Instalación de Gentoo Linux para el Via pc2500 con carga por red (LAN boot)

Guillermo Valdez Lozano

12 de noviembre de 2007

Índice general

Capítulo 1

A cerca de este manual

Esta es una guía sobre la instalación de Gentoo Linux para la computadora Via pc2500 con la modalidad de que este equipo no tenga disco duro y cargue el sistema operativo por la red local. ¿Para qué serviría esto? he aquí algunas buenas ideas:

- Armar un salón de clases con 10 o 20 equipos Via pc2500 con un Servidor. Podrá crear un S.O. a la medida de sus necesidades; o también crear distintos S.O. para enseñar distintos paquetes, por ejemplo, un S.O. con Gnome y otro con KDE.
- Crear un conjunto de equipos que realizen una misma tarea o que trabajen en conjunto como una supercomputadora.
- Como un centro de entretenimiento; si tiene una red local en su hogar, podrá reproducir los archivos multimedia que se encuentren en otra computadora.

Capítulo 2

Requerimientos

2.1. Hardware

Este es el listado del hardware necesario para lograr este proyecto:

- Computadora Via pc2500 con 512 MB de RAM. Será llamado como Cliente.
- Computadora con Gentoo Linux, previamente instalado, con particiones y espacio disponible (mínimo 8 GB). Será llamado como Servidor.
- Switch de red 10/100 mbps.
- Cables de red.

El Servidor no necesariamente es un solo equipo, en mi caso particular, el DHCP trabaja en un equipo y los demás servicios en otro. Adapte este procedimiento a la medida de sus necesidades.

2.2. Software

El Software Libre está en constante desarrollo, por lo que las versiones de las aplicaciones que muestro en este manual pueden ser diferentes. El procedimiento mostrado aquí se elaboró con **Gentoo Linux 2007.0** y el kernel **Linux 2.6.22-r5**.

2.3. Conocimiento previo

Espero que tenga experiencia en el uso e instalación del Gentoo Linux[1]. Al hacer este proyecto pondrá en práctica muchos comandos de la consola.

[1]Le recomiendo la presentación Instalación del Gentoo Linux que puede descargar en `http://movimientolibre.com/presentaciones/instalacion-gentoo.html`

2.4. Convencionalismos

En este manual se usan estas convenciones:

- El **Cliente** es la computadora Via pc2500.
- El **Sistema Operativo** o **S.O.** es el GNU/Linux que se entregará por red local al **Cliente**.
- El **Servidor** es la computadora que aloja y entrega el **S.O.**.

Capítulo 3

Planeación del procedimiento

A *grosso modo* estos serán los pasos a realizar:

1. Tomar notas de las características del Cliente.
2. Instalar el S.O. en una partición el Servidor.
3. Instalar y configurar los paquetes necesarios para que el Servidor entregue el S.O. por red local.
4. Actualizar e instalar paquetes adicionales.
5. Cambiar las opciones de entrega para que sea como sólo lectura.

Tenga en cuenta de que en el Servidor configuraremos e instalaremos los siguientes servicios:

- **DHCP** otorga la IP al equipo conectado a la red local y da el parámetro sobre dónde obtener el S.O.
- **TFTP** es el protocolo que entrega el kernel del S.O.
- **NFS** comparte la carpeta raiz del S.O. a los equipos clientes.

En mi caso particular, el Servidor es también un *gateway* con dos tarjetas de red y tiene además los servicios de Samba, Webmin y OpenSSH. De igual forma, Usted puede trabajar con un equipo que ya tenga trabajando; si ese es su caso, no deje de tomar las debidas precauciones para respaldar y asegurar sus procesos.

Capítulo 4

Tomar notas de las características del Cliente

Como es típico en la instalación de Gentoo Linux, el primer paso es tomar nota de las características del equipo.

La computadora Via pc2500 cuenta con:

- Procesador Via C7-D de 1.5 GHz (compatible con x86)
- Capacidades Multimedia y 3D (MMX, SSE, SSE2, SSE3)
- Hasta de 2 GB de RAM DDR2
- Video Via UniChrome con aceleración para decodificar MPEG
- Audio de 6 canales
- Hasta 8 puertos USB
- Tarjeta de red integrada 10/100 mbps
- Dos conectores IDE
- Dos conectores SATA
- Dos ranuras de expansión PCI

Necesitamos conocer un poco más a cerca de las cualidades de cada componente. Para ello lo mejor es arrancar el equipo con un GNU/Linux de tipo *live*, como los que arrancan desde CD (por ejemplo Knoppix[1]) o llave USB (por ejemplo Damn Small Linux[2]).

Con la carga de un GNU/Linux *live* se pueden descubrir las características del procesador:

```
# cat /proc/cpuinfo
processor       : 0
vendor_id       : CentaurHauls
```

[1] http://www.knoppix-es.org
[2] http://damnsmalllinux.org

```
cpu family      : 6
model           : 10
model name      : VIA Esther processor 1500MHz
stepping        : 9
cpu MHz         : 1496.288
cache size      : 128 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge cmov
                  pat clflush acpi mmx fxsr sse sse2 tm pni xstore
bogomips        : 2981.88
```

Con base a lo anterior y las recomendaciones de Gentoo Wiki - Safe CFlags[3] definimos los parámetros de compilación que vamos a incluir más adelante en el archivo **/etc/make.conf**

```
# Instalación basada en el stage3, NO CAMBIAR
CHOST="i686-pc-linux-gnu"

# Via pc2500
#   VIA Esther processor 1500 MHz
#   CPU family 6
#   Model 10
CFLAGS="-march=i686 -mmmx -msse -msse2 -msse3 -O2 -pipe -fomit-frame-pointer"
CXXFLAGS="${CFLAGS}"
```

Usando también un GNU/Linux *live*, vemos los dispositivos PCI detectados:

```
# lspci
```

Estos son los resultados del comando **lspci**, les he dado una mejor presentación:

```
Host bridge:
  VIA Technologies, Inc.: Unknown device 0314
  VIA Technologies, Inc.: Unknown device 1314
  VIA Technologies, Inc.: Unknown device 2314
  VIA Technologies, Inc.: Unknown device 3208
  VIA Technologies, Inc.: Unknown device 4314
  VIA Technologies, Inc.: Unknown device 7314

PCI bridge:
  VIA Technologies, Inc. VT8237 PCI Bridge

IDE interface:
```

[3] http://gentoo-wiki.com/Safe_Cflags

```
  VIA Technologies, Inc. VIA VT6420 SATA RAID Controller (rev 80)
  VIA Technologies, Inc. VT82C586A/B/VT82C686/A/B/VT823x/A/C
      PIPC Bus Master IDE (rev 06)

USB Controller:
  VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller (rev 81)
  VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller (rev 81)
  VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller (rev 81)
  VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller (rev 81)
  VIA Technologies, Inc. USB 2.0 (rev 86)

ISA bridge:
  VIA Technologies, Inc. VT8237 ISA bridge [K8T800 South]

Multimedia audio controller:
  VIA Technologies, Inc. VT8233/A/8235/8237
     AC97 Audio Controller (rev 60)

Ethernet controller:
  VIA Technologies, Inc. VT6102 [Rhine-II] (rev 78)

VGA compatible controller:
  VIA Technologies, Inc.: Unknown device 3344 (rev 01)
```

Todo esto nos guiará para saber cómo configurar el kernel.

Capítulo 5

Instalación del Gentoo Linux a entregar

5.1. Elegir la partición para el S.O.

Es buena idea destinar una partición para instalar el S.O. que entregaremos vía red local. De ésta forma nos será fácil protegerla del sistema operativo propio del Servidor. Por ejemplo, una vez terminado, puede montarse como *sólo lectura*.

Vamos a decidir en cuál partición nos conviene tener el S.O. Ejecute **fdisk -l** para ver un listado de las particiones en los discos duros del Servidor.

```
# fdisk -l
```

Esta es la salida de ese comando en mi Servidor:

```
Disco /dev/hda: 82.3 GB, 82348277760 bytes
255 cabezas, 63 sectores/pista, 10011 cilindros
Unidades = cilindros de 16065 * 512 = 8225280 bytes

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/hda1   *           1           5       40131   83  Linux
/dev/hda2               6         979     7823655   83  Linux
/dev/hda3             980        1953     7823655   83  Linux
/dev/hda4            1954       10011    64725885    5  Extendida
/dev/hda5            1954        2076      987966   82  Linux swap / Solaris
/dev/hda6            2077        3050     7823623+  83  Linux
/dev/hda7            3051        3537     3911796   83  Linux
/dev/hda8            3538        4024     3911796   83  Linux
/dev/hda9            4025        4754     5863693+  83  Linux
/dev/hda10           4755       10011    42226821   83  Linux
```

En **/etc/fstab** están configuradas las instrucciones de montaje. Para listar sólo lo del disco **/dev/hda** ejecute:

```
# cat /etc/fstab | grep /dev/hda
```

Esta es la salida en mi Servidor:

```
/dev/hda1   /boot         ext2      noauto,noatime  1 2
/dev/hda2   /             ext3      defaults        0 1
/dev/hda3   /mnt/extra1   ext3      noauto          0 0
/dev/hda5   none          swap      sw              0 0
/dev/hda6   /mnt/extra2   ext3      noauto          0 0
/dev/hda7   /usr/portage  reiserfs  defaults        0 1
/dev/hda8   /tmp          reiserfs  noatime         0 1
/dev/hda9   /var/tmp      reiserfs  noatime         0 1
/dev/hda10  /home         ext3      defaults        0 1
```

Por último, para ver las particiones montadas de **/dev/hda** en el momento, ejecute:

```
# mount | grep /dev/hda
```

A mí me entrega esto:

```
/dev/hda2  on /            type ext3     (rw)
/dev/hda7  on /usr/portage type reiserfs (rw)
/dev/hda8  on /tmp         type reiserfs (rw,noatime)
/dev/hda9  on /var/tmp     type reiserfs (rw,noatime)
/dev/hda10 on /home        type ext3     (rw)
```

Analizando esta información, podemos disponer de **/dev/hda3** o de **/dev/hda6** para alojar el S.O. Ambas no están siendo usadas y tienen casi los 8 GB de tamaño. No le recomiendo que la partición para el S.O. sea menor a 4 GB, por que una compilación grande pudiera acabar con el espacio libre de la partición y esto provocaría que no se pueda actualizar.

En este manual voy a usar **/dev/hda3** para alojar al S.O. Vamos a modificar **/etc/fstab** con:

```
# nano -w /etc/fstab
```

Y voy a modificar la línea de **/etc/fstab** para que se monte por defecto en **/mnt/via-pc2500**.

```
/dev/hda3   /mnt/via-pc2500   ext3   defaults   0 0
```

Entonces procedemos a formatear y montar la partición:

```
# mkfs.ext3 /dev/hda3
# mkdir /mnt/via-pc2500
# mount /mnt/via-pc2500
```

En este punto deberá tener la partición limpia y montada. A continuación instalaremos un Gentoo Linux en ella.

5.2. Desempacar el Stage 3 y actualizar el portage

Descargue el *Stage 3* y el *snapshot del portage* más recientes[1]. Estos archivos se actualizan frecuentemente, para este proyecto se usaron:

```
# ls -l /home/guivaloz/software/
total 143416
-rw-r--r-- 1 guivaloz users  38784981 sep 25 13:20 portage-20070924.tar.bz2
-rw-r--r-- 1 guivaloz users 107915722 sep 25 13:16 stage3-i686-2007.0.tar.bz2
```

Desempaque el *Stage 3*.

```
# cd /mnt/via-pc2500
# tar xvjpf /home/guivaloz/software/stage3-i686-2007.0.tar.bz2
```

Mi Servidor tiene en **/dev/hda7** al directorio **/usr/portage**. Pensé que sería viable aprovecharlo en la instalación. Pero me trajo varias complicaciones; una de ellas es que al arrancar el NFS y hacer público **/mnt/via-pc2500** no se comparte el contenido de **/mnt/via-pc2500/usr/portage** por que este subdirectorio está en otra partición y NFS no lo permite.

Otro incoveniente es que el *stage3* no tiene instalado *nfs-utils* y no puedo montar una partición remota en el Cliente vía NFS.

Por lo que, mejor desempaque el contenido del portage en **/mnt/via-pc2500/usr/portage**:

```
# cd /mnt/via-pc2500/usr
# mkdir portage
# tar xvjf /home/guivaloz/software/portage-20070924.tar.bz2
```

5.3. Ingresar al nuevo S.O.

Montamos las otras particiones necesarias para ingresar al nuevo Gentoo Linux:

```
# mount -t proc none /mnt/via-pc2500/proc
# mount -o bind /dev /mnt/via-pc2500/dev
```

Copie su configuración para resolver los nombres de dominio, el archivo **/etc/resolv.conf**:

```
# cp /etc/resolv.conf /mnt/via-pc2500/etc/
```

Ejecutamos chroot y los comandos que actualizarán nuestro entorno:

```
# chroot /mnt/via-pc2500 /bin/bash
# env-update
# source /etc/profile
# export PS1="(via pc2500) $PS1"
```

[1] Le recomiendo que los baje desde el servidor del Oregon State University `http://gentoo.osuosl.org`

5.4. Configurar /etc/make.conf

Edite el archivo **/etc/make.conf**:

```
# nano -w /etc/make.conf
```

Este es el contenido que le recomiendo para ese archivo:

```
# This should not be changed unless you know exactly what you are doing.  You
# should probably be using a different stage, instead.
CHOST="i686-pc-linux-gnu"

# Via pc2500
#   VIA Esther processor 1500 MHz
#   CPU family 6
#   Model 10
CFLAGS="-march=i686 -mmmx -msse -msse2 -msse3 -O2 -pipe -fomit-frame-pointer"
CXXFLAGS="${CFLAGS}"

# Dos compilaciones paralelas
MAKEOPTS="-j2"

# Idioma español
LINGUAS="es"

# Variable USE, ajustela a sus necesidades
USE="-doc -ipv6 -kerberos -ldap -java acpi alsa mmx sse sse2 ssl"

# Xorg
INPUT_DEVICES="keyboard mouse evdev"
VIDEO_CARDS="vesa vga"

# Si tiene un servidor rsync con el portage
# PORTDIR=/usr/portage
# DISTDIR=${PORTDIR}/distfiles
# SYNC=rsync://SU_SERVIDOR/gentoo-portage
# FETCHCOMMAND="rsync rsync://SU_SERVIDOR/gentoo-packages/\${FILE} ${DISTDIR}"
```

5.5. Configurar el kernel

Descargue y desempaque las fuentes del kernel con:

```
# emerge gentoo-sources
```

Lanze la interfaz para configurar el kernel con:

```
# cd /usr/src/linux
# make menuconfig
```

Algunos puntos sobresalientes de la configuración del kernel para el Via pc2500 son:

```
Processor type and features
[ ] Symmetric multi-processing support
    Subarchitecture Type (PC-compatible)
    Processor family (VIA C7)
[*] Generic x86 support
    Preemption Model (Voluntary Kernel Preemption (Desktop))
Networking
  [*] Networking support
      Networking options
        [*] TCP/IP networking
        [*]   IP: kernel level autoconfiguration
        [*]     IP: DHCP support
        [*]     IP: BOOTP support
Device Drivers
  Block devices
    <*> Normal floppy disk support
    <*> Loopback device support
    <*> RAM disk support
  ATA/ATAPI/MFM/RLL support
    <*> Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
    <*>   Include IDE/ATA-2 DISK support
    <*>   Include IDE/ATAPI CDROM support
    [*]   Generic PCI bus-master DMA support
    <*>     VIA82CXXX chipset support
  SCSI device support
    <*> SCSI device support
    <*> SCSI disk support
    <*> SCSI CDROM support
  Serial ATA (prod) and Parallel ATA (experimental) drivers
    <*> VIA SATA support
  Network device support
    [*] Network device support
    [*] EISA, VLB, PCI and on board controllers
    <*>   VIA Rhine support
  Character devices
    <*> /dev/agpgart (AGP Support)
    <*>   VIA chipset support
    < > Direct Rendering Manager (XFree86 4.1.0 and higher DRI support
    < >   Via unichrome video cards
  I2C support
    I2C Hardware Bus support
      <*> VIA VT82C596/82C686/82xx and CX700
  Hardware Monitoring support
    <*>   VIA VT8231
  Graphics support
    <*> Support for frame buffer devices
    <*>   VESA VGA graphics support
            VESA driver type (vesafb-tng)
  Sound
    <*> Sound card support
```

```
      <M> Advanced Linux Sound Architecture
      <M>   Sequencer support
      < >     Sequencer dummy client
      <M>   OSS Mixer API
      <M>   OSS PCM (digital audio) API
      [*]     OSS PCM (digital audio) API - Include plugin system
      [*]   OSS Sequencer API
      <M>   RTC Timer support
      [*]     Use RTC as default sequencer timer
            PCI devices
            <M> VIA 82C686A/B, 8233/8235 AC97 Controller
      < > Open Sound System
  USB support
    <*> Support for Host-side USB
    [*]   USB device filesystem
    <*>   EHCI HCD (USB 2.0) support
    <*>   OHCI HCD support
    <*>   UHCI HCD (most Intel and VIA) support
    <*>   USB Printer support
    <*> USB Mass Storage support
File systems
  <*> Second extended fs support
  <*> Ext3 journalling file system support
  <*> Reiserfs support
  [*] Inotify file change notification support
  Network File Systems
    <*> NFS file system support
    [*]   Provide NFSv3 client support
    <*> NFS server support
    [*]   Provide NFSv3 server support
    [*]   Provide NFS server over TCP support
    [*] Root file system on NFS
Cryptographic options
  [*] Cryptographic API
  <M>   Cryptographic algorithm manager
  <M>   MD5 digest algorithm
  <M>   SHA1 digest algorithm
  <M>   SHA256 digest algorithm
  <M>   ECB support
  <M>   CBC support
  <M>   PCBC support
  <M>   Software async crypto daemon
  <M>   DES and Triple DES EDE cipher algorithms
  <M>   AES cipher algorithms
  <M>   CRC32c CRC algorithm
  Hardware crypto devices
  <M> Support for VIA PadLock ACE
  <M>   PadLock driver for AES algorithm
  <M>   PadLock driver for SHA1 and SHA256 algorithms
```

5.6. Script para crear ramdisks

En este proyecto el Via pc2500 no tendrá disco duro por lo que, como ya sabemos, cargaremos el S.O. por la red local; pero también necesitamos que los archivos temporales sean almacenados en la **RAM** del equipo. Así que inmediatamente depués de cargar el kernel, deberán crearse *discos virtuales* que alojen varios directorios de **/var** y **/etc**.

El script debe estar en la raíz del S.O. bajo el nombre **linuxrc**. Para escribirlo ejecute:

```
# nano -w /linuxrc
```

Este es el contenido de **/linuxrc**:

```
#!/bin/bash

#
# Crear ramdisks
#

echo "creating ramdisk for /tmp"
mount -n -t tmpfs tmpfs /tmp

echo "creating ramdisk for /var/lib/init.d"
mount -n -t tmpfs tmpfs /var/lib/init.d

echo "creating ramdisk for /var/log"
mount -n -t tmpfs tmpfs /var/log

echo "creating ramdisk for /var/run"
mount -n -t tmpfs tmpfs /var/run

echo "creating ramdisk for /var/lock"
mount -n -t tmpfs tmpfs /var/lock

echo "creating ramdisk for /root"
mount -n -t tmpfs tmpfs /root

#
# Crear ramdisk para /etc
# Estan comentados para que las instalaciones
# de los paquetes conserven sus configuraciones.
#

#echo "saving /etc from NFS-Mount"
#cd /etc && tar cfj /tmp/etc.tar.bz2 *

#echo "creating ramdisk for /etc"
#mount -n -t tmpfs tmpfs /etc

#echo "populating /etc"
#tar xfj /tmp/etc.tar.bz2 -C /etc
```

```
#rm /tmp/etc.tar.bz2

#
# Ejecutar /sbin/init y continuar con la carga del S.O.
#

exec /sbin/init </dev/console >/dev/console 2>&1
```

Cambie el permiso para que sea ejecutable por el dueño del mismo con:

```
# chmod u+x /linuxrc
```

5.7. Configurar los directorios montados

Edite **/etc/fstab** con:

```
# nano -w /etc/fstab
```

Recuerde que en el Cliente no hay disco duro ni hay particiones locales; sino que el S.O. será montado vía red por NFS en el arranque. Configuramos **fstab** indicándole que la raíz ya estará montada:

```
192.168.0.1:/mnt/via-pc2500  /            nfs    rw,noauto            0 0
proc                         /proc         proc   nodev,nosuid,noexec  0 0
shm                          /dev/shm      tmpfs  nodev,nosuid,noexec  0 0
```

Cambie **192.168.0.1** por la IP de su Servidor.

5.8. Crear las llaves OpenSSH

Si va usar OpenSSH para accesar al Cliente por consola remota, éste tratará de crear las llaves al primer arranque del servicio. Si se modifica la configuración del S.O. a que opere como sólo lectura, el primer arranque marcaría error ante la imposibilidad de guardar las llaves. Podemos adelantar este procedimiento creando las llaves manualmente con los siguientes comandos:

```
# ssh-keygen -t rsa1 -b 1024 -f /etc/ssh/ssh_host_key -N ''
# ssh-keygen -d -f /etc/ssh/ssh_host_dsa_key -N ''
# ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key -N ''
```

5.9. Deshabilitar el arranque de dispositivos de red

Cuando sea el primer arranque, el S.O. tratará de habilitar la tarjeta de red, cosa que ya habrá sucedido puesto que el mismo S.O. se habrá cargado por la red. Para evitar que el S.O. habilite cualquier dispositivo de red durante el arranque, modificamos el archivo **/etc/conf.d/rc**:

```
# nano -w /etc/conf.d/rc
```

Mofifique la siguiente línea:

```
RC_PLUG_SERVICES="!net.*"
```

5.10. Configuraciones adicionales

Gentoo Linux tiene un buen número de configuraciones que necesitan ajustarse. No deje de revisar los siguientes archivos[2]:

```
# nano -w /etc/conf.d/clock
# nano -w /etc/fstab
# nano -w /etc/conf.d/clock
# nano -w /etc/conf.d/consolefont
# nano -w /etc/conf.d/keymaps
# nano -w /etc/conf.d/hostname
# nano -w /etc/hosts
# nano -w /etc/locale.gen
# nano -w /etc/env.d/02locales
```

5.11. Cambie la contraseña de root

Establezca la contraseña del superusuario del S.O. que está instalando:

```
# passwd
```

5.12. Salimos del chroot

En este punto tendrá el S.O. listo. Abandone el **chroot** con el comando **exit**:

```
# exit
```

Y desmonte las particiones que nos hayan ayudado para su instalación.

```
# umount /mnt/via-pc2500/proc
# umount /mnt/via-pc2500/dev
```

Sólo debe estar montada la partición raíz del S.O. en el Servidor. Revise con:

```
# mount | grep /mnt/via-pc2500
/dev/hda3 on /mnt/via-pc2500 type ext3 (rw)
```

El siguiente capítulo tratará sobre la instalación y configuración de los servicios que harán que el Servidor entregue este S.O. por red local.

[2]Encontrará detalles del contenido de estos archivos en `http://movimientolibre.com/presentaciones/instalacion-gentoo.html`

Capítulo 6

Instalación servicios de entrega

6.1. Servidor DHCP

El servicio DHCP se usa principalmente para otorgar direcciones IP dinámicas. En nuestro caso, se vuelve un pieza fundamental, ya que será quien entregue los parámetros para la carga del S.O. por la red local. Instale el servidor DHCP con:

```
# emerge net-misc/dhcp
```

Vamos a crear un nuevo archivo para la configuración con:

```
# nano -w /etc/dhcp/dhcpd.conf
```

En mi caso particular, el Servidor es también el *gateway* de la red local, tiene dos tarjetas de red y es el puente entre la red local y el internet. No hay DNS en la red local, razón por la cual todas las peticiones de nombres son resueltas en otro equipo con dirección **172.16.0.1**.

Mi red local es pequeña y prefiero usar IP estáticas en los equipos más importantes. Haga los cambios necesarios para su red local. Este es el archivo **/etc/dhcp/dhcpd.conf** que uso:

```
# dhcpd.conf

#
# Opciones comunes para todos los equipos
# El DNS es 172.16.0.1
# El gateway es 192.168.0.1
#
option domain-name "sudominio.com.mx";
option domain-name-servers 172.16.0.1;
option routers 192.168.0.1;

#
# Tiempos de durabilidad
#
default-lease-time 3600;
```

```
max-lease-time 3600;

#
# Para deshabilitar las actualizaciones de DNS
#
ddns-update-style none;

#
# Este servidor tiene autoridad como DNS en la red local
#
authoritative;

#
# Declaración para otorgar una IP dinámica a cualquier estación de trabajo
# Las direcciones dinamicas seran desde la 100 a la 250
#
subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.100 192.168.0.250;
}

#
# Declaracion exclusiva para un equipo VIA pc2500
# Use el comando ifconfig para averiguar la dirección MAC del Cliente
# Se le asigna una IP fija y los datos para la carga del S.O.
#
host via01 {
    hardware ethernet XX:XX:XX:XX:XX:XX;
    fixed-address 192.168.0.11;
    filename "pxelinux.0";
    next-server 192.168.0.1;
}
```

Me conviene configurar el DHCP para que sólo atienda las peticiones que vengan de la tarjeta de red que está conectada a la red local:

```
# nano -w /etc/conf.d/dhcpd
```

Cambiando esta línea:

```
DHCPD_IFACE="eth0"
```

Iniciamos el servicio y configuramos que se cargue al arrancar:

```
# /etc/init.d/dhcpd start
# rc-update add dhcpd default
```

Le recomiendo que haga pruebas del servicio DHCP antes de continuar con el procedimiento.

6.2. Servidor TFTP

El servicio **TFTP** es usado por **PXEBOOT** para obtener el binario **pxelinux**.

TFTP son las siglas de *Trivial File Transfer Protocol* (Protocolo de transferencia de archivos trivial). Es un protocolo de transferencia muy simple semejante a una versión básica de FTP. TFTP a menudo se utiliza para transferir pequeños archivos entre ordenadores en una red, como cuando un terminal X Window o cualquier otro cliente ligero arranca desde un servidor de red.

Instale el servicio con:

```
# emerge net-ftp/tftp-hpa
```

Revise la condiguración con:

```
nano -w /etc/conf.d/in.tftpd
```

Configure este servicio para que el directorio que contenga los archivos por entregar sea **/var/tftp/**:

```
# Path to server files from
# Depending on your application you may have to change this.
# This is commented out to force you to look at the file!
INTFTPD_PATH="/var/tftp/"

# For more options, see in.tftpd(8)
# -R 4096:32767 solves problems with ARC firmware, and obsoletes
# the /proc/sys/net/ipv4/ip_local_port_range hack.
# -s causes $INTFTPD_PATH to be the root of the TFTP tree.
# -l is passed by the init script in addition to these options.
INTFTPD_OPTS="-R 4096:32767 -s ${INTFTPD_PATH}"
```

También hay que crear el directorio del que hablamos:

```
# mkdir /var/tftp
```

Arrancaremos el servicio TFTP más adelante, cuando tengamos los archivos que vaya a entregar.

6.3. Servidor NFS

El *Network File System* (Sistema de archivos de red), o **NFS**, es un protocolo de nivel de aplicación, según el *Modelo OSI*. Es utilizado para sistemas de archivos distribuido en un entorno de red de computadoras de área local. Posibilita que distintos sistemas conectados a una misma red accedan a ficheros remotos como si se tratara de locales.

A menos que Usted lo necesite, le recomiendo que deshabilite el soporte para **LDAP** y **Kerberos** ajustando la variable **USE** de **/etc/make.conf**

```
# emerge -pv nfs-utils
```

```
These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-nds/portmap-6.0  USE="tcpd (-selinux)" 22 kB
[ebuild  N    ] dev-libs/libevent-1.3a  436 kB
[ebuild  N    ] net-libs/libnfsidmap-0.19  USE="-ldap" 319 kB
[ebuild  N    ] net-fs/nfs-utils-1.1.0-r1  USE="tcpd -kerberos -nonfsv4" 764 kB
```

Instalamos el paquete y sus dependencias con:

```
# emerge nfs-utils
```

Los directorios que NFS va exportar se configuran en el archivo **/etc/exports**:

```
# nano -w /etc/exports
```

Vamos a declarar dos directorios para compartir. El primero es la raíz del S.O. en **/mnt/via-pc2500**, el segundo es **/usr/portage** porque está contenido en otra partición de nuestro Servidor:

```
# Opciones
#   ro    Sólo lectura
#   rw    Lectura y escritura
#   sync  Obliga a escribir en el disco duro para considerar
#         terminada la operación
#   root_squash      Más seguro, el UID y GID de root se asocian con anonymous
#   no_root_squash   Deshabilita root_squash
#   subtree_check    Más seguro, sólo el directorio exportado, no lo demás
#   no_subtree_check Deshabilita subtree_check

# S.O. para Via pc2500
#   Primero vamos a permitir lectura y escritura
#   para actualizar e instalar más paquetes
/mnt/via-pc2500    192.168.0.*(rw,sync,no_root_squash,subtree_check)

# Directorio del portage
#   Cuando se instale nfs-utils en el cliente, podrá montarse
#   Esta en lectura y escritura, para que puedan bajarse nuevos paquetes
/usr/portage       192.168.0.*(rw,sync,no_root_squash,subtree_check)
```

Los estamos configurando como lectura y escritura para que nos sea posible instalar más paquetes y conservar las configuraciones. Arrancaremos el servicio NFS más adelante.

6.4. Instalación de Syslinux y configuración del PXE

El *Proyecto SYSLINUX* soporta un ligero gestor de arranque o *Bootstrapping*, para *bootear* (o arrancar) un sistema operativo seleccionado en el ordenador, en cualquier sistema Linux. Instale con:

```
# emerge syslinux
```

Ya instalado, tendremos disponible el archivo pxelinux.0:

```
# ls -l /usr/lib/syslinux/pxelinux.0
-rw-r--r-- 1 root root 13320 oct  3 10:42 /usr/lib/syslinux/pxelinux.0
```

PXELINUX es usado para arrancar de un servidor de red que usa un entorno de ejecución de Pre-arranque. Además es usado en unión con un entorno de ejecución de pre-arranque (PXE), en modo ROM, sólo de lectura, sobre una tarjeta de red. De esta manera, los usos de un entorno PXE, pueden ser para la configuración de DHCP o BOOTP, para permitir interconexiones básicas de TCP/IP, que descarguen un programa de arranque vía TFTP. Este programa de booteo carga y configura un kernel(núcleo), de acuerdo a las directrices que también son descargadas del servidor TFTP.

Típicamente PXELINUX es usado para instalaciones Linux desde un servidor de red (servidor web) central o para arrancar workstations (estaciones de trabajo) sin disco. Justo lo que estamos haciendo.

Observe que en la configuración del DHCP aparece la instrucción **filename "pxelinux.0"**. Vamos a poner una copia de **pxelinux.0** en el directorio **/var/tftp** para que TFTP lo envíe al Cliente. Haga la copia con:

```
# cp /usr/lib/syslinux/pxelinux.0 /var/tftp/
```

El PXE espera que exista un archivo de configuración en **pxelinux.cfg/default** relativo al directorio que usa TFTP. Vamos a crearlo:

```
# cd /var/tftp
# mkdir pxelinux.cfg
# nano -w pxelinux.cfg/default
```

Este es el contenido del archivo **/var/tftp/pxelinux.cfg/default**. Ajuste la IP del Servidor **192.168.0.1** si se requiere.

```
DEFAULT diskless
TIMEOUT 200
PROMPT 1

LABEL diskless
    KERNEL diskless
    APPEND ip=dhcp root=/dev/nfs nfsroot=192.168.0.1:/mnt/via-pc2500 init=/linuxrc
```

Observe que hemos definido que el nombre del archivo con el kernel del S.O. será **diskless**. Vamos a copiar el kernel del S.O. a esta ubicación y con este nombre:

```
# cp /mnt/via-pc2500/usr/src/linux/arch/i386/boot/bzImage /var/tftp/diskless
```

Se ve enorme la ruta de donde copiamos el kernel, le explico, el S.O. está en **/mnt/via-pc2500**, las fuentes del kernel están en **usr/src/linux** y después de compilado, el archivo bzImage está en **arch/i386/boot/bzImage**.

6.5. Arranque de los servicios

Espero que haya arrancado y probado el DHCP como se indicó al inicio de este capítulo. Iniciamos NFS y configuramos su carga al encender el Servidor:

```
# /etc/init.d/nfs start
# rc-update add nfs default
```

Revisamos que se estén entregando los directorios configurados en **/etc/exports** con:

```
# showmount -e
Export list for base:
/usr/portage    192.168.0.*
/mnt/via-pc2500 192.168.0.*
```

Procedemos ahora a iniciar el servicio TFTP y configurar su arranque al encender:

```
# /etc/init.d/in.tftpd start
# rc-update add in.tftpd default
```

En este punto la configuración del Servidor está completa y haremos nuestro primer arranque en el capítulo siguiente.

Capítulo 7

Arranque y actualización

7.1. Configure el BIOS del Cliente

Encienda el Cliente e ingrese al BIOS para que configure que este equipo prefiera arrancar el S.O. por red en lugar de buscarlo en sus dispositivos de almacenamiento.

7.2. Primer arranque

Encienda el Cliente y vea con detenimiento los mensajes en su arranque. El hecho de que estemos trabajando con muchos programas y configuraciones nos hacen muy probable que por un simple error no se cargue el S.O. Si ocurre algo, revise y corrija. Si todo marcha bien, podemos inciar con la actualización del S.O.

7.3. Respalde el S.O.

En este punto tiene un S.O. con amplias capacidades de crecimiento. Como es Gentoo Linux podrá hacer con él desde una terminal de texto hasta un centro de entretenimiento completo, conforme vaya instalando aplicaciones.

Esta gran flexibilidad de Gentoo Linux trae consigo una consecuencia seria, aunque la posiblidad es baja, su S.O. podría corromperse por alguna actualización. Eso me pasó al instalar nfs-utils pues el arranque del S.O. por red marcó error.

Si respalda el S.O. podrá regresarse a un cierto momento de la instalación. E incluso experimentar con variantes, como por ejemplo, tener un S.O. con Gnome y otro con KDE. Este script le facilitará esta tarea:

```
#!/bin/bash

#
# Este script sirve para hacer respaldos frecuentes del S.O.
# para el VIA pc2500
#
```

```
# Directorio donde se encuentra el S.O.
DIR_BASE=/mnt/via-pc2500

# Directorio donde guardaremos el respaldo
DIR_BACKUP=/home/guivaloz/software/via-pc2500

# Definir los nombres de los archivos
NOMBRE=so-via-pc2500_$(date +%F)
NOMBRE_TAR=$NOMBRE.tar
NOMBRE_GZ=$NOMBRE.tar.gz

# Verificar que exista el directorio del S.O.
if [ ! -d "$DIR_BASE" ]; then
        echo "Error: el directorio $DIR_BASE no existe."
        exit
fi

# Si no exite el directorio donde guardar el respaldo, lo creamos
if [ ! -d "$DIR_BACKUP" ]; then
        echo "Creando directorio para respaldos."
        mkdir $DIR_BACKUP
        chown guivaloz $DIR_BACKUP
        chgrp users $DIR_BACKUP
        echo ""
fi

# Nos cambiamos al directorio base
cd $DIR_BASE

# Movemos el portage, para que no forme parte del paquete
mv $DIR_BASE/usr/portage $DIR_BASE/portage

# Empaquetamos los siguientes directorios y archivos
# No hay que empaquetar el directorio lost+found del ext3
for ITEM in "bin" "boot" "dev" "etc" "home" "lib" "linuxrc" "mnt" "opt" \
"proc" "root" "sbin" "tmp" "sys" "usr" "var"
do
        echo "Empaquetando $ITEM..."
        tar -rvf $DIR_BACKUP/$NOMBRE_TAR $ITEM
done

# Regresamos el portage
mv $DIR_BASE/portage $DIR_BASE/usr/portage

# Mostrar el tamaño del empaquetado
echo "Ya está empaquetado:"
ls -l $DIR_BACKUP/$NOMBRE_TAR
echo ""

# Comprimir
echo "Comprimiendo..."
gzip $DIR_BACKUP/$NOMBRE_TAR
```

```
# Cambiando permisos
chown guivaloz $DIR_BACKUP/$NOMBRE_GZ
chgrp users $DIR_BACKUP/$NOMBRE_GZ

# Mensaje final
echo "Ya está comprimido:"
ls -l $DIR_BACKUP/$NOMBRE_GZ
echo ""
echo "Script terminado."
```

7.4. Primeras actualizaciones

Entre más tiempo haya entre la fecha del *stage* y la del *snapshot* del *portage*, su sistema requerirá más actualizaciones. De forma muy cautelosa, comienze actualizando el *portage* y el *baselayout*:

```
# emerge -u portage
# emerge -u baselayout
```

Luego instale los servicios para el registro *syslog-ng* y el calendarizador *vixie-cron*:

```
# emerge syslog-ng
# rc-update add syslog-ng default
# emerge vixie-cron
# rc-update add vixie-cron default
```

Para mantener sincronizado el reloj del Cliente, instale y configure *ntp*:

```
# emerge ntp
# rc-update add ntp-client default
```

La actualización que más tiempo toma es la del *glibc* y sus dependencias. En mi caso, tomó más de 4 horas:

```
# emerge -uD glibc
```

No deje de actualizar openssh y active el servicio en el arranque. Así podrá administrar el Cliente cómodamente desde otro equipo.

```
# emerge -uD openssh
# /etc/init.d/sshd start
# rc-update add sshd default
```

Revise y actualize los demás paquetes que requiera el sistema. No deje de respaldar con regularidad:

```
# emerge -pu world
```

En mi caso, hubo un error al actualizar *sys-apps/net-tools*. Me marcó el error *xgettext: error while loading shared libraries: libexpat.so.0: cannot open shared object file: No such file or directory*. Consultando los foros de Gentoo Linux, encontré que la actualización de *libexpat* requería reconstruir las librerías que dependieran de ésta. Este tipo problemas se solucionan con:

```
# revdep-rebuild
# emerge -u net-tools
```

Así es Gentoo Linux. Ámalo u ódialo. Quienes lo amamos, tenemos un característico regocijo cuando nuestro S.O. está actualizado.

7.5. Instalación nfs-utils

Posiblemente haya un bug en el paquete **nfs-utils-1.1.0-r1** por que al instalarlo corrompe el montaje de la raíz del S.O. por red. Opté por instalar una versión anterior, como muestra el siguiente comando:

```
# emerge -pv "=nfs-utils-1.0.12-r1"
```

Para poder montar comparticiones NFS desde otros equipos, necesitamos arrancar el demonio **portmap**:

```
# /etc/init.d/portmap start
# rc-update add portmap default
```

Con el NFS listo, ahora sí me fue posible montar **/usr/portage** desde el Servidor. La configuración de **/etc/fstab** queda como sigue:

```
# Raiz del S.O.
base:/mnt/via-pc2500  /            nfs    rw,sync,noauto  0 0

# Portage
base:/usr/portage     /usr/portage  nfs    rw,sync         0 0

# Propios del sistema
none                  /proc         proc   nodev,nosuid,noexec  0 0
shm                   /dev/shm      tmpfs  nodev,nosuid,noexec  0 0
```

Verá que continuo montando las particiones como *lectura y escritura*, esto hasta que termine de instalar paquetes.

7.6. Perfil de escritorio

Si el propósito del S.O. va a ser una estación de trabajo, le recomiendo que cambie el perfil del sistema. De esta forma muchas configuraciones que usan los entornos gráficos, como por ejemplo, que la variable *USE* active las *X*, estarán habilitadas por defecto.

Por el contrario, si el S.O. sólo va llegar a consola y ejecutará servicios como ser servidor *http* o bases de datos, no active el perfil *desktop*.

Para hacer el cambio, instale el paquete *eselect* y elija el perfil *desktop*:

```
# emerge eselect
# eselect profile list
# eselect profile set default-linux/x87/2007.0/desktop
# eselect profile show
```

7.7. Interfaz gráfica

El comando **lspci** nos identifica el video y los *chipsets* del **VIA pc2500**, como se ve en las siguientes líneas de salida de ese comando:

```
00:00.0 Host bridge: VIA Technologies, Inc. CN700/VN800/P4M800CE/Pro Host Bridge
00:01.0 PCI bridge:  VIA Technologies, Inc. VT8237 PCI Bridge
01:00.0 VGA compatible controller: VIA Technologies, Inc. UniChrome Pro IGP (rev 01)
```

Al momento de escribir esta documentación, no había soporte para el componente de video **UniChrome Pro** en el paquete **xorg-x11-7.2**. Si intenta usar el driver **via**, marcará error porque este controlador no soporta estos chips.

De todas formas, es posible usar el controlador **vesa** en nuestra instalación de **xorg-x11**, el cual nos permitirá usar las *X's* aunque no se aprovechen todas las capacidades del chip.

Revise la configuración de **/etc/make.conf**, debe tener las siguientes configuraciones:

```
# Agregue el flag X a la variable USE
USE="-doc -ipv6 -kerberos -ldap -java acpi alsa mmx sse sse2 ssl X"

# Agregue vesa a VIDEO_CARDS
INPUT_DEVICES="keyboard mouse evdev"
VIDEO_CARDS="vesa vga"
```

Instale el entorno gráfico *xorg-x11* con la instrucción:

```
# emerge xorg-x11
```

Configure el archivo **/etc/init.d/xorg.conf**, en la sección **Device** use el driver **vesa**.

```
Section "Device"
    Identifier  "Card0"
    Driver      "vesa"
    BusID       "PCI:1:0:0"
EndSection
```

Para comenzar a usar la interfaz gráfica, le recomiendo que comienze con un administrador de ventanas ligero, como el **fluxbox**.

```
# emerge fluxbox

$ echo "fluxbox" > ~/.xinitrc
$ startx
```

7.8. Controlador de video UniChrome

El controlador **OpenChrome** `http://wiki.openchrome.org` puede aprovechar las capacidades de video acelerado para el chip **UniChrome Pro**. Como beneficio, al reproducir un video en el mplayer, el uso del procesador se reduce del 20 % al 10 %.

Para instalar este controlador debemos habilitar un *layout* para el *portage*, esto es, un repositorio alterno para que el comando **emerge** pueda obtener estos paquetes no oficiales.

El primer paso de este proceso es descargar el *snapshot* del *layout*, que está en la página: `http://wiki.openchrome.org/tikiwiki/tiki-index.php?page=Compiling+the+source+code+on+Gentoo`

Luego, desempacamos el *layout* y colocamos sus contenidos en **/usr/local/portage**:

```
# cd ~
# tar xvpf openchrome-20071012.tar.gz
# mkdir /usr/local/portage
# mv ~/openchrome/* /usr/local/portage/
```

Configure **/etc/make.conf**, como se muestra:

```
# Xorg configurado para OpenChrome, agregue via
INPUT_DEVICES="keyboard mouse evdev"
VIDEO_CARDS="vesa vga via"

# Portage overlay para controlador UniChrome
PORTDIR_OVERLAY="/usr/local/portage"
```

El kernel deberá tener desctivado la opción de *Direct Rendering Manager* (DRM). Para revisarlo, ejecute el siguiente comando y vea que NO esté con Y:

```
# cat /usr/src/linux/.config | grep DRM
```

Los paquetes que vamos a instalar están enmascarados, por lo que habrá que agregar las siguientes líneas a **/etc/portage/package.keywords** para que el comando **emerge** nos permita instalarlos:

```
# Controlador video OpenChrome
x11-base/x11-drm
x11-drivers/xf86-video-openchrome
```

Instale los paquetes **x11-base/x11-drm** y **x11-libs/libdrm**, los cuales deberán ser descargados del layout configurado.

```
# FEATURES="-sandbox" emerge x11-base/x11-drm
# emerge x11-libs/libdrm
```

Después de la instalación, podemos probar si se carga exitosamente el controlador. Si el siguiente comando no reporta errores, podemos continuar con el procedimiento:

```
# modprobe -v via
```

Ahora sí, instalamos el controlador:

```
# emerge x11-drivers/xf86-video-openchrome
```

Modificamos la configuración de **/etc/X11/xorg.conf** para que tome el controlador de nombre **via**:

```
Section "Device"
        Identifier  "VIA UniChrome Pro"
        Driver      "via"
        BusID       "PCI:1:0:0"
EndSection
```

E iniciamos el entorno gráfico. Al revisar la bitácora del **Xorg** en **/var/log/Xorg.0.log** encontré estas líneas que nos advierten que este controlador está en desarrollo:

```
(!!) VIA Technologies does not support or endorse this driver in any way.
(!!) For support, please refer to http://www.openchrome.org/ or
(!!) your X vendor.
(!!) (development build, compiled on mié 17 oct 2007 11:00:30 CDT)
```

Más información de este procedimiento lo podrá encontrar en:

- `http://wiki.openchrome.org/`
- `http://gentoo-wiki.com/HOWTO_Unichrome`

Capítulo 8

Modificaciones adicionales

8.1. Home en ramdisk

Cuando haya terminado de actualizar e instalar las aplicaciones que haya elegido, tendrá una estación de trabajo en la que sus archivos personales, los contenidos en **/home**, se almacenarán en el disco duro del servidor. Podrá notar que las luces de las tarjetas de red se encienden cuando abra o guarde un archivo. Esto pudiera causar mucho tráfico en su red y que se tarden las operaciones con sus archivos.

Puede que le convenga que el directorio **/home** sea un ramdisk, es decir, que use la RAM del Cliente para almacenar los archivos que genere con carácter temporal. Tenga en cuenta que al apagar el Cliente o al interrumpirse el suministro eléctrico, perderá los archivos de **/home**.

El tener **/home** en la RAM no es impedimiento para que monte via NFS otra carpeta compartida con derechos de escritura para preservar los archivos que elabore en el Cliente.

Para iniciar este cambio, apague el Cliente y en una consola del Servidor, edite el script **/mnt/via-pc2500/linuxrc**:

```
# nano -w /mnt/via-pc2500/linuxrc
```

Hay que agregar las instrucciones para empacar, crear el *ramdisk* y desempacar **/home**. Le muestro el contenido completo de **/mnt/via-pc2500/linuxrc**

```
#!/bin/bash

#
# Crear ramdisks
#

echo "creando ramdisk para /tmp"
mount -n -t tmpfs tmpfs /tmp

echo "creando ramdisk para /var/lib/init.d"
mount -n -t tmpfs tmpfs /var/lib/init.d

echo "creando ramdisk para /var/log"
mount -n -t tmpfs tmpfs /var/log
```

```
echo "creando ramdisk para /var/run"
mount -n -t tmpfs tmpfs /var/run

echo "creando ramdisk para /var/lock"
mount -n -t tmpfs tmpfs /var/lock

echo "creando ramdisk para /root"
mount -n -t tmpfs tmpfs /root

#
# Crear ramdisk para /etc
# Estan comentados para que las instalaciones
# de los paquetes conserven sus configuraciones.
#

echo "guardando /etc desde NFS-Mount"
cd /etc && tar cfj /tmp/etc.tar.bz2 *

echo "creando ramdisk para /etc"
mount -n -t tmpfs tmpfs /etc

echo "desempacando /etc"
tar xfj /tmp/etc.tar.bz2 -C /etc
rm /tmp/etc.tar.bz2

#
# Home en ramdisk
#

echo "guardando /home desde NFS-Mount"
cd /home && tar cfj /tmp/home.tar.bz2 *

echo "creando ramdisk para /home"
mount -n -t tmpfs tmpfs /home

echo "desempacando /home"
tar xfj /tmp/home.tar.bz2 -C /home
rm /tmp/home.tar.bz2

#
# Ejecutar /sbin/init y continuar con la carga del S.O.
#

exec /sbin/init </dev/console >/dev/console 2>&1
```

8.2. Directorios remotos como sólo lectura

Detenga el servicio NFS:

```
# /etc/init.d/nfs stop
```

Edite el archivo /etc/exports

```
# nano -w /etc/exports
```

Modifique los directorios compartidos:

```
# S.O. para Via pc2500
/mnt/via-pc2500  192.168.0.*(ro,sync,no_root_squash,no_all_squash,no_subtree_check)

# Portage
/usr/portage     192.168.0.*(ro,sync,no_root_squash,no_all_squash,no_subtree_check)
```

Edite también el archivo **fstab** del S.O.:

```
# nano -w /mnt/via-pc2500/etc/fstab
```

Y cambie a sólo lectura los montajes:

```
# Raiz del S.O. noauto,sync,hard,intr,nolock,rsize=8192,wsize=8192
base:/mnt/via-pc2500  /             nfs    ro,sync,noauto  0 0

# Portage
base:/usr/portage     /usr/portage  nfs    ro,sync         0 0

# Propios del sistema
none                  /proc         proc   nodev,nosuid,noexec  0 0
shm                   /dev/shm      tmpfs  nodev,nosuid,noexec  0 0
```

Arranque el servicio NFS:

```
# /etc/init.d/nfs start
```

Y encienda el Cliente.

Apéndice A

Referencias

Hardware:

- http://www.via.com.tw/en/initiatives/empowered/pc2500_mainboard/index.jsp

Software:

- http://www.gentoo.org/
- http://www.openchrome.org/

Manuales de instalación:

- http://gentoo-wiki.com/HOWTO_network_boot_without_write_access_on_server
- http://gentoo-wiki.com/HOWTO_Unichrome
- http://gentoo-wiki.com/HOWTO_Share_Directories_via_NFS
- http://www.viaarena.com/
- http://delta.cs.cinvestav.mx/~fraga/Softwarelibre/Diskless.pdf

Sitio del autor:

- http://movimientolibre.com/