Linux Network Test - LNT

Maicon L. R. da Cruz Alves[1], **André Moraes**[2]

[1]Tecnologo Redes de Computadores – Faculdade Senac Pelotas (Fatec)
Pelotas – RS – Brazil

tec.mlc@bol.com.br

Abstract. *This article will present the creation of a Linux distribution, with tools for vulnerability analysis in the infrastructure of a corporate network, optimizing the work of the network administrator to go directly to the problems.*

Resumo. *Este artigo apresenta a criação de uma distribuição Linux, com ferramentas para análise de vulnerabilidade na infra-estrutura de uma rede coorporativa, otimizando o trabalho do administrador de rede para ir diretamente nos problemas.*

1. Introdução

Ultimamente preocupa-se bastante com segurança de informações e sigilo de dados em ambientes corporativos e empresas.tais como proteção de dados de grande importância e protocolos. As redes de computadores de hoje, vem crescendo muito com diversas tecnologias novas, um administrador sem grandes conhecimentos pode deixar pequenas falhas de segurança possibilitando a invasão de pessoas não autorizadas. Com inúmeros benefícios, o LNT além de ter diversas ferramentas para analisar a rede local, também possui a praticidade de ser um live-cd podendo ser utilizado em qualquer host. O sistema LNT serão construído usando uma distribuição previamente instalada do Linux (tal como Debian, Mandrake, o Red Hat, ou o SuSE). Este sistema (o anfitrião) será usado como ponto de partida e fornecerá os programas necessários, incluindo um compilador, um editor de vínculos (linker), e um shell, para montar o novo sistema. Escolha a opção ”desenvolvimento”, ou similar, durante a instalação da distribuição anfitriã para ter acesso a estas ferramentas.

2. Conceitos Básicos

Breve descrição dos Conceitos bácicos do Linux, Kernel e Linux From Scratch.

2.1. Linux

O núcleo Linux foi, originalmente, escrito por Linus Torvalds do Departamento de Ciência da Computação da Universidade de Helsinki, Finlândia, com a ajuda de vários programadores voluntários através da Usenet (uma espécie de sistema de listas de discussão existente desde os primórdios da Internet).

Linus Torvalds começou o desenvolvimento do núcleo como um projeto particular, inspirado pelo seu interesse no Minix, um pequeno sistema UNIX desenvolvido por Andrew S. Tanenbaum. Ele limitou-se a criar, nas suas próprias palavras, ”um Minix melhor que o Minix”(”a better Minix than Minix”)

Curiosamente, o nome Linux foi criado por Ari Lemmke, administrador do site ”[Funet 1990] ”que deu esse nome ao diretório FTP onde o núcleo Linux estava inicialmente disponível. (Linus tinha-o baptizado como ”Freax”, inicialmente).

No dia 5 de outubro de 1991 Linus Torvalds anunciou a primeira versão ”oficial”do núcleo Linux, versão 0.02. Desde então muitos programadores tem respondido ao seu chamado, e tem ajudado a fazer do Linux o sistema operacional que é hoje. No início era utilizado por programadores ou só por quem tinha conhecimentos, usavam linhas de comando. Hoje isso mudou, existem diversas empresas que criam os ambientes gráficos, as distribuições cada vez mais amigáveis de forma que uma pessoa com poucos conhecimentos consegue usar o Linux. Hoje o Linux é um sistema estável e consegue reconhecer muitos periféricos sem a necessidade de se instalar os drivers de som, vídeo, modem, rede, entre outros.[Wikipédia 1998]

2.2. Kernel

O Kernel é um componente essencial do Sistema Operacional, muitas vezes encarado como ”O cérebro do S.O”. Na verdade ele é o grande responsável por fazer a interação entre as camadas de hardware e software. Traduzindo, é o Kernel que gerencia os recursos do sistema e permite que os programas façam uso deles.

Basicamente o kernel começa a funcionar assim que o computador é ligado; nesse momento ele inicia a detecção de todo o hardware indispensável ao funcionamento da máquina (placa de vídeo etc). O Sistema Operacional é carregado em seguida espera que o usuário faça seu login, feito isso o Kernel passa a administrar as principais funções dentro do S.O.: isso inclui o gerenciamento da memória, dos processos, dos arquivos e de todos os dispositivos.

Dessa forma o Kernel pode ser descrito como um grande organizador: é ele o responsável por garantir que todos os programas terão acesso aos recursos de que necessitam (memória RAM, por exemplo) simultaneamente, fazendo com que haja um compartilhamento concorrente - mas sem oferecer riscos a integridade da máquina.[Amaral 2009]

2.3. LFS

Linux From Scratch (ou lfs, abreviadamente) é um projeto que demostra, através de um guia on-line, os passos necessários é construção de sua própria distribuição Linux, utilizando o código-fonte dos softwares que compõem o sistema. Como resultado, é obtido um sistema Linux personalizado, compacto e eficiente.[Scratch 1998]

2.4. Sistema Anfitrião

O sistema anfitrião deve estar executando pelo menos um kernel 2.6.2 compilado com o GCC 3.0 ou superior. Há duas razões principais para esta exigência. Primeiro O conjunto de testes da biblioteca Native POSIX Threading Library (NPTL) falha se o kernel do anfitrião não for compilado com o GCC 3.0 ou posterior. Segundo A versão 2.6.2, ou superior, do kernel é exigida para o uso do Udev. O Udev cria dispositivos dinamicamente, lendo diretamente do sistema de arquivos sysfs. Entretanto, o suporte para este sistema de arquivos somente foi implementado recentemente na maioria dos gerenciadores de dispositivos do kernel. É necessarios garantir que todos os dispositivos críticos do sistema sejam criados corretamente.

3. Criando a Distribuição LNT

Breve explicação sobre a origem e os procedimentos realizados.

3.1. Origem do LNT

O LNT é uma distribuição baseada no Linux From Scratch (Linux do Zero), a idéia principal é entender como funciona uma distribuição linux, seguido da idéia de criar uma distribuição similar ao Backtrack, uma distribuição com varias ferramentas para análise de uma rede.

3.2. PROCEDIMENTOS

A seguir o passo a passo para criar a estrutura onde sera instalado o lnt.

3.2.1. Particionamento de disco

Como a maioria dos sistemas operacionais, o LNT é instalado geralmente em uma partição dedicada. É recomendado construir um sistema LNT em uma partição vazia disponível ou, se tiver bastante espaço não particionado, criar uma.Um sistema mínimo requer uma partição de aproximadamente 1,3 gigabytes (GB). Isto é o bastante para armazenar todos os pacotes com os fontes e para compilá-los. Entretanto, se o sistema lnt vier a ser o sistema Linux principal, o software adicional que será instalado provavelmente exigirá algum espaço adicional (2-3 GB). O próprio sistema LNT não fará uso de todo este espaço. Uma grande parcela desta exigência se deve á necessidade de fornecer espaço livre suficiente para armazenamento provisório. A compilação dos pacotes requer muito do espaço em disco, que é recuperado depois que o pacote é instalado. Porque nunca há memória de acesso aleatório (RAM) disponível para processos da compilação, é uma boa idéia usar uma partição pequena do disco como swap. Ela é usada pelo kernel para armazenar dados raramente usados e deixar mais memória física disponível para processos ativos. A partição swap para o sistema LNT pode ser a mesma que está sendo usada pelo sistema de anfitrião, e neste caso não é necessário criar outra.

Fazendo uso dos utilitários fdisk ou cfdisk, é necessário criar 3 partições hda1 inicializavel de 1.5gb hda2 swap de 512mb e hda3 para os pacotes com 1gb.

Foram formatadas as partições hda1 e hda3 com o comando ”mke2fs /dev/hda1”, ”mke2fs /dev/hda3/”, montando as partições e adicione no fstab. exemplo:

```
Cria pasta lnt dentro da pasta /mnt
#mkdir /mnt/lnt
Cria pasta lntpacotes dentro da pasta /mnt
#mkdir /mnt/lntpacotes
Montar o hda1 na pasta /mnt/lnt
#mount /dev/hda1 /mnt/lnt
Montar o hda3 na pasta /mnt/lntpacotes
#mount /dev/hda3 /mnt/lntpacotes
```

Devem ser adicionados os pontos de montagem no fstab.

3.2.2. Download dos pacotes necessários

O sistema LNT necessita de uma serie de pacotes e patchs básicos para funcionar, baixe os pacotes para a pasta /mnt/lntpacotes/sources.

Pacotes:

- Autoconf (2.59) - 908 kilobytes (KB):
- Autoconf (2.59) - 908 kilobytes (KB):
- Automake (1.9.5) - 748 KB:
- Bash (3.0) - 1,824 KB:
- Binutils (2.15.94.0.2.2) - 11,056 KB:
- Bison (2.0) - 916 KB:
- Bzip2 (1.0.3) - 596 KB:
- Coreutils (5.2.1) - 4,184 KB:
- DejaGNU (1.4.4) - 852 KB:
- Diffutils (2.8.1) - 648 KB:
- E2fsprogs (1.37) - 3,100 KB:
- Expect (5.43.0) - 416 KB:
- File (4.13) - 324 KB:
- Findutils (4.2.23) - 784 KB:
- Flex (2.5.31) - 672 KB:
- Gawk (3.1.4) - 1,696 KB:
- GCC (3.4.3) - 26,816 KB:
- Gettext (0.14.3) - 4,568 KB:
- Glibc (2.3.4) - 12,924 KB:
- Glibc-Linuxthreads (2.3.4) - 236 KB:
- Grep (2.5.1a) - 520 KB:
- Groff (1.19.1) - 2,096 KB:
- GRUB (0.96) - 768 KB:
- Gzip (1.3.5) - 284 KB:
- Hotplug (2004-09-23) - 40 KB:
- Iana-Etc (1.04) - 176 KB:
- Inetutils (1.4.2) - 752 KB:
- IPRoute2 (2.6.11-050330) - 276 KB:
- Kbd (1.12) - 624 KB:
- Less (382) - 216 KB:
- lnt-Bootscripts (3.2.1) - 32 KB:
- Libtool (1.5.14) - 1,604 KB:
- Linux (2.6.11.12) - 35,792 KB:
- Linux-Libc-Headers (2.6.11.2) - 2,476 KB:
- M4 (1.4.3) - 304 KB:
- Make (3.80) - 904 KB:
- Man (1.5p) - 208 KB:
- Man-pages (2.01) - 1,640 KB:
- Mktemp (1.5) - 68 KB:
- Module-Init-Tools (3.1) - 128 KB:
- Ncurses (5.4) - 1,556 KB:

- Patch (2.5.4) - 156 KB:
- Perl (5.8.6) - 9,484 KB:
- Procps (3.2.5) - 224 KB:
- Psmisc (21.6) - 188 KB:
- Readline (5.0) - 1,456 KB:
- Sed (4.1.4) - 632 KB:
- Shadow (4.0.9) - 1,084 KB:
- Sysklogd (1.4.1) - 72 KB:
- Sysvinit (2.86) - 88 KB:
- Tar (1.15.1) - 1,580 KB:
- Tcl (8.4.9) - 2,748 KB:
- Texinfo (4.8) - 1,492 KB:
- Udev (056) - 476 KB:
- Udev Rules Configuration - 5 KB:
- Util-linux (2.12q) - 1,344 KB:
- Vim (6.3) - 3,620 KB:
- Vim (6.3) language files (optional) - 540 KB:
- Zlib (1.2.2) - 368 KB:

Patches necessários

- Bash Various Fixes - 23 KB:
- Bash Avoid Wcontinued Patch - 1 KB:
- Coreutils Suppress Uptime, Kill, Su Patch - 15 KB:
- Coreutils Uname Patch - 4 KB:
- Expect Spawn Patch - 7 KB:
- Flex Brokenness Patch - 156 KB:
- GCC Linkonce Patch - 12 KB:
- GCC No-Fixincludes Patch - 1 KB:
- GCC Specs Patch - 14 KB:
- Glibc Fix Testsuite Patch - 1 KB:
- Gzip Security Patch - 2 KB:
- Inetutils Kernel Headers Patch - 1 KB:
- Inetutils No-Server-Man-Pages Patch - 4 KB:
- IPRoute2 Disable DB Patch - 1 KB:
- Mktemp Tempfile Patch - 3 KB:
- Perl Libc Patch - 1 KB:
- Readline Fixes Patch - 7 KB:
- Sysklogd Fixes Patch - 27 KB:
- Tar Sparse Fix Patch - 1 KB:
- Util-linux Cramfs Patch - 3 KB:
- Vim Security Patch - 8 KB:
- Zlib Security Patch - 1 KB:

3.2.3. Diretórios

Todos os programas compilados serão instalados no diretório /mnt/lntpacotes/tools. Os programas compilados aqui são ferramentas provisórias e não serão parte do sistema final

LNT. Mantendo estes programas em um diretório separado, eles poderão ser facilmente descartados mais tarde, após seu uso. Isto evita também que estes programas fiquem nos diretórios do sistema anfitrião.

Comando utilizados para a criação dos diretórios:

```
Foi criada a pasta tools dentro do /mnt/lntpacotes
#mkdir /mnt/lntpacotes/tools
Foi criada a pasta sources dentro do /mnt/lntpacotes este diretório fic
#mkdir /mnt/lntpacotes/sources
Link sinbolico soft da pasta tools do sistema anfitrião para tools do L
#ln –s /mnt/lntpacotes/tools /
```

3.2.4. Adicionando usuário e ajustando permissões

Primeiro deve-se criar o grupo do usuário, apos criar o grupo criaremos o usuário lnt.

```
#groupadd lnt
#useradd –s /bin/bash –g lnt –m –k /dev/null lnt
```

As opções da linha de comando realizam a seguinte tarefa: -s /bin/bash Faz o bash shell padrão para o usuário lnt. -g lnt Esta opção adiciona o usuário lnt ao grupo lnt. -m Cria um diretório home para o usuário lnt. -k /dev/null

Este parâmetro impede a cópia dos arquivos-modelo do diretório skeleton (normalmente é /etc/skel) alterando a posição da entrada padrão para o dispositivo especial.

lnt Este é o nome atual para o grupo e o usuário criados. Para iniciar fazer o login como lnt (ao contrário de comutar para o usuário lnt quando se fez o login como root, que não exige que o usuário lnt tenha uma senha), vamos dar ao usuário lnt uma senha:

```
#passwd lnt
```

Deve ser garantido ao usuário lnt acesso irrestrito ao diretório /mnt/lntpacotes/tools fazendo-o proprietário do diretório:

```
#chown lnt /mnt/lntpacotes/tools
```

Se um diretório separado foi criado para trabalho, como sugerio, deve ser feito o lnt dono desse diretório:

```
#chown lnt /mnt/lntpacotes/sources
```

Em seguida, será iniciada uma sessão como usuário lnt. Isto pode ser feito através de um console virtual, gerenciador de display, ou com o comando:

```
#su – lnt
```

3.3. Configurando o profile do usuário

Vamos definir um bom ambiente de trabalho criando dois arquivos de inicialização para o shell bash. Faça o login como usuário lnt, e execute o seguinte comando para criar um novo .bash_profile:

```
#cat > ~/.bashprofile << \"EOF\"
exec env -i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash
EOF
```

Uma nova instância do shell é um non-login, que não lê os arquivos /etc/profile ou .bash_profile, mas lê o arquivo .bashrc Será criar o arquivo .bashrc:

```
#cat > ~/.bashrc << "EOF"
set +h
umask 022
lnt=/mnt/lnt
LC_ALL=POSIX
PATH=/tools/bin:/bin:/usr/bin
export lnt LC_ALL PATH
EOF
```

Finalmente, para possuir um ambiente totalmente preparado para a configuração das ferramentas provisórias, é necessário ativar o perfil de usuário recém-criado:

```
#source ~/.bash_profile
```

3.3.1. Primeiros pacotes

São considerados os mais importantes, porque são a base do sistema, todos pacotes instalados posteriormente tem alguma dependências desses pacotes.

- *Binutils:* É importante que o Binutils seja o primeiro pacote compilado porque tanto o Glibc quanto o GCC executam vários testes no vinculador dinâmico e no assembler disponíveis para determinar qual de suas próprias características serão configuradas. Este pacote é conhecido por apresentar problemas quando suas opções de otimização padrão (incluindo as opções -march e -mcpu) são modificadas. Se alguma variável de ambiente que modifique estas opções de otimização tiver sido definida, tais como CFLAGS e CXXFLAGS, devem ser removidas quando for compilar este pacote. A documentação deste pacote recomenda que sua configuração seja realizada em um diretório de trabalho diferente do diretório dos arquivos fonte:
  Procedimento:
  ```
  #cd /mnt/lntpacotes/sources
  #tar -zxvf binutils-2.15.94.0.2.2.tar.gz
  #mkdir binutils-build
  ```

```
#cd binutils-build
#../binutils-2.15.94.0.2.2/configure --prefix=/tools --disable-nls
#make
#make install
#make -C ld clean
#make -C ld LIB_PATH=/tools/lib
```

- *GCC:*
  Este pacote é conhecido por apresentar problemas quando suas opções de otimização padrão (incluindo as opções -march e -mcpu) são modificadas. Se alguma variável de ambiente que modifique estas opções de otimização tiver sido definida, tais como CFLAGS e CXXFLAGS, devem ser removidas quando este pacote for copilado. A documentação deste pacote recomenda que sua configuração seja realizada em um diretório de trabalho diferente do diretório dos arquivos fonte:
  ```
  #tar -zxvf ../gcc-3.4.3.tar.gz
  #mkdir ../gcc-build
  #cd ../gcc-build
  #../gcc-3.4.3/configure --prefix=/tools \
  --libexecdir=/tools/lib --with-local-prefix=/tools \
  --disable-nls --enable-shared --enable-languages=c
  #make bootstrap
  #make install
  #ln -s gcc /tools/bin/cc
  ```

- *Linux-Libc-Headers:* Devem ser instalados os arquivos de cabeçalho:
  ```
  #tar -zxcv ../Linux-Libc-Headers-2.6.11.2.tar.gz
  #cd ../Linux-Libc-Headers-2.6.11.2
  #cp -R include/asm-i386 /tools/include/asm
  #cp -R include/linux /tools/include
  ```

- *Glibc:* O pacote de Glibc contém a biblioteca C principal. Esta biblioteca fornece as rotinas básicas de alocação de memória, busca em diretórios, abertura e fechamento de arquivos, leitura e escrita de arquivos, manipulação de strings, ”pattern matching”, aritmética e assim por diante.
  ```
  #tar -zxvf ../glibc-2.3.4
  #cd ../glibc-2.3.4
  #patch -Np1 -i ../glibc-2.3.4-fix_test-1.patch
  #mkdir ../glibc-build
  #cd ../glibc-build
  #../glibc-2.3.4/configure --prefix=/tools \
  --disable-profile --enable-add-ons \
  --enable-kernel=2.6.0 --with-binutils=/tools/bin \
  --without-gd --with-headers=/tools/include \
  --without-selinux
  #make
  #make check
  #make install
  ```

4. Ajustando as ferramentas provisórias

Agora que as bibliotecas C provisórias estão instaladas, todas as ferramentas compiladas deverão estar vinculadas ser a estas bibliotecas. Para fazer isto, o vinculador dinâmico e o arquivo de especificações do compilador precisam ser ajustados. O vinculador dinâmico, preparado no final da primeira passagem do Binutils, mas não instalado, será instalado agora executando o seguinte comando dentro do diretório de trabalho do binutils binutils-build:

```
#make -C ld install
```

Devem ser instalados os pacotes restantes conforme sua necessidade.

4.1. Montando os sistemas de arquivos virtuais do kernel

Os vários sistemas de arquivos implementados diretamente pelo kernel são usados para comunicação do e para o próprio kernel. Estes sistemas de arquivos são virtuais, tanto que nenhum espaço em disco é utilizado por eles. O conteúdo destes sistemas de arquivo reside na memória. Comece criando os diretórios onde os sistemas de arquivos serão montados:

```
#mkdir -p /mnt/lnt/{proc,sys}
```

São montados agora os sistemas de arquivo:

```
#mount -t proc proc /mnt/lnt/proc
#mount -t sysfs sysfs /mnt/lnt/sys
```

Se for necessário interromper o fluxo de trabalho no sistema LNT para recomeçar mais tarde, é muito importante montar estes sistemas de arquivos novamente antes de usar o chroot.

Os sistemas de arquivo adicionais serão montados sob o ambiente chroot. Para manter o anfitrião atualizado, deve ser executado um fake mount para cada uma destes agora:

```
#mount -f -t tmpfs tmpfs /mnt/lnt/dev
#mount -f -t tmpfs tmpfs /mnt/lnt/dev/shm
#mount -f -t devpts -o gid=4,mode=620 devpts /mnt/lnt/dev/pts
```

Entrando como chroot: É hora de entrar no ambiente chroot para começar configurar e instalar o sistema LNT final. Como usuário root, deve ser executado o seguinte comando para entrar no sistema lnt que, neste momento, está habilitado somente com as ferramentas provisórias:

```
#chroot "/mnt/lnt" /tools/bin/env -i \
HOME=/root TERM="\$TERM" PS1='\u:\w\$ ' \
PATH=/bin:/usr/bin:/sbin:/usr/sbin:/tools/bin \
/tools/bin/bash --login +h
```

4.2. Criação dos Arquivos /etc/passwd , /etc/group , /etc/fstab

A seguir uma breve explicação da função dos arquivos passwd, group e fstab.

4.2.1. PASSWD

O arquivo passwd é o banco de dados de usuários que podem logar no sistema. Tem um formato de vários campos, separados pelo caracter :(dois pontos) e sempre na mesma ordem: nome de login do usuário, senha (criptografada, evidentemente), id do usuário (identificação única, semelhante a um número de carteira de identidade), grupo primário deste usuário (o usuário poderá participar de vários grupos), nome completo (nome normal, sem ser de login), diretório home deste usuário, e shell inicial.

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

4.2.2. GROUP

Define os grupos aos quais os usuários pertencem. Seu conteúdo são linhas da forma: group_name:passwd:GID:user_list onde group_name é o nome do grupo. Um usuário pode pertencer a qualquer número de grupos, e herdará todas as permissões de acesso a arquivos desses grupos. Opcionalmente um grupo pode ter uma senha (campo passwd). O GID (group id) é um código, como o user_id (no arquivo passwd), mas relativo ao grupo.

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
mail:x:34:
nogroup:x:99:
EOF
```

4.2.3. FSTAB

O arquivo /etc/fstab é usado por alguns programas determinar onde os sistemas de arquivos devem ser montados, em que ordem e como devem ser verificados (em sua integridade) antes da montagem. Crie uma nova tabela dos sistemas de arquivos como esta:

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab
# file system mount-point type options dump fsck
# order
/dev/[xxx] / [fff] defaults 1 1
/dev/[yyy] swap swap pri=1 0 0
proc /proc proc defaults 0 0
sysfs /sys sysfs defaults 0 0
devpts /dev/pts devpts gid=4,mode=620 0 0
shm /dev/shm tmpfs defaults 0 0
# End /etc/fstab
EOF
```

5. Estrutura de pastas

Apos concluir a instalação dos pacotes, e reinicializar o sistema é gerada a estrutura de pastas conforme figura1.

5.1. raiz (root, /)

É específico para cada máquina, geralmente armazenado localmente, mas também em rede ou ramdisk (por exemplo, quando é realizada a instalação do Linux), e deve conter os arquivos para a carga do sistema, de forma a permitir a montagem dos outros sistemas de arquivos. Deverá conter também as ferramentas necessárias para recuperar backups, quando ”logamos”no sistema na forma de usuários único (single user).

5.2. /usr

Contém comandos, bibliotecas, manpages (páginas de manual), e outros arquivos estáveis, isto é, que não precisem ser modificados durante a operação normal do sistema. A idéia desta restrição, é permitir os arquivos serem compartilhados via rede com outras máquinas. Quando este diretório é montado (caso esteja em um disco ou partição separado dos demais), ele poderá inclusive ser montado read-only, para reforçar este requisito. Evidentemente, isto não se aplica quando programas novos estão sendo instalados no sistema.

5.3. /var

Contém em geral os arquivos que sofrem modificações durante a sessão, tais como logfiles, arquivos de spooling, manpages formatadas (o que acontece logo após cada man que é executado e a página não foi ainda consultada), bem como arquivos temporários. O uso tradicional de /var abaixo de /usr, torna impossível a montagem de /usr como read-only, e deve ser evitado. Uma solução é cirar um link simbólico de /var para /usr/var.[14]

5.4. /home

Contém os diretórios dos usuários normais (o superusuário utiliza o diretório /root na maioria dos sistemas). De certa forma, pode-se dizer que aí se localizam todos os dados reais do sistema. Quando este diretório se torna excessivamente grande, ele pode ser quebrado em vários, inroduzindo uma camada de nomes adicional (grupos de usuários), como por exemplo: /home/suporte, /home/clientes.

5.5. /bin

Contém programas (executáveis) que são necessário durante a carga do sistema (boot time).

5.6. /dev

Dispositivos (devices), não arquivos de dados no sentido explícito, mas que podem ser acessados, confome o caso por programas que usualmente editam, filtram ou processam de maneira geral arquivos convencionais. Todo cuidado com estes arquivos é pouco. Por exemplo, a cópia de um texto para um dispositivo como /dev/hda1 (supondo que este seja o seu disco rígido), pode deixar o sistema inoperante, reescrevendo sobre a partição ext2 do Linux. Evidentemente, para fazer isso, é necessário ser um usuário privilegiado (root).

5.7. /mnt

É um diretório com pontos para montagem de dispositivos de bloco, como discos rígidos adicionais, disquetes, cdroms, etc.

5.8. /proc

Na realidade, um diretório virtual, mantido pelo kernel, mas de extrema utilidade. Nele são encontrados todos os ”arquivos”com a configuração atual do sistema, dados estatísticos, dispositivos já montados, interrupções (e quantas vezes ocorreram desde o último boot), endereços e estados das portas físicas, dados sobre as redes, etc. Além disso, um subdiretório com o nome que corresponde ao PID (process id) de cada processo correntemente existente na máquina, aonde se encontram informações detalhadas sobre o estado do processo, linha de comando, ambiente (environment), etc.

5.9. /tmp

Local destinado aos arquivos temporários. Observe a duplicidade aparente deste diretório com o /var. Na realidade o /tmp não necessariamente precisa ser salvo entre uma sessão e outra (após um boot), enquanto que o /var normalmente fica com os dados salvos. Programas executados após o boot do sistema, devem preferencialmente usar o diretório /var/tmp, que provavelmente terá mais espaço disponível, mas nem sempre essa regra é seguida.

5.10. /sbin

Executáveis e ferramentas para a administração do sistema. Este é um dos diretórios que é disponível no PATH durante a execução do processo init.

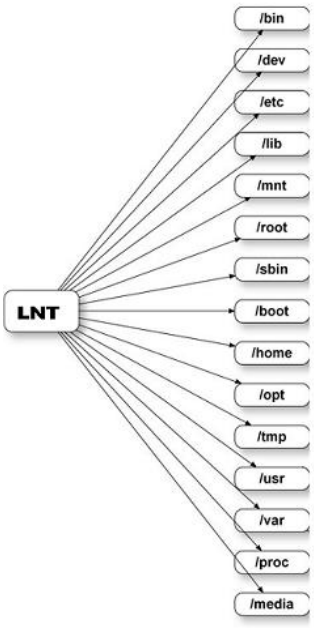

Figura 1. Estrutura de pastas!

5.11. /etc

Este diretório é um dos mais importantes. Contém uma miscelânea de dados de configuração, notadamente no subdiretório /etc/rc.d, aonde estão os scripts de inicialização do sistema em seus vários níveis[15], ademais contém os arquivos fstab (tabela de filesystems), inittab (configuração da inicialização do sistema para cada nível), profile (configuração default para todos os logins), printcap (configuração do spooler de impressão), e um número considerável de arquivos para configuração de rede e outros aspectos do sistema, incluindo o X11.

5.12. /lib

Bibliotecas do sistema, compartilhadas, também os módulos do kernel que são carregados via modprobe ou insmod.

6. Dificuldades do processo

Ao longo do processo da criação tive milhares de problemas na compilação de pacotes, não colocarei os erros pois são muitos, a maior parte dos pacotes apresenta algum erro na compilação por falta de dependências de outros pacotes que seriam instalados futuramente.

Outros não puderam ser instalados por causa da falta de uma dependência que ao tentar instalar a dependência teria que instalar a dependência do pacote que ja é a dependência, ou seja a instalação de um pacote pode afetar todo o sistema se for instalar todas dependências.

Exemplo para melhor compreensão:

- Instalação do pacote iptraf
- É necessário instalar a dependência e2fsprog.
- Ao tentar instalar o pacote e2fsprog é necessário instalar a dependência dele que é o pacote util-linux-ng

7. Conclusão

A criação do Sistema Linux Network Test, foi feita totalmente manual sem qualquer auxilio de gerenciador de pacotes foi necessário estudar sobre cada pacote, seus benefícios e suas dependências cada pacote tem um procedimento para ser instalado vários pacotes para serem instalados precisava de uma configuração especifica dificultando um pouco a instalação, é preciso entender o funcionamento do kernel para habilitar somente o necessário para o sistema ser mais rápido, foi necessário estudar sobre cada diretório para saber onde é instalado os pacotes e seus links simbólicos, o processo de criação foi de grande ajuda para entender como funciona a instalação e instalação de pacotes e estrutura do linux, infelizmente não obtive sucesso para instalar as ferramentas de gerenciamento e análise de redes, por dificuldade técnica na instalação dos pacotes.

Referências

Amaral, F. E. (2009). O que é kernel? Disponível em: <http://www.tecmundo.com.br/1636-o-que-e-kernel-.htm>. Acesso em: mar 2011.

Funet (1990). Repositorios. Disponível em: <ftp://ftp.funet.fi>. Acesso em: jun 2011.

Scratch, L. F. (1998). Linux from scratch criando um sistema linux do zero. Disponível em: <http://www.linuxfromscratch.org/>. Acesso em: mar 2011.

Wikipédia (1998). Linux. Disponível em: <http://pt.wikipedia.org/wiki/Linux>. Acesso em: mar 2011.