```c
#include "sem.h"
#include "spin.h"

static void dummy() //Dummy handler. Does nothing
{
    // return;
}

void sem_init(struct sem *s, int count)
{
    s->spinlock = 0;
    s->max = count;
    s->free = count;
    s->procInd = -1;

    sigfillset(&s->mask_block);
    sigdelset(&s->mask_block, SIGUSR1);
    signal(SIGUSR1, dummy); // Prevent the signal from killing the process
}

int sem_try(struct sem *s)
{
    spin_lock(&s->spinlock);
    if (s->free > 0)
    {
        s->free -= 1;
        spin_unlock(&s->spinlock);
        return 1;
    }
    else
    {
        spin_unlock(&s->spinlock);
        return 0;
    }
}

void sem_wait(struct sem *s)
{
    for (;;)
    {
        spin_lock(&s->spinlock);

        if (s->free > 0)
        {
            s->free -= 1;
            spin_unlock(&s->spinlock);
            break;
        }
        else
        {
```

```
        {
            s->proc_block[s->procInd] = procNum; // Put process on waitlist
            s->procInd += 1;                     //Book keeping
            spin_unlock(&s->spinlock);
            sigsuspend(&s->mask_block); //Put process to sleep
        }
    }
}

void sem_inc(struct sem *s)
{
    spin_lock(&s->spinlock);

    s->free += 1; // Increment semaphore
    if (s->free == 1)
    {
        while (s->procInd != -1) //Loop to wake up all processes when sem
becomes 1
        {
            kill(pid_table[s->proc_block[s->procInd]], SIGUSR1);
            s->procInd -= 1; // Book keeping
        }
    }
    spin_unlock(&s->spinlock);
}
```