THINSTATION EN LAS AULAS DE LA UCO

(v 1.00 24 de febrero de 2005)

1 INTRODUCCION

El mantenimiento de equipos en las aulas de informática es un asunto que siempre ha traído de cabeza a los servicios de informática de las Universidades. El conjugar por una parte las necesidades docentes que requieren software actualizado y cambiante con las posibilidades del personal dedicado al mantenimiento de las aulas, es una tarea compleja.

Si a esto unimos la variabilidad de los equipos de las aulas, la dispersión de las mismas, la ausencia de horas previstas para el mantenimiento, la obsolescencia de los equipos, las actualizaciones de software y la necesidad de resolver los problemas de funcionamiento de una forma ágil y centralizada, la verdad es que el “puzzle” se complica enormemente.

Afortunadamente existen varias alternativas para diseñar un sistema de aulas que elimine algunos de los inconvenientes anteriores. Si bien ninguna alternativa es perfecta, algunas son mejores que otras y sus ventajas superan ampliamente a sus inconvenientes.

2 ALTERNATIVAS DE DISEÑO

Las alternativas para diseñar un sistema para aulas de informática son tantas como aulas de informática existen. No hay nada que pueda ser considerado estándar y es común que los técnicos usen las herramientas más adecuadas de unas y otras posibilidades adaptándolas a su entorno y necesidades. Unas veces se usan herramientas propietarias y otras de Software libre. Aun así existen unas directrices básicas que se repiten de forma más o menos universal. De forma resumida podemos distinguir las siguientes:

2.1 Equipos autónomos con sistema operativo local

Es la más sencilla de concebir y la más cercana al usuario que va a usar las aulas. Los equipos no varían demasiado del que el alumno puede tener en su casa y la forma de trabajo con el equipo recién instalado puede parecerle amigable.

Lamentablemente esta alternativa es la que más problemas aglutina desde el punto de vista de su gestión y mantenimiento. Los problemas son:

- Requiere un trabajo enorme para cada equipo en su instalación. Necesita unos recursos humanos desorbitados.
- No permite la actualización de los equipos de una forma rápida ni centralizada.
- No permite casi ningún control sobre el uso de los equipos.
- Los equipos están expuestos al uso malintencionado o inconsciente de los usuarios lo cual puede hacer que dejen de ser operativos.
- A la larga este sistema consigue que las aulas más nuevas tengan una configuración y las mas antiguas otra, confundiendo al usuario y al propio personal de informática.

2.2 Equipos autónomos con replicación de discos

Es una variante del anterior donde la instalación no se produce manualmente sino mediante herramientas de replicación de discos como Ghost, Imagedrive, System Imager, etc. Eliminada la necesidad de que el personal de informática introduzca uno tras otro decenas de CDs para instalar software, aparecen de nuevo multitud de problemas:

- La replicación es un proceso largo que consume mucho ancho de banda de la red y que normalmente no puede realizarse en horario laboral al coincidir este con el horario de uso del aula.
- Los equipos siguen estando expuestos al mal uso por parte de los usuarios. Si el mal uso desencadena en que el equipo no funcione bien, la única solución es volver a replicarlo completo, normalmente esto es más rápido que intentar averiguar el problema.
- Las actualizaciones de software implican una nueva replicación.

2.3 Equipos autónomos con software a través de la red

En este modelo los equipos ejecutan uno o varios sistemas operativos localmente. El sistema operativo está también instalado localmente pero las aplicaciones se encuentran en unidades de red. Es habitual que los sistemas operativos locales se repliquen como en el caso anterior. Los problemas que aparecen son:

- No todas las aplicaciones permiten su ejecución desde unidades de red.
- Seguimos teniendo el mismo problema de replicación de equipos y mal uso. En este caso el problema se limita a lo instalado localmente. Por tanto hemos evitado el problema cuantitativa pero no cualitativamente.

2.4 Terminales de aplicaciones

Son la panacea de los informáticos encargados de las aulas. Los terminales no ejecutan aplicaciones localmente, solo los clientes de conexión a los servidores de aplicaciones. Estos últimos suelen estar bien protegidos contra usos malintencionados y bajo la exclusiva supervisión del personal de sistemas. Existe un control absoluto sobre lo que pasa en las aulas ya que realmente en ellas no pasa nada, todo pasa en los servidores que estas usan. A este tipo de terminales evolucionados de los primeros terminales *"tontos"* se les denomina **THINCLIENTS**. Aun así seguimos manteniendo algunos problemas:

- Algunas aplicaciones no son adecuadas para su uso en un entorno de terminal. Las aplicaciones multimedia o las que *"devoran"* recursos de los servidores son de este tipo.
- Las tendencias de programación actuales no facilitan el trabajo en estos entornos. Los fabricantes de software desarrollan aplicaciones que sacrifican la agilidad por la vistosidad. Tecnologías como **java**, **flash**, etc. son auténticas devoradoras de recursos para al fin y al cabo hacer lo mismo que se podía haber hecho con otras herramientas de una forma mucho más óptima.
- Los entornos de terminal son muy dependientes de la infraestructura informática. Si la red no funciona o si los servidores no son adecuados, el sistema se viene abajo.

2.5 Entornos mixtos

A la vista de todo lo anterior, resulta evidente que el menos malo de los sistemas será el que incorpore lo mejor de cada uno de ellos. Las claves para el diseño son las siguientes:

- Si mantener uno o varios sistemas operativos en los discos es un problema, no lo hagamos. Es más, incluso quitemos el disco.
- Si el mantenimiento de aplicaciones en unidades de red es posible y sencillo, hagámoslo para todas las que lo permitan.
- Si la replicación de imágenes de disco para que el equipo arranque no es posible dentro del horario del personal, hagamos una imagen lo suficientemente pequeña para que se pueda cargar con cada arranque del equipo de forma rápida.
- Si determinadas aplicaciones colapsan los servidores de aplicaciones, ejecutemos estas localmente en los equipos.
- Si necesitamos mantener una estructura homogénea donde todos los equipos de todas las aulas deben funcionar del mismo modo, usemos un sistema centralizado de configuración de los equipos.
- Si nuestros equipos tienen recursos locales que podemos aprovechar, hagámoslo. Esto es, usemos del disco si lo tienen para labores que agilicen el trabajo sin introducir dependencias del mismo (si se tiene disco se usa, si no se tiene no, si el disco falla el equipo debe seguir funcionando). Lo mismo pasa con la CPU y la memoria del equipo, aprovechémosla pero no hagamos que nuestro sistema dependa de su *"calidad"* en exceso.

2.6 Terminales inteligentes

Si aplicamos las premisas anteriores pasaríamos de tener un cliente ligero (**THINCLIENT**) a un cliente *"inteligente"* (**THINKCLIENT**). El problema que nos surge es si esto existe o bien tenemos que crearlo nosotros mismos.

La respuesta a lo anterior es *"casi"*. Existen *thinclients* que aparte de incluir clientes de conexión a servidores de aplicaciones, permiten la ejecución local de ciertas aplicaciones, normalmente navegadores web. Estos equipos están basados en un *firmware* local y suelen ser caros. Existen otros sistemas basados en equipos normales de consumo que los convierten

en thinclients cambiando el *firmware* por una imagen de arranque que se descarga de la red. Evidentemente son mucho más baratos.

Pero si no nos queremos conformar con la ejecución de unas pocas aplicaciones locales, no nos queda más remedio que fabricarnos nuestro propio sistema o bien extender uno existente. Esto es lo que hemos realizado en el centro de cálculo científico de la UCO. Usamos un sistema de software libre llamado **THINSTATION**, el cual hemos ampliado para dotarle de más funcionalidades locales.

3 THINSTATION

Thinstation (http://thinstation.sourceforge.net) es un proyecto de software libre que nació en mayo de 2003 como evolución de un proyecto anterior llamado **netstation**, el cual dejó de desarrollarse por cuestiones personales de su impulsor. El grupo de usuarios activos de netstation entre los que nos encontrábamos, decidieron impulsar el nuevo proyecto abriendo el número de desarrolladores y haciéndolo más "vivo".

Desde entonces ha llegado a la versión 2.0.2 encontrándose a punto de liberarse la versión 2.1. Los componentes básicos de thinstation son los siguientes:

- **Servidor de arranque:** Normalmente un servidor DHCP y TFTP que puede ser tanto Linux como Windows. Thinstation permite arrancar también de instalaciones locales en el disco duro, Live CDs, Pen drives o cualquier otro disco de estado sólido.
- **Imagen de arranque:** Contiene un kernel de Linux y una imagen initrd empaquetadas en un paquete NBI para su arranque por etherboot o separadas para su arranque por PXE. Existen formas de usar la imagen empaquetada con PXE usando un pequeño cargador previo. El tamaño de la imagen es la clave de thinstation (TS en adelante). Su tamaño oscila de 3Mb a 16Mb. La imagen actual que nosotros usamos no llega a los 8Mb.
- **Paquetes:** TS tiene un diseño modular. Quitando el kernel, el resto de la imagen initrd es un puzzle compuesto por muchas piezas llamadas paquetes. Existe por ejemplo un paquete para cada adaptador de red, otro para cada adaptador gráfico, otro para cada cliente de conexión, etc. Los paquetes pueden depender unos de otros, esto se controla dentro del propio paquete, por lo que no es necesario incluir o conocer todos los paquetes necesarios para ejecutar por ejemplo un cliente ICA, el propio paquete los define.
- **Sistema de generación de imágenes:** Bajo Linux, la distribución de TS nos permite definir en un fichero de configuración (*build.conf*) los paquetes que deseamos y la configuración de las imágenes de arranque. La orden *"build"* nos genera la imagen apropiada para que la coloquemos en nuestro servidor TFTP. Para los que no quieran complicarse la vida o aprender el poco Linux que hace falta, existe una página web (http://ts-o-matic.aaskov.dk/ y algunos *mirrors* más) donde especificamos nuestra configuración y nos descargamos la imagen construida al vuelo.
- **Entorno de desarrollo:** Es una mini distribución de Linux que se ejecuta en un entorno *"chroot'ed"* donde podemos compilar las aplicaciones necesarias para hacer nuestros propios paquetes. No es necesario para la inmensa mayoría de los usuarios, aunque nosotros obviamente sí lo usamos.
- **Ficheros de configuración:** Almacenan variables con las que se toman todas las decisiones sobre cómo arrancan los terminales, como se configuran, que sesiones presentan, etc

A continuación hablaremos con más detalle sobre los paquetes, los ficheros de configuración y las sesiones.

3.1 Paquetes

Ya hemos visto que existen unos paquetes fundamentales que afectan a la configuración del equipo como son los de tarjetas de red y tarjetas de video. Si queremos por comodidad

disponer de una imagen común para todos los equipos, debemos generarla con todos los paquetes necesarios. No es adecuado incluir todos los paquetes posibles de este tipo en la imagen pues aparte de que su tamaño se incrementaría mucho, el arranque de TS se ralentizaría ya que debería probar con todos ellos para quedarse con los adecuados.

Existen otro tipo de paquetes que forman el núcleo de TS. Son el paquete "base", "x-common", etc. Estos incluyen tanto los binarios, librerías y scripts de arranque imprescindibles para que TS (un mini-linux al fin y al cabo) funcione.

Otros paquetes son opcionales. Existen varias categorías:

- **Clientes de conexión:** Citrix ICA, MS Windows terminal services (RDP), Tarantella, X, telnet, tn5250, VMS term, SSH, NX, etc.
- **Clientes de red:** Cliente smb, cliente nfs, etc.
- **Servidores de red:** DHCPD, SMBD, NFSD, TFTPD, WWW, TELNETD, etc. Uno se puede preguntar qué pintan estos en un cliente. Algunos como telnetd y www se utilizan para administración y gestión remota de los clientes. Los demás se usan para poder ver nuestras unidades locales desde los servidores de aplicaciones o incluso actuar como "reenviadores" de imágenes de arranque y configuración al resto de clientes de nuestra red.
- **Utilidades:** Supermount, keymaps, etc. Amplían las posibilidades de thinstation.
- **Clientes locales:** Firefox, Sylpheed, Dillo, E3, Mplayer, Blackbox, etc. Son los que aportan la inteligencia a nuestro equipo, permitiendonos que aparte de establecer conexiones de terminal con nuestros servidores, ejecutemos localmente clientes "pesados" en el propio terminal.

3.2 Ficheros de configuración

Son muchos los ficheros de configuración que TS permite tener, desde ficheros especiíficos para cada equipo (el nombre del fichero incluye la dirección MAC para asociarlo) o ficheros generales para todos los equipos (*thinstation.network*); pero realmente son dos los fundamentales:

- **Thinstation.hosts:** Este fichero alberga la definición de todos los terminales de nuestra red con información sobre su nombre, dirección MAC y grupos a los que pertenece.
- **Thinstation.conf.group-XXX:** Donde *XXX* es el nombre de cada grupo. Son los ficheros de configuración que cada equipo usa en función del grupo o grupos a los que pertenece. Dentro de ellos definimos todas las opciones de configuración de los terminales de dicho grupo.

Por ejemplo, imaginemos que tenemos equipos que disponen de una tarjeta gráfica adecuada y de un monitor capaz de presentar 1024 x 768 puntos en 24 bits. En lugar de definir estos parámetros para cada equipo (cosa que también podríamos hacer), los definimos en un único fichero llamado *thinstation.conf.group-17pulgadas* y asignamos en thinstation.hosts los equipos que queramos a dicho grupo.

Si queremos tener una configuración de inicio de sesiones específica para cada aula, crearíamos un fichero *thinstation.conf.group-AULAXX* para cada aula y asignaríamos los equipos de ese aula al grupo *AULAXX* en el fichero *thinstation.hosts*.

Estos ficheros de configuración se encuentran en la raíz tftp de donde se descarga la imagen el terminal con lo que el administrador tiene un control total sobre el funcionamiento del sistema con solo editar unos ficheros de texto.

La sintaxis de estos ficheros de configuración viene recogida los anexos. Si explicaremos aquí lo que es el concepto de sesión.

3.3 Sesiones

Para TS una sesión consiste en la ejecución de un cliente, normalmente de conexión. Un ejemplo de líneas del fichero de configuración para una sesión ICA contra la aplicación publicada "UCO-ESCRITORIO" usando el ICA browser "UCOWTSDB" sería:

```
SESSION_2_TITLE="WINDOWS"
SESSION_2_TYPE=ica
SESSION_2_ICA_APPLICATION_SET="UCO-ESCRITORIO"
SESSION_2_ICA_SERVER="ucowtsdb.uco.es"
SESSION_2_SCREEN=2
```

Como vemos aparece una línea que especifica que esta sesión se ejecutará en el SCREEN 2. Esto quiere decir que TS usará el display *:2.0* para ella. De esta forma podemos ejecutar cada cliente en un display X diferente y conmutar entre ellos usando las teclas CTRL-ALT-Fx donde Fx es "*2 + el número de screen*" (los dos primeros displays se usan internamente por TS).

Nos podemos preguntar cuantas sesiones podemos tener en nuestro terminal de TS. Pues inicialmente cuantas queramos, solo limitadas por la memoria de nuestro equipo. Lógicamente con más de 10 sesiones se nos acabarían las teclas de función para conmutar entre ellas. Aún en este caso podemos arrancar una sesión con un *windows manager* (**blackbox, icewm**) y ejecutar el resto de sesiones dentro de este *windows manager* usando un solo *display*. O también podemos combinar sesiones con display propio con sesiones dentro de un *windows manager*, todo estará definido en el fichero de configuración.

Es más, aprovechando las características de aplicaciones publicadas de Metaframe, podemos decidir un día que nuestro terminal arranque directamente mostrándonos solo la pantalla de Microsoft Word para un curso, o bien un navegador remoto en servidores Windows, o un navegador remoto en servidores unix/linux o incluso un navegador local. Todo esto simplemente modificando un fichero de texto desde nuestro despacho.

4 THINSTATION EN LA UCO

Thinstation desde un principio se ha adecuado muy bien a nuestro entorno. Con anterioridad usábamos el arranque remoto de equipos mediante el protocolo RPL de Microsoft. Realmente el arranque se producía en MS-DOS y esto nos limitaba en el aprovechamiento de los recursos hardware de los equipos más modernos. De hecho el cliente ICA para msdos no soportaba más de 256 colores, aparte de que está descatalogado. No existen clientes RDP fiables para msdos y no tenemos en principio soporte de dispositivos USB, etc. Esto nos llevó a escoger TS como un entorno más adecuado para nuestras aulas.

La renovación de los equipos para aulas donde todos cuentan con disco local, 256Mb de memoria al menos, cpu's muy rápidas, etc. nos puso en la disyuntiva sobre aprovechar dichos recursos locales o no.

Tal como hemos visto en el apartado 2, las alternativas de tener equipos autónomos aún automatizando la instalación mediante imágenes, resultaban totalmente inadecuadas a nuestro entorno. Ni tenemos disponibilidad de horas en las que podamos replicar los equipos, ni tenemos personal suficiente para realizar este trabajo con la cantidad de aulas que tenemos, ni las necesidades docentes tremendamente cambiantes nos permiten mantener sistemas como estos que suelen ser más "estáticos".

Por tanto tuvimos que decidirnos por un enfoque mixto. Usar TS que tan buenos resultados nos había dado y modificarlo para conseguir una sesión local, en este caso en Linux. De esta forma respondíamos a las peticiones docentes basadas en este sistema operativo.

Haremos aquí un pequeño inciso histórico. Resulta curioso que naciendo Unix (o Linux) con un enfoque basado en servidor y Windows con un enfoque basado en equipo de escritorio, resulte más sencillo y adecuado hacer lo contrario hoy día. La evolución de entornos de

escritorio de Linux (KDE y GNOME principalmente) ha conseguido consumir los recursos de los equipos a un nivel que ni siquiera Windows lo ha hecho. Son entornos tremendamente pesados, devoradores de memoria y consumidores compulsivos de ancho de banda de red cuando se ejecutan desde un servidor. Sin embargo los servicios de Terminal de Windows y sus variantes como Metaframe, han conseguido limitar este consumo y funcionar perfectamente incluso usando líneas de comunicación de baja velocidad (RTC, ADSL, etc.). Aparte los servicios de balanceo de carga de Metaframe y su transparencia para el usuario permiten crear granjas de servidores que soportan cientos de usuarios a base de servidores no demasiado caros. Para conseguir lo mismo en Linux necesitamos tener un servidor tremendamente más caro.

Afortunadamente Linux sigue conservando lo mejor de Unix. Su posibilidad de trabajar con sistemas de ficheros en memoria o por NFS, la deslocalización de aplicaciones y su configuración por ficheros propios para cada una, el control del funcionamiento del sistema y las aplicaciones por variables de entorno y ficheros en el home del usuario, etc. Por todo esto decidimos modificar TS para usar Linux local y usar Windows en sesiones de Terminal.

4.1 Principios de modificación

TS incluye un kernel de Linux completo y moderno pero una imagen initrd ligera. Solo lleva los componentes esenciales para el funcionamiento de los clientes de conexión. La versión de glibc incluida por defecto es la 2.1.3 aunque permite como opción la 2.2.5. La versión actual, 2.3.x incrementa enormemente el tamaño de la imagen de arranque y la complejidad del sistema. Las librerías que incorpora son las mínimas e incluso la shell es busybox que con apenas 300Kb incorpora una shell completa y los comandos más usuales de Linux.

Por tanto éramos conscientes de que deberíamos expandir TS bastante para dar un entorno habitual de Linux. Esta expansión no debe alterar el funcionamiento de TS como thinclient, no debe tocar los paquetes de la distribución de TS ni aumentar demasiado la imagen que se descarga por tftp.

Todo esto lo conseguimos desarrollando paquetes propios de TS e instalando en un export de NFS todo el software adicional.

4.2 Nuevos paquetes desarrollados

4.2.1 Estructura de un paquete

Los paquetes de TS cuentan con una estructura de directorios incluyendo normalmente los siguientes:

- /bin
- /lib
- /etc

El contenido de los mismos será mezclado con el del resto de paquetes en el proceso de construcción de la imagen de TS. En el directorio /etc se encuentran los directorios init.d y rc5.d (o rc0.d si el paquete debe inicializarse pronto). En ellos se encuentran los scripts necesarios para inicializar el paquete.

Como TS usa un file system de tipo *squashfs* en memoria que es de solo lectura para el directorio / y otro de tipo *tmpfs* también en memoria para el directorio /tmp, es necesario en la inicialización del paquete generar ficheros de configuración que habitualmente cuelgan del directorio / , /var, /etc ó /usr; en el directorio /tmp. Esto se consigue mediante enlaces simbólicos dentro de los paquetes que apuntan al directorio /tmp. Por ejemplo, si necesitamos crear *"al vuelo"* un fichero en el directorio /usr/X11R6/lib/xdm, crearemos previamente en el paquete un enlace de este directorio hacia /tmp/usr/X11R6/lib/xdm, y en la inicialización del paquete generaremos el contenido del mismo (ya está en un lugar de lectura/escritura) basándonos en el valor de los ficheros de configuración.

De esta forma nuestro Linux tras arrancar tendrá una estructura propia y particular para cada cliente en función de los ficheros de configuración. Los paquetes que sustentan nuestro Linux local son:

4.2.2 pam_ldap

TS soporta pam de serie pero solo usando el fichero /etc/passwd. Para que nuestro entorno pueda autentificar a nuestros usuarios hemos incorporado la librería **pam-ldap** y los ficheros de configuración necesarios en este paquete aparte de las librerías *nss*. Existe una nueva variable de control en los ficheros de configuración.

```
LDAP_SERVER="server"  # Servidor LDAP a usar
```

4.2.3 rpc_statd

TS incluye soporte *lockd* y *statd* para ficheros locales, pero no lo incluye para ficheros en sistemas de ficheros NFS. Como muchas aplicaciones intentarán usar estas funciones en el *home* del usuario, es necesario incorporarlas a nuestro entorno. No hay variables de configuración asociadas a este paquete.

4.2.4 mount_homes

Para poder montar los *homes* de usuario, usamos los comandos *mount* y *smbmount* incluidos ya en paquetes de TS (el segundo de ellos si incluimos el paquete *smbfs*). Pero previamente debemos enlazar el directorio /home a /tmp/home y buscar alguna forma de montar automáticamente el home del usuario en el momento en que se conecte al sistema y desmontarlo cuando se desconecte. Existen alternativas como automounter pero es complicada su inclusión en TS que ya utiliza supermount para dar soporte a unidades extraíbles.

Intentamos en principio usar el módulo *pam* **pam_mount** hasta comprobar que está lleno de *bugs*. Para simplificar hemos integrado el montaje de los *home* de usuario en el script *Xstartup* de *xdm* y el desmontaje en el *Xreset*, pues ambos se ejecutan como usuario *root* inmediatamente antes y después del *login / logout* del usuario. En el caso del desmontaje es necesario matar antes los procesos que aun tengan abierto el *home* del usuario pues *xdm* funciona en este punto de un modo asíncrono.

Las variables de configuración que emplea este paquete son:

```
FORCE_USE_SMB="ON/OFF" # Forzamos a montar el home por SMB
NFS_HOME_PATH="server:/directorio" # Raíz del home de los por NFS
SMB_HOME_SERVER="Server" # Servidor samba para montar el home
```

En el caso de SMB usamos el mecanismo de directorios *home* de *samba* donde el *home* del usuario es //**Server**/**usuario**. SMB es un mecanismo más seguro que NFS pues requiere clave para proceder al montaje. Hemos parcheado *xdm* para pasar dicha clave desde la ventana *"greeting"* hasta el fichero *Xstartup*. Más adelante desarrollaremos un módulo *pam* específico que lo haga.

El paquete también comprueba que el directorio *home* del usuario se pueda montar por NFS si no hemos forzado a que se monte mediante SMB, consultando los permisos de exportación del servidor NFS. Si no tiene permiso, lo montaría por SMB para proporcionar siempre acceso al home del usuario.

Tenemos que comentar que no es lo mismo montar el home del usuario por un protocolo u otro. SMB no soporta enlaces simbólicos ni cambio de permisos en los ficheros con comandos como *chown* o *chmod*, ni acceso a ficheros que se hayan abierto con la opción O_EXCL, tal como hace la función de la *glibc mkstemp*. Estas cuestiones nos han obligado a parchear algunas aplicaciones (*sylpheed, ROX, etc*) para que funcionen sobre sistemas de ficheros de tipo *smbfs*.

4.2.5 xdm_local

TS soporta de entrada *xdm* pero de forma remota para iniciar una sesión mediante XDMCP en un servidor Unix/Linux. Por tanto hemos incluido en este paquete el cliente *xdm* con soporte *pam*. El script de inicio de la sesión xdm_local es una variación del *script* de inicio de una sesión X normal, donde creamos al vuelo los ficheros de control de *xdm* (*Xserver, Xsession, xdm-config, etc*). Como no tiene sentido usar xdm_local sin tener montado nuestro directorio */usr/local* por NFS, es en este paquete donde lo hacemos y viene controlado por la variable de configuración:

```
NFS_LOCAL=server:export
```

Se incluyen también los enlaces de los shell más comunes a */usr/local/bin/bash*, el fichero */etc/profile*, etc. de forma que el usuario entre con su entorno de escritorio perfectamente configurado. Al usar *icewm* como manejador de ventanas, debemos incluirlo en la generación de la imagen de TS (*icewm* es un paquete estándar de TS).

4.2.6 prepare_hd

Como dijimos en el apartado 2.5, si nuestro equipo tiene un recurso local, lo usaremos pero sin depender de él hasta el punto que si falla, el cliente deje de funcionar. Un buen ejemplo es el disco duro. No pretendemos instalar en el mismo una imagen de los ficheros que necesitamos de forma que se haga imprescindible, pero si lo podemos usar para otros cometidos que aumenten las prestaciones sin introducir más puntos de fallo y de administración. Estos cometidos son:

- **Partición de arranque:** Nos permitirá prescindir de una tarjeta de red con ROM de arranque que soporte PXE, esta mini partición incluye una ROM *etherboot* (http://www.etherboot.org) que soporta todos los adaptadores de red *pci*, aparte de permitirnos en un futuro un hipotético arranque contra otra partición con un sistema operativo local (esta partición incluye *lilo*).
- **Partición de swap:** Permitirá que nuestro Linux local se comporte como un Linux completo donde se puedan ejecutar aplicaciones con mucha demanda de memoria.
- **Partición de caché NFS:** *Cachefs* es una interesante característica de sistemas operativos como **Solaris** o **HpUx** que actualmente se soporta en Linux mediante parches del núcleo. Básicamente consiste en crear un sistema de ficheros en el disco local que sirve como caché de otro montado por NFS. De esta manera las peticiones NFS se consultarán primero en el disco local y si no están, se conseguirán del *export* NFS siendo cacheadas en el disco local, reduciendo drásticamente el tráfico de red.

  Un efecto colateral es que con el tiempo este caché llegará a ser una réplica casi idéntica de nuestro disco NFS (este caché es persistente frente a los arranque del sistema), con lo que prácticamente nuestro equipo funcionaría incluso si fallara el servidor NFS. Como vemos es algo parecido a la replicación de discos mediante imágenes pero realizado en caliente, de forma desatendida y actualizándose automáticamente cuando se producen cambios es el repositorio NFS.

El paquete prepare_hd realiza todas las tareas necesarias para conseguir lo anterior en el arranque del equipo. Se encuentra gobernado por la variable:

```
PREPARE_HD= ON/FORCE/OFF
```

Con el valor OFF no se usa el disco fijo. En ON sí lo usa pero comprobando primero si ya se encuentran creadas y formateadas las particiones. El valor FORCE rehace las particiones aun cuando se encontraran ya creadas correctamente.

Este paquete permite que arrancando con un diskette que contenga una ROM de *etherboot* una sola vez, el equipo se autoconfigure y quede preparado para arrancar posterior y

sucesivamente de forma autónoma. De esta forma es posible montar un aula completa en muy poco tiempo independientemente del estado en el que se encontraran los discos fijos con anterioridad.

4.3 /usr/local por NFS

Aparte de los nuevos paquetes para TS, necesitamos un sistema de ficheros que montaremos por NFS sobre el directorio */usr/local* de TS en el que se incluyan las utilidades y aplicaciones que conviertan a TS en un Linux *casi* completo.

4.3.1 Bases del diseño

Una de las virtudes de TS es que es ligero. Inicialmente esto viene motivado por la necesidad de ocupar poco espacio en la imagen de arranque, usar poca memoria y ejecutarse en cpus que ya estén casi obsoletas. Pero cuando se usa TS en equipos modernos todo esto se traduce en un efecto secundario: su extrema rapidez. TS es rápido porque prescinde de multitud de servicios y componente innecesarios. En ocasiones es mucho más rápido que un Linux completo instalado en el disco duro y, si nuestro servidor NFS es bastante rápido, nuestro Linux local también lo será (unido al caché NFS, mucho más).

Por tanto hemos querido mantener la filosofía de simpleza y ligereza sin sacrificar la funcionalidad, veamos algunos ejemplos:

- No usamos entornos de escritorio como *gnome* o *kde* que necesitan ingentes recursos, pero usamos un entorno de escritorio ligero como **ROX** que nos da la misma funcionalidad.
- No usamos clientes de correo como *thunderbird* o *ximian* que hacen muchas más cosas que mostrar el correo, usando un cliente ligero y rápido como **sylpheed**.
- No activamos más demonios que los necesarios para un puesto de trabajo, aunque disponemos incluso de un servidor Web para administración, que por supuesto no es *apache*.

Aun así disponemos de las últimas versiones de las librerías y componentes esenciales de un Linux moderno: *glib 2, gtk 2, Xorg, Qt,* etc. Si quisiéramos montar *gnome* completo podríamos hacerlo perfectamente, pero la productividad del equipo se reduciría drásticamente. Sí disponemos de aplicaciones complejas como **OpenOffice, Eclipse, DrScheme**, etc.

El directorio así montado por NFS sobre */usr/local* dispone de una estructura de directorios equivalente a la montada sobre el directorio raíz. Los directorios que en la imagen normal de TS no tienen contenido, han sido enlazados en el paquete xdm_local al directorio equivalente de */usr/local*. Las aplicaciones y librerías han sido compiladas para buscar también los ficheros necesarios en esta estructura en lugar de en el raíz de la instalación.

4.3.2 Procedimiento de desarrollo

Esta estructura */usr/local* exportada de un servidor NFS se monta a su vez sobre el directorio */usr/local* de la distribución de desarrollo de TS. Inicialmente partimos del contenido tal cual de dicha distribución fuente que fue replicado al *export* NFS. Una vez montado y conforme se van compilando nuevas paquetes, se van extendiendo paralelamente las funcionalidades del entorno de desarrollo y del de explotación.

Todo esto es necesario pues los script *"configure*" de los paquetes realizan comprobaciones sobre los paquetes previamente instalados (usando *pkg-config*), por lo que debemos tenerlos disponibles en ambos entornos.

4.3.3 Procedimiento de despliegue

Un problema a tener en cuenta es que nuestros equipos en las aulas usarán nuestro */usr/local* al mismo tiempo que nos encontramos ampliando el número de aplicaciones instaladas, corrigiendo problemas, etc. Por tanto debemos contar con un procedimiento para que el segundo proceso no interfiera en el primero.

Para ello hemos montado dos exports de NFS paralelos: TSDEV (**desarrollo**) y TSEXP (**explotación**). La variable NFS_LOCAL nos permitirá que los equipos de las aulas monten el export TSEXP y los equipos de pruebas el TSDEV. Cuando la estructura de desarrollo ha incorporado suficientes novedades y se encuentra suficientemente probada, se realiza un *rsync* de una a otra con lo que de forma inmediata los equipos de explotación tienen disponibles las novedades.

Aun así para contemplar la contingencia de que se haya deslizado algún problema no detectado y al objeto de poder dar *"marcha atrás"*, aprovechamos la característica **checkpoint** de nuestra NAS que es la que exporta los sistema de ficheros por NFS. Antes de cada replicación realizamos un *checkpoint* (*snapshot* o instantánea) tanto de TSEXP como de TSDEV, numerando dicho *checkpoint* con su número de versión. Si algo va mal, podemos volver en caliente a cualquier versión anterior deshaciendo los cambios.

El esquema de numeración de versiones que usamos es el siguiente:

- **Para TSDEV:** Realizamos *checkpoints* de las subversiones antes de cada cambio menor, afectando al segundo orden de numeración (Ej.: 1.01, 1.02, 1.03, etc).
- **Para TSEXP:** Al sincronizar TSDEV con TSEXP afectamos al primer orden de numeración (1.1, 1.2, 1.3, etc). Básicamente TSEXP solo se sincronizará cuando haya suficientes cambios como para considerar que estamos en una nueva versión.

Este procedimiento es más adecuado que mantener una estructura CVS. La instalación de un paquete produce multitud de cambios en el sistema de ficheros, saturando rápidamente un sistema CVS. Aparte el sistema de instantáneas nos permite la restauración inmediata y en caliente del estado del sistema de ficheros, la estructura CVS no.

5 EL FUTURO

Thinstation se ha convertido en un estándar "de facto" en el mundo de los clientes ligeros. Son muchos los terminales que se venden indicando su compatibilidad con TS. Es tanto su tirón que algunas empresas están comenzando a sacar al mercado productos comerciales complementarios o vendiendo servicios de consultoría sobre su implantación, que por cierto no se limita a ámbitos universitarios sino que ha pasado al mundo empresarial.

Nosotros por nuestra parte aportamos nuestras modificaciones a la comunidad de usuarios TS y las mantenemos en línea con las nuevas versiones de TS que se van liberando. Por tanto el futuro de nuestro sistema irá ligado al futuro de Thinstation y al de nuestras necesidades propias.

Existen proyectos dentro TS para desarrollar entornos de administración basados en PHP sobre el servidor DHCP y TFTP que permitan mantener una gran infraestructura de clientes ligeros a personal no especialmente conocedor de sus interioridades. En este proyecto nos hemos comprometido pues supone para nosotros una necesidad.

La inclusión del cacheFS, aún en fase experimental, es otra de las posibilidades futuras que nos darán muchas mejores prestaciones a nuestro sistema.

Por supuesto seguiremos instalando nuevas aplicaciones en nuestro sistema de Linux local en función de las demandas de los usuarios. También mejoraremos la configuración automática de sesiones de manera que el usuario se encuentre nada más conectarse con un entorno totalmente operativo y adaptado.

También sustituiremos lo poco que nos queda de arranque por RPL por arranques con TS. Todos los equipos nuevos de las aulas normalizadas de los diferentes centros que desean disponer de clientes ligeros están siendo instalados de esta manera.

Creemos que TS con nuestras modificaciones nos proporcionará un entorno de aulas genéricas lo suficientemente potente durante los próximos años. Desde que en 1996 el Servicio de Informática de la UCO se hizo cargo de las aulas en el nuevo Campus de Rabanales, este es el tercer sistema de aulas que usamos y, seguramente, no será el último.

ANEXO 1

EJEMPLO DE FICHERO DE CONFIGURACION DE NUESTRAS AULAS PARA SESIONES

```
KEYBOARD_MAP=es
ICA_USE_SERVER_KEYBOARD=ON
NET_TELNETD_ENABLED=ON
NET_REMOTE_ACCESS_FROM="XXX.XXX.XXX.XXX"
SYSLOG_SERVER=local
RECONNECT_PROMPT="On"

#DESCOMENTANDO LA SIGUIENTE LINEA USARIAMOS UN FONT SERVER
#SCREEN_X_FONT_SERVER=150.214.110.200:7100

#SESIONES PARA EL AULA S2
SCREEN=0
WORKSPACE=1
AUTOSTART=On

SESSION_0_TITLE="LINUXLOCAL"
SESSION_0_TYPE=uco
SESSION_0_SCREEN=0

SESSION_1_TITLE="LUCANO"
SESSION_1_TYPE=x
SESSION_1_X_SERVER=lucano.uco.es
SESSION_1_X_OPTIONS="-query"
SESSION_1_SCREEN=1

SESSION_2_TITLE="WINDOWS"
SESSION_2_TYPE=ica
SESSION_2_ICA_APPLICATION_SET="UCO-ESCRITORIO"
SESSION_2_ICA_SERVER="ucowtsdb.uco.es"
SESSION_2_SCREEN=2

#THINSTATION LOCAL: LUGARES DE MONTAJE
NFSLOCAL_MOUNT="nfs:/TSDEV/TSDEV"
FORCE_HOME_SMB=ON
NFS_HOME_SERVER="nfs"
NFS_HOME_ROOT="/HOMESNFS/HOMESNFS"
SMB_HOME_SERVER="hpsrv12"

#THINSTATION LOCAL: PREPARACION DEL DISCO DURO
PREPARE_HD="ON"
```

ANEXO 2

EJEMPLO DE FICHERO DE CONFIGURACION DE NUESTRAS AULAS PARA CAPACIDADES GRAFICAS

```
#GRUPO DE ORDENADORES CON MONITOR DE 17 PULGADAS

# --- XServer Options
#
# SCREEN_RESOLUTION       Screen resolutions available in the
workstations
# SCREEN_COLOR_DEPTH      Number of bits per pixel (8,16,24)
# SCREEN_HORIZSYNC        Monitor horizontal sync frequency in Khz.
# SCREEN_VERTREFRESH      Monitor vertical refresh frecuency in Hz.
# SCREEN_X_FONT_SERVER    IP address or hostname of the font server
for X
# MOUSE_PROTOCOL          Mouse protocol type (Microsoft, PS/2, etc.)
#                         (mouse is autodetected, use this only if it
fails)
# MOUSE_DEVICE            Mouse device:  /dev/ttyS0 -> COM1
#                                        /dev/ttyS1 -> COM2
#                                        /dev/ttyS2 -> COM3
#                                        /dev/ttyS3 -> COM4
#                                        /dev/psaux -> PS/2 mouse port
#                                        /dev/input/mice -> USB mouse
(needs USB package)
# MOUSE_RESOLUTION        Mouse resolution

SCREEN_RESOLUTION="1024x768 | *"
SCREEN_COLOR_DEPTH="16"
#SCREEN_HORIZSYNC="30-64 | *"
#SCREEN_VERTREFRESH="56-87 | 60 | 56 | 70 | 72 | 75"
SCREEN_HORIZSYNC="60"
SCREEN_VERTREFRESH="75"
MOUSE_DEVICE=/dev/psaux
MOUSE_PROTOCOL=IMPS/2
MOUSE_RESOLUTION=100
```

ANEXO 2

ESTRUCTURA GENERAL DE FICHERO DE CONFIGURACION DE TS

```
################################################
# --- Thinstation sample configuration file --- #
################################################
#
# This file must be named "thinstation.conf<xxx>" where <xxx> can be:
# .buildtime   Defines the defaults build into the image (note the
leading .)
# .network     Default config file loaded from the TFTP server.
# .user        Config file on local storage.
# -<name>      Specific config file on the TFTP server for the
terminal
#              "name" (e.g. thinstation.conf-paul). Requires
thinstation.hosts.
# .group-<id>  Config file for a group of terminals (e.g. with printer
setup
#              for those terminal with local printers. Requires
thinstation.hosts.
# -<IP>        Specific config file on the TFTP server for the
terminal with
#              the IP number <IP> specified.
# -<MAC>       Specific config file on the TFTP server for the
terminal with
#              the MAC address specified.

# --- General Options
#
# AUDIO_LEVEL        Audio Level for sound, 0-100
# KEYBOARD_MAP       Keyboard layout
# TIME_ZONE          Used to set time zone on TS client by entering
the UTC offset.
#                    This can be set automatically if the appropriate
dhcpc option is selected
#                    (Option 2, time offset in seconds)
# SYSLOG_SERVER      Log server ip address or hostname.
#                    If the work "local" is used, then syslog starts
logging locally
#                    If not specified syslogd is not loaded.
# USB_ENABLED        Enable USB Drivers into memory if USB package is
chosen
# DAILY_REBOOT       Will reboot server if up over a day and one of
the session
#                    types is closed
# AUTOPLAYCD         If enabled this will autoplay music cds when
inserted.
# CUSTOM_CONFIG      Allows choosing custom boot config, On/Off
# RECONNECT_PROMPT   This displays the reconnection/shutdown options
for when a session is ended
#
#                    OFF    No reconnect prompt
#                    ON     Reconnect prompt will be displayed
#                    MENU   Shows a menu with a shutdown and
reconnection option
```

```
#                     MENUXX As MENU option, but the XX is a time
period in minutes.
#                            After XX minutes shutdown will occur
unless the reconnect option
#                            is choosen

AUDIO_LEVEL=67
KEYBOARD_MAP=en_us
TIME_ZONE="UTC-12:30"
SYSLOG_SERVER=local
USB_ENABLED=On
DAILY_REBOOT=On
#AUTOPLAYCD=On
CUSTOM_CONFIG=Off
RECONNECT_PROMPT=On

# --- Default Settings for all sessions
#
# SCREEN           Display number to run the X server on
# AUTOSTART        On/Off. The application will be placed in a menu on
startup,
#                  but not automatically executed
# WORKSPACE        Workspace in the window manager to run the
application in

SCREEN=0
WORKSPACE=1
AUTOSTART=On

# --- Citrix ICA Specific Options
#
#                        GLOBAL Settings
#ICA_USE_SERVER_KEYBOARD    Use default server keyboard, otherwise use
KEYBOARD_MAP
#                           variable
#ICA_BROWSER_PROTOCOL       Broswer protocol, can be HTTPonTCP or UDP
#
#                           APPLICATION SET settings
#ICA_ENCRYPTION             Encryption level for ICA
#                           Valid Settings Below
#                           "Basic"
#                           "RC5 (128 bit - Login Only)"
#                           "RC5 (40 bit)"
#                           "RC5 (128 bit)"
#                           "RC5 (56 bit)"
#ICA_COMPRESS               Compression, On/Off
#ICA_AUDIO                  Audio, On/Off
#ICA_AUDIO_QUALITY          Audio Quality, Low, Medium, High
#ICA_PRINTER                This will turn on ICA autocreate printers,
see printer section
#                           for details, On/Off
#
#                           NOTE: You must have the lpr package
included for this option
#                           to work.
#
#ICA_APPLICATION_SET        Published Application (Not needed if using
#                           ICA_SERVER)
#ICA_SERVER                 Server to Connect to (Not needed if using
#                           ICA_APPLICATION_SET, but needed if the
ICA-Masterbrowser
```

```
#                            is not on the local network.)

#ICA_USE_SERVER_KEYBOARD=Off
#ICA_BROWSER_PROTOCOL=HTTPonTCP
#ICA_SERVER=
#ICA_ENCRYPTION=Basic
#ICA_COMPRESS=On
#ICA_AUDIO=On
#ICA_AUDIO_QUALITY=Low
#ICA_PRINTER=Off
#ICA_SEAMLESS_WINDOW=Off


# --- Session Options
#
# Note:                     # is a number equal to or greater than 0
#
# SESSION_#_TITLE           Title description for SESSION. Needed for
replimenu.
# SESSION_#_TYPE            Package type, choose beetwen:
#                           - vncviewer         Start vncviewer in X
#                           - rdesktop          Start rdesktop in X
#                           - rdesktop_svga     Start svga rdesktop,
based on rdesktop 1.1 code
#                                               for low memory machines
#                           - x                 Start x-terminal session
(xdm)
#                           - xnest             Start x-terminal session
(xdm) from within blackbox
#                           - ssh               Start ssh client in
linux console
#                           - telnet            Start telnet client in
linux console
#                           - ica               Start Citrix ICA client
in X
#                           - ica_wfc           Start ICA Manager
#                           - blackbox          Start blackbox window
manager session
#                           - icewm             Start icewm window
manager session
#                           - dillo             Start Web Browser in X
#                           - tftpd             Start tftp daemon
#                           - tarantella        Start tarantella client
#                           - rxvt              Start light xterm client
#                           - xterm             Start xterm client
#                           - tn5250            Start AS400 client in
linux console
#                           - nx                Start NX Client Session
# SESSION_#_SCREEN          Display number to run the XF server on
# SESSION_#_AUTOSTART       On/Off Application will be placed in a menu
on startup, but
#                           not automatically executed
# SESSION_#_WORKSPACE       Workspace to run program on in a window
manager
# SESSION_#_type_SERVER     IP address/hostname of the server
# SESSION_#_type_OPTIONS    Command line options for the session type
#
#

# Individual Session Settings, override defaults
#
```

```
# IMPORTANT:  Make sure you minimum have a SESSION_0. Otherwise you
will get an error
# on boot. You may have additional sessions: SESSION_1, SESSION_2 ...
# SESSION_0 is on ctrl-alt-F3
# SESSION_1 is on ctrl-alt-F4
# ... etc.

SESSION_0_TYPE=blackbox

#SESSION_1_TYPE=ica
#SESSION_1_ICA_APPLICATION_SET="Microsoft Word"
#SESSION_1_ICA_OPTIONS="-username donald -clearpassword qwak -domain
disney"

#SESSION_2_TYPE=ica
#SESSION_2_ICA_SERVER=ICA

#SESSION_3_TYPE=ica_wfc

# The -a option here specifies the color depth
# Note certain servers support certain color depths, wrong settings
with this
# May cause your connection to fail.
#SESSION_4_TITLE="Big Bad Server Donald"
#SESSION_4_TYPE=rdesktop
#SESSION_4_SCREEN=1
#SESSION_4_RDESKTOP_SERVER=192.168.1.1
#SESSION_4_RDESKTOP_OPTIONS="-u user -p password -a 16"
#SESSION_4_AUTOSTART=Off

#SESSION_#_TITLE="Big Bad Server Road Runner"
#SESSION_#_TYPE=rdesktop
#SESSION_#_SCREEN=1
#SESSION_#_RDESKTOP_SERVER=192.168.1.1
#SESSION_#_RDESKTOP_OPTIONS="-u 'fred' -a 8"
#SESSION_#_AUTOSTART=Off

#SESSION_#_TYPE=rxvt
#SESSION_#_SCREEN=1
#SESSION_#_RXVT_OPTIONS="-bg black -cr green -fg white -C -sl 500"
#SESSION_#_WORKSPACE=2

#SESSION_#_TYPE=blackbox
#SESSION_#_SCREEN=1

#SESSION_#_TYPE=x
#SESSION_#_SCREEN=2
#SESSION_#_X_SERVER=192.168.1.2
#SESSION_#_X_OPTIONS="-indirect"
#SESSION_#_AUTOSTART=Off

#SESSION_#_TYPE=xnest
#SESSION_#_X_SERVER=192.168.1.3
#SESSION_#_X_OPTIONS="-query"
#SESSION_#_AUTOSTART=On

#SESSION_#_TYPE=telnet
#SESSION_#_TELNET_SERVER=192.168.1.2

#SESSION_#_TYPE=ssh
#SESSION_#_SSH_SERVER=192.168.1.2
```

```
#SESSION_#_TYPE=vncviewer
#SESSION_#_VNCVIEWER_SERVER=192.168.1.2

#SESSION_#_TYPE=tn5250
#SESSION_#_TN5250_SERVER=192.168.1.2

#SESSION_#_TYPE=nx

#SESSION_#_TYPE=icewm


# --- PKG Options
#
# PKG_PACKAGES       Choice of packages to download for PKG
#                    You can also use PKG_PACKAGES1-8 for additional
package selections
#                    This is useful for using multiple network group
files
# PKG_PREFIX         Download PKGs from a subdir of /tftpboot or PKG
Path if PKG_PATH
#                    is set in thinstation.conf
# PKG_PATH           Path to PKG files if not using tftpboot
#                    Floppy: /mnt/floppy
#                    CD-ROM: /mnt/cdrom
#                    HD:     /mnt/disc/disc0/part1   (first disc,
first partition)
# MOD_PACKAGES       Choice of modules to download for MPKG
#                    You can also use MOD_PACKAGES1-8 for additional
package selections
#                    This is useful for using multiple network group
files
# MOD_PREFIX         Downloads and insmod's a module from a subdir of
/tftpboot
#                    if MOD_PREFIX is set in thinstation.conf

#PKG_PACKAGES="blackbox rxvt"
#PKG_PREFIX=pkg
#PKG_PATH=/mnt/cdrom
#MOD_PACKAGES="usb-hid usb-storage"
#MOD_PREFIX=modules


# --- X Server Options
#
# SCREEN_RESOLUTION       Screen resolutions available in the
workstations
# SCREEN_COLOR_DEPTH      Number of bits per pixel (8,16,24)
# SCREEN_HORIZSYNC        Monitor horizontal sync frequency in Khz.
#                         If left blank Xorg will try to detect with
DDC
# SCREEN_VERTREFRESH      Monitor vertical refresh frecuency in Hz.
#                         If left blank Xorg will try to detect with
DDC
# SCREEN_X_FONT_SERVER    IP address or hostname of the font server
for X
# MOUSE_PROTOCOL          Mouse protocol type (Microsoft, PS/2, etc.)
#                         (mouse is autodetected, use this only if it
fails)
# MOUSE_DEVICE            Mouse device:  /dev/ttyS0 -> COM1
#                                        /dev/ttyS1 -> COM2
```

```
#                                   /dev/ttyS2 -> COM3
#                                   /dev/ttyS3 -> COM4
#                                   /dev/psaux -> PS/2 mouse port
#                                   /dev/input/mice -> USB mouse
(needs USB package)
# MOUSE_RESOLUTION        Mouse resolution
# --- Advanced XServer Options - Experts Only
# X_DRIVER_NAME           Driver for X, this will override the
autodetection scripts
# X_DRIVER_BUSID          Screen Card BusID
# X_DRIVER_OPTION1-4      Addditional options for driver

SCREEN_RESOLUTION="800x600 | 1024x768 | 640x480 | *"
SCREEN_COLOR_DEPTH="16 | 8 | 24 | *"
#SCREEN_HORIZSYNC="30-64 | *"
#SCREEN_VERTREFRESH="56-87 | 60 | 56 | 70 | 72 | 75"
#SCREEN_X_FONT_SERVER=192.168.1.2:7100
#MOUSE_PROTOCOL=PS/2
#MOUSE_DEVICE=/dev/psaux
MOUSE_RESOLUTION=100
# --- Advanced Options - Experts Only
#X_DRIVER_NAME="mga"
#X_DRIVER_BUSID="PCI:1:0:0"
X_DRIVER_OPTION1="swcursor On"
#X_DRIVER_OPTION2="ActiveDevice CRT"
#X_DRIVER_OPTION3="noDDC Off"
#X_DRIVER_OPTION4="UseBios Off"
#X_DRIVER_OPTION5="ShadowFB Off"


# --- Printing Options
#
# PRINTER_0_NAME     Workstation Printer Name, Can be Any Valid Name
#                    If you have turned ICA_PRINTER=ON then this is
the
#                    name of the printer driver
# PRINTER_0_DEVICE   Workstation printer device (if not specified
devices
#                    are not loaded).
#                       /dev/printers/[0-2]   for parallel ports
#                       /dev/ttyS[0-3] for serial ports
#                       /dev/usb/lp[0-2] for USB printers
#
# PRINTER_0_TYPE     P for parallel, S for serial, U for USB printer
# PRINTER_0_OPTIONS  Serial port options.
# PRINTER_1_*        See PRINTER_0_*
# PRINTER_2_*        See PRINTER_0_*
# PRINTER_3_*        See PRINTER_0_*

#PRINTER_0_NAME="parallel"
#PRINTER_0_DEVICE=/dev/printers/0
#PRINTER_0_TYPE=P

#PRINTER_1_NAME="serial"
#PRINTER_1_DEVICE=/dev/ttyS1
#PRINTER_1_TYPE=S
#PRINTER_1_OPTIONS="speed 38400 -imaxbel"

#PRINTER_2_NAME="usb"
#PRINTER_2_DEVICE=/dev/usb/lp0
#PRINTER_2_TYPE=U
```

```
#PRINTER_3_NAME="usb"
#PRINTER_3_DEVICE=/dev/usb/lp0
#PRINTER_3_TYPE=U


#ICA Autocreate Printer Example

#PRINTER_0_NAME="HP LaserJet Series II"
#PRINTER_0_DEVICE=/dev/printers/0
#PRINTER_0_TYPE=P

# --- Permanent Storage Options
#
# STORAGE_PATH           Path to where storage device is mounted to
save
#                        profile settings.  This should be one of
#                        /mnt/usbdevice/busX.targetX.lunX/partX or
disc
#                        /mnt/floppy
#                        /mnt/disc/discX/partX
#                        /mnt/nfs
#                        /mnt/smb
#
#                        Note that the profile settings are stored
under
#                        a subfolder for this path.  So the path to
the
#                        stored settings would be something like
#                        /mnt/floppy/thinstation.profile
#                        See the FAQ on the website for more details
on this
#
#                        Also checkout README.IMPORTANT for the valid
#                        config files which you can place here
#                        a typical file to store config file settings
is
#
/mnt/floppy/thinstation.profile/thinstation.user
#
#                        Also note that the .profile can be changed by
using
#                        the below STORAGE_PREFIX
# STORAGE_SERVER         This is the path to the SMB or NFS server, ie
#                        server:/path/to/profile
# STORAGE_USER           This is the username for SMB mounts, password
is
#                        defined at build time in build.conf
# STORAGE_PREFIX         This is prefex for the folder name to store
settings
#                        in on the storage device.   You can also use
one of the
#                        special characters below.
#
#                        M = Mac Address
#                        H = Hostname
#                        I = Ip Address
#
#                        Note default prefix for storing the profile
is
#                        .profile
```

```
# STORAGE_CONFIG1-4      This is the path for any user defined
settings
#                        which will always override the profile path
above
#                        The files are tried on order on each device
specified


#STORAGE_PATH=/mnt/nfs
#STORAGE_SERVER=bigserver:/opt/thinstation
#STORAGE_USER=duck
#STORAGE_PREFIX=H
#STORAGE_CONFIG1=/mnt/floppy
#STORAGE_CONFIG2=/mnt/cdrom
#STORAGE_CONFIG3=/mnt/disc/disc1/part1
#STORAGE_CONFIG4=/mnt/usbdevice/busX.targetX.lunX/partX or disc


# --- Samba Options
#
# SAMBA_SERVER_ENABLED    Enable Samba Server Daemons, On/Off
# SAMBA_SECURITY          Type of security, USER, DOMAIN, SERVER
# SAMBA_SERVER            Server for security
# SAMBA_WINS              Server Name to Enable Samba to be a WINS
Client
# SAMBA_HARDDISK          Enable Harddrive support, On/Off
# SAMBA_CDROM             Enable CDRom Support, On/Off
# SAMBA_FLOPPY            Enable Floppy Support, On/Off
# SAMBA_PRINTER           Enable Printer Support, On/Off
# SAMBA_USB               Enable USB Support, On/Off

#SAMBA_SERVER_ENABLED=Off
#SAMBA_WORKGROUP=BigPeople
#SAMBA_SECURITY=Server
#SAMBA_SERVER=BadMan
#SAMBA_WINS=BadMan
#SAMBA_HARDDISK=Off
#SAMBA_CDROM=On
#SAMBA_FLOPPY=On
#SAMBA_PRINTER=On
#SAMBA_USB=Off


# --- Networking Options
#
# NET_HOSTNAME           Hostname to use if not using a
thinstation.hosts file, note
#                        that the machine MAC address will replace any
* if used.
# NET_REMOTE_ACCESS_FROM List of hostnames/ip address accepted by the
server
#                        for remote control, used by telnetd and www
packages
# NET_SMTP_SERVER        Email server address for error logs
# NET_SMTP_EMAIL         Email address of administrator
# NET_TELNETD            Enables built-in telnetd server package
#

#NET_HOSTNAME=donald
NET_HOSTNAME=ts_*
NET_SMTP_SERVER=donald
```

```
NET_SMTP_EMAIL=donald@duck.org.nz
NET_TIME_SERVER=mickey
NET_TELNETD_ENABLED=On
NET_REMOTE_ACCESS_FROM="duck.quak.org.au 192.168.0.0 .disney.us"


# --- For use in thinstation.conf.buildtime only:
#
# NET_USE_DHCP           Enable DHCP Resolution, On/Off
# NET_IP_ADDRESS         IP Address of client
# NET_MASK               Network mask of client
# NET_GATEWAY            IP Address of gateway
# NET_USE_TFTP           Use tftp server for config file, On/Off
# NET_DNS1               DNS Server 1
# NET_DNS2               DNS Server 2
# NET_DNS_SEARCH         Default DNS domain to search
# NET_ALTERNATE_TFTP     Alternate tftp server for vmware or non-
standard dhcp servers

#NET_USE_DHCP=Off
#NET_IP_ADDRESS=192.168.0.1
#NET_MASK=255.255.255.0
#NET_GATEWAY=192.168.0.254
#NET_USE_TFTP=Off
#NET_DNS1=192.168.0.2
#NET_DNS2=192.168.0.3
#NET_DNS_SEARCH=cartoons.org.nz
#NET_ALTERNATE_TFTP=192.168.0.4


# ************************************************
# ***** EXPERIMENTAL  ******  USE WITH CAUTION *****
# ************************************************

# --- HD Update options
#
# HDUPDATE_ENABLED=Y        Enable Update
# HDUPDATE_WS_VERSION       Version of files on workstation (set in
thinstation.buildtime only)
# HDUPDATE_SERVER_VERSION   Version of files on server (set in both
buildtime and network)
#                           This should be set in buildtime to match
the WS_VERSION to prevent
#                           unintended upgrades
#
# HDUPDATE_SERVER           tftp server name
# HDUPDATE_PATH             Path to files /ts/update
# HDUPDATE_FORCE            Force install even if not needed
# HDUPDATE_FILES            List of files to upgrade
# HDUPDATE_DELETE           List of files to delete (no sanity checks,
use with caution)

#HDUPDATE_WS_VERSION=1
#HDUPDATE_ENABLED=Y
#HDUPDATE_SERVER_VERSION=CO.29
#HDUPDATE_SERVER=roadrunner
#HDUPDATE_PATH=/ts/update
#HDUPDATE_FORCE=N

# Set the list of files to download
```

```
#HDUPDATE_FILES="vmlinuz hires lowres syslinux.cfg thin.txt
firefox.pkg flash.pkg"
#HDUPDATE_FILES="hires lowres"
#HDUPDATE_FILES="lowres"
#HDUPDATE_FILES="hires"
#HDUPDATE_FILES="syslinux.cfg thin.txt"
```