1 Manejo del monitor

1.1 Objetivo

El objetivo de esta primera práctica es la toma de contacto y familiarización con el monitor del sistema entrenador *68fil*.

1.2 Práctica

1.2.1 Descripción del entrenador *68fil*

El sistema entrenador *68fil* está diseñado específicamente para la enseñanza del microprocesador 68000 de Motorola. Este sistema consta de los siguientes elementos:

- microprocesador 68000 a 8 MHz
- 16 Kbytes de memoria ROM (ampliables a 64 Kbytes)
- 64 Kbytes de memoria RAM estática
- PI/T 68230 que proporciona:
  - 16 líneas de E/S más 4 líneas de protocolo
  - Temporizador (timer) de 24 bits
- DUART 68681 que proporciona 2 puertas serie RS-232
- Bus de expansión para ampliaciones del sistema

La memoria RAM está disponible casi en su totalidad para almacenar programas y datos del usuario. Con 64 Kbytes no habrá problemas de espacio. La memoria ROM contiene un programa monitor que permite controlar el funcionamiento del sistema. El monitor permite realizar las siguientes actividades:

- Ver y modificar posiciones de memoria
- Ver y modificar los registros del 68000
- Cargar programas desde otro ordenador
- Ejecutar programas de forma continua o paso a paso

El monitor se comunica con el usuario a través de uno de los puertos serie (el puerto A). A éste debe conectarse un terminal u ordenador, desde el que el usuario puede introducir órdenes y ver sus resultados.

1.2.2 Distribución del espacio de direccionamiento del 68000

Tanto la memoria RAM como la ROM, así como los dispositivos de periféricos 68230 y 68681 tienen asignadas unas direcciónes determinadas en el espacio de direccionamiento del 68000. Este espacio está distribuido de la forma que se muestra en la Figura 1.1.

En los primeros momentos de funcionamiento del *68fil*, la memoria ROM ocupa también las posiciones 000000_{16}-$00FFFF_{16}$. De esta forma el microprocesador obtiene los `PC` y `SSP` iniciales fijados en la memoria no volátil. Acto seguido es la memoria RAM la que ocupa esas posiciones, quedando el mapa del espacio de direccionamiento tal como se ve en la Figura 1.1.

1.2.3 Conexión con la placa *68fil*

Cada entrenador *68fil* se encuentra conectado por un canal serie RS-232 a un puerto serie de uno de los ordenadores del Laboratorio de Informática. En realidad, puesto que estos ordenadores están conectados a la red de la UPV/EHU, es posible acceder a las placas *68fil* desde Internet, siempre, claro está, que tengamos una cuenta de usuario.

Para acceder al *68fil* desde el ordenador basta con utilizar un terminal de texto y disponer de un programa de comunicaciones a través del puerto serie. Por tanto, no es necesario arrancar el sistema de ventanas. En realidad, la forma más cómoda de trabajar es tener abiertas tres consolas de texto, una para editar, otra para ensamblar y otra para conectarse al *68fil*. En cualquier caso lo primero será acceder a nuestra cuenta de usuario:

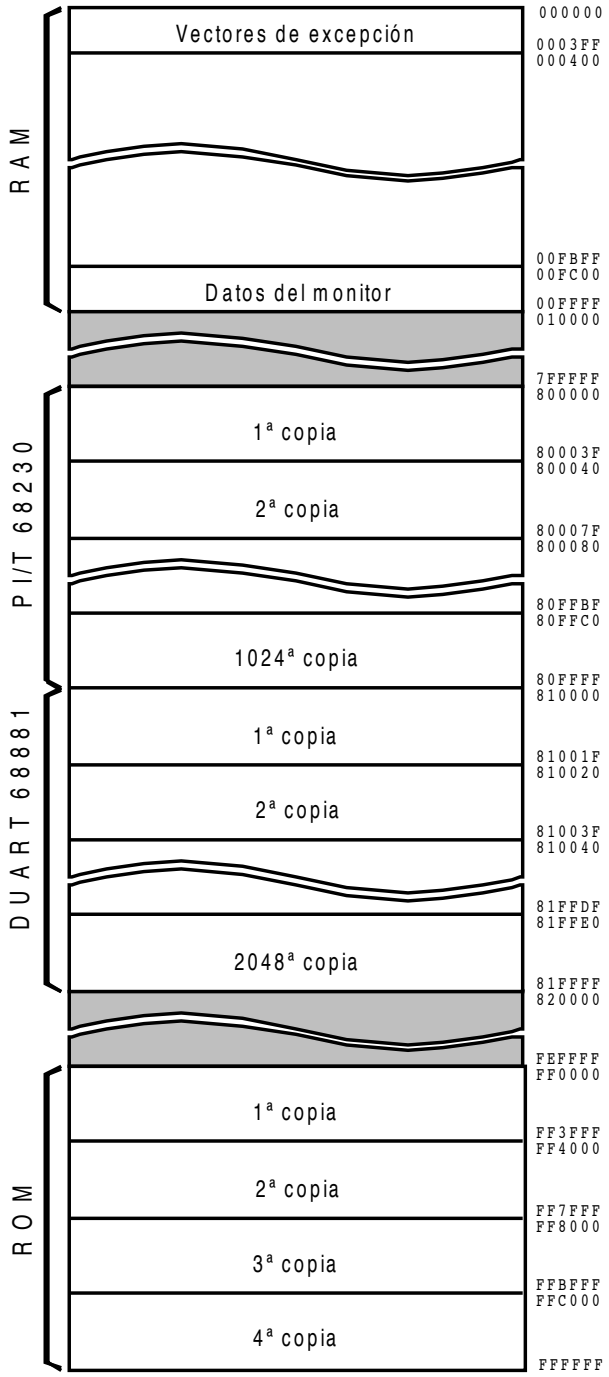

***Figura 1.1**.- Mapa del espacio de direccionamiento del 68fil.*

```
login: luisja [↲]
Password: ******* [↲]
Last login: Tue Apr  9 16:35:29 2002 from sinatra.we.lc.ehu.es
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.
FreeBSD 4.5-RELEASE (LDI) #3: Fri Mar 22 13:16:29 CET 2002

Nice tcsh prompt: set prompt = '%n@%m:%~%# '
                  --  Giorgos Keramidas <keramida@ceid.upatras.gr>
luisja@lcpd07[~]$
```

A continuación ejecutamos el programa de comunicaciones (*kermit*) para conectar con el *68fil*, cuya salida vemos en el terminal de texto:

```
luisja@lcpd07[~]$ kermit [↵]
Executing /home/luisja/.kermrc for UNIX...
Executing /home/luisja/.mykermrc...
Good Afternoon!
C-Kermit 8.0.201, 8 Feb 2002, for FreeBSD 4.0
 Copyright (C) 1985, 2002,
  Trustees of Columbia University in the City of New York.
Type ? or HELP for help.
Kermit> connect [↵]
Connecting to /dev/cuaa0, speed 9600
Escape character: Ctrl-E (ASCII 5, ENQ): enabled
Type the escape character followed by C to get back,
or followed by ? to see other options.
----------------------------------------------------
```

Sólo un usuario en cada momento puede estar conectado con cada *68fil*. Así pues, aunque en principio podamos acceder de manera remota (por ejemplo desde casa) a cualquiera de las máquinas *lcpdXX* (con XX = 01, 02, ..., 16) del laboratorio (y por tanto a cualquiera de los *68fil*), no será posible hacerlo si otro usuario se ha conectado previamente, o si la máquina o el *68fil* están desconectados. Una vez establecida la conexión podemos empezar a dar órdenes al programa monitor. Lo más recomendable es *resetear* la placa mediante el comando *x*, que primero realizará un test de memoria y después mostrará la pantalla de "ayuda":

```
x [↵]
Monitor 68fil  (version: 1.3-2C93)
=============
Compatible con entrenadores 68fil/v2
Copyright FiloSoft 1989

---------- instrucciones ----------|-------------- parametros --------------
                                   |
A           - ayuda                |  r = identificador de registro
C           - cargar programa      |  v = valor
R [r v]     - registros            |  d = direccion de memoria
V [i [f]]   - volcar memoria       |  i = direccion inicial de bloque
M d ["c"]   - modificar memoria    |  f = direccion final de bloque
B i f ["c"] - buscar en memoria    |  b = byte
D i f d     - duplicar memoria     |  l = lista de bytes
I i f [b]   - inicializar memoria  |  n = numero de punto de ruptura (1..7)
P [n [d]]   - punto de ruptura     |  c = cadena de caracteres
E [d]       - ejecutar             |
T [d]       - trazar               |  [z]   "z" parametro opcional
S [d]       - trazar sin saltos    |
L [i [f]]   - listar programa      |
X           - reiniciar el monitor |
? [d]       - detectar errores     |
```

A continuación el entrenador queda a la espera de nuestros mandatos mostrando el *prompt*:

```
68fil:
```

En caso de que el *68fil* no haya respondido, debemos asegurarnos de que la placa conectada a nuestro ordenador está encendida (observar el LED); si lo está, entonces lo más recomendable es accionar el pulsador de `RESET`.

A partir de este momento ya estaremos trabajando directamente con el entrenador *68fil*.

1.2.4 Lectura y modificación de la memoria

El programa monitor permite la posibilidad de ver el contenido de zonas de memoria, así como de modificar posiciones en bloque o individualmente. Para todo ello disponemos de las órdenes `V` (ver memoria), `L` (listar memoria), `I` (inicializar memoria) y `M` (modificar memoria).

Como práctica:

Como norma general cargaremos los programas y datos a partir de la dirección 1000_{16}. La razón principal para ello es que se trata de un número fácil de recordar y queda suficiente espacio por delante

---

para alojar los programas que realizaremos. Podemos ver el contenido de las 256 posiciones de memoria a partir de 1000_{16} introduciendo:

```
68fil: V 1000 [↲]
```

También podemos ordenar al monitor que liste la misma zona de memoria interpretando su contenido como mnemónicos de las instrucciones del 68000 (es decir, desensamblando) si introducimos la orden:

```
68fil: L 1000 [↲]
```

Si queremos inicializar un bloque de memoria con un determinado valor, lo más sencillo es utilizar la orden de inicializar memoria. Con los siguientes mandatos inicializaremos las posiciones de memoria 001000_{16}-$00100F_{16}$ a 55_{16}, comprobando a continuación el resultado:

```
68fil: I 1000 100F 55 [↲]
68fil: V 1000 [↲]
```

Vamos ahora a introducir unas instrucciones sencillas a partir de la dirección 001000_{16}; por ejemplo:

```
<001000> D240                   ADD.W   D0,D1
<001002> C2FC 0007              MULU    #7,D1
<001006> 21C1 2000              MOVE.L  D1,2000
```

Para ello utilizaremos la orden de modificar memoria:

```
68fil: M 1000 [↲]
<001000> 5555 D240 [↲]
<001002> 5555 C2FC [↲]
<001004> 5555 0007 [↲]
<001006> 5555 21C1 [↲]
<001008> 5555 2000 [↲]
<00100A> 5555 [ESC]
```

Podemos comprobar que lo hemos hecho correctamente listando la memoria a partir de 1000_{16}.

1.2.5 Lectura y modificación de los registros

El monitor posee una copia de los registros del 68000. Antes de ejecutar un programa o una instrucción el monitor copia el contenido de sus registros en los del microprocesador. Asimismo, cuando termina la ejecución de un programa o de una instrucción el monitor copia el contenido de los registros del 68000 en los suyos. Mediante la orden `R` (ver/modificar registros) el monitor nos permite ver su copia de los registros o modificar cualquiera de ellos. De esta forma podemos ver cómo han quedado tras la ejecución, o introducir en ellos unos valores determinados antes de ejecutar un programa o instrucción.

Como práctica:

El estado de los registros se puede ver mediante la orden:

```
68fil: R [↲]
```

Al poner en marcha o reinicializar el sistema, todos los registros del monitor toman el valor 0, salvo el `SSP` y `USP`, que toman los valores $0000FA00_{16}$ y $0000F800_{16}$ respectivamente.

Seguidamente vamos a introducir dos valores en los registros `D0` y `D1`:

```
68fil: R D0 16 [↲]
68fil: R D1 39 [↲]
```

Cada vez que un registro es modificado el monitor muestra el conjunto completo de registros con el objeto de que el usuario compruebe la modificación de forma inmediata.

1.2.6 Ejecución de instrucciones y programas

El monitor del *68fil* proporciona la posibilidad de ejecutar programas de forma normal o "paso a paso" (traza). Para lo primero dispone de las órdenes `T` (trazar instrucción) y `S` (trazar instrucción sin saltar). Como complemento a estos mandatos el monitor permite colocar "puntos de ruptura" (*breakpoints*) en cualquier lugar del programa.

La orden T ejecuta una instrucción, tras lo cual muestra los registros y la instrucción apuntada por el PC. Su aplicación principal es el depurado de programas que no funcionan correctamente. Ejecutando instrucción a instrucción podemos estudiar detenidamente el funcionamiento de un programa y hallar los puntos donde se encuentran los fallos.

Como práctica:

Vamos a usar la orden T con las instrucciones que ya hemos introducido. Para ejecutar la instrucción situada en la dirección 001000_{16} debemos introducir el mandato:

```
68fil: T 1000 [↲]
```

Obsérvese el estado de los registros D0 y D1 (recordando el valor que contenían previamente), así como del PC y del CCR (el byte bajo del SR). Véase también la instrucción apuntada por el PC.

Si introducimos la orden T sin suministrar ninguna dirección, el monitor ejecuta la instrucción apuntada por la dirección contenida en el PC. Por otra parte, si pulsamos únicamente la tecla [↲] el monitor repite el último mandato, suprimiendo los argumentos. Por tanto, para ejecutar paso a paso las instrucciones de un programa de forma sucesiva, basta con pulsar la tecla [↲] tantas veces como instrucciones queramos ejecutar (evidentemente para ejecutar la primera instrucción es preciso introducir completa la orden de traza). Nuestro ejemplo consta de dos instrucciones más; por tanto, para ejecutarlas hemos de introducir:

```
68fil: [↲]
68fil: [↲]
```

Llévense a cabo las mismas observaciones que en el caso anterior. Compruébese además el contenido de la *word* larga que ocupa las posiciones de memoria 002000_{16}-002003_{16} utilizando para ello la orden V.

La orden E ejecuta un programa de forma continuada a partir de una dirección determinada. Esta puede ser suministrada como argumento o a través del PC, de forma similar a la orden T. La ejecución puede finalizar de diversas maneras: con una excepción (reinicialización, error de bus, error de direccionamiento, interrupción de nivel 7), cuando se detecta un punto de ruptura, o por medio de una instrucción TRAP (ésta es la manera más adecuada). Finalizada la ejecución el control es devuelto al programa monitor, mediante el cual podremos observar las modificaciones realizadas por el programa en los registros y en la memoria del sistema.

Como práctica:

En primer lugar vamos a añadir una instrucción más a las que ya tenemos introducidas en memoria. Con esta nueva instrucción (una TRAP) lograremos devolver el control al monitor al término de la ejecución de nuestro pequeño programa. La instrucción TRAP de finalización ha de tener un formato especial, específico para el entrenador *68fil*, que es el siguiente:

```
<00100A> 4E4F                    TRAP    #F
<00100C> 0000                    DC.W    0000
```

La *word* utilizada tras el código de operación es un "pseudo-operando" utilizado por la rutina del monitor desencadenada por la instrucción TRAP #F. Cuando esta *word* es cero, indica "retorno al programa monitor". Añádase esta instrucción utilizando la orden M y compruébese listando la memoria mediante la orden L. Obsérvese cómo la instrucción TRAP #F seguida de una *word* con valor 0 es interpretada por el monitor del *68fil* de una manera especial.

A continuación deben introducirse en los registros D0 y D1 los mismos valores que en el caso anterior, e inicializar las posiciones de memoria 002000_{16}- 002003_{16} con algún valor determinado (por ejemplo, AA_{16}). Ejecútese a continuación el programa mediante la orden:

```
68fil: E 1000 [↲]
```

Obsérvense los resultados en los registros y en las posiciones de memoria 002000_{16} a 002003_{16}. Lógicamente deben ser los mismos que los obtenidos tras la ejecución paso a paso.

Los puntos de ruptura (*breakpoints*) son direcciones de memoria que, al ser apuntadas por el `PC`, provocan la detención del programa, devolviendo el control al monitor. Con ellos se facilita enormemente la depuración de código. Por ejemplo, es posible saber si un programa ejecuta cierto grupo de instrucciones colocando un *breakpoint* en la dirección correspondiente a alguna de ellas. También permiten ejecutar de forma continua hasta una determinada instrucción, para a partir de ese punto ejecutar paso a paso. Tienen también otras aplicaciones que veremos en prácticas posteriores.

El monitor del *68fil* permite situar 7 puntos de ruptura (numerados del 1 al 7) en diferentes direcciones. Asimismo permite eliminarlos selectivamente y listarlos. Cuando una ejecución finaliza debido a un punto de ruptura, el monitor indica su número y muestra los registros y la instrucción en la que la ejecución se ha detenido (que aún no ha sido ejecutada). Todo esto se realiza mediante la orden `P` (punto de ruptura). Para situar un *breakpoint* en una dirección determinada (obviamente par) a este mandato se le suministran como argumentos el nº de punto de ruptura y dicha dirección; el monitor responde mostrando todos los puntos de ruptura activados. Para eliminar un *breakpoint* basta con suministrar como argumento el nº de punto de ruptura a desactivar.

Como práctica:

Vamos a situar un punto de ruptura en la dirección correspondiente a la instrucción `MULU` de nuestro conocido programa:

```
68fil: P 1 1002 [↵]
```

A continuación podemos ejecutar a partir de la dirección 001000_{16}:

```
68fil: E 1000 [↵]
```

Obsérvese cómo la ejecución se detiene tras la primera instrucción; la segunda (`MULU`) aún no ha sido ejecutada. A partir de este punto podemos continuar la ejecución de forma normal o paso a paso. Por ejemplo, si introducimos al mandato:

```
68fil: T [↵]
```

con ello hemos ejecutado la instrucción `MULU`. Si ahora ordenamos al monitor

```
68fil: E [↵]
```

serán ejecutadas las instrucciones restantes, finalizando con la instrucción `TRAP #F`. El punto de ruptura nº 1 seguirá activado hasta que lo eliminemos con el mandato

```
68fil: P 1 [↵]
```

1.2.7 Ensamblado y envío de programas desde el ordenador

Hemos visto ya cómo introducir programas y datos en la memoria del *68fil*. También sabemos cómo ejecutar programas previamente cargados en memoria. Sin embargo, es evidente que la codificación (ensamblado) e introducción en memoria de un programa cuyo número de instrucciones sobrepase la media docena es una tarea poco recomendable para la salud mental. A continuación veremos un método para automatizar este tedioso proceso.

El monitor del *68fil* dispone de una utilidad que permite cargar programas en memoria a través de la puerta RS-232. Para que un programa pueda ser cargado de esta manera, debe haber sido previamente codificado y ajustado a un formato denominado "S-records" (un estándar definido por Motorola). Si en nuestro ordenador disponemos de un ensamblador cruzado capaz de codificar nuestro programa y ajustarlo al formato "S-records", podremos enviar nuestro programa al *68fil*, una vez ensamblado, a través de la línea serie RS-232. La orden del monitor que prepara al *68fil* para recibir un programa en este formato es `C` (cargar programa).

Por otra parte, existe un número considerable de ensambladores cruzados para el 68000 que funcionan en los ordenadores más diversos. La única condición que deben cumplir es que sean capaces de generar código en formato "S-records". Veremos ahora cómo realizar este proceso desde el ordenador que utilizamos en estas prácticas.

Como práctica:

En primer lugar es preciso crear un fichero conteniendo el programa fuente a codificar. Para nuestros propósitos lo más conveniente es que el programa sea el mismo que hemos estado manejando anteriormente. Escrito en lenguaje ensamblador toma el siguiente aspecto:

```
                        ; asmsyntax=asm68k
        CPU     68000

        ORG     $1000       ; dirección de carga del programa
MIPROG  ADD.W   D0,D1       ; suma D0 con D1, el resultado
        MULU    #7,D1       ; lo multiplica por 7 y
        MOVE.L  D1,RESUL    ; lo almacena en RESUL

        TRAP    #15         ; vuelve al monitor
        DC.W    0

        ORG     $2000       ; dirección de carga para los datos
RESUL   DS.L    1           ; aquí se guarda el resultado
```

El programa puede escribirse utilizando cualquier editor de texto. Si se utiliza el editor `vim`, puede resultar de gran utilidad añadir al código fuente de los programas una primera línea con el comentario `asmsyntax=asm68k`, que permitirá al `vim` reconocer las instrucciones, pseudoinstrucciones, constantes, comentarios, registros, etc. específicos del 68000, y hará las tareas de edición y lectura mucho más agradables.

Una vez creado el programa, para ensamblarlo se utilizará el comando `asl`. En el código fuente se ha incluido una pseudoinstrucción (`CPU 68000`) que indica al comando `asl` en qué lenguaje se ha escrito el programa, es decir, para qué microprocesador ha de realizar el ensamblado. El nombre del fichero fuente debe tener la extensión `.asm`, por ejemplo `prueba.asm`:

```
luisja@lcpd07[~/68000]$ asl prueba [↲]
macro assembler 1.41r8
(i386-unknown-freebsd)
(C) 1992,1998 Alfred Arnold
Mitsubishi M16C-Generator also (C) 1999 RMS
TMS320C2x-Generator (C) 1994/96 Thomas Sailer
TMS320C5x-Generator (C) 1995/96 Thomas Sailer

assembling prueba.asm
PASS 1
prueba.asm(14)
PASS 2
prueba.asm(14)

0.07 seconds assembly time

        14 lines source file
        2 passes
        0 errors
        0 warnings
luisja@lcpd07[~/68000]$
```

Si el programa no contiene errores, el ensamblador genera un fichero (`prueba.p`) conteniendo las instrucciones y los datos ya completamente codificados, lo que conocemos como código objeto absoluto (ejecutable). El nombre de este fichero es el mismo que el del programa fuente, salvo por la extensión, que en este caso es `.p`. Nótese que ha sido necesario indicar en el código fuente, mediante la pseudoinstrucción `ORG`, las direcciones absolutas de carga de instrucciones y datos.

En este punto conviene aclarar que la herramienta de ensamblado `asl` no es capaz de generar código objeto reubicable, ni tampoco de procesarlo. Es decir no va a poder ensamblar distintos ficheros por separado, ni después enlazar los ficheros resultantes para generar un único ejecutable resolviendo las referencias cruzadas, como por ejemplo podemos hacer con el compilador de C. Así pues debemos suministrar el código fuente en un sólo fichero, en el que han de estar definidos todos los símbolos, ya sean etiquetas, nombres de macros, constantes, etc. Sin embargo, esto no va a suponer ninguna limitación, como ya veremos más adelante, pues `asl` proporciona herramientas para:

1. definir macros,
2. incluir el código de otros ficheros,
3. definir símbolos locales (o privados), y
4. definir símbolos globales.

Volviendo al asunto que nos tenía ocupados, el monitor *68fil* sólo es capaz de cargar programas en formato "S-records". Por ello será necesario aplicar el comando `p2hex` al fichero `.p` obtenido previamente. El comando `p2hex` deberá ejecutarse con la opción `+5`, para evitar que se genere un *record S5*, no válido para el *68fil*. Como resultado se obtendrá un fichero en formato "S-records" y con la extensión `.hex`. El fichero obtenido (`prueba.hex`) es de tipo texto y su formato puede consultarse en el manual del *68fil*. Dedíquese algo de tiempo a la tarea de descifrar este fichero.

```
luisja@lcpd07[~/68000]$ p2hex +5 prueba [↲]
P2HEX/C V1.41r8
(C) 1992,1998 Alfred Arnold
prueba.p==>>prueba.hex  (14 Byte)
luisja@lcpd07[~/68000]$
```

Una vez obtenido el programa ejecutable, pasaremos a trabajar con la placa del entrenador *68fil*. Estando conectados a él mediante el emulador de terminal, lo preparamos para recibir código en formato "S-records" mediante el comando `C` (cargar). A continuación, mediante el comando `xmit`, indicamos al `kermit` que envíe el programa ejecutable (`prueba.hex`) al *68fil*. Por último el comando `C` del `kermit` reanuda la conexión entre el ordenador y el *68fil*, tras lo cual vemos aparecer un mensaje con los parámetros de la conexión seguido del *prompt* del *68fil*:

```
68fil: C [↲]
Listo para recepción
[Ahora pulsar CTRL-E, y a continuación, C]

(Back at lcpd07)
------------------------------------------------------
Kermit> xmit prueba.hex [↲]
Kermit> c [↲]
Connecting to /dev/cuaa0, speed 9600
Escape character: Ctrl-E (ASCII 5, ENQ): enabled
Type the escape character followed by C to get back,
or followed by ? to see other options.
------------------------------------------------------

68fil:
```

Compruébese que el programa ha sido cargado correctamente y compárese con el que se había introducido anteriormente de forma manual.

Para desconectar definitivamente del *68fil*, hemos de teclear `CTRL-E` y a continuación `q`. Esto provoca la finalización de la conexión, y también del programa `kermit`, por lo que veremos de nuevo el *prompt* del shell de UNIX:

```
68fil: CTRL-E q
Closing /dev/cuaa0...OK
luisja@lcpd07[~/68000]$
```