SciTEX 8

1 Introducción

En este documento, explicaré como construir una distribución mínima de Linux, a partir del código fuente (*from scratch*, en inglés). La idea es que sirva de base para construir sobre ella una mayor. A falta de un nombre mejor, la llamaremos SciTEX.

Tengo que explicar el nombre, porque no refleja lo que es, sino lo que en un principio quería llegar a construir. Mi objetivo inicial era crear una distribución centrada en TEX, y que usara SciTE como editor de textos.

Todo sistema operativo consta de tres elementos fundamentales: un sistema de arranque (en inglés *booter*), un núcleo (en inglés *kernel*) y un sistema de ficheros (en inglés *file system*, abreviado `fs`). El *booter* tiene la tarea de arrancar la computadora, cargando el *kernel* en la memoria y pasándole el control. El *kernel* es el programa básico que controla la máquina, haciendo que su *hardware* sea utilizable por el *software*. En el caso de linux, el primer proceso que ejecuta el *kernel* lo obtiene de un fichero denominado `init`, que se encuentra en el sistema de ficheros. Éste es el primer fichero que necesitará nuestro *file system*, pero no será, ni mucho menos, el único.

Para facilitar la tarea de construcción de la distribución, la hemos partido en sus tres elementos. La creación del *booter* se explica en el proyecto Imaging. La creación del *kernel* se explica en el proyecto AA1. En este documento se explica la creación del sistema de ficheros.

Las ventajas de crear tu propia distribución son muchas, ya que puedes adaptarla a tus necesidades concretas, y controlas completamente todo cuanto en ella se incluye. En fin, te dará muchas satisfacciones. Y no es tan difícil.

2 Mi plan

Para poder acometer la tarea, se necesita una computadora con las herramientas adecuadas, entre las que debe haber un compilador de C. A la máquina en la que construiremos la distro, la llamaremos "estación de trabajo" para distinguirla de aquella otra que correrá la distro SciTEX, y que denominaremos "computadora objetivo". De la computadora objetivo únicamente importa el *hardware*, ya que absolutamente todo el *software* se lo vamos a construir nosotros. En cambio, de la estación de trabajo, no importa el *hardware*, siempre que funcione adecuadamente, pero ha de disponer de los programas necesarios para construir el *software* de la otra.

Bueno, en realidad ayuda mucho que el *hardware* de la computadora objetivo sea compatible con el de la estación de trabajo, por dos motivos: 1) Por razones obvias, el compilador de una máquina suele estar configurado para producir binarios que sean ejecutados en esa misma máquina, por lo que, cuando este no es el caso, hay que cuidar los detalles para que los binarios producidos puedan correr sin problemas en la computadora objetivo. 2) Si el *hardware* es compatible, entonces se pueden tomar directamente binarios y librerías de la estación de trabajo a la computadora objetivo.

Mi estación de trabajo es PCBox, que corre debian, en su versión Lenny (5.0) de 64 bits. Y la computadora objetivo será un Acer Aspire One (AA1), de 32 bits. Todo este ejercicio se realizará con estos equipos, de modo que, si tu equipamiento difiere de éste, tendrás que adecuarlo.

Todas las operaciones necesarias para la construcción de la distro SciTEX están automatizadas en un *script* que, si se ejecuta en la estación de trabajo, produce dos resultados: un directorio denominado `initramfs`, y un fichero que contiene el directorio anterior comprimido llamado `SCITEX-8`. El resto del documento será principalmente un comentario a las distintas partes de dicho *script*, denominado: `../input/create.sh`.

3 Preparación

Para empezar hay que preparar el terreno de juego.

Primero se recuerda que hay que ser `root` para ejecutar el *script*, aunque todas las operaciones las puede realizar un usuario ordinario, excepto la de cambiar el propietario de `cpio`.

La función `boldecho` es como `echo` pero en negrita.

Después se definen las variables con las versiones de los programas que se van a utilizar. El *script* admite un parámetro, que sirve para determinar la versión.

Finalmente se crea el directorio de trabajo, si no lo está ya.

../input/create.sh

```
#!/bin/bash

if [ $(whoami) != 'root' ] ; then
 echo 'Only the root can run this script!'
 exit 1
fi

boldecho() {
 echo -e "\033[1m$@\033[0m"
}

BUSYBOX="1.16.1"
VERSION="$1"

CDIR=$(dirname $(readlink -f $0))

if [ -d /mnt/SATA-Projects/SciTeX ]
then
 RDIR="/mnt/SATA-Projects/SciTeX"
else
 if [ ! -d /tmp/SciTeX ] ; then mkdir /tmp/SciTeX ; fi
 RDIR="/tmp/SciTeX"
fi

cd ${RDIR}
if [ ! -d downloads ] ; then mkdir downloads ; fi
if [ ! -d initramroot ] ; then mkdir initramroot ; fi
if [ ! -d CDroot/SciTeX ] ; then mkdir -p CDroot/SciTeX ; fi
if [ ! -d working ] ; then mkdir working ; fi

cd ${RDIR}/initramroot
rm -r *
umask 0022
mkdir -p bin lib dev etc mnt/root proc root sbin sys

cd ${RDIR}/initramroot/dev
mknod console c 5 1
mknod tty c 5 0
for i in $(seq 0 9) ; do
   mknod tty$i c 4 $i
```

```
41 done
42 mknod -m 777 null c 1 3
43 mknod -m 666 zero c 1 5
44
45 ###
```

Y ya podemos comenzar.

4 Los comandos

Las funciones básicas del sistema linux las proporcionan sus comandos, como `cd`, `ls`, y `mkdir`. Una manera fácil de tener un conjunto suficiente y poco espacioso de ellos es la caja de herramientas `busybox`. Como veremos, las nuevas versiones de `busybox` contienen prácticamente todo lo necesario para erigir nuestra distro SciTEX.

La forma de compilar `busybox` es muy similar a la del kernel. También se pueden seleccionar las opciones que, en este caso, consisten básicamente en elegir qué comandos queremos que se incluyan. Para evitar problemas con las librerías, es importante elegir la opción `static`.

Haciendo `make defconfig` se genera la versión por defecto. Esta versión contiene todos los comandos de `busybox`, excepto algunos muy específicos de *debugging*.

La compilación propiamente dicha la hace `make`. Con las variables consigo generar un binario de 32 bits, aunque mi estación de trabajo es de 64 bits.

Y `make install` hace algo muy interesante. Porque, en realidad, `busybox` es un único programa, pero que actúa de muchos modos diferentes, dependiendo de cómo sea llamado. Así que, por ejemplo, si se crean dos enlaces simbólicos a `busybox`, uno llamado `ls` y otro llamado `echo`, entonces, cuando se llama a `ls`, `busybox` responde con el listado del directorio actual, mientras que, si se llama a `echo`, `busybox` actúa como un eco, devolviendo el argumento. Pues bien, lo que hace `make install` es crear, en los directorios `/bin`, `/sbin`, `/usr/bin` y `/usr/sbin`, los enlaces simbólicos correspondientes a las capacidades compiladas, de manera que todo funcione como si efectivamente dispusiésemos de todos los programas.

../input/create.sh

```
46 #BUSYBOX
47 boldecho "Busybox"
48 # Getting busybox, if needed
49 cd ${RDIR}/downloads
50 if [ ! -f busybox-${BUSYBOX}.tar.bz2 ] ; then
51  echo 'Getting busybox ...'
52  wget http://www.busybox.net/downloads/busybox-${BUSYBOX}.tar.bz2
53  cd ${RDIR}/working
54  tar xjf ../downloads/busybox-${BUSYBOX}.tar.bz2
55 fi
56
57 echo 'Configuring busybox ...'
58 cd ${RDIR}
59 if [ ! -f working/busybox-${BUSYBOX}/.config ] ; then
60   cp ${CDIR}/busybox-${BUSYBOX}${VERSION}.config working/busybox-
       ${BUSYBOX}/.config
```

```
61 fi
62 cd ${RDIR}/working/busybox-${BUSYBOX}
63 # make mrproper
64 CPPFLAGS=-m32 LDFLAGS=-m32 make menuconfig
65 # CPPFLAGS=-m32 LDFLAGS=-m32 make oldconfig
66 if [ "$(diff -q .config.old .config)" != "" ] ; then
67  echo "   .config was changed, copying to ${CDIR} as new.config"
68  cp .config ${CDIR}/new.config
69 fi
70
71 echo 'Compiling busybox ...'
72 cd ${RDIR}/working/busybox-${BUSYBOX}
73 CPPFLAGS=-m32 LDFLAGS=-m32 make
74
75 echo 'Installing busybox ...'
76 cd ${RDIR}/working/busybox-${BUSYBOX}
77 CPPFLAGS=-m32 LDFLAGS=-m32 make install
78 cd ${RDIR}/initramroot
79 # rooot suid (needed for telnetd, and other applets)
80 chmod 4755 ${RDIR}/initramroot/bin/busybox
81
82 ###
```

5 La configuración

5.1 Mínima

Como lo que estamos creando es un `initramfs`, el primer proceso resulta de ejecutar `/init`. En otros casos, por ejemplo el típico, con un `rootfs` en una partición `ext3`, el primero de los procesos ejecuta `/sbin/init`.

Nuestro `/init` es muy básico y su comportamiento depende del parámetro `root` utilizado al arrancar el kernel. Si se pone `root=LABEL=etiqueta`, entonces busca una partición denominada `etiqueta` e intenta montarla en `/mnt/root`. Lo mismo, pero con el `UUID` si el parámetro es `root=UUID=98234`. Con el más clásico `root=/dev/sda1`, lo que hace es intentar montar `/dev/sda1` en `/mnt/root`. En cualquiera de estos tres casos, si la operación se completa con éxito, y hay un ejecutable en `/mnt/root/sbin/init`, lo que hace es cambiar el directorio raíz a `/mnt/root`, y ejecutar el `/sbin/init` de la nueva raíz, (o sea, el ejecutable que antes del cambio estaba en `/mnt/root/sbin/init`). Pero, si no se ha encontrado tal ejecutable, entonces ejecuta el `/sbin/init` de `busybox`, quedándonos en SciTEX. Si esto también fallara, que no debería, arranca una *shell* de contingencia.

`../input/create.sh`

```
83 #/init
84 boldecho "/init"
85 cd ${RDIR}/initramroot
86
87 cat > init << "EOF"
88 #!/bin/busybox sh
89
90 # Mount the /proc and /sys filesystems.
91 mount -t proc none /proc
92 mount -t sysfs none /sys
```

```
93 # Mount all the rest.
94 echo 'Waiting to populate /dev'
95 mdev -s
96 echo '/sbin/mdev' > /proc/sys/kernel/hotplug
97
98 for cmd in $(cat /proc/cmdline) ; do
99    case $cmd in
100       root=*)
101          type=$(echo $cmd | cut -d= -f2)
102          if [ $type == "LABEL" ] || [ $type == "UUID" ] ; then
103              uuid=$(echo $cmd | cut -d= -f3)
104              mount -o ro $(/sbin/findfs "$type"="$uuid") /mnt/root
105          else
106              mount -o ro $type /mnt/root
107          fi
108          ;;
109    esac
110 done
111
112 if test -x /mnt/root/sbin/init ; then
113    # Clean up.
114    umount /proc
115    umount /sys
116    # Boot the real thing.
117    exec /sbin/switch_root /mnt/root /sbin/init
118 else
119    if test -x /sbin/init ; then
120       echo "I will start SciTeX!"
121       exec /sbin/init
122    else
123       echo "I couldn't find init. Dropping you to a shell."
124       echo "Type busybox to see what commands are available."
125       exec /bin/sh
126    fi
127 fi
128 EOF
129
130 chmod +x init
131
132 ###
```

5.2 Expansiones

La versión mínima de SciTEX no tiene `/etc/inittab`, de modo que `busybox` utiliza su comportamiento por defecto. Hay perchas que pueden utilizarse para expandir la versión mínima. Después veremos una.

`../input/create.sh`

```
133 #Versions
134 cd ${CDIR}
135 if test -f SCITEX${VERSION} ; then
136  boldecho "including SCITEX${VERSION}"
137  . SCITEX${VERSION}
138 fi
139
```

```
140 ###
```

6 La imágen

La cosa, desde luego, puede complicarse cuanto se quiera, pero ya es hora de empaquetar el sistema de ficheros para probar nuestra micro-distro SciTEX.

../input/create.sh

```
141 #Initramfs
142
143 boldecho "initramfs"
144 cd ${RDIR}/initramroot
145 find . -print0 | cpio -ov -0 -H newc -R 0:0 | \
146                  gzip -9 > ${RDIR}/CDroot/SciTeX/SCITEX${VERSION}.gz
147
148 exit 0
149 ###
```

El resultado de todo este proceso es un único fichero, pero que contiene todo un sistema de ficheros en su interior: `${RDIR}/CDroot/SciTeX/SCITEX${VERSION}.gz`

7 La versión 8

La versión 8, o sea, SciTEX 8, cuelga de la percha el fichero `../input/SCITEX8`.

7.1 El sistema de ficheros raíz

Lo primero es crear los directorios adicionales necesarios.

../input/SCITEX8

```
1 #SCITEX8
2 #ROOTFS
3
4 echo 'Creating some more dir nodes needed by SCITEX ...'
5 cd ${RDIR}/initramroot/
6 mkdir -p etc/init.d home/papa var/log var/run usr/share/udhcpc www/cgi-bin
7
8 ###
```

7.2 Init

Luego hay que inicializar. Lo hacen dos ficheros:

- `/etc/inittab` que controla el comportamiento del `/sbin/init` de `busybox`, y
- `/etc/init.d/rcS`, que es llamado al arrancar desde el fichero anterior.

../input/SCITEX8

```
9 #INIT
10
11 echo 'Writing inittab ...'
12 cd ${RDIR}/initramroot/etc
13 cat > inittab <<EOF
```

```
::sysinit:/etc/init.d/rcS
tty1::respawn:/bin/login
tty2::respawn:/bin/login
::sysinit:/bin/touch /var/log/messages
::respawn:/sbin/syslogd -n -m 0
::respawn:/sbin/klogd -n
tty3::respawn:/usr/bin/tail -f /var/log/messages
::ctrlaltdel:/sbin/reboot
::restart:/sbin/init
::shutdown:/usr/bin/killall klogd
::shutdown:/usr/bin/killall syslogd
::shutdown:/sbin/swapoff -a
::shutdown:/bin/umount -a -r
EOF
chmod +x inittab

echo 'Writing /etc/init.d/rcS ...'

cd ${RDIR}/initramroot/etc/init.d
cat > rcS << EOF
#!/bin/sh

PATH=/usr/bin:/usr/sbin:/sbin/:/bin:/usr/local/bin:.
export PATH

# creating some convenient soft links
/bin/ln -sf /proc/mounts /etc/mtab

/bin/hostname SciTeX$VERSION

# start networking (loopback and eth0)
/sbin/ifconfig lo 127.0.0.1 netmask 255.0.0.0
/sbin/udhcpc -b

# starting httpd
/usr/sbin/httpd -h /www

# starting telnetd
/bin/mkdir /dev/pts
/bin/mount -t devpts devpts /dev/pts
/usr/sbin/telnetd -l /bin/login

# loading workstation keyboard
/sbin/loadkmap < /etc/sp_kmap

# setting time & date
/usr/sbin/rdate ntp.escomposlinux.org

/bin/echo "Bienvenido a SciTeX"
/bin/cat /etc/issue
EOF
chmod +x rcS

###
```

7.3 Teclado

Para facilitar el uso del teclado español. En realidad no toma el teclado español, sino aquel que esté utilizando la estación de trabajo, sea el que sea, y se guarda con el nombre `/etc/sp_kmap`. Al arrancar, `/etc/init.d/rcS` carga el teclado que esté en `/etc/sp_kmap`.

`../input/SCITEX8`

```
68 #KEYBOARD
69 # Take the workstation key map to the target machine
70 cd ${RDIR}/initramroot/etc
71 busybox dumpkmap > sp_kmap
72
73 ###
```

7.4 Particiones

Se escribe un `/etc/fstab` mínimo.

`../input/SCITEX8`

```
74 #FSTAB
75 echo 'Writing /etc/fstab ...'
76 cd ${RDIR}/initramroot/etc
77 cat > fstab << EOF
78 proc /proc proc defaults 0 0
79 sysfs /sys sysfs noauto 0 0
80 devpts /dev/pts devpts defaults 0 0
81 EOF
82
83 ###
```

7.5 Usuarios

Se definen dos usuarios: `papa` y `root`. Las contraseñas iniciales coinciden con los nombres, así que, por seguridad, deben cambiarse.

`../input/SCITEX8`

```
84 #USERS
85 echo 'Creating users ...'
86
87 # /etc/passwd
88 cd ${RDIR}/initramroot/etc
89 cat > passwd << EOF
90 root:x:0:0:root:/root:/bin/sh
91 papa:x:1000:1000:papa:/home/papa:/bin/sh
92 EOF
93 chmod 640 passwd
94
95 # /etc/shadow (root.root, papa.papa)
96 cat > shadow << "EOF"
97 root:$1$WU3/ZX80$5XX5jOlecI0AqgQ75wr4F0:12439:0:99999:7:::
98 papa:$1$mhZPo38r$XqXwLC3DnJOC9GF1hM4/z0:11880:0:99999:7:-1:-1:0
99 EOF
100 chmod 640 shadow
```

```
101
102 # /etc/group
103 cat > group <<EOF
104 root:x:0:root
105 bin:x:1:
106 sys:x:2:
107 mem:x:3:
108 tty:x:4:
109 tape:x:5:
110 daemon:x:6:
111 floppy:x:7:
112 disk:x:8:
113 lp:x:9:
114 dialout:x:10:
115 audio:x:11:
116 video:x:12:
117 utmp:x:13:
118 usb:x:14:
119 cdrom:x:15:
120 papa:x:1000:papa
121 EOF
122 chmod 640 group
123
124 ###
```

7.6 Mensaje inicial

Es un mensaje que se muestra al arrancar. Pueden ponerse indicaciones de uso.

../input/SCITEX8

```
125 #ISSUE
126 cd ${RDIR}/initramroot/etc
127 cat > issue << EOF
128
129 Usuarios/claves iniciales: root/root papa/papa.
130 Teclea busybox para ver los comandos disponibles.
131
132 EOF
133
134 ###
```

7.7 Perfiles

Para que el *shell* se comporte adecuadamente.

../input/SCITEX8

```
135 #PROFILES
136 # /etc/profile
137 cd ${RDIR}/initramroot/etc
138 cat > profile << EOF
139 PATH=/usr/sbin:/usr/bin:/sbin:/bin
140 export PATH
141 EOF
142 # /home/papa
143 cd ${RDIR}/initramroot/home/
```

```
144 chown papa:papa papa
145 cd papa
146 touch .ash_history
147 echo 'export PATH=.:/usr/bin:/bin' > .ashrc
148 ln -s .ashrc .shrc
149 ln -s .ashrc .bashrc
150 chown papa:papa .ash_history .ashrc .shrc .bashrc
151
152 ###
```

7.8 Bitácoras

Hay que gestionar los ficheros de errores, o bitácoras.

../input/SCITEX8

```
153 #LOGS
154 cd ${RDIR}/initramroot/var
155 touch run/utmp log/{btmp,lastlog,wtmp}
156 chgrp utmp run/utmp log/lastlog
157 chmod 664 run/utmp log/lastlog
158
159 ###
```

7.9 Direcciones

Fichero que dirige al comando que pide una dirección IP a la red.

../input/SCITEX8

```
160 #UDHCPC
161 cd ${RDIR}/initramroot/usr/share/udhcpc
162 cat > default.script << "EOF"
163 #!/bin/sh
164
165 # udhcpc script edited by Tim Riker <Tim@Rikers.org>
166
167 [ -z "$1" ] && echo "Error: should be called from udhcpc" && exit 1
168
169 RESOLV_CONF="/etc/resolv.conf"
170 [ -n "$broadcast" ] && BROADCAST="broadcast $broadcast"
171 [ -n "$subnet" ] && NETMASK="netmask $subnet"
172
173 case "$1" in
174 deconfig)
175 /sbin/ifconfig $interface 0.0.0.0
176 ;;
177
178 renew|bound)
179 /sbin/ifconfig $interface $ip $BROADCAST $NETMASK
180
181 if [ -n "$router" ] ; then
182 echo "deleting routers"
183 while route del default gw 0.0.0.0 dev $interface ; do
184 :
185 done
186
```

```
187 metric=0
188 for i in $router ; do
189 route add default gw $i dev $interface metric $((metric++))
190 done
191 fi
192
193 echo -n > $RESOLV_CONF
194 [ -n "$domain" ] && echo search $domain >> $RESOLV_CONF
195 for i in $dns ; do
196 echo adding dns $i
197 echo nameserver $i >> $RESOLV_CONF
198 done
199 ;;
200 esac
201
202 exit 0
203 EOF
204 chmod +x default.script
205
206 ###
```

7.10 Web

SciTEX arranca un servidor de páginas web, capaz de responder formularios (*http forms*), y algo más.

Porque es la semilla de una micro-wiki. Si se añade a una página el formulario:

```
<form name="source" method="post" action="/cgi-bin/source">
    <input type="submit" name="Source" value="Edite">
</form>
```

entonces, cuando se aprieta el botón `Edite`, se ejecuta el *script* `/www/cgi-bin/source`, que muestra las fuentes de la página en un área editable de texto. Y, en la nueva página, hay otro botón que, si se aprieta, ejecuta `/www/cgi-bin/modify`, que modifica el fichero inicial.

`../input/SCITEX8`

```
207 #WEB
208 echo 'Installing Web Server ...'
209
210 # /www/index.html
211 cd ${RDIR}/initramroot/www
212 cat > index.html << EOF
213 <html>
214 <head><title>SciTeX</title></head>
215 <body>
216  <h1>SciTeX</h1>
217  <p>Esto es SciTeX</p>
218  <p>Pronto ser&aacute; m&aacute;s</p>
219  <hr />
220  <p><a href="cgi-bin/test"><code>env</code></a></p>
221  <form name="form1" method="post" action="/cgi-bin/test">
222   <p>Comando: <input type="text" name="command" />
223   <input type="submit" name="submit" value="Ejecutar" /></p>
224  </form>
```

```
 <p><a href="form-post.html">Otra p&aacute;gina</a></p>
 <hr />
 <p>
 <form name="source" method="post" action="/cgi-bin/source">
  <input type="submit" name="Source" value="See source">
 </form>
 </p>
 <hr />
 <p>RMCG</p>
</body>
</html>
EOF

# /www/cgi-bin/test
cd ${RDIR}/initramroot/www/cgi-bin
cat > test <<"EOF"
#!/bin/sh
read QUERY_STRING
eval $(echo "$QUERY_STRING"|awk -F'&' '{for(i=1;i<=NF;i++){print $i}}')
cmd=$(httpd -d "$command")
echo "Content-type: text/html"
echo ""
echo "<html><head><title>Consola HTML</title></head>"
echo "<body>"
echo "<pre>"
if [ "$cmd" == "" ] ; then
 echo "<strong>env</strong>"
 echo "$(env)"
else
 echo "<strong>$cmd</strong>"
 echo "$($cmd)"
fi
echo "</pre>"
echo "</body></html>"
EOF
chmod +x test

# /www/form-post.html
cd ${RDIR}/initramroot/www
cat > form-post.html << EOF
<html>
<head><title>Ejemplo de Formulario</title></head>
<body>
<form name="form1" method="post" action="/cgi-bin/test-post">
<p>Text field<input name="Text_Field" type="text" id="Text_Field"></p>
<p>Radio button</p>
<p><input name="Radio_Button" type="radio" value="1"> 1 </p>
<p><input name="Radio_Button" type="radio" value="2"> 2 </p>
<p><input name="Radio_Button" type="radio" value="3"> 3 </p>
<p>Some text</p>
<p><textarea name="Text_Area" id="Text_Area"></textarea></p>
<p> </p>
<p><input type="submit" name="Submit" value="Submit">
<input type="reset" name="Reset" value="Reset">
</p>
<p> </p>
```

```
281 </form>
282 <hr /><p>
283 <form name="source" method="post" action="/cgi-bin/source">
284  <input type="submit" name="Source" value="See source">
285 </form>
286 </p>
287 </body></html>
288 EOF
289
290 # /www/cgi-bin/test-post
291 cd ${RDIR}/initramroot/www/cgi-bin
292 cat > test-post  <<"EOF"
293 #!/bin/sh
294 read QUERY_STRING
295 eval $(echo "$QUERY_STRING"|awk -F'&' '{for(i=1;i<=NF;i++){print $i}}')
296 echo "Content-type: text/html"
297 echo ""
298 echo "<html><head><title>Resultado de CGI (con POST)</title></head>"
299 echo "<body>"
300 echo ""
301 echo "<h1>Variables de entorno</h1>"
302 echo ""
303 echo "<pre>$(env)</pre>"
304 echo ""
305 echo "<hr />"
306 echo ""
307 echo "<h1>Variables del Formulario</h1>"
308 echo ""
309 echo "<par>Text_Field=<code>$(httpd -d $Text_Field)</code></par>"
310 echo "<br />"
311 echo "<par>Radio_Button=<code>$Radio_Button</code></par>"
312 echo "<br />"
313 echo "<par>Text_Area="
314 echo "<pre>$(httpd -d $Text_Area)</pre></par>"
315 echo "</body></html>"
316 EOF
317 chmod +x test-post
318
319 # /www/cgi-bin/source
320 cd ${RDIR}/initramroot/www/cgi-bin
321 cat > source  <<"EOF"
322 #!/bin/sh
323 read QUERY_STRING
324 eval $(echo "$QUERY_STRING"|awk -F'&' '{for(i=1;i<=NF;i++){print $i}}')
325 SOURCE=$(echo $HTTP_REFERER | sed "s=http://$HTTP_HOST=/www=")
326 echo "Content-type: text/html"
327 echo ""
328 echo "<html><head><title>Source of $HTTP_REFERER</title></head>"
329 echo "<body>"
330 echo "<h1>Source of $HTTP_REFERER</h1>"
331 echo "<hr />"
332 echo "<form name=\"modify\" method=\"post\" action=\"/cgi-bin/modify\">"
333 echo "<p><pre>"
334 echo "<textarea name=\"Contents\" rows=\"15\" cols=\"60\">"
335 cat $SOURCE | sed -e "s/\&/\&amp;/g" -e "s/</\&lt;/g"
336 echo "</textarea>"
```

```
337 echo "</pre></p>"
338 echo "<hr />"
339 echo "<p>"
340 echo "<input type=\"hidden\" name=\"FileName\" value=\"$SOURCE\">"
341 echo "<input type=\"submit\" name=\"Submit\" value=\"Modify\">"
342 echo "<input type=\"reset\" name=\"Reset\" value=\"Reset\">"
343 echo "</p>"
344 echo "</form>"
345 echo "</body></html>"
346 EOF
347 chmod +x source
348
349 # /www/cgi-bin/modify
350 cd ${RDIR}/initramroot/www/cgi-bin
351 cat > modify <<"EOF"
352 #!/bin/sh
353 read QUERY_STRING
354 eval $(echo "$QUERY_STRING"|awk -F'&' '{for(i=1;i<=NF;i++){print $i}}')
355 echo $(httpd -d $Contents) > $(httpd -d $FileName)
356 echo "Content-type: text/html"
357 echo ""
358 echo "<html><head><title>Modified $(httpd -d $FileName)</title></head>"
359 echo "<body>"
360 echo "<h1>Modified $(httpd -d $FileName)</h1>"
361 echo "<hr />"
362 echo "<p><pre>"
363 cat $(httpd -d $FileName) | sed -e "s/\&/\&amp;/g" -e "s/</\&lt;/g"
364 echo "</pre></p>"
365 echo "<hr />"
366 echo "</body></html>"
367 EOF
368 chmod +x modify
369
370
371 ###
```

8 Uso

Para compilarlo hay que hacer:

```
# cd ~/src/projects/SciTeX
# input/create.sh 8
```

Para probarlo sin salir del sistema operativo, se puede usar `qemu`. Para simplificar su uso he creado el script `run.sh`, que se usa así:

```
$ cd ~/src/projects/SciTeX
$ input/run.sh 8
```

El *script* es así:

../input/run.sh

```
1  #!/bin/bash
2
3  VERSION="$1"
4  KERNEL="vmlinuz-linux-2.6.33.2-AA1"
5
6  if [ -d /mnt/SATA-Projects ]
7  then
8   RDIR="/mnt/SATA-Projects"
9  else
10  RDIR="/tmp"
11 fi
12
13 qemu -localtime \
14      -kernel ${RDIR}/AA1/CDroot/boot/$KERNEL \
15      -initrd ${RDIR}/SciTeX/CDroot/SciTeX/SCITEX$VERSION.gz \
16      -append "root=LABEL=Linuxboot" \
17      -hda ${RDIR}/Imaging/output/SD-ext.img
18
```