Diff:

sysproc.c

added:

```
int
sys_alsonice(void) {
    int c;
    argint(0,&c);
    return alsonice(c);
}
```

proc.h

added:

```
struct proc {
  uint sz;                     // Size of process memory (bytes)
  pde_t* pgdir;                // Page table
  char *kstack;                // Bottom of kernel stack for this
process
  enum procstate state;        // Process state
  int pid;                     // Process ID
  struct proc *parent;         // Parent process
  struct trapframe *tf;        // Trap frame for current syscall
  struct context *context;     // swtch() here to run process
  void *chan;                  // If non-zero, sleeping on chan
  int killed;                  // If non-zero, have been killed
  struct file *ofile[NOFILE];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)
  int counter;                 // time slice counter, default is 1
  int slice_alloc;             // so that the process 'remembers' how
long to run
};
```

proc.c

added:

```
int
alsonice(int n)
{
 if(n<0) {
     return -1;
```

```
 }
 myproc()->counter = myproc()->slice_alloc = n;
 return 1;          //succeeded in multiplying proc time slice
}
```

trap.c

added:

```
  if(myproc() && myproc()->state == RUNNING &&
     tf->trapno == T_IRQ0+IRQ_TIMER){
        if(!myproc()->counter || myproc()->counter == 1) {
            myproc()->counter = myproc()->slice_alloc;  //proc
'remembers' the number of timeslices it can run for the next time
gets the cpu
            yield();
        }
        else {
            myproc()->counter = myproc()->counter -1;
        }
     }
```

Added function prototype alsonice() in defs.h and user.h