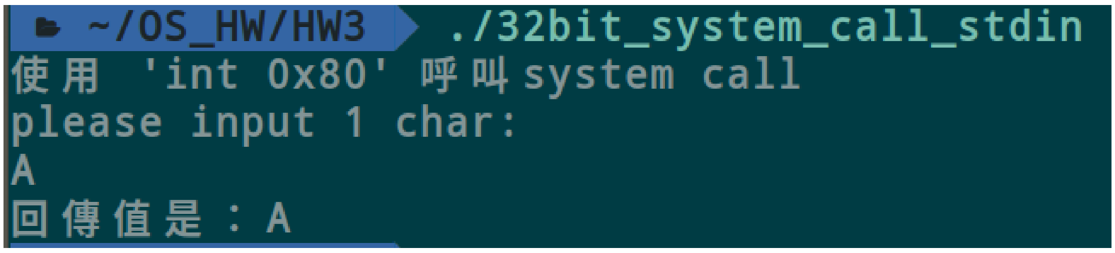
hw03.system call 406410114 資工三 郭晏誠

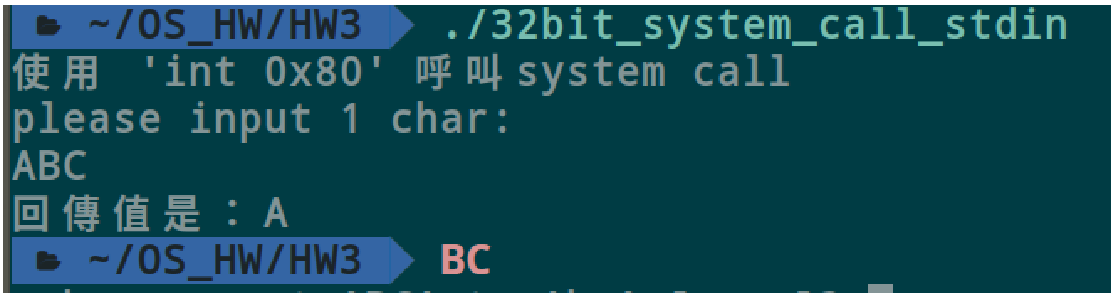
1. 使用組合語言呼叫 system call，從 stdin 讀進一個字元

(1)32bit

(i)輸出結果：只吃一個字元



```
 ~/OS_HW/HW3  ./32bit_system_call_stdin
使用 'int 0x80' 呼叫system call
please input 1 char:
A
回傳值是：A
```

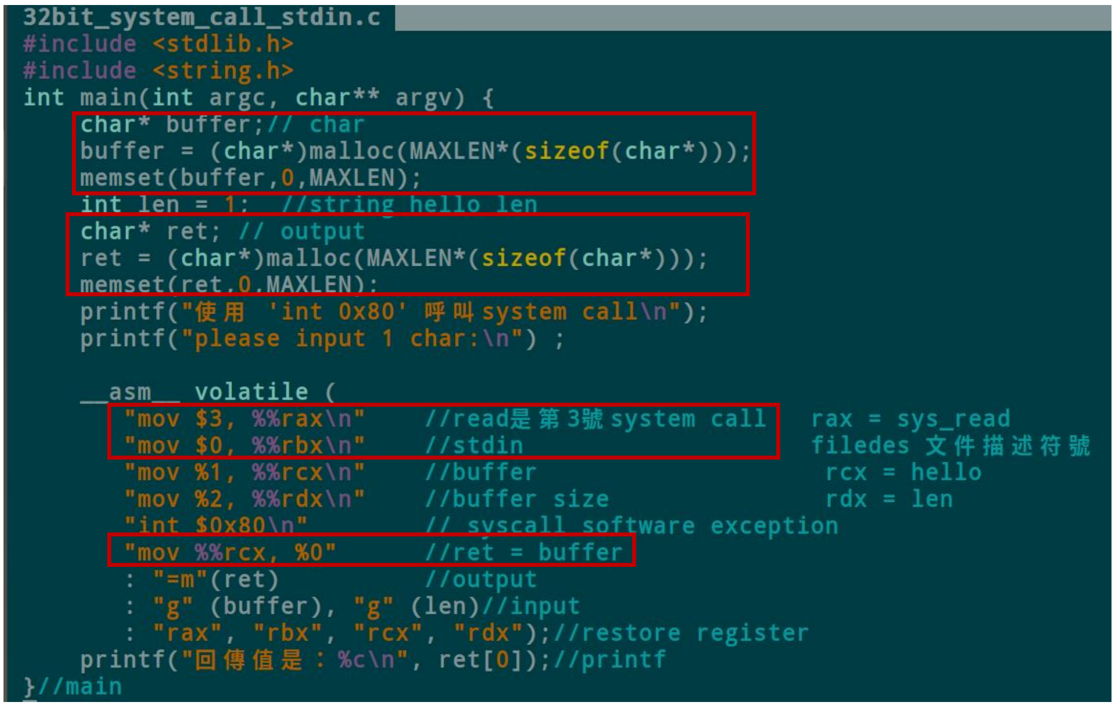


```
 ~/OS_HW/HW3  ./32bit_system_call_stdin
使用 'int 0x80' 呼叫system call
please input 1 char:
ABC
回傳值是：A
 ~/OS_HW/HW3  BC
```

(ii)修改使得 buffer 和 ret 有指定的記憶體

(iii)使用 3 號中斷 sys_read、並把參數改成 0 代表 stdin

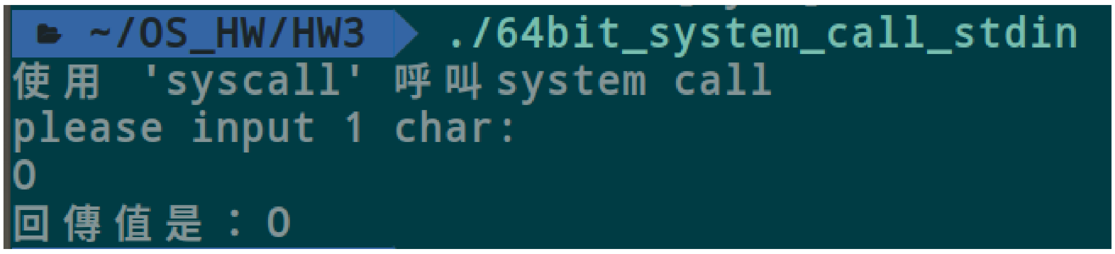
(iv)並且把 buffer read 進來的字串放到 ret



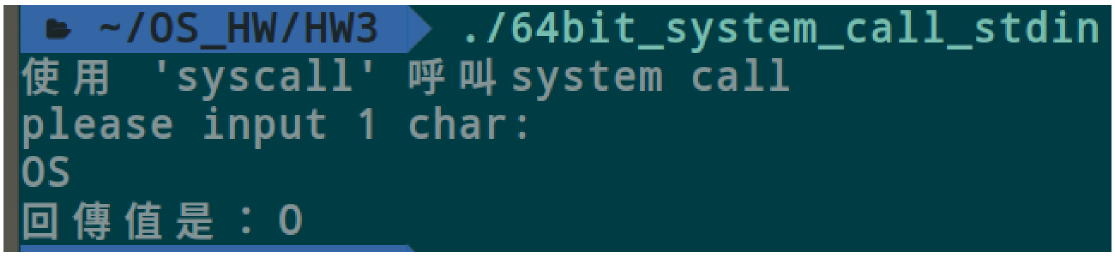
```
32bit_system_call_stdin.c
#include <stdlib.h>
#include <string.h>
int main(int argc, char** argv) {
    char* buffer;// char
    buffer = (char*)malloc(MAXLEN*(sizeof(char*)));
    memset(buffer,0,MAXLEN);
    int len = 1;  //string hello len
    char* ret; // output
    ret = (char*)malloc(MAXLEN*(sizeof(char*)));
    memset(ret,0,MAXLEN);
    printf("使用 'int 0x80' 呼叫system call\n");
    printf("please input 1 char:\n") ;

    __asm__ volatile (
       "mov $3, %%rax\n"    //read是第3號system call   rax = sys_read
       "mov $0, %%rbx\n"    //stdin                    filedes 文件描述符號
       "mov %1, %%rcx\n"    //buffer                    rcx = hello
       "mov %2, %%rdx\n"    //buffer size               rdx = len
       "int $0x80\n"        // syscall software exception
       "mov %%rcx, %0"      //ret = buffer
       : "=m"(ret)          //output
       : "g" (buffer), "g" (len)//input
       : "rax", "rbx", "rcx", "rdx");//restore register
    printf("回傳值是：%c\n", ret[0]);//printf
}//main
```

(2)64bit

(i)輸出結果：只吃一個字元



```
 ~/OS_HW/HW3  ./64bit_system_call_stdin
使用 'syscall' 呼叫 system call
please input 1 char:
O
回傳值是：O
```

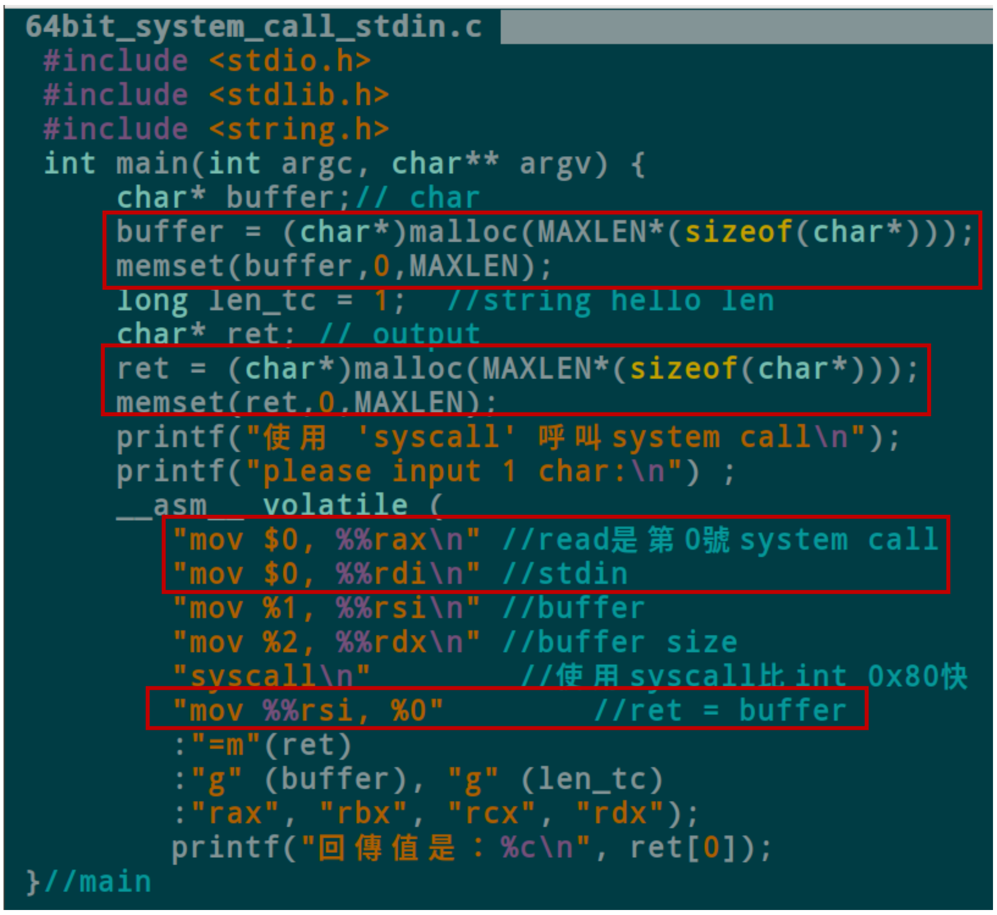


```
 ~/OS_HW/HW3  ./64bit_system_call_stdin
使用 'syscall' 呼叫 system call
please input 1 char:
OS
回傳值是：O
```

(ii)修改使得 buffer 和 ret 有指定的記憶體

(iii)使用 3 號中斷 sys_read、並把參數改成 0 代表 stdin

(iv)並且把 buffer read 進來的字串放到 ret



```
64bit_system_call_stdin.c
 #include <stdio.h>
 #include <stdlib.h>
 #include <string.h>
 int main(int argc, char** argv) {
     char* buffer;// char
     buffer = (char*)malloc(MAXLEN*(sizeof(char*)));
     memset(buffer,0,MAXLEN);
     long len_tc = 1;  //string hello len
     char* ret; // output
     ret = (char*)malloc(MAXLEN*(sizeof(char*)));
     memset(ret,0,MAXLEN);
     printf("使用 'syscall' 呼叫 system call\n");
     printf("please input 1 char:\n") ;
     __asm__ volatile (
        "mov $0, %%rax\n" //read是第0號 system call
        "mov $0, %%rdi\n" //stdin
        "mov %1, %%rsi\n" //buffer
        "mov %2, %%rdx\n" //buffer size
        "syscall\n"         //使用syscall比int 0x80快
        "mov %%rsi, %0"          //ret = buffer
        :"=m"(ret)
        :"g" (buffer), "g" (len_tc)
        :"rax", "rbx", "rcx", "rdx");
        printf("回傳值是：%c\n", ret[0]);
}//main
```

```
//youtube 筆記
32bit_system_call.c
 #include <stdio.h>
 #include <string.h>
 int main(int argc, char** argv) {
     char* hello = "hello world\n";// char
     int len = strlen(hello)+1;  //string hello len
     long ret; // output
     printf("使用 'int 0x80' 呼叫 system call\n");
     __asm__ volatile (
        "mov $4, %%rax\n"    //write 是第 4 號 system call
        "mov $2, %%rbx\n"    //stderr           filedes 文件描述符號
        "mov %1, %%rcx\n"    //buffer           rcx = hello
        "mov %2, %%rdx\n"    //buffer size      rdx = len
        "int $0x80\n"       //發出 system call exception
        //int $0x80 將系統調用號傳入 eax(this is 4 sys_write),
        //各個參數按照 ebx、ecx、edx 的順序傳遞到寄存器中，系統調用返
        回值儲存到 eax 寄存器。
        //ssize_t write(int fd, const void *buf, size_t count);

        "mov %%rax, %0"
        //system call 的回傳值放在 rax  ret = sys_write_return
        : "=m"(ret)
        : "g" (hello), "g" (len)
        : "rax", "rbx", "rcx", "rdx");
     printf("回傳值是：%ld\n", ret);
 }//main
```

64bit_system_call.c
```
 #include <stdio.h>
 #include <string.h>
 int main(int argc, char** argv) {
     char* hello_tc = "全世界，你好\n";
     long len_tc = strlen(hello_tc)+1; //注意我宣告為 long，因為 long
    是 64 位元
     long ret;
     printf("使用 'syscall' 呼叫 system call\n");
     __asm__ volatile (
         "mov $1, %%rax\n" //write 是第 1 號 system call
         "mov $2, %%rdi\n" //stderr  register 用法不一樣
         "mov %1, %%rsi\n" //buffer  register 用法不一樣
         "mov %2, %%rdx\n" //buffer size
         "syscall\n"        //使用 syscall 比 int 0x80 快 AMD 提出
         "mov %%rax, %0"        //system call 的回傳值依然放在 AX
         :"=m"(ret)
         :"g" (hello_tc), "g" (len_tc)
         :"rax", "rbx", "rcx", "rdx");
         printf("回傳值是：%ld\n", ret);
 }//main
```