408410035 鄭 x 辰

為方便說明 以下是反組譯後結果

```
4       int main(int argc, char** argv) {
   0x0000000000401d35 <+0>:     endbr64
   0x0000000000401d39 <+4>:     push   %rbp
   0x0000000000401d3a <+5>:     mov    %rsp,%rbp
   0x0000000000401d3d <+8>:     push   %rbx
   0x0000000000401d3e <+9>:     sub    $0x38,%rsp
   0x0000000000401d42 <+13>:    mov    %edi,-0x34(%rbp)
```

此區塊把區域變數放進 stack 中

rbp 指向的是 stack 的底部

```
   0x0000000000401d45 <+16>:    mov    %rsi,-0x40(%rbp)
   0x0000000000401d49 <+20>:    mov    %fs:0x28,%rax
   0x0000000000401d52 <+29>:    mov    %rax,-0x18(%rbp)
   0x0000000000401d56 <+33>:    xor    %eax,%eax

5           long ret;
6           char ch;
7           long len=1;
   0x0000000000401d58 <+35>:    movq   $0x1,-0x20(%rbp)
```

把 1 放進 len 所在的位置

```
8           printf("使用 'syscall' 呼叫 system call\n");
   0x0000000000401d60 <+43>:    lea    0x932a1(%rip),%rdi        # 0x495008
```

準備 printf 的資料（實際呼叫 puts）

```
   0x0000000000401d67 <+50>:    callq  0x418920 <puts>
```

0x418920 是 puts 所在的位置

```
9           __asm__ volatile (
   0x0000000000401d6c <+55>:    lea    -0x29(%rbp),%rsi
```

這一段做我們行內組語的內容

```
   0x0000000000401d70 <+59>:    mov    $0x0,%rax
```

0 放進 rax

```
   0x0000000000401d77 <+66>:    mov    $0x0,%rdi
```

0 放進 rdi

```
   0x0000000000401d7e <+73>:    mov    %rsi,%rsi
```

什麼都沒做（實際上把&ch 放進 rdi）

```
   0x0000000000401d81 <+76>:    mov    -0x20(%rbp),%rdx
```

len 放進 rdx

```
   0x0000000000401d85 <+80>:    syscall
```

system call read

```
   0x0000000000401d87 <+82>:    mov    %rax,-0x28(%rbp)
```

把 return 值（rax）放進 ret 所在位置

```
10              "mov $0, %%rax\n"   //system call number
11              "mov $0, %%rdi\n"   //stderr
12              "mov %1, %%rsi\n"   //
13              "mov %2, %%rdx\n"
14              "syscall\n"
15              "mov %%rax, %0\n"
16              :  "=m"(ret)
17              : "g" (&ch), "g" (len)
18              : "rax", "rbx", "rcx", "rdx");
19          printf("回傳值是：%ld\n", ret);
```

```
   0x0000000000401d8b <+86>:    mov    -0x28(%rbp),%rax 把 ret 放進 rax 作為參數
   0x0000000000401d8f <+90>:    mov    %rax,%rsi
   0x0000000000401d92 <+93>:    lea    0x93292(%rip),%rdi        # 0x49502b
準備 printf 的資料
   0x0000000000401d99 <+100>:   mov    $0x0,%eax
   0x0000000000401d9e <+105>:   callq  0x410c60 <printf>0x418920 是 printf 所在位置

20          printf("讀入的字元為\" %c \"\n",ch);
   0x0000000000401da3 <+110>:   movzbl -0x29(%rbp),%eax
   0x0000000000401da7 <+114>:   movsbl %al,%eax
   0x0000000000401daa <+117>:   mov    %eax,%esi 這一塊也是準備 printf 資料並呼叫
   0x0000000000401dac <+119>:   lea    0x9328c(%rip),%rdi        # 0x49503f
   0x0000000000401db3 <+126>:   mov    $0x0,%eax
   0x0000000000401db8 <+131>:   callq  0x410c60 <printf>
   0x0000000000401dbd <+136>:   mov    $0x0,%eax

21      }
   0x0000000000401dc2 <+141>:   mov    -0x18(%rbp),%rdx
   0x0000000000401dc6 <+145>:   xor    %fs:0x28,%rdx
   0x0000000000401dcf <+154>:   je     0x401dd6 <main+161>
   0x0000000000401dd1 <+156>:   callq  0x4544f0 <__stack_chk_fail_local>
   0x0000000000401dd6 <+161>:   add    $0x38,%rsp
   0x0000000000401dda <+165>:   pop    %rbx
   0x0000000000401ddb <+166>:   pop    %rbp
   0x0000000000401ddc <+167>:   retq
```