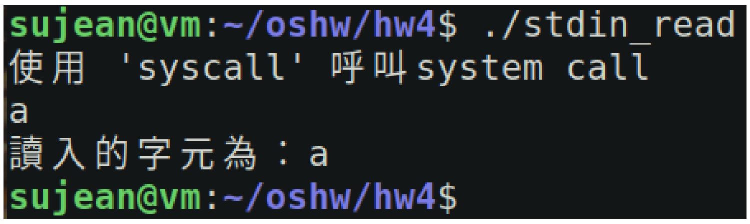
HW4 系統呼叫

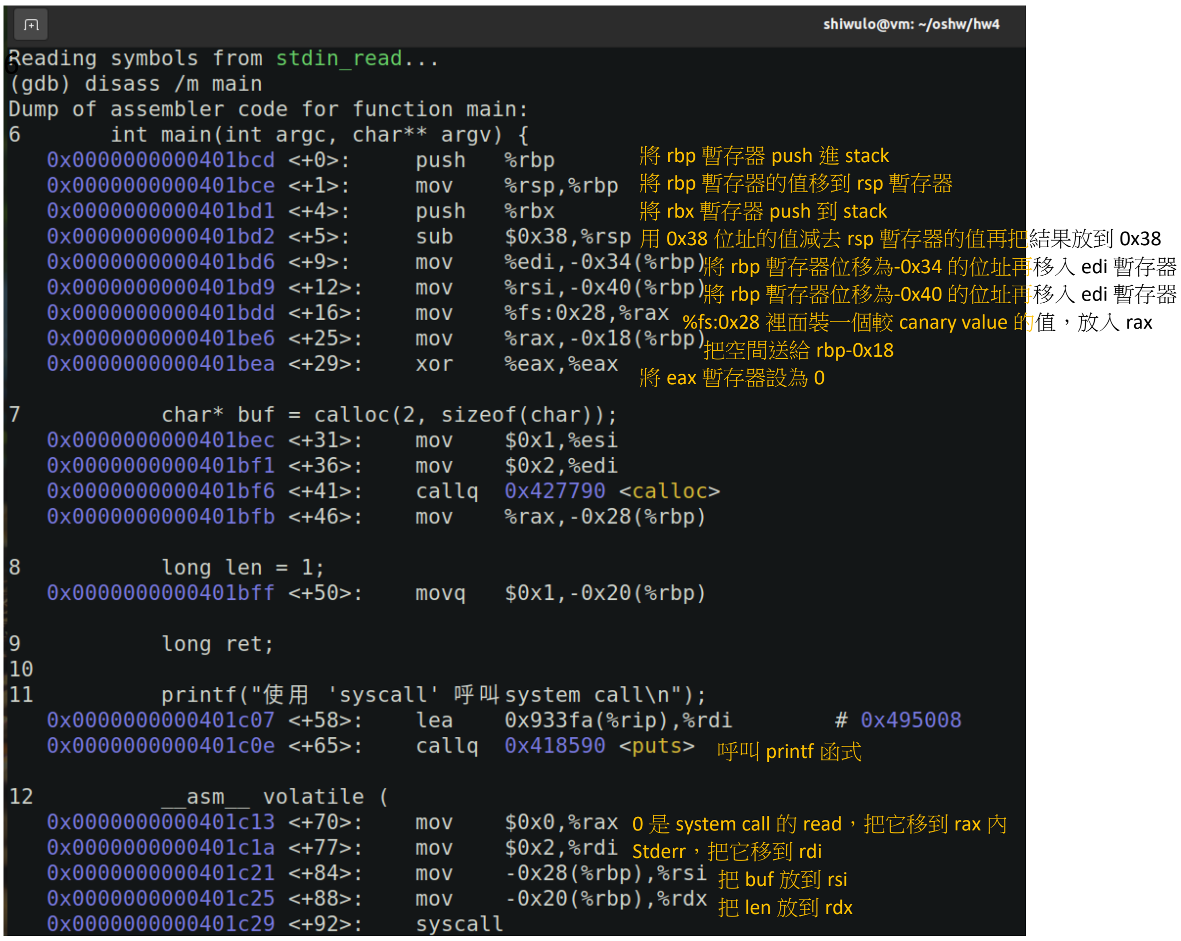
系級：資工二　學號：409410114　姓名：周述君

1. 撰寫一支程式名為：「stdin_read」，在這個程式中使用組合語言呼叫 systemcall，從 stdin 讀進一個字元，假設讀入的字元為 a，隨後使用 printf 在螢幕上印出『讀入的字元為“a”』



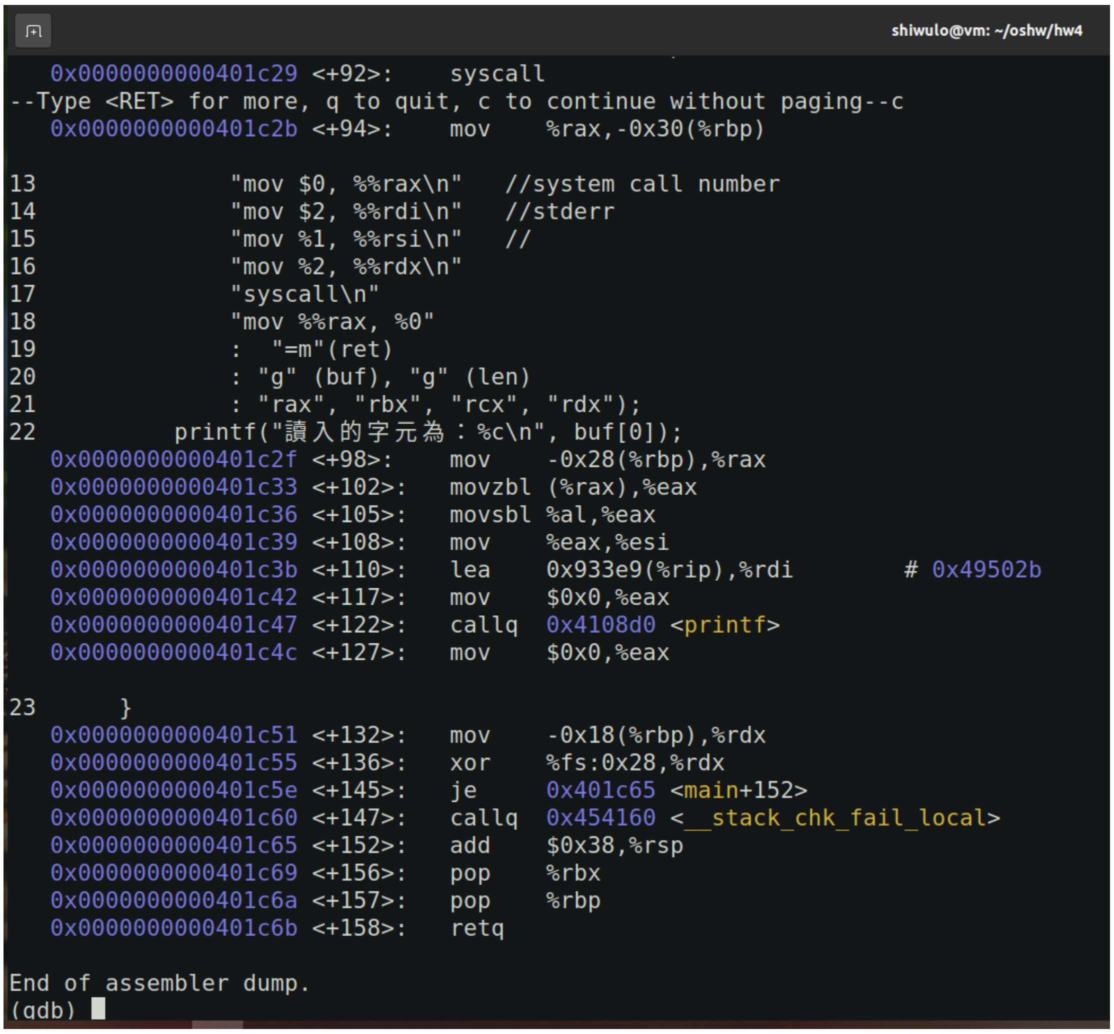
```
sujean@vm:~/oshw/hw4$ ./stdin_read
使用 'syscall' 呼叫system call
a
讀入的字元為：a
sujean@vm:~/oshw/hw4$
```

2. 使用 gdb 內的 disass /m main 所產生的結果：



```
Reading symbols from stdin_read...
(gdb) disass /m main
Dump of assembler code for function main:
6       int main(int argc, char** argv) {
   0x0000000000401bcd <+0>:     push   %rbp            將 rbp 暫存器 push 進 stack
   0x0000000000401bce <+1>:     mov    %rsp,%rbp       將 rbp 暫存器的值移到 rsp 暫存器
   0x0000000000401bd1 <+4>:     push   %rbx            將 rbx 暫存器 push 到 stack
   0x0000000000401bd2 <+5>:     sub    $0x38,%rsp      用 0x38 位址的值減去 rsp 暫存器的值再把結果放到 0x38
   0x0000000000401bd6 <+9>:     mov    %edi,-0x34(%rbp)將 rbp 暫存器位移為-0x34 的位址再移入 edi 暫存器
   0x0000000000401bd9 <+12>:    mov    %rsi,-0x40(%rbp)將 rbp 暫存器位移為-0x40 的位址再移入 edi 暫存器
   0x0000000000401bdd <+16>:    mov    %fs:0x28,%rax   %fs:0x28 裡面裝一個較 canary value 的值，放入 rax
   0x0000000000401be6 <+25>:    mov    %rax,-0x18(%rbp)把空間送給 rbp-0x18
   0x0000000000401bea <+29>:    xor    %eax,%eax       將 eax 暫存器設為 0

7               char* buf = calloc(2, sizeof(char));
   0x0000000000401bec <+31>:    mov    $0x1,%esi
   0x0000000000401bf1 <+36>:    mov    $0x2,%edi
   0x0000000000401bf6 <+41>:    callq  0x427790 <calloc>
   0x0000000000401bfb <+46>:    mov    %rax,-0x28(%rbp)

8               long len = 1;
   0x0000000000401bff <+50>:    movq   $0x1,-0x20(%rbp)

9               long ret;
10
11              printf("使用 'syscall' 呼叫system call\n");
   0x0000000000401c07 <+58>:    lea    0x933fa(%rip),%rdi        # 0x495008
   0x0000000000401c0e <+65>:    callq  0x418590 <puts>   呼叫 printf 函式

12              __asm__ volatile (
   0x0000000000401c13 <+70>:    mov    $0x0,%rax  0 是 system call 的 read，把它移到 rax 內
   0x0000000000401c1a <+77>:    mov    $0x2,%rdi  Stderr，把它移到 rdi
   0x0000000000401c21 <+84>:    mov    -0x28(%rbp),%rsi  把 buf 放到 rsi
   0x0000000000401c25 <+88>:    mov    -0x20(%rbp),%rdx  把 len 放到 rdx
   0x0000000000401c29 <+92>:    syscall
```

6 和 7 主要在做前置的準備(stack 和暫存器內的資料)



shiwulo@vm: ~/oshw/hw4

```
   0x0000000000401c29 <+92>:    syscall
--Type <RET> for more, q to quit, c to continue without paging--c
   0x0000000000401c2b <+94>:    mov    %rax,-0x30(%rbp)

13                "mov $0, %%rax\n"   //system call number
14                "mov $2, %%rdi\n"   //stderr
15                "mov %1, %%rsi\n"   //
16                "mov %2, %%rdx\n"
17                "syscall\n"
18                "mov %%rax, %0"
19                :  "=m"(ret)
20                : "g" (buf), "g" (len)
21                : "rax", "rbx", "rcx", "rdx");
22            printf("讀入的字元為：%c\n", buf[0]);
   0x0000000000401c2f <+98>:    mov    -0x28(%rbp),%rax
   0x0000000000401c33 <+102>:   movzbl (%rax),%eax
   0x0000000000401c36 <+105>:   movsbl %al,%eax
   0x0000000000401c39 <+108>:   mov    %eax,%esi
   0x0000000000401c3b <+110>:   lea    0x933e9(%rip),%rdi        # 0x49502b
   0x0000000000401c42 <+117>:   mov    $0x0,%eax
   0x0000000000401c47 <+122>:   callq  0x4108d0 <printf>
   0x0000000000401c4c <+127>:   mov    $0x0,%eax

23        }
   0x0000000000401c51 <+132>:   mov    -0x18(%rbp),%rdx
   0x0000000000401c55 <+136>:   xor    %fs:0x28,%rdx
   0x0000000000401c5e <+145>:   je     0x401c65 <main+152>
   0x0000000000401c60 <+147>:   callq  0x454160 <__stack_chk_fail_local>
   0x0000000000401c65 <+152>:   add    $0x38,%rsp
   0x0000000000401c69 <+156>:   pop    %rbx
   0x0000000000401c6a <+157>:   pop    %rbp
   0x0000000000401c6b <+158>:   retq

End of assembler dump.
(gdb)
```

22 下面 mov 的指令再為 printf 做準備，要把印出來的放在對的暫存器內，等值移完再叫 callq 呼叫 printf。

23 下面的指令是在城市結束前把 stack 狀態恢復，再把暫存器恢復原本的值。