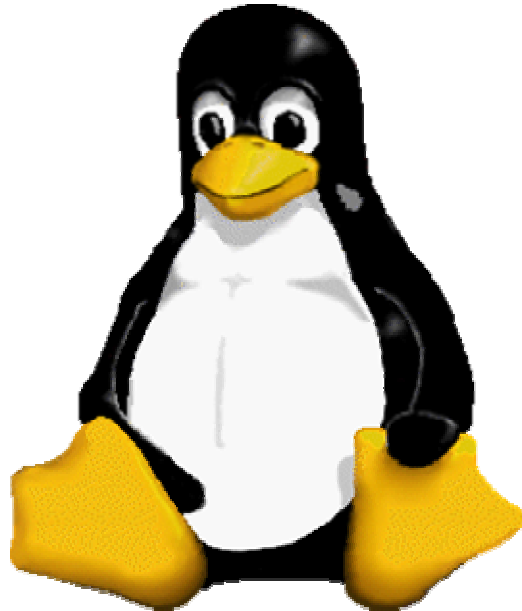


CURSO LINUX



Módulo Serviços Internet

por

Celso Kopp Webber

SUMÁRIO

| | | |
|----------|---|-----------|
| 1 | INTRODUÇÃO | 1 |
| 2 | INTERNET DAEMON (INETD) | 2 |
| 2.1 | Configurando o inetd | 2 |
| 2.2 | O Arquivo services | 4 |
| 2.3 | Reinicializando o inetd | 5 |
| 3 | FTP ANÔNIMO (ANONYMOUS FTP) | 7 |
| 3.1 | Criando um Servidor de FTP Anônimo | 8 |
| 4 | SERVIDOR WWW (WORLD WIDE WEB) | 10 |
| 4.1 | Configurando um Servidor WWW | 10 |
| 4.1.1 | httpd.conf | 11 |
| 4.1.2 | srm.conf | 15 |
| 4.1.3 | access.conf | 18 |
| 4.1.4 | mime.types | 19 |
| 5 | SENDMAIL | 20 |
| 5.1 | Opções | 20 |
| 5.2 | Aliases | 21 |
| 5.3 | Sendmail.cf | 22 |
| 5.3.1 | Estrutura Geral | 22 |
| 5.3.2 | Configuração do sendmail - Comandos | 23 |
| 5.3.2.1 | Comando de Definição de Macro - D | 24 |
| 5.3.2.2 | Condicionais | 25 |
| 5.3.2.3 | Comandos de Definição de Classe - C , F | 25 |
| 5.3.2.4 | Comando Para Setar Opção - O | 26 |
| 5.3.2.5 | Definindo Usuários Confiáveis - T | 27 |
| 5.3.2.6 | Definindo Precedência de Mail - P | 27 |

| | | |
|----------|--------------------------------------|-----------|
| 5.3.2.7 | Definindo Cabeçalhos - H | 28 |
| 5.3.2.8 | Definindo Mailers - M | 28 |
| 5.3.2.9 | Regras de Reescrita de Endereços - R | 30 |
| 5.3.2.10 | Comando Ruleset - S | 31 |
| 6 | EXERCÍCIOS | 32 |
| 7 | BIBLIOGRAFIA | 33 |

1 INTRODUÇÃO

Da mesma forma que um sistema operacional sem aplicativos úteis que rodem sobre ele perde seu valor, uma rede também precisa oferecer serviços.

A arquitetura TCP/IP, utilizada na Internet deve grande parte de seu sucesso à disponibilidade de serviços diversos, e também devido à facilidade de criar novos serviços e protocolos.

Em vários sistemas, alguns serviços possuem a natureza particular de serem utilizados apenas esporadicamente. Entretanto, manter um processo (*daemon*) em *background* rodando todo o tempo por um serviço raramente utilizado pode ser dispendioso quanto aos recursos da máquina, principalmente no que se refere à utilização de memória. Se considerarmos que existem vários serviços que se enquadram nesta situação, o resultado é um servidor com boa parte de sua memória utilizada, sem no entanto estar atendendo requisições pela rede que gerem esta carga. Os sistemas UNIX tradicionalmente resolvem este problema com um super servidor, chamado **inetd** (Internet *daemon*).

Com o **inetd**, ao invés de manter rodando diversos serviços na máquina, um único processo, o **inetd**, fica escutando por conexões vindas da rede nas portas específicas dos serviços implementados (ex: porta 23 para o **telnet**, porta 21 para o **ftp**, etc.). Assim que um pedido de conexão chega, o **inetd** dispara o *daemon* correspondente ao serviço, repassando a ele a conexão recém-aceita.

Em outros casos, quando temos serviços que são utilizados com frequência, é necessário manter os *daemons* correspondentes rodando o tempo na máquina, de forma independente do **inetd**. Nestes casos o **inetd** é configurado para não escutar na porta específica do serviço, e o *daemon* do serviço é executado independentemente, sendo responsável por escutar e aceitar suas próprias conexões em sua porta específica.

Este o caso de boa parte dos servidores **www**, que normalmente devem responder rapidamente a requisições por uma página na rede, e o *overhead* de dispara um novo processo via **inetd** a cada pedido por uma página ou figura de uma página, tornaria o serviço lento demais nos tempos de resposta desejados.

Mesmo sistemas operacionais não-UNIX seguem uma abordagem semelhante (vide o caso do Microsoft Windows NT), onde um super servidor é responsável por atender os diversos serviços mais simples, e os serviços mais complexos são atendidos por processos executados de forma independente.

Veremos a seguir como configurar o **inetd**, bem como configurar os principais serviços usados em uma rede TCP/IP: **FTP e FTP Anônimo**, **Telnet**, **WWW** (World Wide Web) e **E-mail**.

2 INTERNET DAEMON (INETD)

inetd é um *daemon* que gerencia outros *daemons*. Ele dispara os *daemons* clientes quanto recebe solicitações de seus serviços e permite que eles terminem sua execução quando acabarem de atender aos serviços solicitados. Para trabalhar sob a direção do **inetd**, os clientes devem observar algumas convenções especiais.

O **inetd** trabalha apenas com *daemons* que fornecem serviços para a rede. A fim de descobrir quando alguém está tentando acessar algum de seus clientes, o **inetd** se associa às portas de rede que normalmente seriam gerenciadas pelos *daemons* clientes (que agora estão inativos). Quando ocorre uma conexão, o **inetd** dispara o *daemon* apropriado e conecta seus canais de I/O padrão à porta de rede.

A maioria dos *daemons* pode ser usada tanto na forma tradicional, na qual eles são disparados uma única vez e continuam a executar até que o sistema seja desligado (*shutdown*), ou com o **inetd**. Na verdade, a maioria dos *daemons* que prestam serviços de rede são disparados pelo **inetd**. O **telnetd**, servidor que suporta o protocolo de terminal virtual TELNET, é um exemplo. A configuração necessária para se disponibilizar um serviço via **inetd** é feita num único arquivo, o **/etc/inetd.conf**.

2.1 Configurando o inetd

O **inetd** usa um arquivo de configuração (geralmente **/etc/inetd.conf**, mas algumas vezes pode ser **/usr/etc/inetd.conf** ou **/etc/servers**) que determina quais são as portas de rede que ele deve gerenciar. O formato deste arquivo é o mesmo em todas as plataformas:

| # | comentarios | | | | | |
|------|-------------|-----------|------|---------|--------|------------|
| nome | tipo | protocolo | modo | usuario | daemon | argumentos |

Observe que no arquivo **inetd.conf**, linhas que começam com o caractere cerquilha (#) são consideradas comentários. Da mesma forma, quando o caractere cerquilha é encontrado em uma linha, daquele ponto até o final da linha é também considerado um comentário.

Todos os campos são obrigatórios. Observe que o parâmetro argumentos especifica um argumento a ser passado para o *daemon*. Nem todos os *daemons* são construídos para aceitarem argumentos, ou às vezes os argumentos aceitos servem apenas para depuração do programa. Como o campo é obrigatório, sempre que não for necessário passar argumento algum para o *daemon*, utiliza-se como argumento o próprio nome do *daemon*, sem incluir o caminho de diretórios para o executável.

Vejamos um exemplo para o servidor **telnet** e outros serviços:

```
# /etc/inetd.conf - Configuração do daemon inetd.
# Execute kill -1 <pid-do-inted> apos modificar este arquivo.
ftp      stream  tcp      nowait  root    /usr/sbin/in.ftp      -l -a
telnet   stream  tcp      nowait  root    /usr/sbin/in.telnetd  in.telnetd
mountd/1 stream  rpc/tcp  wait    root    /etc/mountd mountd
mountd/1 dgram   rpc/udp  wait    root    /etc/mountd mountd
```

Os significados dos campos de uma entrada do **inetd.conf**, da esquerda para a direita, são descritos a seguir:

| Campo | Significado |
|-------------|---|
| nome | O nome de um serviço. Nomes de serviços são mapeados para números de porta, através do arquivo /etc/services (para serviços TCP e UDP) ou do <i>daemon portmap</i> (para serviços que usam RPC). Serviços RPC (<i>Remote Procedure Call</i>) são identificados por nomes na forma nome/número e pela designação rpc na terceira coluna. No exemplo apresentado anteriormente, as últimas duas linhas são serviços RPC. |
| tipo | Determina o tipo de socket que o serviço vai usar: stream , dgram ou raw . Em geral, stream é usado com serviços TCP (orientados a conexão), dgram é usado com serviços UDP (serviços não orientados a conexão) e raw para serviços de datagramas IP diretos (este tipo raramente é visto). O exemplo anterior mostra que o telnetd usa um <i>socket</i> do tipo stream . |
| protocolo | Identifica o protocolo de comunicação usado pelo serviço. Os tipos disponíveis estão listados no arquivo /etc/protocols . O protocolo é quase sempre tcp ou udp . Serviços RPC acrescentam rpc/ antes do tipo do protocolo, como mostrado nas últimas duas linhas do exemplo. |
| modo | O valor deste campo pode ser tanto wait quanto nowait . Geralmente, servidores que usam serviços de datagrama requerem wait , e servidores que usam serviços de stream requerem nowait . Se o modo é wait , o inetd deve esperar que o servidor libere o <i>socket</i> antes de esperar por mais conexões com este <i>socket</i> . Se o modo é nowait , o inetd pode esperar por mais pedidos de conexão com este <i>socket</i> imediatamente. |
| usuario | Identifica o nome do usuário que é responsável pela execução do <i>daemon</i> . No exemplo, o telnetd está executando como root . |
| daemon | Identifica o caminho completo do <i>daemon</i> que será disparado pelo inetd . No exemplo, o telnetd se encontra no diretório /usr/sbin . |
| argumento s | Determina os argumentos que devem ser passados para o <i>daemon</i> quando ele é disparado. A lista de argumentos válidos para cada programa está documentada nas <i>man pages</i> dos programas. No exemplo, apenas o nome do <i>daemon</i> (telnetd) é usado. |

O arquivo **/etc/protocols** é consultado para que o sistema saiba o número correspondente ao nome especificado no terceiro campo do arquivo **/etc/inetd.conf**. Este arquivo raramente precisa ser modificado, pois protocolos neste nível, como TCP e UDP, dificilmente são acrescentados. Assim, os sistemas UNIX já instalam um arquivo **/etc/protocols** adequado, como mostra o exemplo:

| | | | |
|-------|----|-------|---|
| ip | 0 | IP | # internet protocol, pseudo protocol number |
| icmp | 1 | ICMP | # internet control message protocol |
| igmp | 2 | IGMP | # Internet Group Management |
| tcp | 6 | TCP | # transmission control protocol |
| egp | 8 | EGP | # exterior gateway protocol |
| udp | 17 | UDP | # user datagram protocol |
| ospf | 89 | OSPF | # Open Shortest Path First IGP |
| ipip | 94 | IPIP | # Yet Another IP encapsulation |
| encap | 98 | ENCAP | # Yet Another IP encapsulation |

2.2 O Arquivo services

Para identificar as portas associadas a um determinado nome (1º campo do arquivo **inetd.conf**) de serviço, um arquivo foi definido para este mapeamento, estando localizado em **/etc/services**. É interessante notar que outros sistemas operacionais também usam este mesmo arquivo para mapear nomes de serviços a portas específicas. No Microsoft Windows NT, por exemplo, este arquivo está em **%SystemRoot%\SYSTEM32\DRIVERS\ETC**, onde **%SystemRoot%** é a variável de ambiente que o Windows NT ajusta para apontar para o diretório base do sistema operacional, geralmente **C:\WINNT**. Neste mesmo diretório, o Windows NT mantém os arquivos **hosts**, **networks** e **protocols**, equivalentes aos do UNIX, geralmente localizados sob o diretório **/etc**.

Cada entrada do arquivo **/etc/services** possui o seguinte formato:

| nome | porta/protocolo | sinonimo1 | sinonimo2 | ... | # comentarios |
|------|-----------------|-----------|-----------|-----|---------------|
|------|-----------------|-----------|-----------|-----|---------------|

Note que neste arquivo, da mesma forma que em outros arquivos UNIX, textos precedidos de cerquilha (#) são comentários.

Este arquivo é usado por várias rotinas de bibliotecas que mapeiam nomes de serviços em números de portas. Por exemplo, quando você digita o comando:

```
frajola$ telnet piupiu smtp
```

o número da porta para o serviço **smtp** é procurado no arquivo **services**. Em seguida o comando **telnet** irá fazer uma conexão na porta correspondente (neste caso 25, como mostra o exemplo abaixo), com a máquina **piupiu**. A maioria dos sistemas já vem configurado com os serviços mais usados. Desta forma, você precisará editar o arquivo **services** apenas se adicionar um novo serviço.

O arquivo **services** é usado apenas para registrar serviços TCP/IP. Informações similares para serviços baseados em RPC são registradas em um arquivo de configuração separado, chamado **/etc/rpc**:

| # /etc/rpc - Define mapeamentos de portas RPC para nomes | | | |
|--|--------|---------|-----------|
| nfs | 100003 | nfsprog | |
| ypserv | 100004 | ypprog | |
| mountd | 100005 | mount | showmount |
| yplib | 100007 | | |

Vejamos um exemplo do arquivo **/etc/services**:

| | | | |
|----------|--------|------------|----------------------|
| ftp-data | 20/tcp | | |
| ftp | 21/tcp | | |
| telnet | 23/tcp | | |
| smtp | 25/tcp | mail | |
| domain | 53/tcp | nameserver | # name-domain server |
| domain | 53/udp | nameserver | |
| www | 80/tcp | http | # WorldWideWeb HTTP |

| | | | |
|-------------|---------|------------|-----------------------------|
| www | 80/udp | | |
| pop-3 | 110/tcp | | # POP version 3 |
| pop-3 | 110/udp | | |
| sunrpc | 111/tcp | portmapper | # RPC 4.0 portmapper TCP |
| sunrpc | 111/udp | portmapper | # RPC 4.0 portmapper UDP |
| netbios-ns | 137/tcp | | # NETBIOS Name Service |
| netbios-ns | 137/udp | | |
| netbios-dgm | 138/tcp | | # NETBIOS Datagram Service |
| netbios-dgm | 138/udp | | |
| netbios-ssn | 139/tcp | | # NETBIOS session service |
| netbios-ssn | 139/udp | | |
| imap2 | 143/tcp | imap | # Interim Mail Access Proto |
| imap2 | 143/udp | imap | |
| snmp | 161/udp | | # Simple Net Mgmt Proto |
| snmp-trap | 162/udp | snmptrap | # Traps for SNMP |
| cmip-man | 163/tcp | | # ISO mgmt over IP (CMOT) |
| cmip-man | 163/udp | | |
| cmip-agent | 164/tcp | | |
| cmip-agent | 164/udp | | |
| https | 443/tcp | | # MCom |
| https | 443/udp | | # MCom |

O significado dos campos é apresentado a seguir:

| Campo | Significado |
|------------|---|
| nome | Indica o nome simbólico do serviço (o nome que usado no arquivo inetd.conf). |
| porta | O número da porta pela qual o serviço responde aos pedidos de conexão. Se o serviço é gerenciado pelo inetd , esta é a porta que o inetd vai esperar os pedidos de conexões. Os números das portas não são arbitrários. Todas as máquinas da rede devem concordar sobre quais serviços devem estar associados a quais portas, caso contrário a requisição de serviço será direcionada para a porta errada. Se você está criando um serviço específico do seu <i>site</i> , escolha um número bem alto para a porta (na ordem de milhares), que não esteja listado no arquivo services (portas de número menor que 1024 estão reservadas para o sistema). |
| protocolo | Indica o protocolo usado pelo serviço (na prática, é sempre tcp ou udp). Se um serviço usa tanto TCP quanto UDP, deve ser incluída uma linha para cada configuração (no exemplo, o serviço time ilustra bem esta situação). |
| sinonimos | Especifica nomes adicionais para um serviço (no exemplo, o serviço smtp pode ser referenciado também como mail). |
| comentário | Usado para fazer qualquer comentário sobre o serviço. |

2.3 Reinicializando o inetd

As mudanças feitas no arquivo de configuração (**inetd.conf**) não têm efeito até que o **inetd** leia novamente este arquivo. Você pode fazer com que o **inetd** leia o arquivo de configuração

mandando um sinal de *hangup* (`kill -HUP inetd_pid`) para ele. Recebendo este sinal, o **inetd** irá reler o arquivo e efetuar as mudanças. Você deve inspecionar os arquivos de *log* a fim de verificar se as suas mudanças não produziram nenhum erro (o **inetd** reporta os erros através do **syslog**). Finalmente, você deve testar cada um dos novos serviços para ter certeza de que estão funcionando corretamente.

O exemplo ilustra o **inetd** sendo reinicializado. Observe que o *PID* do **inetd** foi obtido com o comando **ps**:

```
[root@piupiu /etc]# ps -ax |grep inetd
 357 ?          S          0:00 inetd
1009 pts/0     S          0:00 grep inetd
[root@piupiu /etc]# kill -1 357
```

3 FTP ANÔNIMO (ANONYMOUS FTP)

O comando `ftp` provê uma maneira simples de transferir arquivos na rede. Muitas vezes é necessário ter uma conta na máquina remota para poder copiar arquivos, mas muitos servidores de `ftp` permitem que você se conecte usando o nome “**anonymous**”, ao invés de seu nome de usuário (nome de *login*) atual. Na maioria dos *sites*, qualquer senha é aceita; é de costume usar o endereço de *e-mail* como senha, para que o administrador do *site* saiba quem está acessando o servidor. Alguns servidores exigem que o endereço de *e-mail* fornecido como senha seja um endereço válido (possa ser verificado via SMTP). `ftp` anônimo é uma das maneiras mais comuns para se distribuir programas e documentos na Internet.

O exemplo a seguir ilustra uma sessão `ftp` com o servidor `mercurio.inf.ufsc.br`.

```
mercurio% ftp mercurio.inf.ufsc.br
Connected to mercurio.inf.ufsc.br.
220 mercurio FTP server (ftpd-wu Version 6.14) ready.
Name (mercurio.inf.ufsc.br:marchez): anonymous
331 Guest login ok, send e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
ftp> cd pub
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 18
dr-xr-xr-x  2 ftp      daemon      512 Apr  2 14:57 IR
dr-xr-xr-x  7 labcas  daemon      512 Oct 27 1995 LabCAS
drwxr-xr-x 13 ftp      daemon     1536 May 29 14:11 contrib
dr-xr-xr-x  5 ftp      daemon      512 Apr  3 14:10 doc
drwxr-xr-x  2 raul    daemon      512 May 31 19:11 ex-alunos
-rw-r--r--  1 root    daemon     6595 Mar 25 12:07 ls-lR.gz
dr-xr-xr-x  2 ftp      daemon      512 Oct 19 1995 os2
dr-xr-xr-x  5 ftp      daemon      512 Sep  6 1995 pc
drwxrwxr-x  3 secchi  pet         512 May 27 18:14 pet
drwxr-xr-x  2 cpgcc   daemon      512 Feb 12 20:35 pos-graduacao
dr-xr-xr-x 12 ftp      daemon      512 Mar 18 22:42 unix
226 Transfer complete.
686 bytes received in 0.13 seconds (5.3 Kbytes/s)
ftp> bin
200 Type set to I.
ftp> get ls-lR.gz
200 PORT command successful.
150 Opening BINARY mode data connection for ls-lR.gz (6595 bytes).
226 Transfer complete.
local: ls-lR.gz remote: ls-lR.gz
6595 bytes received in 0.042 seconds (1.5e+02 Kbytes/s)
ftp> bye
221 Goodbye.
```

Figura 3.1 - Exemplo de uma sessão de `ftp` anônimo

Neste exemplo, o usuário conectou-se ao servidor `mercurio.inf.ufsc.br` usando `anonymous` como nome de usuário e `marchez@inf.ufsc.br` como senha, que é seu real endereço de e-mail. Com FTP anônimo ele pode conectar-se, mesmo sem ter uma conta no servidor `mercurio.inf.ufsc.br`. Suas ações estão limitadas, mas ele pode copiar certos arquivos do sistema. Ele passou para o diretório `pub` e copiou o arquivo `pub/ls-lR.gz`. O arquivo foi copiado no modo binário, pois é um arquivo compactado.

3.1 Criando um Servidor de FTP Anônimo

A fim de compartilhar seus programas via **ftp** anônimo, você deve criar uma conta para o usuário **ftp**, configurar seu diretório *home* e o *daemon ftp*, o **ftpd**.

O **ftpd** é gerenciado pelo **inetd** e, portanto, deve existir uma entrada com sua configuração nos arquivos **/etc/inetd.conf** e **/etc/services**. Quando é feita uma conexão com o **ftpd**, ele faz uma chamada de sistema **chroot** para que os arquivos que estão fora do caminho **~ftp** fiquem invisíveis e inacessíveis ao usuário que acaba de se conectar. Este procedimento aumenta a segurança, pois o **ftpd** deve executar como **root** para poder manipular portas privilegiadas.

Para criar um servidor de FTP anônimo, siga os passos:

- Adicione o usuário **ftp** ao arquivo **/etc/passwd** (seu arquivo de senhas);
- Crie um diretório *home* para o **ftp** tendo como dono o próprio usuário **ftp** ou o usuário **root**. Este diretório não deve ter permissão de escrita para ninguém;
- Crie um diretório **bin**, abaixo do diretório **ftp**, tendo como dono o usuário **root** e sem permissão de escrita para ninguém. O comando **ls** deve ser copiado para este diretório e suas permissões mudadas para 111 (apenas execução para todos);
- Se o programa **ls** que você está copiando é ligado dinamicamente com as bibliotecas de sistema, crie também um diretório **lib**, abaixo do diretório **ftp**, com as mesmas permissões do diretório **bin**. Neste diretório copie as bibliotecas utilizadas pelo programa **ls**. As bibliotecas geralmente utilizadas pelos programas de forma dinâmica, são a **libc** (**/lib/libc-*.so** e **/lib/libc.so.***), e a **ld** (**/lib/ld-***).
- Crie um diretório **etc**, abaixo do diretório **ftp**, tendo como dono o usuário **root** e sem permissão de escrita para ninguém. Crie os arquivos **passwd** e **group** neste diretório e mude suas permissões para 444 (apenas leitura para todos);
- Crie um diretório **pub**, abaixo do diretório **ftp**, tendo como dono o usuário **root** e mude suas permissões para 555 (leitura, execução para todos). Este é o diretório onde os arquivos serão disponibilizados para os usuários.
- Se você quiser que os usuários possam armazenar arquivos em seu servidor, crie um diretório para isso, geralmente chamado **contrib** ou **incoming**, e com as permissões 777 (leitura, escrita, execução para todos).

O exemplo que segue mostra cada um destes passos. Primeiro crie o diretório *home ftp* e todos os sub-diretórios. Em nosso exemplo, nós criamos o diretório **ftp** abaixo do diretório **/home/mercurio**.

```
mercurio# mkdir /home/mercurio/ftp
mercurio# cd /home/mercurio/ftp
mercurio# mkdir bin
mercurio# mkdir etc
mercurio# mkdir pub
```

Depois copie o arquivo **ls** para o diretório **/home/mercurio/ftp/bin** e mude as permissões para 111 (apenas execução para todos). Estas permissões aumentam a segurança, pois proibem que os usuários copiem o arquivo binário.

```
mercurio# cp /bin/ls /home/mercurio/ftp/bin
mercurio# chmod 111 /home/mercurio/ftp/bin/ls
```

Em seguida crie um grupo que será usado apenas pelo **ftp** anônimo. Nenhum outro usuário

deve fazer parte deste grupo. No exemplo, criamos o arquivo chamado `/home/mercurio/ftp/etc/group` que contém uma única entrada.

```
anonymous:*:15:
```

Crie um usuário chamado `ftp`, adicionando uma entrada para este usuário no arquivo `/etc/passwd`. Especifique como `shell` um arquivo que não exista (no exemplo, `/bin/false`) para este usuário, por questões de segurança. Nem todos os daemons aceitam um `shell` inválido, como o `/bin/false`. Na dúvida é sempre bom fazer um teste. Crie também um arquivo chamado `/home/mercurio/ftp/etc/passwd` que contém apenas a entrada `ftp`. A entrada que deve ser adicionada em ambos os arquivos é:

```
ftp:*:15:15:Anonymous ftp:/home/mercurio/ftp:/bin/false
```

Estes exemplos usaram o número 15 para identificar o `gid` e o `uid`, mas são apenas exemplos. Você deve escolher um `gid` e um `uid` que não esteja sendo usado por ninguém mais no seu sistema. Após ter criado os dois arquivos mude suas permissões para 444.

```
mercurio# chmod 444 /home/mercurio/ftp/etc/passwd
mercurio# chmod 444 /home/mercurio/ftp/etc/group
```

Mude o nome do dono e as permissões para cada um dos diretórios criados. A informação de dono dos diretórios `/home/mercurio/ftp/etc` e `/home/mercurio/ftp/bin` não precisa ser alterada, pois estes diretórios foram criados pelo usuário `root`.

```
mercurio# cd /home/mercurio/ftp
mercurio# chown ftp pub
mercurio# chmod 555 pub
mercurio# chmod 555 bin
mercurio# chmod 555 etc
mercurio# cd ..
mercurio# chown ftp ftp
mercurio# chmod 555 ftp
```

Neste momento, você pode copiar os arquivos que deseja disponibilizar para o diretório `/home/mercurio/ftp/pub`. Para prevenir que estes arquivos seja removidos por usuários remotos, mude suas permissões para 644 e certifique-se que eles não pertencem ao usuário `ftp`.

Após fazer todas as configurações necessárias para seu sistema, teste o novo serviço antes de anunciá-lo. Verifique se seu servidor provê o serviço de `ftp` anônimo, sem fornecer “serviços” adicionais que você não deseja (como permitir que usuários anônimos acessem arquivos que estejam fora do diretório `home` do `ftp`). `ftp` anônimo pode apresentar riscos potenciais de segurança, por isso é necessário bastante cuidado na sua instalação.

4 SERVIDOR WWW (WORLD WIDE WEB)

O *daemon* `httpd` permite que seu *site* se torne um servidor WWW de hiperdocumentos. O `httpd` é responsável por atender às requisições dos clientes, e pode lhes enviar textos, figuras e sons. O `httpd` pode executar na forma tradicional (*standalone*) ou como um processo gerenciado pelo `inetd` (é a forma mais utilizada para disparar o `httpd`).

4.1 Configurando um Servidor WWW

Antes de mais nada é necessário que você obtenha algum servidor WWW e o descompacte no diretório que melhor lhe convier. Os servidores estão disponíveis na Internet tanto na versão compilada quanto na versão não compilada (o código fonte está incluído e deve ser compilado). Note que nem todos os desenvolvedores de servidores disponibilizam o código fonte; neste caso, procure por uma versão compilada.

Basicamente todos os servidores têm um conjunto de diretórios e arquivos que são iguais ou, pelo menos, exercem a mesma função. Após a descompactação do servidor é comum obter um conjunto de diretórios que inclui o `cgi-bin`, `cgi-src`, `conf`, `icons`, `logs` e `support`.

A seguir, temos a descrição do conteúdo de cada um destes diretórios.

| Diretório | Conteúdo |
|----------------------|--|
| <code>cgi-bin</code> | Neste diretório ficam todos os <i>scripts</i> CGI (<i>Common Gateway Interface</i>) que serão referenciados pelos clientes. Cada novo <i>script</i> criado pelo administrador do serviço deve ser colocado neste diretório. |
| <code>cgi-src</code> | Contém o código fonte dos <i>scripts</i> CGI disponíveis junto com o servidor. |
| <code>conf</code> | Este é o diretório que contém todos os arquivos de configuração do servidor (os mais importantes). São eles: <code>httpd.conf-dist</code> , <code>access.conf-dist</code> , <code>srn.conf-dist</code> , <code>mime.types</code> . Os arquivos com a extensão <code>-dist</code> devem ser copiados para <code>httpd.conf</code> , <code>access.conf</code> , <code>srn.conf</code> , respectivamente, antes de serem alterados. |
| <code>icons</code> | Contém as imagens que são usadas pelo servidor para formar páginas HTML de aviso aos clientes (geralmente quando acontecem erros), entre outros. |
| <code>logs</code> | Este diretório contém os arquivos que catalogam os acessos feitos ao servidor e os erros que ocorrem na interação dos clientes com o servidor. Estes arquivos são úteis para que o <i>webmaster</i> (administrador do serviço) possa monitorar o funcionamento do serviço. No arquivo <code>access_log</code> estão catalogados todos os acessos e atitudes dos clientes. No arquivo <code>error_log</code> estão as mensagens de erros provocados pelos clientes na tentativas de interagir e navegar pelas páginas do servidor. Através das mensagens deste arquivo pode-se também descobrir motivos para o mau funcionamento do servidor. A maioria dos servidores possui um arquivo chamado <code>httpd.pid</code> , no qual é cadastrado o <i>pid</i> do <code>httpd</code> (na hora em que ele é disparado). O <i>pid</i> pode ser usado para que o administrador reinicialize ou termine a execução do <i>daemon</i> . Um sinal de <i>hangup</i> (<code>kill -HUP pid</code>) faz com que o <i>daemon</i> releia os |

| | |
|---------|--|
| | arquivos de configuração. |
| support | Neste diretório existem alguns programas que dão suporte para a manutenção do serviço. Eles fazem uso de arquivos, como access_log e error_log para extrair informações úteis ao <i>webmaster</i> no gerenciamento do sistema. Geralmente apresentam alguns dados estatísticos sobre o funcionamento do serviço, permitindo que o <i>webmaster</i> tome atitudes para melhorar o seu desempenho e uso. |

Tabela 4.1 - Conteúdo dos diretórios comuns dos vários servidores WWW

Atenção especial tem que ser dada aos arquivos de configuração do servidor WWW, localizados no diretório **conf**. Os itens 4.1.1 a 4.1.4 descrevem os arquivos de configuração e as opções mais comuns a serem configuradas em cada um.

4.1.1 httpd.conf

Este é o principal arquivo de configuração do servidor. Ele contém diretivas que controlam a operação do *daemon*. Vejamos o conteúdo do **httpd.conf** que acompanha o servidor **Apache** (o mais usado na Internet atualmente).

```
# This is the main server configuration file. See URL http://www.apache.org/
# for instructions.

# Do NOT simply read the instructions in here without understanding
# what they do, if you are unsure consult the online docs. You have been
# warned.

# Originally by Rob McCool

# ServerType is either inetd, or standalone.
ServerType standalone

# If you are running from inetd, go to "ServerAdmin".

# Port: The port the standalone listens to. For ports < 1023, you will
# need httpd to be run as root initially.
Port 80

# 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
User nobody
Group #-1

# ServerAdmin: Your address, where problems with the server should be
# e-mailed.
ServerAdmin you@your.address

# ServerRoot: The directory the server's config, error, and log files
# are kept in
ServerRoot /usr/local/etc/httpd

# BindAddress: You can support virtual hosts with this option. This option
# is used to tell the server which IP address to listen to. It can either
# contain "*", an IP address, or a fully qualified Internet domain name.
# See also the VirtualHost directive.

#BindAddress *

# ErrorLog: The location of the error log file. If this does not start
# with /, ServerRoot is prepended to it.
```

```
ErrorLog logs/error_log

# TransferLog: The location of the transfer log file. If this does not
# start with /, ServerRoot is prepended to it.
TransferLog logs/access_log

# PidFile: The file the server should log its pid to
PidFile logs/httpd.pid

# ScoreBoardFile: The file used for temporary internal data about the server
ScoreBoardFile logs/apache_runtime_status

# ServerName allows you to set a host name which is sent back to clients for
# your server if it's different than the one the program would get (i.e. use
# "www" instead of the host's real name).
#
# Note: You cannot just invent host names and hope they work. The name you
# define here must be a valid DNS name for your host. If you don't understand
# this, ask your network administrator.

#ServerName new.host.name

# CacheNegotiatedDocs: By default, Apache sends Pragma: no-cache with each
# document that was negotiated on the basis of content. This asks proxy
# servers not to cache the document. Uncommenting the following line disables
# this behavior, and proxies will be allowed to cache the documents.

#CacheNegotiatedDocs

# Timeout: The number of seconds before receives and sends time out
# n.b. the compiled default is 1200 (20 minutes !)
Timeout 400

# Server-pool size regulation. Rather than making you guess how many
# server processes you need, Apache dynamically adapts to the load it
# sees --- that is, it tries to maintain enough server processes to
# handle the current load, plus a few spare servers to handle transient
# load spikes (e.g., multiple simultaneous requests from a single
# Netscape browser).

# It does this by periodically checking how many servers are waiting
# for a request. If there are fewer than MinSpareServers, it creates
# a new spare. If there are more than MaxSpareServers, some of the
# spares die off. These values are probably OK for most sites ---
MinSpareServers 5
MaxSpareServers 10

# Number of servers to start --- should be a reasonable ballpark figure.
StartServers 5

# Limit on total number of servers running, i.e., limit on the number
# of clients who can simultaneously connect --- if this limit is ever
# reached, clients will be LOCKED OUT, so it should NOT BE SET TOO LOW.
# It is intended mainly as a brake to keep a runaway server from taking
# Unix with it as it spirals down...

MaxClients 150

# MaxRequestsPerChild: the number of requests each child process is
# allowed to process before the child dies.
# The child will exit so as to avoid problems after prolonged use when
# Apache (and maybe the libraries it uses) leak. On most systems, this
# isn't really needed, but a few (such as Solaris) do have notable leaks
# in the libraries.
MaxRequestsPerChild 30

# VirtualHost: Allows the daemon to respond to requests for more than one
# server address, if your server machine is configured to accept IP packets
# for multiple addresses. This can be accomplished with the ifconfig
# alias flag, or through kernel patches like VIF.

# Any httpd.conf or srm.conf directive may go into a VirtualHost command.
```

```
# See also the BindAddress entry.

#<VirtualHost host.foo.com>
#ServerAdmin webmaster@host.foo.com
#DocumentRoot /www/docs/host.foo.com
#ServerName host.foo.com
#ErrorLog logs/host.foo.com-error_log
#TransferLog logs/host.foo.com-access_log
#</VirtualHost>
```

Figura 4.1 - Exemplo do arquivo **httpd.conf** distribuído com o servidor **Apache**

As diretivas básicas que devem ser configuradas são:

- **ServerType**

Sintaxe: *ServerType tipo*

Esta diretiva indica o modo no qual o servidor será executado pelo sistema. O tipo pode ser:

inetd - O servidor vai ser executado a partir do **inetd**. Neste caso deve-se incluir no arquivo **inetd.conf** a seguinte entrada (a sintaxe das entradas do **inet.conf** foi exposta no capítulo 2):

```
http stream tcp nowait nobody /usr/local/www/apache/httpd httpd -d
/usr/local/www/apache
```

Deve-se também incluir no arquivo **services** a seguinte entrada (a sintaxe das entradas foi exposta no capítulo 2):

```
http 80/tcp # Servidor WWW
```

Após terminar a configuração do servidor, não esqueça de fazer com que o **inetd** releia o arquivo de configuração (kill -HUP *inetd_pid*).

standalone - O servidor vai executar na forma tradicional. O comando para executar o servidor deve ser colocado em algum script de inicialização do sistema, por exemplo, **/etc/rc.local** ou **/etc/rc3.d/...** (para garantir que o serviço estará disponível mesmo que a máquina seja reinicializada).

A opção *inetd* é menos usada, pois para cada conexão **http** uma nova cópia do servidor é criada, terminando sua execução quando a conexão é completada. Paga-se um preço alto por cada nova conexão, mas por questões de segurança alguns administradores preferem esta opção. A opção *standalone* é mais eficiente, pois o servidores é inicializado apenas uma vez. Se você pretende instalar o servidor em um *site* com muitos acessos, a opção *standalone* é mais adequada.

- **Port**

Sintaxe: *Port número*

Esta diretiva configura a porta de rede na qual o servidor vai esperar as conexões. O número varia de 0 a 65535. Algumas portas (abaixo de 1024) são reservadas para protocolos particulares. A porta padrão para o **http** é 80, que é uma porta especial. Estas portas só podem ser usadas por processos que têm como dono o usuário **root**. Os usuários regulares (que não o **root**) devem usar portas acima de 1024. Para usar a porta 80 você deve inicializar o servidor via conta do **root**. Após ter se associado à porta e antes de aceitar requisições, o servidor irá mudar seu dono para um usuário de menor privilégio, definido pela diretiva **User**. Definir **User** como **root** pode ser uma falha de segurança.

- **User**

Sintaxe: *User uid*

Esta diretiva define os privilégios que o **httpd** terá durante sua execução (os privilégios serão iguais aos do usuário *uid*). *uid* pode ser tanto o nome de um usuário quanto o seu número de identificação (precedido do caracter #). Se você executar o servidor como **root**, não esqueça de configurar a diretiva *User* com um nome de usuário diferente de **root**, caso contrário o **httpd** estará executando com privilégios de **superusuário**.

- **Group**

Sintaxe: *Group grupo*

Esta diretiva define o grupo sob o qual o servidor irá responder às requisições. O *grupo* pode ser tanto o nome de um grupo quanto o seu número de identificação (precedido do caracter #). Alguns administradores usam o grupo **nobody**, mas nem sempre é possível ou desejado.

- **ServerAdmin**

Sintaxe: *ServerAdmin e-mail*

Esta diretiva define o endereço e-mail que o servidor incluirá nas mensagens de erro que ele retorna para o cliente.

- **ServerRoot**

Sintaxe: *ServerRoot diretório*

Esta diretiva informa o diretório onde o servidor está. Geralmente, os sub-diretórios **conf/** e **logs/** estão presentes neste diretório. Caminhos relativos para outros arquivos são considerados relativos a este diretório.

- **ErrorLog**

Sintaxe: *ErrorLog nome_do_arquivo*

Esta primitiva define o nome do arquivo que o servidor usará para cadastrar as mensagens de erro. Se *nome_do_arquivo* não começar com uma barra (/), então será assumido que ele é relativo ao caminho definido em *ServerRoot*.

- **TransferLog**

Sintaxe: *TransferLog nome_do_arquivo*

Esta primitiva define o nome do arquivo que o servidor usará para cadastrar as mensagens produzidas durante a navegação do cliente. Informações de recursos acessados, data e hora de um acesso específico são exemplos de mensagens cadastradas neste arquivo. Se *nome_do_arquivo* não começar com uma barra (/), então será assumido que ele é relativo ao caminho definido em *ServerRoot*.

- **PidFile**

Sintaxe: *PidFile nome_do_arquivo*

Esta primitiva define o nome do arquivo que o servidor usará para cadastrar o **pid** do *daemon*. Se *nome_do_arquivo* não começar com uma barra (/), então será assumido que ele é relativo ao caminho definido em *ServerRoot*. Esta diretiva só é usada se estivermos executando o servidor no modo *standalone*.

- **ServerName**

Sintaxe: *ServerName nome_domínio_completo*

Esta diretiva define o nome do *host* no qual o servidor está executando. *nome_domínio_completo* deve ser um nome cadastrado no **DNS** do sistema. Se não for especificado, o servidor tenta deduzi-lo a partir de seu próprio endereço IP. Por exemplo,

www.inf.ufsc.br poderia ser usado como nome fictício, ao invés do nome real mercurio.inf.ufsc.br.

- **MinSpareServers**

Sintaxe: *MinSpareServers número*

Esta diretiva define o número mínimo de filhos do processo servidor que devem ficar desocupados, esperando conexões. Se num determinado instante existirem menos do que *número* processos desocupados, o processo pai cria novos filhos, a uma taxa máxima de 1 por segundo.

- **MaxSpareServers**

Sintaxe: *MaxSpareServers número*

Esta diretiva define o número máximo de filhos do processo servidor que devem ficar desocupados, esperando conexões. Se num determinado instante existirem mais do que *número* processos desocupados, o processo pai irá matar os excedentes.

- **StartServers**

Sintaxe: *StartServers número*

Esta diretiva define o número de processos filhos que serão criados quando o servidor for inicializado. Como o número de processos é controlado dinamicamente, dependendo da demanda, não existe razões muito fortes para se preocupar com esta diretiva.

- **MaxClients**

Sintaxe: *MaxClients número*

Esta diretiva define o número máximo de requisições simultâneas que podem ser tratadas pelo servidor. Serão criados no máximo *número* processos filhos do servidor.

4.1.2 srm.conf

Neste arquivo é definido o espaço de nomes do seu servidor WWW que será visto pelos usuários. São definidas algumas informações importantes, tais como *aliases* para *scripts* e diretórios, nome da *home page* e nome do diretório *home* do usuário.

Vejam os conteúdo do **srm.conf** que acompanha o servidor **Apache**.

```
# With this document, you define the name space that users see of your http
# server. This file also defines server settings which affect how requests are
# serviced, and how results should be formatted.

# See the tutorials at http://www.apache.org/ for
# more information.
# Originally by Rob McCool; Adapted for Apache

# 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 /usr/local/etc/httpd/htdocs

# UserDir: The name of the directory which is appended onto a user's home
# directory if a ~user request is recieved.
UserDir public_html

# DirectoryIndex: Name of the file or files to use as a pre-written HTML
# directory index. Separate multiple entries with spaces.
DirectoryIndex index.html

# FancyIndexing is whether you want fancy directory indexing or standard
FancyIndexing on

# AddIcon tells the server which icon to show for different files or filename
# extensions
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/text.gif .ps .shtml
AddIcon /icons/movie.gif .mpg .qt
AddIcon /icons/binary.gif .bin
AddIcon /icons/burst.gif .wrl
AddIcon /icons/binhex.gif .hqx .sit
AddIcon /icons/uu.gif .uu
AddIcon /icons/tar.gif .tar .tar
AddIcon /icons/back.gif ..
AddIcon /icons/dir.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.
# Format: AddDescription "description" filename
# ReadmeName is the name of the README file the server will look for by
# default. Format: ReadmeName name
#
# The server will first look for name.html, include it if found, and it will
# then look for name and include it as plaintext if found.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.
ReadmeName README
HeaderName HEADER

# IndexIgnore is a set of filenames which directory indexing should ignore
# Format: IndexIgnore name1 name2...
IndexIgnore */.??* *~ *# */HEADER* */README* */RCS

# AccessFileName: The name of the file to look for in each directory
# for access control information.
AccessFileName .htaccess

# DefaultType is the default MIME type for documents which the server
# cannot find the type of from filename extensions.
DefaultType text/plain

# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+) uncompress
# information on the fly. Note: Not all browsers support this.
AddEncoding x-compress Z
AddEncoding x-gzip gz

# 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
# it can understand. Note that 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.
AddLanguage en .en
AddLanguage fr .fr
AddLanguage de .de
AddLanguage da .da
AddLanguage el .el
AddLanguage it .it

# 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.
LanguagePriority en fr de

# 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.
# Format: Redirect fakename url
```

```
# Aliases: Add here as many aliases as you need (with no limit). The format is
# Alias fakename realname
Alias /icons/ /usr/local/etc/httpd/icons/

# ScriptAlias: This controls which directories contain server scripts.
# Format: ScriptAlias fakename realname
ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/

# If you want to use server side includes, or CGI outside
# ScriptAliased directories, uncomment the following lines.

# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.
# Format: AddType type/subtype ext1

#AddType text/x-server-parsed-html .shtml
#AddType application/x-httpd-cgi .cgi

# For server-side includes which will be treated as HTML3
# for purposes of content negotiation, use
#AddType text/x-server-parsed-html3 .shtml3

# Uncomment the following line to enable Apache's send-asis HTTP file
# feature
#AddType httpd/send-as-is asis

# To enable type maps, you might want to use
#AddType application/x-type-map var

# If you wish to use server-parsed imagemap files, use
#AddType application/x-httpd-imap map

# Customizable error response (Apache style)
# these come in three flavors
#
# 1) plain text
#ErrorDocument 500 "The server made a boo boo."
# n.b. the (") marks it as text, it does not get output
#
# 2) local redirects
#ErrorDocument 404 /missing.html
# to redirect to local url /missing.html
#ErrorDocument 404 /cgi-bin/missing_handler.pl
# n.b. can redirect to a script or a document using server-side-includes.
#
# 3) external redirects
#ErrorDocument 402 http://other.server.com/subscription_info.html
```

Figura 4.2 - Exemplo do arquivo **srm.conf** distribuído com o servidor **Apache**

As diretivas básicas que devem ser configuradas são:

- **DocumentRoot**

Sintaxe: *DocumentRoot nome_do_diretório*

Esta diretiva define o diretório a partir do qual estarão os arquivos servidos pelo **httpd**. O servidor adiciona o caminho da **URL** requisitado ao *nome_do_diretório* para formar o caminho do documento. Por exemplo, se *DocumentRoot* for `/usr/www/htdocs` um acesso ao documento `http://www.inf.ufsc.br/index.html` fará referência ao caminho `/usr/www/htdocs/index.html`.

- **UserDir**

Sintaxe: *UserDir diretório*

Esta diretiva define o diretório que é adicionado ao diretório *home* do usuário quando o servidor receber uma requisição do tipo `~user`. Por exemplo, se *UserDir* for `public_html` um

acesso ao documento `http://www.inf.ufsc.br/~marchez/timao.html` fará referência ao caminho `/home/vesta/marchez/public_html/timao.html`, onde `/home/vesta/marchez/` é o diretório *home* do usuário **marchez**.

- **DirectoryIndex**

Sintaxe: *DirectoryIndex nome_do_arquivo*

Esta diretiva define arquivo que o servidor tentará acessar se nenhum nome for especificado. Por exemplo, se *DirectoryIndex* for `index.html`, um acesso ao documento `http://www.inf.ufsc.br/` fará referência a um arquivo chamado `index.html`, localizado dentro do diretório especificado por *DocumentRoot*.

- **DefaultType**

Sintaxe: *DefaultType tipo*

Às vezes o servidor recebe requisições de documentos cujos tipos não podem ser determinados pelos MIME *types* do servidor. Nestes casos o servidor irá usar o *DefaultType* para responder ao cliente.

- **Alias**

Sintaxe: *Alias nome_fictício nome_real*

Esta diretiva define apelidos para diretórios. Por exemplo, se configurarmos um apelido com sendo `Alias /times/ /usr/www/times/` um acesso ao documento `http://www.inf.ufsc.br/times/internacional.html` fará referência ao caminho `/usr/www/times/internacional.html`.

- **ScriptAlias**

Sintaxe: *ScriptAlias nome_fictício nome_real*

Esta diretiva define apelidos para os diretórios que podem conter *scripts* que serão executados pelo servidor. Restringir o número de diretórios que podem conter *scripts* é uma boa medida para evitar possíveis problemas de segurança (causados por *scripts* “mal intencionados”).

- **AddType**

Sintaxe: *AddType tipo/subtipo extensão*

Esta diretiva permite que você adicione novos tipos, sem ter que editar o arquivo **mime.types**, ou que você diga que “certos arquivos serão de certos tipos”. Por exemplo, se você deseja que *scripts* localizados fora dos diretórios definidos pela diretiva *ScriptAlias* possam ser executados pelo servidor, adicione um novo tipo usando a diretiva `AddType application/x-httpd-cgi .cgi`. Se você fizer esta escolha, tenha em mente os problemas de segurança que tal abertura pode trazer.

4.1.3 access.conf

Neste arquivo é definido a configuração global de acesso aos diretórios. Uma informação importante a configurar é a do diretório raiz dos documentos. Vejamos o conteúdo do arquivo **access.conf** que acompanha o servidor **Apache**.

```
# access.conf: Global access configuration
# Online docs at http://www.apache.org/

# This file defines server settings which affect which types of services
# are allowed, and in what circumstances.

# 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).

# Originally by Rob McCool

# /usr/local/etc/httpd/ should be changed to whatever you set ServerRoot to.
<Directory /usr/local/etc/httpd/cgi-bin>
Options Indexes FollowSymLinks
</Directory>

# This should be changed to whatever you set DocumentRoot to.
<Directory /usr/local/etc/httpd/htdocs>

# This may also be "None", "All", or any combination of "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or "MultiViews".
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you (or at least, not yet).
Options Indexes FollowSymLinks

# This option allows you to turn on the XBitHack behavior, which allows you
# to make text/html server-parsed by activating the owner x bit with chmod.
# This directive may be used wherever Options may, and has three
# possible arguments: Off, On or Full. If set to full, Apache will also
# add a Last-Modified header to the document if the group x bit is set.

# Unless the server has been compiled with -DXBITHACK, this function is
# off by default. To use, uncomment the following line:
#XBitHack Full

# This controls which options the .htaccess files in directories can
# override. Can also be "None", or any combination of "Options", "FileInfo",
# "AuthConfig", and "Limit"
AllowOverride All

# Controls who can get stuff from this server.
<Limit GET>
order allow,deny
allow from all
</Limit>

</Directory>
# You may place any other directories you wish to have access
# information for after this one.
```

Figura 4.3 - Exemplo do arquivo **access.conf** distribuído com o servidor **Apache**

Neste arquivo você precisa mudar basicamente duas ocorrências da diretiva **<Directory>**. Na primeira ocorrência, configure o mesmo diretório que foi definido em *ServerRoot* e na segunda ocorrência configure o mesmo diretório que foi definido em *DocumentRoot*.

4.1.4 mime.types

Neste arquivo são definidos o tipo e sub-tipo das mídias envolvidas nas transações que envolvem o HTTP. Ainda são definidas as extensões de cada tipo e sub-tipo de mídia. Os tipos e sub-tipos das mídias envolvidas são definidos pelo RFC 1521.

5 SENDMAIL

Há várias formas de entrega de mail, dependendo do endereço do destinatário. Para poupar o usuário da preocupação de qual método correto utilizar, há o programa **sendmail**. Ele recebe a mensagem do usuário, interpreta o e-mail do destinatário e então a envia da forma necessária. Da mesma maneira procede com mensagens recebidas da rede.

sendmail - envia mail através da Internet

Sinopse:

```
/usr/lib/sendmail [ -bd ] [ -bi ] [ -bt ] [ -bv ] [ -bq ] [ -q[ time ] ] [ address ... ]
```

Além disto, **sendmail** possui as seguintes características:

- recebe e envia **SMTP** mail (Internet);
- implementa o mecanismo de **aliases**, que permite redirecionamentos de mensagens e a criação de listas de discussão.

Todas as mensagens recebidas para um usuário são armazenadas em sua caixa postal, normalmente localizada em **/var/spool/mail/usuario**. Note que para cada usuário do sistema que já recebeu algum e-mail, um arquivo com o nome igual ao seu *login* existe no diretório **/var/spool/mail**, correspondendo exatamente à sua caixa postal.

Usado sem *flags*, **sendmail** lê a entrada padrão até encontrar EOF ou uma linha com apenas um ponto, e envia a mensagem escrita para todos os endereços listados.

Para receber **SMTP** mail da rede, executa-se **sendmail** como *daemon* durante a inicialização do sistema. Ele passa a observar a porta 25 e processar mensagens que chegam. Um exemplo de como disparar **sendmail**:

```
if [ -f /usr/lib/sendmail ]; then
  (cd /usr/spool/mqueue; rm -f lf*)
  /usr/lib/sendmail -bd -q1h; echo -n " sendmail" > /dev/console
```

Neste exemplo, primeiro se verifica a existência do programa **sendmail**, removem-se os arquivos de *lock* da fila de mail e então executa-se **sendmail** como *daemon* (-bd), determinando que a fila de mail deve ser processada a cada hora (-q1h).

5.1 Opções

As opções de execução do **sendmail** são:

| | |
|------------------|--|
| -bd | executa como daemon, esperando por conexões SMTP |
| -bi: | refaz aliases |
| -bq: | imprime um sumário da fila |
| -bt: | executa em modo de teste de endereço. Este modo lê endereços e mostra os passos da determinação do endereço final; usado para depuração. |
| -bv: | verifica apenas a validade de e-mails, sem enviar a mensagem. |
| -q[time]: | processa a fila de mail a cada intervalo time . |

5.2 Aliases

Mail *aliasing* é um mecanismo que provê:

- Nomes alternativos (*nicknames*) para usuários;
- Redirecionamento de mails para outros *hosts*;
- Listas de discussão (*mailing lists*).

Sua importância é tal que, sem ele, **sendmail** não conseguiria agir como servidor central de mail.

Aliases são definidos primariamente no arquivo */etc/aliases*. Em sistemas que usam NIS, se encontram no mapa aliases, gerado em grande parte das vezes a partir do arquivo */var/etc/aliases*. O formato desses arquivos é como segue:

```
alias: destinatário[,destinatário,...]
```

“*alias*” é o nome a quem o mail é endereçado, e “*destinatário*” o endereço de quem deve recebê-lo. “*destinatário*” pode ser o *login* de um usuário, outro *alias* ou um e-mail completo.

Pode haver múltiplos destinatários para um mesmo *alias*, o que permite criar listas de discussão.

É possível repassar a mensagem para um programa, da seguinte forma:

```
alias: "| programa"
```

Pode-se pegar os destinatários de um arquivo:

```
alias: :include:/pathname
```

sendmail usa aliases no formato DBM. Assim, deve ser executado o comando **newaliases** para transformar o arquivo de aliases para DBM. Este comando é apenas um *link* para o próprio **sendmail**, que se executa com a opção **-bi**.

5.3 Sendmail.cf

O arquivo de configuração do **sendmail** se chama */etc/sendmail.cf*. Ele contém a maior parte da configuração do **sendmail**, incluindo a informação necessária para rotear mail entre os programas de mail dos usuários e aqueles que irão efetivamente entregar as mensagens. Esse arquivo tem três funções principais:

- Definir o ambiente do **sendmail**;
- Reescrever endereços na sintaxe apropriada para o sistema receptor;
- Mapeia endereços em instruções necessárias para entregar o mail.

Vários comandos são necessários para implantar todas essas funções:

- Macros e opções para definir o ambiente;
- Regras de reescrita para reescrever endereços de e-mail;
- Definições de mailers para indicar as instruções de entrega do mail.

A sintaxe desse arquivo é complicada. Felizmente pode-se evitar ter que escrevê-lo. Uma forma é localizando um arquivo com uma configuração semelhante à do sistema em que se deseja criar o *sendmail.cf* e copiá-lo para o diretório */etc*, alterando o que mais for necessário.

Exemplos de *sendmail.cf* podem ser obtidos por **FTP anonymous** de *ftp.uu.net*, no arquivo *mail/sendmail/sendmail.n.tar.Z*, sendo “n” a versão do **sendmail**. Esse arquivo inclui também manuais de instalação, operação e funcionamento do **sendmail**. Em relação aos exemplos, estes diferem basicamente no princípio e no fim, e praticamente tudo que possa se precisar alterar reside nas duas primeiras seções.

Outra forma de gerar o *sendmail.cf* é com o programa **m4**, um processador de linguagem de macros.

O **sendmail** de domínio público, que pode ser obtido por **ftp anonymous** de *ftp.cr-df.rnp.br* no diretório *pub/packages/sendmail*, usa esse processo, fornecendo as macros que devem ser agrupadas e definidas para compor o *sendmail.cf*. Junto às fontes do **sendmail** e das macros vêm instruções simples e detalhadas sobre as macros e como defini-las.

5.3.1 Estrutura Geral

O *sendmail.cf* é dividido nas seguintes seções:

- **Informação local:** define informações específicas para o *host*. Por exemplo, são definidos o nome na Internet, o nome UUCP e conexões UUCP (caso hajam). Esta seção é normalmente alterada;
- **Macros gerais:** definem a informação relativa à rede local. Por exemplo, o domínio, o nome oficial do *host* (completo=nome+domínio), e os *relay hosts* (máquinas encarregadas de enviar mensagens que não possam ser enviadas pelo mailer local). Esta seção também é normalmente modificada;
- **Classes:** define grupos de *hosts* ou domínios usados para roteamento especial de mail. Modificações normalmente não são necessárias;
- **Versão:** identifica o número da versão do *sendmail.cf*. Deve-se incrementar esse número a cada vez que se modifica o arquivo;
- **Macros Especiais:** definem algumas macros especiais, tais como o nome que o **sendmail** usa para se identificar quando ocorrem mensagens de erro, e a mensagem que é mostrada

num login SMTP. Esta seção não é alterada;

- **Opções:** Opções do **sendmail** que normalmente não necessitam de modificações;
- **Precedência de mensagens:** vários valores de precedência de mensagens. Não se modifica esta seção;
- **Usuários confiáveis:** usuários que têm permissão para sobrescrever o endereço do remetente quando estão enviando mail. Adicionar usuários a esta lista é um problema potencial de segurança. Esta seção não se altera;
- **Formato dos cabeçalhos:** define o formato dos cabeçalhos que o **sendmail** inclui nas mensagens, e não necessita de modificação;
- **Regras de reescrita:** são as regras gerais usadas para reescrever os endereços de mail. Esta seção só é modificada para corrigir um problema ou adicionar um serviço;
- **Mailers:** instruções usadas pelo **sendmail** para invocar os programas de entrega de mail corretos. As regras de reescrita específicas associadas a cada mailer também estão incluídas nesta seção, que normalmente não é alterada;
- **Conjunto de regras zero (rule set zero):** um conjunto especial de regras que é aplicado ao endereço de entrega. Esta seção não é modificada. A parte que pode ser alterada tem sua própria seção;
- **Parte da “rule set zero” dependente da máquina:** são as partes da rule set zero específicas à configuração do sistema. Esta seção varia de acordo com os mailers configurados. Um sistema que entregue SMTP e UUCP terá diferentes regras que um que apenas entregue UUCP. Escolhendo-se um *sendmail.cf* exemplo de acordo com sua configuração poupa o trabalho de se alterar esta seção.

Esta é uma estrutura geral para o *sendmail.cf*. Apesar das variações de cada local, algumas similaridades se mantêm :

- A informação adaptável a cada *host* fica provavelmente no início;
- Tipos semelhantes de comandos, como de opções e cabeçalhos, são agrupados;
- A parte central, a maior delas, contém as regras de reescrita;
- O final do arquivo contém definições de mailers misturadas com as regras de reescrita associadas a cada mailer.

5.3.2 Configuração do sendmail - Comandos

sendmail lê */etc/sendmail.cf* cada vez que é disparado. Por isto, a sintaxe do *sendmail.cf* é fácil de ser usada pelo **sendmail**, e não por um ser humano. A sintaxe segue a regra de que o primeiro caracter da linha é sempre o comando. A tabela 5.1 mostra os comandos e suas sintaxes:

| Nome | Sintaxe | Função |
|---------------------|----------------------------------|--|
| Define Macro | D <i>xvalor</i> | Seta macro <i>x</i> para <i>valor</i> |
| Define Classe | C <i>word1[word2...]</i> | Seta classe <i>c</i> para <i>word1 word2 ...</i> |
| Define Classe | F <i>carquivo</i> | Carrega classe <i>c</i> de <i>arquivo</i> |
| Seta Opção | O <i>ovalor</i> | Seta opção <i>o</i> para <i>valor</i> |
| Usuários Confiáveis | T <i>user1[user2...]</i> | Usuários confiáveis são <i>user1 user2 ...</i> |
| Seta Precedência | P <i>nome=número</i> | Seta <i>nome</i> à precedência <i>número</i> |

| | | |
|-------------------------|------------------------------------|---|
| Seta Conjunto de Regras | <i>Sn</i> | Inicia conjunto de regras <i>n</i> |
| Define Regra | R <i>lhs rhs comentário</i> | Reescreve padrão <i>lhs</i> para formato <i>rhs</i> |

Tabela 5.1 - Comandos do `sendmail`

5.3.2.1 Comando de Definição de Macro - D

D é um comando que define uma macro e nela armazena um valor.

Uma vez definida, uma macro é usada para prover seu valor a outros comandos do `sendmail.cf` e para o próprio `sendmail`. Isto permite que suas configurações sejam reaproveitadas por vários sistemas, bastando modificar algumas macros específicas.

O nome de uma macro pode ser qualquer caractere *ASCII*, sendo que letras minúsculas são usadas por macros internas do `sendmail`. Macros definidas pelo usuário usam letras maiúsculas.

Para obter o valor guardado numa macro deve-se referenciá-la por `$x`, onde *x* é o nome da macro.

A tabela 5.2 apresenta as macros internas a serem setadas no `sendmail.cf`:

| Nome | Função |
|------|---|
| a | Data de origem no formato RFC 822 |
| b | Data corrente em formato RFC 822 |
| c | <i>hop count</i> : número de hosts que intermediam a mensagem |
| d | data em formato UNIX (ctime) |
| e | Mensagem de entrada SMTP |
| f | endereço “from” do remetente |
| g | endereço do remetente relativo ao destinatário |
| h | host destinatário |
| i | <i>id</i> da fila |
| j | Nome oficial do domínio deste <i>site</i> |
| l | formato da linha “from” do UNIX |
| n | Nome do <i>daemon</i> (para mensagens de erro) |
| o | Conjunto de operadores em endereços |
| p | <i>pid</i> do <code>sendmail</code> |
| q | Formato <i>default</i> do endereço do remetente |
| r | Protocolo usado |
| s | Nome do <i>host</i> do remetente |
| t | Representação numérica da hora corrente |
| u | Usuário remetente |

| | |
|---|--------------------------------|
| v | Número da versão do sendmail |
| w | Nome deste site |
| x | Nome completo do remetente |
| z | Diretório home do destinatário |

Tabela 5.2 - Macros internas do *sendmail.cf*

5.3.2.2 Condicionais

A condicional serve para testar se uma macro está definida e associar um valor em caso afirmativo, ou outro em caso negativo.

A sintaxe completa é:

```
$?x texto1 $| texto2$.
```

Sua interpretação:

```
Se ($?x) x esta'definida
                use texto1
Senão (?|)
                use texto2
Fim (?.)
```

Exemplo:

```
Dq$g$?x ($x) $.
```

Aqui associa-se à macro *q* (endereço do remetente) o endereço do remetente relativo ao destinatário (*g*), e mais o nome completo do remetente, caso esteja definida a macro *x*, que armazena este valor.

5.3.2.3 Comandos de Definição de Classe - C, F

Uma classe é um vetor de valores. É usada quando se manipula múltiplos valores da mesma forma, como nomes para um *host*. Para defini-las, usam-se os comandos **C** e **F**. Como macros, classes usam caracteres como nomes, e classes de usuário usam letras maiúsculas.

Valores podem ser associados em uma ou mais linhas (comando **C**), ou lidos de um arquivo (comando **F**).

Exemplos:

Definindo a classe `w` em uma linha apenas.

```
Cwvenus inf inf-gw
```

Definindo a classe `w` em múltiplas linhas:

```
Cwvenus
  Cwinf
  Cwinf-gw
```

Definindo a classe `w` a partir do conteúdo do arquivo `/etc/sendmail.cw`:

```
Fw/etc/sendmail.cw
```

5.3.2.4 Comando Para Setar Opção - O

Este comando **O** associa valores a opções internas do sendmail, que não são as mesmas que as opções da linha de comando.

O valor pode ser:

- Uma cadeia de caracteres;
- Um inteiro;
- Um booleano;
- Um intervalo de tempo.

Não há opções criadas pelo usuário. A tabela 5.3 mostra as opções internas:

| Nome | Função |
|--------------|---|
| <i>Afile</i> | Define o nome do arquivo de aliases |
| aN | Espera <i>N</i> minutos por <i>@: @</i> , então refaz o arquivo de aliases |
| Bc | Define o caracter de substituição branco |
| c | Enfileira mail para mailers custosos |
| D | Refaz a base de dados de aliases |
| db | Distribui mail em <i>background</i> |
| di | Distribui interativamente |
| dq | Distribui durante o próximo processamento da fila |
| ee | Envia mensagens de erro por mail e sempre retorne 0 para <i>exit status</i> |
| em | Envia mensagens de erro para o remetente |
| ep | Imprime mensagens de erro |
| eq | Apenas retorna <i>exit status</i> , sem mensagens de erro |
| ew | Escreve mensagens de erro para o remetente |

| | |
|--------------|---|
| Fn | Seta permissões <i>n</i> para arquivos temporários |
| f | Mantém linhas "From" ao estilo UNIX |
| gn | Seta o gid default <i>n</i> para mailers |
| Hfile | Define o nome do arquivo de help SMTP |
| I | Usa o BIND name server para resolver todos os nomes de hosts |
| i | Ignora pontos em mensagens que estão chegando |
| Ln | Seta o nível de logging para <i>n</i> |
| Mxval | Seta macro <i>x</i> para <i>val</i> |
| m | Envia para o remetente também |
| Nnet | Define o nome da rede |
| o | Aceita cabeçalhos no formato antigo |
| Qdir | Define o nome do diretório de fila |
| qn | Define um fator <i>n</i> usado para decidir quando enfileirar trabalhos |
| rt | Seta intervalo <i>t</i> para <i>timeout</i> de leitura |
| Sfile | Define o nome do arquivo de registro de estatísticas |
| s | Sempre cria o arquivo de fila antes de tentar distribuir o mail |
| Tt | Seta o <i>timeout</i> da fila para <i>t</i> |
| un | Seta o uid default <i>n</i> para mailers |
| v | Executa em modo de exposição (mostrando o que está fazendo) |
| Wpass | Define senha usada para depuração remota |
| Xl | Recusa conexões SMTP se a taxa de carga ultrapassar <i>l</i> mensagens na fila no último minuto |
| xl | Enfileira mensagens se a taxa de carga ultrapassar <i>l</i> |
| Y | Envia cada trabalho da fila em processo separado |
| yn | Reduz a prioridade dos trabalhos para <i>n</i> para cada destinatário |
| Zn | Decrementa em <i>n</i> a prioridade de um trabalho cada vez que ele for processado |
| zn | Fator usado com a precedência para determinar a prioridade de uma mensagem |

Tabela 5.3 - Opções do *sendmail.cf*

5.3.2.5 Definindo Usuários Confiáveis - T

O comando **T** define uma lista de usuários que podem sobrescrever o endereço do remetente usando a flag **-f** do mailer. Isto implica redução da segurança e possibilidade de mails adulterados.

Definições mais comuns:

| |
|---|
| T root T daemon T uucp |
|---|

5.3.2.6 Definindo Precedência de Mail - P

Precedência é um dos fatores para associar prioridade a mensagens entrando na fila. O comando **P** define esses valores de precedência disponíveis para os usuários. Quanto maior o valor,

maior a precedência da mensagem, sendo o valor default 0.

Alguns valores:

```
Pfirst-class=0
Pspecial-delivery=100
Pbulk=-60
Pjunk=-100
```

Para usar este mecanismo, o usuário adiciona uma linha *Precedence* ao seu mail, como no exemplo:

```
Precedence: bulk
```

5.3.2.7 Definindo Cabeçalhos - H

O comando **H** define o formato do cabeçalho inserido nas mensagens. Seu formato é **H** seguido de:

- *Flags* opcionais do cabeçalho delimitadas por sinais de interrogação;
- Um nome de cabeçalho;
- Dois-pontos e um template de cabeçalho (combinação de textos e macros).

A função da *flag* de cabeçalho é controlar quando o cabeçalho deve ser inserido na mensagem associada a um mailer específico (tal mailer deve setar uma *flag* de mesmo nome na sua definição, vista mais adiante). Caso não seja especificada, o cabeçalho é inserido para todos mailers.

Como exemplo:

```
H?P?Return-Path: <$ g>
HReceived: $?sfrom$s $.by $j ($v/$Z)
H?D?Resent-Date: $a
H?D?Date: $a
```

5.3.2.8 Definindo Mailers - M

Os comandos **M** definem os programas de entrega de mail usados pelo **sendmail**. A sintaxe é:

```
Mnome, {campo=valor}
```

“**nome**” é um nome arbitrário usado internamente pelo **sendmail** para se referir a esse mailer.

Dois mailers devem ser referenciados por nomes específicos:

- *local* (entrega de mail local);
- *prog* (entrega de mail a programas).

A lista *{campo=valor}* possui as características do mailer, e é separada por vírgulas.

A tabela 5.4 apresenta os campos de mailers:

| Campo | Significado | Comentário | Exemplo |
|--------------|--------------------|---|----------------|
| P | Path | <i>Pathname</i> do mailer | P=/bin/mail |
| F | Flags | <i>Flags</i> do sendmail para o mailer | F=lsDFMe |
| S | Sender | Regras para endereço do remetente | S=10 |
| R | Recipient | Regras para o destinatário | R=20 |
| A | Argv | O vetor de argumentos do mailer | A=sh -c \$u |
| E | Eol | <i>String</i> de final-de-linha para o mailer | E=\r\n |
| M | Maxsize | Tamanho máximo da mensagem | M=100000 |

Tabela 5.4 - Campos de mailers

Se o campo **P** contiver a *string* IPC, então **sendmail** é usado para entregar o mail via SMTP. O campo **F** contém as *flags* usadas pelos mailers. As flags podem ser vistas na tabela 5.5.

| Nome | Função |
|-------------|--|
| C | Adicionar <i>@domínio</i> ao endereço que não tenha <i>@</i> |
| D | O mailer quer uma linha de cabeçalho <i>Date</i> : |
| E | Adicionar <i>></i> para as linhas da mensagem que comecem com <i>From</i> : |
| e | Indica que é um mailer custoso |
| F | O mailer quer de uma linha de cabeçalho <i>From</i> : |
| f | O mailer aceita a <i>flag -f</i> de usuários confiáveis |
| h | Mantém maiúsculas em nomes de <i>hosts</i> |
| I | O mailer irá falar SMTP com outro sendmail |
| L | Limita o comprimento das linhas como especificado na RFC 821 |
| l | Este é um mailer local |
| M | O mailer quer de uma linha de cabeçalho <i>Message-Id</i> : |
| m | O mailer consegue enviar para múltiplos usuários em uma transição |
| n | Não inserir uma linha <i>From</i> : no estilo UNIX na mensagem |
| P | O mailer quer uma linha de cabeçalho <i>Return-Path</i> : |
| R | Usar <i>MAIL-FROM: return-path</i> ao invés do endereço de retorno |
| r | O mailer aceita a <i>flag -r</i> de usuários confiáveis |
| S | Não resetar o userid antes de chamar o mailer |
| s | Retirar aspas do endereço antes de chamar o mailer |
| U | O mailer quer linha de cabeçalho <i>From</i> : no estilo UNIX |
| u | Mantém maiúsculas em nomes de usuários |
| X | Inserir um ponto no início de linhas que comecem com um ponto |
| x | O mailer quer uma linha de cabeçalho <i>Full-Name</i> : |

Tabela 5.5 - *Flags* dos mailers

Por exemplo, a definição de mailer local:

```
Mlocal, P=/bin/mail, F=rlsDFMmn, S=10, R=20, A=mail -d $u
```

5.3.2.9 Regras de Reescrita de Endereços - R

São o coração do *sendmail.cf*, e muito consolidadas. Dificilmente um administrador precisará alterá-las. A título de informação, são definidas por:

```
Rpadrão transformação comentário
```

A idéia é procurar o *padrão* no endereço processado. Quando esse padrão for encontrado, aplica-se a *transformação* sobre o endereço.

As tabelas 5.6 e 5.7 mostram respectivamente os símbolos para procura de padrões e transformações de endereços.

| Símbolo | Significado |
|---------|--|
| \$* | Combina com zero ou mais <i>tokens</i> |
| \$+ | Combina um ou mais <i>tokens</i> |
| \$- | Combina com exatamente um <i>token</i> |
| \$=x | Combina com qualquer <i>token</i> na classe x |
| \$~x | Combina com qualquer <i>token</i> que não esteja na classe x |
| \$x | Combina com todos os <i>tokens</i> na classe x |
| \$%x | Combina com qualquer <i>token</i> no mapa NIS nomeado na macro x |
| \$!x | Combina com qualquer <i>token</i> fora do mapa NIS nomeado na macro x |
| \$%y | Combina com qualquer <i>token</i> no mapa NIS hosts.byname |

Tabela 5.6 - Símbolos para procura de padrões

| Símbolo | Significado |
|------------|---|
| \$n | Substituir <i>token</i> indefinido <i>n</i> |
| \$[nome\$] | Substituir nome canônico |
| \$>n | Chamar ruleset <i>n</i> |
| \$@ | Terminar ruleset |
| \$: | Terminar regra de reescrita |

Tabela 5.7 - Tranformações de endereços

Para ilustrar a aplicação de uma regra, assumamos que o endereço a ser transformado seja *alguem@ufsc.bitnet*, e após algum processamento anterior torne-se *alguem<@ufsc.bitnet>* . Aplica-se, então, a seguinte regra:

```
R$+<@$+.bitnet>          $1%$2<@$B>          Use Bitnet relay
```

O endereço combina com o padrão porque há um ou mais *tokens* antes dos caracteres <@ , um ou mais *tokens* após <@ , e então há a sequência *string.bitnet* . A procura do padrão resulta em dois *tokens* indefinidos, usados na transformação do endereço. A transformação contém o token indefinido *\$1*, um caractere %, o token indefinido *\$2*, a sequência <@, a macro *B*, e o caracter >. Após a aplicação do padrão, *\$1* contém *alguem* e *\$2* contém *ufsc*. Assumindo que a macro *B* (Bitnet relay) tenha sido definida em algum lugar do *sendmail.cf* como *ibm.ufsc.br*, o endereço será reescrito como:

```
alguem%ufsc<@ibm.ufsc.br>
```

5.3.2.10 Comando Ruleset - S

As regras podem ser agrupadas em conjuntos (**rulesets**), tendo um número associado, tornando possível de serem referenciadas nas definições de mailers vistas anteriormente. O comando **S** marca o início de um conjunto de regras, identificando-o com um índice.

```
Sn
```

Cinco conjuntos (**rulesets**) possuem função específica e são chamados diretamente pelo **sendmail**:

- **Ruleset 3**: O maior e mais complicado, é o primeiro aplicado ao endereço, convertendo para a forma canônica “*parte-local@host.domínio*”;
- **Ruleset 0**: Aplicado após o 3, e somente para endereços de destinatários usados para entrega do mail. Gera a tripla (*mailer,host,domínio*);
- **Rulesets 1 e 2**: Aplicados a todos os endereços na mensagem;
- **Ruleset 4**: Aplicado a todos os endereços na mensagem, traduzindo endereços internos para externos.

6 EXERCÍCIOS

1. Crie um “serviço”, utilizando **inetd**, que permita que uma máquina remotamente conecte-se a uma porta em seu servidor, e obtenha a lista de usuários atualmente “logados” no sistema.
2. Configure seu servidor WWW, de modo que os arquivos disponibilizados em seu servidor de FTP anônimo, sejam também disponibilizados via Web. Assim, se o servidor FTP está em `ftp://maquina.com.br/pub`, o servidor WWW deve acessar os mesmos arquivos como `http://maquina.com.br/pub`.
3. Mude a configuração do servidor WWW para escutar na porta 80.000, via `inetd`.

7 BIBLIOGRAFIA

1. FEIT, Sidnie. TCP/IP – Architecture, Protocols, and Implementation with IP v6 and IP Security. 2nd. ed. McGraw-Hill, 1996.
2. FRISCH, Ellen. Essential System Administration. 2nd. ed. O'Reilly & Associates, 1996.
3. HUNT, Craig. TCP/IP Network Administration. 2nd. ed. O'Reilly & Associates, 1997.
4. NEMETH, E.; SNYDER, G.; SEEBASS, S; HEIN, T. Unix System Administration Handbook. 2nd. ed. Prentice-Hall, 1995.
5. UDELL, Jon. Practical Internet Groupware. O'Reilly & Associates, 1999.

Dúvidas, críticas e sugestões sobre esta apostila: <mailto:webber@sj.univali.br>