IMPLEMENTACIÓN DE UNA INTRANET EN SISTEMA OPERATIVO LINUX PARA LA INTEGRACIÓN DE SERVICIOS EN LA CONTRALORÍA MUNICIPAL DE BARRANCABERMEJA

YOLANDA SANCHEZ ARCE
CARLOS ALBERTO MENDOZA SAAD

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTA DE INGENIERIA ELÉCTRICA Y ELECTRÓNICA
DEPARTAMENTO DE INGENIERIA
ESPECIALIZACION EN TELECOMUNICACIONES – COHORTE VI
BUCARAMANGA, 2007

IMPLEMENTACIÓN DE UNA INTRANET EN SISTEMA OPERATIVO LINUX PARA LA INTEGRACIÓN DE SERVICIOS EN LA CONTRALORÍA MUNICIPAL DE BARRANCABERMEJA

YOLANDA SANCHEZ ARCE
CARLOS ALBERTO MENDOZA SAAD

Trabajo presentado como requisito para optar por el Título de Especialista en Telecomunicaciones

Director: Fredy Alfonso Beltrán Miranda
Ingeniero de Sistemas
Docente: Faculta de Ingeniería
Universidad Industrial de Santander

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTA DE INGENIERIA ELÉCTRICA Y ELECTRÓNICA
DEPARTAMENTO DE INGENIERIA
ESPECIALIZACION EN TELECOMUNICACIONES – COHORTE VI
BUCARAMANGA, 2007

A Dios

A mi Madre, muy especialmente,

A mis sobrinos, y a mi gran familia,

A mi compañero de tesis, con quien hice un buen grupo de trabajo en equipo.

A la ESE IMSALUD empresa en la que anhelo aplicar mis conocimientos,

A todos los que de una u otra manera me apoyaron para cumplir mi objetivo les agradezco de corazón su apoyo y comprensión.

YOLANDA

A mis padres y mis hermanos que siempre me han apoyado en los proyectos de mi vida.
A la familia Contraloría Municipal de Barrancabermeja, por darme la oportunidad de demostrar mis conocimientos.

CARLOS

TÍTULO: IMPLEMENTACION DE UNA INTRANET EN SISTEMA OPERATIVO LINUX PARA LA INTEGRACION DE SERVICIOS EN LA CONTRALORIA MUNICIPAL DE BARRANCABERMEJA[*]

AUTORES: CARLOS ALBERTO MENDOZA SAAD
YOLANDA SANCHEZ ARCE[**]

PALABRAS CLAVE:
SAMBA: Servicio de Linux que permite compartir archivos, carpetas e impresoras en un servidor, además del manejo de cuotas de disco y seguridad.
APACHE: Servicio de Linux que permite el manejo, publicación y administración de páginas Web.
SQUID: Servicio de Linux que permite el manejo y la administración de un servidor Proxy para la seguridad, navegación, permisos del Servicio de Internet.
LINUX FEDORA CORE 4: Versión de Linux que se utilizó en este trabajo.
INTRANET: Término utilizado para la implantación de tecnologías de Internet en una red corporativa.

Dada la necesidad de que el ente de control fiscal la CONTRALORIA MUNICIPAL DE BARRANCABERMEJA (CMB) de mejorar sus procedimientos, realizar su objetivo control fiscal de manera eficiente, eficaz y darle resultados rápidos y positivos a la comunidad, se planteó este proyecto asignándole la configuración de un servidor que manejo los servicios para este fin. El sistema operativo que se escogió fue Linux ya que este programa posee grandes ventajas como son código abierto, mayor seguridad, robustez porque no solo permite la caída del mismo sino la facilidad para subir y bajar servicios de red de acuerdo a como se necesiten.

Para suplir esta necesidad, inicialmente se realizó un diseño de la red, asignándoles a cada uno de los usuarios su dirección lógica y su identificación en la red, una vez instalado esto se procedió a realizar el montaje y la configuración de cada uno de los servicios de red como lo son: el servidor web APACHE, configuración de manejo de correo electrónico con SENDMAIL, protección de virus y spam con MAILSCANNER y SPAMASSASSIN respectivamente, servidor Proxy para el control y la navegación en Internet con SQUID, y para la compartición de archivos y carpetas con SAMBA. Además de esto se realizó un estudio de Seguridad Informática donde se realizan unas recomendaciones para el buen manejo y la confidencialidad de la información.

Esta implementación fue de gran utilidad porque proporciona herramientas de Internet, a la vez permite la integración lógica de las aplicaciones, auxiliando la producción de cada uno de los procesos; es también un importante medio de difusión de información interna a nivel de grupos de trabajo, y proporciona mecanismos de restricción de acceso a nivel de programación. Además esta Intranet fue creada para mayor seguridad y fundamentalmente para compartir archivos, carpetas y recursos.

---

[*] Proyecto de Grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: Freddy Alfonso Beltrán.

TITLE: IMPROVEMENT OF LINUX OPERATING SYSTEM IN INTRANET FOR THE SERVICES INTEGRATION IN THE MUNICIPAL COMPTROLLER OFFICE OF BARRACABERMEJA[*]

AUTHORS: CARLOS ALBERTO MENDOZA SAAD
YOLANDA SANCHEZ ARCE[**]

KEY WORDS:
SAMBA: Linux service that allows sharing files, folders and printings using one server, in addition to the disk quote and safety.

APACHE: Linux service that allows web pages management, publishing and administration.

SQUID: Linux service that allows Proxy severs management and administration for the safety, sailing, and intranet service license.

LINUX FEDORA CORE 4: Linux version used in this work.

INTRANET: Term used for performing the internet technologies into a cooperative net.

Because of the entity control fiscal, the Municipal comptroller office of Barracabermeja, necessities of improving their procedures, achieving their aim of fiscal control in a efficient, effective way, and also giving quick and positive results to the community, it was set up this project assigning the server configuration which operating the services for these aim. The operative system chosen was Linux, because of the advantages it has, as open code, high safety robustness, in order that it does not allow its downfall except it facilitates to the net services going up and down according to its necessities.

To supply these necessities, at the beginning t was achieved a net design, giving to each one of the users his/her own logic address and identification in the net, once it was installed, the second step was to carry out the assembling and configuration of each net services, like: the APACHE web server, management configuration of the e-mail using SENDMAIL, virus and spam protection using MAILSCANNER and SPAMASSASSIN, respectively, proxy server to Internet control and sailing using SQUID, and SAMBA for the files and folders sharing. Besides this it was achieved a Data Processing Safety, in which was given some advises for the high management and security of the information.

This execution was useful because it gives Internet tools, at the same time it allows the logic integration of the applications, helps to the each process productions; also, it is an important way of broadcasting of the boarder information in level of the work groups; and it provides some mechanism of accessing restriction in level of programming. Moreover this Intranet was created for the high safety and mainly to shared files, folders and resources.

---

[*] Degree Work

[**] Faculty of Physic-Mechanic Engineering. School of Electrical, Electronic and Telecommunication Engineering. Director: Freddy Alfonso Beltrán.

TABLA DE CONTENIDO

TABLA DE GRAFICOS

TABLA DE TABLAS

INTRODUCCION

Para los entes estatales es importante estar a la vanguardia en la tecnología de información y realizar procesos de modernización en ese campo Es por eso que la CONTRALORÍA MUNICIPAL DE BARRANCABERMEJA, ha iniciado un proceso para mejorar sus procedimientos, realizar su objetivo de control Fiscal de manera eficiente y eficaz y darle resultados rápidos y positivos a la comunidad.

El presente trabajo tiene como objetivo Implementar una intranet en Sistema Operativo Linux Fedora para la integración de servicios como el control de span, configurar el servidor Proxy para control de navegación, configurar el servidor samba para comunicar máquinas linus – windows, copias de seguridad y crear una solución eficiente, de bajo costo y mayor productividad.

Se pretende Integrar las aplicaciones o software que se encuentran diversificadas en los terminales de los funcionarios de la entidad; con el objetivo de ofrecer óptima utilización de los recursos informáticos con los que cuentan. La Contraloría Municipal adquirió unos equipos de cómputo y plataformas de red de última tecnología pero aun no se han implementado. Es por esto que se ve en la necesidad de implementarse una Intranet para continuar con sus procesos y procedimientos en pro de mejorar sus sistemas de información, seguridad informática de manera interna y externa, y optimizar el servicio de Internet que posee.

Teniendo en cuenta que los sistemas UNIX, son utilizados actualmente en la mayoría de organizaciones, ya sean publicas o privadas que necesitan optimizar sus comunicaciones, gracias a que se caracterizan por combinar de forma armónica la mayoría de detalles anteriormente mencionados, nosotros conocedores de los requerimientos del sistema en la entidad desarrollamos como trabajo de grado la: del IMPLEMENTACIÓN DE UNA INTRANET EN SISTEMA OPERATIVO LINUX PARA LA INTEGRACIÓN DE SERVICIOS EN LA CONTRALORÍA MUNICIPAL DE BARRANCABERMEJA.

Se evaluará la situación actual en relación a los problemas de seguridad informática que presenta la CONTRALORÍA MUNICIPAL DE BARRANCABERMEJA, se implementara la intranet en sistema operativo LINUX, se realizará un análisis de los resultados obtenidos, se sacaran las respectivas conclusiones y se dará unas pautas a seguir al respecto.

1. FUNDAMENTOS DE INTRANET

En términos simples, Intranet es el término descriptivo que está siendo usado para la implantación de tecnologías de Internet en una organización corporativa, en lugar de la conexión externa a Internet global. Esta implementación funciona como una alternativa al intercambio transparente de los inmensos recursos de información de una organización entre cada uno de los equipos individuales con un mínimo costo, tiempo y esfuerzo. Este documento intenta explicar en términos simples cómo configurar una intranet usando herramientas y servicios de Linux que son fácilmente obtenibles y que generalmente son de bajo coste o libres.

1.1. FUNDAMENTOS DE INTRANET EN LINUX FEDORA

1.1.1. Historia: Red Hat Software Inc. fue fundada en 1994 por Bob Young y Marc Ewing. En agosto de 1999, Red Hat salió a bolsa y sus acciones obtuvieron la octava ganancia de primer día más grande en toda la historia de Wall Street. Cuatro años más tarde, el valor de las acciones de Red Hat es en torno a una centésima parte del máximo valor que llegara a alcanzar antes de la crisis de las puntocom. Aun así, sus comienzos exitosos en el mercado de valores sirvieron para que Red Hat fuera portada en periódicos y revistas no directamente relacionadas con temas informáticos. En cualquier caso, parece ser que Red Hat ha sabido superar los problemas de otras compañías del mundo de los negocios en torno al software libre y anunció números negros por primera vez en su historia en el último cuarto del año 2002.

Otro de los hechos históricos más importantes de Red Hat fue la adquisición en noviembre de 1999 de Cygnus Solutions, una empresa fundada una década antes y que ya había demostrado cómo con una estrategia integral basada en software libre se puede ganar dinero.

En septiembre de 2003, Red Hat ha decidido concentrar sus esfuerzos de desarrollo en la versión corporativa de su distribución y delegó la versión común a Fedora Core, un proyecto abierto independiente de Red Hat.

Fedora Core (también conocida como Fedora Linux) es una distribución GNU/Linux desarrollada por la comunidad Fedora y promovida por la compañía estadounidense Red Hat.

Gráfica 1. Pantalla de Escritorio de Fedora.

El objetivo del proyecto Fedora es conseguir un sistema operativo de propósito general y basado exclusivamente en software libre con el apoyo de la comunidad Linux. Los ingenieros de Red Hat continúan participando en la construcción y desarrollo de este proyecto e invitan y fomentan la participación de miembros de la comunidad Linux.

Originalmente, Red Hat Linux fue desarrollado exclusivamente dentro de Red Hat, con la sola realimentación de informes de usuarios que recuperaban fallos y contribuciones a los paquetes de software incluidos; y no contribuciones a la distribución como tal. Esto cambió el 22 de septiembre de 2003, cuando Red Hat Linux se derivó dando origen al Proyecto Fedora que

está orientado a la comunidad de usuarios y así mismo, sirve de base para que Red Hat Enterprise Linux se desarrolle con más efectividad y adopte las nuevas características que se añaden en el Proyecto Fedora.

1.1.2. Características especiales. Red Hat Linux es instalado con un ambiente gráfico llamado Anaconda, diseñado para su fácil uso por novatos. También incorpora una herramienta llamada Lokkit para configurar las capacidades de Cortafuegos.

Al igual que en el Red Hat Linux 8.0, UTF-8 fue habilitado como el sistema de codificación de tipografías para el sistema. Esto tiene poco efecto en usuarios angloparlantes, pero cuando se usa la parte superior del juego de caracteres ISO 8859-1, éstos se codifican de manera radicalmente diferente. Esto puede ser visto, por ejemplo, por usuarios que hablan francés o sueco como algo agresivo, pues sus antiguos sistemas de archivo lucen muy diferentes y pueden ser luego inutilizables. Puede deshacerse este cambio quitando la parte ".UTF-8" de la configuración de lenguaje.

La versión 8.0 fue además la primera en incluir el entorno de escritorio gráfico Bluecurve.

1.1.3. Fedora Core. Es un Sistema Operativo y de plataforma, basado en Linux, es y siempre será libre para que cualquier persona lo use, modifique y distribuya, ahora y por siempre. Fedora es desarrollado por un gran número de personas de la comunidad que se esfuerza en proveer y mantener lo mejor del software libre, el software de código abierto y de los estándares. Fedora Core es parte del Proyecto Fedora y es patrocinado por Red Hat, Inc.

Originalmente el Red Hat Linux fue desarrollado exclusivamente dentro de Red Hat, con la sola realimentación de informes de usuarios que recuperaban de fallos y contribuciones a los paquetes de software incluidos; y no contribuciones a la distribución como tal. Esto cambió tardíamente en el 2003 cuando Red Hat Linux se fusionó con el Proyecto Fedora Linux orientado a la comunidad de usuarios. El nuevo plan es extraer el código base de Fedora para crear nuevas distribuciones de Red Hat Enterprise Linux.

Fedora Core sustituye a las versiones originales de Red Hat Linux para descarga y venta al detalle. Este modelo es similar a la relación entre Netscape Communicator y Mozilla, o entre StarOffice y OpenOffice.org, aunque en este caso el producto comercial resultante es software libre.

Historial de Versiones

- 5 de noviembre de 2003: Core 1 (Yarrow).
- 19 de mayo de 2004: Core 2 (Tettnang)
- 8 de noviembre de 2004: Core 3 (Heidelberg)
- 13 de junio de 2005: Core 4 (Stentz)

2. INTRODUCCIÓN Y ADMINISTRACIÓN DE REDES BAJO LINUX FEDORA

2.1. ADMINISTRACION LOCAL DEL SISTEMA

El Administrador se debe enfrentar a la gestión de los recursos locales presentes en la máquina.

Inicialmente se analiza el proceso de arranque de un sistema GNU/Linux, para comprender la estructura inicial del sistema y su relación con los diversos servicios que éste proporciona.

Para obtener una visión general del estado actual del sistema por medio de los diferentes procedimientos y comandos de que se dispone para evaluar las diversas partes del sistema; de este modo podremos tomar decisiones de administración si se detecta algún fallo o deficiencia de rendimiento, o la falta de algún recurso.

La gestión de usuarios es uno de los principales puntos de la administración es, ya que cualquier configuración de la máquina estará destinada a que pueda ser utilizada por éstos; se deben definir nuevos usuarios al sistema y controlar su nivel de acceso a los recursos.

En cuanto a los periféricos del sistema, como discos e impresoras, Se debe disponer de diferentes posibilidades de gestión, ya sea vía diferentes servidores (caso impresión) o diferentes sistemas de archivos, así como algunas técnicas de optimización del rendimiento de los discos.

También se debe examinar el problema de la actualización del sistema y cómo mantenerlo actualizado; así como la nueva incorporación de software de aplicación y cómo hacerlo disponible a los usuarios.

Asimismo analizar la problemática de ejecutar trabajos temporizados en el sistema.

2.1.1. Distribuciones. Intentamos destacar ahora algunas diferencias menores en las distribuciones (Red Hat y Debian) utilizadas [Mar03].

Cambios o particularidades de Red Hat:

- Uso del gestor de arranque *grub* (un software GNU), a diferencia de la mayoría de distribuciones que suelen usar *lilo*, Red Hat utiliza *grub*. GRUB (*Grand Unified Bootloader*) tiene una configuración en modo texto (normalmente en /boot/grub/grub.conf) bastante sencilla, y que puede modificarse en el arranque.

- Gestión de alternativas. En el caso de que haya más de un software presente, mediante un directorio (/etc/alternatives), se indica cuál es la alternativa que se usa. Este sistema se tomó prestado de Debian, que hace un uso amplio de él en su distribución.

- Programa de escucha de puertos TCP/IP basado en xinetd; en /etc/xinetd.d podemos encontrar de forma modular los ficheros de configuración para algunos de los servicios TCP/IP, junto con el fichero de configuración /etc/xinetd.conf. En los sistemas UNIX clásicos, el programa utilizado es el *inetd*, que poseía un único fichero de configuración en /etc/inetd.conf, caso, por ejemplo, de la distribución Debian que utiliza *inetd*.

- Algunos directorios de configuración especiales: /etc/profile.d, archivos que se ejecutan cuando un usuario abre un *shell*; /etc/xinetd.d, configuración de algunos servicios de red; /etc/sysconfig, datos de configuración de varios aspectos del sistema; /etc/cron., varios directorios donde se especifican trabajos para hacer periódicamente (mediante *crontab*); /etc/pam.d, donde PAM son los denominados *módulos de autentificación*: en cada uno de cuyos archivos se configuran permisos para el programa o servicio articular; /etc/logrotate.d, configuración de rotación (cuando hay que limpiar, comprimir, etc.) de algunos de los ficheros de *log* para diferentes servicios.

- Dispone de un software llamado *kudzu*, que examina el hardware en arranque para detectar posibles cambios de configuración y generar los dispositivos o configuraciones adecuadas.

En el caso de Debian:

- Sistema de empaquetado propio basado en los paquetes DEB, con herramientas de varios niveles para trabajar con los paquetes como: *dpkg*, *apt-get*, *dselect*, *tasksel*.

- Debian sigue el FHS, sobre la estructura de directorios, añadiendo algunos particulares en /etc, como por ejemplo: /etc/default, archivos de configuración, y valores por defecto para algunos programas; /etc/network, datos y guiones de configuración de las interfaces de red;

/etc/dpkg y /etc/apt, información de la configuración de las herramientas de gestión de paquetes; /etc/alternatives, enlaces a los programas por defecto, en aquellos que hay (o puede haber) varias alternativas disponibles.

- Sistema de configuración de muchos paquetes de software por medio de la herramienta dpkg-reconfigure. Por ejemplo:

  dpkg-reconfigure gdm
  permite escoger el gestor de entrada para X, o:
  dpkg-reconfigure X-Window-system
  nos permite configurar los diferentes elementos de X.

- Utiliza configuración de servicios TCP/IP por *inetd*, configuración en fichero /etc/inetd.conf; dispone de una herramienta *update-inetd* para inhabilitar o crear entradas de servicios.

- Algunos directorios de configuración especiales: /etc/cron., varios directorios donde se especifican trabajos para hacer periódicamente (mediante *crontab*); /etc/pam.d, donde PAM son módulos de autentificación.

2.1.2. Niveles de Arranque y Servicios. Un primer punto importante en el análisis del comportamiento local del sistema es su posible funcionamiento en los llamados niveles de ejecución, y en los servicios que se proporcionan en cada nivel.

Un servicio es una funcionalidad proporcionada por la máquina.

La activación o parada de servicios se realiza mediante la utilización de *scripts*. La mayoría de los servicios estándar, los cuales suelen tener su configuración en el directorio /etc, suelen controlarse mediante los *scripts* presentes en /etc/init.d/. En este directorio suelen aparecer *scripts* con nombres similares al servicio donde van destinados, y se suelen aceptar parámetros de activación o parada. Se realiza:

/etc/init.d/servicio start	arranque del servicio.
/etc/init.d/servicio stop	parada del servicio.
/etc/init.d/servicio restart	parada y posterior arranque del servicio.

Cuando un sistema GNU/Linux arranca, primero se carga el *kernel* del sistema, después se inicia el primer proceso, denominado *init*, que es el responsable de ejecutar y activar el resto del sistema, mediante la gestión de los niveles de ejecución (o *runlevels*).

Los niveles típicos suelen ser (puede haber algunas diferencias):

RunLevel	Función	Ejecución
0	Parada	Finaliza servicios y programas activos, así como desmonta *filesystems* activos para la CPU
1	Monousuario	Finaliza la mayoría de los servicios, permitiendo solo la entrada del administrador (root). Se usa para tareas de mantenimiento y corrección de errores
2	Multiusuario sin Red	No se inician servicios de Red, permitiendo solo entradas locales al sistema
3	Multiusuario	Inicia todos los servicios excepto los graficos asociados a X Windows
4	Multiusuario	No suele usarse, tipicamente es igual que 3
5	Multiusuario	Igual que 3, pero con soporte X para la entrada de usuarios (login gráfico)
6	Reinicio	Para todos los programas y servicios. Reinicia el sistema

Tabla 1. Gestión de Niveles de Ejecución

Estos niveles suelen estar configurados en los sistemas GNU/Linux (yUNIX) por dos sistemas diferentes, el BSD, o el System V. En el caso de Red Hat y Debian, se utiliza el sistema System V, que es el que explicaremos, pero otros UNIX y alguna distribución GNU/Linux (comoSlackware) utilizan el modelo BSD.

En el caso del modelo *runlevel* de System V, cuando el proceso *init* arranca, utiliza un fichero de configuración llamado /etc/inittab para decidir el modo de ejecución en el que va a entrar. En este fichero se define el *runlevel* por defecto en arranque, y una serie de servicios de terminal por activar para atender la entrada del usuario.

Después, el sistema, según el *runlevel* escogido, consulta los ficheros contenidos en /etc/rcn.d, donde *n* es el número asociado al *runlevel*, en el que se encuentra una lista de servicios por activar o parar en caso de que arranquemos en el *runlevel*, o lo abandonemos. Dentro del directorio encontraremos una serie de *scripts* o enlaces a los
scripts que controlan el servicio.

Cada *script* posee un nombre relacionado con el servicio, una S o K inicial que indica si es el *script* para iniciar (S) o matar (K) el servicio, y un número que refleja el orden en que se ejecutarán los servicios.

Una serie de comandos de sistema sirven de ayuda para manejar los niveles de ejecución, cabe mencionar:

- Los *scripts*, que ya hemos visto, en /etc/init.d/ nos permiten arrancar, parar o reiniciar servicios individuales.

- *telinit*, nos permite cambiar de nivel de ejecución, sólo tenemos que indicar el número. Por ejemplo, necesitamos hacer una tarea crítica en *root*; sin usuarios trabajando, podemos hacer un *telinit* 1 (también puede usarse S) para pasar a *runlevel* monousuario, y después de la tarea un *telinit 3* para volver a multiusuario. También puede utilizarse el comando *init* para la misma tarea, aunque *telinit* aporta algún parámetro extra. Por ejemplo, el reinicio típico de un sistema UNIX se hacía con *sync; sync; sync; init 6*, el comando *sync* fuerza el vaciado de los *buffers* del sistema de archivos, y luego reiniciamos en *runlevel* 6.

- *shutdown*, permite parar ('h' de *halt*) o reiniciar el sistema ('r' de *reboot*). Puede darse también un intervalo de tiempo para hacerse, o bien inmediatamente. Para estas tareas también existen los comandos *halt* y *reboot*.

- *wall*, permite enviar mensajes de advertencia a los usuarios del sistema. Concretamente, el administrador puede anunciar que se va a parar la máquina en un determinado momento. Comandos como *shutdown* suele utilizarlo de forma automática.

- *pidof*, permite averiguar el PID (*process ID*) asociado a un proceso. Con *ps* obtenemos los listados de procesos, y si queremos eliminar un servicio o proceso mediante *kill*, necesitaremos su PID.

Respecto a todo el modelo de arranque, las distribuciones presentan algún pequeño cambio:

- Red Hat: el *runlevel* 4 no tiene un uso declarado. Los directorios /etc/rcn.d existen como enlaces hacia subdirectorios de /etc/rc.d, donde están centralizados los *scripts* de arranque. Los directorios son, así: /etc/rc.d/rcn.d; pero como existen los enlaces, es transparente al usuario.

  El *runlevel* por defecto es el 5 con arranque con X un programa llamado *prefdm* gestiona el escritorio preferido.

  Los comandos y ficheros relacionados con el arranque del sistema están en los paquetes de software *sysvinit* y *initscripts*.

  Respecto a los cambios de ficheros y guiones en Red Hat, cabe destacar: en /etc/sysconfig podemos encontrar archivos que especificanvalores por defecto de la configuración de dispositivos o servicios. El guión /etc/rc.d/rc.sysinit es invocado una vez cuando el sistema

arranca; el guión /etc/rc.d/rc.local se invoca al final del proceso de carga y sirve para indicar configuraciones específicas de la máquina.

El arranque real de los servicios se hace por medio de los guiones almacenados en /etc/rc.d/init.d. Hay también un enlace desde /etc/init.d. Además, Red Hat proporciona unos *scripts* de utilidad para manejar servicios: /sbin/service para parar o iniciar un servicio por el nombre; y /sbin/chkconfig, para añadir enlaces a los ficheros S y K necesarios para un servicio.

- Debian dispone de comandos de gestión de los *runlevels* como *updaterc.d*, que permite instalar o borrar servicios arrancándolos o parándolos en uno o más *runlevels*; *invoke-rc.d*, permite las clásicas acciones de arrancar, parar o reiniciar el servicio.

  El *runlevel* por defecto en Debian es el 2, el X Window System no se gestiona desde /etc/inittab, sino que existe el gestor (por ejemplo, *gdm* o *kdm*) como si fuera un servicio más del *runlevel* 2.

2.1.3. Observar el Estado del Sistema. Una de las principales tareas del administrador (*root*) en su día a día, será verificar el correcto funcionamiento del sistema y vigilar la existencia de posibles errores o de saturación de los recursos de la máquina (memoria, discos, etc.). Pasaremos a detallar en los siguientes subapartados los métodos básicos para examinar el estado del sistema en un determinado momento y llevar a cabo las acciones necesarias para evitar problemas posteriores.

En el taller final de esta unidad, realizaremos un examen completo del sistema para que se puedan ver algunas de estas técnicas.

2.1.4. Arranque del Sistema. En el arranque de un sistema GNU/Linux, se produce todo un volcado de información interesante; cuando el sistema arranca, suelen aparecen los datos de detección de las características de la máquina, detección de dispositivos, arranque de servicios de sistema, etc., y se mencionan los problemas aparecidos.

En la mayoría de las distribuciones esto puede verse en la consola del sistema directamente durante el proceso de arranque. Sin embargo, o la velocidad de los mensajes o algunas modernas distribuciones que los ocultan tras carátulas gráficas pueden impedir seguir los mensajes correctamente, con lo que necesitaremos una serie de herramientas para este proceso.

Básicamente, podemos utilizar:

- Comando *dmesg*: da los mensajes del último arranque del *kernel*.

- Fichero /var/*log*/messages: *log* general del sistema, que contiene los mensajes generados por el *kernel* y otros *daemons*.

- Comando *uptime*: indica cuánto tiempo hace que el sistema está activo.

- Ficheros /proc: ficheros que utiliza el *kernel* para almacenar la información que gestiona.

Kernel: Directorio /proc

El *kernel* durante su arranque pone en funcionamiento un *pseudofilesystem* llamado /proc, donde vuelca la información que recopila de la máquina, así como muchos de sus datos internos. El directorio /proc está implementado sobre memoria y no se guarda en disco. Los datos contenidos son tanto de naturaleza estática como dinámica (varían durante la ejecución).

Hay que tener en cuenta que al ser /proc fuertemente dependiente del *kernel*, propicia que su estructura dependa del *kernel* de que dispone el sistema y la estructura y los ficheros incluidos pueden cambiar (los comentados pertenecen a una serie 2.4 del *kernel*).

Una de las características interesantes es que en el directorio /proc podremos encontrar las imágenes de los procesos en ejecución, junto con la información que el *kernel* maneja acerca de ellos. Cada proceso del sistema se puede encontrar en el directorio /proc/pid-proceso, donde hay un directorio con ficheros que representan su estado. Esta información es útil para programas de depuración, o bien para los propios comandos del sistema como *ps* o *top*, que pueden utilizarla para ver el estado de los procesos.

Por otra parte, en /proc podemos encontrar otros ficheros de estado global del sistema. Comentamos brevemente algunos ficheros que podremos examinar:

Fichero	Descripción
/proc/bus	Directorio de Información con los buses de PCI y USB
/proc/cdmline	Linea de Arranque del Kernel
/proc/cpuinfo	Información de la CPU
/proc/devices	Dispositivos del sistema de caracteres o bloques
/proc/drive	Información de algunos módulos de hardware
/proc/filesystems	Sistema de ficheros habilitados en el Kernel
/proc/ide	Directorio de información del bus IDE, características de disco
/proc/interrups	Mapa de interrupciones hardware (IRQ) utilizadas
/proc/ioports	Puertos de E/S utilizados
/proc/meminfo	Datos del uso de la memoria
/proc/modules	Modulos del Kernel
/proc/net	Directorio con toda la información de la red
/proc/pci	Dispositivos PCI del sistema
/proc/scsi	Dispositivos SCSI o IDE emulados por SCSI
/proc/version	Versión y fecha actual de sistema

2.1.5. Procesos. Los procesos que se encuentren en ejecución en un determinado momento serán, en general, de diferente naturaleza. Podemos encontrar:

- Procesos de sistema, o bien procesos asociados al funcionamiento local de la máquina y del *kernel*, o bien procesos (denominados *daemons*) asociados al control de diferentes servicios, ya sean locales, o de red, porque estamos ofreciendo el servicio (actuamos de servidor) o estamos recibiendo el servicio (actuamos de clientes). La mayoría de estos procesos aparecerán asociados al usuario *root*, aunque no estemos presentes en ese momento como usuarios. Puede haber algunos servicios asociados a otros usuarios de sistema (lp, bin, www, mail, etc.), estos son usuarios “virtuales” que utiliza el sistema para ejecutar ciertos procesos.

- Procesos del usuario administrador: en caso de actuar como *root*, nuestros procesos interactivos o aplicaciones lanzadas también aparecerán como procesos asociados al usuario *root*.

- Procesos de usuarios del sistema: asociados a la ejecución de sus aplicaciones, ya sea tareas interactivas en modo texto o en modo gráfico.

Como comandos rápidos y más útiles podemos utilizar:

- *ps*: el comando estándar, lista los procesos con sus datos de usuario, tiempo, identificador de proceso y línea de comandos usada. Una de las opciones utilizada es *ps ef*, pero hay muchas opciones disponibles (ver *man*).

- *top*: una versión que nos da una lista actualizada a intervalos.

- *kill*: nos permite eliminar procesos del sistema mediante el envío de señales al proceso como, por ejemplo, la de terminación *kill -9 PID*, donde indicamos el identificador del proceso. Útil para procesos con comportamiento inestable o programas interactivos que han dejado de responder.

Logs del Sistema

Tanto el *kernel* como muchos de los *daemons* de servicios, así como diferentes aplicaciones o subsistemas de GNU/Linux, pueden generar mensajes que vayan a parar a ficheros *log*, ya sea para tener una traza de su funcionamiento, o bien para detectar errores o advertencias de malfuncionamiento o situaciones críticas. Este tipo de *logs* son imprescindibles en muchos casos para las tareas de administración y se suele emplear bastante tiempo de administración en el procesamiento y análisis de sus contenidos.

La mayor parte de los *logs* se generan en el directorio /var/log, aunque algunas aplicaciones pueden modificar este comportamiento; la mayoría de *logs* del propio sistema sí que se encuentran en este directorio.

Un *daemon* particular del sistema (importante) es el *daemon* Syslogd. Este *daemon* se encarga de recibir los mensajes que se envían por parte del *kernel* y otros *daemons* de servicios y los envía a un fichero *log* que se encuentra en /var/*log*/messages. Éste es el fichero por defecto, pero Syslogd es también configurable (fichero /etc/syslog.conf), de manera que se puede generar otro fichero o bien, según el *daemon* que envía el fichero, dirigirlo a un *log* u a otro (clasificar por fuente) o clasificar los mensajes por importancia: *alarm*, *warning*, *error*, *critical* , etc.

Dependiendo de la distribución, puede estar configurado de diferentes modos por defecto, en /var/log suele generar (por ejemplo) en Debian ficheros como: *kern.log*, *mail.err*, *mail.info*, ... que son los *logs* de diferentes servicios. Podemos examinar la configuración para determinar de dónde provienen los mensajes y en qué ficheros los guarda. Una opción que suele ser útil es la posibilidad de enviar los mensajes a una consola virtual de texto, de manera que podremos ir viéndolos a medida que se produzcan. Esto suele ser útil para monitorizar la ejecución del sistema sin tener que estar mirando el fichero a cada momento. Una modificación simple de este método podría ser introducir, desde un terminal, la instrucción siguiente:

tail -f /var/log/messages

Esta sentencia nos permite dejar el terminal o ventana de terminal, de manera que irán apareciendo los cambios que se produzcan en el fichero.

Otros comandos relacionados:

- *uptime*: tiempo que hace que el sistema está activo.

- *last*: analiza *log* de entradas/salidas del sistema (/var/*log*/wtmp).

Memoria

Respecto a la memoria del sistema, tendremos que tener en cuenta que disponemos: a) de la memoria física de la propia máquina, b) memoria virtual, que puede ser direccionada por los procesos. Normalmente, no dispondremos de cantidades demasiado grandes, de modo que la memoria física será menor que el tamaño de memoria virtual necesario. Esto obligará a utilizar una zona de intercambio (*swap*) sobre disco.

Esta zona de intercambio (*swap*) puede implementarse como un fichero en el sistema de archivos, pero es más habitual encontrarla como una partición de intercambio (llamada de *swap*), creada durante la instalación del sistema. En el momento de particionar el disco, se declara como de tipo Linux Swap.

Para examinar la información sobre memoria tenemos varios métodos y comandos útiles:

- Fichero /etc/fstab: aparece la partición *swap* (si existiese). Con un comando *fdisk* podemos averiguar su tamaño.

- Comando *ps*: permite conocer qué procesos tenemos, y con las opciones de porcentaje y memoria usada.

- Comando *top*: es una versión *ps* dinámica actualizable por periodos de tiempo. Puede clasificar los procesos según la memoria que usan o el tiempo de CPU.

- Comando *free*: informa sobre el estado global de la memoria. Da también el tamaño de la memoria virtual.

- Comando *vmstat*: informa sobre el estado de la memoria virtual, y el uso que se le da.

Discos y Filesystems

Examinaremos qué discos tenemos disponibles, cómo están organizados y de qué particiones y archivos de sistemas (*filesystems*) disponemos. Para crear una partición y asociarla a un determinado *filesystem* accesible, tendremos que realizar un proceso de montaje, ya sea explícitamente o bien programada en arranque. En el proceso de montaje se conecta el sistema de archivos asociado a la partición a un punto del árbol de directorios.

Para conocer los discos (o dispositivos de almacenamiento) que tenemos en el sistema, podemos basarnos en la información de arranque del sistema (*dmesg*), donde se detectan los presentes, como los /dev/hdx para los dispositivos IDE o los SCSI con dispositivos /dev/sdx.

Otros dispositivos, como discos duros conectados por USB, discos flash (los de tipo *pen drive*), unidades zips, CD-ROM externos, suelen ser dispositivos con algún tipo de emulación SCSI, por lo que se verán como dispositivo de este tipo.

Cualquier dispositivo de almacenamiento presentará una serie de particiones de su espacio. Típicamente, un disco IDE soporta un máximo de 4 particiones físicas, o más si éstas son lógicas (que permiten colocar varias particiones de este tipo sobre una física). Cada partición puede contener tipos de *filesystems* diferentes, ya sean de un mismo operativo o de operativos diferentes.

Para examinar la estructura de un dispositivo conocido, o cambiar su estructura particionando el disco, podemos utilizar el comando *fdisk*, o cualquiera de sus variantes más o menos interactivas (*cfdisk, sfdisk*).

- Fichero /etc/fstab. Indica dispositivos que están preparados para montarse en el arranque o los extraíbles que podrán ser montados. No tienen por qué estar todos los del sistema, sino sólo aquellos que queramos tener en arranque. Los demás podemos montarlos bajo demanda con el comando *mount*, o desmontarlos con *umount*.

- Comando *mount*. Nos informa de los *filesystems* montados en ese momento (ya sean dispositivos reales o *filesystem* virtuales como /proc). Podemos obtener esta información también desde el fichero /etc/mtab.

- Comando *df -k*. Nos informa de los *filesystems* de almacenamiento, y nos permite verificar el espacio usado y disponible. Comando básico para controlar espacio de disco disponible.

Respecto a este ultimo comando *df –k*, una de nuestras tareas básicas de administración de la máquina es controlar los recursos de la máquina, y en este caso el espacio disponible en los

filesystems utilizados. Estos tamaños hay que monitorizarlos con cierta frecuencia para evitar la caída del sistema; nunca tendría que dejarse un *filesystem* (y más si es el /) por debajo de un 10 o 15%, ya que hay muchos procesos *daemons* que están escribiendo normalmente información temporal o *logs*, que pueden generar gran información; un caso particular lo forman los ficheros *core,* ya comentados, que pueden suponer (dependiendo del proceso) tamaños muy grandes de archivo. Normalmente, habrá que tener algunas precauciones de "limpieza del sistema" si se detectan situaciones de saturación del *filesystem*:

Eliminar temporales antiguos. Los directorios /tmp y /var/tmp suelen acumular muchos archivos generados por diferentes usuarios o aplicaciones. Algunos sistemas o distribuciones ya toman medidas de limpieza, como limpiar /tmp en cada arranque del sistema.

- *Logs*: evitar su crecimiento excesivo, según la configuración del sistema (por ejemplo de Syslogd) la información generada de mensajes puede ser muy grande. Normalmente, habrá que limpiar periódicamente al llegar a determinados tamaños, y en todo caso, si necesitamos la información para posteriores análisis, podemos realizar *backups* en medios extraíbles.

- Hay otros puntos del sistema que suelen crecer mucho, como pueden ser: a) ficheros core de los usuarios: podemos eliminarlos periódicamente o eliminar su generación; b) el sistema de correo electrónico: almacena todos los correos enviados y recibidos, podemos pedir a los usuarios que hagan limpieza periódica, o bien poner sistemas de cuotas; c) las cachés de los navegadores u otras aplicaciones: también suelen tener tamaños grandes, otra limpieza que habrá que hacer periódicamente; d) las cuentas de los propios usuarios: pueden tener cuotas para no exceder los tamaños prefijados.

Sistema de Ficheros

En cada máquina con un sistema GNU/Linux podemos encontrarnos con diferentes sistemas de ficheros de diferentes tipos [Hin00].

Para empezar, es habitual encontrarse con los propios sistemas de ficheros Linux creados en diversas particiones de los discos [Koe01]. La configuración habitual suele ser de dos particiones: la correspondiente a "/" (*root filesystem*) y la correspondiente al fichero de intercambio o de *swap*.

La de *swap* es de tipo Linux *swap*, y la correspondiente a / suele ser de alguno de los sistemas de ficheros estándar, ya sea ext2 (el tipo por defecto hasta los *kernels* 2.4), o el nuevo ext3, que es una mejora del ext2 compatible pero con *journaling*, lo que permite tener un *log* de lo que va pasando al sistema de ficheros, para recuperaciones más rápidas en caso de error.

Otra configuración habitual puede ser de 3 particiones: /, *swap*, /home, donde la /home se dedicará a las cuentas de los usuarios. Esto permite separar las cuentas de los usuario del sistema, aislando en dos particiones separadas, y podemos dar el espacio necesario para las cuentas en otra partición.

Otro esquema muy utilizado es el de separar en particiones las partes estáticas del sistema de las dinámicas, por ejemplo, una partición donde se coloca / con la parte estática (/bin /sbin y /usr en algunos casos) que se espera que no va a crecer o lo va a hacer muy poco, y otra o varias con la parte dinámica (/var /tmp /opt), suponiendo que /opt, por ejemplo, es el punto de instalación del software nuevo. Esto permite ajustar mejor el espacio de disco y dejar más espacio para las partes del sistema que lo necesiten.

Puntos de Montaje

Aparte del *filesystem* principal / y de sus posibles divisiones en particiones extras (/usr /var /tmp /home), cabe tener en cuenta la posibilidad de dejar puntos de montaje preparados para el montaje de otros *filesystems*, ya sea particiones de disco u otros dispositivos de almacenamiento.

En las máquinas en que GNU/Linux comparte la partición con otros sistemas operativos, mediante algún sistema de arranque (*lilo* o *grub*), pueden existir varias particiones asignadas a los diferentes operativos.

Muchas veces es interesante compartir datos con estos sistemas, ya sea para leer sus ficheros o modificarlos. A diferencia de otros sistemas (que sólo tienen en cuenta sus propios datos, léase Windows, en el cual en algunas versiones no se soportan algunos de sus propios sistemas de ficheros), GNU/Linux es capaz de tratar con una cantidad enorme de sistemas de ficheros de diferentes operativos y poder compartir la información.

Dependiendo de la distribución, se usan unos u otros, o también los podemos crear nosotros. Normalmente suelen existir o bien como subdirectorios de la raíz, por ejemplo /cdrom /win /floppy, o bien son subdirectorios dentro de /mnt, el punto estándar de montaje (aparecen como /mnt/cdrom /mnt/floppy).

El proceso de montaje se realiza mediante la orden *mount* con el siguiente formato:

mount -t filesystem-type device mount-point

El tipo de *filesystem* puede ser msdos (fat), vfat (fat32), ntfs (ntfs lectura), iso9660 (para cdrom)

El dispositivo es la entrada correspondiente en el directorio /dev a la localización del dispositivo, los IDE tenían /dev/hdxy donde *x* es a,b,c, o d (1 master, 1 slave, 2 master, 2 slave) e *y*, el número de partición, los SCSI (/dev/sdx) donde *x* es a,b,c,d ... (según el ID SCSI asociado 0,1,2,3,4 ...).

Si estas particiones son más o menos estables en el sistema (o sea, no cambian frecuentemente) y las queremos utilizar, lo mejor será incluir los montajes para que se hagan en tiempo de ejecución, mediante la configuración del fichero /etc/fstab:

/etc/fstab

Por ejemplo, esta configuración incluye algunos de los sistemas estándar, como la raíz en /dev/hda2, la partición de *swap* que está en hdb3, el sistema *proc* (que utiliza el *kernel* para guardar su información).

Y el disquete, el CD-ROM, y en este caso un disco USB de tipo Flash (que se ve como un dispositivo SCSI). En algunos casos, se especifica *auto* como tipo de *filesystem*. Esto permite que lo autodetecte.

Si se conoce, es mejor darlo en la configuración, y por otra parte, el *noauto* en las opciones permite que no sea montado de forma automática siempre, sino bajo petición.

Si tenemos esta información en el fichero, el proceso de montaje se simplifica mucho, ya que se hará o bien en ejecución o bien bajo demanda (para los *noauto*). Y puede hacerse ahora simplemente pidiendo que se monte el dispositivo o el punto de montaje:

mount /mnt/cdrom
mount /dev/fd0

dado que el sistema ya tiene el resto de la información.

El proceso contrario, el desmontaje, es bastante sencillo, el comando *umount* con punto o dispositivo:

umount /mnt/cdrom
umount /dev/fd0

En el caso de medios extraíbles, tipo CD-ROM (u otros), puede usarse *eject* para la extracción del medio:

eject /dev/cdrom o, en este caso, sólo: eject

Los comandos *mount* y *umount* montan o desmontan todos los sistemas disponibles. En el fichero /etc/mtab se mantiene una lista de los sistemas montados en un momento concreto se puede consultar o ejecutar *mount* sin parámetros para obtener esta información.

Permisos

Otro tema que habrá que controlar en el caso de los ficheros y directorios es el de los permisos que queremos establecer en cada uno de ellos, recordando que cada fichero puede disponer de la serie de permisos: *rwxrwxrwx* donde se corresponden con *rwx* del propietario, *rwx* del grupo al que el usuario pertenece, y *rwx* para otros usuarios.

En cada uno se puede establecer el permiso de lectura (*r*), escritura (*w*) o ejecución (*x*). En el caso de un directorio, *x* denota el permiso para poder entrar en ese directorio (con el comando *cd*, por ejemplo).

Para modificar los derechos sobre un directorio o fichero, existen los comandos:

- *chown*: cambiar propietario de los ficheros.
- *chgrp*: cambiar grupo de los ficheros.
- *chmod*: cambiar permisos específicos (*rwx*) de los archivos.

Estos comandos también permiten la opción *–R*, que es recursiva si se trata de un directorio.

Usuarios y grupos

Los usuarios de un sistema GNU/Linux disponen normalmente de una cuenta asociada con sus datos, junto con el espacio en disco para que puedan desarrollar sus archivos y directorios. Este espacio está asignado al usuario, y sólo puede ser usado por éste (a no ser que los permisos especifiquen cosas diferentes.

Dentro de las cuentas asociadas a usuarios podemos encontrar diferentes tipos:

• La del administrador, con identificador *root*, que sólo es (o debería ser) utilizada para las operaciones de administración. El usuario *root* es el que dispone de más permisos y acceso completo a la máquina y a los archivos de configuración. Por lo tanto, también es el que más daño puede causar por errores u omisiones. Es mejor evitar usar la cuenta de *root* como si fuese un usuario más, por lo que se recomienda dejarla sólo para operaciones de administración.

• Cuentas de usuarios: las cuentas normales para cualquier usuario de la máquina tienen los permisos restringidos al uso de ficheros de su cuenta, y a algunas otras zonas particulares (por ejemplo, los temporales en /tmp), así como a utilizar algunos dispositivos para los que se le haya habilitado.

• Cuentas especiales de los servicios: *lp*, *news*, *wheel*, *www-data*, ... cuentas que no son usadas por personas, sino por servicios internos del sistema, que los usa bajo estos nombres de usuario. Algunos de los servicios también son usados bajo el nombre de *root*.

Un usuario normalmente se crea mediante la especificación de un nombre (o identificador de usuario), una palabra de paso (*password*) y un directorio personal asociado (la cuenta).

La información de los usuarios del sistema está incluida en los siguientes archivos:

/etc/passwd
/etc/shadow
/etc/group
/etc/gshadow

Comandos para esta administración de usuarios:

- *useradd*: añadir un usuario al sistema.
- *userdel*: borrar un usuario del sistema.
- *usermod*: modificar un usuario del sistema.
- *groupadd, groupdel, groupmod* lo mismo para grupos.
- *newusers, chpasswd*: pueden ser de utilidad en grandes instalaciones con muchos usuarios, ya que permiten crear varias cuentas desde la información introducida en un

fichero (*newusers*) o bien cambiar las contraseñas a un gran número de usuarios (*chpasswd*).

- *chsh*: cambiar el *shell* de *login* del usuario.

- *chfn*: cambiar la información del usuario, la presente en el comentario del fichero /etc/passwd.

- *passwd*: cambia la contraseña de un usuario. Puede ejecutarse como usuario, y entonces pide la contraseña nueva y la vieja. En el caso de hacerlo, el *root* tiene que especificar el usuario al que va a cambiar la contraseña (si no, estaría cambiando la suya) y no necesita la contraseña antigua. Es quizás el comando más usado por el *root*, de cara a los usuarios cuando se les olvida la contraseña antigua.

- *su*: una especie de cambio de identidad. Lo utilizan tanto usuarios, como el *root* para cambiar el usuario actual. En el caso del administrador, es bastante utilizado para testar que la cuenta del usuario funcione correctamente; hay diferentes variantes: su (sin parámetros, sirve para pasar a usuario *root*, siempre que se tenga el *passwd* de *root*, nos permite, cuando estamos en una cuenta de usuario, pasar a *root* para hacer alguna tarea). La sentencia *su iduser* (cambia el usuario a *iduser*, pero dejando el entorno como está, o sea, en el mismo directorio, ...).

Respecto a la administración de usuarios y grupos, lo que hemos comentado aquí hace referencia a la administración local de una sola máquina. En sistemas con múltiples máquinas que comparten los usuarios suele utilizarse otro sistema de gestión de la información de los usuarios. Estos sistemas, denominados genéricamente sistemas de información de red, como NIS, NIS+ o LDAP, utilizan bases de datos para almacenar la información de los usuarios y grupos, de manera que se utilizan máquinas servidoras, donde se almacena la base de datos, y otras máquinas clientes, donde se consulta esta información.

Podemos comprobar si estamos en un entorno de tipo NIS si en las líneas *passwd* y *group* del archivo de configuración /etc/nsswitch.conf aparece *compat*, si estamos trabajando con los ficheros locales, o bien *nis* o *nisplus* según el sistema con que estemos trabajando. En general, para el usuario simple no supone ninguna modificación, ya que la gestión de las máquinas le es transparente, y más si se combina con ficheros compartidos por NFS que permite disponer de su cuenta sin importar con qué máquina. La mayor parte de los comandos anteriores pueden seguir usándose sin problema bajo NIS o NIS+, son equivalentes a excepción del cambio de contraseña, que en lugar de *passwd*, se suele hacer con *yppasswd* (NIS) o *nispasswd* (NIS+);

pero el administrador los puede renombrar (por un enlace) a *passwd*, y los usuarios no notarán la diferencia.

Veremos este modo de configuración en las unidades de administración de red.

Servidores de Impresión

El sistema de impresión de GNU/Linux está heredado de la variante BSD de UNIX; este sistema se denominaba LPD (*Line Printer Daemon*). Éste un sistema de impresión muy potente, ya que integra capacidades para gestionar tanto impresoras locales como de red. Y ofrece dentro del mismo, tanto el cliente como
el servidor de impresión.

LPD es un sistema bastante antiguo, ya que se remonta a las orígenes de la rama BSD de UNIX (mediados de los ochenta). Por lo tanto, a LPD le suele faltar soporte para los dispositivos modernos, ya que en origen el sistema no estuvo pensado para el tipo de impresoras actuales.

El sistema LPD no estuvo pensado como un sistema basado en controladores de dispositivo, ya que normalmente se producían sólo impresoras serie o paralelas de escritura de caracteres texto.

Para la situación actual, el sistema LPD, se combina con otro software común, como el sistema Ghostscript, que ofrece salida de tipo *postscript* para un rango muy amplio de impresoras para las que posee controladores. Además, se suele combinar con algún software de filtrado, que según el tipo de documento a imprimir, selecciona filtros
adecuados. Así, normalmente el proceso que se sigue es (básicamente):

1. El trabajo es iniciado por un comando del sistema LPD.

2. El sistema de filtro identifica qué tipo de trabajo (o fichero) es utilizado y convierte el trabajo a un fichero *postscript* de salida, que es el que se envía a la impresora. En GNU/Linux y UNIX, la mayoría de aplicaciones suponen que la salida será hacia una impresora *postscript*, y muchas de ellas generan salida *postscript* directamente, y por esta razón se necesita el siguiente paso.

3. Ghostscript se encarga de interpretar el fichero *postscript* recibido, y según el controlador de la impresora a la que ha sido enviado el trabajo, realiza la conversión al formato propio

de la impresora; si es de tipo *postscript*, la impresión es directa, si no, habrá que realizar la traducción. El trabajo se manda a la cola de impresión.

Además del sistema de impresión LPD (con origen en los BSD UNIX), también existe el denominado sistema System V (de origen en la otra rama UNIX de System V). Normalmente, por compatibilidad, actualmente la mayor parte de UNIX integran ambos, de manera que o bien uno u otro es el principal, y el otro se simula sobre el principal.

En el caso de GNU/Linux, pasa algo parecido, según la instalación que hagamos podemos tener sólo los comandos LPD de sistema de impresión, pero también será habitual disponer de los comandos System V. Una forma sencilla de identificar los dos sistemas (BSD o System V), es con el comando principal de impresión (el que envía los trabajos al sistema), en BSD es lpr, y en System V es lp.

Éste era el panorama inicial de los sistemas de impresión de GNU/Linux, pero en los últimos años han surgido más sistemas, que permiten una mayor flexibilidad y una mayor disposición de controladores para las impresoras. Los dos principales sistemas
son LPRng y CUPS.

Los dos son una especie de sistema de más alto nivel, pero que no se diferencia en mucho de cara al usuario respecto a los BSD y System V estándar, por ejemplo, se utilizan los mismos comandos clientes para imprimir. Para el administrador sí que supondrá diferencias, ya que los sistemas de configuración son diferentes. En cierto modo podemos considerar a LPRng y CUPS como nuevas arquitecturas de sistemas de impresión, que son compatibles de cara al usuario con los comandos antiguos.

En las distribuciones GNU/Linux actuales podemos encontrarnos con los diferentes sistemas de impresión. Si la distribución es antigua, puede que lleve incorporado tan sólo el sistema BSD LPD; en las actuales: Red Hat utiliza LPRng (por defecto), pero puede cambiarse a CUPS (hay una herramienta, Print switch, que permite cambiar el sistema).

En el caso de Fedora Core, el sistema de impresión por defecto es CUPS (desapareciendo LPRng), y la herramienta Print switch ya no existe por no ser necesaria. Debian por defecto utiliza BSD LPD, pero es común instalar CUPS, y también puede utilizar LPRng. Además, cabe recordar que también teníamos la posibilidad (vista en la unidad de migración) de interaccionar con sistemas Windows mediante protocolos Samba, que permitían compartir las impresoras y el acceso a éstas.

- Sistema de filtros usado, cada sistema de impresión soporta uno o varios.

- Y los controladores de las impresoras, en GNU/Linux hay muchos, podemos mencionar por ejemplo, controladores de: CUPS, propios o de los fabricantes (por ejemplo HP y Epson los proporcionan); Gimp, el programa de retoque de imágenes también posee *drivers* optimizados para la impresión de imágenes; Foomatic es un sistema de gestión de controladores que funciona con la mayoría de sistemas (CUPS, LPD, LPRng, y otros); los controladores de Ghostscript, etc. En casi todas las impresoras tienen uno o
- más controladores de estos conjuntos.

Respecto a la parte cliente del sistema, los comandos básicos son iguales para los diferentes sistemas, y éstos son los comandos del sistema BSD (cada sistema soporta emulación de estos comandos:

- *lpr*: envía un trabajo a la cola de la impresora por defecto (o a la que se selecciona), el *daemon* de impresión (lpd) se encarga de enviarlo a la cola correspondiente, y asigna un número de trabajo, que será usado con los otros comandos. Normalmente, la impresora por defecto estaría indicada por una variable de sistema PRINTER, o se utilizará la primera que exista definida, o en algunos sistemas se utiliza la cola *lp*.

  Ejemplo de *lpr*:
  lpr -Pepson datos.txt
  Esta instrucción mandaría el fichero datos.txt a la cola de impresión asociada a una impresora que hemos definido como “epson”.

- *lpq*: nos permite examinar los trabajos existentes en la cola.

```
# lpq -P epson
Rank    Owner   Job Files                 Total Size
1st     juan    15 datos.txt              74578 bytes
2nd     marta   16 fpppp.F                12394 bytes
```

Este comando nos muestra los trabajos en cola, con el orden y tamaños de éstos; los ficheros pueden aparecer con nombres diferentes, ya que depende de si los hemos enviado con *lpr*, o con otra aplicación que puede cambiarlos de nombre al enviarlos, o
si han tenido que pasar por algún filtro al convertirlos.

- *lprm*: elimina trabajos de la cola, podemos especificar un número de trabajo, o un usuario para cancelar los trabajos.

lprm -Pepson 15
Eliminar el trabajo con *id* 15 de la cola.

Respecto a la parte administrativa (en BSD), el comando principal sería *lpc*; este comando permite activar, desactivar colas, mover trabajos en el orden de las colas y activar o desactivar las impresoras (se pueden recibir trabajos en las colas pero no se envían a las impresoras).

Cabe mencionar asimismo que, para el caso de System V, los comandos de impresión suelen también estar disponibles, normalmente simulados sobre los de BSD. En el caso cliente, los comandos son: *lp*, *lpstat, cancel* y para temas de administración: *lpadmin, accept, reject, lpmove, enable, disable, lpshut.*

En las siguientes secciones veremos cómo hay que configurar un servidor de impresión para los tres sistemas principales. Estos servidores sirven tanto para la impresión local, como para atender las impresiones de clientes de red (si están habilitados).

2.2. ADMINISTRACION DE RED

El sistema operativo UNIX (GNU/Linux) es tomado como ejemplo de una arquitectura de comunicaciones estándar. Desde el mítico UUCP (servicio de copia entre sistemas operativos UNIX) hasta las redes actuales, UNIX siempre ha mostrado su versatilidad en aspectos relacionados a la comunicación y el intercambio de información.

Con la introducción de redes de ordenadores (área local LAN, área amplia WAN o las más actuales área metropolitana MAN) ofreciendo enlaces multipunto a diferentes velocidades (56kbits/seg hasta 1Gbit/seg), han ido surgiendo nuevos servicios basados en protocolos más rápidos, portables entre diferentes ordenadores y mejor adaptados, como el TCP/IP (Transport Control Program / Internet Protocol). [Com01, Mal96, Cis00, Gar98, KD00]

2.2.1 TCP/IP Suite. El protocolo TCP/IP sintetiza un ejemplo de estandarización y una voluntad de comunicación a nivel global.

El protocolo TCP/IP es en realidad un conjunto de protocolos básicos que se han ido agregando a principal para satisfacer las diferentes necesidades en la comunicación ordenador-ordenador como son TCP, UDP, IP, ICMP, ARP. [Mal96]

La utilización más frecuente de TCP/IP para el usuario en la actualidad son la conexión remota a otros ordenadores (telnet, SSH *Secure Shell*), la utilización de ficheros remotos (*Network File System* NFS) o su transferencia (*File Transfer Protocol* FTP, *Hipertext Markup Protocol,* HTTP).

2.2.1.1 Servicios sobre TCP/IP. Los servicios TCP/IP tradicionales más importantes son:

- Transferencia de archivos: el *File Transfer Protocol* (FTP) permite a un usuario de un ordenador obtener/enviar archivos de un ordenador hacia otro ordenador. Para ello, el usuario deberá tener una cuenta en el ordenador remoto e identificarse a través de su nombre (*login*) y una palabra clave (*password*) o a ordenadores donde existe un repositorio de información (software, documentación, ...), y el usuario se conectará como anónimo (*anonymous*) para transferir (leer) estos archivos a su ordenador. Esto no es lo mismo que los más recientes sistemas de archivos de red, NFS, *Network File System*, (o protocolos netbios sobre tcp/ip, "invento" totalmente inseguro sobre Windows y que es mejor reemplazar por una versión más antigua pero más segura del mismo concepto llamado netbeui) que permiten virtualizar el sistema de archivos de una máquina para que pueda ser accedido en forma interactiva sobre otro ordenador.

- Conexión (*login*) remota: el protocolo de terminal de red (telnet) permite a un usuario conectarse a un ordenador remotamente. El ordenador local se utiliza como terminal del ordenador remoto y todo es ejecutado sobre éste permaneciendo el ordenador localinvisible desde el punto de vista de la sesión. Este servicio en la actualidad se ha reemplazado por el SHH (*Secure Shell*) por razones de seguridad. En una conexión remota mediante telnet, los mensajes circulan tal cual (texto plano), o sea, si alguien "observa" los mensajes en la red, equivaldrá a mirar la pantalla del usuario. SSH codifica la información (que significa un coste añadido a la comunicación) que hace que los paquetes en la red sean ilegibles a un nodo extraño.

- eMail: este servicio permite enviar mensajes a los usuarios de otros ordenadores. Este modo de comunicación se ha transformado en un elemento vital en la vida de los usuarios y permiten que los *e-mails* (correos electrónicos) sean enviados a un servidor central para que después puedan ser recuperados por medio de programas específicos (clientes) o leídos a través de una conexión web.

El avance de la tecnología y el bajo coste de los ordenadores ha permitido que determinados servicios se hayan especializado en ello y se ofrecen configurados sobre determinados ordenadores trabajando en un modelo cliente-servidor. Un servidor es un sistema que ofrece un servicio específico para el resto de la red. Un cliente es otro ordenador que utiliza este servicio. Todos estos servicios generalmente son ofrecidos dentro de TCP/IP:

- Sistemas de archivos en red (*Network File Systems*): permite a un sistema acceder a los archivos sobre un sistema remoto en una forma más integrada que FTP. Los dispositivos

de almacenamiento (o parte de ellos) son exportados hacia el sistema que desea acceder y éste los puede "ver" como si fueran dispositivos locales. Este protocolo permite a quien exporta poner las reglas y la formas de acceso, lo que (bien configurado) hace independiente el lugar donde se encuentra la información físicamente del sitio donde se "ve" la información.

- Impresión remota: permite acceder a impresoras conectadas a otros ordenadores.

- Ejecución remota: permite que un usuario ejecute un programa sobre otro ordenador. Existen diferentes maneras de realizar esta ejecución: o bien a través de un comando (*rsh*, ssh, *rexec*) o a través de sistemas con RPC (*Remote Procedure Call*) que permiten a un programa en un ordenador local ejecutar una función de un programa sobre otro ordenador. Los mecanismos RPC han sido objeto de estudio y existen diversas implementaciones, pero las más comunes son Xerox's Courier y Sun's RPC (esta última adoptada por la mayoría de los UNIX).

- Servidores de nombre (*name servers*): en grandes instalaciones existen un conjunto de datos que necesitan ser centralizados para mejorar su utilización, por ejemplo, nombre de usuarios, palabras claves, direcciones de red, etc. Tod ello facilita que un usuario disponga de una cuenta para todas las máquinas de una organización. Por ejemplo, Sun's Yellow Pages (NIS en las versiones actuales de *sun's*) está diseñado para manejar todo este tipo de datos y se encuentra disponible para la mayoría de UNIX. El DNS (*Domain Name System*) es otro servicio de nombres pero que guarda una relación entre el nombre de la máquina y la identificación lógica de esta máquina (dirección IP).

- Servidores de terminal (*terminal servers*): conecta terminales a un servidor que ejecuta telnet para conectarse al ordenador central. Este tipo de instalaciones permite básicamente reducir costes y mejorar las conexiones al ordenador central (en determinados casos).

- Servidores de terminales gráficas (*network-oriented window systems*): permiten que un ordenador pueda visualizar información gráfica sobre un *display* que está conectado a otro ordenador. El más común de estos sistemas es X Window.

2.2.1.2 ¿Qué es TCP/IP?

TCP/IP son en realidad dos protocolos de comunicación entre ordenadores independientes uno del otro.

Por un lado, TCP (transmission control protocol), define las reglas de comunicación para que un ordenador (host) pueda 'hablar' con otro (si se toma como referencia el modelo de comunicaciones OSI/ISO se describe la capa 4, ver tabla siguiente).

TCP es orientado a conexión, es decir, equivalente a un teléfono, y la comunicación se trata como un flujo de datos (*stream*).

Por otro lado, IP (*Internet Protocol*), define el protocolo que permite identificar las redes y establecer los caminos entre los diferentes ordenadores.

Es decir, encamina los datos entre dos ordenadores a través de las redes. Corresponde a la capa 3 del modelo OSI/ISO y es un protocolo sin conexión (ver tabla siguiente) [Com01, Rid00, Dra99]. Una alternativa al TCP la conforma el protocolo UDP (*User Datagram Protocol*), el cual trata los datos como un mensaje (datagrama) y envía paquetes. Es un protocolo sin conexión (el ordenador destino no debe necesariamente estar escuchando cuando un ordenador establece comunicación con él) y tiene la ventaja de que ejerce una menor sobrecarga a la red que las conexiones de TCP, pero la comunicación no es fiable (los paquetes pueden no llegar o llegar duplicados).

Existe otro protocolo alternativo llamado ICMP (*Internet Control Message Protocol*). ICMP se utiliza para mensajes de error o control. Por ejemplo, si uno intenta conectarse un equipo (*host*), el ordenador local puede recibir un mensaje ICMP indicando *"host unreachable*". ICMP también puede ser utilizado para extraer información sobre una red. ICMP es similar a UDP, ya que maneja mensajes (datagramas), pero es más simple que UDP, ya que no posee identificación de puertos (los puertos son buzones donde se depositan los paquetes de datos y desde donde las aplicaciones servidoras leen dichos paquetes) en el encabezamiento del mensaje.

En el modelo de comunicaciones de la OSI/ISO (*OSI, Open Systems Interconnection Reference Model, ISO, International Standards Organization*), es un modelo teórico adoptado por muchas redes. Existen siete capas de comunicación donde cada una tiene una interfaz para comunicarse con la anterior y la posterior:

En resumen, TCP/IP es una familia de protocolos (que incluyen IP, TCP, UDP) que proveen un conjunto de funciones a bajo nivel utilizadas por la mayoría de las aplicaciones. [KD00, Dra99].

Algunos de los protocolos que utilizan los servicios mencionados han sido diseñados por Berkeley, Sun u otras organizaciones. Ellos no forman oficialmente parte de *Internet Protocol*

Suite (IPS). Sin embargo, son implementados utilizando TCP/IP y por lo tanto considerados como parte formal de IPS. Una descripción de los protocolos disponibles
en Internet puede consultarse la RFC 1011 (ver referencias sobre RFC [IET03]) que lista todos los protocolos disponibles. Existe actualmente una nueva versión del protocolo IPv6, también llamado IPng (IP *next generation*) que reemplaza al IPv4. Este protocolo mejora notablemente el anterior en temas tales como mayor número de nodos, control de tráfico, seguridad o mejoras en aspectos de *routing* .

2.2.1.3. Dispositivos Físicos (Hardware) de Red. Desde el punto de vista físico (capa 1 del modelo OSI), el hardware más utilizado para LAN es conocido como Ethernet (o FastEthernet o GigaEthernet). Sus ventajas son su bajo coste, velocidades aceptables (10, 100, o 1,000 megabits por segundo) y facilidad en su instalación.

Interconexión por cables (pares) trenzados y conectores similares a los telefónicos (se conocen como RJ45). La conexión par trenzado es conocida como 10baseT o 100baseT (según la velocidad) y utiliza repetidores llamados *hubs* como puntos de interconexión. La tecnología Ethernet utiliza elementos intermedios de comunicación (*hubs*, *switchs*, *routers*) para configurar múltiples segmentos de red y dividir el tráfico para mejorar las prestaciones de transferencia de información.

Existe además otro tipo de hardware menos común, pero no menos interesante como es ATM (*Asynchronous Transfer Mode*). Este hardware permite montar LAN con una calidad de servicio elevada y es una buena opción cuando deben montarse redes de alta velocidad y baja latencia, como por ejemplo aquellas que involucren distribución de vídeo en tiempo real.

Existe otro hardware soportado por GNU/Linux para la interconexión de ordenadores, entre los cuales podemos mencionar: Frame Relay o X.25 (utilizada en ordenadores que acceden o interconectan WAN y para servidores con grandes necesidades de transferencias de datos), Packet Radio (interconexión vía radio utilizando protocolos como AX.25, NetRom o Rose) o dispositivos *dialing up*, que utilizan líneas series, lentas pero muy baratas, a través de módems analógicos o digitales (RDSI, DSL, ADSL, etc.). Estas últimas son las que comúnmente se utilizan en pymes o uso doméstico y requieren otro protocolo para la transmisión de paquetes, tal como SLIP o PPP. Para virtualizar la diversidad de hardware sobre una red, TCP/IP define una interfaz abstracta mediante la cual se concentrarán todos los paquetes que serán enviados por un dispositivo físico (lo cual también significa una red o un segmento de esta red). Por ello, por cada dispositivo de comunicación en la máquina tenderemos una interfaz correspondiente en el *kernel* del sistema operativo.

En GNU/Linux puede significar tener que incluir los módulos adecuados para el dispositivo (*network interface card* NIC) adecuado (en el *kernel* o como módulos), esto significa compilar el *kernel* después de haber escogido con, porejemplo, *make menuconfig* el NIC adecuado, indicándole como interno o como módulo (en este último caso se deberá compilar el módulo adecuado también).

Los dispositivos de red se pueden mirar en el directorio /dev que es donde existe un archivo (especial, ya sea de bloque o de caracteres según su transferencia), que representa a cada dispositivo hardware.

2.2.2 Conceptos en TCP/IP. Como se ha observado, la comunicación significa una serie de conceptos que ampliaremos a continuación:

Internet/intranet: el término *intranet* se refiere a la aplicación de tecnologías de Internet (red de redes) dentro de una organización básicamente para distribuir y tener disponible información dentro de la compañía. Por ejemplo, los servicios ofrecidos por GNU/Linux como servicios Internet e intranet incluyen correo electrónico, WWW, *news*, etc.

- Nodo: se denomina nodo (*host*) a una máquina que se conecta a la red (en un sentido amplio un nodo puede ser un ordenador, una impresora, una torre (*rack*) de CD, etc.), es decir, un elemento activo y diferenciable en la red que reclama o presta algún servicio
- y/o comparte información.

- Dirección de red Ethernet (*Ethernet address* o *MAC address*): un número de 48 bits (por ejemplo 00:88:40:73:AB:FF –en octal– 0000 0000 1000 1000 0100 0000 0111 0011 1010 1011 1111 1111 en binario) que se encuentra en el dispositivo físico (hardware) del controlador (NIC) de red Ethernet y es grabado por el fabricante del mismo (este número debe ser único en el mundo, por lo que cada fabricante de NIC tiene un rango preasignado).

- *Host name*: cada nodo debe tener además un único nombre en la red. Ellos pueden ser sólo nombres o bien utilizar un esquema de nombres jerárquico basado en dominios (*hierarchical domain naming scheme*). Los nombres de los nodos deben ser únicos, lo cual resulta fácil en pequeñas redes, más dificultoso en redes extensas e imposible en Internet si no se realiza algún control. Los nombres deben ser de un máximo de 32 caracteres entre a-zA-Z0-9.-, y que no contengan espacios o # comenzando por un carácter alfabético.

- Dirección de Internet (*IP address*): está compuesto por cuatro números en el rango 0-255 separados por puntos (por ejemplo 192.168.0.1) y es utilizado universalmente para

identificar los ordenadores sobre una red o Internet. La traslación de nombres en direcciones IP es realizada por un servidor DNS (*Domain Name System*) que transforma los nombres de nodo (legibles por humanos) en direcciones IP (este servicio es realizado por una aplicación denominada *named*).

- Puerto (*port*): identificador numérico del buzón en un nodo que permite que un mensaje (TCP, UDP) pueda ser leído por una aplicación concreta dentro de este nodo (por ejemplo, dos máquinas que se comuniquen por telnet lo harán por el puerto 23, pero las
- dos mismas máquinas pueden tener una comunicación ftp por el puerto 21). Se pueden tener diferentes aplicaciones comunicándose entre dos nodos a través de diferentes puertos simultáneamente.

- Nodo *router* (*gateway*): es un nodo que realiza encaminamientos (transferencia de datos *routing*). Un *router*, según sus características, podrá transferir información entre dos redes de protocolos similares o diferentes y puede ser además selectivo.

- *Domain Name System* (DNS): permite asegurar un único nombre y facilitar la administración de las bases de datos que realizan la traslación entre nombre y dirección de Internet y se estructuran en forma de árbol. Para ello, se especifican dominios separados por puntos, de los que el más alto (de derecha a izquierda) describe una categoría, institución o país (COM, comercial, EDU, educación, GOV, gubernamental, MIL, militar (gobierno), ORG, sin fin de lucro, XX dos letras por país, ...). El segundo nivel representa la organización, el tercero y restantes departamentos, secciones o divisiones dentro de una organización (por ejemplo, www.uoc.edu o nteum@pirulo.remix.es). Los dos primeros nombres (de derecha a izquierda, *uoc.edu* en el primer caso, *remix.es* (en el segundo) deben ser asignados (aprobados) por el SRI-NIC (órgano mundial gestor de Internet) y los restantes pueden ser configurados/asignados por la institución.

- DHCP, bootp: DHCP y bootp son protocolos que permiten a un nodo cliente obtener información de la red (tal como la dirección IP del nodo). Muchas organizaciones con gran cantidad de máquinas utilizan este mecanismo para facilitar la administración en grandes redes o donde existe una gran cantidad de usuarios móviles.

- ARP, RARP: en algunas redes (como por ejemplo IEEE 802 LAN que es el estándar para Ethernet), las direcciones IP son descubiertas automáticamente a través de dos protocolos miembros de *Internet protocol suite: Address Resolution Protocol* (ARP) y *Reverse Address Resolution Protocol* (RARP). ARP utiliza mensajes (*broadcast messages*) para determinar la dirección Ethernet (especificación MAC de la capa 3 del modelo OSI) correspondiente a una dirección de red particular (IP). RARP utiliza mensajes de tipo *broadcast* (mensaje que

llega a todos los nodos) para determinar la dirección de red asociada con una dirección hardware en particular. RARP es especialmente importante en máquinas sin disco, en las cuales la dirección de red generalmente no se conoce en el momento del inicio (*boot*).

- Biblioteca de *sockets*: en UNIX toda la implementación de TCP/IP forma parte del *kernel* del sistema operativo (o bien dentro del mismo o como un módulo que se carga en el momento del inicio como el caso de GNU/Linux con los controladores de dispositivos).

2.2.3 Como se asigna una dirección Internet. Esta dirección es asignada por el NIC y tiene dos campos. El izquierdo representa la identificación de la red y el derecho la identificación del nodo. Considerando lo mencionado anteriormente (4 números entre 0-255, o sea 32 bits o cuatro bytes), cada byte representa o bien la red o bien el nodo. La parte de red es asignada por el NIC y la parte del nodo es asignada por la institución o el proveedor).

Existen algunas restricciones: 0 (por ejemplo, 0.0.0.0) en el campo de red está reservado para el *routing* por defecto y 127 (por ejemplo, 127.0.0.1) es reservado para la autorreferencia (*local loopback* o *local host*), 0 en la parte de nodo se refiere a esta red (por ejemplo, 192.168.0.0) y 255 es reservado para paquetes de envío a todas las máquinas (*broadcast*) (por ejemplo, 198.162.255.255). En las diferentes asignaciones se puede tener diferentes tipos de redes o direcciones:

- Clase A (*red*.*host*.*host*.*host*): 1.0.0.1 a 126.254.254.254 (126 redes, 16 millones de nodos) definen las grandes redes. El patrón binario es 0 + 7 bits red + 24 bits de nodos.

- Clase B (*red.red.host*.*host*): 128.1.0.1 a 191.255.254.254 (16K redes, 65K nodos) generalmente se utiliza el primer byte de nodo para identificar subredes dentro de una institución). El patrón binario es 10 + 14 bits de red + 16 bits de nodos.

- Clase C (*red.red.red.host*): 192.1.1.1 a 223.255.255.254 (2 millones de bits de redes, 254 de nodos). El patrón binario es 110 + 21 bits red + 8 bits de nodos.

- Clase D y E (*red*.*red*.*red*.*host*): 224.1.1.1 a 255.255.255.254 reservado para *multicast* (desde un nodo a un conjunto de nodos que forman parte de un grupo) y propósitos experimentales.

Algunos rangos de direcciones han sido reservados para que no correspondan a redes públicas, sino a redes privadas (máquinas que se conectan entre ellas sin tener conexión con el exterior) y los mensajes no serán encaminados a través de Internet, lo cual es comúnmente conocido como Intranet). Éstas son para la clase A 10.0.0.0 hasta 10.255.255.255, clase B

172.16.0.0 hasta 172.31.0.0 y clase C 192.168.0.0 hasta 192.168.255.0. La dirección de *broadcast* es especial, ya que cada nodo en una red escucha todos los mensajes (además de su propia dirección). Esta dirección permite que datagramas (generalmente información de *routing* y mensajes de aviso) puedan ser enviados a una red y todos los nodos del mismo segmento de red los puedan leer. Por ejemplo, cuando ARP busca encontrar la dirección Ethernet correspondiente a una IP, éste utiliza un mensaje de *broadcast*, el cual es enviado a todas las máquinas de la red simultáneamente. Cada nodo en la red lee este mensaje y compara la IP que se busca con la propia y le retorna un mensaje al nodo que hizo la pregunta si hay coincidencia.

Dos conceptos complementarios a lo descrito anteriormente es el de subredes y *routing* entre ellas. Subredes significa subdividir la parte del nodo en pequeñas redes dentro de la misma red para, por ejemplo, mejorar el tráfico. Una subred toma la responsabilidad de enviar el tráfico a ciertos rangos de direcciones IP extendiendo el mismo concepto de redes A, B, C, pero sólo aplicando esta redirección en la parte nodo de la IP. El número de bits que son interpretados como identificador de la subred es dado por una máscara de red (*netmask*) que es un número de 32 bits (igual que la IP). Para obtener el identificador de la subred, se deberá hacer una operación lógica Y (*AND*) entre la máscara y la IP, lo cual dará la IP de la subred. Por ejemplo, sea una institución que tienen una red clase B con número 172.17.0.0, y su *netmask* es, por lo tanto, 255.255.0.0. Internamente, esta red está formada por pequeñas redes (una planta del edificio por ejemplo). Así, el rango de direcciones es reasignado en 20 subnets (plantas para nosotros) 172.17.1.0 hasta 172.17.20.0. El punto que conecta todas estas plantas (*backbone*) tiene su propia dirección, por ejemplo 172.17.1.0.

2.2.4 ¿Cómo se debe construir una red?

2.2.4.1 Configuración de la interfaz NIC. Una vez cargado el *kernel* de GNU/Linux, éste ejecuta el comando *init* que a su vez lee el archivo de configuración /etc/inittab y comienza el proceso de inicialización. Generalmente, el *inittab* tiene secuencias tales como: *si::sysinit:/etc/init.d/boot* , que representa el nombre del archivo de comandos (*script*) que controla las secuencias de inicialización. Generalmente este *script* llama a otros *scripts*, entre los cuales se encuentra la inicialización de la red.

En Debian se ejecuta etc/init.d/network para la configuración de la interfaz de red y en función del nivel de arranque; por ejemplo, en el 2 se ejecutarán todos los ficheros S* del directorio /etc/rc2.d (que son enlaces al directorio /etc/initd), y en el nivel de apagado, todos los K* del mismo directorio. De este modo, el *script* está sólo una vez (/etc/init.d) y de acuerdo a los servicios deseados en ese estado se crea un enlace en el directorio correspondiente a la configuración del nodo-estado.

Los dispositivos de red se crean automáticamente cuando se inicializa el hardware correspondiente. Por ejemplo, el controlador de Ethernet crea las interfaces *eth*[0..*n*] secuencialmente cuando se localiza el hardware correspondiente.

A partir de este momento, se puede configurar la interfaz de red, lo cual implica dos pasos: asignar la dirección de red al dispositivo e inicializar los parámetros de la red al sistema. El comando utilizado para ello es el *ifconfig* (*interface configure*). Un ejemplo será:

ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up

Lo cual indica configurar el dispositivo *eth0* con dirección IP 192.168.110.23 y máscara de red 255.255.255.0. El *up* indica que la interfaz pasará al estado activo (para desactivarla debería ejecutarse *ifconfig eth0 down*). El comando asume que si algunos valores no se indican, son tomados por defecto. En este caso, el *kernel* configurará esta máquina como Tipo-C y configurará la red con 192.168.110.23 y la dirección de *broadcast* con 192.168.110.255.
Por ejemplo:

ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up

Existen comandos como el *ifup* e *ifdown*, que permite configurardesconfigurar la red en forma más simple utilizando el archivo /etc/network/interfaces para obtener todos los parámetros necesarios (consultar *man interfaces* para su sintaxis).

2.2.4.2. Configuración del Name Resolver. El siguiente paso es configurar el *name resolver* que convierte nombres tales como *pirulo.remix.com* en 192.168.110.23. El archivo /etc/resolv.conf es el utilizado para tal fin. Su formato es muy simple (una línea de texto por sentencia). Existen tres palabras clave para tal fin: *domain* (dominio local), *search* (lista de dominios alternativos) y *name server* (la dirección IP del *Domain Name Server*).

Un archivo importante es el /etc/*host*.conf, que permite configurar el comportamiento del *name* resolver. Su importancia reside en indicar dónde se resuelve primero la dirección o el nombre de un nodo. Esta consulta puede ser realizada al servidor DNS o a tablas locales dentro de la máquina actual (/etc/hosts).

Esta configuración indica que primero se verifique el /etc/hosts antes de solicitar una petición al DNS y también indica (2.ª línea) que retorne todas las direcciones válidas que se encuentren en /etc/hosts. Por lo cual, el archivo /etc/hosts es donde se colocan las direcciones locales o también sirve para acceder a nodos sin tener que consultar al DNS.

La consulta es mucho más rápida, pero tiene la desventaja de que si el nodo cambia, la dirección será incorrecta. En un sistema correctamente configurado, sólo deberán aparecer el nodo local y una entrada para la interfaz *loopback*.

Para el nombre de una máquina pueden utilizarse alias, que significa que esa máquina puede llamarse de diferentes maneras para la misma dirección IP. En referencia a la interfaz *loopback,* éste es un tipo especial de interfaz que permite realizar a nodo conexiones consigo misma (por ejemplo, para verificar que el subsistema de red funciona sin acceder a la red). Por defecto, la dirección IP 127.0.0.1 ha sido asignada específicamente al *loopback* (un comando telnet 127.0.0.1 conectará con la misma máquina). Su configuración es muy fácil (la realizan generalmente los *script* de inicialización de red).

2.2.4.3. Configuración del *routing.* Otro aspecto que hay que configurar es el *routing* . Si bien existe el tópico sobre su dificultad, generalmente se necesitan unos requerimientos de *routing* muy simples. En un nodo con múltiples conexiones, el *routing* consiste en decidir dónde hay que enviar y qué se recibe. Un nodo simple (una sola conexión de red) también necesita *routing*, ya que todos los nodos disponen de un *loopback* y una conexión de red (por ejemplo, Ethernet, PPP, SLIP, ...). Como se explicó anteriormente, existe una tabla llamada *routing table* que contiene filas con diversos campos, pero con tres campos sumamente importantes: dirección de destino, interfaz por donde saldrá el mensaje y dirección IP, que efectuará el siguiente paso en la red (*gateway*).

El comando *route* permite modificar esta tabla para realizar las tareas de *routing* adecuadas. Cuando llega un mensaje, se mira su dirección destino, se compara con las entradas en la tabla y se envía por la interfaz en la cual la dirección que mejor coincide con el destino del paquete. Si un *gateway* es especificado, se envía a la interfaz adecuada.

2.2.4.4. Configuración del *inetd.* El siguiente paso en la configuración de red es la configuración de los servidores y servicios que permitirán a otro usuario acceder a la máquina local o a sus servicios. Los programas servidores utilizarán los puertos para escuchar las peticiones de los clientes, los cuales se dirigirán a este servicio como *IP:port*. Los servidores pueden funcionar de dos maneras diferentes: *standalone* (en el cual el servicio escucha en el puerto asignado y siempre se encuentra activo) o a través del *inetd*.

El inetd es un servidor que controla y gestiona las conexiones de red de los servicios especificados en el archivo /etc/inetd.conf, el cual, ante una petición de servicio, pone en marcha el servidor adecuado y le transfiere la comunicación.

Dos archivos importantes necesitan ser configurados: /etc/services y /etc/inetd.conf. En el primero se asocian los servicios, los puertos y el protocolo y en el segundo que programas servidores responderán ante una petición a un puerto determinado. El formato de /etc/services es name port/protocol aliases, donde el primer campo es nombre del servicio, el segundo, el puerto donde atiende este servicio y el protocolo que utiliza, y el siguiente un alias del nombre.

El archivo /etc/inetd.conf es la configuración para el servicio maestro de red (*inetd server daemon*). Cada línea contiene siete campos separados por espacios: *service socket_type proto flags user server_path server_args*, donde *service* es el servicio descrito en la primera columna de /etc/services, *socket_type* es el tipo de *socket* (valores posibles *stream*, *dgram*, *raw*, *rdm*, o *seqpacket*), *proto* es el protocolo válido para esta entrada (debe coincidir con el de /etc/services), *flags* indica la acción que tomar cuando existe una nueva conexión sobre un servicio que se encuentra atendiendo a otra conexión (*wait* le dice a *inetd* no poner en marcha un nuevo servidor o *nowait* significa que inetd debe poner en marcha un nuevo servidor). *user* será el usuario con el cual se identificará quien ha puesto en marcha el servicio, *server_path* es el directorio donde se encuentra el servidor y *server_args* son argumentos posibles que serán pasados al servidor. Un ejemplo de algunas líneas de /etc/inetd.conf es (recordar que # significa comentario, por lo cual, si un servicio tiene # antes de nombre, significa que no se encuentra disponible):

2.2.4.5. Configuración adicional: *protocols* y *networks.* Existen otros archivos de configuración que en la mayoría de los casos no se utilizan pero que pueden ser interesantes. El /etc/protocols es un archivo que relaciona identificadores de protocolos con nombres de protocolos, así, los programadores pueden especificar los protocolos por sus nombres en los programas.

El archivo /etc/networks tiene una función similar a /etc/hosts, pero con respecto a las redes, indica nombres de red en relación con su dirección IP (el comando *route* mostrará el nombre de la red y no su dirección en este caso).

2.2.4.6. Aspectos de seguridad. Es importante tener en cuenta los aspectos de seguridad en las conexiones a red, ya que una fuente de ataques importantes se produce a través de la red. Ya se hablará más sobre este tema en la unidad correspondiente a seguridad; sin embargo, hay unas cuantas recomendaciones básicas que deben tenerse en cuenta para minimizar los riesgos inmediatamente antes y después de configurar la red de nuestro ordenador:

- No activar servicios en /etc/inetd.conf que no se utilizarán, insertar un # antes del nombre para evitar fuentes de riesgo.

- Modificar el archivo /etc/ftpusers para denegar que ciertos usuarios puedan tener conexión vía ftp con su máquina.

- Modificar el archivo /etc/securetty para indicar desde qué terminales (un nombre por línea), por ejemplo: tty1 tty2 tty3 tty4, se permite la conexión del superusuario (*root*).

Desde las terminales restantes, *root* no podrá conectarse.

- Utilizar el programa tcpd. Este servidor es un *wrapper* que permite aceptar-negar un servicio desde un determinado nodo y se coloca en el /etc/inetd.conf como intermediario de un servicio. El tcpd verifica unas reglas de acceso en dos archivos:

  /etc/hosts.allow y /etc/host.deny

  Si se acepta la conexión, pone en marcha el servicio adecuado pasado como argumento (por ejemplo, la línea del servicio de ftp antes mostrada en inetd.conf:

  ftp stream tcp nowait *root* /usr/sbin/tcpd/usr/sbin/in.ftpd.

  tcpd primero busca /etc/hosts.allow y luego /etc/hosts.deny. El archivo hosts.deny contiene las reglas de cuáles son los nodos que no tienen acceso a un servicio dentro de esta máquina. Una configuración restrictiva es ALL: ALL, ya que sólo se permitirá el acceso a los servicios desde los nodos declarados en /etc/hosts.allow.

- El archivo /etc/hosts.equiv permite el acceso a esta máquina sin tener que introducir una clave de acceso (*password*). Se recomienda no utilizar este mecanismo y aconsejar a los usuarios no utilizar el equivalente desde la cuenta de usuario a través de archivo *.rhosts*.

3. MONTAJE DE SERVICIOS BAJO LA PLATAFORMA DEL SISTEMA OPERATIVO LINUX

3.1. CONFIGURACIÓN DE SERVICIOS DE RED

Una vez terminado de instalar el sistema operativo Linux Fedora Core 4, se dispuso a configurar los programas aplicativos para la configuración de los servicios de red.

3.1.1. SERVIDOR WEB APACHE. El servidor web apache se puede configurar de varias formas:

a) Por las herramientas propias del sistema:

- Se puede ingresar por el inicio, configuración del sistema, configuración de servidores y httpd.

b) Por el webmin:

- Se puede hacer la configuración básica del servidor web. Si se le indica adicionar un servidor, se debe dar el nombre al que responderá, la dirección ip y el puerto.

Instalación de Webmin. Esta herramienta se instala para facilitar las labores de administración y operación del sistema, aunque Linux Fedore Core 4 provee utilitarios por inicio, donde se pueden crear usuarios, manejo de disco y otros servicios, se tomó la decisión de utilizar estas herramientas para la configuración de estas tareas.

En la carpeta */home/servidor/software* se aloja el instalador de webmin.

Al entrar a la linea de comandos, desde el directorio */home/servidor/software* se digita el siguiente comando

tar xvzf webmin-1200.tar.gz

Al descomprimir se crea la carpeta webmin-1.200, se corre el programa con *./setup.sh*

Se establece el usuario *admin*, con contraseña *admin.*, además se dejó el puerto por defecto 10000.

c) Manipulación directa de los archivos.

En la captura vemos la imagen de una pantalla configurada por el webmin.

Gráfica 2. Pantalla de Configuración por el Webmin.

3.1.2. Configuración de correo electrónico básico en linux

3.1.2.1. Control de Virus Y Spam. Un alto porcentaje de los correos que llegan a los usuarios de las empresas, contienen virus, gusanos, programas espías, etc. Si se tiene activado en los equipos clientes un antivirus que revise los correos entrantes o salientes, es frecuente ver mensajes como:

"Alerta de Virus"

Esta labor de chequeo de virus, se debe delegar en primera instancia, al servidor de correo, para evitar posibles infecciones en los PCs, y además rechazar todos los correos que los contengan.

Estas situaciones causan un gran desgaste en administradores, operadores, y usuarios finales.
Por tal razón se debe implementar estos controles en el servidor, para minimizar la llegada de estos tipos de correo a los usuarios.

Sendmail es un programa que proporciona el servicio de correo electrónico en sistemas Linux (y Unix.) Entre sus objetivos de diseño destaca un gran poder de configuración - casi ilimitado - capaz de procesar mensajes de email en prácticamente cualquier tipo de red.

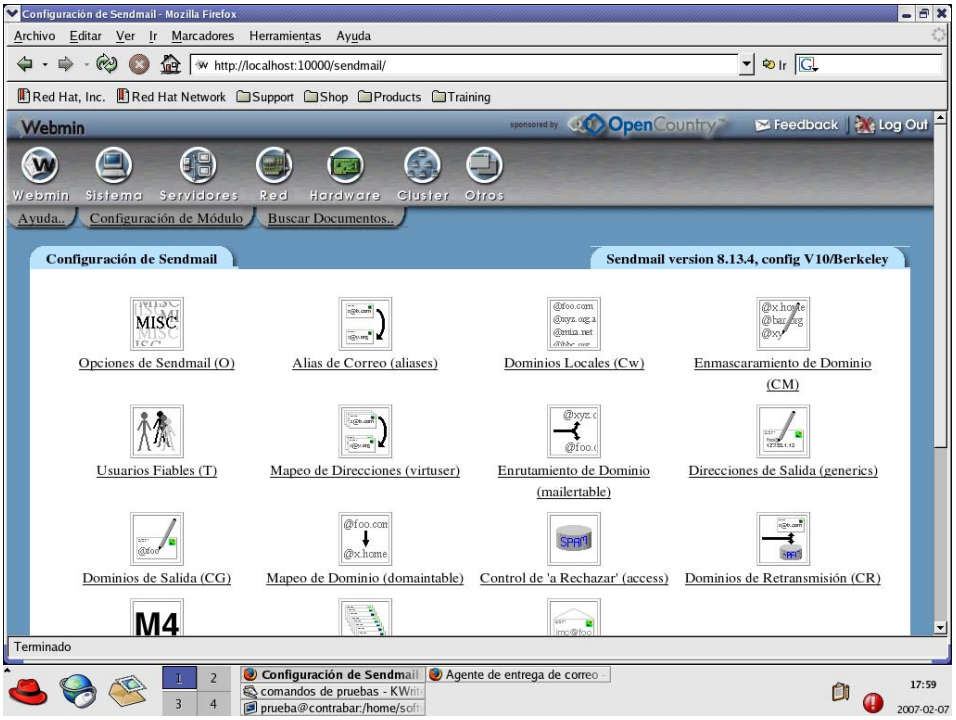
Sendmail se encarga de enviar (y reintentar de ser necesario) los mensajes redactados por los usuarios de la organización. Igualmente, recibe los mensajes dirigidos a usuarios de la organización y los coloca en sus respectivos "Buzones de correo" para su posterior lectura**.**

Sendmail es extremadamente configurable, aunque no necesariamente de un modo sencillo. Para esto posee un archivo de configuración principal que en RedHat es: /etc/mail/sendmail.cf que tiene una sintaxis poco intuitiva, y ha sido diseñado principalmente para que el computador lo lea de un modo eficiente (mas no los humanos). El archivo sendmail.cf define generalmente la ruta de otros archivos de configuración auxiliares que evitan la modificación directa del primero, simplificando la administración de Sendmail.

Las máquinas unix, traen un producto de correo nativo como es el sendmail. En estas versiones de Linux, se incorporan una serie de utilidades para configurar y mejorar la seguridad del correo, evitando SPAM, relay, etc.

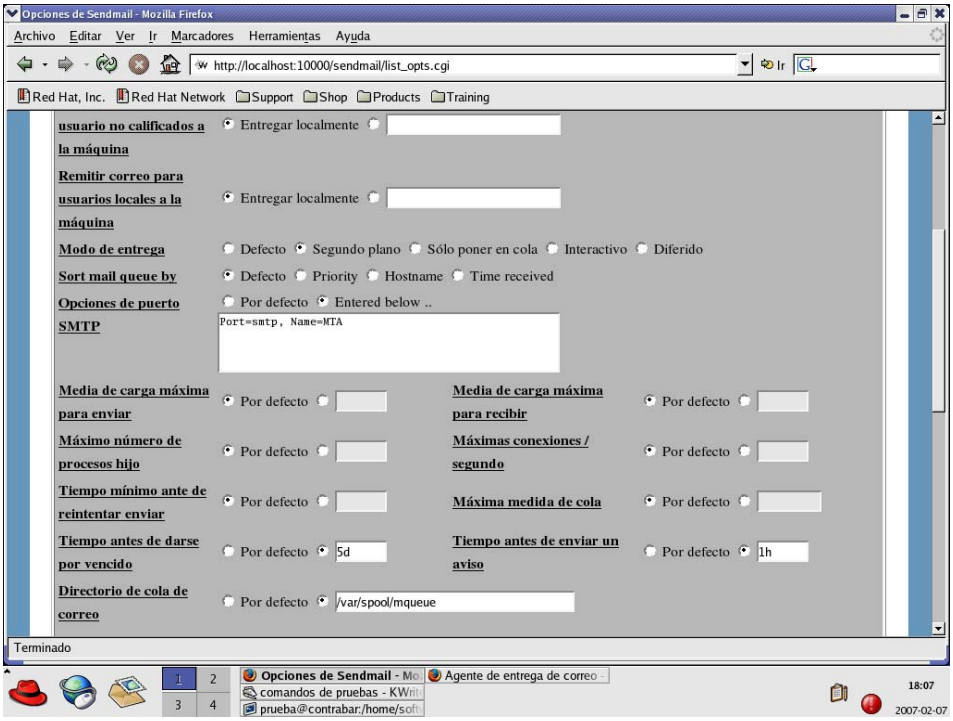
El sendmail como tal debe ya estar configurado en la máquina, y si no esta, se procede a generar un archivo sendmail.cf desde el interfaz webmin.

Se ingresa por servidores y se escoge sendmail, luego por opciones configuración del sendmail,



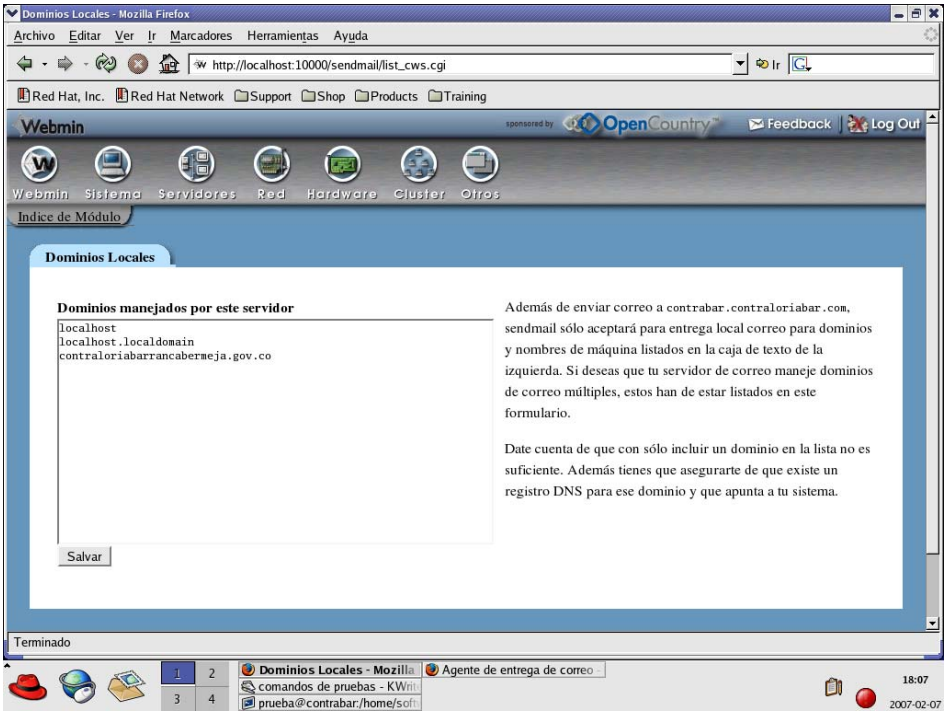
Gráfica 3. Opciones de configuración del sendmail.

En esta parte escogemos opciones de sendmail (O), Para que se pueda enviar correo a sitios en internet, se debe modificar la opción el puerto SMTP, se quita la dirección 127.0.0.1.



Gráfica 4. Opciones de configuración del sendmail sitios de internet.

Se le debe de indicar al servidor de correo, que dominio o para que dominios va a recibir:



Gráfica 5. Opciones de configuración del sendmail servidor de correo.

Luego al final de esta ventana hay un botón de salvar y aplicar.

3.1.3. Instalación, configuración mailscaner e integración. En los últimos años, se ha hecho muy común que los virus viajen como archivos adjuntos en mensajes de correo electrónico. Por esta razón los archivos con las extensiones EXE, JS, LNK, PIF, SRC y VBS entre otras han sido considerados peligrosos y no son admitidos por los servidores de correo seguros. De igual forma los correos conocidos como SPAM pueden hacer que los buzones de correo se llenen rápidamente de publicidad e informaciones que no hacer referencia al objetivo principal de la empresa. Debido a esto, surge la necesidad de implementar un mecanismo que se encargue de garantizar el filtro de correo SPAM y la no entrega de archivos infectados con virus a los usuarios finales.

Para esto se cuenta con una herramienta de filtrado MailScanner que garantiza la fiabilidad de lo correos que llegan a los buzones de los servidores, inicialmente se plantea un esquema haciendo del MailScanner un muro de fuego para la llegada de estos correos.

Mail Scanner es un spammer y scanner para correo electrónico.

Es capaz de detectar un gran número de tipos de correos electrónicos comerciales o spam.

Se instaló el producto MailScanner 4.54.6-1 integrado con el antivirus f-prot: Desde /opt se descomprimió el instalador del MailScanner. Finalmente no se pudo instalar porque nos pedía licencia y lamentablemente no la teníamos.

Conociendo que la configuración del MailScanner, se puede hacer por edición del archivo de configuración o por medio de un interfaz gráfico (como el webmin), procedimos a configurarlo por ahí.

Se instaló el módulo del webmin para administrar MailScanner por webmin, para no tener que recurrir a la configuración por manipulación del archivo. El módulo del webmin se descargó por internet. Luego desde configuración del webmin:

Fuimos al modulo de webmin, configuración del webmin, indicamos la ruta del archivo con el modulo de webmin, le adicionamos instalar modulo, y se verifico por servidores que apareciera el icono de MailScanner.

La primera vez que se ingresamos, hubo que configurar el modulo, Se guardo y pudimos ya ingresar a las opciones.

3.1.4. Servidor PROXY. El término en ingles «Proxy» tiene un significado muy general y al mismo tiempo ambiguo, aunque invariablemente se considera un sinónimo del concepto de «Intermediario». Se suele traducir, en el sentido estricto, como delegado o apoderado (el que tiene el que poder sobre otro).

Un Servidor Intermediario (Proxy) se define como una computadora o dispositivo que ofrece un servicio de red que consiste en permitir a los clientes realizar conexiones de red indirectas hacia otros servicios de red. Durante el proceso ocurre lo siguiente:

- Cliente se conecta hacia un Servidor Intermediario (Proxy).

- Cliente solicita una conexión, fichero u otro recurso disponible en un servidor distinto.
- Servidor Intermediario (Proxy) proporciona el recurso ya sea conectándose hacia el servidor especificado o sirviendo éste desde un caché.
- En algunos casos el Servidor Intermediario (Proxy) puede alterar la solicitud del cliente o bien la respuesta del servidor para diversos propósitos.

Los Servidores Intermediarios (Proxies) generalmente se hacen trabajar simultáneamente como muro cortafuegos operando en el Nivel de Red, actuando como filtro de paquetes, como en el caso de iptables, o bien operando en el Nivel de Aplicación, controlando diversos servicios, como es el caso de TCP Wrapper. Dependiendo del contexto, el muro cortafuegos también se conoce como BPD o Border Protection Device o simplemente filtro de paquetes.

Una aplicación común de los Servidores Intermediarios (Proxies) es funcionar como caché de contenido de Red (principalmente HTTP), proporcionando en la proximidad de los clientes un caché de páginas y ficheros disponibles a través de la Red en servidores HTTP remotos, permitiendo a los clientes de la red local acceder hacia éstos de forma más rápida y confiable.

Cuando se recibe una petición para un recurso de Red especificado en un URL (Uniform Resource Locator) el Servidor Intermediario busca el resultado del URL dentro del caché. Si éste es encontrado, el Servidor Intermediario responde al cliente proporcionado inmediatamente el contenido solicitado. Si el contenido solicitado no estuviera disponible en el caché, el Servidor Intermediario lo traerá desde servidor remoto, entregándolo al cliente que lo solicitó y guardando una copia en el caché. El contenido en el caché es eliminado luego a través de un algoritmo de expiración de acuerdo a la antigüedad, tamaño e historial de respuestas a solicitudes (hits) (ejemplos: LRU, LFUDA y GDSF).

Los Servidores Intermediarios para contenido de Red (Web Proxies) también pueden actuar como filtros del contenido servido, aplicando políticas de censura de acuerdo a criterios arbitrarios.

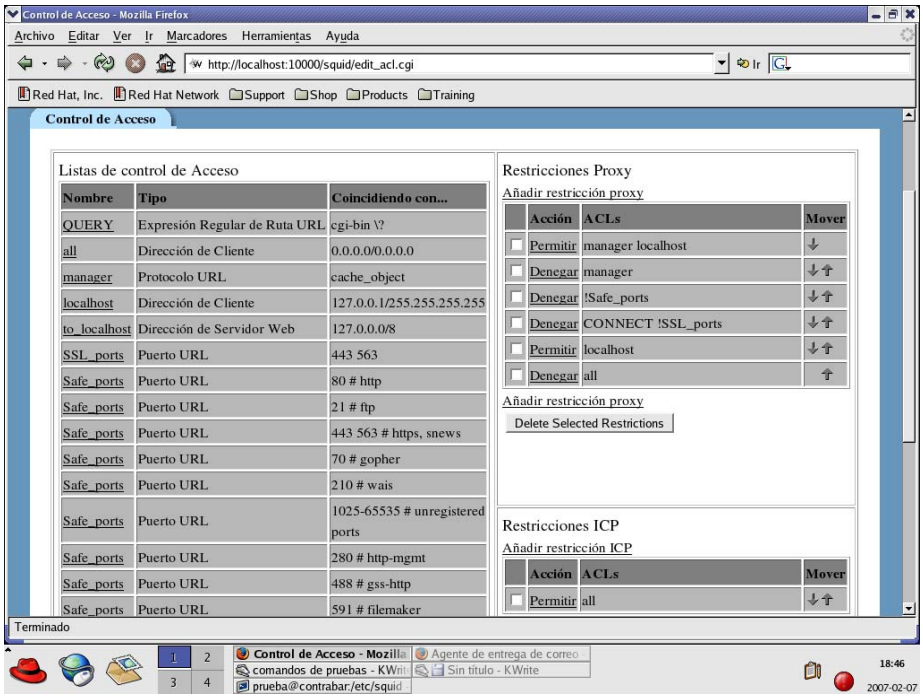
3.1.4.1. Definición de Squid. **S**quid es un Servidor Intermediario (Proxy) de alto desempeño que se ha venido desarrollando desde hace varios años y es hoy en día un muy popular y ampliamente utilizado entre los sistemas operativos como GNU/Linux y derivados de Unix®. Es muy confiable, robusto y versátil y se distribuye bajo los términos de la Licencia Pública General GNU (GNU/GPL). Siendo sustento lógico libre, está disponible el código fuente para quien así lo requiera.

Linux viene con un servicio de Proxy llamado SQUID. Este servicio nos permitirá controlar quienes pueden acceder el servicio de internet, restringir algunos sitios a donde no pueden salir y prestar el servicio de caché en disco.

También se puede manipular desde el interfaz gráfica llamada webmin, que permite configurar y administrar la mayoría de los servicios Linux (entrar por servidores y escoger squid).

Buscamos http://www.vsvear.com:100000/squid/
Debe aparecer una pantalla como: servidor Proxy squid

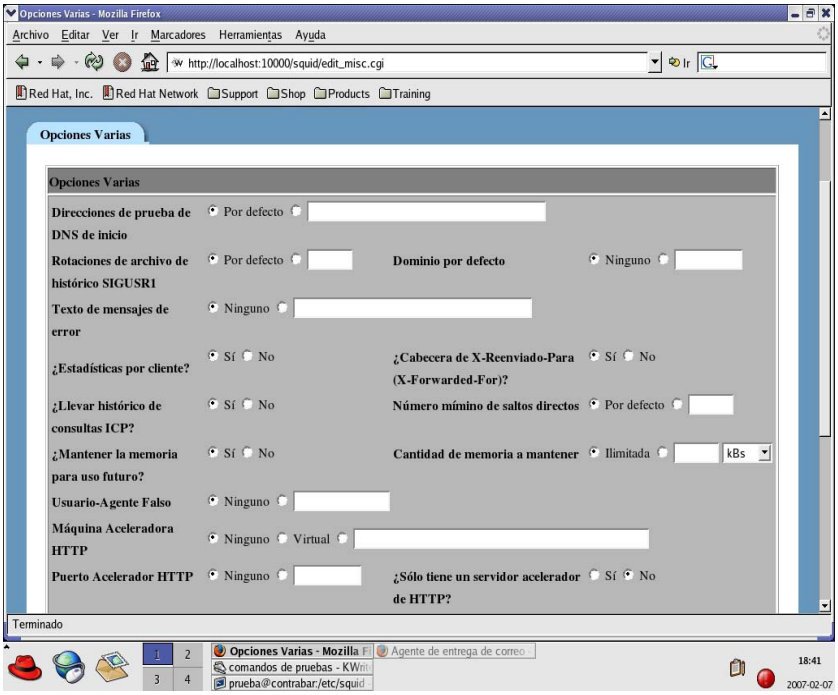
Si entramos por control de acceso podemos ver las lista de acceso mencionadas en el archivo squid.conf, tales como sitios negados, lista para elemento llamado red, etc.



Gráfica 6. Lista de correo (SQUID).

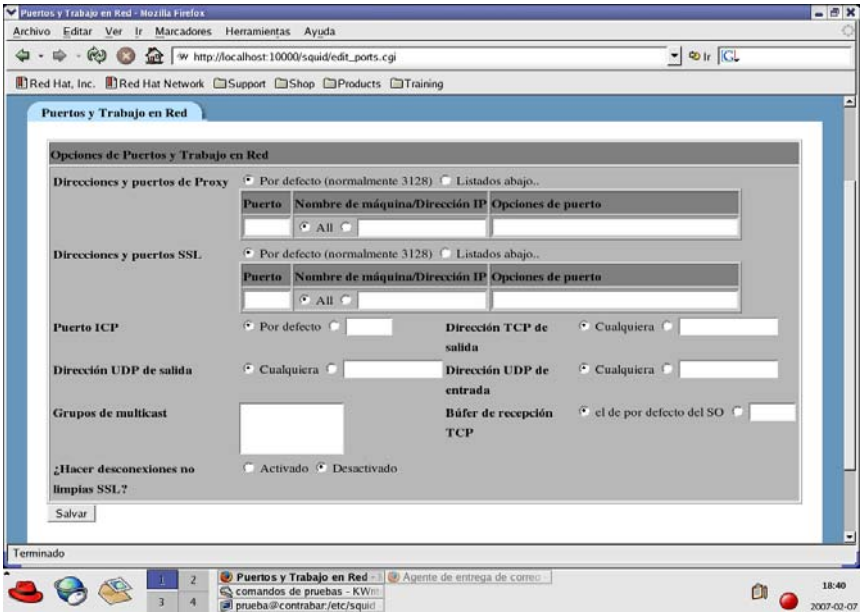
Escogemos el elemento red al lado izquierdo de la pantalla (en columna de listas de control de acceso), Y se abre la lista de equipos asociados Allí podemos eliminar o insertar nuevas direcciones IP, para suspender o permitir el uso de internet. De igual forma se puede escoger la lista de sitios negados modificarlos. Luego se debe proceder a reiniciar el servicio de Proxy (squid) desde una ventana de comandos Linux,

Configuramos en opciones varias :



Gráfica 7. Squid opciones varias.

En Puertos y Trabajos en Red, terminamos de configurar



Gráfica 8. Squid opciones de puertos y trabjo en red.

3.1.5. Servicio SAMBA. Historia: Samba es una suite de aplicaciones Unix que habla el protocolo SMB (Server Message Block). Muchos sistemas operativos, incluídos Windows y OS/2, usan SMB para operaciones de red cliente-servidor. Mediante el soporte de este protocolo, Samba permite a los servidores Unix entrar en acción, comunicando con el mismo protocolo de red que los productos de Microsoft Windows. De este modo, una máquina Unix con Samba puede enmascararse como servidor en tu red Microsoft y ofrecer los siguientes servicios:

- Compartir uno o más sistemas de archivos.
- Autentificar clientes logeándose contra un dominio Windows.
- Proporcionar o asistir con un servidor de resolución de nombres WINS.
- Compartir impresoras, instaladas tanto en el servidor como en los clientes.
- Ayudar a los clientes, con visualizador de Clientes de Red.

Samba es la idea de Andrew Tridgell, quien actualmente lidera el equipo de desarrollo de Samba development desde su casa de Canberra, Australia. El proyecto nació en 1991 cuando Andrew creó un programa servidor de ficheros para su red local, que soportaba un raro protocolo DEC de Digital Pathworks. Aunque él no lo supo en ese momento, aquel protocolo más tarde se convertiría en SMB. Unos cuantos años después, él lo expandió como su servidor SMB particular y comenzó a distribuirlo como producto por Internet bajo el nombre de servidor SMB.

Hoy, la suite Samba implica a un par de demonios que proporcionan recursos compartidos a clientes SMB sobre la red (las comparticiones son denominadas a veces también como servicios). Estos demonios son:

Smbd

Un demonio que permite compartición de archivos e impresoras sobre una red SMB y proporciona autentificación y autorización de acceso para clientes SMB.

El demonio smbd es responsable de manejar los recursos compartidos entre la máquina servidora Samba y sus clientes. Proporciona servicios de archivos, impresión y visualización a los clientes SMB a través de una o más redes. smdb controla todas las notificaciones entre el servidor Samba y los clientes de red. En adición, es responsable de la autentificación de usuarios, bloqueo de recursos y la compartición de datos a través del protocolo SMB.

nmbd

Un demonio que busca a través del Windows Internet Name Service (WINS), y ayuda mediante un visualizador.

El demonio nmbd es un sencillo servidor de nombres que imita la funcionalidad de los servidores WINS y de resolución de nombres NetBIOS. Este demonino está a la escucha de peticiones para el servidor de nombres y proporciona la información apropiada cuando se le llama. También proporciona listas de visualización del Entorno de Red y participa en las elecciones de los visualizadores.

Samba se encuentra actualmente mantenido y es ampliado por un grupo de voluntarios bajo la supervisión activa de Andrew Tridgell. Al igual que el sistema operativo Linux, Samba es considerado por sus autores Open Source software (OSS), y es distribuido bajo la the GNU General Public License (GPL). Desde su concepción, el desarrollo de Samba ha sido patrocinado en parte por la Australian National University, donde Andrew Tridgell hizo su doctorado. En adición, algunas partes del desarrollo han sido patrocinadas por distribuidores independientes como Whistle and SGI. Es algo verdaderamente testimonial el que entidades tanto comerciales como no comerciales estén dispuestas a gastar dinero para dar soporte a un esfuerzo Open Source.

La distribución de Samba también está acompañada por un pequeño grupo de herramientas tipo línea de comandos Unix:

smbpasswd

Este es un programa que permite a un administrador cambiar las passwords encriptadas usadas por Samba.

smbstatus

Este es un programa para reportar las conexiones de red actuales hacia los recursos compartidos por el servidor Samba.

testparm

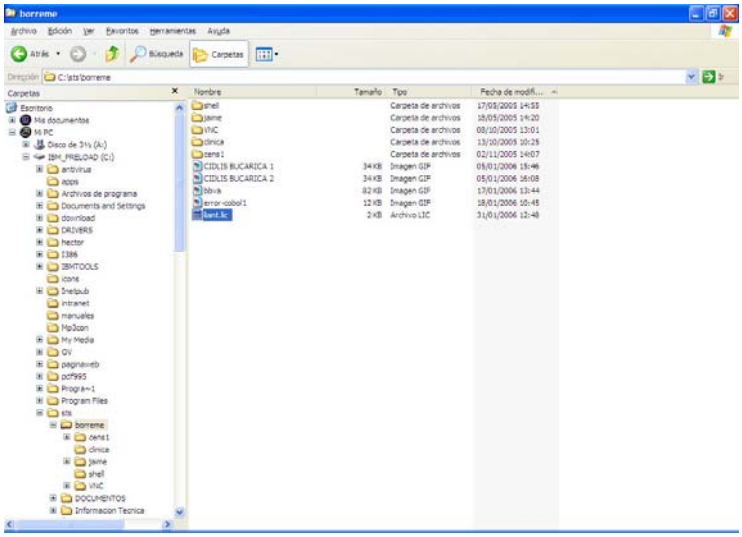
Este es un simple programa para validar el fichero de configuración de Samba.

Microsoft también ha contribuido materialmente poniendo a disposición su definición de SMB y del Internet-savvy Common Internet File System (CIFS), como Public Request for Comments (RFC), y otros documentos estandar. El protocolo CIFS es el nuevo nombre de las futuras versiones del protocolo SMB que serán usadas en los productos Windows -los dos términos pueden ser usados aleatoriamente en éste libro-. De hecho, verás el protocolo escrito como "SMB/CIFS".

3.1.5.1. Funcionalidades de Samba. Samba puede ayudar a las máquinas Windows y LINUX a cohabitar en la misma red. Sin embargo, existen algunas razones específicas por las cuales instalamos Samba en la intranet de la Contraloría Municipal de Barrancabermeja.

- No tener que adquirir un servidor Windows y adquirir de licencias para obtener las funcionalidades que este sistema proporciona.
- Lograr proporcionar un área común para datos o directorios de usuarios en orden a realizar una transición desde un servidor hacia un Unix, o viceversa.
- Dar seguridad informática a los aplicativos utilizados por los funcionarios de la contraloría.
- Poder compartir impresoras a entre usuarios Windows y linux.
- Poder acceder a carpetas de Windows desde un servidor linux.

En la máquina Windows se debe compartir la carpeta que se desea prestar. En este caso la carpeta “pública”.



Desde la máquina linux se montó esa carpeta del PC compartida (con la dirección IP 192.168.1.123), con el comando de samba smbmount, pero se debe crear previamente la carpeta donde se va a montar, por ejemplo llamada windows.

mkdir /windows
smbmount //192.168.1.240/publica /windows
//192.168.1.123/publica

Luego muestra la carpeta montada ya se ve como una partición de la máquina y los usuarios la pueden desde /windows.

3.1.5.2. Uso de samba para compartir archivos e impresoras. Permite colocar a disposición de la red windows, un sistema de archivos para almacenar información, con la seguridad de los entornos conocidos por ustedes. También podemos compartir otros recursos como impresoras.

Ahora si entraremos en la configuración del SAMBA para permitir que desde el entorno de red de windows se acceden carpetas de usuarios de la máquina Linux que tiene mayor capacidad de almacenamiento.

Se tiene una serie de usuarios linux, que son los mismos usuarios de correo.

Podemos proceder a manipular el archivo de configuración o por los intefaz gráficos de webmin.

Aquí se modificó directamente el archivo /etc/samba/smb.conf (Ver anexo).

Luego se puede reiniciar el servicio samba :

service smb restart

Se procede a crear usuarios samba. Es aconsejable que el nombre del usuario samba sea el mismo del usuario linux, así como el password (así se evita que se le pregunte al picar sobre la carpeta compartida, una vez se ha logeado en XP con el usuario samba creado (es el mismo usuario creado en los perfiles de XP).

Se usa el comando linux smbadduser nombreuserlinux:nombreuserwindows
-- ENTER password for
New SMB password:
Retype new SMB password:

Si se desea cambiar el password de un usuario samba , se puede recurrir al comando linux : smbpasswd

Ya se puede proceder a acceder estas carpetas compartidas desde el entorno de red de windows XP, entrando primero con el perfil de XP y el nombre del usuario samba creado.

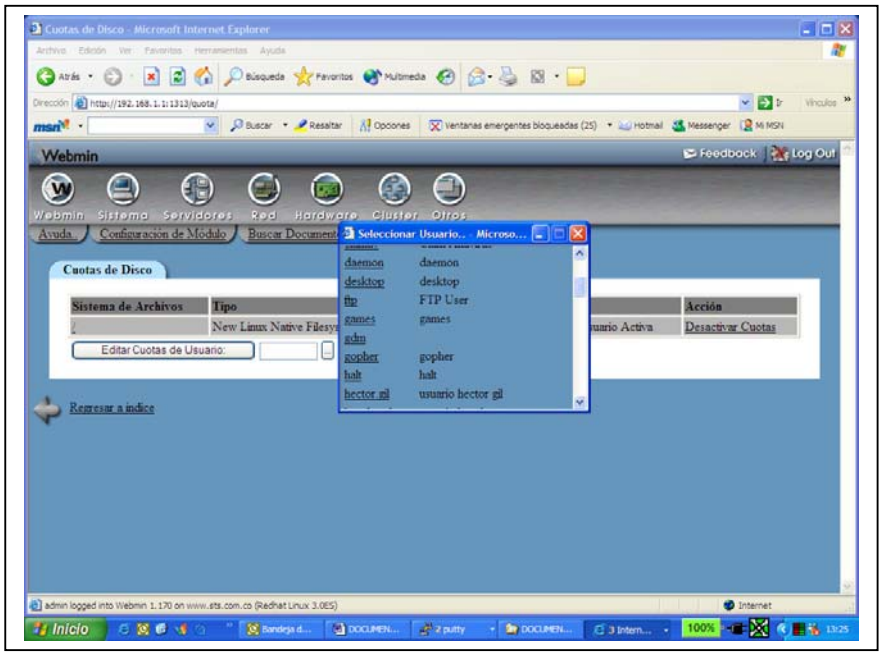
Los archivos involucrados, residen en la carpeta /etc/samba/

3.1.5.3. Manejo de Cuotas De Disco. Cuando un usuario tiene asignado un directorio dentro de una partición de disco, este puede escribir en ella hasta que la partición se llene, perjudicando a los demás usuarios y procesos que residen en esa partición. Es conveniente en algunas circunstancias asignar límites de consumo de espacio en disco dentro de una partición a un usuario y esto es llamado cuota de disco. Para el caso del correo electrónico, nos permite controlar el espacio que un usuario tiene para almacenar sus correos (tamaño de buzón), o el tamaño de los archivos manipulados por el interfaz web.

3.1.5.3.1. Asignación de Cuotas A Los Usuarios. A cada usuario se le asigno una cuota de 10 Mb.

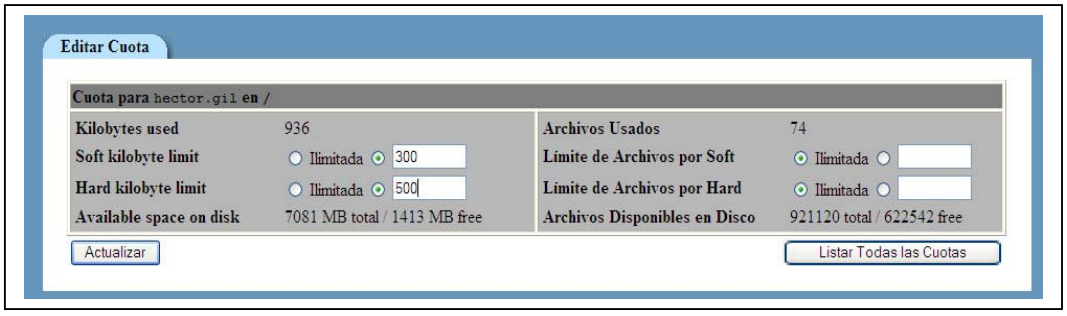
Se puede hace por línea de comandos, o por medio del interfaz web webmin.

Si se va a asignar la cuota por medio del interfaz gráfico webmin, se hace así: Entrando por el icono de sistema, de la parte superior, podemos observar un icono para manejo de cuotas del sistema operativo:



Gráfica 9. Cuotas a usuarios.

Como pueden existir varias particiones en las cuales se defina cuota, aparece una lista de ellas y se debe escoger en cual partición se requiere editar estos valores y aparece:



Gráfica 10. Configuración de Cuotas a usuarios.

Y se cambian los valores deseados y se indica actualizar.

4. CONFIGURACION DE SERVIDOR (SEGURIDAD)

4.1. ASPECTOS DE DISEÑO DE LA RED

Inicialmente se analizaron las conexiones del cableado estructurado y de los computadores e impresoras de los diferentes usuarios de la red hallada en la Contraloría, los cuales trabajaban en sistema monousuario, ya que anteriormente no se encontraba configurada la intranet.

La entidad actualmente cuenta con una infraestructura de red de 30 puntos, dos switch 3com de 16 puertos cada uno, un servidor HP Proliant ML350 y una conexión a Internet banda ancha de 456 Kbps, saliendo por un modem DSL.

En vista de eso se procedió a configurar la red de la siguiente forma:

Funcionario	División	Dirección IP	Punto de Red	Login
Jairo Edwin Garzón	Responsabilidad Fiscal y Jurisdicción Coactiva	192.168.1.101	1	jairogarzon
Juan de Dios Sepúlveda	Fiscalización	192.168.1.102	2	juansepulveda
Luis Aurelio Sánchez	Fiscalización	192.168.1.103	3	aureliosanchez
Cristina Ruiz Jazbón	Contable	192.168.1.104	4	cristinaruiz1
Cristina Ruiz Jazbón	Contable	192.168.1.105	5	cristinaruiz2
William González	Fiscalización	192.168.1.106	6	williamgonzalez
Lavoissier Royero Mancera	Fiscalización	192.168.1.107	7	lavoissierroyero
Jose Antonio Riaño	Fiscalización	192.168.1.108	8	antonioriano
Reinaldo Gómez	Fiscalización	192.168.1.109	9	reinaldogomez
Jaime Enrique Bustos	Fiscalización	192.168.1.110	10	jaimebustos
Edgar Ortega	Fiscalización	192.168.1.111	11	edgarortega
Alvaro Carrascal	Fiscalización	192.168.1.112	12	alvarocarrascal
Diana Jaramillo	Fiscalización	192.168.1.113	13	dianajaramillo
Luis Alberto Rivero	Fiscalización	192.168.1.114	14	albertorivero
Victor Hugo Florez	Fiscalización	192.168.1.115	15	victorflorez
Hober Torres	Compras	192.168.1.116	16	hobertorres
Arnulfo Rodríguez	Responsabilidad Fiscal y Jurisdicción Coactiva	192.168.1.118	18	arnulforodriguez
Walter Rangel	Responsabilidad Fiscal y Jurisdicción Coactiva	192.168.1.119	19	walterrangel
Sandra Milena Ibáñez	Secretaria de Despacho	192.168.1.120	20	sandraibanez

Maribel López	Secretaria General	192.168.1.121	21	maribellopez
Carlos Oscar Rodríguez	Contralor	192.168.1.122	22	carlosrodriguez
Mario Capdevilla	Tesorería	192.168.1.123	23	mariocapdevilla
Servidor	Sistemas	192.168.1.110	24	mariocapdevilla

Mascara de Subred	**255.255.255.0**
Puerta de Enlace	**192.168.1.1**
DNS Preferido	**200.21.200.2**
DNS Alternativo	**200.21.200.79**
Grupo de Trabajo	**CONTRALORIA**

Tabla 3. Parámetros de Diseño de la red.

4.2. CRITERIOS DE DECISIÓN DEL SISTEMA OPERATIVO

Dado que Linux es un sistema operativo libre, es decir, de código abierto, que da la libertad para que cualquier programador acceda a su código fuente para modificar aquellas características que estime oportuna para hacer funcionar alguna aplicación programada por él.

Además implica mayor seguridad ya que cuando se detecta algún fallo en cualquier servicio se una comunidad de programadores establecida a lo largo del mundo busca o desarrolla los respectivos parches de seguridad

También se tuvo en cuenta la fortaleza y la robustez que le da el sistema operativo Linux Fedora Core 4 a los servidores, permitiendo no solo la caída del mismo sino la facilidad para subir y bajar servicios de red de acuerdo a como se necesiten.

4.3. INSTALACIÓN DE LINUX FEDORA CORE 4

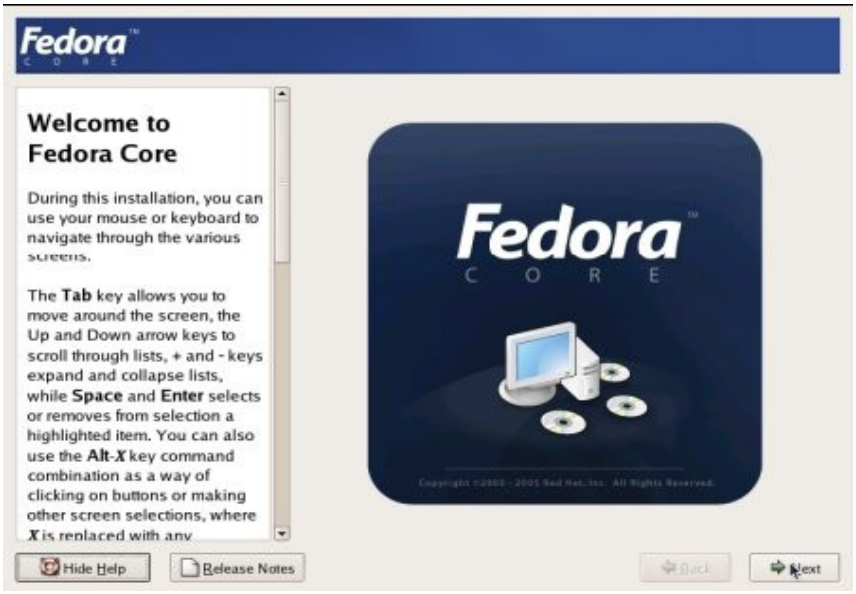
Al instalar Fedora Core 4 se inició el equipo por medio de la unidad óptica. Colocando el primer CD de instalación (Es importante recordar que Fedora Core 4 consta de 4 cd's de instalación más uno de rescate). Al arrancar el equipo aparece la siguiente ventana, en la cual solo se presiona la tecla Enter.

Gráfica 11. Instalación de Linux Fedora.

Inmediatamente después aparece una caja de diálogo preguntando si se desean verificar los cd's o no. Si estos no se desean verificar se debe escoger la opción SKIP.

Gráfica 12. Instalación de Linux Fedora. Vefiricación.

Normalmente el programa de instalación reconoce el hardware del equipo como monitor, teclado, mouse, tarjeta de video, etc.; una vez realizado este proceso aparece la siguiente ventana, sobre la cual se debe presionar la opción NEXT.

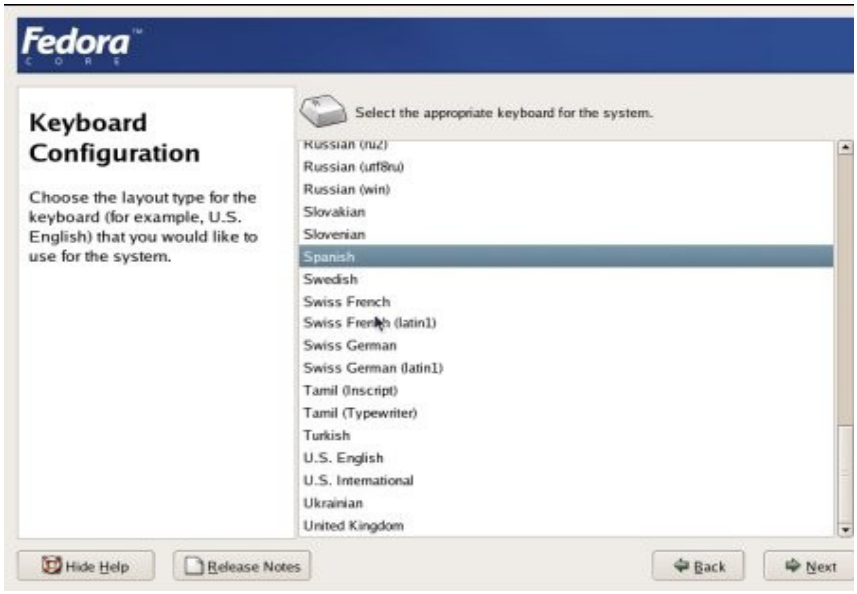


Gráfica 13. Continuación Instalación de Linux Fedora.

A continuación el programa de instalación pregunta el idioma, en este caso se seleccionó Español.

Gráfica 14. Instalación de Linux Fedora. Idioma.

Luego solicita el idioma de la configuración regional del teclado.



Gráfica 15. Instalación de Linux Fedora. Teclado.

Luego aparece una pantalla que pregunta el tipo de instalación que se desea realizar. En este caso se tomó la opción Personalizada.

Gráfica 16. Instalación de Linux Fedora. Tipo instalación

Después de esto el programa de instalación pregunta por las aplicaciones que se desean instalar.

Después en el programa solicita las particiones que se van a realizar. En este caso se tomaron los siguientes espacios en disco duro para cada partición:

Luego el programa de instalación solicita la zona horaria en la cual se va a trabajar

Gráfica 17. Instalación de Linux Fedora. Zona horaria

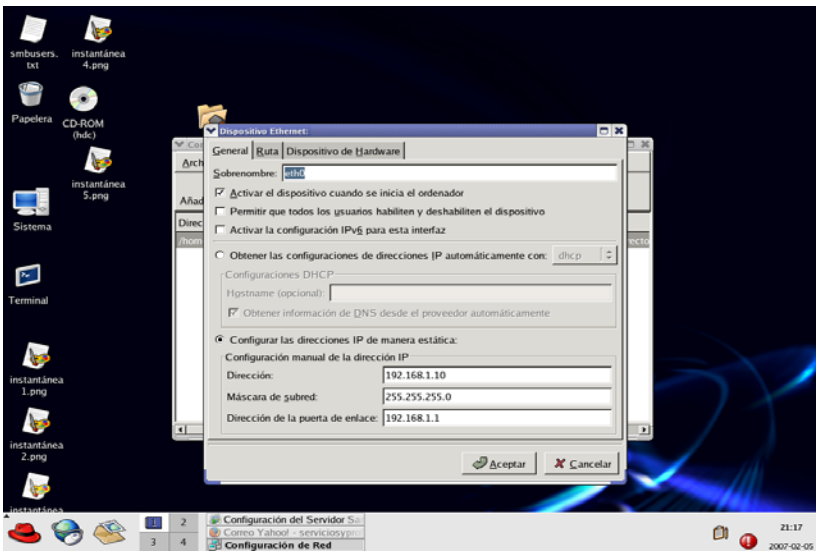
A continuación solicita las claves para el superusuario ROOT.

Gráfica 18. Instalación de Linux Fedora. Contraseña.

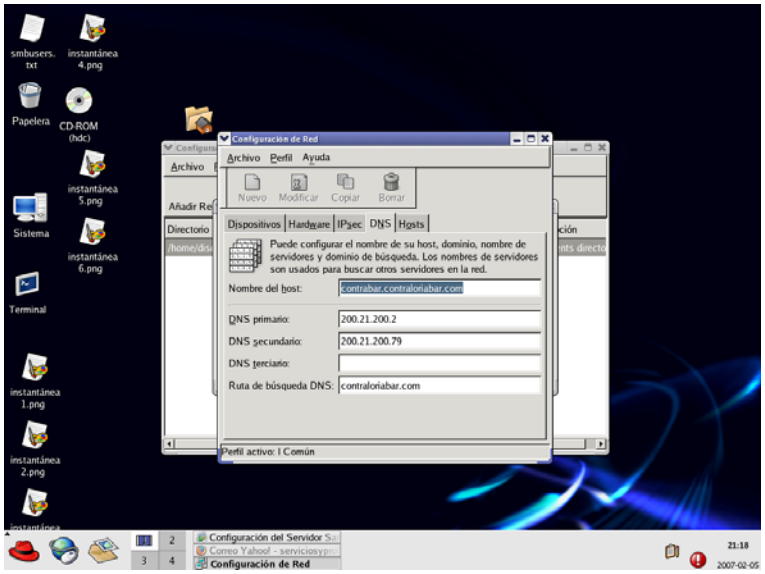
Hasta este punto la configuración de la instalación se encuentra lista y a partir de este momento solicitará cada uno de los cd's en su orden respectivo orden.

4.4. CONFIGURACIÓN DEL SERVIDOR DE LA RED

Para configurar la dirección lógica del servidor se entró a la aplicación de red que se encuentra en el menú inicio del Linux Fedora Core 4. Como se muestra en las siguientes figuras.



Gráfica 19. Configuración de servidor de red. Dirección Lógica.



Gráfica 20. Configuración de servidor de red. DNS

Luego se entró a la utilidad webmin para configurar los usuarios de acuerdo a la tabla de usuarios mostrada en la sección 4.1., teniendo cuidado de asignar a cada usuario de acuerdo a la división a la cual pertenecen y al grupo Contraloría.

4.5. SEGURIDAD INFORMÁTICA

Con el ánimo de tener las herramientas necesarias para analizar la situación real por la que atraviesa la Contraloría Municipal de Barrancabermeja en materia de Seguridad Informática; realizamos una encuesta al personal de la entidad; luego analizamos los resultados y por último se presenta una alternativa de solución.

4.5.1. Encuesta de Seguridad Informática

4.5.1.1. Metodología de la Encuesta: Se efectuó un estudio de carácter exploratorio baja la forma de encuesta escrita con un muestreo de los funcionarios de la contraloría municipal.

Se entregaron 10 cuestionarios con 7 preguntas; fueron contestados 10.

El objetivo del control en seguridad informática es proteger los recursos informáticos de la organización, tales como: la información, el hardware (computadores) y el software (windows, word, excel etc). A través de la adopción de las medidas adecuadas, la seguridad informática ayuda a la organización cumplir sus objetivos, protegiendo sus recursos financieros, sus sistemas, su reputación; además de otros bienes tangibles o inmateriales. Desafortunadamente, en ocasiones se ve a la seguridad informática como algo que dificulta la consecución de los propios objetivos de la organización, imponiendo normas y procedimientos rígidos a los usuarios, a los sistemas y a los gestores. Sin embargo debe verse a la seguridad informática, no como un objetivo en sí

mismo, sino como un medio de apoyo a la consecución de los objetivos de la organización.

LUGAR: CONTRALORIA MUNICIPAL BARRANCABERMEJA
FECHA: OCTUBRE 2006
CANTIDAD DE PERSONAS ENCUESTADAS: 10 FUNCIONARIOS.

1. Considera que la Contraloría municipal de Barrancabermeja posee un eficiente manejo de la seguridad informática para salvaguardar la información interna de la entidad.
   Si _____ No _____

2. De que manera usted protege la información MAGNÉTICA (realizada en computador) relevante (importante y vital) que maneja?
   a. Archiva la información solo en diskette Si _____ No _____ NR
   b. Archiva la información en archivadores y en diskette Si _____ No _____ NR
   c. Archiva la información en BACKUP disco duro central (SERVIDOR) de la entidad y protegido con claves secretas? Si _____ No _____ NR

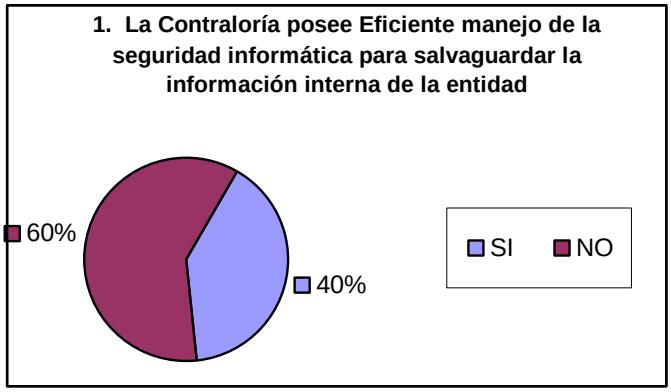
3. Considera que se deben implementar políticas de seguridad de la información Magnética que posee la entidad?
   Si _____ No ____

4. Ha tenido la entidad problemas de seguridad de la información (pérdida o fuga de información)?
   Si _____ No ____ No sabe _____
5. Que tipo de problemas de seguridad han tenido?
   a. Robo de información confidencial.
   b. Virus informáticos.
   c. Abuso del acceso de internet
   d. Robo de equipos de computo.
   e. Otros.

6. De su opinión al respecto. Considera que a futuro la inversión en seguridad informática será?
   a. Menor a la actual.
   b. Mayor a la actual.
   c. No sabe

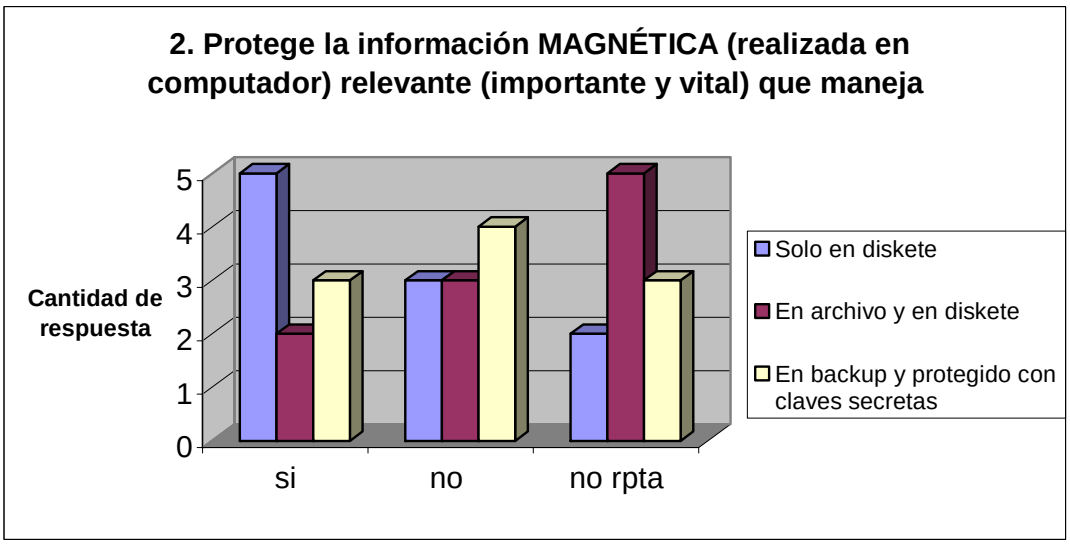
7. Conoce algunas de las siguientes tecnología de Seguridad en Informática?
   a. Control de acceso (password)
   b. Encriptación de archivos
   c. Software Antivirus
   d. Firevalls
   e. Logs servers
   f. Certificados digitales
   g. Redes Virtuales VPN
   h. Sistema de Detección de Intrusos
   i. Ninguna o no sabe.

4.5.2. Análisis de Resultados Encuesta a Funcionarios de La Contraloría Municipal de Barrancabermeja.



Gráfica 21. Respuesta a la Pregunta uno.

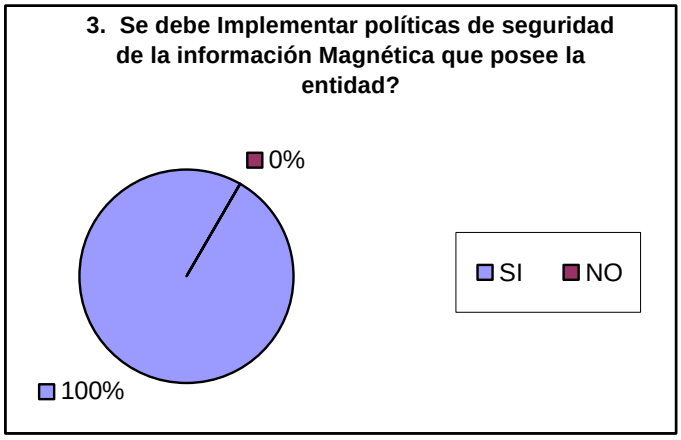
El 60% considera que no es eficiente el manejo de la seguridad informática en la contraloría.



Gráfica 21. Respuesta a la Pregunta dos.

Es muy grave comprobar que el 50% de los encuestados indiquen que solo guardan su información en diskette, siendo muy vulnerable la pérdida de la información almacenada en este medio.

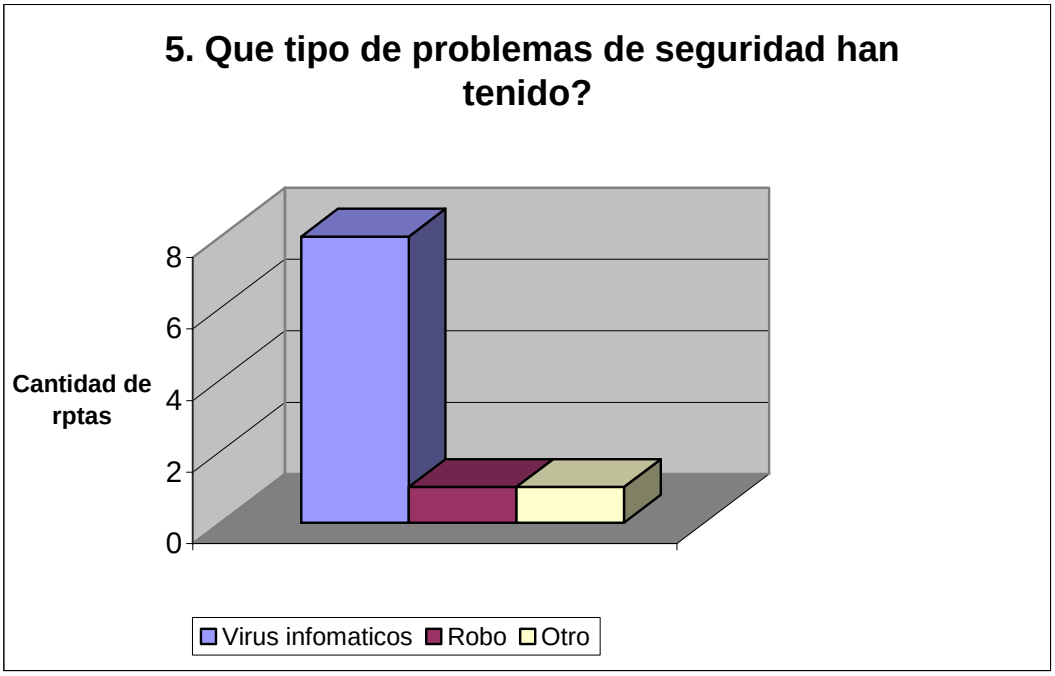
Solo el 30% almacena la información en el servidor y con claves secreta.



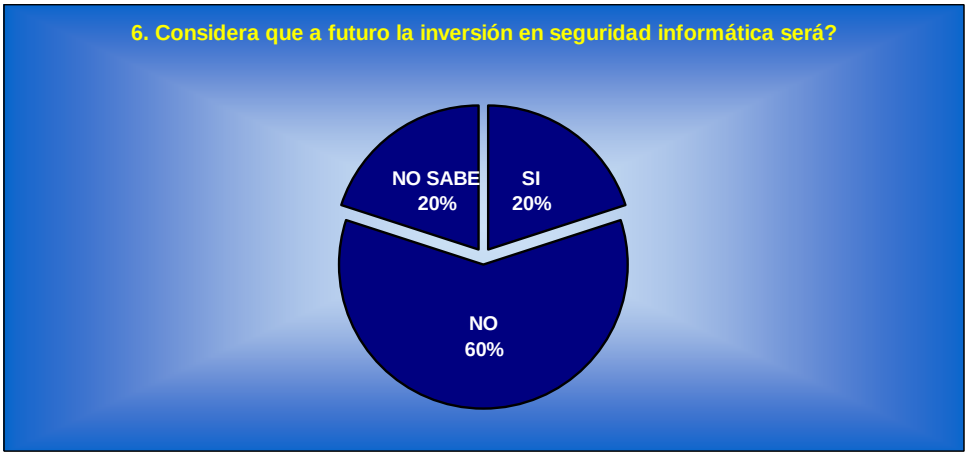
Gráfica 23. Respuesta a la Pregunta tres.

De acuerdo a los resultado de la encuesta, el 100% de los entrevistaos indica que se hace necesario el implementación de políticas de Seguridad Informática.

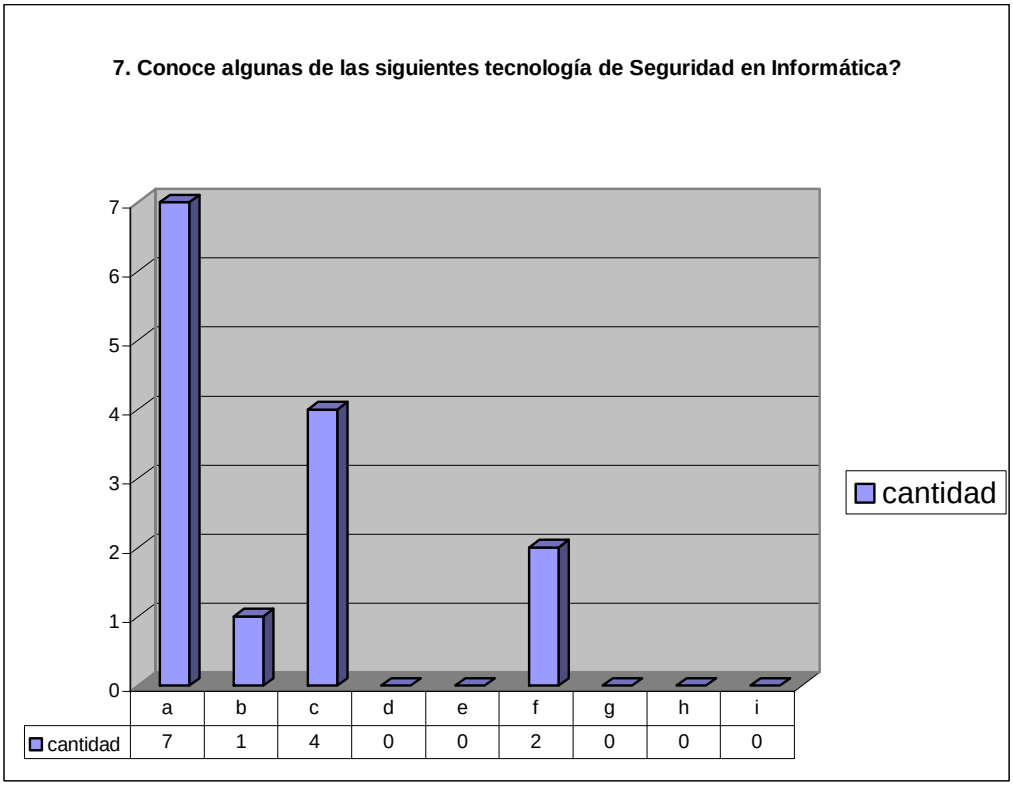
Gráfica 24. Respuesta a la Pregunta cuatro



Gráfica 25. Respuesta a la Pregunta cinco.



Gráfica 26. Respuesta a la Pregunta seis.



Gráfica 27. Respuesta a la Pregunta siete.

Al revisar todos estos datos se puede deducir que la seguridad informática dentro de esta organización se encuentra bastante débil basados en:

- Los usuarios continúan realizando sus copias de seguridad dentro de copias blandas (como disquetes) algunos llevan su información dentro de su memoria USB.
- La información dentro de los equipos de cómputo se encuentra demasiado frágil y poco segura ya que se encuentra vulnerable al ataque que se presenten ataque de virus como múltiples veces se ha presentado.
- Los equipos de cómputo solo se encuentran protegidos por una clave de seguridad de entrada al sistema operativo (en su gran mayoría Windows XP Profesional).
- Además todos los usuarios encuestados coinciden que se debe implementar una política de seguridad informática dentro de la organización.
- La conexión a Internet está haciendo vulnerable la información no solo con ataque de virus informático sino con entradas no permitidas del sistema, permitiendo por este medio la fuga de información.
- El cambio de personal supernumerario dentro de la entidad hace que existan usuarios transitorios poniendo en riesgo la seguridad informática de la entidad.

- La clase de información que maneja la entidad es información confidencial (a pesar que cuando se da un concepto, un informe o un fallo se libera) y requiere de un cuidado especial, aparte de eso existe información delicada como lo es la información contable que se encuentra almacenada dentro de un software contable (protegido por claves de usuarios).

De acuerdo a los grandes analista a nivel mundial cada vez más las empresas van adquiriendo mayor tecnología y deberá incrementarse por ende la inversión en seguridad informática.

4.6. ADMINISTRACIÓN DE SEGURIDAD

El salto tecnológico desde los sistemas de escritorio aislados, hasta los sistemas actuales integrados en redes locales e Internet, ha traído una nueva dificultad a las tareas habituales del administrador: el control de la seguridad de los sistemas.

La seguridad es un campo complejo, en el cual se mezclan técnicas de análisis, con otras de detección o de prevención de los posibles ataques, como el análisis de factores "psicológicos", como el comportamiento de los usuarios del sistema o las posibles intenciones de los atacantes.

Los ataques pueden provenir de muchas fuentes y afectar desde a una aplicación o servicio, hasta a algún usuario, o a todos, o al sistema informático entero.

Los ataques pueden cambiar el comportamiento de los sistemas, incluso hacerlos caer (inutilizarlos), o dar una falsa impresión de seguridad, que puede ser difícilmente detectable. Existen ataques de autentificación (obtener acceso por parte de programas o usuarios previamente no habilitados), escuchas (re-dirigir o pinchar los canales de comunicación y los datos que circulan), o sustitución (sustituir programas, máquinas o usuarios por otros, sin que se noten los cambios).

4.6.1. Tipos y métodos de los ataques. La seguridad computacional en administración puede ser entendida como el proceso que ha de permitir al administrador del sistema prevenir y detectar usos no autorizados de éste. Las medidas de prevención ayudan a parar los intentos de usuarios no autorizados (los conocidos como intrusos), para acceder a cualquier parte del sistema. La detección ayuda a descubrir cuándo se produjeron estos intentos o, en el caso de llevarse a cabo, establecer las barreras para que no se repitan y poder recuperar el sistema si éste ha sido quebrantado.

Normalmente nos referimos a hacker, como aquella persona con grandes conocimientos en informática, más o menos apasionada por los temas de programación y seguridad informática y que, normalmente sin finalidad malévola, utiliza sus conocimientos para protegerse a sí mismo, introducirse en redes para demostrar sus fallos de seguridad, y, en algunos casos, como reconocimiento de sus habilidades.

También existen los crackers. Son aquellos que utilizan sus habilidades para corromper (o destrozar) sistemas, ya sea sólo por fama propia, por motivos económicos, por ganas de causar daño o simplemente por molestar; por motivos de espionaje tecnológico, actos de ciberterrorismo, etc. Asimismo, se habla de hacking o cracking, cuando nos referimos a técnicas de estudio, detección y protección de la seguridad, o por el contrario, de técnicas destinadas a causar daño rompiendo la seguridad de los sistemas.

Desafortunadamente, es bastante más fácil de lo que parece obtener acceso a un sistema, (ya sea desprotegido o parcialmente seguro). Los intrusos descubren permanentemente nuevas vulnerabilidades (llamadas 'agujeros', o exploits), que les permiten introducirse en las diferentes capas de software. La complejidad cada vez mayor del software (y del hardware), hace que aumente la dificultad para idear de manera razonable la seguridad de los sistemas informáticos. El uso habitual de los sistemas GNU/Linux en red, ya sea la propia Internet o en redes propias con tecnología TCP/IP como las intranets, nos lleva a exponer nuestros sistemas a ser víctimas de ataques de seguridad.

Nuestro objetivo debe ser el ser capaz de establecer en el sistema de la Contraloría, unas políticas de seguridad que nos permitan prevenir, identificar y reaccionar ante los posibles ataques. Por lo tanto, no h ay que descuidar ninguna de las políticas implementadas y hay que mantenerlas al día, así como nuestros conocimientos del tema.

Los posibles ataques son una amenaza constante a nuestros sistemas y pueden comprometer su funcionamiento, así como los datos que manejamos; ante todo lo cual, siempre tenemos que definir una cierta política de requerimientos de seguridad sobre nuestros sistemas y datos.

Los métodos utilizados y las técnicas precisas pueden variar mucho, y nos obligan, como administradores, a estar en contacto permanente con el campo de la seguridad para conocer a qué nos podemos enfrentar diariamente.

Respecto a dónde se produzca el ataque, hemos de tener claro qué puede hacerse o cual será el

objetivo de los métodos:

- Hardware: a este respecto, la amenaza está directamente sobre la accesibilidad, ¿qué podrá hacer alguien que tenga acceso al hardware? En este caso normalmente necesitaremos medidas "físicas", como controles de seguridad para acceder a los locales donde estén las máquinas para evitar problemas de robo o rotura del equipo con el fin de eliminar su servicio.

- Software: si la accesibilidad se ve comprometida en un ataque, puede haber borrado o inutilización de programas, denegando el acceso. En caso de confidencialidad, puede provocar copias no autorizadas de software. En integridad podría alterarse el funcionamiento por defecto del programa.

- Datos: ya sean estructurados, como en los servicios de base de datos, o simples archivos. Mediante ataques que amenacen la accesibilidad, pueden ser destruidos o eliminados, denegando así el acceso a los mismos.

- Canal de comunicación (en la red, por ejemplo): para los métodos que afecten a la accesibilidad, nos puede provocar destrucción o eliminación de mensajes, e impedir el acceso a la red. En confidencialidad, lectura y observación del tráfico de mensajes, desde o hacia la máquina. Y respecto a la integridad, cualquier modificación, retardo, reordenación, duplicación o falsificación de los mensajes entrantes y/o salientes.

4.6.2 Técnicas utilizadas en los ataques. Los métodos utilizados son múltiples y pueden depender de un elemento (hardware o software) o de la versión de éste. Por lo tanto, hay que mantener actualizado el software para las correcciones de seguridad que vayan apareciendo, y seguir las indicaciones del fabricante o distribuidor para proteger el elemento. A pesar de ello, normalmente siempre hay técnicas o métodos de "moda", del momento actual, algunas breves indicaciones de estas técnicas de ataque (de hoy en día) son:

- Bug exploits: o explotación de errores o agujeros, ya sea de un hardware, software, servicio, protocolo o del propio sistema operativo (por ejemplo, en el kernel), y normalmente de alguna de las versiones de éstos en concreto.

- Virus: programa normalmente anexo a otros y que utiliza mecanismos de autocopia y transmisión. Son habituales los virus anexados a programas ejecutables, a mensajes de correo electrónico, o incorporados en documentos o programas que permiten algún lenguaje de macros (no verificado). Son realmente la mayor plaga de seguridad d de hoy en día,

además lo demuestan las estadísticas analizadas de la evaluación a los funcionarios de la Contraloría.

Los sistemas Linux están protegidos casi totalmente contra estos mecanismos por varias razones: en los programas ejecutables, tienen un acceso muy limitado al sistema, en particular a la cuenta del usuario. Con excepción del usuario root, con el cual hay que tener mucho cuidado con lo que éste ejecuta. El correo no suele utilizar lenguajes de macros no verificados (como en el caso de Outlook y Visual Basic Script en Windows, que es un agujero de entrada de virus), y en el caso de los documentos, estamos en condiciones parecidas, ya que no soportan lenguajes de macros no verificados (como el VBA en Microsoft Office).

En todo caso, habrá que prestar atención a lo que pueda pasar en un futuro, ya que podrían surgir algunos virus específicos para Linux aprovechando algunos bugs o exploits. Un punto que sí que hay que tener en cuenta es el de los sistemas de correo, ya que aunque el sistema linux no genera virus, pero si lo pueden transmitir.

- Worm (o 'gusano'): normalmente se trata de un tipo de programas que aprovechan algún agujero del sistema para realizar ejecuciones de código sin permiso. Suelen ser utilizados para aprovechar recursos de la máquina, como el uso de CPU, bien cuando se detecta que el sistema no funciona o no está en uso, o bien si son malintencionados, con el objetivo de robar recursos o bien utilizarlos para parar o bloquear el sistema. También suelen Utilizar técnicas de transmisión y copia.

- Trojan Horse (o 'troyanos'): programas útiles que incorporan alguna funcionalidad, pero ocultan otras que son las utilizadas para obtener información del sistema o comprometerlo. Un caso particular puede ser el de los códigos de tipo móvil en aplicaciones web, como los Java, JavaScript o ActiveX; éstos normalmente piden su consentimiento para ejecutarse (ActiveX en Windows), o tienen modelos limitados de lo que pueden hacer (Java, JavaScript). Pero como todo software, también tienen agujeros y son un método ideal para transmitir troyanos.

- Back Door (o 'puerta trasera'): método de acceso a un programa escondido que puede utilizarse con fines de otorgar acceso al sistema o a los datos manejados sin que lo conozcamos. Otros efectos pueden ser cambiar la configuración del sistema, o permitir introducir virus. El mecanismo usado puede ser desde venir incluidos en algún software común, o bien en un troyano.

- Bombas lógicas: programa incrustado en otro, que comprueba que se den algunas condiciones (temporales, acciones del usuario, etc.), para activarse y emprender acciones no autorizadas.

- Keyloggers: programa especial que se dedica a secuestrar las interacciones con el teclado del usuario. Pueden ser programas individuales o bien troyanos incorporados en otros programas. Normalmente, necesitarían introducirse en un sistema abierto al que se dispusiese de acceso. La idea es captar cualquier introducción de teclas, de manera que se capturen contraseñas, interacción con aplicaciones, sitios visitados por la red, formularios rellenados, etc.

- Scanner (escaneo de puertos): más que un ataque, sería un paso previo, que consistiría en la recolección de posibles objetivos. Básicamente, consiste en utilizar herramientas que permitan examinar la red en busca de máquinas con puertos abiertos, sean TCP, UDP u otros protocolos, los cuales indican presencia de algunos servicios. Por ejemplo, escanear máquinas buscando el puerto 80 TCP, indica la presencia de servidores Web, de los cuales podemos obtener información sobre el servidor y la versión que utilizan para aprovecharnos de vulnerabilidades conocidas.

- Sniffers ('husmeadores'): permiten la captura de paquetes que circulan por una red. Con las herramientas adecuadas podemos analizar comportamientos de máquinas: cuáles son servidores, clientes, qué protocolos se utilizan y en muchos casos obtener contraseñas de servicios no seguros. Tanto los sniffers (como los scanners) no son necesariamente una herramienta de ataque, ya que también pueden servir para analizar nuestras redes y detectar fallos, o simplemente analizar nuestro tráfico. Normalmente, tanto las técnicas de scanners como las de sniffers suelen utilizarse por parte de un intruso con el objetivo de encontrar las vulnerabilidades del sistema, ya sea para conocer datos de un sistema desconocido (scanners), o bien para analizar la interacción interna (sniffer).

- Hijacking (o 'secuestro'): son técnicas que intentan colocar una máquina de manera que intercepte o reproduzca el funcionamiento de algún servicio en otra máquina que ha pinchado la comunicación. Suelen ser habituales los casos para correo electrónico, transferencia de ficheros o web. Por ejemplo, en el caso web, se puede capturar una sesión y reproducir lo que el usuario está haciendo, páginas visitadas, interacción con formularios, etc.

- Buffer overflows : técnica bastante compleja que aprovecha errores de programación en las aplicaciones. La idea básica es aprovechar desbordamientos (overflows) en buffers de la aplicación, ya sean colas, arrays, etc. Si no se controlan los límites, un programa atacante puede generar un mensaje o dato más grande de lo esperado y provocar fallos.

- Denial of Service (‘ataque DoS’): este tipo de ataque provoca que la máquina caiga o que se sobrecarguen uno o más servicios, de manera que no sean utilizables. Otra técnica es la DDoS (Distributed DoS), que se basa en utilizar un conjunto de máquinas distribuidas para que produzcan el ataque o sobrecarga de servicio.

- spoofing: las técnicas de spoofing engloban varios métodos (normalmente muy complejos) de falsificar tanto la información, como los participantes en una transmisión (origen y/o destino). Existen spoofing como los de:

    1. IP spoofing, falsificación de una máquina, permitiendo que genere trafico falso, o escuche tráfico que iba dirigido a otra máquina, combinado con otros ataques puede saltarse incluso protección de firewalls.

    2. ARP spoofing, técnica compleja (utiliza un DDoS), que intenta falsificar las direcciones de fuentes y destinatarios en una red, mediante ataques de las cachés de ARP, que poseen las máquinas, de manera que se sustituyan las direcciones reales por otras en varios puntos de una red.
    3. E-mail, es quizás el más sencillo. Se trata de generar contenidos de correos falsos, tanto por el contenido como por la dirección de origen. En este tipo son bastante utilizadas las técnicas de lo que se denomina ingeniería social, que básicamente intenta engañar de una forma razonable al usuario.

4.6.3 Contramedidas. Algunos tipos de medidas que se deben tomar en los campos de prevención y detección de intrusos.

- Password cracking: en ataques de fuerza bruta para romper las contraseñas, suele ser habitual intentar obtener acceso por logins de forma repetida. Para prevenir este tipo de ataques, hay que reforzar la política de contraseñas, pidiendo una longitud mínima y cambios de contraseña periódicos. Una buena elección suelen ser contraseñas de entre 6 a 8 caracteres como mínimo, con contenido de caracteres alfabéticos, numéricos y algún carácter especial.

- Bug exploits: evitar disponer de programas que no se utilicen, sean antiguos, o no se actualicen (por estar obsoletos). Aplicar los últimos parches y actualizaciones que estén disponibles, tanto para las aplicaciones, como para el sistema operativo. Probar herramientas que detecten vulnerabilidades. Mantenerse al día de las vulnerabilidades que se vayan descubriendo.

- Virus: utilizar mecanismos o programas antivirus, sistemas de filtrado de mensajes sospechosos, evitar la ejecución de sistemas de macros (que no se puedan verificar). No hay que minimizar los posibles efectos de los virus, cada día se perfeccionan más.

- Worm (o gusano): controlar el uso de nuestras máquinas o usuarios en horas no previstas, y el control del tráfico de salida y/o entrada.

- Trojan horse (o caballos de Troya, o troyanos): verificar la integridad de los programas periódicamente, mediante mecanismos de suma o firmas. Detección de tráfico anómalo de salida o entrada al sistema. Utilizar firewalls para bloquear tráfico sospechoso. Una versión bastante peligrosa de los troyanos la forman los rootkits (comentados más adelante), que realizan más de una función gracias a un conjunto variado de herramientas.

- Back door (o trap door, puerta trasera): hay que obtener de los proveedores o vendedores del software la certificación de que éste no contiene ningún tipo de backdoor escondido no documentado, y por supuesto aceptar el software proveniente sólo de sitios que ofrezcan garantías. Otro problema puede consistir en la alteración del software a posteriori. En este caso pueden ser también útiles los sistemas de firmas o sumas para crear códigos sobre software ya instalado y controlar que no se produzcan cambios en software vital.

- Bombas lógicas : en este caso suelen ocultarse tras activaciones por tiempo o por acciones del usuario. Podemos verificar que no existan en el sistema trabajos no interactivos introducidos de tipo crontab , at, y otros procesos (por ejemplo, de tipo nohup), que dispongan de ejecución periódica, o estén en ejecución en segundo plano desde hace mucho tiempo (comandos w, jobs). En todo caso, podrían utilizarse medidas preventivas que impidieran trabajos no interactivos a los usuarios, o que solamente los permitiesen a aquellos que lo necesitasen.

- Keyloggers y rootKits: en este caso habrá algún proceso intermediario que intentará capturar nuestras pulsaciones de teclas y las almacenará en algún lugar. Para probar un funcionamiento muy básico de un keylogger muy sencillo, puede verse el comando de sistema script (ver man script). El otro caso, el rootkit (que suele incluir también algún keylogger) suele ser un pack de unos cuantos programas con varias técnicas, y permite al atacante, una vez entra en una cuenta, utilizar diversos elementos como un keylogger, backdoors, troyanos (sustituyendo a comandos del sistema), etc., con tal de obtener información y puertas de entrada al sistema, muchas veces se acompaña de programas que realizan

limpieza de los logs, para eliminar las pruebas de la intrusión.

- Escáner (escaneo de puertos): los escáneres suelen lanzar sobre uno o más sistemas bucles de escaneo de puertos conocidos para detectar los que quedan abiertos y aquellos servicios que están funcionando y que podrían ser susceptibles de ataques.

- Sniffers (husmeadores): evitar intercepciones e impedir así la posibilidad de que se introduzcan escuchas. Una técnica es la construcción hardware de la red, que puede dividirse en segmentos para que el tráfico sólo circule por la zona que se va a utilizar, poner firewalls para unir estos segmentos y poder controlar el tráfico de entrada y salida. Usar técnicas de encriptación para que los mensajes no puedan ser leídos e interpretados por alguien que escuche la red.

- Hijacking (o 'secuestro'): implementar mecanismos de encriptación en los servicios, requerir autentificación y, si es posible, que esta autentificación se renueve periódicamente. Controlar el tráfico entrante o saliente por firewalls. Monitorizar la red para detectar flujos de tráfico sospechosos.

- Buffer overflows: suelen ser comunes como bugs o agujeros del sistema, suelen solucionarse por actualización del software. En todo caso, pueden observarse por logs situaciones extrañas de caída de servicios que deberían estar funcionando.

- Denial of Service ('ataque DoS'), y otros como SYN flood, o correos bombing tomar medidas de bloqueo de tráfico innecesario en nuestra red (por ejemplo, por medio de firewalls). En aquellos servicios que se pueda, habrá que controlar tamaños de buffer, número de clientes por atender, timeouts de cierre de conexiones, capacidades del servicio, etc.

- spoofing: a) IP spoofing, b) ARP spoofing, c) correo electrónico. Estos casos necesitan una fuerte encriptación de los servicios, control por firewalls, mecanismos de autentificación basados en varios aspectos (por ejemplo, no basarse en la IP, si ésta pudiera verse comprometida), se pueden implementar mecanismos que controlen sesiones establecidas y se basen en varios parámetros de la máquina a la vez (sistema operativo, procesador, IP, dirección Ethernet, etc.). También monitorizar sistemas DNS, cachés de ARP, spools de correo, etc., para detectar cambios en la información que invaliden a anteriores.

- Ingeniería social: no es propiamente una cuestión informática, pero también es necesaria para que las personas no empeoren la seguridad. Medidas adecuadas como aumentar la información o educar a usuarios y técnicos en temas de seguridad: controlar

qué personal dispondrá de información crítica de seguridad y en qué condiciones puede cederla a otros. Los servicios de ayuda y mantenimiento de una empresa pueden ser un punto crítico: controlar quién posee información de seguridad, y cómo la usa.

Respecto a usuarios finales, mejorar su cultura de contraseñas, evitar que la dejen apuntada en cualquier sitio, a la vista de terceros, o simplemente la divulguen.

4.7. SEGURIDAD DEL SISTEMA

Ante los posibles ataques, tenemos que disponer de mecanismos de prevención, detección y recuperación de nuestros sistemas.

Para la prevención local, hay que examinar los diferentes mecanismos de autentificación y permisos de accesos a los recursos para definirlos correctamente y poder garantizar la confidencialidad y la integridad de nuestra información. En este caso, nos estaremos protegiendo de atacantes que hayan obtenido acceso a nuestro sistema, o bien de usuarios hostiles que quieran saltarse las restricciones impuestas en el sistema.

Respecto a la seguridad en red, tenemos que garantizar que los recursos que ofrecemos (si proporcionamos unos determinados servicios) tienen los parámetros de confidencialidad necesarios y que los servicios no pueden ser usados por terceros no deseados, de modo que, un primer paso será controlar que los servicios ofrecidos sean los que realmente queremos, y que no estamos ofreciendo además otros servicios que no tenemos controlados.

Respecto a las aplicaciones y a los mismos servicios, además de garantizar la correcta configuración de niveles de acceso mediante permisos y de autentificación de los usuarios permitidos, tendremos que vigilar la posible explotación de bugs en el software. Cualquier aplicación, por muy bien diseñada e implementada que esté, puede tener un número más o menos alto de errores que pueden ser aprovechados para, con ciertas técnicas, saltarse las restricciones impuestas.

4.8. SEGURIDAD LOCAL

La seguridad local es básica para la protección del sistema, ya que normalmente, tras un primer intento de acceso desde la red, es la segunda barrera de protección antes de que un ataque se haga con parte del control de la máquina. Además, la mayoría de los ataques acaban haciendo uso de recursos internos del sistema.

4.8.1 Bootloaders. Respecto a la seguridad local, ya en arranque se nos pueden presentar

problemas debido al acceso físico que un intruso pudiera tener a la máquina.

Uno de los problemas ya está en el arranque del sistema. Si el sistema puede arrancarse de disco o CD, un atacante podría acceder a los datos de una partición Linux (o también Windows) tan sólo con montar el sistema de ficheros y podría colocarse como usuario root sin necesidad de conocer ninguna contraseña.

4.8.2. Passwords y shadows. Las contraseñas típicas de los sistema UNIX iniciales (y de primeras versiones GNU/Linux), estaban encriptadas mediante unos algoritmos DES (pero con claves pequeñas, y una llamada de sistema se encargaba de encriptar y desencriptar, en concreto crypt (ver man).

Pero el problema está en que este fichero es legible por cualquier usuario, con lo que un atacante podía obtener el fichero y utilizar un ataque de fuerza bruta, hasta que desencriptase passwords contenidos en el fichero, o bien mediante ataques de fuerza bruta por medio de diccionarios.

El primer paso es utilizar los nuevos ficheros /etc/shadow, donde se guarda ahora la contraseña. Este fichero es sólo legible por el root, y por nadie más. En este caso, en /etc/passwd aparece un asterisco (*) donde antes estaba la contraseña encriptada. Por defecto, las distribuciones de GNU/Linux actuales usan contraseñas de tipo shadow a no ser que se les diga que no las usen.

Respecto a las herramientas, es interesante disponer de un cracker de contraseñas (o sea, un programa para obtener contraseñas), para comprobar la situación real de seguridad de las cuentas de nuestros usuarios, y forzar así el cambio en las que detectemos inseguras

4.8.3. Suid y sticky bits. Otro problema importante son algunos permisos especiales que son utilizados sobre ficheros o script.

El bit sticky se utiliza sobre todo en directorios temporales en que queremos que en algunos grupos (a veces no relacionados), cualquier usuario pueda escribir, pero sólo pueda borrar bien el propietario del directorio, o bien el propietario del fichero que esté en el directorio. Un ejemplo clásico de este bit, es el directorio temporal /tmp. Hay que vigilar que no haya directorios de este tipo, ya que pueden permitir que cualquiera es- criba en ellos, por lo que habrá que comprobar que no haya más que los puramente necesarios como temporales. El bit se coloca mediante (chmod +t dir), y puede quitarse con -t. En un ls aparecerá como un directorio con permisos drwxrwxrwt .

El bit setuid permite a un usuario ejecutar (ya sea un ejecutable o un shell script) con los permisos de

otro usuario. Esto en algún caso puede ser de utilidad, pero es potencialmente peligroso. Es el caso, por ejemplo, de programas con setuid de root: un usuario, aunque no tiene permisos de root, puede ejecutar un programa con setuid que puede disponer de permisos internos de usuario root. Esto es muy peligroso en caso de scripts, ya que podrían editarse y modificarse para hacer cualquier cosa. Por lo tanto, hay que tener controlados estos programas, y en caso de que no se necesite el setuid, eliminarlo. El bit se coloca mediante chmod +s, ya sea aplicándolo al propietario (se llama entonces suid) o al grupo (se llama bit sgid); puede quitarse con -s. En el caso de visualizar con ls, el fichero aparece- rá con –rwSrw-rw, si es sólo suid, en sgid la S aparecería tras la segunda w.

4.8.4. Habilitación de hosts. En el sistema hay unos cuantos ficheros de configuración especiales que permiten habilitar el acceso a una serie de hosts a algunos ser- vicios de red, pero cuyos errores pueden permitir atacar después la Seguridad local. Nos podemos encontrar con:

- rhosts de usuario: permite que un usuario pueda especificar una serie de máquinas (y usuarios) que pueden usar su cuenta mediante comandos "r" (rsh, rcp, ...) sin necesidad de introducir la contraseña de la cuenta. Esto es potencialmente peligroso, ya que una mala configuración del usuario podría permitir entrar a usuarios no deseados, o que un atacante (con acceso a la cuenta del usuario) cambiase las direcciones en .rhosts para poder entrar cómodamente sin ningún control. Normalmente, no se tendría que permitir crear estos archivos, e incluso habría que borrarlos completamente y deshabilitar los comandos "r".

- /etc/hosts.equiv: es exactamente lo mismo que los ficheros .rhosts pero a nivel de máquina, especificando qué servicios, qué usuarios y qué grupos pueden acceder sin control de contraseña a los servicios "r". Además, un error como poner en una línea de ese fichero un "+", permite el acceso a "cualquier" máquina. Normalmente, hoy en día tampoco suele existir este fichero, y siempre existe la alternativa del servicio ssh a los "r".

- /etc/hosts.lpd: en el sistema de impresión LPD se utilizaba para poner las máquinas que podían acceder al sistema de impresión. Hay que tener mucho cuidado, si no estamos sirviendo, de deshabilitar completamente el acceso al sistema, y en caso de que sí lo estemos, restringir al máximo las máquinas que realmente hacen uso del mismo. O intentar cambiar a un sistema CUPS o LPRng, que tienen mucho más control sobre los servicios.

4.8.5. Alteraciones del sistema. Otro problema puede ser la alteración de comandos o

configuraciones básicas del sistema, mediante la introducción de troyanos o backdoors en el sistema, por la simple introducción de software que sustituya o modifique ligeramente el comportamiento del software de sistema.

En caso de estas alteraciones, será imprescindible usar políticas de auditoría de los cambios, ya sea por medio de cálculo de firmas o sumas (gpg o md5), o bien mediante algún software de control como Tripwire o AIDE. Para los troyanos podemos hacer diferentes tipos de detecciones, o bien utilizar herramientas como chkrootkit, si éstos viniesen de la instalación de algún rootkit conocido.

4.9. SEGURIDAD EN LA RED

4.9.1. Cliente de servicios. Básicamente se tiene que asegurar que no ponemos en peligro a nuestros usuarios (o se pongan en peligro solos) por usar servicios inseguros. Evitar el uso de servicios que no utilicen encriptación de datos y contraseñas (ftp, telnet, correo no seguro). Utilizar en este caso técnicas de conexión encriptada, como SSH y SSL.

4.9.2. Servidor: inetd y xinetd. La configuración de los servicios de red, tal como hemos visto, se realiza desde varios lugares:

- En /etc/inetd.conf o el equivalente directorio/etc/xinetd.d: estos sistemas son una especie de "superservidores", ya que controlan varios servicios hijos y sus condiciones de arranque. El servicio inetd es utilizado en Debían, y xinetd, en Red Hat (y en Debían puede ser puesto opcionalmente en sustitución de inetd).

- Servidores iniciados en arranque: según el runlevel tendremos una serie de servicios arrancados. El arranque se originará en el directorio asociado al runlevel.

- Otros servicios de tipo RPC: asociados a llamadas remotas entre máquinas son, por ejemplo, utilizados en NIS y NFS. Puede examinarse cuáles hay con el comando rpcinfo –p.

Otros ficheros de apoyo (con información útil) son: /etc/services, que está formado por una lista de servicios locales o de red conocidos, junto con el nombre del protocolo (tcp, udp u otros), que se utiliza en el servicio, y el puerto que utiliza; /etc/protocols es una lista de protocolos conocidos; y /etc/rpc es una lista de servidores RPC, junto con los puertos usados.

La alternativa es usar clientes y servidores seguros que soporten la encriptación de los mensajes y autentificación de los participantes. Actualmente la solución más usada es mediante la utilización

del paquete OpenSSH (que puede combinarse también con OpenSSL para entornos web).

Respecto a SSH, también hay que tener la precaución de usar ssh version 2 (o simplemente ssh2). La primera versión tiene algunos exploits conocidos; hay que tener cuidado al instalar OpenSSH, y si no necesitamos la primera versión, instalar sólo soporte para SSH2.

4.10. DETECCIÓN DE INTRUSOS

Con los sistemas de detección de intrusos (IDS) se quiere dar un paso más adelante. Una vez hayamos podido configurar más o menos correctamente nuestra seguridad, el paso siguiente consistirá en una detección y prevención activa de las intrusiones.

Los sistemas IDS crean procedimientos de escucha y generación de alertas al detectar situaciones sospechosas, o sea, buscamos síntomas de posibles accidentes de seguridad.

4.11. PROTECCIÓN A TRAVÉS FILTRADO (wrappers y firewalls)

Los TCP wrappers son un software que actúa de intermediario entre las peticiones del usuario de servicio y los daemons de los servidores que ofrecen el servicio. Muchas de las distribuciones vienen ya con los wrappers activados y podemos configurar los niveles de acceso. Los wrappers se suelen utilizar en combinación con inetd o xinetd, de manera que protejan los servicios que ofrecen.

El wrapper básicamente sustituye al daemon (demonio) del servicio por otro que actúa de intermediario (llamado tcpd, normalmente en /usr/sbin/tcpd). Cuando éste recibe una petición, verifica el usuario y origen de ésta, para determinar si la configuración del wrapper del servicio permite o no utilizarlo. Además, incorpora facilidades de ge- nerar logs, o bien informar por correo de los posibles intentos de ac- ceso y después ejecuta el daemon adecuado asignado al servicio.

También existe un método alternativo de wrapper TCP, que consiste en que la aplicación original se compile con la biblioteca de wrappers. Entonces la aplicación no tiene por qué estar en inetd, y podremos controlarla igual que el primer caso con la configuración que comentamos a continuación.

El sistema de wrappers se controla donde los ficheros /etc/hosts.deny, en que especificamos qué servicios denegamos y a quién, por medio de opciones, como un pequeño shell para guardar la información del intento, y el fichero /etc/hosts.allow, donde solemos colocar qué servicio vamos a utilizar, seguido de la lista de a quién dejamos entrar.

4.11.1. Firewalls. Un firewall (o cortafuegos) es un sistema o grupo de sistemas que refuerza las políticas de control de acceso entre redes. El firewall puede estar implementado en software, como una aplicación especializada corriendo en un computador individual, o bien puede tratarse de un dispositivo especial dedicado a proteger uno o más computadores.

En general dispondremos, o bien de la aplicación de firewall para proteger una máquina concreta conectada directamente a Internet (ya sea directa o por proveedor), o bien podemos colocar en nuestra red una o varias máquinas dedicadas a esta función, de modo que protejan nuestra red interna.

4.11.2. Netfilter: IPtables. El kernel Linux (a partir 2.4) proporciona un subsistema de filtrado denominado Netfilter que proporciona características de filtrado de paquetes y también NAT. Este sistema permite usar diferentes interfaces de filtrado, entre ellas, la más usada se denomina IPtables. La interfaz del comando iptables permite realizar las diferentes tareas de configuración de las reglas que afectan al sistema de filtrado: ya sea generación de logs, acciones de pre y post routing de paquetes, NAT, y forwarding (reenvío) de puertos.

Arranque del servicio con: **/**etc**/**init.d**/**iptables start, si no estaba configurado ya en el runlevel.

El comando iptables -L lista las reglas activas en ese momento en cada una de las cadenas (chains). Si no se han configurado previamente, suelen ser de aceptar todos los paquetes de las cadenas (o chains) de input (entrada), output (salida) y forward (reenvío).

4.12. HERRAMIENTAS DE SEGURIDAD

A continuación, se describirán muy brevemente algunas herramientas de software, así como algunos usos que se le pueden dar:

a) Ethereal: es un analizador de protocolos y captura el tráfico de la red (es un sniffer). Permite visualizar el tráfico capturado, ver estadísticas y datos de los paquetes individuales, y agruparlos, ya sea por origen, destino, puertos o protocolo. Puede incluso reconstruir el tráfico de una sesión entera de un protocolo TCP.

b) TripWire: esta herramienta mantiene una base de datos de sumas de comprobación de los ficheros importantes del sistema. Puede servir como sistema IDS preventivo. Sirve para "tomar" una foto del sistema, poder comparar después cualquier modificación introducida y comprobar que éste no haya sido corrompido por un atacante. El objetivo aquí es proteger los archivos de la propia máquina, evitar que se produzcan cambios.

c) Nmap : es una herramienta de escaneo de puertos para redes grandes. Puede escanear desde máquinas individuales a segmentos de red. Permite diversos modelos de scan, dependiendo de las protecciones que tenga el sistema. También tiene técnicas que permiten determinar el sistema operativo que usan las máquinas remotas. Puede emplear diferentes paquetes de TCP y UDP para probar las conexiones. Existe una interfaz gráfica denominada xnmap.

d) Snort: es un sistema IDS que permite realizar análisis de tráfico en tiempo real y guardar logs de los mensajes. Permite realizar análisis de los protocolos, búsquedas por patrones (protocolo, origen, destino, etc.). Puede ser usado para detectar di- versos tipos de ataques

d) Nessus: es un detector de vulnerabilidades conocidas (ya soporta más de 1.200) y asesora sobre cómo mejorar la seguridad para las detectadas. Es un programa modular que incluye una serie de plugins para los diferentes análisis.

5. RESULTADOS Y PRUEBAS

En este capítulo se mostrar y evidenciar la configuración del servidor de la Contraloría Municipal de Barrancabermeja.

Información de las interfaces de red, como se puede ver en este caso la interface es llamada eth0

```
root@contrabar ~]# netstat -i
Kernel Interface table
Iface       MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR
TX-DRP TX-OVR Flg
eth0       1500   0     4265      0      0      0     3185      0
0      0 BMRU
lo        16436   0     1420      0      0      0     1420      0
0      0 LRU
```

Continuando con la idea de identificar las interfaces de red del servidor, se ejecuta el comando ifconfig para evidenciar la información de los parámetros cargados en la tarjeta de red del servidor.

```
[root@contrabar ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:17:A4:0F:68:CD
          inet addr:192.168.1.10  Bcast:192.168.1.255
Mask:255.255.255.0
          inet6 addr: fe80::217:a4ff:fe0f:68cd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4775 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3193 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:683174 (667.1 KiB)  TX bytes:256507 (250.4 KiB)
          Interrupt:209
```

Al ejecutar el comando nmap con la dirección ip del servidor, este nos muestra la información asociada entre los servicios arrancados y los servicios de red

```
root@contrabar ~]# nmap 192.168.1.10
```

```
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2007-02-07
16:10 COT
Interesting ports on 192.168.1.10:
(The 1654 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
3128/tcp  open  squid-http
10000/tcp open  snet-sensor-mgmt
```

5.1. VERIFICACION DEL SERVICIO APACHE

Para probar el buen funcionamiento del servicio de Apache, se cargó una pagina web diferente a la que trae el servidor apache por defecto para verificar que al llamar el localhost nos llama la pagina principal de la entidad.

Esta página se encuentra localizada en el directorio /var/www/html. Dentro de este directorio la página se llama index.php.

5.2 VERIFICACIÓN DE SERVICIO DE CORREO SENDMAIL

Como es sabido el correo nativo que trae Linux es manejado por el servicio sendmail, a este servicio se le encuentran incorporadas algunas utilidades para mejorar su seguridad y evitar spam.

A continuación se reinicia el servicio

```
Apagando sendmail:                                         [  OK  ]
Desactivación de sm-client:                                [  OK  ]
[root@contrabar ~]# service sendmail start
Iniciando sendmail:                                        [  OK  ]
Inicio de sm-client:                                       [  OK  ]
```

A continuación se muestran los directorios donde se encuentra alojado el servicio, para verificar que los archivos de configuración se encuentran disponibles.

```
[root@contrabar ~]# which sendmail
/usr/sbin/sendmail
[root@contrabar ~]# ls -l /usr/sbin/sendmail
lrwxrwxrwx  1 root root 21 ene  5 17:31 /usr/sbin/sendmail ->
/etc/alternatives/mta
[root@contrabar ~]# ls -l /usr/sbin/sendmail.sendmail
-rwxr-sr-x  1 root smmsp 774264 may  6  2005
/usr/sbin/sendmail.sendmail
```

Luego se realiza un test completo de sendmail, esta prueba se realizó con un envío de correo desde el usuario prueba a una cuenta en el servidor www.contraloriabarrancabermeja.gov.co, en este caso se envia desde el usuario prueba al usuario info, a continuación veremos los pasos que se siguieron:

```
[root@contrabar ~]# su - prueba
[prueba@contrabar ~]$ cat correo
esta es una prueba de correo electronico por favor replicarlo tan
pronto lo reci[prueba@contrabar ~]$ /usr/lib/sendmail -v
info@contraloriabarrancabermeja.gov.co <correo
info@contraloriabarrancabermeja.gov.co... Connecting to [127.0.0.1]
via
relay...
220 contrabar.contraloriabar.com ESMTP Sendmail 8.13.4/8.13.4; Wed, 7
Feb 2007 16:26:53 -0500
>>> EHLO contrabar.contraloriabar.com
250-contrabar.contraloriabar.com Hello contrabar.contraloriabar.com
[127.0.0.1], pleased to meet you
```

```
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH DIGEST-MD5 CRAM-MD5
250-DELIVERBY
250 HELP
>>> MAIL From:<prueba@contrabar.contraloriabar.com> SIZE=82
AUTH=prueba@contrabar.contraloriabar.com
250 2.1.0 <prueba@contrabar.contraloriabar.com>... Sender ok
>>> RCPT To:<info@contraloriabarrancabermeja.gov.co>
>>> DATA
250 2.1.5 <info@contraloriabarrancabermeja.gov.co>... Recipient ok
354 Enter mail, end with "." on a line by itself
>>> .
250 2.0.0 l17LQrOS004170 Message accepted for delivery
info@contraloriabarrancabermeja.gov.co... Sent (l17LQrOS004170 Message
accepted for delivery)
Closing connection to [127.0.0.1]
>>> QUIT
221 2.0.0 contrabar.contraloriabar.com closing connection
```

Además para verificar la configuración de los servicios POP3, IMAP bajo el servicio de Dovecot, se realiza un mapeo de los puertos para verificar que los puertos de salida se encuentren disponibles, como se puede ver se encuentran abiertos los puertos 110 y 143 que corresponden a pop3 e imap respectivamente.

```
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2007-02-07
16:54 COT
Interesting ports on 192.168.1.10:
(The 1650 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
143/tcp   open  imap
443/tcp   open  https
445/tcp   open  microsoft-ds
```

```
993/tcp   open  imaps
995/tcp   open  pop3s
3128/tcp  open  squid-http
10000/tcp open  snet-sensor-mgmt

Nmap finished: 1 IP address (1 host up) scanned in 0.216 seconds
```

Para evidenciar un poco mas sendmail, se considera prudente mostrar los archivos de access y local-host-name.

```
 [prueba@contrabar mail]$ cat access
# Check the /usr/share/doc/sendmail/README.cf file for a description
# of the format of this file. (search for access_db in that file)
# The /usr/share/doc/sendmail/README.cf is part of the sendmail-doc
# package.
#
# by default we allow relaying from localhost...
localhost.localdomain           RELAY
localhost                       RELAY
127.0.0.1                       RELAY
contraloriabarrancabermeja.gov.co       RELAY
192.168.0.0                     RELAY


[prueba@contrabar mail]$ cat local-host-names
localhost
localhost.localdomain
contraloriabarrancabermeja.gov.co
[prueba@contrabar mail]$
```

Además como protección a todo esto se instaló el servicio de MailScanner para vacunar los correos electrónicos y algunas restricciones de spam para el servicio de spamassassin, teniendo en cuenta reglas como que no diga sexo en ninguno de los encabezados, no dentro de su contenido.

5.3. PRUEBAS CON SQUID (PROXY)

Se muestra en los siguientes archivos la seguridad y los permisos para navegación dentro del Proxy que genera el servicio squid

Archivo ERR_ACCESS_DENIED

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<HTML><HEAD><META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=iso-8859-1">
<TITLE>ERROR: The requested URL could not be retrieved</TITLE>
<STYLE
type="text/css"><!--BODY{background-color:#ffffff;font-
family:verdana,sans-serif}PRE{font-family:sans-serif}--></STYLE>
</HEAD><BODY>
<H1>ERROR</H1>
<H2>The requested URL could not be retrieved</H2>
<HR noshade size="1px">
<P>
While trying to retrieve the URL:
<A HREF="%U">%U</A>
<P>
The following error was encountered:
<UL>
<LI>
<STRONG>
Access Denied.
</STRONG>
<P>
El Acceso a internet esta restringido en este sitio,
por favor contacte al administrador de la Red</UL>
<P>Your cache administrator is <A HREF="mailto:%w">%w</A>.
```

Archivo ERR_DNS_FAIL

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML><HEAD><META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=iso-8859-1">
<TITLE>ERROR: The requested URL could not be retrieved</TITLE>
<STYLE
type="text/css"><!--BODY{background-color:#ffffff;font-
family:verdana,sans-serif}PRE{font-family:sans-serif}--></STYLE>
</HEAD><BODY>
<H2>The requested URL could not be retrieved</H2>
<HR noshade size="1px">
<P>
While trying to retrieve the URL:
<A HREF="%U">%U</A>
<P>
The following error was encountered:
<BLOCKQUOTE>
Unable to determine IP address from host name for
<I>%H</I>
</BLOCKQUOTE>
```

```
<P>
The dnsserver returned:
<BLOCKQUOTE>
%z
</BLOCKQUOTE>

<P>
This means that:
<PRE>
 The cache was not able to resolve the hostname presented in the URL.
 Check if the address is correct.
</PRE>
<P>Your cache administrator is <A HREF="mailto:%w">%w</A>.
```

Archivo De Sitios Negados

```
www.fotosprivadas.com
www.pollasgay.com
www.playboy.com
www.chicas.com
www.sexo.com
www.vayachorrada.com
www.cachondos.com
napster
sex
pron
mp3
xxx
adult
```

Parte del Archivo squid.conf

```
#Default
#http_access deny all
http_access deny sitios_negados
http_access allow localhost
http_access allow red
http_access deny all
#
```

Y en la tabla a continuación quisimos mostrar parte de las estadísticas que genera el squid como lo es la utilización de memoria.

***ARCHIVO DE ESTADISTICAS DE UTILIZACION DE MEMORIA EN SQUID*]**

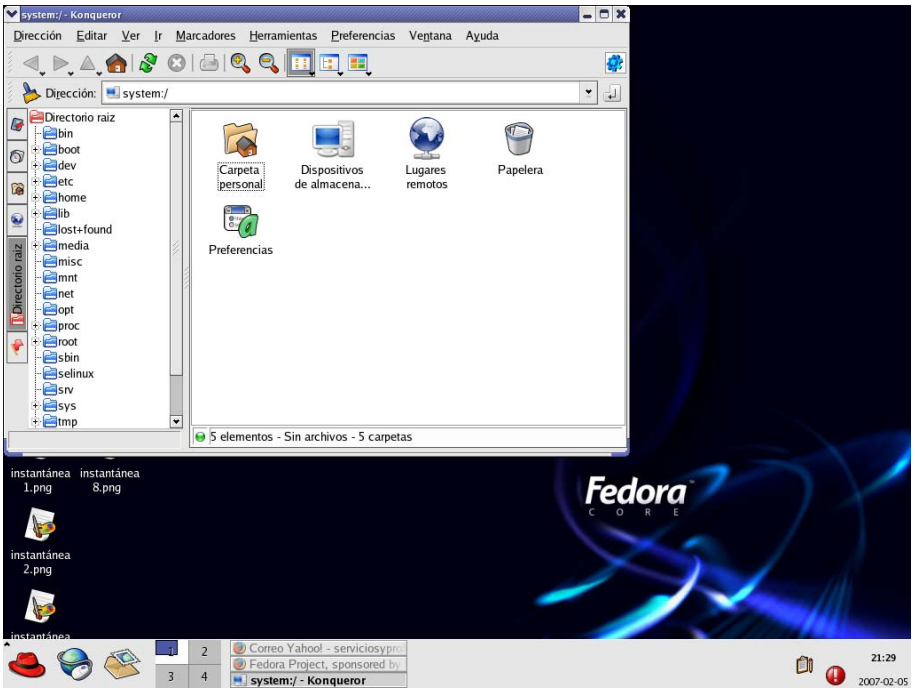
```
Current memory usage:

Pool    Obj Size        Allocated       In Use  Idle    Allocations Saved       Hit
        (bytes) (#)     (KB)    high (KB)       high (hrs)      impact
(%total)        (#)     (KB)    high
(KB)    portion (%alloc)        (#)     (KB)    high (KB)       (number)        (%nu
(number)
2K Buffer       2048    3       6       6       0.03    4       2       4       6
4K Buffer       4096    1       4       4       2.92    2       0       0       4
8K Buffer       8192    1       8       8       0.03    5       0       0       8
Client Socket Buffer    4096    2       8       8       0.01    5       1       4
95.00   40
acl     48      8       1       1       2.92    0       8       1       1       100
acl_ip_data     16      3       1       1       2.92    0       3       1       1
acl_list        12      13      1       1       2.92    0       13      1       1
dwrite_q        24      1       1       1       2.92    0       0       0       1
HttpReply       112     27      3       3       0.03    2       26      3       3
HttpHeaderEntry 20      190     4       4       0.03    2       182     4       4
309
HttpHdrCc       24      25      1       1       2.92    0       25      1       1
intlist 8       2       1       1       2.92    0       2       1       1       100
MemObject       128     26      4       4       0.03    2       26      4       4
mem_node        4108    27      109     109     0.03    65      26
        105     109     96      1       5       9       12      0.04
        2.72    30.77
39
relist  40      2       1       1       2.92    0       2       1       1       100
request_t       728     1       1       1       2.92    0       1       1       1
StoreEntry      48      26      2       2       0.03    1       26      2       2
wordlist        8       10      1       1       2.92    0       6       1       1
ClientInfo      288     2       1       1       0.03    0       2       1       1
MD5 digest      16      26      1       1       0.03    0       26      1       1
clientRequestBuffer     4096    1       4       4       1.84    2       1       4
92.31   13
storeSwapLogData        48      1       1       1       2.92    0       0       0
Short Strings   36      291     11      11      0.03    6       283     10      11
32.01   428
Medium Strings 128      1       1       1       2.92    0       0       0       1
cbdata acl_access (1)   32      11      1       1       2.92    0       11      1
0.00    11
cbdata aclCheck_t (2)   256     2       1       1       0.03    0       1       1
```

```
92.31   26
cbdata clientHttpRequest (3)   564     1       1       1       0.03    0       1
0.28    90.00   10
cbdata ConnStateData (4)       184     1       1       1       1.84    0       1
92.31   13
cbdata RemovalPolicy (15)      56      2       1       1       2.92    0       2
0.00    2
cbdata RemovalPurgeWalker (17) 44      1       1       1       2.92    0       0
22.07   99.99   9097
cbdata store_client (18)       80      1       1       1       0.03    0       1
90.00   10
event   32      10      1       1       2.92    0       7       1       1       70
cbdata body_size (19)  40      1       1       1       2.92    0       1       1
1
CommWriteStateData     24      2       1       1       0.03    0       0       0
close_handler  12      1       1       1       1.84    0       1       1       1
ipcache_entry  68      4       1       1       2.92    0       4       1       1
fqdncache_entry84      1       1       1       2.92    0       1       1       1
cbdata StatObjectsState (22)   28      1       1       1       0.01    0       0
0.00    1
cbdata RebuildState (23)       616     1       1       1       2.92    0       0
0.00    1
Total   -       730     167     167     0.01    100     692     147
        157     88      38      20      39      29972   97.62   90.64
97.62   30702

Cumulative allocated volume: 1.82 MB
Current overhead: 11180 bytes (7.466%)
Idle pool limit: 5.00 MB
memPoolAlloc calls: 30702
memPoolFree calls: 30009

String Pool    Impact
        (%strings)     (%volume)
Short Strings  100     100
Medium Strings 0       0
Long Strings   0       0
Other Strings  0       0
```

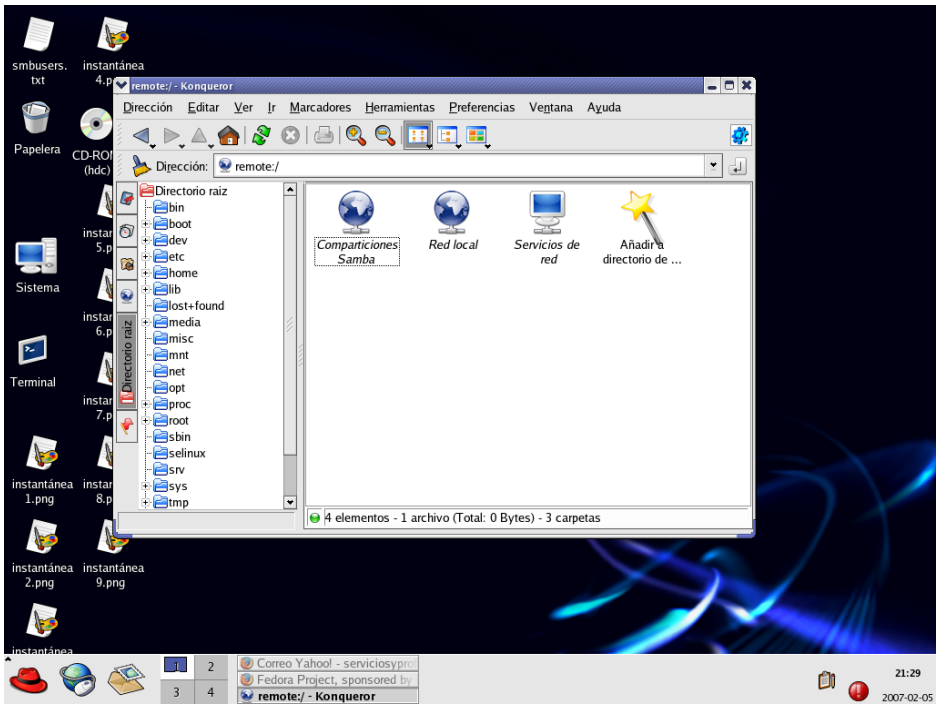
5.4. PRUEBAS SERVICIO SAMBA

Este es uno de los servicios mas necesarios dentro del servidor, ya que las terminales manejan el sistema operativo Linux y para crear integridad en los datos y evitar duplicidades parte de la información se debe guardar en una carpeta compartida dentro del servidor.

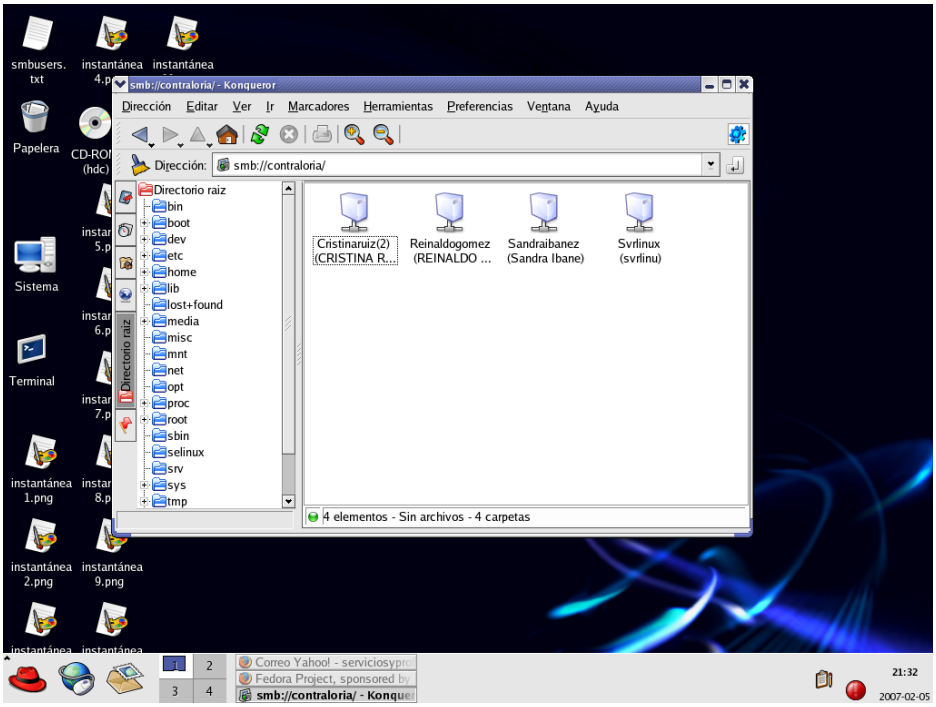
Aquí se muestra entrando al servidor.



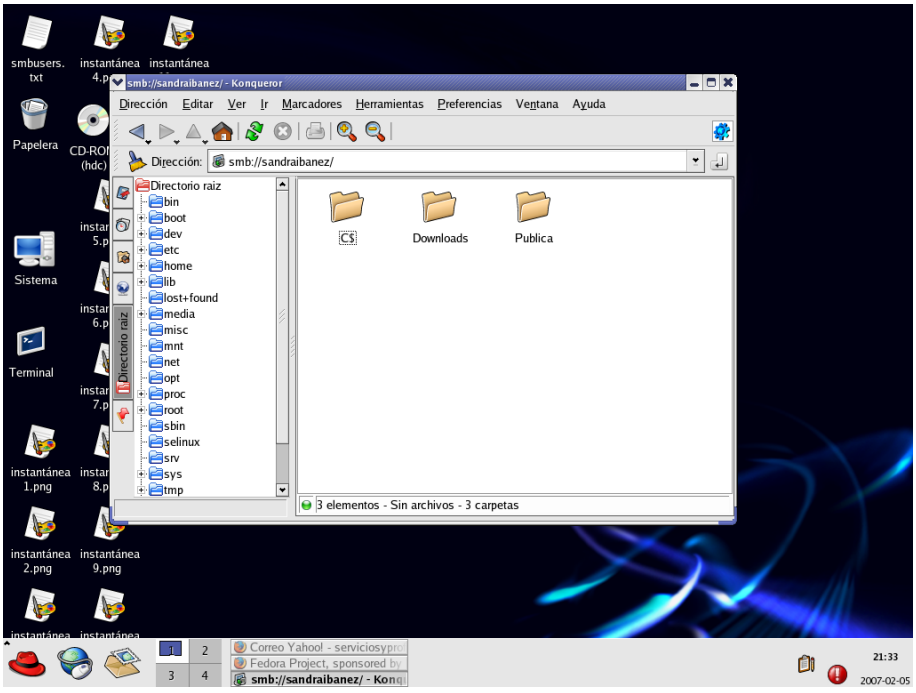
Al dar doble clic sobre la opción lugares remotos.



Al pulsar doble clic sobre con particiones samba y luego sobre la red Contraloría.



Al entrar a cualquiera de los usuarios



Para soportar un poco más la configuración presentamos los apartes de configuración de samba.

```
workgroup = contraloría
netbios name = SvrLinux
server string = %L
encrypt passwords = yes
smb passwd file = /etc/samba/smbpasswd
unix password sync = yes
null passwords = yes

[compartida]
      comment = documents directory
      path = /home/discopublico
      writeable = yes
      printable = no
      browseable = yes
      guest ok = yes
```

El resto del documento completo podrá ser visto dentro de los anexos.

CONCLUSIONES

Una vez realizado este trabajo podemos concluir lo siguiente:

- La IMPLEMENTACION DE UNA INTRANET EN LA CONTRALORIA MUNICIPAL DE BARRANCABERMEJA, fue de gran utilidad porque proporciona herramientas de Internet, a la vez permite la integración lógica de aplicaciones, auxiliando la producción de cada uno de los procesos; es también un importante medio de difusión de información interna a nivel de grupos de trabajo, y proporciona mecanismos de restricción de acceso a nivel de programación. Además esta Intranet fue creada para mayor seguridad y fundamentalmente al compartir archivos, carpetas y recursos.

- Esta implementación mejoró los procesos y procedimientos de la entidad y las aplicaciones que se encentraban aisladas en equipos terminales fueron agrupadas, integrando los departamentos a través de la red, mejorando la comunicación interna, al darle mayor seguridad a la utilización del correo electrónico y optimizando el servicio de Internet, garantizando la seguridad y confiabilidad en la transmisión de datos.

- Es útil porque proporciona herramientas de Internet, a su vez permiten la integración lógica de aplicaciones, auxiliando la producción de cada uno de los procesos, es también un importante medio de difusión de información interna a nivel de grupos de trabajo. Integrando las diferentes dependencias, mejorando las comunicaciones internas y externas y dándole mejor manejo a la información para apoyar cada uno de los procesos que allí se realizan, y en la continuación del mejoramiento de la acreditación de calidad ISO 9000 donde se necesitan diversos procesos a nivel de sistemas y telecomunicaciones.

- Se proporcionan mecanismos de restricción de acceso a nivel de programación (como lo son usuarios y contraseñas) que pueda permitir el ingreso a la red organizacional. Además esta Intranet fue creada para mayor seguridad y fundamentalmente para compartir archivos, carpetas y recursos, garantizándole a los usuarios seguridad en sus transacciones, disminuyendo el riesgo y la incertidumbre por el uso de estas tecnologías, dando mayor credibilidad y aumentando la utilización de la Intranet como plataforma de comunicación y colaboración.

RECOMENDACIONES

A nivel de seguridad informática, aunque dispongamos de firewalls bien configurados, hay que tener presente que no son una medida de seguridad absoluta, ya que hay ataques complejos que pueden saltarse el control, o bien falsear datos que creen confusión. Además, las necesidades de conectividad modernas obligan a veces a crear software que permita el bypass (cruce) de los firewalls:

- Los códigos móviles por web (ActiveX, Java, y JavaScript), cruzan los firewalls, y por lo tanto es difícil proteger los sistemas si éstos son vulnerables a los ataques contra agujeros descubiertos.

Así, aunque los firewalls son una muy buena solución a la mayor parte de la seguridad, siempre pueden tener vulnerabilidades y dejar pasar tráfico que se considere válido, pero que incluya otras fuentes posibles de ataques o vulnerabilidades. En seguridad nunca hay que considerar (y confiar en) una única solución, y esperar que nos proteja absolutamente; hay que examinar los diversos problemas, plantear soluciones que los detecten a tiempo y políticas de prevención que nos curen en salud, por lo que pueda pasar.

Como recomendación de Seguridad Informática:

- Controlar un factor problemático, los usuarios: uno de los factores que puede afectar más a la seguridad es la confidencialidad de las contraseñas, y ésta se ve afectada por el comportamiento de los usuarios en sus labores cotidianas.

- No utilizar ni ejecutar programas de los que no podamos garantizar su origen. Normalmente, muchos distribuidores utilizan mecanismos de comprobación de firmas para verificar que los paquetes de software.

- No utilizar usuarios privilegiados (como root) para el trabajo normal de la máquina; cualquier programa (o aplicación) tendría los permisos para acceder a cualquier parte.

- No acceder remotamente con usuarios privilegiados ni ejecutar programas que puedan

tener privilegios. Y más, si no conocemos, o hemos comprobado, los niveles de seguridad del sistema.

- No utilizar elementos que no sabemos cómo actúan ni intentar descubrirlo a base de repetidas ejecuciones.

- Adquirir un aplicativo como Nesus ya que detectar vulnerabilidades y sirve como asesor en mejorar la seguridad informática.

BIBLIOGRAFIA

GIL TRIANA, Hector

http://www.espaweb.com/ayuda_online/correo_electronico/configurando_spamassassin.html

http://www.linuxparatodos.net/geeklog/staticpages/index.php?page=19-1-como-squid-autenticacion

http://www.linuxparatodos.net/geeklog/staticpages/index.php? : manuales-indice

http://www.linuxparatodos.net/geeklog/staticpages/index.php?page=19-0-como-squid-general

http://mundogeek.net/traducciones/odonovan.html: página de seguridad en linux.

http://www.insecure.org/tools.html: página de software de seguridad informática.

ANEXO No. 1
ARCHIVO httpd.conf

```
#
# Based upon the NCSA server configuration files originally by Rob
McCool.
#
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See <URL:http://httpd.apache.org/docs-2.0/> for detailed information
about
# the directives.
#
# Do NOT simply read the instructions in here without understanding
# what they do.  They're here only as hints or reminders.  If you are
unsure
# consult the online docs. You have been warned.
#
# The configuration directives are grouped into three basic sections:
#  1. Directives that control the operation of the Apache server
process as
a
#     whole (the 'global environment').
#  2. Directives that define the parameters of the 'main' or 'default'
server,
#     which responds to requests that aren't handled by a virtual
host.
#     These directives also provide default values for the settings
#     of all virtual hosts.
#  3. Settings for virtual hosts, which allow Web requests to be sent
to
#     different IP addresses or hostnames and have them handled by the
#     same Apache server process.
#
# Configuration and logfile names: If the filenames you specify for
many
# of the server's control files begin with "/" (or "drive:/" for
Win32), the
# server will use that explicit path.  If the filenames do *not* begin
# with "/", the value of ServerRoot is prepended -- so "logs/foo.log"
# with ServerRoot set to "/etc/httpd" will be interpreted by the
# server as "/etc/httpd/logs/foo.log".
#

### Section 1: Global Environment
#
```

```
# The directives in this section affect the overall operation of
Apache,
# such as the number of concurrent requests it can handle or where it
# can find its configuration files.
#

#
# Don't give away too much information about all the subcomponents
# we are running.  Comment out this line if you don't mind remote
sites
# finding out what major optional modules you are running
ServerTokens OS

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE!  If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the LockFile documentation
# (available at
<URL:http://httpd.apache.org/docs-2.0/mod/mpm_common.html#lockfile>);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
ServerRoot "/etc/httpd"

#
# PidFile: The file in which the server should record its process
# identification number when it starts.
#
PidFile run/httpd.pid

#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 120

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive Off

#
# MaxKeepAliveRequests: The maximum number of requests to allow
```

```
# during a persistent connection. Set to 0 to allow an unlimited
amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request
from
the
# same client on the same connection.
#
KeepAliveTimeout 15

##
## Server-Pool Size Regulation (MPM specific)
##

# prefork MPM
# StartServers: number of server processes to start
# MinSpareServers: minimum number of server processes which are kept
spare
# MaxSpareServers: maximum number of server processes which are kept
spare
# ServerLimit: maximum value for MaxClients for the lifetime of the
server
# MaxClients: maximum number of server processes allowed to start
# MaxRequestsPerChild: maximum number of requests a server process
serves
<IfModule prefork.c>
StartServers       8
MinSpareServers    5
MaxSpareServers   20
ServerLimit      256
MaxClients       256
MaxRequestsPerChild  4000
</IfModule>

# worker MPM
# StartServers: initial number of server processes to start
# MaxClients: maximum number of simultaneous client connections
# MinSpareThreads: minimum number of worker threads which are kept
spare
# MaxSpareThreads: maximum number of worker threads which are kept
spare
# ThreadsPerChild: constant number of worker threads in each server
```

```
process
# MaxRequestsPerChild: maximum number of requests a server process
serves
<IfModule worker.c>
StartServers         2
MaxClients         150
MinSpareThreads     25
MaxSpareThreads     75
ThreadsPerChild     25
MaxRequestsPerChild  0
</IfModule>

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, in addition to the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
#Listen 12.34.56.78:80
Listen 80

#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a
DSO
you
# have to place corresponding `LoadModule' lines at this location so
the
# directives contained in it are actually available _before_ they are
used.
# Statically compiled modules (those listed by `httpd -l') do not need
# to be loaded here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule access_module modules/mod_access.so
LoadModule auth_module modules/mod_auth.so
LoadModule auth_anon_module modules/mod_auth_anon.so
LoadModule auth_dbm_module modules/mod_auth_dbm.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule ldap_module modules/mod_ldap.so
LoadModule auth_ldap_module modules/mod_auth_ldap.so
```

```
LoadModule include_module modules/mod_include.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule logio_module modules/mod_logio.so
LoadModule env_module modules/mod_env.so
LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule cern_meta_module modules/mod_cern_meta.so
LoadModule expires_module modules/mod_expires.so
LoadModule deflate_module modules/mod_deflate.so
LoadModule headers_module modules/mod_headers.so
LoadModule usertrack_module modules/mod_usertrack.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule mime_module modules/mod_mime.so
LoadModule dav_module modules/mod_dav.so
LoadModule status_module modules/mod_status.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule asis_module modules/mod_asis.so
LoadModule info_module modules/mod_info.so
LoadModule dav_fs_module modules/mod_dav_fs.so
LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule dir_module modules/mod_dir.so
LoadModule actions_module modules/mod_actions.so
LoadModule speling_module modules/mod_speling.so
LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule cache_module modules/mod_cache.so
LoadModule suexec_module modules/mod_suexec.so
LoadModule disk_cache_module modules/mod_disk_cache.so
LoadModule file_cache_module modules/mod_file_cache.so
LoadModule mem_cache_module modules/mod_mem_cache.so
LoadModule cgi_module modules/mod_cgi.so

#
# Load config files from the config directory "/etc/httpd/conf.d".
#
Include conf.d/*.conf

#
# ExtendedStatus controls whether Apache will generate "full" status
# information (ExtendedStatus On) or just basic information
(ExtendedStatus
```

```
# Off) when the "server-status" handler is called. The default is Off.
#
#ExtendedStatus On

#
# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run httpd as.
#  . On SCO (ODT 3) use "User nouser" and "Group nogroup".
#  . On HPUX you may not be able to use shared memory as nobody, and
the
#    suggested workaround is to create a user www and use that user.
#  NOTE that some kernels refuse to setgid(Group) or semctl(IPC_SET)
#  when the value of (unsigned)Group is above 60000;
#  don't use Group #-1 on these systems!
#
User apache
Group apache

### Section 2: 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <VirtualHost> definition.  These values also provide defaults for
# any <VirtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <VirtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.
#

#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed.  This address appears on some server-generated pages, such
# as error documents.  e.g. admin@your-domain.com
#
ServerAdmin root@localhost

#
# ServerName gives the name and port that the server uses to identify
itself.
# This can often be determined automatically, but we recommend you
specify
# it explicitly to prevent problems during startup.
#
```

```
# If this is not set to valid DNS name for your host, server-generated
# redirections will not work.  See also the UseCanonicalName
directive.
#
# If your host doesn't have a registered DNS name, enter its IP
address
here.
# You will have to access it by its address anyway, and this will make
# redirections work in a sensible way.
#
#ServerName www.example.com:80

#
# UseCanonicalName: Determines how Apache constructs self-referencing
# URLs and the SERVER_NAME and SERVER_PORT variables.
# When set "Off", Apache will use the Hostname and Port supplied
# by the client.  When set "On", Apache will use the value of the
# ServerName directive.
#
UseCanonicalName Off

#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory,
but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/var/www/html"

#
# Each directory to which Apache has access can be configured with
respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# features.
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
```

```
# you might expect, make sure that you have specifically enabled it
# below.
#

#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/var/www/html">

#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI
MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important.  Please see
# http://httpd.apache.org/docs-2.0/mod/core.html#options
# for more information.
#
    Options Indexes FollowSymLinks

#
# AllowOverride controls what directives may be placed in .htaccess
files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
    AllowOverride None

#
# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from all

</Directory>

#
# UserDir: The name of the directory that is appended onto a user's
home
# directory if a ~user request is received.
#
# The path to the end user account 'public_html' directory must be
```

```
# accessible to the webserver userid.  This usually means that ~userid
# must have permissions of 711, ~userid/public_html must have
permissions
# of 755, and documents contained therein must be world-readable.
# Otherwise, the client will only receive a "403 Forbidden" message.
#
# See also: http://httpd.apache.org/docs/misc/FAQ.html#forbidden
#
<IfModule mod_userdir.c>
    #
    # UserDir is disabled by default since it can confirm the presence
    # of a username on the system (depending on home directory
    # permissions).
    #
    UserDir disable

    #
    # To enable requests to /~user/ to serve the user's public_html
    # directory, remove the "UserDir disable" line above, and
uncomment
    # the following line instead:
    #
    #UserDir public_html

</IfModule>

#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
#<Directory /home/*/public_html>
#    AllowOverride FileInfo AuthConfig Limit
#    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
#    <Limit GET POST OPTIONS>
#        Order allow,deny
#        Allow from all
#    </Limit>
#    <LimitExcept GET POST OPTIONS>
#        Order deny,allow
#        Deny from all
#    </LimitExcept>
#</Directory>

#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
```

```
#
# The index.html.var file (a type-map) is used to deliver content-
# negotiated documents.  The MultiViews Option can be used for the
# same purpose, but it is much slower.
#
DirectoryIndex index.php index.html default.html default.php
index.html.var

#
# AccessFileName: The name of the file to look for in each directory
# for additional configuration directives.  See also the AllowOverride
# directive.
#
AccessFileName .htaccess

#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>

#
# TypesConfig describes where the mime.types file (or equivalent) is
# to be found.
#
TypesConfig /etc/mime.types

#
# DefaultType is the default MIME type the server will use for a
document
# if it cannot otherwise determine one, such as from filename
extensions.
# If your server contains mostly text or HTML documents, "text/plain"
is
# a good value.  If most of your content is binary, such as
applications
# or images, you may want to use "application/octet-stream" instead to
# keep browsers from trying to display binary files as though they are
# text.
#
DefaultType text/plain

#
```

```
# The mod_mime_magic module allows the server to use various hints
from
the
# contents of the file itself to determine its type.  The
MIMEMagicFile
# directive tells the module where the hint definitions are located.
#
<IfModule mod_mime_magic.c>
#   MIMEMagicFile /usr/share/magic.mime
    MIMEMagicFile conf/magic
</IfModule>

#
# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if
people
# had to knowingly turn this feature on, since enabling it means that
# each client request will result in AT LEAST one lookup request to
the
# nameserver.
#
HostnameLookups Off

#
# EnableMMAP: Control whether memory-mapping is used to deliver
# files (assuming that the underlying OS supports it).
# The default is on; turn this off if you serve from NFS-mounted
# filesystems.  On some systems, turning it off (regardless of
# filesystem) can improve performance; for details, please see
# http://httpd.apache.org/docs-2.0/mod/core.html#enablemmap
#
#EnableMMAP off

#
# EnableSendfile: Control whether the sendfile kernel support is
# used to deliver files (assuming that the OS supports it).
# The default is on; turn this off if you serve from NFS-mounted
# filesystems.  Please see
# http://httpd.apache.org/docs-2.0/mod/core.html#enablesendfile
#
#EnableSendfile off

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
```

```
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a
<VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog logs/error_log

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn

#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{User-Agent}i\""
combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# "combinedio" includes actual counts of actual bytes received (%I)
and
sent
(%O); this
# requires the mod_logio module to be loaded.
#LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{User-Agent}i\" %I
%O" combinedio

#
# The location and format of the access logfile (Common Logfile
Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here.  Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
#CustomLog logs/access_log common

#
# If you would like to have separate agent and referer logfiles,
```

```
uncomment
# the following directives.
#
#CustomLog logs/referer_log referer
#CustomLog logs/agent_log agent

#
# For a single logfile with access, agent, and referer information
# (Combined Logfile Format), use the following directive:
#
CustomLog logs/access_log combined

#
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (internal error documents, FTP
directory
# listings, mod_status and mod_info output etc., but not CGI generated
# documents or custom error documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of:  On | Off | EMail
#
ServerSignature On

#
# Aliases: Add here as many aliases as you need (with no limit). The
format
is
# Alias fakename realname
#
# Note that if you include a trailing / on fakename then the server
will
# require it to be present in the URL.  So "/icons" isn't aliased in
this
# example, only "/icons/".  If the fakename is slash-terminated, then
the
# realname must also be slash terminated, and if the fakename omits
the
# trailing slash, the realname must also omit it.
#
# We include the /icons/ alias for FancyIndexed directory listings.
If
you
# do not use FancyIndexing, you may comment this out.
#
Alias /icons/ "/var/www/icons/"
```

```
<Directory "/var/www/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

#
# WebDAV module configuration section.
#
<IfModule mod_dav_fs.c>
    # Location of the WebDAV lock database.
    DAVLockDB /var/lib/dav/lockdb
</IfModule>

#
# ScriptAlias: This controls which directories contain server scripts.
# ScriptAliases are essentially the same as Aliases, except that
# documents in the realname directory are treated as applications and
# run by the server when requested rather than as documents sent to
the
client.
# The same rules about trailing "/" apply to ScriptAlias directives as
to
# Alias.
#
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"

#
# "/var/www/cgi-bin" should be changed to whatever your ScriptAliased
# CGI directory exists, if you have that configured.
#
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>

#
# Redirect allows you to tell clients about documents which used to
exist in
# your server's namespace, but do not anymore. This allows you to tell
the
# clients where to look for the relocated document.
# Example:
```

```
# Redirect permanent /foo http://www.example.com/bar

#
# Directives controlling the display of server-generated directory
listings.
#

#
# IndexOptions: Controls the appearance of server-generated directory
# listings.
#
IndexOptions FancyIndexing VersionSort NameWidth=*

#
# AddIcon* directives tell the server which icon to show for different
# files or filename extensions.  These are only displayed for
# FancyIndexed directories.
#
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^
```

```
#
# DefaultIcon is which icon to show for files which do not have an
icon
# explicitly set.
#
DefaultIcon /icons/unknown.gif

#
# AddDescription allows you to place a short description after a file
in
# server-generated indexes.  These are only displayed for FancyIndexed
# directories.
# Format: AddDescription "description" filename
#
#AddDescription "GZIP compressed document" .gz
#AddDescription "tar archive" .tar
#AddDescription "GZIP compressed tar archive" .tgz

#
# ReadmeName is the name of the README file the server will look for
by
# default, and append to directory listings.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.
ReadmeName README.html
HeaderName HEADER.html

#
# IndexIgnore is a set of filenames which directory indexing should
ignore
# and not include in the listing.  Shell-style wildcarding is
permitted.
#
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t

#
# DefaultLanguage and AddLanguage allows you to specify the language
of
# a document. You can then use content negotiation to give a browser a
# file in a language the user can understand.
#
# Specify a default language. This means that all data
# going out without a specific language tag (see below) will
# be marked with this one. You probably do NOT want to set
```

```
# this unless you are sure it is correct for all cases.
#
# * It is generally better to not mark a page as
# * being a certain language than marking it with the wrong
# * language!
#
# DefaultLanguage nl
#
# Note 1: The suffix does not have to be the same as the language
# keyword --- those with documents in Polish (whose net-standard
# language code is pl) may wish to use "AddLanguage pl .po" to
# avoid the ambiguity with the common suffix for perl scripts.
#
# Note 2: The example entries below illustrate that in some cases
# the two character 'Language' abbreviation is not identical to
# the two character 'Country' code for its country,
# E.g. 'Danmark/dk' versus 'Danish/da'.
#
# Note 3: In the case of 'ltz' we violate the RFC by using a three
char
# specifier. There is 'work in progress' to fix this and get
# the reference data for rfc1766 cleaned up.
#
# Catalan (ca) - Croatian (hr) - Czech (cs) - Danish (da) - Dutch (nl)
# English (en) - Esperanto (eo) - Estonian (et) - French (fr) - German
(de)
# Greek-Modern (el) - Hebrew (he) - Italian (it) - Japanese (ja)
# Korean (ko) - Luxembourgeois* (ltz) - Norwegian Nynorsk (nn)
# Norwegian (no) - Polish (pl) - Portugese (pt)
# Brazilian Portuguese (pt-BR) - Russian (ru) - Swedish (sv)
# Simplified Chinese (zh-CN) - Spanish (es) - Traditional Chinese
(zh-TW)
#
AddLanguage ca .ca
AddLanguage cs .cz .cs
AddLanguage da .dk
AddLanguage de .de
AddLanguage el .el
AddLanguage en .en
AddLanguage eo .eo
AddLanguage es .es
AddLanguage et .et
AddLanguage fr .fr
AddLanguage he .he
AddLanguage hr .hr
AddLanguage it .it
```

```
AddLanguage ja .ja
AddLanguage ko .ko
AddLanguage ltz .ltz
AddLanguage nl .nl
AddLanguage nn .nn
AddLanguage no .no
AddLanguage pl .po
AddLanguage pt .pt
AddLanguage pt-BR .pt-br
AddLanguage ru .ru
AddLanguage sv .sv
AddLanguage zh-CN .zh-cn
AddLanguage zh-TW .zh-tw

#
# LanguagePriority allows you to give precedence to some languages
# in case of a tie during content negotiation.
#
# Just list the languages in decreasing order of preference. We have
# more or less alphabetized them here. You probably want to change
this.
#
LanguagePriority en ca cs da de el eo es et fr he hr it ja ko ltz nl
nn
no
pl pt pt-BR ru sv zh-CN zh-TW

#
# ForceLanguagePriority allows you to serve a result page rather than
# MULTIPLE CHOICES (Prefer) [in case of a tie] or NOT ACCEPTABLE
(Fallback)
# [in case no accepted languages matched the available variants]
#
ForceLanguagePriority Prefer Fallback

#
# Specify a default charset for all content served; this enables
# interpretation of all content as UTF-8 by default.  To use the
# default browser choice (ISO-8859-1), or to allow the META tags
# in HTML content to override this choice, comment out this
# directive:
#
AddDefaultCharset UTF-8

#
# AddType allows you to add to or override the MIME configuration
```

```
# file mime.types for specific file types.
#
#AddType application/x-tar .tgz

#
# AddEncoding allows you to have certain browsers uncompress
# information on the fly. Note: Not all browsers support this.
# Despite the name similarity, the following Add* directives have
nothing
# to do with the FancyIndexing customization directives above.
#
#AddEncoding x-compress .Z
#AddEncoding x-gzip .gz .tgz

# If the AddEncoding directives above are commented-out, then you
# probably should define those extensions to indicate media types:
#
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz

#
# AddHandler allows you to map certain file extensions to "handlers":
# actions unrelated to filetype. These can be either built into the
server
# or added with the Action directive (see below)
#
# To use CGI scripts outside of ScriptAliased directories:
# (You will also need to add "ExecCGI" to the "Options" directive.)
#
#AddHandler cgi-script .cgi

#
# For files that include their own HTTP headers:
#
#AddHandler send-as-is asis

#
# For type maps (negotiated resources):
# (This is enabled by default to allow the Apache "It Worked" page
#  to be distributed in multiple languages.)
#
AddHandler type-map var

#
# Filters allow you to process content before it is sent to the
client.
```

```
#
# To parse .shtml files for server-side includes (SSI):
# (You will also need to add "Includes" to the "Options" directive.)
#
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml

#
# Action lets you define media types that will execute a script
whenever
# a matching file is called. This eliminates the need for repeated URL
# pathnames for oft-used CGI file processors.
# Format: Action media/type /cgi-script/location
# Format: Action handler-name /cgi-script/location
#

#
# Customizable error responses come in three flavors:
# 1) plain text 2) local redirects 3) external redirects
#
# Some examples:
#ErrorDocument 500 "The server made a boo boo."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
#

#
# Putting this all together, we can internationalize error responses.
#
# We use Alias to redirect any /error/HTTP_<error>.html.var response
to
# our collection of by-error message multi-language collections.  We
use
# includes to substitute the appropriate text.
#
# You can modify the messages' appearance without changing any of the
# default HTTP_<error>.html.var files by adding the line:
#
#   Alias /error/include/ "/your/include/path/"
#
# which allows you to create your own set of files by starting with
the
# /var/www/error/include/ files and
# copying them to /your/include/path/, even on a per-VirtualHost
basis.
```

```
#

Alias /error/ "/var/www/error/"

<IfModule mod_negotiation.c>
<IfModule mod_include.c>
    <Directory "/var/www/error">
        AllowOverride None
        Options IncludesNoExec
        AddOutputFilter Includes html
        AddHandler type-map var
        Order allow,deny
        Allow from all
        LanguagePriority en es de fr
        ForceLanguagePriority Prefer Fallback
    </Directory>

#    ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
#    ErrorDocument 401 /error/HTTP_UNAUTHORIZED.html.var
#    ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
#    ErrorDocument 404 /error/HTTP_NOT_FOUND.html.var
#    ErrorDocument 405 /error/HTTP_METHOD_NOT_ALLOWED.html.var
#    ErrorDocument 408 /error/HTTP_REQUEST_TIME_OUT.html.var
#    ErrorDocument 410 /error/HTTP_GONE.html.var
#    ErrorDocument 411 /error/HTTP_LENGTH_REQUIRED.html.var
#    ErrorDocument 412 /error/HTTP_PRECONDITION_FAILED.html.var
#    ErrorDocument 413 /error/HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
#    ErrorDocument 414 /error/HTTP_REQUEST_URI_TOO_LARGE.html.var
#    ErrorDocument 415 /error/HTTP_UNSUPPORTED_MEDIA_TYPE.html.var
#    ErrorDocument 500 /error/HTTP_INTERNAL_SERVER_ERROR.html.var
#    ErrorDocument 501 /error/HTTP_NOT_IMPLEMENTED.html.var
#    ErrorDocument 502 /error/HTTP_BAD_GATEWAY.html.var
#    ErrorDocument 503 /error/HTTP_SERVICE_UNAVAILABLE.html.var
#    ErrorDocument 506 /error/HTTP_VARIANT_ALSO_VARIES.html.var

</IfModule>
</IfModule>

#
# The following directives modify normal HTTP response behavior to
# handle known problems with browser implementations.
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0
force-response-1.0
BrowserMatch "RealPlayer 4\.0" force-response-1.0
```

```
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0

#
# The following directive disables redirects on non-GET requests for
# a directory that does not include the trailing slash.  This fixes a
# problem with Microsoft WebFolders which does not appropriately
handle
# redirects for folders with DAV methods.
# Same deal with Apple's DAV filesystem and Gnome VFS support for DAV.
#
BrowserMatch "Microsoft Data Access Internet Publishing Provider"
redirect-carefully
BrowserMatch "^WebDrive" redirect-carefully
BrowserMatch "^WebDAVFS/1.[012]" redirect-carefully
BrowserMatch "^gnome-vfs" redirect-carefully

#
# Allow server status reports generated by mod_status,
# with the URL of http://servername/server-status
# Change the ".example.com" to match your domain to enable.
#
#<Location /server-status>
#    SetHandler server-status
#    Order deny,allow
#    Deny from all
#    Allow from .example.com
#</Location>

#
# Allow remote server configuration reports, with the URL of
#  http://servername/server-info (requires that mod_info.c be loaded).
# Change the ".example.com" to match your domain to enable.
#
#<Location /server-info>
#    SetHandler server-info
#    Order deny,allow
#    Deny from all
#    Allow from .example.com
#</Location>

#
# Proxy Server directives. Uncomment the following lines to
# enable the proxy server:
#
#<IfModule mod_proxy.c>
```

```
#ProxyRequests On
#
#<Proxy *>
#    Order deny,allow
#    Deny from all
#    Allow from .example.com
#</Proxy>

#
# Enable/disable the handling of HTTP/1.1 "Via:" headers.
# ("Full" adds the server version; "Block" removes all outgoing Via:
headers)
# Set to one of: Off | On | Full | Block
#
#ProxyVia On

#
# To enable a cache of proxied content, uncomment the following lines.
# See http://httpd.apache.org/docs-2.0/mod/mod_cache.html for more
details.
#
#<IfModule mod_disk_cache.c>
#   CacheEnable disk /
#   CacheRoot "/var/cache/mod_proxy"
#</IfModule>
#

#</IfModule>
# End of proxy directives.

### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple domains/hostnames on
your
# machine you can setup VirtualHost containers for them. Most
configurations
# use only name-based virtual hosts so the server doesn't need to
worry
about
# IP addresses. This is indicated by the asterisks in the directives
below.
#
# Please see the documentation at
# <URL:http://httpd.apache.org/docs-2.0/vhosts/>
# for further details before you try to setup virtual hosts.
#
```

```
# You may use the command line option '-S' to verify your virtual host
# configuration.

#
# Use name-based virtual hosting.
#
#NameVirtualHost *:80
#
# NOTE: NameVirtualHost cannot be used without a port specifier
# (e.g. :80) if mod_ssl is being used, due to the nature of the
# SSL protocol.
#

#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
# The first VirtualHost section is used for requests without a known
# server name.
#
#<VirtualHost *:80>
#    ServerAdmin webmaster@dummy-host.example.com
#    DocumentRoot /www/docs/dummy-host.example.com
#    ServerName dummy-host.example.com
#    ErrorLog logs/dummy-host.example.com-error_log
#    CustomLog logs/dummy-host.example.com-access_log common
#</VirtualHost>
```

ANEXO No. 2
ARCHIVO squid.conf

```
#               WELCOME TO SQUID 2
#               ----------
#
#               This is the default Squid configuration file. You may wish
#               to look at the Squid home page (http://www.squid-cache.org/)
#               for the FAQ and other documentation.
#
#               The default Squid config file shows what the defaults for
#               various options happen to be.  If you don't need to change the
#               default, you shouldn't uncomment the line.  Doing so may cause
#               run-time problems.  In some cases "none" refers to no default
#               setting at all, while in other cases it refers to a valid
#               option - the comments for that keyword indicate if this is the
#               case.
#


# NETWORK OPTIONS
#
-----------------------------------------------------------------------------

#  TAG: http_port
#               Usage:      port
#                           hostname:port
#                           1.2.3.4:port
#
#               The socket addresses where Squid will listen for HTTP client
#               requests.  You may specify multiple socket addresses.
#               There are three forms: port alone, hostname with port, and
#               IP address with port.  If you specify a hostname or IP
#               address, Squid binds the socket to that specific
#               address.  This replaces the old 'tcp_incoming_address'
#               option.  Most likely, you do not need to bind to a specific
#               address, so you can use the port number alone.
#
#               The default port number is 3128.
#
#               If you are running Squid in accelerator mode, you
#               probably want to listen on port 80 also, or instead.
#
#               The -a command line option will override the *first* port
#               number listed here.   That option will NOT override an IP
#               address, however.
#
#               You may specify multiple socket addresses on multiple lines.
#
#               If you run Squid on a dual-homed machine with an internal
#               and an external interface we recommend you to specify the
#               internal address:port in http_port. This way Squid will only be
#               visible on the internal address.
#
#Default:
# http_port 3128

#  TAG: https_port
#       Usage:  [ip:]port cert=certificate.pem [key=key.pem] [options...]
#
#       The socket address where Squid will listen for HTTPS client
#       requests.
#
#       This is really only useful for situations where you are running
#       squid in accelerator mode and you want to do the SSL work at the
#       accelerator level.
#
#               You may specify multiple socket addresses on multiple lines,
#               each with their own SSL certificate and/or options.
#
#               Options:
#
```

```
#                cert=      Path to SSL certificate (PEM format)
#
#                key=                  Path to SSL private key file (PEM format)
#                                      if not specified, the certificate file is
#                                      assumed to be a combined certificate and
#                                      key file
#
#                version=   The version of SSL/TLS supported
#                                         1          automatic (default)
#                                         2          SSLv2 only
#                                         3          SSLv3 only
#                                         4          TLSv1 only
#
#                cipher=    Colon separated list of supported ciphers
#
#                options=   Varions SSL engine options. The most important
#                                      being:
#                                         NO_SSLv2  Disallow the use of SSLv2
#                                         NO_SSLv3  Disallow the use of SSLv3
#                                         NO_TLSv1  Disallow the use of TLSv1
#                                      See src/ssl_support.c or OpenSSL documentation
#                                      for a more complete list.
#
#Default:
# none

#  TAG: ssl_unclean_shutdown
#               Some browsers (especially MSIE) bugs out on SSL shutdown
#               messages.
#
#Default:
# ssl_unclean_shutdown off

#  TAG: icp_port
#               The port number where Squid sends and receives ICP queries to
#               and from neighbor caches.  Default is 3130.  To disable use
#               "0".  May be overridden with -u on the command line.
#
#Default:
# icp_port 3130

#  TAG: htcp_port
# Note: This option is only available if Squid is rebuilt with the
#       --enable-htcp option
#
#               The port number where Squid sends and receives HTCP queries to
#               and from neighbor caches.  Default is 4827.  To disable use
#               "0".
#
#Default:
# htcp_port 4827

#  TAG: mcast_groups
#               This tag specifies a list of multicast groups which your server
#               should join to receive multicasted ICP queries.
#
#               NOTE!  Be very careful what you put here!  Be sure you
#               understand the difference between an ICP _query_ and an ICP
#               _reply_.  This option is to be set only if you want to RECEIVE
#               multicast queries.  Do NOT set this option to SEND multicast
#               ICP (use cache_peer for that).  ICP replies are always sent via
#               unicast, so this option does not affect whether or not you will
#               receive replies from multicast group members.
#
#               You must be very careful to NOT use a multicast address which
#               is already in use by another group of caches.
#
#               If you are unsure about multicast, please read the Multicast
#               chapter in the Squid FAQ (http://www.squid-cache.org/FAQ/).
#
#               Usage: mcast_groups 239.128.16.128 224.0.1.20
#
#               By default, Squid doesn't listen on any multicast groups.
```

```
#
#Default:
# none

#  TAG: udp_incoming_address
#  TAG: udp_outgoing_address
#               udp_incoming_address    is used for the ICP socket receiving packets
#                                                   from other caches.
#               udp_outgoing_address    is used for ICP packets sent out to other
#                                                   caches.
#
#               The default behavior is to not bind to any specific address.
#
#               A udp_incoming_address value of 0.0.0.0 indicates Squid
#               should listen for UDP messages on all available interfaces.
#
#               If udp_outgoing_address is set to 255.255.255.255 (the default)
#               it will use the same socket as udp_incoming_address. Only
#               change this if you want to have ICP queries sent using another
#               address than where this Squid listens for ICP queries from other
#               caches.
#
#               NOTE, udp_incoming_address and udp_outgoing_address can not
#               have the same value since they both use port 3130.
#
#Default:
# udp_incoming_address 0.0.0.0
# udp_outgoing_address 255.255.255.255


# OPTIONS WHICH AFFECT THE NEIGHBOR SELECTION ALGORITHM
#
------------------------------------------------------------------------------

#  TAG: cache_peer
#               To specify other caches in a hierarchy, use the format:
#
#                                   cache_peer hostname type http_port icp_port
#
#               For example,
#
#               #                                proxy  icp
#               #         hostname            type     port   port  options
#               #         -------------------- -------- ----- -----  -----------
#               cache_peer parent.foo.net       parent    3128  3130  [proxy-only]
#               cache_peer sib1.foo.net         sibling   3128  3130  [proxy-only]
#               cache_peer sib2.foo.net         sibling   3128  3130  [proxy-only]
#
#                    type:  either 'parent', 'sibling', or 'multicast'.
#
#               proxy_port:  The port number where the cache listens for proxy
#                                   requests.
#
#                icp_port:  Used for querying neighbor caches about
#                                   objects.  To have a non-ICP neighbor
#                                   specify '7' for the ICP port and make sure the
#                                   neighbor machine has the UDP echo port
#                                   enabled in its /etc/inetd.conf file.
#
#                  options: proxy-only
#                                   weight=n
#                                   ttl=n
#                                   no-query
#                                   default
#                                   round-robin
#                                   multicast-responder
#                                   closest-only
#                                   no-digest
#                                   no-netdb-exchange
#                                   no-delay
#                                   login=user:password | PASS | *:password
#                                   connect-timeout=nn
#                                   digest-url=url
```

```
#                       allow-miss
#                       max-conn
#                       htcp
#                       carp-load-factor
#
#                       use 'proxy-only' to specify objects fetched
#                       from this cache should not be saved locally.
#
#                       use 'weight=n' to specify a weighted parent.
#                       The weight must be an integer.  The default weight
#                       is 1, larger weights are favored more.
#
#                       use 'ttl=n' to specify a IP multicast TTL to use
#                       when sending an ICP queries to this address.
#                       Only useful when sending to a multicast group.
#                       Because we don't accept ICP replies from random
#                       hosts, you must configure other group members as
#                       peers with the 'multicast-responder' option below.
#
#                       use 'no-query' to NOT send ICP queries to this
#                       neighbor.
#
#                       use 'default' if this is a parent cache which can
#                       be used as a "last-resort." You should probably
#                       only use 'default' in situations where you cannot
#                       use ICP with your parent cache(s).
#
#                       use 'round-robin' to define a set of parents which
#                       should be used in a round-robin fashion in the
#                       absence of any ICP queries.
#
#                       'multicast-responder' indicates the named peer
#                       is a member of a multicast group.  ICP queries will
#                       not be sent directly to the peer, but ICP replies
#                       will be accepted from it.
#
#                       'closest-only' indicates that, for ICP_OP_MISS
#                       replies, we'll only forward CLOSEST_PARENT_MISSes
#                       and never FIRST_PARENT_MISSes.
#
#                       use 'no-digest' to NOT request cache digests from
#                       this neighbor.
#
#                       'no-netdb-exchange' disables requesting ICMP
#                       RTT database (NetDB) from the neighbor.
#
#                       use 'no-delay' to prevent access to this neighbor
#                       from influencing the delay pools.
#
#                       use 'login=user:password' if this is a personal/workgroup
#                       proxy and your parent requires proxy authentication.
#                       Note: The string can include URL escapes (i.e. %20 for
#                       spaces). This also means % must be written as %%.
#
#                       use 'login=PASS' if users must authenticate against
#                       the upstream proxy. This will pass the users credentials
#                       as they are to the peer proxy. This only works for the
#                       Basic HTTP authentication sheme. Note: To combine this
#                       with proxy_auth both proxies must share the same user
#                       database as HTTP only allows for one proxy login.
#                       Also be warned this will expose your users proxy
#                       password to the peer. USE WITH CAUTION
#
#                       use 'login=*:password' to pass the username to the
#                       upstream cache, but with a fixed password. This is meant
#                       to be used when the peer is in another administrative
#                       domain, but it is still needed to identify each user.
#                       The star can optionally be followed by some extra
#                       information which is added to the username. This can
#                       be used to identify this proxy to the peer, similar to
#                       the login=username:password option above.
#
#                       use 'connect-timeout=nn' to specify a peer
```

```
#                               specific connect timeout (also see the
#                               peer_connect_timeout directive)
#
#                               use 'digest-url=url' to tell Squid to fetch the cache
#                               digest (if digests are enabled) for this host from
#                               the specified URL rather than the Squid default
#                               location.
#
#                               use 'allow-miss' to disable Squid's use of only-if-cached
#                               when forwarding requests to siblings. This is primarily
#                               useful when icp_hit_stale is used by the sibling. To
#                               extensive use of this option may result in forwarding
#                               loops, and you should avoid having two-way peerings
#                               with this option. (for example to deny peer usage on
#                               requests from peer by denying cache_peer_access if the
#                               source is a peer)
#
#                               use 'max-conn' to limit the amount of connections Squid
#                               may open to this peer.
#
#                               use 'htcp' to send HTCP, instead of ICP, queries
#                               to the neighbor.  You probably also want to
#                               set the "icp port" to 4827 instead of 3130.
#
#                               use 'carp-load-factor=f' to define a parent
#                               cache as one participating in a CARP array.
#                               The 'f' values for all CARP parents must add
#                               up to 1.0.
#
#
#               NOTE: non-ICP/HTCP neighbors must be specified as 'parent'.
#
#Default:
# none

#  TAG: cache_peer_domain
#               Use to limit the domains for which a neighbor cache will be
#               queried.  Usage:
#
#               cache_peer_domain cache-host domain [domain ...]
#               cache_peer_domain cache-host !domain
#
#               For example, specifying
#
#                               cache_peer_domain parent.foo.net      .edu
#
#               has the effect such that UDP query packets are sent to
#               'bigserver' only when the requested object exists on a
#               server in the .edu domain.  Prefixing the domainname
#               with '!' means the cache will be queried for objects
#               NOT in that domain.
#
#               NOTE:       * Any number of domains may be given for a cache-host,
#                             either on the same or separate lines.
#                           * When multiple domains are given for a particular
#                             cache-host, the first matched domain is applied.
#                           * Cache hosts with no domain restrictions are queried
#                             for all requests.
#                           * There are no defaults.
#                           * There is also a 'cache_peer_access' tag in the ACL
#                             section.
#
#Default:
# none

#  TAG: neighbor_type_domain
#               usage: neighbor_type_domain neighbor parent|sibling domain domain ...
#
#               Modifying the neighbor type for specific domains is now
#               possible.  You can treat some domains differently than the the
#               default neighbor type specified on the 'cache_peer' line.
#               Normally it should only be necessary to list domains which
#               should be treated differently because the default neighbor type
```

```
#               applies for hostnames which do not match domains listed here.
#
#EXAMPLE:
#               cache_peer  parent cache.foo.org 3128 3130
#               neighbor_type_domain cache.foo.org sibling .com .net
#               neighbor_type_domain cache.foo.org sibling .au .de
#
#Default:
# none

#  TAG: icp_query_timeout                   (msec)
#               Normally Squid will automatically determine an optimal ICP
#               query timeout value based on the round-trip-time of recent ICP
#               queries.  If you want to override the value determined by
#               Squid, set this 'icp_query_timeout' to a non-zero value.  This
#               value is specified in MILLISECONDS, so, to use a 2-second
#               timeout (the old default), you would write:
#
#                               icp_query_timeout 2000
#
#Default:
# icp_query_timeout 0

#  TAG: maximum_icp_query_timeout  (msec)
#               Normally the ICP query timeout is determined dynamically.  But
#               sometimes it can lead to very large values (say 5 seconds).
#               Use this option to put an upper limit on the dynamic timeout
#               value.  Do NOT use this option to always use a fixed (instead
#               of a dynamic) timeout value. To set a fixed timeout see the
#               'icp_query_timeout' directive.
#
#Default:
# maximum_icp_query_timeout 2000

#  TAG: mcast_icp_query_timeout       (msec)
#               For Multicast peers, Squid regularly sends out ICP "probes" to
#               count how many other peers are listening on the given multicast
#               address.  This value specifies how long Squid should wait to
#               count all the replies.  The default is 2000 msec, or 2
#               seconds.
#
#Default:
# mcast_icp_query_timeout 2000

#  TAG: dead_peer_timeout                   (seconds)
#               This controls how long Squid waits to declare a peer cache
#               as "dead."  If there are no ICP replies received in this
#               amount of time, Squid will declare the peer dead and not
#               expect to receive any further ICP replies.  However, it
#               continues to send ICP queries, and will mark the peer as
#               alive upon receipt of the first subsequent ICP reply.
#
#               This timeout also affects when Squid expects to receive ICP
#               replies from peers.  If more than 'dead_peer' seconds have
#               passed since the last ICP reply was received, Squid will not
#               expect to receive an ICP reply on the next query.  Thus, if
#               your time between requests is greater than this timeout, you
#               will see a lot of requests sent DIRECT to origin servers
#               instead of to your parents.
#
#Default:
# dead_peer_timeout 10 seconds

#  TAG: hierarchy_stoplist
#               A list of words which, if found in a URL, cause the object to
#               be handled directly by this cache.  In other words, use this
#               to not query neighbor caches for certain objects.  You may
#               list this option multiple times.
#We recommend you to use at least the following line.
hierarchy_stoplist cgi-bin ?

#  TAG: no_cache
#               A list of ACL elements which, if matched, cause the request to
```

```
#               not be satisfied from the cache and the reply to not be cached.
#               In other words, use this to force certain objects to never be cached.
#
#               You must use the word 'DENY' to indicate the ACL names which should
#               NOT be cached.
#
#We recommend you to use the following two lines.
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY


# OPTIONS WHICH AFFECT THE CACHE SIZE
#
-----------------------------------------------------------------------------

#  TAG: cache_mem       (bytes)
#               NOTE: THIS PARAMETER DOES NOT SPECIFY THE MAXIMUM PROCESS SIZE.
#               IT ONLY PLACES A LIMIT ON HOW MUCH ADDITIONAL MEMORY SQUID WILL
#               USE AS A MEMORY CACHE OF OBJECTS. SQUID USES MEMORY FOR OTHER
#               THINGS AS WELL. SEE THE SQUID FAQ SECTION 8 FOR DETAILS.
#
#               'cache_mem' specifies the ideal amount of memory to be used
#               for:
#                       * In-Transit objects
#                       * Hot Objects
#                       * Negative-Cached objects
#
#               Data for these objects are stored in 4 KB blocks.  This
#               parameter specifies the ideal upper limit on the total size of
#               4 KB blocks allocated.  In-Transit objects take the highest
#               priority.
#
#               In-transit objects have priority over the others.  When
#               additional space is needed for incoming data, negative-cached
#               and hot objects will be released.  In other words, the
#               negative-cached and hot objects will fill up any unused space
#               not needed for in-transit objects.
#
#               If circumstances require, this limit will be exceeded.
#               Specifically, if your incoming request rate requires more than
#               'cache_mem' of memory to hold in-transit objects, Squid will
#               exceed this limit to satisfy the new requests.  When the load
#               decreases, blocks will be freed until the high-water mark is
#               reached.  Thereafter, blocks will be used to store hot
#               objects.
#
#Default:
# cache_mem 8 MB

#  TAG: cache_swap_low(percent, 0-100)
#  TAG: cache_swap_high                 (percent, 0-100)
#
#               The low- and high-water marks for cache object replacement.
#               Replacement begins when the swap (disk) usage is above the
#               low-water mark and attempts to maintain utilization near the
#               low-water mark.  As swap utilization gets close to high-water
#               mark object eviction becomes more aggressive.  If utilization is
#               close to the low-water mark less replacement is done each time.
#
#               Defaults are 90% and 95%. If you have a large cache, 5% could be
#               hundreds of MB. If this is the case you may wish to set these
#               numbers closer together.
#
#Default:
# cache_swap_low 90
# cache_swap_high 95

#  TAG: maximum_object_size             (bytes)
#               Objects larger than this size will NOT be saved on disk.  The
#               value is specified in kilobytes, and the default is 4MB.  If
#               you wish to get a high BYTES hit ratio, you should probably
#               increase this (one 32 MB object hit counts for 3200 10KB
#               hits).  If you wish to increase speed more than your want to
```

```
#               save bandwidth you should leave this low.
#
#               NOTE: if using the LFUDA replacement policy you should increase
#               this value to maximize the byte hit rate improvement of LFUDA!
#               See replacement_policy below for a discussion of this policy.
#
#Default:
# maximum_object_size 4096 KB

#  TAG: minimum_object_size             (bytes)
#               Objects smaller than this size will NOT be saved on disk.  The
#               value is specified in kilobytes, and the default is 0 KB, which
#               means there is no minimum.
#
#Default:
# minimum_object_size 0 KB

#  TAG: maximum_object_size_in_memory           (bytes)
#       Objects greater than this size will not be attempted to kept in
#       the memory cache. This should be set high enough to keep objects
#       accessed frequently in memory to improve performance whilst low
#       enough to keep larger objects from hoarding cache_mem .
#
#Default:
# maximum_object_size_in_memory 8 KB

#  TAG: ipcache_size       (number of entries)
#  TAG: ipcache_low        (percent)
#  TAG: ipcache_high       (percent)
#               The size, low-, and high-water marks for the IP cache.
#
#Default:
# ipcache_size 1024
# ipcache_low 90
# ipcache_high 95

#  TAG: fqdncache_size   (number of entries)
#               Maximum number of FQDN cache entries.
#
#Default:
# fqdncache_size 1024

#  TAG: cache_replacement_policy
#               The cache replacement policy parameter determines which
#               objects are evicted (replaced) when disk space is needed.
#
#                  lru      : Squid's original list based LRU policy
#                  heap GDSF : Greedy-Dual Size Frequency
#                  heap LFUDA: Least Frequently Used with Dynamic Aging
#                  heap LRU  : LRU policy implemented using a heap
#
#               Applies to any cache_dir lines listed below this.
#
#               The LRU policies keeps recently referenced objects.
#
#               The heap GDSF policy optimizes object hit rate by keeping smaller
#               popular objects in cache so it has a better chance of getting a
#               hit.  It achieves a lower byte hit rate than LFUDA though since
#               it evicts larger (possibly popular) objects.
#
#               The heap LFUDA policy keeps popular objects in cache regardless of
#               their size and thus optimizes byte hit rate at the expense of
#               hit rate since one large, popular object will prevent many
#               smaller, slightly less popular objects from being cached.
#
#               Both policies utilize a dynamic aging mechanism that prevents
#               cache pollution that can otherwise occur with frequency-based
#               replacement policies.
#
#               NOTE: if using the LFUDA replacement policy you should increase
#               the value of maximum_object_size above its default of 4096 KB to
#               to maximize the potential byte hit rate improvement of LFUDA.
#
```

```
#               For more information about the GDSF and LFUDA cache replacement
#               policies see http://www.hpl.hp.com/techreports/1999/HPL-1999-69.html
#               and http://fog.hpl.external.hp.com/techreports/98/HPL-98-173.html.
#
#Default:
# cache_replacement_policy lru

#  TAG: memory_replacement_policy
#               The memory replacement policy parameter determines which
#               objects are purged from memory when memory space is needed.
#
#               See cache_replacement_policy for details.
#
#Default:
# memory_replacement_policy lru


# LOGFILE PATHNAMES AND CACHE DIRECTORIES
#
-----------------------------------------------------------------------------

#  TAG: cache_dir
#               Usage:
#
#               cache_dir Type Directory-Name Fs-specific-data [options]
#
#               You can specify multiple cache_dir lines to spread the
#               cache among different disk partitions.
#
#               Type specifies the kind of storage system to use. Only "ufs"
#               is built by default. To eanble any of the other storage systems
#               see the --enable-storeio configure option.
#
#               'Directory' is a top-level directory where cache swap
#               files will be stored.  If you want to use an entire disk
#               for caching, this can be the mount-point directory.
#               The directory must exist and be writable by the Squid
#               process.  Squid will NOT create this directory for you.
#
#               The ufs store type:
#
#               "ufs" is the old well-known Squid storage format that has always
#               been there.
#
#               cache_dir ufs Directory-Name Mbytes L1 L2 [options]
#
#               'Mbytes' is the amount of disk space (MB) to use under this
#               directory.  The default is 100 MB.  Change this to suit your
#               configuration.  Do NOT put the size of your disk drive here.
#               Instead, if you want Squid to use the entire disk drive,
#               subtract 20% and use that value.
#
#               'Level-1' is the number of first-level subdirectories which
#               will be created under the 'Directory'.  The default is 16.
#
#               'Level-2' is the number of second-level subdirectories which
#               will be created under each first-level directory.  The default
#               is 256.
#
#               The aufs store type:
#
#               "aufs" uses the same storage format as "ufs", utilizing
#               POSIX-threads to avoid blocking the main Squid process on
#               disk-I/O. This was formerly known in Squid as async-io.
#
#               cache_dir aufs Directory-Name Mbytes L1 L2 [options]
#
#               see argument descriptions under ufs above
#
#               The diskd store type:
#
#               "diskd" uses the same storage format as "ufs", utilizing a
#               separate process to avoid blocking the main Squid process on
```

```
#               disk-I/O.
#
#               cache_dir diskd Directory-Name Mbytes L1 L2 [options] [Q1=n] [Q2=n]
#
#               see argument descriptions under ufs above
#
#               Q1 specifies the number of unacknowledged I/O requests when Squid
#               stops opening new files. If this many messages are in the queues,
#               Squid won't open new files. Default is 64
#
#               Q2 specifies the number of unacknowledged messages when Squid
#               starts blocking.  If this many messages are in the queues,
#               Squid blocks until it receives some replies. Default is 72
#
#               When Q1 < Q2 (the default), the cache directory is optimized
#               for lower response time at the expense of a decrease in hit
#               ratio.  If Q1 > Q2, the cache directory is optimized for
#               higher hit ratio at the expense of an increase in response
#               time.
#
#               The coss store type:
#
#               block-size=n defines the "block size" for COSS cache_dir's.
#               Squid uses file numbers as block numbers.  Since file numbers
#               are limited to 24 bits, the block size determines the maximum
#               size of the COSS partition.  The default is 512 bytes, which
#               leads to a maximum cache_dir size of 512<<24, or 8 GB.  Note
#               you should not change the coss block size after Squid
#               has written some objects to the cache_dir.
#
#               Common options:
#
#               read-only, this cache_dir is read only.
#
#               max-size=n, refers to the max object size this storedir supports.
#               It is used to initially choose the storedir to dump the object.
#               Note: To make optimal use of the max-size limits you should order
#               the cache_dir lines with the smallest max-size value first and the
#               ones with no max-size specification last.
#
#               Note that for coss, max-size must be less than COSS_MEMBUF_SZ
#               (hard coded at 1 MB).
#
#Default:
# cache_dir ufs /var/spool/squid 100 16 256

#  TAG: cache_access_log
#               Logs the client request activity.  Contains an entry for
#               every HTTP and ICP queries received. To disable, enter "none".
#
#Default:
# cache_access_log /var/log/squid/access.log

#  TAG: cache_log
#               Cache logging file. This is where general information about
#               your cache's behavior goes. You can increase the amount of data
#               logged to this file with the "debug_options" tag below.
#
#Default:
# cache_log /var/log/squid/cache.log

#  TAG: cache_store_log
#               Logs the activities of the storage manager.  Shows which
#               objects are ejected from the cache, and which objects are
#               saved and for how long.  To disable, enter "none". There are
#               not really utilities to analyze this data, so you can safely
#               disable it.
#
#Default:
# cache_store_log /var/log/squid/store.log

#  TAG: cache_swap_log
#               Location for the cache "swap.state" file. This log file holds
```

```
#               the metadata of objects saved on disk.  It is used to rebuild
#               the cache during startup.  Normally this file resides in each
#               'cache_dir' directory, but you may specify an alternate
#               pathname here.  Note you must give a full filename, not just
#               a directory. Since this is the index for the whole object
#               list you CANNOT periodically rotate it!
#
#               If %s can be used in the file name it will be replaced with a
#               a representation of the cache_dir name where each / is replaced
#               with '.'. This is needed to allow adding/removing cache_dir
#               lines when cache_swap_log is being used.
#
#               If have more than one 'cache_dir', and %s is not used in the name
#               these swap logs will have names such as:
#
#                       cache_swap_log.00
#                       cache_swap_log.01
#                       cache_swap_log.02
#
#               The numbered extension (which is added automatically)
#               corresponds to the order of the 'cache_dir' lines in this
#               configuration file.  If you change the order of the 'cache_dir'
#               lines in this file, these log files will NOT correspond to
#               the correct 'cache_dir' entry (unless you manually rename
#               them).  We recommend you do NOT use this option.  It is
#               better to keep these log files in each 'cache_dir' directory.
#
#Default:
# none

#  TAG: emulate_httpd_log               on|off
#               The Cache can emulate the log file format which many 'httpd'
#               programs use.  To disable/enable this emulation, set
#               emulate_httpd_log to 'off' or 'on'.  The default
#               is to use the native log format since it includes useful
#               information Squid-specific log analyzers use.
#
#Default:
# emulate_httpd_log off

#  TAG: log_ip_on_direct on|off
#               Log the destination IP address in the hierarchy log tag when going
#               direct. Earlier Squid versions logged the hostname here. If you
#               prefer the old way set this to off.
#
#Default:
# log_ip_on_direct on

#  TAG: mime_table
#               Pathname to Squid's MIME table. You shouldn't need to change
#               this, but the default file contains examples and formatting
#               information if you do.
#
#Default:
# mime_table /etc/squid/mime.conf

#  TAG: log_mime_hdrs   on|off
#               The Cache can record both the request and the response MIME
#               headers for each HTTP transaction.  The headers are encoded
#               safely and will appear as two bracketed fields at the end of
#               the access log (for either the native or httpd-emulated log
#               formats).  To enable this logging set log_mime_hdrs to 'on'.
#
#Default:
# log_mime_hdrs off

#  TAG: useragent_log
#               Squid will write the User-Agent field from HTTP requests
#               to the filename specified here.  By default useragent_log
#               is disabled.
#
#Default:
# none
```

```
#  TAG: referer_log
#               Squid will write the Referer field from HTTP requests to the
#               filename specified here.  By default referer_log is disabled.
#
#Default:
# none

#  TAG: pid_filename
#               A filename to write the process-id to.  To disable, enter "none".
#
#Default:
# pid_filename /var/run/squid.pid

#  TAG: debug_options
#               Logging options are set as section,level where each source file
#               is assigned a unique section.  Lower levels result in less
#               output,  Full debugging (level 9) can result in a very large
#               log file, so be careful.  The magic word "ALL" sets debugging
#               levels for all sections.  We recommend normally running with
#               "ALL,1".
#
#Default:
# debug_options ALL,1

#  TAG: log_fqdn            on|off
#               Turn this on if you wish to log fully qualified domain names
#               in the access.log. To do this Squid does a DNS lookup of all
#               IP's connecting to it. This can (in some situations) increase
#               latency, which makes your cache seem slower for interactive
#               browsing.
#
#Default:
# log_fqdn off

#  TAG: client_netmask
#               A netmask for client addresses in logfiles and cachemgr output.
#               Change this to protect the privacy of your cache clients.
#               A netmask of 255.255.255.0 will log all IP's in that range with
#               the last digit set to '0'.
#
#Default:
# client_netmask 255.255.255.255


# OPTIONS FOR EXTERNAL SUPPORT PROGRAMS
#
-----------------------------------------------------------------------------

#  TAG: ftp_user
#               If you want the anonymous login password to be more informative
#               (and enable the use of picky ftp servers), set this to something
#               reasonable for your domain, like wwwuser@somewhere.net
#
#               The reason why this is domainless by default is the
#               request can be made on the behalf of a user in any domain,
#               depending on how the cache is used.
#               Some ftp server also validate the email address is valid
#               (for example perl.com).
#
#Default:
# ftp_user Squid@

#  TAG: ftp_list_width
#               Sets the width of ftp listings. This should be set to fit in
#               the width of a standard browser. Setting this too small
#               can cut off long filenames when browsing ftp sites.
#
#Default:
# ftp_list_width 32

#  TAG: ftp_passive
#               If your firewall does not allow Squid to use passive
```

```
#               connections, turn off this option.
#
#Default:
# ftp_passive on

#  TAG: ftp_sanitycheck
#               For security and data integrity reasons Squid by default performs
#               sanity checks of the addresses of FTP data connections ensure the
#               data connection is to the requested server. If you need to allow
#               FTP connections to servers using another IP address for the data
#               connection turn this off.
#
#Default:
# ftp_sanitycheck on

#  TAG: ftp_telnet_protocol
#               The FTP protocol is officially defined to use the telnet protocol
#               as transport channel for the control connection. However, many
#               implemenations are broken and does not respect this aspect of
#               the FTP protocol.
#
#               If you have trouble accessing files with ASCII code 255 in the
#               path or similar problems involving this ASCII code you can
#               try setting this directive to off. If that helps, report to the
#               operator of the FTP server in question that their FTP server
#               is broken and does not follow the FTP standard.
#
#Default:
# ftp_telnet_protocol on

#  TAG: cache_dns_program
# Note: This option is only available if Squid is rebuilt with the
#       --disable-internal-dns option
#
#               Specify the location of the executable for dnslookup process.
#
#Default:
# cache_dns_program /usr/lib/squid/dnsserver

#  TAG: dns_children
# Note: This option is only available if Squid is rebuilt with the
#       --disable-internal-dns option
#
#               The number of processes spawn to service DNS name lookups.
#               For heavily loaded caches on large servers, you should
#               probably increase this value to at least 10.  The maximum
#               is 32.  The default is 5.
#
#               You must have at least one dnsserver process.
#
#Default:
# dns_children 5

#  TAG: dns_retransmit_interval
#               Initial retransmit interval for DNS queries. The interval is
#               doubled each time all configured DNS servers have been tried.
#
#
#Default:
# dns_retransmit_interval 5 seconds

#  TAG: dns_timeout
#               DNS Query timeout. If no response is received to a DNS query
#               within this time all DNS servers for the queried domain
#               are assumed to be unavailable.
#
#Default:
# dns_timeout 2 minutes

#  TAG: dns_defnames    on|off
# Note: This option is only available if Squid is rebuilt with the
#       --disable-internal-dns option
#
```

```
#               Normally the 'dnsserver' disables the RES_DEFNAMES resolver
#               option (see res_init(3)).  This prevents caches in a hierarchy
#               from interpreting single-component hostnames locally.  To allow
#               dnsserver to handle single-component names, enable this
#               option.
#
#Default:
# dns_defnames off

#  TAG: dns_nameservers
#               Use this if you want to specify a list of DNS name servers
#               (IP addresses) to use instead of those given in your
#               /etc/resolv.conf file.
#               On Windows platforms, if no value is specified here or in
#               the /etc/resolv.conf file, the list of DNS name servers are
#               taken from the Windows registry, both static and dynamic DHCP
#               configurations are supported.
#
#               Example: dns_nameservers 10.0.0.1 192.172.0.4
#
#Default:
# none

#  TAG: hosts_file
#               Location of the host-local IP name-address associations
#               database.  Most Operating Systems have such a file: under
#               Un*X it's by default in /etc/hosts.  MS-Windows NT/2000 places
#               it in %SystemRoot%(by default
#               c:\winnt)\system32\drivers\etc\hosts, while Windows 9x/ME
#               places it in %windir%(usually c:\windows)\hosts
#
#               The file contains newline-separated definitions, in the
#               form ip_address_in_dotted_form name [name ...] names are
#               whitespace-separated.  lines beginnng with an hash (#)
#               character are comments.
#
#               The file is checked at startup and upon configuration.  If
#               set to 'none', it won't be checked.  If append_domain is
#               used, that domain will be added to domain-local (i.e. not
#               containing any dot character) host definitions.
#
#Default:
# hosts_file /etc/hosts

#  TAG: diskd_program
#               Specify the location of the diskd executable.
#               Note that this is only useful if you have compiled in
#               diskd as one of the store io modules.
#
#Default:
# diskd_program /usr/lib/squid/diskd

#  TAG: unlinkd_program
#               Specify the location of the executable for file deletion process.
#
#Default:
# unlinkd_program /usr/lib/squid/unlinkd

#  TAG: pinger_program
# Note: This option is only available if Squid is rebuilt with the
#       --enable-icmp option
#
#               Specify the location of the executable for the pinger process.
#
#Default:
# pinger_program /usr/lib/squid/pinger

#  TAG: redirect_program
#               Specify the location of the executable for the URL redirector.
#               Since they can perform almost any function there isn't one included.
#               See the FAQ (section 15) for information on how to write one.
#               By default, a redirector is not used.
#
```

```
#Default:
# none

#  TAG: redirect_children
#               The number of redirector processes to spawn. If you start
#               too few Squid will have to wait for them to process a backlog of
#               URLs, slowing it down. If you start too many they will use RAM
#               and other system resources.
#
#Default:
# redirect_children 5

#  TAG: redirect_rewrites_host_header
#               By default Squid rewrites any Host: header in redirected
#               requests.  If you are running an accelerator this may
#               not be a wanted effect of a redirector.
#
#Default:
# redirect_rewrites_host_header on

#  TAG: redirector_access
#               If defined, this access list specifies which requests are
#               sent to the redirector processes.  By default all requests
#               are sent.
#
#Default:
# none

#  TAG: auth_param
#               This is used to define parameters for the various authentication
#               schemes supported by Squid.
#
#               format: auth_param scheme parameter [setting]
#
#               The order in which authentication schemes are presented to the client is
#               dependant on the order the scheme first appears in config file. IE
#               has a bug (it's not rfc 2617 compliant) in that it will use the basic
#               scheme if basic is the first entry presented, even if more secure
#               schemes are presented. For now use the order in the recommended
#               settings section below. If other browsers have difficulties (don't
#               recognise the schemes offered even if you are using basic) either
#               put basic first, or disable the other schemes (by commenting out their
#               program entry).
#
#               Once an authentication scheme is fully configured, it can only be
#               shutdown by shutting squid down and restarting. Changes can be made on
#               the fly and activated with a reconfigure. I.E. You can change to a
#               different helper, but not unconfigure the helper completely.
#
#               Please note that while this directive defines how Squid processes
#               authentication it does not automatically activate authentication.
#               To use authenticaiton you must in addition make use of acls based
#               on login name in http_access (proxy_auth, proxy_auth_regex or
#               external with %LOGIN used in the format tag). The browser will be
#               challenged for authentication on the first such acl encountered
#               in http_access processing and will also be rechallenged for new
#               login credentials if the request is being denied by a proxy_auth
#               type acl.
#
#               === Parameters for the basic scheme follow. ===
#
#               "program" cmdline
#               Specify the command for the external authenticator.  Such a program
#               reads a line containing "username password" and replies "OK" or
#               "ERR" in an endless loop. "ERR" responses may optionally be followed
#               by a error description available as %m in the returned error page.
#
#               By default, the basic authentication sheme is not used unless a
#               program is specified.
#
#               If you want to use the traditional proxy authentication, jump over to
#               the helpers/basic_auth/NCSA directory and type:
#                               % make
```

```
#                       % make install
#
#       Then, set this line to something like
#
#       auth_param basic program /usr/libexec/ncsa_auth /usr/etc/passwd
#
#       "children" numberofchildren
#       The number of authenticator processes to spawn.
#       If you start too few Squid will have to wait for them to process a
#       backlog of usercode/password verifications, slowing it down. When
#       password verifications are done via a (slow) network you are likely to
#       need lots of authenticator processes.
#       auth_param basic children 5
#
#       "realm" realmstring
#       Specifies the realm name which is to be reported to the client for
#       the basic proxy authentication scheme (part of the text the user
#       will see when prompted their username and password).
#       auth_param basic realm Squid proxy-caching web server
#
#       "credentialsttl" timetolive
#       Specifies how long squid assumes an externally validated
#       username:password pair is valid for - in other words how often the
#       helper program is called for that user. Set this low to force
#       revalidation with short lived passwords.  Note that setting this high
#       does not impact your susceptability to replay attacks unless you are
#       using an one-time password system (such as SecureID). If you are using
#       such a system, you will be vulnerable to replay attacks unless you
#       also use the max_user_ip ACL in an http_access rule.
#       auth_param basic credentialsttl 2 hours
#
#       "casesensitive" on|off
#       Specifies if usernames are case sensitive. Most user databases are
#       case insensitive allowing the same username to be spelled using both
#       lower and upper case letters, but some are case sensitive. This
#       makes a big difference for user_max_ip ACL processing and similar.
#       auth_param basic casesensitive off
#
#       === Parameters for the digest scheme follow ===
#
#       "program" cmdline
#       Specify the command for the external authenticator.  Such a program
#       reads a line containing "username":"realm" and replies with the
#       appropriate H(A1) value base64 encoded or ERR if the user (or his H(A1)
#       hash) does not exists.  See rfc 2616 for the definition of H(A1).
#       "ERR" responses may optionally be followed by a error description
#       available as %m in the returned error page.
#
#       By default, the digest authentication scheme is not used unless a
#       program is specified.
#
#       If you want to use a digest authenticator, jump over to the
#       helpers/digest_auth/ directory and choose the authenticator to use.
#       It it's directory type
#       % make
#             % make install
#
#       Then, set this line to something like
#
#       auth_param digest program /usr/libexec/digest_auth_pw /usr/etc/digpass
#
#
#       "children" numberofchildren
#       The number of authenticator processes to spawn (no default). If you
#       start too few Squid will have to wait for them to process a backlog of
#       H(A1) calculations, slowing it down.  When the H(A1) calculations are
#       done via a (slow) network you are likely to need lots of authenticator
#       processes.
#       auth_param digest children 5
#
#       "realm" realmstring
#       Specifies the realm name which is to be reported to the client for the
#       digest proxy authentication scheme (part of the text the user will see
```

```
#		when prompted their username and password).
#		auth_param digest realm Squid proxy-caching web server
#
#		"nonce_garbage_interval" timeinterval
#		Specifies the interval that nonces that have been issued to clients are
#		checked for validity.
#		auth_param digest nonce_garbage_interval 5 minutes
#
#		"nonce_max_duration" timeinterval
#		Specifies the maximum length of time a given nonce will be valid for.
#		auth_param digest nonce_max_duration 30 minutes
#
#		"nonce_max_count" number
#		Specifies the maximum number of times a given nonce can be used.
#		auth_param digest nonce_max_count 50
#
#		"nonce_strictness" on|off
#		Determines if squid requires strict increment-by-1 behaviour for nonce
#		counts, or just incrementing (off - for use when useragents generate
#		nonce counts that occasionally miss 1 (ie, 1,2,4,6)).
#		auth_param digest nonce_strictness off
#
#		"check_nonce_count" on|off
#		This directive if set to off can disable the nonce count check
#		completely to work around buggy digest qop implementations in certain
#		mainstream browser versions. Default on to check the nonce count to
#		protect from authentication replay attacks.
#		auth_param digest check_nonce_count on
#
#		"post_workaround" on|off
#		This is a workaround to certain buggy browsers who sends an incorrect
#		request digest in POST requests when reusing the same nonce as aquired
#		earlier in response to a GET request.
#		auth_param digest post_workaround off
#
#		=== NTLM scheme options follow ===
#
#		"program" cmdline
#		Specify the command for the external ntlm authenticator. Such a
#		program participates in the NTLMSSP exchanges between Squid and the
#		client and reads commands according to the Squid ntlmssp helper
#		protocol. See helpers/ntlm_auth/ for details. Recommended ntlm
#		authenticator is ntlm_auth from Samba-3.X, but a number of other
#		ntlm authenticators is available.
#
#		By default, the ntlm authentication scheme is not used unless a
#		program is specified.
#
#		auth_param ntlm program /path/to/samba/bin/ntlm_auth
--helper-protocol=squid-2.5-ntlmssp
#
#		"children" numberofchildren
#		The number of authenticator processes to spawn (no default). If you
#		start too few Squid will have to wait for them to process a backlog
#		of credential verifications, slowing it down. When crendential
#		verifications are done via a (slow) network you are likely to need
#		lots of authenticator processes.
#		auth_param ntlm children 5
#
#		"max_challenge_reuses" number
#		The maximum number of times a challenge given by a ntlm authentication
#		helper can be reused. Increasing this number increases your exposure
#		to replay attacks on your network. 0 (the default) means use the
#		challenge is used only once. See also the max_ntlm_challenge_lifetime
#		directive if enabling challenge reuses.
#		auth_param ntlm max_challenge_reuses 0
#
#		"max_challenge_lifetime" timespan
#		The maximum time period a ntlm challenge is reused over. The
#		actual period will be the minimum of this time AND the number of
#		reused challenges.
#		auth_param ntlm max_challenge_lifetime 2 minutes
#
```

```
#		"use_ntlm_negotiate" on|off
#		Enables support for NTLM NEGOTIATE packet exchanges with the helper.
#		The configured ntlm authenticator must be able to handle NTLM
#		NEGOTIATE packet. See the authenticator programs documentation if
#		unsure. ntlm_auth from Samba-3.0.2 or later supports the use of this
#		option.
#		The NEGOTIATE packet is required to support NTLMv2 and a
#		number of other negotiable NTLMSSP options, and also makes it
#		more likely the negotiation is successful. Enabling this parameter
#		will also solve problems encountered when NT domain policies
#		restrict users to access only certain workstations. When this is off,
#		all users must be allowed to log on the proxy servers too, or they'll
#		get "invalid workstation" errors - and access denied - when trying to
#		use Squid's services.
#		Use of ntlm NEGOTIATE is incompatible with challenge reuse, so
#		enabling this parameter will OVERRIDE the max_challenge_reuses and
#		max_challenge_lifetime parameters and set them to 0.
#		auth_param ntlm use_ntlm_negotiate off
#
#Recommended minimum configuration:
#auth_param digest program <uncomment and complete this line>
#auth_param digest children 5
#auth_param digest realm Squid proxy-caching web server
#auth_param digest nonce_garbage_interval 5 minutes
#auth_param digest nonce_max_duration 30 minutes
#auth_param digest nonce_max_count 50
#auth_param ntlm program <uncomment and complete this line to activate>
#auth_param ntlm children 5
#auth_param ntlm max_challenge_reuses 0
#auth_param ntlm max_challenge_lifetime 2 minutes
#auth_param ntlm use_ntlm_negotiate off
#auth_param basic program <uncomment and complete this line>
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 2 hours
auth_param basic casesensitive off

#  TAG: authenticate_cache_garbage_interval
#		The time period between garbage collection across the username cache.
#		This is a tradeoff between memory utilisation (long intervals - say
#		2 days) and CPU (short intervals - say 1 minute). Only change if you
#		have good reason to.
#
#Default:
# authenticate_cache_garbage_interval 1 hour

#  TAG: authenticate_ttl
#		The time a user & their credentials stay in the logged in user cache
#		since their last request. When the garbage interval passes, all user
#		credentials that have passed their TTL are removed from memory.
#
#Default:
# authenticate_ttl 1 hour

#  TAG: authenticate_ip_ttl
#		If you use proxy authentication and the 'max_user_ip' ACL, this
#		directive controls how long Squid remembers the IP addresses
#		associated with each user.  Use a small value (e.g., 60 seconds) if
#		your users might change addresses quickly, as is the case with
#		dialups. You might be safe using a larger value (e.g., 2 hours) in a
#		corporate LAN environment with relatively static address assignments.
#
#Default:
# authenticate_ip_ttl 0 seconds

#  TAG: external_acl_type
#		This option defines external acl classes using a helper program to
#		look up the status
#
#		  external_acl_type name [options] FORMAT.. /path/to/helper [helper
arguments..]
#
#		Options:
```

```
#
#		ttl=n			TTL in seconds for cached results (defaults to 3600
#					for 1 hour)
#		negative_ttl=n
#					TTL for cached negative lookups (default same
#					as ttl)
#		children=n	Concurrency level / number of processes spawn
#					to service external acl lookups of this type.
#					Note: see compatibility note below
#		cache=n		result cache size, 0 is unbounded (default)
#		protocol=3.0		Use URL-escaped strings instead of quoting
#
#	FORMAT specifications
#
#		%LOGIN	Authenticated user login name
#		%IDENT	Ident user name
#		%SRC			Client IP
#		%DST			Requested host
#		%PROTO	Requested protocol
#		%PORT			Requested port
#		%METHOD			Request method
#		%{Header}	HTTP request header
#		%{Hdr:member}		HTTP request header list member
#		%{Hdr:;member}
#					HTTP request header list member using ; as
#					list separator. ; can be any non-alphanumeric
#					character.
#
#	In addition, any string specified in the referencing acl will
#	also be included in the helper request line, after the specified
#	formats (see the "acl external" directive)
#
#	The helper receives lines per the above format specification,
#	and returns lines starting with OK or ERR indicating the validity
#	of the request and optionally followed by additional keywords with
#	more details.
#
#	General result syntax:
#
#	 OK/ERR keyword=value ...
#
#	Defined keywords:
#
#		user=			The users name (login)
#		error=		Error description (only defined for ERR results)
#
#	Keyword values need to be enclosed in quotes if they may contain
#	whitespace, or the whitespace escaped using \. Any quotes or \
#	characters within the keyword value must be \ escaped.
#
#	If protocol=3.0 then URL escaping of the strings is used instead
#	of the above described quoting format.
#
#	Compatibility Note: The children= option was named concurrency= in
#	Squid-2.5.STABLE3 and earlier and such syntax is still accepted to
#	keep compatibility within the Squid-2.5 release. However, the meaning
#	of concurrency= option has changed in Squid-3 and the old syntax of
#	the directive is therefore depreated from Squid-2.5.STABLE4 and later.
#	If you want to be able to easily downgrade to earlier Squid-2.5
#	releases you may want to continue using the old name, if not
#	please use the new name.
#
#Default:
# none


# OPTIONS FOR TUNING THE CACHE
#
-----------------------------------------------------------------------------

#  TAG: wais_relay_host
#  TAG: wais_relay_port
#	Relay WAIS request to host (1st arg) at port (2 arg).
```

```
#
#Default:
# wais_relay_port 0

#  TAG: request_header_max_size      (KB)
#               This specifies the maximum size for HTTP headers in a request.
#               Request headers are usually relatively small (about 512 bytes).
#               Placing a limit on the request header size will catch certain
#               bugs (for example with persistent connections) and possibly
#               buffer-overflow or denial-of-service attacks.
#
#Default:
# request_header_max_size 20 KB

#  TAG: request_body_max_size        (KB)
#               This specifies the maximum size for an HTTP request body.
#               In other words, the maximum size of a PUT/POST request.
#               A user who attempts to send a request with a body larger
#               than this limit receives an "Invalid Request" error message.
#               If you set this parameter to a zero (the default), there will
#               be no limit imposed.
#
#Default:
# request_body_max_size 0 KB

#  TAG: refresh_pattern
#               usage: refresh_pattern [-i] regex min percent max [options]
#
#               By default, regular expressions are CASE-SENSITIVE.  To make
#               them case-insensitive, use the -i option.
#
#               'Min' is the time (in minutes) an object without an explicit
#               expiry time should be considered fresh. The recommended
#               value is 0, any higher values may cause dynamic applications
#               to be erroneously cached unless the application designer
#               has taken the appropriate actions.
#
#               'Percent' is a percentage of the objects age (time since last
#               modification age) an object without explicit expiry time
#               will be considered fresh.
#
#               'Max' is an upper limit on how long objects without an explicit
#               expiry time will be considered fresh.
#
#               options: override-expire
#                                override-lastmod
#                                reload-into-ims
#                                ignore-reload
#
#                               override-expire enforces min age even if the server
#                               sent a Expires: header. Doing this VIOLATES the HTTP
#                               standard.  Enabling this feature could make you liable
#                               for problems which it causes.
#
#                               override-lastmod enforces min age even on objects
#                               that were modified recently.
#
#                               reload-into-ims changes client no-cache or ``reload''
#                               to If-Modified-Since requests. Doing this VIOLATES the
#                               HTTP standard. Enabling this feature could make you
#                               liable for problems which it causes.
#
#                               ignore-reload ignores a client no-cache or ``reload''
#                               header. Doing this VIOLATES the HTTP standard. Enabling
#                               this feature could make you liable for problems which
#                               it causes.
#
#               Basically a cached object is:
#
#                               FRESH if expires < now, else STALE
#                               STALE if age > max
#                               FRESH if lm-factor < percent, else STALE
#                               FRESH if age < min
```

```
#                               else STALE
#
#               The refresh_pattern lines are checked in the order listed here.
#               The first entry which matches is used.  If none of the entries
#               match the default will be used.
#
#               Note, you must uncomment all the default lines if you want
#               to change one. The default setting is only active if none is
#               used.
#
#Suggested default:
refresh_pattern ^ftp:                   1440            20%             10080
refresh_pattern ^gopher:  1440          0%              1440
refresh_pattern .                       0               20%             4320

#  TAG: quick_abort_min (KB)
#  TAG: quick_abort_max (KB)
#  TAG: quick_abort_pct  (percent)
#               The cache by default continues downloading aborted requests
#               which are almost completed (less than 16 KB remaining). This
#               may be undesirable on slow (e.g. SLIP) links and/or very busy
#               caches.  Impatient users may tie up file descriptors and
#               bandwidth by repeatedly requesting and immediately aborting
#               downloads.
#
#               When the user aborts a request, Squid will check the
#               quick_abort values to the amount of data transfered until
#               then.
#
#               If the transfer has less than 'quick_abort_min' KB remaining,
#               it will finish the retrieval.
#
#               If the transfer has more than 'quick_abort_max' KB remaining,
#               it will abort the retrieval.
#
#               If more than 'quick_abort_pct' of the transfer has completed,
#               it will finish the retrieval.
#
#               If you do not want any retrieval to continue after the client
#               has aborted, set both 'quick_abort_min' and 'quick_abort_max'
#               to '0 KB'.
#
#               If you want retrievals to always continue if they are being
#               cached set 'quick_abort_min' to '-1 KB'.
#
#Default:
# quick_abort_min 16 KB
# quick_abort_max 16 KB
# quick_abort_pct 95

#  TAG: negative_ttl        time-units
#               Time-to-Live (TTL) for failed requests.  Certain types of
#               failures (such as "connection refused" and "404 Not Found") are
#               negatively-cached for a configurable amount of time.  The
#               default is 5 minutes.  Note that this is different from
#               negative caching of DNS lookups.
#
#Default:
# negative_ttl 5 minutes

#  TAG: positive_dns_ttl   time-units
#               Upper limit on how long Squid will cache positive DNS responses.
#               Default is 6 hours (360 minutes). This directive must be set
#               larger than negative_dns_ttl.
#
#Default:
# positive_dns_ttl 6 hours

#  TAG: negative_dns_ttl  time-units
#               Time-to-Live (TTL) for negative caching of failed DNS lookups.
#               This also makes sets the lower cache limit on positive lookups.
#               Minimum value is 1 second, and it is not recommendable to go
#               much below 10 seconds.
```

```
#
#Default:
# negative_dns_ttl 1 minute

#  TAG: range_offset_limit                  (bytes)
#               Sets a upper limit on how far into the the file a Range request
#               may be to cause Squid to prefetch the whole file. If beyond this
#               limit Squid forwards the Range request as it is and the result
#               is NOT cached.
#
#               This is to stop a far ahead range request (lets say start at 17MB)
#               from making Squid fetch the whole object up to that point before
#               sending anything to the client.
#
#               A value of -1 causes Squid to always fetch the object from the
#               beginning so it may cache the result. (2.0 style)
#
#               A value of 0 causes Squid to never fetch more than the
#               client requested. (default)
#
#Default:
# range_offset_limit 0 KB


# TIMEOUTS
#
-----------------------------------------------------------------------------

#  TAG: forward_timeout  time-units
#               This parameter specifies how long Squid should at most attempt in
#               finding a forwarding path for the request before giving up.
#
#Default:
# forward_timeout 4 minutes

#  TAG: connect_timeout time-units
#               This parameter specifies how long to wait for the TCP connect to
#               the requested server or peer to complete before Squid should
#               attempt to find another path where to forward the request.
#
#Default:
# connect_timeout 1 minute

#  TAG: peer_connect_timeout            time-units
#               This parameter specifies how long to wait for a pending TCP
#               connection to a peer cache.  The default is 30 seconds.   You
#               may also set different timeout values for individual neighbors
#               with the 'connect-timeout' option on a 'cache_peer' line.
#
#Default:
# peer_connect_timeout 30 seconds

#  TAG: read_timeout       time-units
#               The read_timeout is applied on server-side connections.  After
#               each successful read(), the timeout will be extended by this
#               amount.  If no data is read again after this amount of time,
#               the request is aborted and logged with ERR_READ_TIMEOUT.  The
#               default is 15 minutes.
#
#Default:
# read_timeout 15 minutes

#  TAG: request_timeout
#               How long to wait for an HTTP request after initial
#               connection establishment.
#
#Default:
# request_timeout 5 minutes

#  TAG: persistent_request_timeout
#               How long to wait for the next HTTP request on a persistent
#               connection after the previous request completes.
#
```

```
#Default:
# persistent_request_timeout 1 minute

#  TAG: client_lifetime      time-units
#               The maximum amount of time a client (browser) is allowed to
#               remain connected to the cache process.  This protects the Cache
#               from having a lot of sockets (and hence file descriptors) tied up
#               in a CLOSE_WAIT state from remote clients that go away without
#               properly shutting down (either because of a network failure or
#               because of a poor client implementation).  The default is one
#               day, 1440 minutes.
#
#               NOTE:  The default value is intended to be much larger than any
#               client would ever need to be connected to your cache.  You
#               should probably change client_lifetime only as a last resort.
#               If you seem to have many client connections tying up
#               filedescriptors, we recommend first tuning the read_timeout,
#               request_timeout, persistent_request_timeout and quick_abort values.
#
#Default:
# client_lifetime 1 day

#  TAG: half_closed_clients
#               Some clients may shutdown the sending side of their TCP
#               connections, while leaving their receiving sides open.                Sometimes,
#               Squid can not tell the difference between a half-closed and a
#               fully-closed TCP connection.  By default, half-closed client
#               connections are kept open until a read(2) or write(2) on the
#               socket returns an error.  Change this option to 'off' and Squid
#               will immediately close client connections when read(2) returns
#               "no more data to read."
#
#Default:
# half_closed_clients on

#  TAG: pconn_timeout
#               Timeout for idle persistent connections to servers and other
#               proxies.
#
#Default:
# pconn_timeout 120 seconds

#  TAG: ident_timeout
# Note: This option is only available if Squid is rebuilt with the
#       --enable-ident-lookups option
#
#               Maximum time to wait for IDENT lookups to complete.
#
#               If this is too high, and you enabled IDENT lookups from untrusted
#               users, you might be susceptible to denial-of-service by having
#               many ident requests going at once.
#
#Default:
# ident_timeout 10 seconds

#  TAG: shutdown_lifetime                time-units
#               When SIGTERM or SIGHUP is received, the cache is put into
#               "shutdown pending" mode until all active sockets are closed.
#               This value is the lifetime to set for all open descriptors
#               during shutdown mode.  Any active clients after this many
#               seconds will receive a 'timeout' message.
#
#Default:
# shutdown_lifetime 30 seconds


# ACCESS CONTROLS
#
-----------------------------------------------------------------------------

#  TAG: acl
#               Defining an Access List
#
```

```
#	acl aclname acltype string1 ...
#	acl aclname acltype "file" ...
#
#	when using "file", the file should contain one item per line
#
#	acltype is one of the types described below
#
#	By default, regular expressions are CASE-SENSITIVE.  To make
#	them case-insensitive, use the -i option.
#
#	acl aclname src	ip-address/netmask ... (clients IP address)
#	acl aclname src	addr1-addr2/netmask ... (range of addresses)
#	acl aclname dst	ip-address/netmask ... (URL host's IP address)
#	acl aclname myip	ip-address/netmask ... (local socket IP address)
#
#	acl aclname arp	mac-address ... (xx:xx:xx:xx:xx:xx notation)
#	  # The arp ACL requires the special configure option --enable-arp-acl.
#	  # Furthermore, the arp ACL code is not portable to all operating
systems.
#	  # It works on Linux, Solaris, FreeBSD and some other *BSD variants.
#	  #
#	  # NOTE: Squid can only determine the MAC address for clients that are on
#	  # the same subnet. If the client is on a different subnet, then Squid
cannot
#	  # find out its MAC address.
#
#	acl aclname srcdomain	.foo.com ...	# reverse lookup, client IP
#	acl aclname dstdomain	.foo.com ...	# Destination server from URL
#	acl aclname srcdom_regex [-i] xxx ...	# regex matching client name
#	acl aclname dstdom_regex [-i] xxx ...	# regex matching server
#	  # For dstdomain and dstdom_regex  a reverse lookup is tried if a IP
#	  # based URL is used and no match is found. The name "none" is used
#	  # if the reverse lookup fails.
#
#	acl aclname time	[day-abbrevs]  [h1:m1-h2:m2]
#	   day-abbrevs:
#			S - Sunday
#			M - Monday
#			T - Tuesday
#			W - Wednesday
#			H - Thursday
#			F - Friday
#			A - Saturday
#	   h1:m1 must be less than h2:m2
#	acl aclname url_regex [-i] ^http:// ...	# regex matching on whole URL
#	acl aclname urlpath_regex [-i] \.gif$ ...	# regex matching on URL path
#	acl aclname urllogin [-i] [^a-zA-Z0-9] ...# regex matching on URL login
field
#	acl aclname port	80 70 21 ...
#	acl aclname port	0-1024 ...			# ranges allowed
#	acl aclname myport	3128 ...			# (local socket TCP port)
#	acl aclname proto	HTTP FTP ...
#	acl aclname method	GET POST ...
#	acl aclname browser	[-i] regexp ...
#	  # pattern match on User-Agent header (see also req_header below)
#	acl aclname referer_regex	[-i] regexp ...
#	  # pattern match on Referer header
#	  # Referer is highly unreliable, so use with care
#	acl aclname ident	username ...
#	acl aclname ident_regex [-i] pattern ...
#	  # string match on ident output.
#	  # use REQUIRED to accept any non-null ident.
#	acl aclname src_as	number ...
#	acl aclname dst_as	number ...
#	  # Except for access control, AS numbers can be used for
#	  # routing of requests to specific caches. Here's an
#	  # example for routing all requests for AS#1241 and only
#	  # those to mycache.mydomain.net:
#	  # acl asexample dst_as 1241
#	  # cache_peer_access mycache.mydomain.net allow asexample
#	  # cache_peer_access mycache_mydomain.net deny all
#
#	acl aclname proxy_auth username ...
```

```
#		acl aclname proxy_auth_regex [-i] pattern ...
#		  # list of valid usernames
#		  # use REQUIRED to accept any valid username.
#		  #
#		  # NOTE: when a Proxy-Authentication header is sent but it is not
#		  # needed during ACL checking the username is NOT logged
#		  # in access.log.
#		  #
#		  # NOTE: proxy_auth requires a EXTERNAL authentication program
#		  # to check username/password combinations (see
#		  # auth_param directive).
#		  #
#		  # WARNING: proxy_auth can't be used in a transparent proxy. It
#		  # collides with any authentication done by origin servers. It may
#		  # seem like it works at first, but it doesn't.
#
#		acl aclname snmp_community string ...
#		  # A community string to limit access to your SNMP Agent
#		  # Example:
#		  #
#		  #	acl snmppublic snmp_community public
#
#		acl aclname maxconn number
#		  # This will be matched when the client's IP address has
#		  # more than <number> HTTP connections established.
#
#		acl aclname max_user_ip [-s] number
#		  # This will be matched when the user attempts to log in from more
#		  # than <number> different ip addresses. The authenticate_ip_ttl
#		  # parameter controls the timeout on the ip entries.
#		  # If -s is specified the limit is strict, denying browsing
#		  # from any further IP addresses until the ttl has expired. Without
#		  # -s Squid will just annoy the user by "randomly" denying requests.
#		  # (the counter is reset each time the limit is reached and a
#		  # request is denied)
#		  # NOTE: in acceleration mode or where there is mesh of child proxies,
#		  # clients may appear to come from multiple addresses if they are
#		  # going through proxy farms, so a limit of 1 may cause user problems.
#
#		acl aclname req_mime_type mime-type1 ...
#		  # regex match against the mime type of the request generated
#		  # by the client. Can be used to detect file upload or some
#		  # types HTTP tunelling requests.
#		  # NOTE: This does NOT match the reply. You cannot use this
#		  # to match the returned file type.
#
#		acl aclname req_header header-name [-i] any\.regex\.here
#		  # regex match against any of the known request headers.  May be
#		  # thought of as a superset of "browser", "referer" and "mime-type"
#		  # acls.
#
#		acl aclname rep_mime_type mime-type1 ...
#		  # regex match against the mime type of the reply recieved by
#		  # squid. Can be used to detect file download or some
#		  # types HTTP tunelling requests.
#		  # NOTE: This has no effect in http_access rules. It only has
#		  # effect in rules that affect the reply data stream such as
#		  # http_reply_access.
#
#		acl aclname rep_header header-name [-i] any\.regex\.here
#		  # regex match against any of the known response headers.
#		  # Example:
#		  #
#		  # acl many_spaces rep_header Content-Disposition -i [[:space:]]{3,}
#
#		acl acl_name external class_name [arguments...]
#		  # external ACL lookup via a helper class defined by the
#		  # external_acl_type directive.
#
#Examples:
#acl macaddress arp 09:00:2b:23:45:67
#acl myexample dst_as 1241
#acl password proxy_auth REQUIRED
```

```
#acl fileupload req_mime_type -i ^multipart/form-data$
#acl javascript rep_mime_type -i ^application/x-javascript$
#
#Recommended minimum configuration:
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
acl SSL_ports port 443 563
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443 563     # https, snews
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # unregistered ports
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
acl CONNECT method CONNECT

#  TAG: http_access
#               Allowing or Denying access based on defined access lists
#
#               Access to the HTTP port:
#               http_access allow|deny [!]aclname ...
#
#               NOTE on default values:
#
#               If there are no "access" lines present, the default is to deny
#               the request.
#
#               If none of the "access" lines cause a match, the default is the
#               opposite of the last line in the list.  If the last line was
#               deny, the default is allow.  Conversely, if the last line
#               is allow, the default will be deny.  For these reasons, it is a
#               good idea to have an "deny all" or "allow all" entry at the end
#               of your access lists to avoid potential confusion.
#
#Default:
# http_access deny all
#
#Recommended minimum configuration:
#
# Only allow cachemgr access from localhost
http_access allow manager localhost
http_access deny manager
# Deny requests to unknown ports
http_access deny !Safe_ports
# Deny CONNECT to other than SSL ports
http_access deny CONNECT !SSL_ports
#
# We strongly recommend the following be uncommented to protect innocent
# web applications running on the proxy server who think the only
# one who can access services on "localhost" is a local user
#http_access deny to_localhost
#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS

# Example rule allowing access from your local networks. Adapt
# to list your (internal) IP networks from where browsing should
# be allowed
#acl our_networks src 192.168.1.0/24 192.168.2.0/24
#http_access allow our_networks

# And finally deny all other access to this proxy
http_access allow localhost
http_access deny all

#  TAG: http_reply_access
#       Allow replies to client requests. This is complementary to
http_access.
#
```

```
#       http_reply_access allow|deny [!] aclname ...
#
#       NOTE: if there are no access lines present, the default is to allow
#             all replies
#
#       If none of the access lines cause a match the opposite of the
#       last line will apply. Thus it is good practice to end the rules
#       with an "allow all" or "deny all" entry.
#
#Default:
# http_reply_access allow all
#
#Recommended minimum configuration:
#
# Insert your own rules here.
#
#
# and finally allow by default
http_reply_access allow all

#  TAG: icp_access
#             Allowing or Denying access to the ICP port based on defined
#             access lists
#
#             icp_access  allow|deny [!]aclname ...
#
#             See http_access for details
#
#Default:
# icp_access deny all
#
#Allow ICP queries from everyone
icp_access allow all

#  TAG: miss_access
#             Use to force your neighbors to use you as a sibling instead of
#             a parent.  For example:
#
#                          acl localclients src 172.16.0.0/16
#                          miss_access allow localclients
#                          miss_access deny  !localclients
#
#             This means only your local clients are allowed to fetch
#             MISSES and all other clients can only fetch HITS.
#
#             By default, allow all clients who passed the http_access rules
#             to fetch MISSES from us.
#
#Default setting:
# miss_access allow all

#  TAG: cache_peer_access
#             Similar to 'cache_peer_domain' but provides more flexibility by
#             using ACL elements.
#
#             cache_peer_access cache-host allow|deny [!]aclname ...
#
#             The syntax is identical to 'http_access' and the other lists of
#             ACL elements.  See the comments for 'http_access' below, or
#             the Squid FAQ (http://www.squid-cache.org/FAQ/FAQ-10.html).
#
#Default:
# none

#  TAG: ident_lookup_access
# Note: This option is only available if Squid is rebuilt with the
#       --enable-ident-lookups option
#
#             A list of ACL elements which, if matched, cause an ident
#             (RFC 931) lookup to be performed for this request.  For
#             example, you might choose to always perform ident lookups
#             for your main multi-user Unix boxes, but not for your Macs
#             and PCs.  By default, ident lookups are not performed for
```

```
#               any requests.
#
#               To enable ident lookups for specific client addresses, you
#               can follow this example:
#
#               acl ident_aware_hosts src 198.168.1.0/255.255.255.0
#               ident_lookup_access allow ident_aware_hosts
#               ident_lookup_access deny all
#
#               Only src type ACL checks are fully supported.  A src_domain
#               ACL might work at times, but it will not always provide
#               the correct result.
#
#Default:
# ident_lookup_access deny all

#  TAG: tcp_outgoing_tos
#               Allows you to select a TOS/Diffserv value to mark outgoing
#               connections with, based on the username or source address
#               making the request.
#
#               tcp_outgoing_tos ds-field [!]aclname ...
#
#               Example where normal_service_net uses the TOS value 0x00
#               and normal_service_net uses 0x20
#
#               acl normal_service_net src 10.0.0.0/255.255.255.0
#               acl good_service_net src 10.0.1.0/255.255.255.0
#               tcp_outgoing_tos 0x00 normal_service_net 0x00
#               tcp_outgoing_tos 0x20 good_service_net
#
#               TOS/DSCP values really only have local significance - so you should
#               know what you're specifying. For more, see RFC 2474
#
#               The TOS/DSCP byte must be exactly that - a byte, value  0 - 255, or
#               "default" to use whatever default your host has.
#
#               Processing proceeds in the order specified, and stops at first fully
#               matching line.
#
#Default:
# none

#  TAG: tcp_outgoing_address
#               Allows you to map requests to different outgoing IP addresses
#               based on the username or sourceaddress of the user making
#               the request.
#
#               tcp_outgoing_address ipaddr [[!]aclname] ...
#
#               Example where requests from 10.0.0.0/24 will be forwareded
#               with source address 10.1.0.1, 10.0.2.0/24 forwarded with
#               source address 10.1.0.2 and the rest will be forwarded with
#               source address 10.1.0.3.
#
#               acl normal_service_net src 10.0.0.0/255.255.255.0
#               acl good_service_net src 10.0.1.0/255.255.255.0
#               tcp_outgoing_address 10.0.0.1 normal_service_net
#               tcp_outgoing_address 10.0.0.2 good_service_net
#               tcp_outgoing_address 10.0.0.3
#
#               Processing proceeds in the order specified, and stops at first fully
#               matching line.
#
#Default:
# none

#  TAG: reply_header_max_size          (KB)
#               This specifies the maximum size for HTTP headers in a reply.
#               Reply headers are usually relatively small (about 512 bytes).
#               Placing a limit on the reply header size will catch certain
#               bugs (for example with persistent connections) and possibly
#               buffer-overflow or denial-of-service attacks.
```

```
#
#Default:
# reply_header_max_size 20 KB

#  TAG: reply_body_max_size             bytes allow|deny acl acl...
#       This option specifies the maximum size of a reply body in bytes.
#               It can be used to prevent users from downloading very large files,
#               such as MP3's and movies. When the reply headers are recieved,
#               the reply_body_max_size lines are processed, and the first line with
#               a result of "allow" is used as the maximum body size for this reply.
#               This size is checked twice. First when we get the reply headers,
#               we check the content-length value.  If the content length value exists
#               and is larger than the allowed size, the request is denied and the
#               user receives an error message that says "the request or reply
#               is too large." If there is no content-length, and the reply
#               size exceeds this limit, the client's connection is just closed
#               and they will receive a partial reply.
#
#               WARNING: downstream caches probably can not detect a partial reply
#               if there is no content-length header, so they will cache
#               partial responses and give them out as hits.  You should NOT
#               use this option if you have downstream caches.
#
#               If you set this parameter to zero (the default), there will be
#               no limit imposed.
#
#Default:
# reply_body_max_size 0 allow all


# ADMINISTRATIVE PARAMETERS
#
-----------------------------------------------------------------------------

#  TAG: cache_mgr
#               Email-address of local cache manager who will receive
#               mail if the cache dies.  The default is "root".
#cache_mgr root
#
#Default:
# cache_mgr root

#  TAG: cache_effective_user
#               If you start Squid as root, it will change its effective/real
#               UID/GID to the user specified below.  The default is to change
#               to UID to "squid".  If you define cache_effective_user, but not
#               cache_effective_group, Squid sets the GID to the effective
#               user's default group ID (taken from the password file) and
#               supplementary group list from the from groups membership of
#               cache_effective_user.
#cache_effective_user squid
#
#Default:
# cache_effective_user squid

#  TAG: cache_effective_group
#               If you want Squid to run with a specific GID regardless of
#               the group memberships of the effective user then set this
#               to the group (or GID) you want Squid to run as. When set
#               all other group privileges of the effective user is ignored
#               and only this GID is effective. If Squid is not started as
#               root the user starting Squid must be member of the specified
#               group.
#cache_effective_group squid
#
#Default:
# cache_effective_group squid

#  TAG: visible_hostname
#               If you want to present a special hostname in error messages, etc,
#               define this.  Otherwise, the return value of gethostname()
#               will be used. If you have multiple caches in a cluster and
#               get errors about IP-forwarding you must set them to have individual
```

```
#               names with this setting.
#
#Default:
# none

#  TAG: unique_hostname
#               If you want to have multiple machines with the same
#               'visible_hostname' you must give each machine a different
#               'unique_hostname' so forwarding loops can be detected.
#
#Default:
# none

#  TAG: hostname_aliases
#               A list of other DNS names your cache has.
#
#Default:
# none


# OPTIONS FOR THE CACHE REGISTRATION SERVICE
#
----------------------------------------------------------------------------
#
#               This section contains parameters for the (optional) cache
#               announcement service.  This service is provided to help
#               cache administrators locate one another in order to join or
#               create cache hierarchies.
#
#               An 'announcement' message is sent (via UDP) to the registration
#               service by Squid.  By default, the announcement message is NOT
#               SENT unless you enable it with 'announce_period' below.
#
#               The announcement message includes your hostname, plus the
#               following information from this configuration file:
#
#                               http_port
#                               icp_port
#                               cache_mgr
#
#               All current information is processed regularly and made
#               available on the Web at http://www.ircache.net/Cache/Tracker/.

#  TAG: announce_period
#               This is how frequently to send cache announcements.  The
#               default is `0' which disables sending the announcement
#               messages.
#
#               To enable announcing your cache, just uncomment the line
#               below.
#
#Default:
# announce_period 0
#
#To enable announcing your cache, just uncomment the line below.
#announce_period 1 day

#  TAG: announce_host
#  TAG: announce_file
#  TAG: announce_port
#               announce_host and announce_port set the hostname and port
#               number where the registration message will be sent.
#
#               Hostname will default to 'tracker.ircache.net' and port will
#               default default to 3131.  If the 'filename' argument is given,
#               the contents of that file will be included in the announce
#               message.
#
#Default:
# announce_host tracker.ircache.net
# announce_port 3131
```

```
# HTTPD-ACCELERATOR OPTIONS
#
-----------------------------------------------------------------------------

#  TAG: httpd_accel_host
#  TAG: httpd_accel_port
#               If you want to run Squid as an httpd accelerator, define the
#               host name and port number where the real HTTP server is.
#
#               If you want IP based virtual host support specify the
#               hostname as "virtual". This will make Squid use the IP address
#               where it accepted the request as hostname in the URL.
#
#               If you want virtual port support specify the port as "0".
#
#               NOTE: enabling httpd_accel_host disables proxy-caching and
#               ICP.  If you want these features enabled also, set
#               the 'httpd_accel_with_proxy' option.
#
#Default:
# httpd_accel_port 80

#  TAG: httpd_accel_single_host         on|off
#               If you are running Squid as an accelerator and have a single backend
#               server set this to on. This causes Squid to forward the request
#               to this server, regardles of what any redirectors or Host headers
#               say.
#
#               Leave this at off if you have multiple backend servers, and use a
#               redirector (or host table or private DNS) to map the requests to the
#               appropriate backend servers. Note that the mapping needs to be a
#               1-1 mapping between requested and backend (from redirector) domain
#               names or caching will fail, as cacing is performed using the
#               URL returned from the redirector.
#
#               See also redirect_rewrites_host_header.
#
#Default:
# httpd_accel_single_host off

#  TAG: httpd_accel_with_proxy          on|off
#               If you want to use Squid as both a local httpd accelerator
#               and as a proxy, change this to 'on'. Note however your
#               proxy users may have trouble to reach the accelerated domains
#               unless their browsers are configured not to use this proxy for
#               those domains (for example via the no_proxy browser configuration
#               setting)
#
#Default:
# httpd_accel_with_proxy off

#  TAG: httpd_accel_uses_host_header                on|off
#               HTTP/1.1 requests include a Host: header which is basically the
#               hostname from the URL.  The Host: header is used for domain based
#               virutal hosts. If your accelerator needs to provide domain based
#               virtual hosts on the same IP address you will need to turn this
#               on.
#
#               Note Squid does NOT check the value of the Host header matches
#               any of your accelerated server, so it may open a big security hole
#               unless you take care to set up access controls proper.  We recommend
#               this option remain disabled unless you are sure of what you
#               are doing.
#
#               However, you will need to enable this option if you run Squid
#               as a transparent proxy.  Otherwise, virtual servers which
#               require the Host: header will not be properly cached.
#
#Default:
# httpd_accel_uses_host_header off

#  TAG: httpd_accel_no_pmtu_disc      on|off
#               In many setups of transparently intercepting proxies Path-MTU
```

```
#               discovery can not work on traffic towards the clients. This is
#               the case when the intercepting device does not fully track
#               connections and fails to forward ICMP must fragment messages
#               to the cache server.
#
#               If you have such setup and experience that certain clients
#               sporadically hang or never complete requests set this to on.
#
#Default:
# httpd_accel_no_pmtu_disc off


# MISCELLANEOUS
#
-----------------------------------------------------------------------------

#  TAG: dns_testnames
#               The DNS tests exit as soon as the first site is successfully looked up
#
#               This test can be disabled with the -D command line option.
#
#Default:
# dns_testnames netscape.com internic.net nlanr.net microsoft.com

#  TAG: logfile_rotate
#               Specifies the number of logfile rotations to make when you
#               type 'squid -k rotate'.  The default is 10, which will rotate
#               with extensions 0 through 9.  Setting logfile_rotate to 0 will
#               disable the rotation, but the logfiles are still closed and
#               re-opened.  This will enable you to rename the logfiles
#               yourself just before sending the rotate signal.
#
#               Note, the 'squid -k rotate' command normally sends a USR1
#               signal to the running squid process.  In certain situations
#               (e.g. on Linux with Async I/O), USR1 is used for other
#               purposes, so -k rotate uses another signal.  It is best to get
#               in the habit of using 'squid -k rotate' instead of 'kill -USR1
#               <pid>'.
#
#logfile_rotate 0
#
#Default:
# logfile_rotate 0

#  TAG: append_domain
#               Appends local domain name to hostnames without any dots in
#               them.  append_domain must begin with a period.
#
#               Be warned there are now Internet names with no dots in
#               them using only top-domain names, so setting this may
#               cause some Internet sites to become unavailable.
#
#Example:
# append_domain .yourdomain.com
#
#Default:
# none

#  TAG: tcp_recv_bufsize (bytes)
#               Size of receive buffer to set for TCP sockets.  Probably just
#               as easy to change your kernel's default.  Set to zero to use
#               the default buffer size.
#
#Default:
# tcp_recv_bufsize 0 bytes

#  TAG: err_html_text
#               HTML text to include in error messages.  Make this a "mailto"
#               URL to your admin address, or maybe just a link to your
#               organizations Web page.
#
#               To include this in your error messages, you must rewrite
#               the error template files (found in the "errors" directory).
```

```
#		Wherever you want the 'err_html_text' line to appear,
#		insert a %L tag in the error template file.
#
#Default:
# none

#  TAG: deny_info
#		Usage:   deny_info err_page_name acl
#		or       deny_info http://... acl
#		Example: deny_info ERR_CUSTOM_ACCESS_DENIED bad_guys
#
#		This can be used to return a ERR_ page for requests which
#		do not pass the 'http_access' rules.  A single ACL will cause
#		the http_access check to fail.  If a 'deny_info' line exists
#		for that ACL Squid returns a corresponding error page.
#
#		You may use ERR_ pages that come with Squid or create your own pages
#		and put them into the configured errors/ directory.
#
#		Alternatively you can specify an error URL. The browsers will
#		get redirected (302) to the specified URL. %s in the redirection
#		URL will be replaced by the requested URL.
#
#		Alternatively you can tell Squid to reset the TCP connection
#		by specifying TCP_RESET.
#
#Default:
# none

#  TAG: memory_pools    on|off
#		If set, Squid will keep pools of allocated (but unused) memory
#		available for future use.  If memory is a premium on your
#		system and you believe your malloc library outperforms Squid
#		routines, disable this.
#
#Default:
# memory_pools on

#  TAG: memory_pools_limit		(bytes)
#		Used only with memory_pools on:
#		memory_pools_limit 50 MB
#
#		If set to a non-zero value, Squid will keep at most the specified
#		limit of allocated (but unused) memory in memory pools. All free()
#		requests that exceed this limit will be handled by your malloc
#		library. Squid does not pre-allocate any memory, just safe-keeps
#		objects that otherwise would be free()d. Thus, it is safe to set
#		memory_pools_limit to a reasonably high value even if your
#		configuration will use less memory.
#
#		If set to zero, Squid will keep all memory it can. That is, there
#		will be no limit on the total amount of memory used for safe-keeping.
#
#		To disable memory allocation optimization, do not set
#		memory_pools_limit to 0. Set memory_pools to "off" instead.
#
#		An overhead for maintaining memory pools is not taken into account
#		when the limit is checked. This overhead is close to four bytes per
#		object kept. However, pools may actually _save_ memory because of
#		reduced memory thrashing in your malloc library.
#
#Default:
# memory_pools_limit 5 MB

#  TAG: forwarded_for     on|off
#		If set, Squid will include your system's IP address or name
#		in the HTTP requests it forwards.  By default it looks like
#		this:
#
#				X-Forwarded-For: 192.1.2.3
#
#		If you disable this, it will appear as
#
```

```
#                               X-Forwarded-For: unknown
#
#Default:
# forwarded_for on

#  TAG: log_icp_queries  on|off
#               If set, ICP queries are logged to access.log. You may wish
#               do disable this if your ICP load is VERY high to speed things
#               up or to simplify log analysis.
#
#Default:
# log_icp_queries on

#  TAG: icp_hit_stale       on|off
#               If you want to return ICP_HIT for stale cache objects, set this
#               option to 'on'.  If you have sibling relationships with caches
#               in other administrative domains, this should be 'off'.  If you only
#               have sibling relationships with caches under your control,
#               it is probably okay to set this to 'on'.
#               If set to 'on', your siblings should use the option "allow-miss"
#               on their cache_peer lines for connecting to you.
#
#Default:
# icp_hit_stale off

#  TAG: minimum_direct_hops
#               If using the ICMP pinging stuff, do direct fetches for sites
#               which are no more than this many hops away.
#
#Default:
# minimum_direct_hops 4

#  TAG: minimum_direct_rtt
#               If using the ICMP pinging stuff, do direct fetches for sites
#               which are no more than this many rtt milliseconds away.
#
#Default:
# minimum_direct_rtt 400

#  TAG: cachemgr_passwd
#               Specify passwords for cachemgr operations.
#
#               Usage: cachemgr_passwd password action action ...
#
#               Some valid actions are (see cache manager menu for a full list):
#                               5min
#                               60min
#                               asndb
#                               authenticator
#                               cbdata
#                               client_list
#                               comm_incoming
#                               config *
#                               counters
#                               delay
#                               digest_stats
#                               dns
#                               events
#                               filedescriptors
#                               fqdncache
#                               histograms
#                               http_headers
#                               info
#                               io
#                               ipcache
#                               mem
#                               menu
#                               netdb
#                               non_peers
#                               objects
#                               offline_toggle *
#                               pconn
#                               peer_select
```

```
#                               redirector
#                               refresh
#                               server_list
#                               shutdown *
#                               store_digest
#                               storedir
#                               utilization
#                               via_headers
#                               vm_objects
#
#               * Indicates actions which will not be performed without a
#                 valid password, others can be performed if not listed here.
#
#               To disable an action, set the password to "disable".
#               To allow performing an action without a password, set the
#               password to "none".
#
#               Use the keyword "all" to set the same password for all actions.
#
#Example:
# cachemgr_passwd secret shutdown
# cachemgr_passwd lesssssssecret info stats/objects
# cachemgr_passwd disable all
#
#Default:
# none

#  TAG: store_avg_object_size           (kbytes)
#               Average object size, used to estimate number of objects your
#               cache can hold.  See doc/Release-Notes-1.1.txt.  The default is
#               13 KB.
#
#Default:
# store_avg_object_size 13 KB

#  TAG: store_objects_per_bucket
#               Target number of objects per bucket in the store hash table.
#               Lowering this value increases the total number of buckets and
#               also the storage maintenance rate.  The default is 50.
#
#Default:
# store_objects_per_bucket 20

#  TAG: client_db           on|off
#               If you want to disable collecting per-client statistics,
#               turn off client_db here.
#
#Default:
# client_db on

#  TAG: netdb_low
#  TAG: netdb_high
#               The low and high water marks for the ICMP measurement
#               database.  These are counts, not percents.  The defaults are
#               900 and 1000.  When the high water mark is reached, database
#               entries will be deleted until the low mark is reached.
#
#Default:
# netdb_low 900
# netdb_high 1000

#  TAG: netdb_ping_period
#               The minimum period for measuring a site.  There will be at
#               least this much delay between successive pings to the same
#               network.  The default is five minutes.
#
#Default:
# netdb_ping_period 5 minutes

#  TAG: query_icmp        on|off
#               If you want to ask your peers to include ICMP data in their ICP
#               replies, enable this option.
#
```

```
#               If your peer has configured Squid (during compilation) with
#               '--enable-icmp' that peer will send ICMP pings to origin server
#               sites of the URLs it receives.  If you enable this option the
#               ICP replies from that peer will include the ICMP data (if available).
#               Then, when choosing a parent cache, Squid will choose the parent with
#               the minimal RTT to the origin server.  When this happens, the
#               hierarchy field of the access.log will be
#               "CLOSEST_PARENT_MISS".  This option is off by default.
#
#Default:
# query_icmp off

#  TAG: test_reachability  on|off
#               When this is 'on', ICP MISS replies will be ICP_MISS_NOFETCH
#               instead of ICP_MISS if the target host is NOT in the ICMP
#               database, or has a zero RTT.
#
#Default:
# test_reachability off

#  TAG: buffered_logs      on|off
#               cache.log log file is written with stdio functions, and as such
#               it can be buffered or unbuffered. By default it will be unbuffered.
#               Buffering it can speed up the writing slightly (though you are
#               unlikely to need to worry unless you run with tons of debugging
#               enabled in which case performance will suffer badly anyway..).
#
#Default:
# buffered_logs off

#  TAG: reload_into_ims  on|off
#               When you enable this option, client no-cache or ``reload''
#               requests will be changed to If-Modified-Since requests.
#               Doing this VIOLATES the HTTP standard.  Enabling this
#               feature could make you liable for problems which it
#               causes.
#
#               see also refresh_pattern for a more selective approach.
#
#Default:
# reload_into_ims off

#  TAG: always_direct
#               Usage: always_direct allow|deny [!]aclname ...
#
#               Here you can use ACL elements to specify requests which should
#               ALWAYS be forwarded by Squid to the origin servers without using
#               any peers.  For example, to always directly forward requests for
#               local servers ignoring any parents or siblings you may have use
#               something like:
#
#                               acl local-servers dstdomain my.domain.net
#                               always_direct allow local-servers
#
#               To always forward FTP requests directly, use
#
#                               acl FTP proto FTP
#                               always_direct allow FTP
#
#               NOTE: There is a similar, but opposite option named
#               'never_direct'.  You need to be aware that "always_direct deny
#               foo" is NOT the same thing as "never_direct allow foo".  You
#               may need to use a deny rule to exclude a more-specific case of
#               some other rule.  Example:
#
#                               acl local-external dstdomain external.foo.net
#                               acl local-servers dstdomain  .foo.net
#                               always_direct deny local-external
#                               always_direct allow local-servers
#
#               NOTE: If your goal is to make the client forward the request
#               directly to the origin server bypassing Squid then this needs
#               to be done in the client configuration. Squid configuration
```

```
#		can only tell Squid how Squid should fetch the object.
#
#		NOTE: This directive is not related to caching. The replies
#		is cached as usual even if you use always_direct. To not cache
#		the replies see no_cache.
#
#		This option replaces some v1.1 options such as local_domain
#		and local_ip.
#
#Default:
# none

#  TAG: never_direct
#		Usage: never_direct allow|deny [!]aclname ...
#
#		never_direct is the opposite of always_direct.  Please read
#		the description for always_direct if you have not already.
#
#		With 'never_direct' you can use ACL elements to specify
#		requests which should NEVER be forwarded directly to origin
#		servers.  For example, to force the use of a proxy for all
#		requests, except those in your local domain use something like:
#
#			acl local-servers dstdomain .foo.net
#			acl all src 0.0.0.0/0.0.0.0
#			never_direct deny local-servers
#			never_direct allow all
#
#		or if Squid is inside a firewall and there are local intranet
#		servers inside the firewall use something like:
#
#			acl local-intranet dstdomain .foo.net
#			acl local-external dstdomain external.foo.net
#			always_direct deny local-external
#			always_direct allow local-intranet
#			never_direct allow all
#
#		This option replaces some v1.1 options such as inside_firewall
#		and firewall_ip.
#
#Default:
# none

#  TAG: header_access
#		Usage: header_access header_name allow|deny [!]aclname ...
#
#		WARNING: Doing this VIOLATES the HTTP standard.  Enabling
#		this feature could make you liable for problems which it
#		causes.
#
#		This option replaces the old 'anonymize_headers' and the
#		older 'http_anonymizer' option with something that is much
#		more configurable. This new method creates a list of ACLs
#		for each header, allowing you very fine-tuned header
#		mangling.
#
#		You can only specify known headers for the header name.
#		Other headers are reclassified as 'Other'. You can also
#		refer to all the headers with 'All'.
#
#		For example, to achieve the same behaviour as the old
#		'http_anonymizer standard' option, you should use:
#
#			header_access From deny all
#			header_access Referer deny all
#			header_access Server deny all
#			header_access User-Agent deny all
#			header_access WWW-Authenticate deny all
#			header_access Link deny all
#
#		Or, to reproduce the old 'http_anonymizer paranoid' feature
#		you should use:
#
```

```
#                       header_access Allow allow all
#                       header_access Authorization allow all
#                       header_access WWW-Authenticate allow all
#                       header_access Cache-Control allow all
#                       header_access Content-Encoding allow all
#                       header_access Content-Length allow all
#                       header_access Content-Type allow all
#                       header_access Date allow all
#                       header_access Expires allow all
#                       header_access Host allow all
#                       header_access If-Modified-Since allow all
#                       header_access Last-Modified allow all
#                       header_access Location allow all
#                       header_access Pragma allow all
#                       header_access Accept allow all
#                       header_access Accept-Charset allow all
#                       header_access Accept-Encoding allow all
#                       header_access Accept-Language allow all
#                       header_access Content-Language allow all
#                       header_access Mime-Version allow all
#                       header_access Retry-After allow all
#                       header_access Title allow all
#                       header_access Connection allow all
#                       header_access Proxy-Connection allow all
#                       header_access All deny all
#
#       By default, all headers are allowed (no anonymizing is
#       performed).
#
#Default:
# none

#  TAG: header_replace
#       Usage:   header_replace header_name message
#       Example: header_replace User-Agent Nutscrape/1.0 (CP/M; 8-bit)
#
#       This option allows you to change the contents of headers
#       denied with header_access above, by replacing them with
#       some fixed string. This replaces the old fake_user_agent
#       option.
#
#       By default, headers are removed if denied.
#
#Default:
# none

#  TAG: icon_directory
#       Where the icons are stored. These are normally kept in
#       /usr/share/squid/icons
#
#Default:
# icon_directory /usr/share/squid/icons

#  TAG: short_icon_urls
#       If this is enabled Squid will use short URLs for icons.
#
#       If off the URLs for icons will always be absolute URLs
#       including the proxy name and port.
#
#Default:
# short_icon_urls off

#  TAG: error_directory
#       Directory where the error files are read from.
#       /usr/lib/squid/errors contains sets of error files
#       in different languages. The default error directory
#       is /etc/squid/errors, which is a link to one of these
#       error sets.
#
#       If you wish to create your own versions of the error files,
#       either to customize them to suit your language or company,
#       copy the template English files to another
#       directory and point this tag at them.
```

```
#
#error_directory /usr/share/squid/errors/English
#
#Default:
# error_directory /usr/share/squid/errors/English

#  TAG: maximum_single_addr_tries
#               This sets the maximum number of connection attempts for a
#               host that only has one address (for multiple-address hosts,
#               each address is tried once).
#
#               The default value is one attempt, the (not recommended)
#               maximum is 255 tries.  A warning message will be generated
#               if it is set to a value greater than ten.
#
#               Note: This is in addition to the request reforwarding which
#               takes place if Squid fails to get a satisfying response.
#
#Default:
# maximum_single_addr_tries 1

#  TAG: retry_on_error
#               If set to on Squid will automatically retry requests when
#               receiving an error response. This is mainly useful if you
#               are in a complex cache hierarchy to work around access
#               control errors.
#
#Default:
# retry_on_error off

#  TAG: snmp_port
#               Squid can now serve statistics and status information via SNMP.
#               A value of "0" disables SNMP support. If you wish to use SNMP,
#               set this to "3401" to use the normal SNMP support.
#
#Default:
# snmp_port 0

#  TAG: snmp_access
#               Allowing or denying access to the SNMP port.
#
#               All access to the agent is denied by default.
#               usage:
#
#               snmp_access allow|deny [!]aclname ...
#
#Example:
# snmp_access allow snmppublic localhost
# snmp_access deny all
#
#Default:
# snmp_access deny all

#  TAG: snmp_incoming_address
#  TAG: snmp_outgoing_address
#               Just like 'udp_incoming_address' above, but for the SNMP port.
#
#               snmp_incoming_address is used for the SNMP socket receiving
#                                                       messages from SNMP agents.
#               snmp_outgoing_address  is used for SNMP packets returned to SNMP
#                                                       agents.
#
#               The default snmp_incoming_address (0.0.0.0) is to listen on all
#               available network interfaces.
#
#               If snmp_outgoing_address is set to 255.255.255.255 (the default)
#               it will use the same socket as snmp_incoming_address. Only
#               change this if you want to have SNMP replies sent using another
#               address than where this Squid listens for SNMP queries.
#
#               NOTE, snmp_incoming_address and snmp_outgoing_address can not have
#               the same value since they both use port 3401.
#
```

```
#Default:
# snmp_incoming_address 0.0.0.0
# snmp_outgoing_address 255.255.255.255

#  TAG: as_whois_server
#               WHOIS server to query for AS numbers.  NOTE: AS numbers are
#               queried only when Squid starts up, not for every request.
#
#Default:
# as_whois_server whois.ra.net
# as_whois_server whois.ra.net

#  TAG: wccp_router
#               Use this option to define your WCCP ``home'' router for
#               Squid.   Setting the 'wccp_router' to 0.0.0.0 (the default)
#               disables WCCP.
#
#Default:
# wccp_router 0.0.0.0

#  TAG: wccp_version
#               According to some users, Cisco IOS 11.2 only supports WCCP
#               version 3.  If you're using that version of IOS, change
#               this value to 3.
#
#Default:
# wccp_version 4

#  TAG: wccp_incoming_address
#  TAG: wccp_outgoing_address
#       wccp_incoming_address   Use this option if you require WCCP
#                                                       messages to be received on only one
#                                                       interface.  Do NOT use this option if
#                                                       you're unsure how many interfaces you
#                                                       have, or if you know you have only one
#                                                       interface.
#
#               wccp_outgoing_address  Use this option if you require WCCP
#                                                       messages to be sent out on only one
#                                                       interface.  Do NOT use this option if
#                                                       you're unsure how many interfaces you
#                                                       have, or if you know you have only one
#                                                       interface.
#
#       The default behavior is to not bind to any specific address.
#
#       NOTE, wccp_incoming_address and wccp_outgoing_address can not have
#       the same value since they both use port 2048.
#
#Default:
# wccp_incoming_address 0.0.0.0
# wccp_outgoing_address 255.255.255.255


# DELAY POOL PARAMETERS (all require DELAY_POOLS compilation option)
#
-----------------------------------------------------------------------------

#  TAG: delay_pools
#               This represents the number of delay pools to be used.  For example,
#               if you have one class 2 delay pool and one class 3 delays pool, you
#               have a total of 2 delay pools.
#
#Default:
# delay_pools 0

#  TAG: delay_class
#               This defines the class of each delay pool.  There must be exactly one
#               delay_class line for each delay pool.  For example, to define two
#               delay pools, one of class 2 and one of class 3, the settings above
#               and here would be:
#
#Example:
```

```
# delay_pools 2      # 2 delay pools
# delay_class 1 2    # pool 1 is a class 2 pool
# delay_class 2 3    # pool 2 is a class 3 pool
#
#               The delay pool classes are:
#
#                       class 1                 Everything is limited by a single aggregate
#                                               bucket.
#
#                       class 2         Everything is limited by a single aggregate
#                                               bucket as well as an "individual" bucket chosen
#                                               from bits 25 through 32 of the IP address.
#
#                       class 3                 Everything is limited by a single aggregate
#                                               bucket as well as a "network" bucket chosen
#                                               from bits 17 through 24 of the IP address and a
#                                               "individual" bucket chosen from bits 17 through
#                                               32 of the IP address.
#
#               NOTE: If an IP address is a.b.c.d
#                       -> bits 25 through 32 are "d"
#                       -> bits 17 through 24 are "c"
#                       -> bits 17 through 32 are "c * 256 + d"
#
#Default:
# none

#  TAG: delay_access
#               This is used to determine which delay pool a request falls into.
#
#               delay_access is sorted per pool and the matching starts with pool 1,
#               then pool 2, ..., and finally pool N. The first delay pool where
the
#               request is allowed is selected for the request. If it does not
allow
#               the request to any pool then the request is not delayed (default).
#
#               For example, if you want some_big_clients in delay
#               pool 1 and lotsa_little_clients in delay pool 2:
#
#Example:
# delay_access 1 allow some_big_clients
# delay_access 1 deny all
# delay_access 2 allow lotsa_little_clients
# delay_access 2 deny all
#
#Default:
# none

#  TAG: delay_parameters
#               This defines the parameters for a delay pool.  Each delay pool has
#               a number of "buckets" associated with it, as explained in the
#               description of delay_class.  For a class 1 delay pool, the syntax is:
#
#delay_parameters pool aggregate
#
#               For a class 2 delay pool:
#
#delay_parameters pool aggregate individual
#
#               For a class 3 delay pool:
#
#delay_parameters pool aggregate network individual
#
#               The variables here are:
#
#                       pool                    a pool number - ie, a number between 1 and the
#                                               number specified in delay_pools as used in
#                                               delay_class lines.
#
#                       aggregate       the "delay parameters" for the aggregate bucket
#                                               (class 1, 2, 3).
#
```

```
#                           individual      the "delay parameters" for the individual
#                                           buckets (class 2, 3).
#
#                           network         the "delay parameters" for the network buckets
#                                           (class 3).
#
#       A pair of delay parameters is written restore/maximum, where restore is
#       the number of bytes (not bits - modem and network speeds are usually
#       quoted in bits) per second placed into the bucket, and maximum is the
#       maximum number of bytes which can be in the bucket at any time.
#
#       For example, if delay pool number 1 is a class 2 delay pool as in the
#       above example, and is being used to strictly limit each host to 64kbps
#       (plus overheads), with no overall limit, the line is:
#
#delay_parameters 1 -1/-1 8000/8000
#
#       Note that the figure -1 is used to represent "unlimited".
#
#       And, if delay pool number 2 is a class 3 delay pool as in the above
#       example, and you want to limit it to a total of 256kbps (strict limit)
#       with each 8-bit network permitted 64kbps (strict limit) and each
#       individual host permitted 4800bps with a bucket maximum size of 64kb
#       to permit a decent web page to be downloaded at a decent speed
#       (if the network is not being limited due to overuse) but slow down
#       large downloads more significantly:
#
#delay_parameters 2 32000/32000 8000/8000 600/8000
#
#       There must be one delay_parameters line for each delay pool.
#
#Default:
# none

#  TAG: delay_initial_bucket_level      (percent, 0-100)
#       The initial bucket percentage is used to determine how much is put
#       in each bucket when squid starts, is reconfigured, or first notices
#       a host accessing it (in class 2 and class 3, individual hosts and
#       networks only have buckets associated with them once they have been
#       "seen" by squid).
#
#Default:
# delay_initial_bucket_level 50

#  TAG: incoming_icp_average
#  TAG: incoming_http_average
#  TAG: incoming_dns_average
#  TAG: min_icp_poll_cnt
#  TAG: min_dns_poll_cnt
#  TAG: min_http_poll_cnt
#       Heavy voodoo here.  I can't even believe you are reading this.
#       Are you crazy?  Don't even think about adjusting these unless
#       you understand the algorithms in comm_select.c first!
#
#Default:
# incoming_icp_average 6
# incoming_http_average 4
# incoming_dns_average 4
# min_icp_poll_cnt 8
# min_dns_poll_cnt 8
# min_http_poll_cnt 8

#  TAG: max_open_disk_fds
#       To avoid having disk as the I/O bottleneck Squid can optionally
#       bypass the on-disk cache if more than this amount of disk file
#       descriptors are open.
#
#       A value of 0 indicates no limit.
#
#Default:
# max_open_disk_fds 0

#  TAG: offline_mode
```

```
#               Enable this option and Squid will never try to validate cached
#               objects.
#
#Default:
# offline_mode off

#  TAG: uri_whitespace
#               What to do with requests that have whitespace characters in the
#               URI.  Options:
#
#               strip:  The whitespace characters are stripped out of the URL.
#                               This is the behavior recommended by RFC2396.
#               deny:   The request is denied.  The user receives an "Invalid
#                               Request" message.
#               allow:  The request is allowed and the URI is not changed.  The
#                               whitespace characters remain in the URI.  Note the
#                               whitespace is passed to redirector processes if they
#                               are in use.
#               encode:         The request is allowed and the whitespace characters are
#                               encoded according to RFC1738.  This could be considered
#                               a violation of the HTTP/1.1
#                               RFC because proxies are not allowed to rewrite URI's.
#               chop:           The request is allowed and the URI is chopped at the
#                               first whitespace.  This might also be considered a
#                               violation.
#
#Default:
# uri_whitespace strip

#  TAG: broken_posts
#               A list of ACL elements which, if matched, causes Squid to send
#               an extra CRLF pair after the body of a PUT/POST request.
#
#               Some HTTP servers has broken implementations of PUT/POST,
#               and rely on an extra CRLF pair sent by some WWW clients.
#
#               Quote from RFC 2068 section 4.1 on this matter:
#
#                 Note: certain buggy HTTP/1.0 client implementations generate an
#                 extra CRLF's after a POST request. To restate what is explicitly
#                 forbidden by the BNF, an HTTP/1.1 client must not preface or follow
#                 a request with an extra CRLF.
#
#Example:
# acl buggy_server url_regex ^http://....
# broken_posts allow buggy_server
#
#Default:
# none

#  TAG: mcast_miss_addr
# Note: This option is only available if Squid is rebuilt with the
#       -DMULTICAST_MISS_STREAM option
#
#               If you enable this option, every "cache miss" URL will
#               be sent out on the specified multicast address.
#
#               Do not enable this option unless you are are absolutely
#               certain you understand what you are doing.
#
#Default:
# mcast_miss_addr 255.255.255.255

#  TAG: mcast_miss_ttl
# Note: This option is only available if Squid is rebuilt with the
#       -DMULTICAST_MISS_TTL option
#
#               This is the time-to-live value for packets multicasted
#               when multicasting off cache miss URLs is enabled.  By
#               default this is set to 'site scope', i.e. 16.
#
#Default:
# mcast_miss_ttl 16
```

```
#  TAG: mcast_miss_port
# Note: This option is only available if Squid is rebuilt with the
#       -DMULTICAST_MISS_STREAM option
#
#               This is the port number to be used in conjunction with
#               'mcast_miss_addr'.
#
#Default:
# mcast_miss_port 3135

#  TAG: mcast_miss_encode_key
# Note: This option is only available if Squid is rebuilt with the
#       -DMULTICAST_MISS_STREAM option
#
#               The URLs that are sent in the multicast miss stream are
#               encrypted.  This is the encryption key.
#
#Default:
# mcast_miss_encode_key XXXXXXXXXXXXXXXX

#  TAG: nonhierarchical_direct
#               By default, Squid will send any non-hierarchical requests
#               (matching hierarchy_stoplist or not cachable request type) direct
#               to origin servers.
#
#               If you set this to off, Squid will prefer to send these
#               requests to parents.
#
#               Note that in most configurations, by turning this off you will only
#               add latency to these request without any improvement in global hit
#               ratio.
#
#               If you are inside an firewall see never_direct instead of
#               this directive.
#
#Default:
# nonhierarchical_direct on

#  TAG: prefer_direct
#               Normally Squid tries to use parents for most requests. If you for some
#               reason like it to first try going direct and only use a parent if
#               going direct fails set this to on.
#
#               By combining nonhierarchical_direct off and prefer_direct on you
#               can set up Squid to use a parent as a backup path if going direct
#               fails.
#
#               Note: If you want Squid to use parents for all requests see
#               the never_direct directive. prefer_direct only modifies how Squid
#               acts on cachable requests.
#
#Default:
# prefer_direct off

#  TAG: strip_query_terms
#               By default, Squid strips query terms from requested URLs before
#               logging.  This protects your user's privacy.
#
#Default:
# strip_query_terms on

#  TAG: coredump_dir
#               By default Squid leaves core files in the directory from where
#               it was started. If you set 'coredump_dir' to a directory
#               that exists, Squid will chdir() to that directory at startup
#               and coredump files will be left there.
#
#Default:
# coredump_dir none
#
# Leave coredumps in the first cache dir
coredump_dir /var/spool/squid
```

```
#  TAG: redirector_bypass
#               When this is 'on', a request will not go through the
#               redirector if all redirectors are busy.  If this is 'off'
#               and the redirector queue grows too large, Squid will exit
#               with a FATAL error and ask you to increase the number of
#               redirectors.  You should only enable this if the redirectors
#               are not critical to your caching system.  If you use
#               redirectors for access control, and you enable this option,
#               users may have access to pages they should not
#               be allowed to request.
#
#Default:
# redirector_bypass off

#  TAG: ignore_unknown_nameservers
#               By default Squid checks that DNS responses are received
#               from the same IP addresses they are sent to.  If they
#               don't match, Squid ignores the response and writes a warning
#               message to cache.log.  You can allow responses from unknown
#               nameservers by setting this option to 'off'.
#
#Default:
# ignore_unknown_nameservers on

#  TAG: digest_generation
# Note: This option is only available if Squid is rebuilt with the
#       --enable-cache-digests option
#
#               This controls whether the server will generate a Cache Digest
#               of its contents.  By default, Cache Digest generation is
#               enabled if Squid is compiled with USE_CACHE_DIGESTS defined.
#
#Default:
# digest_generation on

#  TAG: digest_bits_per_entry
# Note: This option is only available if Squid is rebuilt with the
#       --enable-cache-digests option
#
#               This is the number of bits of the server's Cache Digest which
#               will be associated with the Digest entry for a given HTTP
#               Method and URL (public key) combination.  The default is 5.
#
#Default:
# digest_bits_per_entry 5

#  TAG: digest_rebuild_period           (seconds)
# Note: This option is only available if Squid is rebuilt with the
#       --enable-cache-digests option
#
#               This is the number of seconds between Cache Digest rebuilds.
#
#Default:
# digest_rebuild_period 1 hour

#  TAG: digest_rewrite_period           (seconds)
# Note: This option is only available if Squid is rebuilt with the
#       --enable-cache-digests option
#
#               This is the number of seconds between Cache Digest writes to
#               disk.
#
#Default:
# digest_rewrite_period 1 hour

#  TAG: digest_swapout_chunk_size   (bytes)
# Note: This option is only available if Squid is rebuilt with the
#       --enable-cache-digests option
#
#               This is the number of bytes of the Cache Digest to write to
#               disk at a time.  It defaults to 4096 bytes (4KB), the Squid
#               default swap page.
```

```
#
#Default:
# digest_swapout_chunk_size 4096 bytes

#  TAG: digest_rebuild_chunk_percentage             (percent, 0-100)
# Note: This option is only available if Squid is rebuilt with the
#       --enable-cache-digests option
#
#               This is the percentage of the Cache Digest to be scanned at a
#               time.  By default it is set to 10% of the Cache Digest.
#
#Default:
# digest_rebuild_chunk_percentage 10

#  TAG: chroot
#               Use this to have Squid do a chroot() while initializing.  This
#               also causes Squid to fully drop root privileges after
#               initializing.  This means, for example, that if you use a HTTP
#               port less than 1024 and try to reconfigure, you will get an
#               error.
#
#Default:
# none

#  TAG: client_persistent_connections
#  TAG: server_persistent_connections
#               Persistent connection support for clients and servers.  By
#               default, Squid uses persistent connections (when allowed)
#               with its clients and servers.  You can use these options to
#               disable persistent connections with clients and/or servers.
#
#Default:
# client_persistent_connections on
# server_persistent_connections on

#  TAG: detect_broken_pconn
#               Some servers have been found to incorrectly signal the use
#               of HTTP/1.0 persistent connections even on replies not
#               compatible, causing significant delays. This server problem
#               has mostly been seen on redirects.
#
#               By enabling this directive Squid attempts to detect such
#               broken replies and automatically assume the reply is finished
#               after 10 seconds timeout.
#
#Default:
# detect_broken_pconn off

#  TAG: balance_on_multiple_ip
#               Some load balancing servers based on round robin DNS have been
#               found not to preserve user session state across requests
#               to different IP addresses.
#
#               By default Squid rotates IP's per request. By disabling
#               this directive only connection failure trigers rotation.
#
#Default:
# balance_on_multiple_ip on

#  TAG: pipeline_prefetch
#               To boost the performance of pipelined requests to closer
#               match that of a non-proxied environment Squid can try to fetch
#               up to two requests in parallell from a pipeline.
#
#               Defaults to off for bandwidth management and access logging
#               reasons.
#
#Default:
# pipeline_prefetch off

#  TAG: extension_methods
#               Squid only knows about standardized HTTP request methods.
#               You can add up to 20 additional "extension" methods here.
```

```
#
#Default:
# none

#  TAG: request_entities
#               Squid defaults to deny GET and HEAD requests with request entities,
#               as the meaning of such requests are undefined in the HTTP standard
#               even if not explicitly forbidden.
#
#               Set this directive to on if you have clients which insists
#               on sending request entities in GET or HEAD requests. But be warned
#               that there is server software (both proxies and web servers) which
#               can fail to properly process this kind of request which may make you
#               vulnerable to cache pollution attacks if enabled.
#
#Default:
# request_entities off

#  TAG: high_response_time_warning (msec)
#               If the one-minute median response time exceeds this value,
#               Squid prints a WARNING with debug level 0 to get the
#               administrators attention.  The value is in milliseconds.
#
#Default:
# high_response_time_warning 0

#  TAG: high_page_fault_warning
#               If the one-minute average page fault rate exceeds this
#               value, Squid prints a WARNING with debug level 0 to get
#               the administrators attention.  The value is in page faults
#               per second.
#
#Default:
# high_page_fault_warning 0

#  TAG: high_memory_warning
#               If the memory usage (as determined by mallinfo) exceeds
#               value, Squid prints a WARNING with debug level 0 to get
#               the administrators attention.
#
#Default:
# high_memory_warning 0

#  TAG: store_dir_select_algorithm
#               Set this to 'round-robin' as an alternative.
#
#Default:
# store_dir_select_algorithm least-load

#  TAG: forward_log
# Note: This option is only available if Squid is rebuilt with the
#       -DWIP_FWD_LOG option
#
#               Logs the server-side requests.
#
#               This is currently work in progress.
#
#Default:
# none

#  TAG: ie_refresh          on|off
#               Microsoft Internet Explorer up until version 5.5 Service
#               Pack 1 has an issue with transparent proxies, wherein it
#               is impossible to force a refresh.  Turning this on provides
#               a partial fix to the problem, by causing all IMS-REFRESH
#               requests from older IE versions to check the origin server
#               for fresh content.  This reduces hit ratio by some amount
#               (~10% in my experience), but allows users to actually get
#               fresh content when they want it.  Note that because Squid
#               cannot tell if the user is using 5.5 or 5.5SP1, the behavior
#               of 5.5 is unchanged from old versions of Squid (i.e. a
#               forced refresh is impossible).  Newer versions of IE will,
#               hopefully, continue to have the new behavior and will be
```

```
#               handled based on that assumption.  This option defaults to
#               the old Squid behavior, which is better for hit ratios but
#               worse for clients using IE, if they need to be able to
#               force fresh content.
#
#Default:
# ie_refresh off

#  TAG: vary_ignore_expire              on|off
#               Many HTTP servers supporting Vary gives such objects
#               immediate expiry time with no cache-control header
#               when requested by a HTTP/1.0 client. This option
#               enables Squid to ignore such expiry times until
#               HTTP/1.1 is fully implemented.
#               WARNING: This may eventually cause some varying
#               objects not intended for caching to get cached.
#
#Default:
# vary_ignore_expire off

#  TAG: sleep_after_fork  (microseconds)
#               When this is set to a non-zero value, the main Squid process
#               sleeps the specified number of microseconds after a fork()
#               system call. This sleep may help the situation where your
#               system reports fork() failures due to lack of (virtual)
#               memory. Note, however, that if you have a lot of child
#               processes, these sleep delays will add up and your
#               Squid will not service requests for some amount of time
#               until all the child processes have been started.
#
#Default:
# sleep_after_fork 0

#  TAG: relaxed_header_parser           on|off|warn
#               In the default "on" setting Squid accepts certain forms
#               of non-compliant HTTP messages where it is unambiguous
#               what the sending application intended even if the message
#               is not correctly formatted. The messages is then normalized
#               to the correct form when forwarded by Squid.
#
#               If set to "warn" then a warning will be emitted in cache.log
#               each time such HTTP error is encountered.
#
#               If set to "off" then such HTTP errors will cause the request
#               or response to be rejected.
#
#Default:
# relaxed_header_parser on
```

ANEXO No. 3
ARCHIVO smb.conf

```
  # This is the main Samba configuration file. You should
                            read the
# smb.conf(5) manual page in order to understand the
options listed
# here. Samba has a huge number of configurable options
(perhaps too
# many!) most of which are not shown in this example
#
# Any line which starts with a ; (semi-colon) or a # (hash)
# is a comment and is ignored. In this example we will use
a #
# for commentry and a ; for parts of the config file that
you
# may wish to enable
#
# NOTE: Whenever you modify this file you should run the
command "testparm"
# to check that you have not made any basic syntactic
errors.
#
#======================= Global Settings
=====================================
[global]

# workgroup = NT-Domain-Name or Workgroup-Name
   workgroup = contraloria

# server string is the equivalent of the NT Description
field
   netbios name = SvrLinux
   server string = %L

# This option is important for security. It allows you to
restrict
# connections to machines which are on your local network.
The
# following example restricts access to two C class
networks and
# the "loopback" interface. For more examples of the syntax
see
# the smb.conf man page
;   hosts allow = 192.168.1. 192.168.2. 127.

# if you want to automatically load your printer list
rather
# than setting them up individually then you'll need this
```

```
   printcap name = /etc/printcap
   load printers = yes

# It should not be necessary to spell out the print system
type unless
# yours is non-standard. Currently supported print systems
include:
# bsd, sysv, plp, lprng, aix, hpux, qnx
   printing = cups

# This option tells cups that the data has already been
rasterized
cups options = raw

# Uncomment this if you want a guest account, you must add
this to
/etc/passwd
# otherwise the user "nobody" is used
;  guest account = pcguest

# this tells Samba to use a separate log file for each
machine
# that connects
log file = /var/log/samba/%m.log
# all log information in one file
#   log file = /var/log/samba/log.smbd

# Put a capping on the size of the log files (in Kb).
   max log size = 50

# Security mode. Most people will want user level security.
See
# security_level.txt for details.
   security = user
# Use password server option only with security = server
;   password server = <NT-Server-Name>

# Password Level allows matching of _n_ characters of the
password for
# all combinations of upper and lower case.
;  password level = 8
;  username level = 8

# You may wish to use password encryption. Please read
# ENCRYPTION.txt, Win95.txt and WinNT.txt in the Samba
documentation.
```

```
# Do not enable this option unless you have read those
documents

  encrypt passwords = yes
  smb passwd file = /etc/samba/smbpasswd
  unix password sync = yes
  null passwords = yes

# The following are needed to allow password changing from
Windows to
# update the Linux system password also.
# NOTE: Use these with 'encrypt passwords' and 'smb passwd
file' above.
# NOTE2: You do NOT need these to allow workstations to
change only
#        the encrypted SMB passwords. They allow the Unix
password
#        to be kept in sync with the SMB password.
;  unix password sync = Yes
;  passwd program = /usr/bin/passwd %u
;  passwd chat = *New*UNIX*password* %n\n
*ReType*new*UNIX*password* %n\n
*passwd:*all*authentication*tokens*updated*successfully*

# Unix users can map to different SMB User names
  username map = /etc/samba/smbusers

# Using the following line enables you to customise your
configuration
# on a per machine basis. The %m gets replaced with the
netbios name
# of the machine that is connecting
;   include = /etc/samba/smb.conf.%m

# Most people will find that this option gives better
performance.
# See speed.txt and the manual pages for details
   socket options = TCP_NODELAY SO_RCVBUF=8192
SO_SNDBUF=8192

# Configure Samba to use multiple interfaces
# If you have multiple network interfaces then you must
list them
# here. See the man page for details.
;   interfaces = 192.168.12.2/24 192.168.13.2/24
```

```
# Configure remote browse list synchronisation here
#  request announcement to, or browse list sync from:
#     a specific host or from / to a whole subnet (see
below)
;   remote browse sync = 192.168.3.25 192.168.5.255
# Cause this host to announce itself to local subnets here
;   remote announce = 192.168.1.255 192.168.2.44

# Browser Control Options:
# set local master to no if you don't want Samba to become
a master
# browser on your network. Otherwise the normal election
rules apply
;   local master = no

# OS Level determines the precedence of this server in
master browser
# elections. The default value should be reasonable
;   os level = 33
socket options = IPTOS_LOWDELAY TCP_NODELAY SO_SNDBUF=4096
SO_RCVBUF=4096
# Domain Master specifies Samba to be the Domain Master
Browser. This
# allows Samba to collate browse lists between subnets.
Don't use this
# if you already have a Windows NT domain controller doing
this job
;   domain master = yes

# Preferred Master causes Samba to force a local browser
election on startup
# and gives it a slightly higher chance of winning the
election
;   preferred master = yes

# Enable this if you want Samba to be a domain logon server
for
# Windows95 workstations.
;   domain logons = yes

# if you enable domain logons then you may want a per-
machine or
# per user logon script
# run a specific logon batch file per workstation (machine)
;   logon script = %m.bat
# run a specific logon batch file per username
```

```
;   logon script = %U.bat

# Where to store roving profiles (only for Win95 and WinNT)
#        %L substitutes for this servers netbios name, %U
is username
#        You must uncomment the [Profiles] share below
;   logon path = \\%L\Profiles\%U

# All NetBIOS names must be resolved to IP Addresses
# 'Name Resolve Order' allows the named resolution
mechanism to be specified
# the default order is "host lmhosts wins bcast". "host"
means use the unix
# system gethostbyname() function call that will use either
/etc/hosts OR
# DNS or NIS depending on the settings of /etc/host.config,
/etc/nsswitch.conf
# and the /etc/resolv.conf file. "host" therefore is system
configuration
# dependant. This parameter is most often of use to prevent
DNS lookups
# in order to resolve NetBIOS names to IP Addresses. Use
with care!
# The example below excludes use of name resolution for
machines that are
NOT
# on the local network segment
# - OR - are not deliberately to be known via lmhosts or
via WINS.
; name resolve order = wins lmhosts bcast

# Windows Internet Name Serving Support Section:
# WINS Support - Tells the NMBD component of Samba to
enable it's WINS
Server
;   wins support = yes

# WINS Server - Tells the NMBD components of Samba to be a
WINS Client
#     Note: Samba can be either a WINS Server, or a WINS
Client, but NOT both
;   wins server = w.x.y.z

# WINS Proxy - Tells Samba to answer name resolution
queries on
```

```
# behalf of a non WINS capable client, for this to work
there must be
# at least one     WINS Server on the network. The default
is NO.
;   wins proxy = yes

# DNS Proxy - tells Samba whether or not to try to resolve
NetBIOS names
# via DNS nslookups. The built-in default for versions
1.9.17 is yes,
# this has been changed in version 1.9.18 to no.
   dns proxy = no

# Case Preservation can be handy - system default is _no_
# NOTE: These can be set on a per share basis
;  preserve case = no
;  short preserve case = no
# Default case is normally upper case for all DOS files
;  default case = lower
# Be very careful with case sensitivity - it can break
things!
;  case sensitive = no

#============================ Share Definitions
==============================
   idmap uid = 16777216-33554431
   idmap gid = 16777216-33554431
   template shell = /bin/false
   winbind use default domain = no
[homes]
   comment = Home Directories
   browseable = no
   writable = yes

# Un-comment the following and create the netlogon
directory for Domain
Logons
; [netlogon]
;   comment = Network Logon Service
;   path = /home/netlogon
;   guest ok = yes
;   writable = no
;   share modes = no
```

```
# Un-comment the following to provide a specific roving
profile share
# the default is to use the user's home directory
;[Profiles]
;    path = /home/profiles
;    browseable = no
;    guest ok = yes


# NOTE: If you have a BSD-style print system there is no
need to
# specifically define each individual printer
[printers]
   comment = All Printers
   path = /var/spool/samba
   browseable = no
# Set public = yes to allow user 'guest account' to print
   guest ok = no
   writable = no
   printable = yes

# This one is useful for people to share files
;[tmp]
;   comment = Temporary file space
;   path = /tmp
;   read only = no
;   public = yes

# A publicly accessible directory, but read only, except
for people in
# the "staff" group
;[public]
;   comment = Public Stuff
;   path = /home/samba
;   public = yes
;   read only = yes
;   write list = @staff

# Other examples.
#
# A private printer, usable only by fred. Spool data will
be placed in
fred's
# home directory. Note that fred must have write access to
the spool
directory,
```

```
# wherever it is.
;[fredsprn]
;   comment = Fred's Printer
;   valid users = fred
;   path = /homes/fred
;   printer = freds_printer
;   public = no
;   writable = no
;   printable = yes

# A private directory, usable only by fred. Note that fred
requires write
# access to the directory.
;[fredsdir]
;   comment = Fred's Service
;   path = /usr/somewhere/private
;   valid users = fred
;   public = no
;   writable = yes
;   printable = no

# a service which has a different directory for each
machine that connects
# this allows you to tailor configurations to incoming
machines. You could
# also use the %u option to tailor it by user name.
# The %m gets replaced with the machine name that is
connecting.
;[pchome]
;  comment = PC Directories
;  path = /usr/pc/%m
;  public = no
;  writable = yes

# A publicly accessible directory, read/write to all users.
Note that all
files
# created in the directory by users will be owned by the
default user, so
# any user with access can delete any other user's files.
Obviously this
# directory must be writable by the default user. Another
user could of
course
# be specified, in which case all files would be owned by
that user instead.
```

```
;[public]
;   path = /usr/somewhere/else/public
;   public = yes
;   only guest = yes
;   writable = yes
;   printable = no

# The following two entries demonstrate how to share a
directory so that two
# users can place files there that will be owned by the
specific users. In
this
# setup, the directory should be writable by both users and
should have the
# sticky bit set on it to prevent abuse. Obviously this
could be extended to
# as many users as required.
;[myshare]
;   comment = Mary's and Fred's stuff
;   path = /usr/somewhere/shared
;   valid users = mary fred
;   public = no
;   writable = yes
;   printable = no
;   create mask = 0765


[compartida]
      comment = documents directory
      path = /home/discopublico
      writeable = yes
      printable = no
      browseable = yes
      guest ok = yes
```