4. Where is the header file?

聽說從前從前有個東西叫作"Include Path"，去看看好了

隨意用以下指令編譯隨意c source file會得到:

```
gcc -v -std=c11 -O2 -Wall -Wextra -o main main.c
```

以下僅擷取關鍵區段

```
#include "..." search starts here:
#include <...> search starts here:
 /usr/lib/gcc/x86_64-linux-gnu/9/include
 /usr/local/include
 /usr/include/x86_64-linux-gnu
 /usr/include
End of search list.
```

所以就去看看這些目錄吧

```
ls /usr/lib/gcc/x86_64-linux-gnu/9/include \
                          /usr/local/include \
                          /usr/include/x86_64-linux-gnu \
                          /usr/include | less
```

使用`less`的搜尋功能後，就可以找到他出現在**`/usr/include`**的路徑之中

在刪除之前，先試試看把`#include <stdio.h>`從原始碼中刪掉看會怎樣。

`gcc`:

```
main.c: In function ‘main’:
main.c:4:5: warning: implicit declaration of function ‘printf’ [-Wimplicit-
function-declaration]
    4 |     printf("Hello World\n");
      |     ^~~~~~
main.c:4:5: warning: incompatible implicit declaration of built-in function
‘printf’
main.c:1:1: note: include ‘<stdio.h>’ or provide a declaration of ‘printf’
  +++ |+#include <stdio.h>
    1 | //#include <stdio.h>
```

這太天理不容了吧，~~毒瘤~~gcc，看我用clang小天使制裁你(X

`clang`:

```
main.c:4:5: warning: implicitly declaring library function 'printf' with type
'int (const char *, ...)' [-Wimplicit-function-declaration]
    printf("Hello World\n");
    ^
main.c:4:5: note: include the header <stdio.h> or explicitly provide a
declaration for 'printf'
1 warning generated.
```

哇，小天使也變惰天使了(X
看來沒辦法了，只能刪掉~~萬惡~~的`/usr/include/stdio.h`了

```
mv stdio.h stdio.h.disabled
```

```
gcc -std=c11 -O2 -Wall -Wextra -o main main.c
```

```
main.c:1:10: fatal error: stdio.h: 沒有此一檔案或目錄
    1 | #include <stdio.h>
      |          ^~~~~~~~~
compilation terminated.
```

哈哈哈哈，看你怎麼編譯 (?

是說，若不include stdio.h，反而可以再沒有stdio.h的狀況下編譯OAO，*怕*

小結

- `stdio.h`在Include Path的`/usr/include/`的資料夾中
- 刪掉之後基本上include stdio.h就會報編譯錯誤

先不說了，我先去拯救我的電腦了(#

5. Decimal to Hex

在計算機中，負數以2-Complement的方式表示。

規則為，正數不變，負數以其對應正數(也就是絕對值)全部位元反轉(not)後再+1
比如:

`1`: `00000000000000000000000000000001`(`0x1`)

`-1`: `11111111111111111111111111111110` + `1` =
`11111111111111111111111111111111`(`0xffffffff`)

或是

`2147483647`: `01111111111111111111111111111111`(`0x7fffffff`)

`-2147483647`: `10000000000000000000000000000000` + `1` =
`10000000000000000000000000000001`(`0x80000001`)

6. Bonus: Where is my `cd`

Linux

P.S. 預設shell已改為`fish`

```
which cd
```

哇，什麼都沒講，還回傳非0回傳值。在試過各種shell都未果QQ

STFG後，找到此資料，原來是shell內建的指令阿，which抓不到

但可以用`type`來調戲`cd`

```
type cd
```

然後就得到了`fish`在執行`cd`時的shell script. (以fish的講法要叫作function)

其他的shell(`sh`, `bash`等)都說「cd 是 shell 內建」，差別只是有沒有翻譯

Mac OS X

P.S. 預設shell依舊是`fish`

```
which cd
```

居然沒有報錯，還給我

```
/usr/bin/cd
```

嚇死寶寶了
趕緊

```
ls -lah /usr/bin/cd
```

```
-rwxr-xr-x 15 root wheel 190B 6 6 08:42 /usr/bin/cd*
```

哇，居然不是symlink居然是執行檔ㄟ，印出來看看好了

裡面是個shell script

```
#!/bin/sh
# $FreeBSD: src/usr.bin/alias/generic.sh,v 1.2 2005/10/24 22:32:19 cperciva Exp $
# This file is in the public domain.
builtin `echo ${0##*/} | tr \[:upper:] \[:lower:]` ${1+"$@"}
```

雖然沒看的很懂，不過看起來不用reverse engineering真是皆大歡喜，

在`sh`跟`bash`中測試結果同上，`zsh`會告訴你`cd: shell built-in command`

小結

- 基本上cd是**shell的內建指令**，不是一個執行檔，畢竟他的作用也不是真的「執行」什麼，只是改變shell運行時的work directory