SISTEMA OPERATIVO MINIX

BOOT MONITOR

Componentes del grupo:

Roberto Jorge Alejandro
Carlos Díaz Suárez

INDICE

1. BOOT MONITOR

El ***Boot Monitor*** es un programa interactivo que se ejecuta en tiempo de arranque. Ha sido diseñado no a cargar e iniciar el sistema operativo **MINIX**, su más importante tarea, sino también para proporcionar una sencilla interfaz que permita configurar MINIX y arrancar otros sistemas operativos.

2. CARGA Y ARRANQUE DEL BOOT MONITOR

Cuando el ordenador se enciende, un programa en ROM (la BIOS) carga en memoria principal el primer sector de la primera pista del disco de arranque para ejecutar un programa llamado ***bootstrap***. Dicho programa no ha de ocupar más del espacio disponible en el sector de arranque.

Ciertos detalles varían según el disco de arranque sea un disquete o un disco duro. En un disquete, el sector de arranque contiene el programa *bootstrap*, sin embargo, en un disco duro el sector de arranque contiene un pequeño programa y la tabla de partición del disco, que en conjunto forman lo que se denomina ***Master Boot Record***. En el primer sector de la partición activa del disco duro es donde se encuentra el programa *bootstrap*.

El programa *bootstrap* varía según el sistema operativo instalado en el disco. El *bootstrap* de MINIX carga un programa de mayor tamaño llamado *boot*, que a su vez carga el sistema operativo en sí. Dicho programa recibe el nombre de *Boot Monitor.*

3. MENÚ PRINCIPAL

Una vez arranca el *Boot Monitor*, este presenta la siguiente pantalla:

```
Minix boot monitor 2.5
Press ESC to enter the monitor
Hit a key as follows:
= Start Minix
```

La primera línea muestra la versión del *Boot Monitor*, la segunda indica que pulsando la tecla “ESC” se accede a modo monitor, por último, se muestra una serie de opciones para arrancar distintos sistemas operativos, en este caso la única opción disponible permite arrancar el sistema operativo MINIX tras pulsar la

4. LÍNEA DE COMANDOS

Al entrar en el modo monitor se muestra el ***prompt*** correspondiente a la linea de comandos del *Boot Monitor*. Dicho *prompt* aparece como la concatenación del nombre del dispositivo desde el cual ha arrancado

Para aprender a interactuar con el *Boot Monitor* desde la linea de comandos es conveniente ejecutar el comando ***help*** y estudiar la pantalla que se presenta a continuación:

```
Names:

Rootdev                 - Root device
Ramimagedev             - RAM disk image if root is RAM
Ramsize                 - RAM disk size if root is not RAM
Bootdev                 - Special name for the boot device
fd0,hd3,hd2a            - Devices (as in /dev)
image                   - Name of the Kernel image
main                    - Startup function
```

```
Comands:

name = [device] value       - Set enviromment variable
name () {...}               - Define function
name (key,text) {...}       - A menu function like: minix(=,Start Minix){boot}
name                        - Call function
boot [device]               - Boot Minix or another O.S.
delay[msec]                 - Delay (500 msec default)
echo word...                - Print the words
ls [directory]              - List contents of directory
menu                        - Choose a menu function
save                        - Save enviroment
set                         - Show enviroment
trap msec command           - Schedule command
unset name                  - unset variable or set to default
exit                        - exit the monitor
```

En esta pantalla se distinguen dos secciones, variables de entorno (names) y comandos (commands) disponibles.

5. VARIABLES DE ENTORNO

El *Boot Monitor* dispone de 512 bytes en el segudo sector del disco (llamado sector de parámetros) para guardar los valores asignados a una serie de variables de entorno. De entre dichas variables, unas configuran el arranque del sistema, otras proporcionan información acerca del mismo y otras son propias del *Boot Monitor*.

Estas variables son inicializadas al arrancar el *Boot Monitor* bien con el valor por defecto que depende de la máquina sobre la que se está ejecutando el monitor, o bien con el valor guardado en el sector de parámetros.

Con el comando ***set*** se muestra en pantalla la configuración actual del entorno, donde los valores por defecto de las variables están entre paréntesis:

```
rootdev =  hd2a
ramimagedev = hd2a
ramsize = (0)
processor = (386)
bus = (at)
memsize = (639)
emssize = (1280)
video = (vga)
chrome = (color)
image = (minix)
main ( ) {menu}
```

rootdev: Indica el dispositivo utilizado como dispositivo raíz. Por defecto tiene el valor "*ram*", que significa que el dispositivo especificado por la variable *ramimagedev* será cargado en el disco virtual RAM (zona de memoria con la estructura de un sistema de ficheros) y usado como raíz.

Si se cambia el valor por defecto, entonces el dispositivo físico asignado será usado como raíz y el disco virtual RAM quedará sin inicializar y con el tamaño especificado por la variable *ramsize.*

ramimagedev: Describe el dispositivo a usar para inicializar el disco virtual RAM si *rootdev* tiene el valor "*ram*". Por defecto, esta variable tiene asignado el valor "*bootdev*", un nombre especial para el dispositivo desde el cual ha arrancado el monitor.

ramsize: Tamaño del disco virtual RAM.

processor: El valor por defecto puede ser 86, 186, 286, 386, 486, ... dependiendo del hardware de la máquina.

bus: Tipo de bus del sistema, que puede ser *xt*, *at* o *mca*.

memsize: Kilobytes de memoria convencional.

emssize: Kilobytes de memoria extendida.

video: Indica el tipo de tarjeta de video: *mda*, *cga*, *ega* o *vga*.

chrome: Admite dos posibles valores, *color* o *mono*.

image: nombre de la imagen del núcleo, por defecto "*minix*".

6. COMANDOS

El *Boot Monitor* ofrece una serie de comandos, los cuales se comentan a continuación, que permiten manipular el entorno en tiempo de arranque desde el *prompt* del monitor:

boot: cuando se ejecuta sin especificar un dispositivo, carga la imagen del núcleo del sistema operativo MINIX. Si se especifica un dispositivo, carga y ejecuta el sector de arranque de dicho dispositivo.

delay: suspende la ejecución del monitor durante un cierto tiempo especificado por el usuario.

echo: muestra en pantalla el mensaje escrito tras dicho comando.

ls: lista el contenido de un directorio.

menu: muestra las funciones de menu definidas y permite escoger una de ellas.

set: muestra la configuración actual del entorno.

trap: permite retrasar la ejecución de un comando durante un cierto tiempo especificado por el usuario, sin que ello impida que mientras puedan ejecutarse otros comandos.

unset: borra una variable o función, y en el caso de las variables de entorno les asigna el valor por defecto.

exit: termina la ejecución del monitor y reinicia la máquina.

7. DECLARACIÓN Y ASIGNACIÓN DE VARIABLES

El *Boot Monitor* permite modificar la variables de entorno, crear nuevas variables y asignar valores a éstas. Para ello se emplea la siguiente sintaxis:

nombre_variable=valor.

8. EDICIÓN DE FUNCIONES

El *Boot Monitor* también ofrece una sintaxis para la creación de funciones en base a los comandos vistos anteriormente. Existen dos tipos, aquellas que pueden invocarse desde el menú y otras que pueden invocarse omandos o ser llamadas por las anteriores, la sintaxis correspondiente a cada uno de los tipos de función se muestra a continuación:

Desde la línea de comandos: nombre_funcion () {...}

Desde el menú: nombre_funcion (tecla,texto) {...}

9. EJEMPLOS

A continuación se muestra algunos ejemplos que ilustran lo expuesto en apartados anteriores:

Arranque automático desde el menú:

```
main()={trap 5000 boot; menu}
```

De esta forma, pasados 5 segundos (en tiempo de la máquina), se carga el s.o. MINIX directamente si el usuario no ha seleccionado previamente una de las opciones del menú.

Adición de opciones al menú:

```
nueva_opcion(w,Windows 98) {boot hd1}
```

Esta función se añade al conjunto de funciones del menú, tal que pulsando la tecla w se puede arrancar el s.o. Windows, que en este caso se encuentra en la partición hd1.

Mensajes

```
mensaje () {echo hola mundo; delay 1000}
main () {mensaje; menu}
```

En este caso la función principal muestra un mensaje, espera un cierto tiempo y a continuación muestra las opciones del menú.

Nota: En la edición de funciones no es necesario utilizar el ";" explícitamente ya que cada nueva línea que aparece al pulsar return es considerada como una nueva sentencia.

10. ARRANQUE DEL MINIX

Tal y como se dijo anteriormente, cuando el comando *boot* se ejecuta sin especificar un dispositivo, carga la imagen del núcleo del sistema operativo MINIX, mientras que si se especifica un dispositivo, carga y ejecuta el sector de arranque de dicho dispositivo, es decir, arranca el *bootstrap* correspondiente.

En el caso de no especificar dispositivo, el comando *boot* busca un fichero o un directorio con el mismo nombre que el valor de la variable *image* en el directorio raíz del dispositivo especificado por la variable rootdev. Si fuese un directorio, el *boot* tomaría el fichero más reciente en dicho directorio (por lo general éste es el comportamiento por defecto).

El fichero antes mencionado no es más que la imagen del s.o. MINIX, formada por la concatenación de ficheros individuales producidos por el compilador cuando se compilan el **núcleo**, el **manejador de memoria**, el **sistema de ficheros** y el programa ***init*** (primer proceso de usuario que se ejecuta).

Cada una de estas partes contiene una cabecera con información utilizada por el *Boot Monitor* para reservar espacio en memoria suficiente y a continuación cargar cada una de ellas. Una vez se ha completado la carga, el control pasa al código ejecutable del núcleo, y el *Boot Monitor* termina su ejecución.

El siguiente ejemplo muesta la carga en memoria de los diferentes módulos que componen la imagen del s.o. MINIX.

```
Loading Minix

     cs      ds     text    data     bss    stack
 000800  00bd00   53120    8924   40076      0     kernel
 100000  103100   12544    1176   29576   1024      mm
 10ae00  112000   29152    2196  108084   2048      fs
 12d700  12d700    6828    2032    1356    768     init
```

11. DIRECTORIO BOOT

Dentro del directorio /usr/src/boot se encuentran los ficheros fuentes del Boot Monitor, así como el archivo makefile utilizado para compilar dichos ficheros:

masterboot.s - boot master de arranque para el primer sector del disco.
extboot.s - boot master de arranque para particiones extendidas de MS-DOS.
bootblock.s - bootstrap para el primer sector de la partición activa, carga el boot.
boot.c - Boot Monitor, boot secundario, carga y arranca el MINIX.
boot.h - información entre diferentes partes del boot.
boothead.s - soporte de la BIOS para el boot.c.
edparams.c - permite modificar parámetros del sistema.
installboot.c - instala el sector de arranque (bootblock), o instala el ejecutable del MINIX (image), para crear un disco de arranque.
image.h - información entre installboot y boot.
bootimage.c - carga y ejecuta una imagen del sistema operativo.
rawfs.c - soporte para manejar un sistema de ficheros tipo RAW.
rawfs.h

El fichero Makefile de este directorio tiene las órdenes para que se compilen estos programas y se dejen en los siguientes directorios:

Directorio /usr/mdec contiene los compilados:

- *boot*
- *bootblock*
- *extboot*
- *masterboot*

Directorio /usr/bin contiene los compilados:

- *installboot*
- *edparams*

12. FUNCIONES DEL BOOT.C

DISCO

name2dev - traduce el nombre de un dispositivo a su código numérico.
dev_geometry - establece los parámetros del dispositivo actual.
readsectors - lee uno o más sectores del dispositivo.
writesectors - escribe uno o más sectores del dispositivo.

MEMORIA

mon2abs - devuelve la dirección absoluta de una dirección del monitor.
vec2abs - devuelve la dirección absoluta de un vector.
raw_copy - copia bytes de una posición de memoria a otra.
get_word - lee una palabra de memoria.
put_word - escribe una palabra en memoria.
get_memsize - obtiene la cantidad de memoria convencional disponible.
get_ext_memsize - obtiene la cantidad de memoria extendida disponible.

CACHE

init_cache - inicializa la caché.
invalidate_cache - la caché no puede usarse cuando se carga el MINIX.
readblock - lee bloques para el paquete rawfs con caché.

BUS

get_bus - obtiene el tipo de bus del sistema.

ENTRADA/SALIDA ESTANDAR

getchar - lee un carácter del teclado bloqueando la entrada.
peekchar - lee un carácter del teclado sin bloquear la entrada.
putchar - imprime un caráter en pantalla.
get_video - devuelve el tipo de tarjeta de video.
reset_video - resetea y limpia la pantalla.

ARRANQUE

migrate - reubica el boot en la zona final de memoria.
relocate - pasa a ejecutar la copia del boot monitor.
get_master - lee el sector del master boot y la tabla de partición.
initialize - averigua cual fue el dispositivo y partición de arranque.

TEMPORIZACION

get_tick - valor actual del contador de pulsos de reloj.
milli_time - retorna el tiempo en milisegundos.
milli_since - retorna el tiempo transcurrido en milisegundos.
unschedule - invalida la ejecución de un comando en espera.
schedule - retrasa la ejecución de un comando un cierto tiempo.
expired - comprueba si el tiempo de espera ha finalizado.
delay - pausa la ejecución durante un cierto número de milisegundos.

MANEJO DE ERRORES

bios_err - traduce los códigos de error de la BIOS a un mensaje legible.
unix_err - traduce aquellos errores que pueda dar el rawfs.
rwerr - trata los errores de lectura o escritura del rawfs.
readerr - trata los errores de lectura del rawfs.
writerr - trata los errores de escritura del rawfs.

MANEJO DE RISTRAS

sfree - libera la memoria ocupada por una ristra no nula.
copystr - devuelve una copia de una ristra no nula.
numprefix - devuelve verdadero si es una ristra con un prefijo numérico.
numeric - devuelve verdadero si es una ristra numérica.

CONVERSION DE TIPOS

a2l - devuelve el valor numérico de una ristra.
u2a - devuelve la ristra correspondiente a un valor numérico.
a2x - devuelve el valor hexadecimal de una ristra.

ANALISIS LEXICO

readline - lee una nueva línea.
sugar - reconoce tokens especiales.
onetoken - devuelve una ristra con un token.
tokenize - obtiene los tokens que de una línea para formar una cadena de comandos.
poptoken - devuelve el primer token de la cadena de comandos.
voidtoken - elimina el primer token de la cadena de comandos.
interrupt - detecta la pulsación de la tecla ESC.

ENTORNO

is_default - comprueba si una variable o función tiene asignado el valor por defecto.
menufun - retorna la clase de la función pasada como parámetro.
searchenv - busca una variable o función en el entorno.
show_env - muestra el estado del entorno (variables y funciones).
b_getenv - devuelve la estructura de una variable o función del entorno.
b_value - devuelve el valor de una variable.
b_body - devuelve el cuerpo de una función.
b_setenv - cambia el valor de una variable de entorno.
b_setvar - declara o modifica una variable o función.
b_unset - elimina una variable del entorno o en el caso de que sea una variable especial le asigna el valor por defecto.

PARAMETROS

get_parameters - declara las variables y funciones especiales asignando el valor por defecto a cada una de ellas.
save_parameters - salva las variables de entorno que no tienen valores por defecto en el sector de parámetros.

PRINCIPALES

execute - obtiene un comando de la cadena de comandos y lo ejecuta.
monitor - lee una o más líneas y obtiene los tokens correspondientes.
boot - carga y arranca el MINIX, entre otras cosas...
bootstrap - ejecuta una rutina bootstrap para un sistema operativo diferente.
bootminix - arranca el minix.
exec_bootstrap - carga el sector de arranque de un disco o disquete y lo ejecuta.
boot_device - arranca el dispositivo dado como parámetro.
remote_code - comprueba si el MINIX ha dejado alguna orden tras rearrancar.

OTRAS

exit	- sale del modo monitor.
menu	- muestra el conjunto de funciones de menú declaradas.
help	- muestra la pantalla de ayuda.
ls	- muestra el contenido de un directorio.

13. DESCRIPCION DE LAS FUNCIONES PRINCIPALES

La función *boot* tiene dos partes, inicialización e interpretación de comandos. En la inicialización configura la salida por pantalla (*get_video* y *reset_video*) y presenta el mensaje de bienvenida, reubica el programa en la zona final de memoria (*migrate*), averigua cuál es el dispositivo y partición de arranque (*initialize*), y activa la caché de disco (*init_cache*).

La interpretación de comandos es un bucle que se ejecuta continuamente. Mientras haya comandos los ejecuta (*execute*) y en caso contrario se mantiene en espera de nuevos comandos (*monitor*).

La función *monitor* se encarga de leer la entrada estándar y cuando le llega una línea la transforma en una cadena de comandos (*tokenize*), la ejecución de dichos comandos corre a cargo de la función *execute*. Esta es un analizador sintáctico que una vez reconocida la cadena de comandos ejecuta la acción correspondiente.

Una de las acciones principales es el arranque del s.o. MINIX u otro s.o. Tras reconocer el comando *boot* sin parámetros la función *execute* llama a la función *bootminix*, la cual se encarga de realizar la carga y arranque del MINIX tal y como se describió en el apartado 10.

Si el comando *boot* va acompañado del nombre de un dispositivo, se arranca el s.o. instalado en el mismo. Esta tarea es realizada por la función *boot_device* que, entre otras cosas, llama a la función *exec_bootstrap* que carga el sector de arranque de dicho dispositivo en memoria y a su vez invoca a la función *bootstrap* la cual inicia su ejecución.

Antes de que termine la ejecución del s.o. MINIX, éste puede activar una variable que indica al *Boot Monitor* la existencia de una serie de comandos que se han de ejecutar, por ejemplo, el arranque automático del s.o. en caso de un reinicio.

14. BIBLIOGRAFÍA

La bibliografía empleada para desarrollar este tema fue:

- Operating Systems. Design and Implementation. Second edition. Andrew S. Tanenbaum. Albert S. Woodhull. International Edition. Prentice-Hall International, INC.

- Páginas del manual de MINIX y en especial la página /usr/man/man8/monitor.8 (man monitor).

- Capítulo 1, Introducción al MINIX, de los apuntes de la asignatura.