```
/*-------------------------------------------
 * Diseño de Sistemas Operativos (DAC) *
        para compilar el módulo:
        $ MODULE=media_mod_v2 make
        para instalar el módulo:
        $ sudo insmod media_mod_v2.ko
        para comprobar si el módulo fue cargado:
        $ sudo lsmod
        $ dmesg | tail
        para desinstalar el módulo:
        $ sudo rmmod media_mod_v2
-------------------------------------------*/

#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <linux/string.h>
#include <linux/vmalloc.h>
#include <linux/uaccess.h>

#define MAX_COOKIE_LENGTH       PAGE_SIZE

static struct proc_dir_entry *proc_entry;

static int suma_numeros = 0;
static int cantidad_numeros = 0;


/*************************************************************************/
/* file operations                                                       */
/*************************************************************************/
// escritura del fichero
ssize_t media_write (struct file *filp,const char *buf, size_t count, loff_t
*off)
{
  /*int space_available = (MAX_COOKIE_LENGTH-cookie_index)+1;
  if (count > space_available) {
    printk(KERN_INFO "media: cookie pot is full!\n");
    return -ENOSPC;
  }
  if (copy_from_user( &cookie_pot[cookie_index], buf, count )) {
    return -EFAULT;
  }
  cookie_index += count;
  *off+=count;  // avanzo el offset del fichero
  cookie_pot[cookie_index-1] = 0; // marco el fin de la frase
  return count;*/

  char copia_buffer[512];
  int space_avaiable = 512;
  int variable;
   printk(KERN_INFO "media: write\n");

  if(count > space_avaiable) count = space_avaiable;
```

```
  if(*off>0) return 0;

  if(copy_from_user(copia_buffer, buf, (count>512)?512:count)) {
          return -EFAULT;
  }

  printk(KERN_INFO "media: write %d\n",count);

  if (strncmp(copia_buffer, "CLEAR", 5) == 0){
          cantidad_numeros = 0;
          suma_numeros = 0;

}   else if (sscanf(copia_buffer, " %d ", &variable) == 1){

          suma_numeros += variable;
          cantidad_numeros++;
  }else{
          return -EINVAL;
        }

         *off += count;
         return count;
}

// lectura del fichero
ssize_t media_read (struct file *filp, char __user *buf, size_t count, loff_t
*off ){
/* int len;
  int pos;

  //si el offset es mayor que cero, es la seguna lectura, devuelvo 0 = EOF
  if (*off > 0)  return 0;
  //si he llegado al final empiezo a leer la primera frase
  if (next_media >= cookie_index) next_media = 0;
  // busco el final de la frase
  for(pos=next_media ; cookie_pot[pos]!='\0' && pos<MAX_COOKIE_LENGTH ; pos ++);
  len=pos-next_media;  //calculo el tamaño
  if(len >= count) len=count-1;  // si es más de lo que me piden recorto
  //copio la frase
  if (copy_to_user(buf, &cookie_pot[next_media], len))  return -EFAULT;
  if (copy_to_user(buf+len, "\n", 1))  return -EFAULT;  //añado fin de linea
  len++;
  next_media += len;  //avanzo puntero de lectura interno
  *off+=len+1;                  //avanzo offset del fichero

  return len+1;                 //devuelvo número de bytes leidos (incluido
\n)*/

  char respuesta [1024];
  int entera, fraccion, len;

  if(*off > 0) return 0;
   entera = suma_numeros/cantidad_numeros;
```

```
   fraccion = ((suma_numeros % cantidad_numeros))*100/cantidad_numeros;

  if(cantidad_numeros > 0){
          len = sprintf(respuesta, "La media es: %d.%d (de %d numeros)\n",
entera, fraccion, cantidad_numeros);
  }else{
          len = sprintf(respuesta, "No hay respuesta\n");
  }
  if(copy_to_user(buf, respuesta, len)){
          return -EFAULT;
  }

  *off += len;
  return len;
}

struct file_operations proc_fops = {
        read : media_read,
        write: media_write
};


/*************************************************************************/
/* Module init / cleanup block.                                          */
/*************************************************************************/

// al cargar el modulo
int init_media_module( void )
{
  int ret = 0;

    proc_entry = proc_create("media", 0666, NULL, &proc_fops);

    if (proc_entry == NULL) {
      ret = -ENOMEM;
      printk(KERN_INFO "media: Couldn't create proc entry\n");
    } else {

      printk(KERN_INFO "media: Module loaded.\n");
    }

  return ret;
}

// al descargar el modulo
void cleanup_media_module( void )
{
  remove_proc_entry("media", NULL);
  printk(KERN_INFO "media: Module unloaded.\n");
}

module_init( init_media_module );
module_exit( cleanup_media_module );
```

```
/************************************************************************/
/* Module licensing/description block.                                  */
/************************************************************************/


MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Media Cookie Kernel Module (DSO)");
MODULE_AUTHOR("basado en el trabajo de M. Tim Jones, adaptado para kernel >
3.10");
```