实验 4 补充：一开始没有用用户调用系统中断，而是直接调用系统中断，后来补上

proto.h 增加用户调用的声明

```
/* 系统调用 - 用户级 */
PUBLIC  int     get_ticks();
PUBLIC  void    write(char* buf, int len);
PUBLIC void  process_sleep(int unused, int milli_sec, PROCESS * p);
PUBLIC void  tem_p(int unused,  semaphore * s,  PROCESS * p);
PUBLIC void  tem_v(int unused,  semaphore * s,  PROCESS * p);
```

syscall.asm 实现三个方法，其实就是直接调用系统中断,导出符号

```
; ========================================================================================
;       void  process_sleep(int unused1 int milli_sec, PROCESS * p)
; ========================================================================================
process_sleep:
                                mov    eax, _NR_sleep
    mov    ebx, [esp + 4]
    mov    ecx, [esp + 8]
    mov    edx, [esp + 12]
    int    INT_VECTOR_SYS_CALL
    ret

; ========================================================================================
;         void  tem_p(int unused,  semaphore * s,  PROCESS * p)
; ========================================================================================
tem_p:
                                mov    eax, _NR_p
    mov    ebx, [esp + 4]
    mov    ecx, [esp + 8]
    mov    edx, [esp + 12]
    int    INT_VECTOR_SYS_CALL
    ret
; ========================================================================================
;         void  tem_p(int unused,  semaphore * s,  PROCESS * p)
; ========================================================================================
tem_v:
                                mov    eax, _NR_v
    mov    ebx, [esp + 4]
    mov    ecx, [esp + 8]
    mov    edx, [esp + 12]
    int    INT_VECTOR_SYS_CALL
    ret
```

并在 global.c 中增加系统调用表

```
PUBLIC  system_call sys_call_table[NR_SYS_CALL] = {sys_get_ticks, sys_write , sys_process_sleep , sys_tem_p , sys_tem_v};
```