[作業系統概論 HW4]

409410025 邱 X 恩

Q1 ：將你的程式碼反組譯，然後「大致」解釋組語的意義

A1 ：使用 gdb 內的 disass /m main，反組譯結果說明如下。

```
(gdb) disass /m main
Dump of assembler code for function main:
7       {
   0x0000000000401bfd <+0>:     push   %rbp
   0x0000000000401bfe <+1>:     mov    %rsp,%rbp
   0x0000000000401c01 <+4>:     sub    $0x30,%rsp
   0x0000000000401c05 <+8>:     mov    %edi,-0x24(%rbp)
   0x0000000000401c08 <+11>:    mov    %rsi,-0x30(%rbp)
   0x0000000000401c0c <+15>:    mov    %fs:0x28,%rax
   0x0000000000401c15 <+24>:    mov    %rax,-0x8(%rbp)
   0x0000000000401c19 <+28>:    xor    %eax,%eax

8           char buffer[2] ; // only need one character
9           long len_tc = 1 ; // define long because long is 64 bits
   0x0000000000401c1b <+30>:    movq   $0x1,-0x18(%rbp)
```

- 可以看到上面有 push rbp 和 mov rsp rbp 的指令
  主要是先 push main 函數到 stack 記憶體位置的頂端
  再來把 rsp 複製到 rbp ，讓 rsp 會指向 rbp。
  而接下來有一連串的 mov 指令
  可以推測他是在做整個程是一開始所需要的 register 的準備

```
10          long ret ;
11          printf("使用 'syscall' 呼叫system call\n");
   0x0000000000401c23 <+38>:    lea    0x933de(%rip),%rdi        # 0x495008
   0x0000000000401c2a <+45>:    callq  0x4185c0 <puts>

12          __asm__ volatile (
   0x0000000000401c2f <+50>:    lea    -0xa(%rbp),%rcx
   0x0000000000401c33 <+54>:    mov    $0x0,%rax
   0x0000000000401c3a <+61>:    mov    $0x0,%rdi
   0x0000000000401c41 <+68>:    mov    %rcx,%rsi
   0x0000000000401c44 <+71>:    mov    -0x18(%rbp),%rdx
   0x0000000000401c48 <+75>:    syscall
   0x0000000000401c4a <+77>:    mov    %rax,-0x20(%rbp)
```

- 準備好暫存器後，接下來的重點就是呼叫 syscall 了。
  可以看到 12 行後有一個 ” syscall ” 。
  這邊的 syscall 功能就是呼叫 read。
  而 syscall 的上一行可以看到有 rdx 暫存器，這邊是用來
  把 len 放到 rdx 裡面，作為 read 的參數。

- 這邊也能夠看到我們寫的 inline assembly 中出現的 register
  rax 放 0 是因為 sytem call 的 read， rdi 放 0 是因為 stdin
  而-0x18(rbp)， %rdx 是把 buffer 的東西放入 rdx，作為 read 的參數。

```
14              "mov $0, %%rdi\n"    //stdin
15              "mov %1, %%rsi\n"    // buffer address
16              "mov %2, %%rdx\n"    // lenth
17              "syscall\n"
18              "mov %%rax, %0" // syscall return number
19              :   "=m"(ret)
20              : "g" (buffer), "g" (len_tc)
21              : "rax", "rdi", "rsi", "rdx");
22          printf("讀入的字元是 : %c\n",buffer[0]);
```

```
22          printf("讀入的字元是 : %c\n",buffer[0]);
   0x0000000000401c4e <+81>:    movzbl -0xa(%rbp),%eax
   0x0000000000401c52 <+85>:    movsbl %al,%eax
   0x0000000000401c55 <+88>:    mov    %eax,%esi
   0x0000000000401c57 <+90>:    lea    0x933cd(%rip),%rdi        # 0x49502b
   0x0000000000401c5e <+97>:    mov    $0x0,%eax
   0x0000000000401c63 <+102>:   callq  0x410900 <printf>

23          printf("回傳值是：%ld\n", ret);
   0x0000000000401c68 <+107>:   mov    -0x20(%rbp),%rax
   0x0000000000401c6c <+111>:   mov    %rax,%rsi
   0x0000000000401c6f <+114>:   lea    0x933ce(%rip),%rdi        # 0x495044
   0x0000000000401c76 <+121>:   mov    $0x0,%eax
   0x0000000000401c7b <+126>:   callq  0x410900 <printf>
   0x0000000000401c80 <+131>:   mov    $0x0,%eax

24      }
   0x0000000000401c85 <+136>:   mov    -0x8(%rbp),%rdx
   0x0000000000401c89 <+140>:   xor    %fs:0x28,%rdx
   0x0000000000401c92 <+149>:   je     0x401c99 <main+156>
   0x0000000000401c94 <+151>:   callq  0x4541a0 <__stack_chk_fail_local>
   0x0000000000401c99 <+156>:   leaveq
   0x0000000000401c9a <+157>:   retq
```

- 這邊是 movbl 的指令以下的部分，
  這些指令目的是把接下來要印出的東西，放到對的 register 裡面。
  可以看到這邊有一個 callq 0x410900 <printf>的指令，
  這邊就如同字面上一樣，是呼叫 printf 的意思。

```
22          printf("讀入的字元是 : %c\n",buffer[0]);
   0x0000000000401c4e <+81>:    movzbl -0xa(%rbp),%eax
   0x0000000000401c52 <+85>:    movsbl %al,%eax
   0x0000000000401c55 <+88>:    mov    %eax,%esi
   0x0000000000401c57 <+90>:    lea    0x933cd(%rip),%rdi        # 0x49502b
   0x0000000000401c5e <+97>:    mov    $0x0,%eax
   0x0000000000401c63 <+102>:   callq  0x410900 <printf>

23          printf("回傳值是：%ld\n", ret);
   0x0000000000401c68 <+107>:   mov    -0x20(%rbp),%rax
   0x0000000000401c6c <+111>:   mov    %rax,%rsi
   0x0000000000401c6f <+114>:   lea    0x933ce(%rip),%rdi        # 0x495044
   0x0000000000401c76 <+121>:   mov    $0x0,%eax
   0x0000000000401c7b <+126>:   callq  0x410900 <printf>
   0x0000000000401c80 <+131>:   mov    $0x0,%eax

24      }
   0x0000000000401c85 <+136>:   mov    -0x8(%rbp),%rdx
   0x0000000000401c89 <+140>:   xor    %fs:0x28,%rdx
   0x0000000000401c92 <+149>:   je     0x401c99 <main+156>
   0x0000000000401c94 <+151>:   callq  0x4541a0 <__stack_chk_fail_local>
   0x0000000000401c99 <+156>:   leaveq
   0x0000000000401c9a <+157>:   retq
```

- 此外，mov $0x0 %eax 指的是把 eax 歸零。
  把 eax 歸零的意義在於說，
  寫 main fuction 的時候，main 函示結束，
  最後會有一個 return 0 的意思。

- 最後可以看到有一個 leave q 的指令
  做的事情很簡單，就是程是一開始有 push rbp
  現在程式要結束了，就 pop 掉。