https://pixabay.com/vectors/magnifying-glass-magnifier-glass-189254/
https://zh.wikipedia.org/wiki/Linux

http://wap.yesky.com/gameonline/405/11790405.shtml

作業八:
從使用者模式追蹤到核心模式

中正大學 作業系統實驗室
指導大叔: 羅習五

作業目標及負責助教

- 作業目標：
  - 了解作業系統對上層軟體提供的進入點（錯誤、system call、breakpoint)
  - 與作業三、四合併在一起，進一步的了解system call從user space到kernel space如何運作。

撰寫程式碼或者直接使用

- 只要能夠呼叫 system call
- 一定要知道呼叫system call時的address
- 因為屆時會放到『Linux in QEMU』去跑，使用靜態編譯
  - gcc –g --static

程式碼

```
1.  void call_sys() {
2.      char* hello_tc = "全世界，你好\n";
3.      long len_tc = strlen(hello_tc); //注意我宣告為long，因為long是64位元
4.      long ret;
5.      __asm__ volatile (
6.          "mov $1, %%rax\n"   //system call number
7.          "mov $2, %%rdi\n"   //stderr
8.          "mov %1, %%rsi\n"   //
9.          "mov %2, %%rdx\n"
10.         "syscall\n"
11.         "mov %%rax, %0"
12.         :  "=m"(ret)
13.         : "g" (hello_tc), "g" (len_tc)
14.         : "rax", "rbx", "rcx", "rdx");
15. }
16. int main() {
17.     printf("%p\n", call_sys);
18.     getchar();
19.     call_sys();
20. }
```

追蹤system call的流程

- 在『Linux in QEMU - debug』中執行前一頁的程式碼
- 讓『Linux in QEMU - debug』完成開機
- 執行應用程式
- 設定中斷點在call_sys
- 使用「si」的方式追蹤
- 可以在gdb中使用file切換「執行檔」
  - 我們追蹤的程式有二個，分別是call_sys，及Linux kernel，因此需要切換

作業繳交

- 設定中斷點在test_syscall發出system call之前，請在這個地方截圖
- 使用單步追蹤（si），直到Linux kernel，請在進入Linux kernel時截圖
- 請說明Linux kernel如何用RAX暫存器判斷要呼叫哪個Linux內部的函數
- 請大致說明作業系統如何處理write。
- 將上述的東西寫成文件，並以「pdf」的方式繳交，詳細繳交方式由助教公布