her own debug function. I also told you sometimes we do not want to print too many debug messages or the important message will be covered by lots of unimportant messages. Now please develop a header file to support debugging functions.

```
debug_printf( level, fmt, ... )
```

There will be five levels. You need to define them in your header file. If the higher debug level is selected, all messages under this level will be printed. For example, if DEBUG_LEVEL_VVERBOSE is selected, the debug messages labeled with DEBUG_LEVEL_INFO will also be printed. If DEBUG_LEVEL_INFO is selected, the debug messages labeled with DEBUG_LEVEL_VERBOSE will **not** be printed. Of course, developers should not print any debug messages with DEBUG_LEVEL_NONE. This is only a configuration option that can turn off all debug messages.

◎題意說明

→課堂上有演示過如何使用 DEBUG_MESSAGE，今天想要你發展自己的 debug function，我們有時候不想印太多囉唆的 debug 資訊，以免真正重要的資訊被不重要的 messages 蓋掉了，今天請發展一個標頭檔支援 debugging functions

→本題中共有五個層級，你必須在 header file 中定義他們，如果使用者選擇叫高層低的 debug level，所有在這層級或以下的層級的 debug message 都應該被印出，反之若較高層級的則不能印出，此外你的 messages 應該印出 file name, function name, line number.

※注意

1) 若沒定義 level 的話，預設為 debug_message 不開啟，若使用 DEBUG_LEVEL_NONE 也是不開啟的效果

2) Makefile 有寫上六種編譯方式，助教可選擇需要的把 mark 刪除，不必要的 mark，就可以使用了

◎輸入&輸出格式

編譯時，輸入”$ gcc hw0503.c -D [DEBUG_LEVEL] -o hw0503”

執行”$ ./hw0503”

其中 DEBUG_LEVEL 的選項可以參考下表：

```
1 DEBUG_LEVEL_NONE
2 DEBUG_LEVEL_ERROR
3 DEBUG_LEVEL_INFO
4 DEBUG_LEVEL_VERBOSE
5 DEBUG_LEVEL_VVERBOSE
```

◎執行結果

#1 debug_level 不開啟，單純編譯的情況

```
yuchen0515@mathlin:~/Programming/hw05$ gcc hw0503.c -o hw0503
yuchen0515@mathlin:~/Programming/hw05$ ./hw0503
Hello, user!

------------------DEBUG MESSAGE OUTPUT----------------------



----------------------REVERSE OUTPUT-------------------------



------------------------------------------------------------

All message is output already above.
```

#2 DEBUG_LEVEL_NONE 開啟

```
yuchen0515@mathlin:~/Programming/hw05$ gcc -D DEBUG_LEVEL_NONE hw0503.c -o hw0503
yuchen0515@mathlin:~/Programming/hw05$ ./hw0503
Hello, user!

------------------DEBUG MESSAGE OUTPUT----------------------



----------------------REVERSE OUTPUT-------------------------



------------------------------------------------------------

All message is output already above.
```

#3 DEBUG_LEVEL_ERROR 開啟

```
yuchen0515@mathlin:~/Programming/hw05$ gcc -D DEBUG_LEVEL_ERROR hw0503.c -o hw0503
yuchen0515@mathlin:~/Programming/hw05$ ./hw0503
Hello, user!

------------------DEBUG MESSAGE OUTPUT----------------------

hw0503.c:17:main(): This is a LEVEL_ERROR message 1.


----------------------REVERSE OUTPUT------------------------

hw0503.c:6:test(): This is a LEVEL_ERROR message 1.


------------------------------------------------------------

All message is output already above.
```

#4 DEBUG_LEVEL_INFO 開啟

```
yuchen0515@mathlin:~/Programming/hw05$ gcc -D DEBUG_LEVEL_INFO hw0503.c -o hw0503
yuchen0515@mathlin:~/Programming/hw05$ ./hw0503
Hello, user!

------------------DEBUG MESSAGE OUTPUT----------------------

hw0503.c:17:main(): This is a LEVEL_ERROR message 1.
hw0503.c:19:main(): This is a LEVEL_INFO message 2.


----------------------REVERSE OUTPUT------------------------

hw0503.c:28:main(): This is a LEVEL_INFO message 2.
hw0503.c:6:test(): This is a LEVEL_ERROR message 1.


------------------------------------------------------------

All message is output already above.
```

#5 DEBUG_LEVEL_VERBOSE 開啟

```
yuchen0515@mathlin:~/Programming/hw05$ gcc -D DEBUG_LEVEL_VERBOSE hw0503.c -o hw0503
yuchen0515@mathlin:~/Programming/hw05$ ./hw0503
Hello, user!

------------------DEBUG MESSAGE OUTPUT----------------------

hw0503.c:17:main(): This is a LEVEL_ERROR message 1.
hw0503.c:19:main(): This is a LEVEL_INFO message 2.
hw0503.c:21:main(): This is a LEVEL_VERBOSE message 3.


----------------------REVERSE OUTPUT------------------------

hw0503.c:27:main(): This is a LEVEL_VERBOSE message 3.
hw0503.c:28:main(): This is a LEVEL_INFO message 2.
hw0503.c:6:test(): This is a LEVEL_ERROR message 1.


------------------------------------------------------------

All message is output already above.
```

#6 DEBUG_LEVEL_VVERBOSE 開啟

```
yuchen0515@mathlin:~/Programming/hw05$ gcc -D DEBUG_LEVEL_VVERBOSE hw0503.c -o hw0503
yuchen0515@mathlin:~/Programming/hw05$ ./hw0503
Hello, user!

------------------DEBUG MESSAGE OUTPUT----------------------

hw0503.c:17:main(): This is a LEVEL_ERROR message 1.
hw0503.c:19:main(): This is a LEVEL_INFO message 2.
hw0503.c:21:main(): This is a LEVEL_VERBOSE message 3.
hw0503.c:23:main(): This is a LEVEL_VVERBOSE message 4.


----------------------REVERSE OUTPUT------------------------

hw0503.c:26:main(): This is a LEVEL_VVERBOSE message 4.
hw0503.c:27:main(): This is a LEVEL_VERBOSE message 3.
hw0503.c:28:main(): This is a LEVEL_INFO message 2.
hw0503.c:6:test(): This is a LEVEL_ERROR message 1.


------------------------------------------------------------

All message is output already above.
```

◎程式設計思路

1) 將使用者的 LEVEL 善用 ifdef 去定義 LEVEL 的數值，接著把所有層級 define 成數字

2) 利用巨集，使用 do, while 語法將小於等於 LEVEL 的 debug_print 展開

3) 在本程式中，亦塞入一個 test()函式，可看到 line, function 是在哪裡發生的

說明

4 Airport Information (25 pts)

In this class, I have shown you how to use **curl** in C. Now I want you to use this library to get airport information. PTX is a platform which is to effectively satisfy the increasing demand for the information of public transport, including the needs of cross-modes, cross-regions and cross-agencies. It provides lots of application programming interfaces (APIs) for developers to get transportation information. PTX APIs are Web APIs. So you can get information through URLs. This time, I want you to get flight arrival information. Before development, you can use the following command to get the information.

```
curl "https://ptx.transportdata.tw/MOTC/v2/Air/FIDS/Airport/Arrival"
```

So many data, right? I want you to list information in the given format in some different orders.

◎題意說明

→此題即使用 libcurl 去存取 PTX 的 opendata 機場資訊，其中特別的是此題的網站必須使用 API key, API ID，且 opendata 的 URL 為 https 開頭，因此必須使用到 libssl-dev 才可以，存取資訊後並提供使用者使用參數-t,-f,-a 按照要求的欄位排序。

※注意

1) 先安裝 **curl-7-70** 版本，勿安裝任何其它版本，若欲安裝其它版本，請自行修改 Makefile 中的版本資訊，否則無法編譯(檔案中有提供安裝好的 7.70 版本)
2) 此題主要使用 hw0504_2.c 檔案用 libcurl 得到機場資訊，另使用 hw0504.c 負責接收參數，並 popen hw0504_2 從裡面分析、排序資料並輸出
3) 務必**安裝 openssl, libssl-dev**，才能處理 https 開頭的網址內容
4) 務必要設置參數，而且只能輸入一個(為 t, f, a)
5) 請注意，**此題排序時，是依照「字典序」排序的**，因此在 Flight number 排序中，我並不是按照數字大小排序，而是字典序！(因其它的 ScheduleArrivalTime 和 Airline ID 基本上都是字串型式，依字典序排序會比較有一致性)
6) **請連網**，否則無法存取資料
7) **已將所有資訊不完全的航班過濾掉**(有的雖有資訊但航班號碼之類的卻是空號，可能是有特殊情況的班次，因此選擇過濾掉)

◎輸入&輸出格式

編譯後，執行”$ ./hw0504 [options]”

其後應輸入 option (-t, --arrival-time／-f, --flight-number／-a, --airline-id)

```
./hw0504 [options]
    -t, --arrival-time        Sorting based on the ScheduleArrivalTime.
    -f, --flight-number       Sorting based on the Flight number.
    -a, --airline-id          Sorting based on the Airline ID.
2020-05-14T23:35 PEK-->TPE CI 518
2020-05-14T23:40 TNA-->TPE SC 103
...
```

◎執行結果

#1 按照 **ScheduleArrivalTime** 依**字典序**排序 [部份內容]

```
yuchen0515@mathlin:~/Programming/hw05$ ./hw0504 --arrival-time
2020-06-12T00:05 HKG-->TPE CI 179
2020-06-12T00:10 KIX-->TPE BR 4933
2020-06-12T00:10 KIX-->TPE NZ 9449
2020-06-12T00:10 KIX-->TPE TK 6361
2020-06-12T00:10 KIX-->TPE TG 5841
2020-06-12T00:10 KIX-->TPE NH 394
2020-06-12T00:20 DMK-->TPE SL 922
2020-06-12T00:30 HKG-->TPE CI 261
2020-06-12T00:35 PVG-->TPE CK 284
2020-06-12T00:40 HKG-->TPE HX 2896
2020-06-12T00:40 HKG-->TPE BR 310
2020-06-12T00:55 MNL-->TPE 5J 5255
2020-06-12T00:55 ANC-->TPE CI 1091
2020-06-12T01:00 PVG-->TPE CA 7243
2020-06-12T01:05 NRT-->TPE 5Y 568
2020-06-12T01:05 HKG-->TPE RH 7243
2020-06-12T01:05 NRT-->TPE KZ 6312
2020-06-12T01:05 XMN-->TPE BR 5870
```

#2 按照 **Flight number** 依**字典序**排序 [部份內容]

```
yuchen0515@mathlin:~/Programming/hw05$ ./hw0504 --flight-number
2020-06-12T12:25 HNL-->TPE CI 1
2020-06-13T20:10 HNL-->TPE CI 1
2020-06-14T12:25 HNL-->TPE CI 1005
2020-06-12T11:10 PVG-->TPE MU 1005
2020-06-14T11:10 PVG-->TPE MU 101
2020-06-12T15:20 TAO-->TPE B7 101
2020-06-12T15:20 TAO-->TPE SC 101
2020-06-12T17:20 NRT-->TPE CI 101
2020-06-12T22:40 FUK-->TPE BR 101
2020-06-13T17:20 NRT-->TPE CI 101
2020-06-13T20:30 FUK-->TPE BR 101
2020-06-14T17:20 NRT-->TPE CI 101
2020-06-14T22:40 FUK-->TPE BR 103
2020-06-14T23:40 TNA-->TPE SC 103
2020-06-13T15:20 NRT-->KHH CI 105
2020-06-12T13:45 FUK-->TPE BR 105
2020-06-12T21:10 NRT-->TPE CI 105
2020-06-12T23:10 DLC-->TPE B7 105
2020-06-13T13:45 FUK-->TPE BR 105
2020-06-13T21:10 NRT-->TPE CI 105
2020-06-14T13:45 FUK-->TPE BR 105
2020-06-14T21:10 NRT-->TPE CI 1055
```

#3 按照 **Airline ID** 依**字典序**排序 [部份內容]

```
yuchen0515@mathlin:~/Programming/hw05$ ./hw0504 --airline-id
2020-06-12T18:10 KMG-->TPE 3U 8977
2020-06-13T16:45 CTU-->TSA 3U 8979
2020-06-13T19:15 CKG-->TSA 3U 310
2020-06-12T00:55 MNL-->TPE 5J 312
2020-06-12T09:10 MNL-->TPE 5J 310
2020-06-13T00:55 MNL-->TPE 5J 312
2020-06-13T09:10 MNL-->TPE 5J 310
2020-06-14T00:55 MNL-->TPE 5J 312
2020-06-14T09:10 MNL-->TPE 5J 61
2020-06-12T17:55 HKG-->TPE 5X 26
2020-06-14T00:01 ICN-->TPE 5X 59
2020-06-14T01:20 HKG-->TPE 5X 64
2020-06-14T15:50 ICN-->TPE 5X 7243
2020-06-12T01:05 NRT-->TPE 5Y 7243
2020-06-14T23:00 NRT-->TPE 5Y 4501
2020-06-13T15:10 ICN-->KHH 7C 8951
2020-06-12T10:10 PVG-->TPE 9C 8951
2020-06-14T10:10 PVG-->TPE 9C 8458
2020-06-13T15:15 NRT-->TPE AA 8459
2020-06-13T14:55 NRT-->KHH AA 17
```

◎程式設計思路

1) 按照官方範例的 shell 檔去寫入 libcurl 的 header 內，並且從 hw0504 中拿取參數，直接輸出內容 (此動作寫在 hw0504_2.c 內)

2) hw0504.c 則負責 popen hw0504_2 的內容，將其內容字串處理，取出所需的資料，其中最特別的是官方有提供「網址 sort」的方式，依靠這個 sort

3) 主要癥結點都在於 curl 的 header, signature, date 必須寫好，並且 libcurl 必須安裝好，https 格式的網址則務必要安裝 libcurl-dev 否則無法使用

4) 若 3 的動作沒做好，卻可以成功存取資料，想必此程式一定是使用 python, ruby 等其它語言存取的，還請助教多多留意

說明

5 Bonus: dirent.h (5 pts)

dirent.h is a C POSIX library. That is, it is not a standard header file. Please study it and use its functions to list all files and directories in the given directory.

```
./hw0505 /etc
...
```

◎題意說明

→dirent.h 是一個 C POSIX 的 library (這代表他不是標準的函式庫)，請研究他然後使用這個東西列印出目錄下的所有檔案(使用者可指定目錄)

※注意

1) 助教的 linux 中，必須要有 dirent.h 檔案(基本上都會內建)

2) 使用方式如題目敘述，像是 $./hw0505 ./

3) 只包含一**個參數**(目錄)

4) 對於 list all files and directories in the given directory 的解讀，應是**列出使用者給定的目錄下的檔案以及目錄，即類似 ls 的效果**，且應只需印出檔名，沒其它特殊要求

◎執行結果

#1

列印出/etc 中的檔案

```
yuchen0515@mathlin:~/Programming/hw05$ ./hw0505 /etc
        DIR FILENAME
1)      .
2)      ..
3)      inputrc
4)      vmware-vix
5)      console-setup
6)      hdparm.conf
7)      brltty
8)      depmod.d
9)      debconf.conf
10)     passwd
11)     mke2fs.conf
12)     gai.conf
13)     kernel
14)     anthy
15)     samba
16)     skel
17)     polkit-1
18)     grub.d
19)     mtab
20)     libnl-3
21)     environment.d
22)     shadow
```

#2

列印出前一層目錄的檔案

```
yuchen0515@mathlin:~/Programming/hw05$ ./hw0505 ../
        DIR FILENAME
1)      hw05
2)      cJSON-master.zip
3)      hw03
4)      .
5)      lib
6)      ..
7)      project
8)      hw04
```

#3

列印出作業目錄的所有檔案(此為寫作業時的檔案，正式檔案會刪掉一些)

```
yuchen0515@mathlin:~/Programming/hw05$ ./hw0505 ./
        DIR FILENAME
1)      hw0501
2)      hw0504
3)      hw0505.c
4)      debug
5)      hw0504.c
6)      .~lock.README.odt#
7)      hw0503
8)      20.04_ifconfig
9)      .
10)     hw0502_2.c
11)     hw0503_header.h
12)     hw05.pdf
13)     hw0502
14)     其它
15)     Makefile
16)     hw0505
17)     hw0502.c
18)     ..
19)     hw0501.c
20)     hw0503.c
21)     hw0502_3.c
```