```c
int syscall_try_acquire_console(void)
{
    return msyscall(SYS_try_acquire_console, 0, 0, 0, 0, 0);
}

int syscall_release_console(void)
{
    return msyscall(SYS_release_console, 0, 0, 0, 0, 0);
}
```

```c
/*  lib/syscall_all.c  */
// 若锁处于空闲状态，该函数设置锁由当前进程持有，并返回 0；否则，该函数返回 -1。
int sys_try_acquire_console(int sysno, u_int envid, u_int value, u_int srcva, u_int perm) {
    if (lock != 0)
        return -1;
    lock = curenv->env_id;
    return 0;
}

int sys_release_console(int sysno, u_int envid, u_int value, u_int srcva, u_int perm) {
    if (lock != curenv->env_id)
        return -1;
    lock = 0;
    return 0;
}
```

Extra

```c
void sys_ipc_recv(int sysno, u_int dstva) {
    struct Page *p;
    if (dstva >= UTOP) {
        return;
    }
    int toid = ENVX(curenv->env_id);
    if (head[toid] < tail[toid]) {
        struct Env *e = extra_buffer[toid][head[toid]];
        u_int value = extra_value[toid][head[toid]];
        u_int srcva = extra_srcva[toid][head[toid]];
        u_int perm = extra_perm[toid][head[toid]];
        head[toid]++;

        curenv->env_ipc_value = value;
        curenv->env_ipc_from = e->env_id;
        curenv->env_ipc_perm = perm;
        curenv->env_ipc_recving = 0;
        curenv->env_status = ENV_RUNNABLE;
        e->env_status = ENV_RUNNABLE;
        if (srcva != 0) {
            p = page_lookup(e->env_pgdir, srcva, NULL);
            if (p == NULL || dstva >= UTOP) {
                return;
            }
            page_insert(curenv->env_pgdir, p, dstva, perm);
        }
        return;
    }
    curenv->env_ipc_recving = 1;
    curenv->env_ipc_dstva = dstva;
    curenv->env_status = ENV_NOT_RUNNABLE;
    sys_yield();
}

/* Overview:
 *  Try to send 'value' to the target env 'envid'.
 *
 *  The send fails with a return value of -E_IPC_NOT_RECV if the
 * target has not requested IPC with sys_ipc_recv.
 *  Otherwise, the send succeeds, and the target's ipc fields are
 * updated as follows:
 *    env_ipc_recving is set to 0 to block future sends
 *    env_ipc_from is set to the sending envid
 *    env_ipc_value is set to the 'value' parameter
 *  The target environment is marked runnable again.
 *
 * Post-Condition:
 *  Return 0 on success, < 0 on error.
 *
 * Hint: the only function you need to call is envid2env.
 */
/*** exercise 4.7 ***/
int sys_ipc_can_send(int sysno, u_int envid, u_int value, u_int srcva,
                     u_int perm) {

    int r;
    struct Env *e;
    struct Page *p;

    if (srcva >= UTOP) {
        return -E_INVAL;
    }
    r = envid2env(envid, &e, 0);
    if (r < 0) {
        return r;
    }
    if (e->env_ipc_recving == 0) {
        int toid = ENVX(envid);
        extra_buffer[toid][tail[toid]] = curenv;
        extra_value[toid][tail[toid]] = value;
        extra_perm[toid][tail[toid]] = perm;
        extra_srcva[toid][tail[toid]] = srcva;
        tail[toid]++;
        curenv->env_status = ENV_NOT_RUNNABLE;
        sys_yield();
        return -E_IPC_NOT_RECV;
    }
    e->env_ipc_value = value;
    e->env_ipc_from = curenv->env_id;
    e->env_ipc_perm = perm;
    e->env_ipc_recving = 0;
    e->env_status = ENV_RUNNABLE;
    if (srcva != 0) {
        p = page_lookup(curenv->env_pgdir, srcva, NULL);
        if (p == NULL || e->env_ipc_dstva >= UTOP) {
            return -E_INVAL;
        }
        r = page_insert(e->env_pgdir, p, e->env_ipc_dstva, perm);
        if (r) {
            return r;
        }
    }
    return 0;
}
```