8a. Parte:

O Kernel e a Compilação

Índice

ABERTURA

Conforme enfatizado na *1a. Parte – Os sistemas GNU/Linux*, todas as distribuições baseadas nos sistemas *GNU/Linux* existentes são compatíveis, graças ao uso de um *kernel* único desenvolvido por uma equipe de excelentes e restritos programadores, onde *Linus Torvalds* é o coordenador.

Outra situalização analisada está no fato do *Slackware*, em comparação à maioria das demais distribuições, ter o diferencial de possui vários *kernels* customizados para o uso em situações específicas. Ainda assim, existirão circunstâncias em que necessitaremos de ter habilitadas algumas funcionalidades específicas, bem como outras que não terão necessidade de estarem presentes, ocupando memória e carga do processador.

Pelo fato do código-fonte ser aberto e da existência de diversos recursos para a sua modificação, o *kernel* possui uma grande flexibilidade para ser adaptado e recompilado com o intuito de atender os mais variados motivos. É justamente por isso é que iremos estudar o *kernel* e detalhar como funciona o processo de compilação do *kernel*, para que o usuário possa obter suporte à determinadas tecnologias ou periféricos, além de otimizar o sistema para obter uma melhor performance.

Existem inúmeras funcionalidades ao qual o *kernel* pode ser adaptado. Para nós, simples usuários *desktops*, apesar deste processo não ser tão necessário, e na maioria das vezes dispensável, nos interessará apenas conhecer suas características e possivelmente realizar algumas intervenções caso necessárias para o uso dos sistemas *GNU/Linux* em nosso cotidiano. E para isto, nos próximos capítulos iremos conhecer as características e particularidades do *kernel Linux*, além de obter informações necessárias para personalizá-lo de forma à atender as mais variadas necessidades conforme dito anteriormente. &;-D

I. Visão geral

Introdução

- ✔ <http://www.kernel.org/>.

Como todos já devem saber, o *kernel* é considerado o coração do sistema; é ele quem "*coordena*" todo o trabalho de gerenciamento e acesso aos principais recursos do sistema. Ao acessarmos a memória principal, ao executarmos aplicações, ao exibirmos informações no vídeo, ao gravarmos arquivos, enfim, todos os eventos inerentes de um sistema operacional, acreditem: é o *kernel* que entra em cena.

Neste capítulo iremos apenas conhecer um pouco sobre o *kernel Linux*, suas características, particularidades e aplicações para o uso em *desktops*.

Particularidades

O *kernel Linux* possui diversas características e particularidades técnicas, que por sua vez é associado diretamente às distribuições; por isto, optamos inserir estas informações na *1a. Parte: Os sistemas GNU/Linux -> Linux*.

Tipos e adaptações

Existem diversos tipos e adaptações possíveis para o *kernel*, mas para nós simples usuários, o que será necessário conhecer são suas características e funcionalidades básicas.

Monolítico

Um *kernel* é considerado monolítico quando os *drivers* e/ou programas de controle dos dispositivos são compilados diretamente no *kernel*. Estes *kernels* são utilizados para a inicialização de sistemas para os mais variados tipos de *hardware*. Porém em contrapartida exigem um consumo muito maior de memória *RAM* que o *kernel* modular, onde diversos destes *drivers* e/ou programas são compilados como módulos.

Modular

Ao contrário do *kernel* monolítico, no *kernel* modular os programas de controle de dispositivos são compilados separadamente como módulos ao invés de serem inclusos no *kernel*. A principal vantagem deste processo está no fato de se obter um *kernel* leve, rápido e bem enxuto. Porém, dependendo das operações à serem realizadas e dos possíveis módulos existentes, possivelmente ocorrerá uma perda de performance face ao carregamento destes módulos durante a utilização intensa dos dispositivos do sistema. Este é o principal motivo pelo qual determinadas tecnologias

devem ser compiladas diretamente (uso constante), enquanto outras devem ser compiladas como módulos (usos ocasionais).

Semi-monolítico?

Tanto o *kernel* monolítico quanto o *kernel* modular são temas de debates e discussões entre vários especialistas em desenvolvimento de sistemas operacionais. Com os desenvolvedores do *kernel Linux*, com certeza não seria diferente: enquanto alguns defendem a utilização do *kernel* modular em argumento à sua simplicidade e formato compacto, outros defendem a utilização do *kernel* monolítico, tendo como principal argumento a dificuldade em obter trocas de informações confiáveis entre os desenvolvedores, pelo fato destes estarem divididos em numerosos grupos para a administração de seus respectivos módulos.

Para intermediar estas circunstâncias, foi incorporado ao *kernel Linux* algumas características do *kernel* modular, onde estas apresentam as suas principais qualidades sem apresentar suas limitações, como a perda de desempenho e simplicidade.

As versões

Tecnicamente existem duas versões do *kernel*: A versão de desenvolvimento e a versão estável.

A versão de desenvolvimento é composta por um número de série cujo o *2o.* termo é sempre ímpar (*2.1.x*, *2.3.x*, *2.5.x*), onde são acrescentadas melhorias e modificações para que no futuro venha à ser largamente utilizado. Ao chegar à um certo grau de amadurecimento, este *kernel* se tornar a versão estável, onde é alterada a sua composição numérica para uma nova série, onde este *2o.* termo passa à ser par (*2.2.x*, *2.4.x*, *2.6.x*). Á partir da versão estável, inicia-se novamente o trabalho para desenvolvimento de uma nova versão do *kernel*, e isto gera um ciclo de desenvolvimento contínuo:

- A versão em desenvolvimento gera a versão estável – *2.1.x* -> *2.2.x*;
- A nova versão estável torna-se a base para o novo desenvolvimento – *2.2.x* -> *2.3.x*;
- Onde continua o ciclo de aperfeiçoamento – *2.3.x* -> *2.4.x* -> *2.5.x* -> *2.6.x* -> ...

Dependendo das circunstâncias, algumas modificações são incluídas no *kernel* estável para que ele possa suportar novas tecnologias que sejam bastante necessárias, procedimento este que geralmente é implementado no *kernel* de desenvolvimento para que esteja presente no novo *kernel* estável. Para exemplo, à partir da versão *2.0.31*, o *kernel* passou a suportar nomes extensos do *Windows* nas partições *VFAT*; já acima da versão *2.4.16*, passou a suportar o sistema de dados *ReiserFS*.

O ARMAZENAMENTO

OS BINÁRIOS...

Conforme as instruções da norma *FHS* e respectivamente da *LSB*, os arquivos de inicialização e o *kernel Linux* do sistema deverá estar instalado no diretório */boot* da estrutura de diretórios. Ao checarmos este diretório, encontraremos o seguinte conjunto de arquivos:

```
# ls -l
total 2912
-rw-r--r--    1 root     root          512 Jun 28 17:14 boot.0300
-rw-r--r--    1 root     root          226 Jun 28 17:14 boot_message.txt
lrwxrwxrwx    1 root     root           17 Jun 28 17:01 config -> config-
ide-2.4.20
-rw-r--r--    1 root     root        36222 Mar 18 03:03 config-ide-2.4.20
-rw-------    1 root     root        28672 Ago 17 21:05 map
lrwxrwxrwx    1 root     root           21 Jun 28 17:01 System.map ->
System.map-ide-2.4.20
-rw-r--r--    1 root     root       563791 Mar 18 03:03 System.map-ide-
2.4.20
-r--------    1 root     root      1170308 Jun 28 17:13 vmlinuz
-rw-r--r--    1 root     root      1170308 Mar 18 03:03 vmlinuz-ide-2.4.20
# _
```

A imagem do *kernel* principal corresponde ao arquivo *vmlinuz-ide-[VERSÃO]*, ao passo que o arquivo *System.map-ide-[VERSÃO]* é utilizado pelas ferramentas de manipulação de módulos. Já o *config-ide-[VERSÃO]* refere-se às opções de configuração padrão da distribuição que foram utilizados na compilação do *kernel* – correspondente ao *.config* situado no pacote do código-fonte.

O CÓDIGO-FONTE

Já o código-fonte do *kernel Linux* deverá estar disponibilizado em */usr/src*, onde deverá existir o diretório *linux-[VERSÃO]*. Dentro deste diretório, poderemos ver a seguinte árvore de diretórios:

```
# ls -l
total 211
drwxr-xr-x   19 573      573           456 Nov 28  2002 arch
-rw-r--r--    1 573      573         18691 Ago  2  2002 COPYING
-rw-r--r--    1 573      573         79594 Nov 28  2002 CREDITS
drwxr-xr-x   30 573      573          3416 Nov 28  2002 Documentation
drwxr-xr-x   41 573      573          1032 Nov 28  2002 drivers
drwxr-xr-x   46 573      573          2240 Nov 28  2002 fs
drwxr-xr-x   26 573      573           696 Nov 28  2002 include
drwxr-xr-x    2 573      573           136 Nov 28  2002 init
drwxr-xr-x    2 573      573           192 Nov 28  2002 ipc
drwxr-xr-x    2 573      573           736 Nov 28  2002 kernel
drwxr-xr-x    4 573      573           536 Nov 28  2002 lib
-rw-r--r--    1 573      573         42744 Nov 28  2002 MAINTAINERS
-rw-rw-r--    1 573      573         18780 Nov 28  2002 Makefile
drwxr-xr-x    2 573      573           608 Nov 28  2002 mm
```

```
drwxr-xr-x   28 573       573           888 Nov 28  2002 net
-rw-r--r--    1 573       573         14239 Ago  2  2002 README
-rw-r--r--    1 573       573          2815 Abr  6  2001 REPORTING-BUGS
-rw-r--r--    1 573       573          9291 Ago  2  2002 Rules.make
drwxr-xr-x    4 573       573           928 Nov 28  2002 scripts
# _
```

Este código-fonte compõe-se da seguinte estrutura:

Código-fonte do kernel Linux	
arch/	Códigos independente de arquitetura.
documentation/	Documentação relativa ao *kernel*.
drivers/	Diversos códigos de drivers de dispositivos.
fs/	Sistema de arquivos.
include/	Definições e protótipos de funções (linguagem *C*).
init/	Definições de inicialização.
ipc/	Comunicação entre processos.
kernel/	Código-base do *kernel*.
lib/	Funções básicas (bibliotecas).
mm/	Definições de gerenciamento de memória.
modules/	Módulos compilados (versões anteriores à *2.4*).
net/	Interligação em redes.
scripts/	*Scripts* diversos para configuração.

Além do subdiretório */Documentation*, existem diversos documentos para consulta, como o *README*, o *REPORTING-BUGS*, o *COPYING* e o *CREDITS*. Sempre que puderem, gastem alguns minutos e dê uma olhada neles ou pelo menos no arquivo *README*. &;-D

Poderemos armazenar diversas versões do *kernel*, apenas atribuindo aos seus respectivos diretórios o nome e a versão corrente destes para que, na eminência de uma necessidade, estes estejam sempre à disponibilidade.

Desenvolvimento

Conforme brevemente comentado na *1a. Parte: Os sistemas GNU/Linux -> O Linux*, o *kernel* é desenvolvido graças à ajuda de milhares de colaboradores, que tendo em mãos o seu código-fonte, ajudam nas mais diversas formas: sejam aperfeiçoando o próprio código, analizando, comunicando erros, dando sugestões, etc., enfim, colaborações que hoje o tornam um dos maiores projetos de código-aberto atualmente.

Encabeçado por *Linus Torvalds* e com grande apoio de seu braço direito *Alan Cox*, o *kernel Linux* possui um grupo de desenvolvedores, subdividido

por uma rígida hierarquia, onde as sugestões enviadas são avaliadas pelos membros da equipe, porém em casos de opiniões adversas, somente os líderes de cada hierarquia é que dão a palavra final.

Tendo um grupo de desenvolvedores formado por programadores de diversos países do mundo inteiro, fica a pergunta no ar: "*será que não tem nenhum representante brasileiro neste grupo*"? Ah, tem sim! Este representante é o gaucho *Martelo Tosatti*, que faz parte do grupo de desenvolvedores do *kernel Linux*. Há pouco tempo, ele era somente o responsável pela manutenção da versão corrente do *kernel*, porém em virtude de um convite do próprio *Linus Torvalds*, *Tosatti* passará a desenvolver outros componentes importantes deste sistema operacional.

Conclusão

Enfim, após a leitura de todo este capítulo, acreditamos que os usuários estejam pensando que aprenderam praticamente tudo sobre o *kernel Linux*, correto? Errado. De acordo com a visão e a necessidade de cada usuário ou entidade, o *kernel Linux* possui aspectos e particularidades tais que poderiam render um livro inteiro se tivéssemos que detalhá-lo por completo. Aqui apenas colocamos as considerações mais básicas e importantes para obtermos uma visão geral. Com estas informações, o usuário apenas terá um entendimento básico, porém suficiente para entender a mais importante peça do sistema operacional – não é à toa que é chamado de núcleo! &;-D

II. Bons motivos para a compilação

Introdução

A compilação do *kernel* está entre as maiores vantagens para a adoção de sistemas operacionais livres como o *GNU/Linux* e *BSDs* em comparação aos outros proprietários existentes, justamente por ser possível realizar a otimização de seus recursos em virtude de se encontrar disponivel o seu código-fonte para a realização desta operação. Este motivo, por si só, já é o suficiente para justificar o uso deste processo. Mas especificamente para nós, simples usuários, quais motivos poderemos ter para realizar este procedimento? Quais são suas vantagens? Em quê mais específicamente nós iremos nos beneficiar? Quais são os riscos?

Enfim, para estas e muitas outras perguntas, elaboramos um capítulo especial à parte para o esclarecimento das dúvidas mais comuns dos usuários iniciantes que se decidiram em usar deste processo.

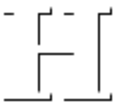
Bons motivos...

Suporte a periféricos e programas

Um dos principais motivos que levam os usuários a recompilarem o *kernel* está na necessidade de suporte à novos periféricos e tecnologias, habilitando-as conforme necessário. Em virtude da imensa variedade máquinas com perfis de *hardwares* existentes, os sistemas *GNU/Linux* geralmente reconhecem a grande maioria destes, porém ainda existem muitos componentes que, por mais variados motivos, não são suportados pelo *kernel* ou seu suporte não se encontram habilitados. O mesmo se dá para determinados programas; determinadas tecnologias deverão estar habilitadas no *kernel* do sistema para que esses programas funcionem corretamente, como é o caso dos sistemas de gerenciamento de *hardware*.

Otimização e personalização

Por padrão e de acordo com cada *kernel* disponibilizado pelas mais diversas distribuições, existem recursos diversos que são disponibilizados para atender a grande maioria dos usuários; porém, por mais "*exóticos*" que sejam tais usuários, a grande maioria dos recursos presentes serão desnecessários para seu uso. A compilação do *kernel* provê um conjunto de recursos mais enxuto ao sistema o qual se encontra instalado. Basta apenas personalizá-lo de acordo com o equipamento utilizado, onde teremos um novo *kernel*, mais simples, limpo, rápido, enxuto e mais eficiente.



Suporte específico para cada arquitetura

Em virtude da existência de diversas arquiteturas, a grande maioria das distribuições fornecem o *kernel* pré-compilado para atender as mais variadas existentes. Por exemplo, o *kernel* do *Slackware* é compilado para suportar processadores à partir do *i386/i486*, ao passo que o *Mandrake*, com seus vastos recursos, somente suporta processadores *i585* em diante em virtude da demanda de processamento e de *hardware*.

O *kernel* pode ser otimizado para suportar os recursos específicos de uma única plataforma, habilitando suas extensões e otimizando as instruções do sistema somente para aquela arquitetura, onde o ganho de desempenho são sensíveis graças à este procedimento.

Compatibilidade

A possibilidade de ajustar o *kernel* para específicas finalidades o torna altamente compatível com as tecnologias e requesitos atuais. Suporte à dispositivos exóticos e periféricos "*desconhecidos*" são também garantidos com algumas alterações na configuração do *kernel* e respectiva utilização.

Experiência e aprendizado

Para os engenheiros, técnicos de manutenção, profissionais de diversos ramos e até mesmos simples usuários do sistema, a compilação do *kernel* é uma ótima oportunidade de aprendizado para melhor conhecer os fundamentos básicos do funcionamento de um sistema operacional. Conhecendo à fundo este processo, poderemos dominar quaisquer aspectos referentes às suas particularidades e conseguiremos condicioná-lo para as mais variadas situações.

Conclusão

Apesar de ser até um certo ponto desnecessário – e até mesmo não recomendável –, existem diversas vantagens oferecidas pela utilização deste procedimento. Além de um pequeno ganho de performance e desempenho (às vezes sensível), o aprendizado é um dos fatores que consideramos de suma importância. Conhecer algumas características e particularidades do *kernel Linux* nos beneficiará muito quando houver necessidade de realizar intervenções, como a instalação e suporte de um periférico, a adaptação de determinados parâmetros, entre outros. Além disso, aprendizado nunca é demais! &;-D

III. Preparativos iniciais

Introdução

Como qualquer outra operação que envolve riscos de perdas de dados, será necessária a realização de alguns preparativos iniciais que visam dar assistência ao desenvolvimento do processo e ao mesmo tempo resguardar o usuário contra possíveis falhas e sinistros que poderão ocorrer sob os mais variados motivos. Por se tratar da compilação do *kernel* – justamente o núcleo do sistema operacional – é lógico que toda a atenção dada à este aspecto é no mínimo... merecida! &;-D

Procedimentos necessários

Segue os principais preparativos básicos e essenciais para a realização deste procedimento com segurança e praticidade, lembrando ainda que poderá haver a necessidade de realização de outras atividades durante a compilação do *kernel*. Enfim, estejam preparados! &;-D

Cópia de segurança dos dados do sistema

Na verdade, não são muito comuns os casos em que os processos de compilação do *kernel* feitos incorretamente produzem perdas de dados, porém, de acordo com o nosso nível de conhecimento e a possibilidade de não termos mais acesso ao sistema, será mais cômodo a utilização desta operação para resgate. Caso algo não dê certo, bastará apenas realizarmos a reinstalação do sistema, restaurando todas as definições anteriores e os dados previamente salvaguardados.

Para esta atividade, consultem a *2a. Parte: Conhecimentos Gerais -> Manipulação de arquivos e diretórios*, para obterem maiores informações para a realização da cópia de segurança com segurança e praticidade.[1]

Disquetes de inicialização

A utilização de disquetes de inicialização é necessária especialmente quando ocorrem erros no processo de compilação que possam resultar a não inicialização do sistema, o qual somente poderá ser feito utilizando-se este recurso. Este tipo de ocorrência é bastante comum o que torna praticamente indispensável a sua disponibilidade.

Para isto, bastará inserir uma mídia no *drive* e digitar...

```
$ mkbootdisk --device /dev/[DISQUETE] [KERNEL]
```

1 Utilizem os processos que desejarem, porém dêem preferência às ferramentas nativas do sistema, como o empacotador *TAR* e os compactadores *bzip2* e *gzip*, ou que ainda utilizem estes formatos. Em situações mais drásticas, será mais fácil intervirmos com estes utilitários à mão.

Para obterem maiores informações, consultem a *2a. Parte: Conhecimentos Básicos: -> Unidades, partições e formatos*.

Cópia de segurança da compilação anterior

Ao realizarmos a compilação do *kernel* e o sistema estiver rodando sem maiores problemas, poderemos optar por realizar uma cópia de segurança do arquivo de configuração gerado pela compilação anterior. Este arquivo se encontra na raiz deste diretório e se chama *.config*. Salvem-o em um local adequado, caso venha à ocorrer algum problema.

Se por algum motivo não o encontrarem, entrem no diretório do antigo código-fonte do *kernel* e executem o programa de configuração...

```
# make xconfig
# make gconfig
# make menuconfig
```

... com qualquer uma das três opções acima.

Em todas elas haverá uma opção para salvarmos a configuração padrão em um arquivo separado. Utilizem-na, definindo o nome *.config* e salvem-no em um local adequado, de preferência junto com os demais dados.

Se preferirem, podem também copiar o arquivo *config.in* disponível em */usr/src/linux/arch/i386* para */usr/src/linux*, renomeando-o para *.config*.

Manutenção da versão anterior do kernel

É opcional, porém recomendada a manutenção da antiga versão do código-fonte do *kernel* compilado anteriormente no sistema. Evitem excluí-lo, pois dependendo das circunstâncias, poderemos necessitar dele. Isto ocorre muito com as novas versões do *kernel*, logo após que estas são lançadas.

Caso o código-fonte esteja armazenado em um diretório simplesmente chamado *linux* – e não um atalho – deveremos renomeá-lo para a versão o qual pertence, no seguinte formato:

```
# mv linux linux-[VERSÃO]
```

Caso não saibam qual a versão do *kernel* utilizado pelo sistema...

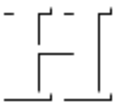
```
# uname -r
2.4.18
# _
```

Utilizem os dados exibidos e renomeiem o diretório, conforme solicitado.

```
# mv linux linux-2.4.18
```

Requerimentos básicos

Dentre os principais requerimentos básicos, destacam-se a satisfação das seguintes pendências:



- O compilador *GCC* e a biblioteca padrão *GNU C*;
- As ferramentas *GNU* que acompanham a distribuição;
- Os pacotes *TCL/Tk* e o servidor *XFree86* e/ou *X.org* instalados para a opção *xconfig*. Já no *kernel 2.6*, esta opção requer a biblioteca *Qt*;
- O pacote *ncurses* para a opção *menuconfig*;
- O arquivador *TAR* e os compactadores *gzip* e *bzip2*;
- O código-fonte do *kernel* – lógico... &;-D

Todas as pendências necessárias encontram-se disponíveis no *CD-ROM* de instalação em qualquer distribuição *GNU/Linux*, onde estas são requeridas durante a instalação para o perfeito funcionamento do sistema.

Obtendo o código-fonte do kernel

- ✔ *<http://www.kernel.org/>.*
- ✔ *<ftp://ftp.kernel.org/pub/linux/kernel/>.*

O código-fonte do *kernel* pode também ser obtido diretamente no próprio *FTP* ou na página eletrônica oficial do projeto, onde poderemos encontrar as versões estável (série *2.4.x*) ou de desenvolvimento (*série 2.5.x*). Após o término da operação, copiem-no para a pasta */usr/src*, onde por padrão deverão ser armazenados os fontes do *kernel*. No *FTP*, os códigos-fontes são armazenados no diretório */pub/linux/kernel/v.x.y/*, onde *v.x.y* corresponde à versão do *kernel* que desejamos utilizar.

Existem diversos endereços eletrônicos e espelhos os quais poderemos baixar o código-fonte do *kernel*, caso os endereços eletrônicos disponíveis estejam sobrecarregados.

Preparando a compilação

Instalação do novo código-fonte

Após obtermos o novo código-fonte, descompactem-no no local indicado conforme a norma *FHS* – neste caso, em */usr/src*.

```
# tar -xpzvf linux-[VERSÃO].tar
```

Após descompactá-lo, criem um atalho denominado *linux* para apontar para o diretório onde se encontra o novo *kernel* do sistema.

```
# ln -s /usr/src/linux-[VERSÃO] linux
```

Possivelmente poderão existir casos em que encontremos outras versões do *kernel* disponíveis em */usr/src*, além do respectivo atalho *linux* apontando para um destes diretórios. Neste caso excluam o antigo atalho e recriem-no novamente conforme exemplo abaixo:

```
# rm linux
```

```
# ln -s /usr/src/linux-[VERSÃO] linux
```

A necessidade da criação do atalho simbólico caracteriza-se pelo fato da existência de inúmeros programas que necessitam acessar as instruções contidas no código-fonte do *kernel* operante.

Mantendo ou removendo as definições de parâmetro anteriores

Quando resolvemos realizar a recompilação de um mesmo *kernel* utilizado anteriormente, poderemos utilizar as mesmas definições realizadas na compilação anterior. Para isto basta apenas dar continuidade ao processo de compilação, iniciando os programas de configuração necessários. Isto também é válido para os *kernels* da mesma versão com correção de *bugs* (por exemplo, *2.4.10* -> *2.4.18*, etc.).

Porém, quando desejamos realizar um novo perfil de configuração, deveremos então "*limpar*" as definições de arquivos de configurações anteriores. Para isto, faz-se necessário a utilização do comando *make mpromper*...

```
# make mpromper
```

... o qual se encarregará de realizar a exclusão das definições contidas na árvore de diretórios onde se situa as definições anteriores.

Conclusão

A realização de preparativos iniciais em processos complexos e críticos é fator de fundamental importância para o seu sucesso. Na compilação do *kernel*, com certeza não seria diferente. Para isto, solicitamos uma certa atenção na utilização das instruções contidas neste capítulo, pois de acordo com suas necessidades, elas serão fatores preponderantes no sucesso da compilação do *kernel*.

Existirão também diversos outros aspectos importantes à serem observados, como também estes aspectos deverão variar de acordo com o equipamento utilizado; aqui citamos apenas as situações mais comuns. &;-D

IV. Configuração de suporte e parâmetros

Introdução

Chegamos à uma nova etapa para a realização da compilação do *kernel*. Trata-se da configuração de suporte e parâmetros específicos para a máquina o qual desejamos utilizar. Neste capítulo, iremos conhecer os principais utilitários de configuração para o ajuste de parâmentros do *kernel*, além de tecer breves comentários sobre as principais categorias de opções existentes, visto que será praticamente inviável descrever item-a-item em virtude da sua imensa quantidade.

Utilitários de configuração

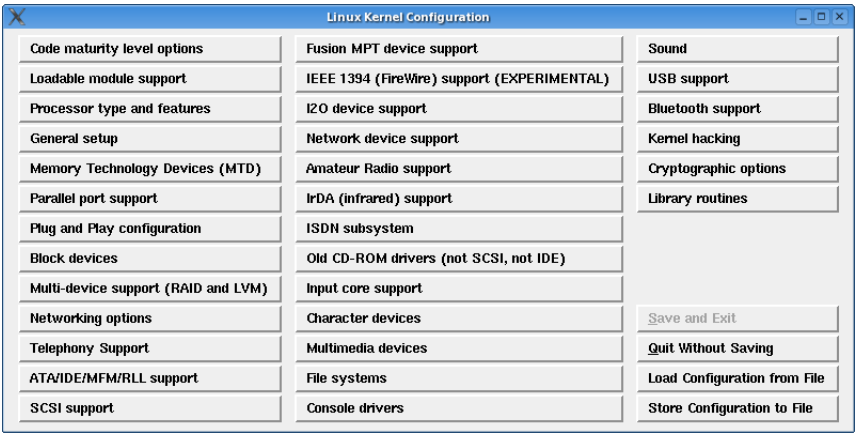
Após esta operação será necessário iniciar os programas de configuração para que possamos habilitar os parâmetros de configuração do *kernel*. Felizmente dispomos três excelentes opções:

- *make xconfig;*
- *make menuconfig*;
- *make config*.

Todos estes três utilitários fornecem um menu de acesso e, apesar de possuírem diferentes telas de apresentação, todas as opções do *kernel* se encontram disponibilizadas. Atendem perfeitamente bem às expectativas, porém cada um possui um melhor rendimento em circunstâncias específicas, das quais iremos analisar.

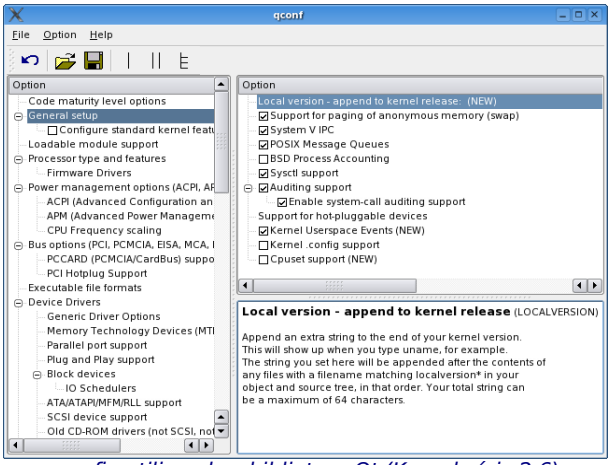
MAKE XCONFIG

Esta opção fornece aos usuários, que podem disponibilizar a interface gráfica *X*, um menu gráfico simples e bem organizado, onde com apenas alguns cliques do *mouse* poderemos ter acesso à todos as categorias.



Xconfig utilizando a biblioteca TK (Kernel série 2.4).

Na versão atual (*2.6*) dispomos somente da opção make *xconfig*, onde o mesmo utiliza a biblioteca *Qt* para a construção da caixa de diálogo.

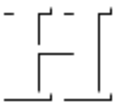


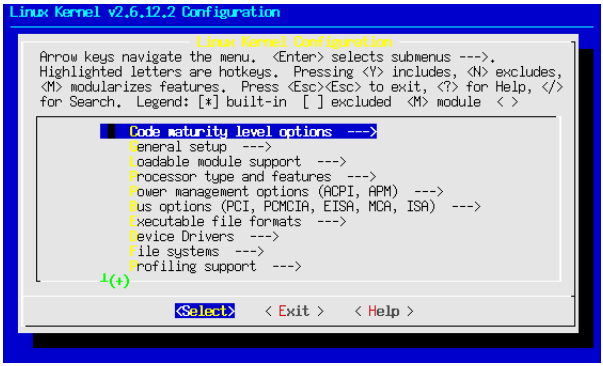
xconfig utilizando a biblioteca Qt (Kernel série 2.6).

Para equipamentos com razoáveis recursos de *hardware* e quantidade de memória, estas são as opções mais indicada, pelas facilidades e confortos proporcionados aos administradores.

MAKE MENUCONFIG

A opção *menuconfig* também oferece um menu prático, porém em modo texto, onde a seleção dos ítens é feita utilizando-se o teclado.





Menuconfig utilizando a biblioteca ncurses

É recomendado o uso deste utilitário principalmente quando ocorre algum problema que impeça o uso da interface gráfica ou quando se deseja sobrecarregar o mínimo possível os recursos do sistema. Realizar a compilação do *kernel* com equipamentos de pouco recurso e ainda no modo texto, onde existe a necessidade de uma boa quantidade de memória realmente esta é a opção mais recomendável.

MAKE CONFIG

Por último, a opção *config* somente é utilizada quando existe algum problema que impeça a utilização das demais opções citadas.

```
* Linux Kernel Configuration
*
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers (EXPERIMENTAL) [Y/n/?]
_
```

Esta opção é vista por muitos técnicos experientes como "*espartana*", pois além de ser a menos amigável, infelizmente apresenta as opções de forma seqüencial, onde impossibilita alterar qualquer configuração realizada previamente, ou seja, não existe a possibilidade de retornar para uma opção anterior para corrigir. Outro grande inconveniente é que, quando se utiliza os demais utilitários (*menuconfig* e *xconfig*), de acordo com ítens específicos que por ventura sejam desabilitados, suas sub-opções, exibidas de forma identadas, ficam indisponíveis, porém podemos visualizá-las; já esta opção simplesmente "*pula*" estes ítens, dos quais infelizmente sequer poderemos ter noção do que deixamos para trás.

SOBRE A HABILITAÇÃO DE PARÂMETROS E SUPORTES

Para habilitar e/ou desabilitarmos as opções disponíveis durante a configuração do *kernel*, teremos que redefinir como estes serão utilizados pelo próprio *kernel*. Para isto, deveremos marcar as opções presentes no

menu com as seguintes siglas:

	Opções & Finalidades
Y	*Yes* – Habilita as opções selecionadas para que sejam embutidas no *kernel* principal (monolítico).
M	*Modules* – Habilita a opções selecionadas, porém somente serão carregadas sob demanda, conforme a necessidade do sistema para o seu uso (modular).
N	*No* – Desabilita as opções selecionadas.

Em muitas circunstâncias não teremos disponível a opção de habilitação ora como módulos (*M*), ora embutidos (*Y*), de diversas recursos e tecnologias de acordo com suas características e utilização.

MÓDULOS: QUANDO UTILIZAR?

É somente recomendada a habilitação das opções como módulos todos e quaisquer dispositivos que são utilizados ocasionalmente, pois além de mantermos o *kernel* enxuto, utilizaremos menor carga de processamento e *hardware*, ganhando com isto melhor performance geral em sua utilização.

AJUSTANDO OS PARÂMETROS DE CONFIGURAÇÃO

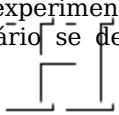
Ao acessarmos a interface de configuração para a compilação do *Kernel*, nós teremos à disposição uma série de opções estruturadas por categorias, onde basta apenas navegarmos por estas estruturas e marcar e/ou desmarcar as opções que desejarmos habilitar e/ou desabilitar. Dada as circunstâncias, talvez necessitaremos atribuir valores para configurações específicas, mas nada que a consulta de seu respectivo *help* (mesmo que em inglês) não possa ajudar. &;-D

Utilizaremos somente as principais opções onde encontra-se disponibilizado pelo *xconfig* / *gconfig*, pois além de ser mais fácil de visualizar, nos sentiremos mais confortáveis com apenas a realização de alguns simples cliques. Mas nada impede que utilizemos as demais opções existentes.

Segue abaixo a descrição das principais categorias para ajustes e configuração, onde descrevemos instruções básicas sobre as opções mais importantes do sistema. Conforme dito anteriormente, em virtude da extensa possibilidade de configuração, será inviável uma descrição mais detalhada das opções subseqüentes, pois o excesso de informações poderá comprometer o bom aprendizado ao invés de auxiliar. &;-(

CODE MATURITY LEVEL OPTION

Suporte à códigos imaturos e experimentais. Não existe muito mistério. Basta apenas informar ao utilitário se deseja habilitar as opções que se



encontram disponíveis ao *kernel* em caráter experimental, como diversos *drivers* existentes para o suporte de recentes periféricos e tecnologias. Quando desabilitada, as demais opções disponíveis em outras categorias de caráter experimental permanecem sombreadas e/ou indisponíveis, sendo impossível habilitá-las.

IMPORTANTE: CASO TENHAM ESCOLHIDO A OPÇÃO *Y* (SIM), A HABILITAÇÃO DE SUPORTE *EXPERIMENTAL* PODERÁ TORNAR O *KERNEL* INSTÁVEL E SUJEITO À FALHAS COM GRAU DE RISCOS VARIÁVEIS. UTILIZEM-NAS SOMENTE SE HOUVER REAL NECESSIDADE.

Loadable module support

Suporte à módulos. Habilita/desabilita o suporte ao carregamento de módulos dinâmicos ao *kernel*. É de grande importância e extremamente necessário que esta opção esteja habilitada para se sejam utilizados os módulos para a utilização de diversos dispositivos do sistema.

Processor type and features

Tipos de processador e implementações. Realiza a compilação do *kernel* ajustando-o com os recursos e características técnicas do processador processador selecionado. Lembrem-se que, quando configurada esta opção, o novo *kernel* suportará somente o modelo do processador instalado na máquina em uso, não servindo praticamente para nenhum outra arquitetura. Como muito provavelmente não iremos utilizar o novo *kernel* em outras máquinas, esta restrição tem pouca importância.

Nesta categoria encontra-se também o suporte à *SMP* – opção *Symetric multi-processing support* – que habilita o recurso de suporte à múltiplos processadores de uma única máquina. Utilizem-na somente se possuírem um equipamento dotado de *2* processadores centrais ou se possuírem processadores de núcleo duplo, como o *Pentium HT*.

Também poderemos realizar a emulação do processador matemático, através da opção *Math emulation*. Recomenda-se manter desabilitada esta opção, pois somente processadores *386* e *486 SX* não possuem processador matemático, o que provavelmente não será o caso.

General setup

Opções gerais de configuração, onde a maior parte se encontram aqui.

Dentre as opções de destaque, encontra-se a *ISA bus support*. Nesta opção, muitos usuários donos das mais recentes placas-mães tendem à desabilitá-lo. Isto porque estes julgam que, pelo fato destas não utilizarem mais o *slot ISA*, acreditam que será desnecessário habilitar este suporte. Não confundam barramento com interface, pois o termo "*barramento*" refere-se à implementação das vias de comunicação dos dispositivos com a *BIOS*, ao contrário do termo "*interface*", que refere-se apenas ao encaixe existentes

para antigos periféricos. O barramento ainda é utilizado para ter acesso à alguns dispositivos do computador, como as portas seriais e controladores do *drive* de disquetes. Sem esta opção, simplesmente não poderemos fazer o de periféricos que utilizam esses canais, como é o caso do *drive* de disquete, por exemplo.

Memory Technology Devices (MTD)

A seção *Memory Technology Devices* refere-se ao tratamento dado ao *kernel* aos dispositivos de memória eletrônica (*RAM*, *flash*, etc.) usados como sistema de arquivos em dispositivos embutidos. Para os usuários *desktops*, não há parâmetros específicos para serem alterados, visto que por padrão esta se encontra desabilitada.

Parallel port support

Especificações gerais para o suporte à porta paralela. A porta paralela é necessário para a utilização muitos dispositivos e periféricos, em especial as impressoras, onde encontram-se opções que poderão melhorar a integração e performance destes periféricos no sistema. Recomenda-se habilitá-la como módulo, face à sua pouca utilização.

Plug and Play Configuration

As opções contidas nesta seção são extremamente importantes para a detecção de diversos *hardwares* disponíveis.

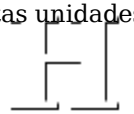
Das opções importantes, destaca-se a *Plug and Play Support*, que refere-se ao suporte à dispositivos *Plug-and-Play*. Pelo fato da maioria dos periféricos suportarem esta tecnologia, é recomendável a sua habilitação.

Também é de importante crédito a opção *ISA Plug and Play support*, o qual habilita o suporte à dispositivos *Plug-and-Play* que utilizam interfaces *ISA*. É recomendado especialmente para periféricos antigos, como as antigas placas de som *SoundBlaster AWE32/64* e *hardmodens* de *33.6 kbps*.

Block devices

Em *Block devices* encontram-se as opções de configuração para o suporte à diferentes unidades, controladoras e *chipsets* do sistema, que vão desde uma simples unidade de disquete, passado por diferentes controladoras de discos rígidos, *chipset* específicos às unidades *SCSI*.

Nesta seção apenas deveremos conferir se estão habilitadas todas as opções referente às unidades do sistema em uso (disquete, disco rígido, *CD-ROM*, etc.), onde as demais poderão ser desabilitadas desde que tenham certeza de que não fazem uso, pois caso contrário o sistema não dará suporte à leitura e gravação nestas unidades.



Multi-device support (RAID and LVM)

Nesta seção encontraremos suporte às tecnologias *RAID* e *LVM*.

O *RAID – Redundant Array of Inexpensive Disks* – é um recente recurso utilizado para garantir a confiabilidade de dados em unidades de disco rígido, o qual consiste em espelhar os dados de um sistema simultaneamente duas simples unidades (sem grandes garantias de qualidade), de forma que, na iminência de uma destas apresentarem problemas de funcionamento, teremos disponível outra unidade com os mesmos dados armazenados, possibilitando assim a substituição da unidade defeituosa sem maiores perdas. Já o *LVM – Logical Volume Manager* – é uma metodologia de gerenciamento de partições de disco rígido que visa melhorar a utilização do espaço disponível através de definições lógicas (não reais) de tamanho.

Geralmente os usuários comuns sequer necessitam destes recursos, onde a maioria das distribuições optam por deixá-los desabilitados.

Networking options

Como o próprio título da seção indica, nesta categoria estarão inclusas as opções gerais de configuração para suporte à rede, pois mesmo que o equipamento esteja isolado de quaisquer outro, será necessária a habilitação de algumas opções aqui descritas (por exemplo o acesso à *Internet*). Lá também se encontra o suporte ao *PPP*, necessário para realizar a conexão com a *Internet*. Marquem-na como módulo.

Telephony Support

A seção *Telephony Support* serve unicamente para habilitar aplicações específicas "*voz-sobre-IPs*". Não se enganem ao pensar que este recurso será necessário para a configuração da *Internet*; podemos desabilitá-lo sem maiores preocupações, se assim desejarmos.

ATA/IDE/MFM/RLL support

Suporte à controladoras necessárias para a utilização das unidades de discos rígidos e leitores de *CD-ROM* que utilizam a interface *ATAPI/IDE*.

Basta manter as opções padrões, visto que praticamente todas as distribuições mantém estas opções habilitadas por padrão. Na subseção *IDE, ATA and ATAPI Block devices* encontra-se uma série de opções que visam habilitar determinados recursos e periféricos específicos. Verifiquem cautelosamente e – se assim desejarem – tenham cuidados com os quais deverão ser redefinidos ou não. Para isto, deveremos conhecer profundamente as placas-mãe e controladoras das unidades em uso.

SCSI SUPPORT

Esta seção é referente ao suporte à todos os periféricos *SCSI*.

Uma atenção importante está na utilização de dispositivos dotados de memória eletrônica e de gravadores de *CD-R/RW*, pois apesar de não utilizarem uma interface *SCSI*, estes periféricos necessitam da emulação *SCSI* para que possam ser utilizados nos sistemas *GNU/Linux*.[2] Em virtude da grande popularidade dos gravadores de *CD-R/RW*, se encontram habilitadas por padrão em praticamente todas as distribuições.

FUSION MPT DEVICE SUPPORT

Suporte ao barramento *SCSI* para a melhora de desempenho em periféricos compatíveis com *Fusion MPT*. Habilitem-no, se estiverem providos destes periféricos. Caso contrário, mantenham as definições padrões.

IEEE 1394 (FIREWIRE) SUPPORT (EXPERIMENTAL)

Suporte ao barramento serial *FireWire*, também conhecido como *IEEE 1394*. Com os recursos e conceitos similares ao *USB*, sua principal diferença está possibilidade de intercomunicação entre os dispositivos conectados, além de obter uma taxa de transferência superior[3].

Para os micros – e especialmente os *notebooks* – dotados de portas *FireWire*, que por sua vez possibilita a conexão com periféricos que utilizam esta tecnologia, estes necessitarão de ter o suporte do *kernel* habilitado, levando-se em consideração que esta implementação encontra-se em fase EXPERIMENTAL. Exemplos de dispositivos que utilizem esta conexão estão àqueles que trabalham com grande volume de dados, como discos rígidos, câmeras digitais e algumas placas de som para uso profissional.

I2O DEVICE SUPPORT

O *I2O* – *Intelligent Input/Output* – é uma arquitetura de *hardware* que possibilita à *CPU* central deixar à cargo de um periférico o processamento de endereços *I/O*, deixando-o livre para a realização de outras atividades e conseqüentemente melhorando a performance geral do sistema.

Para utilizar este recurso, será necessário a obtenção de uma placa especial dotada de um processador exclusivo para esta função. A mesma utiliza um *slot PCI*, e para habilitar o funcionamento desta, será necessário ter o suporte *I2O* do *kernel*. Neste caso, estas opções deverão estar habilitadas.

2 Estas considerações são válidas somente para o *kernel 2.4*. A versão *2.6* dispensa o uso do recurso de emulação *SCSI* para o uso dessas unidades.

3 Isto era até antes do lançamento da versão *2.0* do barramento *USB*.

Network device support

Aqui se encontrarão diversas opções referentes à tecnologias, recursos, protocolos e periféricos de uso exclusivo para redes e *Internet*. Sem maiores detalhes, deveremos manter habilitada a opção principal – *Network device support* –, além de checar cuidadosamente as demais opções conforme sua utilização.

Amateur radio support

Suporte à recursos gerais para rádio amador. Como pouquíssimos usuários *desktops* utilizam estes recursos, poderemos desabilitar estas opções.

IrDA (infrared) support

Suporte à dispositivos infravermelhos. Sem maiores explicações, somente deveremos utilizar este suporte para periféricos que utilizem este recurso, como é o caso de alguns teclados e *mouses*, entre outros periféricos.

ISDN subsystem

O *ISDN – Integrated Services Digital Network* – é outra modalidade de conexão discada para a *Internet*, onde a velocidade de conexão é dobrada à *128 kbps*: a linha é dividida em dois canais de *64 kbps* independentes, os quais possibilitam a utilização da linha telefônica tanto para uso normal quanto para a *Internet* simultaneamente. Para isto, será necessário a utilização de placas de rede que suporte a tecnologia. Por padrão mantenham esta opção ativada.

Old CD-ROM drivers (not SCSI, not IDE)

Suporte à antigos *drives* de *CD-ROM* que não utilizam conectores *IDE* e *SCSI*. Lembram-se daqueles antigos *kits* multimídia com placa de som, microfone e drive de *CD-ROM*, onde este último é conectado à placa de som para a transferência de dados? Pois bem, estes somente poderão ser suportados pelo *kernel* quando habilitadas estas opções.

Input core support

Necessário para a habilitação de dispositivos *USB*, tais como *mouses*, teclados, *josticks*, etc. Mantenha habilitada, caso utilize-los.

Character devices

Suporte à diversos periféricos do sistema.

Importante: na seção referente aos *chipsets* das placas-mãe, não habilite o suporte para àqueles fabricados pela *SiS*, pois ainda existem alguns

problemas que impedirão a sua correta compilação.

Dentre outras opções importantes, destaca-se a *Virtual terminal* deverá permanecer habilitada, para que possamos trabalhar com múltiplos terminais (consoles) no sistema, a *Direct Rendering Manager (X.org DRI)*, necessário para algumas aceleradoras gráficas, como a *série GeForce / nVidia*, o suporte a *Framebuffer* e *VESA* e uma categoria especial para suporte à *joysticks* e placas de som.

Multimedia devices

Nesta seção encontraremos as opções *Video for Linux* e *Radio Adapters*, para a habilitação de captura de vídeo e rádio amador respectivamente. Se a estação de trabalho do usuário se encontra nesta categoria – o que é bem raro –, mantenha estas opções habilitadas.

File systems

Seleção e habilitação de diferentes sistemas de arquivos suportados pelo *kernel Linux*. Além dos sistemas de arquivos tradicionais do ambiente (*ext2*, *ext3*, *ReiserFS*, *swap*, etc.) os sistemas de arquivos *MSDOS* e *VFAT* deverão ser habilitados para o uso de outros dispositivos, como os disquetes (deverão ser habilitados como módulos, face à pouca utilização).

Console drivers

Mais algumas configurações e suportes para a placa de vídeo no modo texto, especialmente para os terminais. Dentre diversas opções de suma importância, asseguintes deverão estar marcadas:

Console drivers	
VGA text console	Suporte ao modo texto para a inicialização do sistema.
Video mode selection support	Ajuste de diferentes modo de vídeo de acordo com as personalizações do *LILO*.

Sound

Suporte às tecnologias e recursos de áudio em geral. Não só ativaremos o suporte do *kernel* à determinadas placas de som como também seus recursos, como suporte à *joystick*, canais de áudio, módulos do projeto *OSS*, etc. Recomendamos cuidados para a manipulação destas opções.

USB support

Sem grandes mistérios, aqui se encontram os principais parâmetros para a habilitação e suporte das portas *USB* do sistema. Como poderemos ver em sua subseção, existe uma infinidade de periféricos que utilizam o

barramento *USB*, o que implica a necessidade de sua habilitação para evitar maiores inconvenientes. Caso optemos por desabilitá-la, deveremos também desabilitar o barramento na configuração da *BIOS* do sistema.

Das principais opções existentes, merecem destaque o *USB MIDI support* para suporte à *joystick USB* e *USB Mass Storage support* para suporte às memórias eletrônicas (*flash memory*), disponíveis nos famosos *pendrives* e câmeras fotográficas digitais.

Bluetooth support

Bluetooth é uma recente tecnologia que tem como finalidade conectar diversos equipamentos e componentes eletrônicos sem a utilização das tradicionais fiações e cabos elétricos. A transmissão de dados se dá por ondas de rádio de curto alcance.

No *kernel* encontramos estas opções centralizadas em seu menu de opções, bastando apenas defini-las conforme necessário.

Cryptographic options

Opções gerais para a utilização de recursos de criptografia. Disponível à partir da versão *2.4.22* do *kernel*, este recurso é necessário para a utilização do *VPN* em redes internas, por exemplo. Como as instruções deste livro enfoca o sistema para o uso em *desktops*, não há grande importância nesta seção, ficando à critério dos usuários a edição das opções aqui descritas.

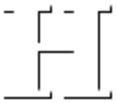
Kernel hacking / Library routines

Segue as instruções do próprio *Linus Torvalds* sobre esta categoria:

> *"Os detalhes de configuração utilizados no "kernel hacking" resultam em geral num kernel maior ou mais lento (ou ambos) e, podem tornar o kernel mais instável, pela criação de elementos de código "agressivo" e cujo objetivo é a detecção de erros do kernel (kmalloc()). Deveria pois, provavelmente, responder 'n' a todas as perguntas se quer criar um kernel de 'produção'." -> [Linus Torvalds, "O Linux Kernel HOWTO"].*

Sem maiores detalhes, as opções desta seção somente deverão ser habilitadas para os propósitos acima mencionados.

Da mesma forma que o *kernel hacking*, as opções disponíveis em *Library routines* são para o uso de programadores do *kernel*, das quais pouca influência benéfica farão ao sistema ao editarmos. Portando recomendamos mantê-las em seu estado original.



Salvando as alterações realizadas

Após concluir a seleção das opções desejadas na configuração do novo *kernel*, encontraremos no final da janela quatro opções para finalizar o utilitário e retornar à linha de comando para dar continuidade ao processo de compilação.

Salvando as alterações realizadas	
Save and Exit	Salvas as alterações realizadas.
Quit Without Saving	Sai do menu sem salvar as alterações.
Load Configuration from File	Carrega uma configuração já existente.
Store Configuration to File	Restaura as alterações realizadas de um arquivo previamente gravado.

Caso tenham utilizado os demais menus de configuração, as opções para salvar e carregar o arquivo de configuração gerado serão diferentes, mas os mesmos conceitos aqui descritos estarão presentes.

Ao salvarem as alterações realizadas, estas serão gravadas por padrão em um arquivo chamado *.config*, situado em */usr/src/linux*, que será a base para que as ferramentas de compilação criem o que será o novo *kernel* do sistema e seus respectivos módulos.

Conclusão

A grande maioria dos erros e complicações que ocorrem durante e após a execução deste processo encontra-se justamente na configuração de suportes e parâmetros do *kernel*. A omissão, a incorreta definição ou ainda a utilização de parâmetros que vão além do suportado pelo sistema trazem diversos tipos de inconvenientes, como mau funcionamento, instabilidade, travamentos, perdas de dados, entre diversas outras ocorrências indesejadas, além da impossibilidade da inicialização do sistema.

Ao realizarem a criação do novo *kernel*, optem por realizá-las com calma, tranqüilidade e consciência, as personalizações necessárias sobre as definições disponíveis, reservando também o tempo necessário para estas atividades. Teremos menos aborrecimentos com a perda de alguns bons minutos no desenvolvimento deste processo à ter que realizar a restauração do sistema, pois de acordo com as anomalias apresentadas, este último processo poderá ser bastante desgastante, não tendo ainda com isto pouco ou nenhum ganho efetivo. Conforme diz o velho ditado, "*a pressa é inimiga da perfeição...*". &;-D

V. A COMPILAÇÃO

INTRODUÇÃO

Após o término das operações de ajustes e configuração das opções disponíveis do *kernel*, chegamos finalmente à parte mais importante: a realização da compilação propriamente dita.

INICIANDO A COMPILAÇÃO

Para iniciar o processo de compilação, digitem na linha de comando...

```
# make dep
```

O comando *make dep* realizará a checagem das pendências necessárias e condicionará o código-fonte para a compilação do *kernel*.

Caso já tenha sido feito o processo de compilação, deveremos utilizar...

```
# make clean
```

... para que seja feito uma limpeza dos arquivos temporários e de instalação criados durante a compilação anterior. Apesar de opcional, é recomendada a sua utilização, pois eliminará quaisquer "*resíduos*" deixados por outras compilações, além de ser uma operação bastante rápida.

CONSTRUINDO A IMAGEM COMPACTADA DO KERNEL

Para realizarmos a compilação propriamente dita, deveremos utilizar...

```
# make bzImage
```

Este comando realizará a compilação do *kernel* baseando-se nas definições previamente realizadas durante o processo de configuração. Dependendo da capacidade de carga para o processamento, será necessário aguardarmos alguns preciosos minutos para o término da operação.

Podemos também utilizar os três últimos comandos citados neste capítulo em uma única linha para a construção da imagem compactada do *kernel*, se desejarmos. Neste caso necessitaremos digitar apenas...

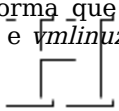
```
# make dep clean bzImage
```

OPÇÃO AUTOMATIZADA

Uma opção interessante disponível em diversas páginas da *Internet* para a compilação do *kernel*, é a utilização do comando *make bzlilo*.

```
# make bzlilo
```

Este procedimento, ao ser utilizado com o *LILO* corretamente configurado, compilará o *kernel* da mesma forma que *make bzImage*, porém logo em seguida os arquivos *System.map* e *vmlinuz* serão copiados para o diretório



/boot, e o *kernel* padrão anterior será renomeado para *vmlinuz.old*. Isso tudo será feito de forma automática, e assim bastará apenas recondicionar o arquivo de configuração *lilo.conf* para habilitar o *kernel* recentemente compilado como padrão.

Compilação e instalação dos módulos

Os módulos do *kernel* atualmente em uso se encontram no diretório */lib/modules/[VERSÃO]*. Porém, dependendo da escolha do *kernel*, poderá haver a necessidade de renomeá-lo de acordo com as necessidades.

Existem *2* pontos importantes para serem analisados:

- Se optarmos por suportar os módulos, além de marcar diversas outras opções para suporte como módulos;
- Se o *kernel* à ser compilado é uma nova versão ou apenas a recompilação da versão corrente do sistema.

Se optarmos por não suportar os módulos (o que é algo raro), não será necessário realizarmos a recompilação e reinstalação os módulos, bastando apenas não serem executados os comandos *make modules* e *make modules_install*. Caso contrário, se tivermos optado por compilar uma nova versão do *kernel*, não se preocupem com a permanência dos módulos antigos, pois será criado um novo diretório com o nomenclatura da nova versão compilada. Porém, caso tenham preferido compilar o mesmo *kernel* corrente do sistema...

Suporte aos módulos em kernels da mesma versão

Existem *2* opções para serem consideradas:

1. Renomeando o diretório que contém os módulos;
2. Editando o arquivo *Makefile*.

Caso optem por utilizar a *1a.* opção, deveremos utilizar a sintaxe...

```
# mv /lib/modules/[VERSÃO]/lib/modules/[NOVA_NOMENCLATURA]
```

Mantenham as definições da versão do diretório antigo, acrescentando apenas alguns caracteres para facilitar o entendimento em futuras consultas:

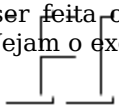
```
# mv /lib/modules/2.4.16 /lib/modules/2.4.16-original
```

Após isto, retornem ao diretório padrão do código-fonte do *kernel* para que possamos dar continuidade ao processo.

```
# cd /usr/src/linux/
```

A recomendação é que utilizem a *2a.* opção: editando o arquivo *Makefile*.

Este *Makefile* se encontra presente no diretório */usr/src/linux*, onde a modificação necessária deverá ser feita com a criação de um parâmetro especial na seção *Extraversion*. Vejam o exemplo abaixo:



```
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 20
EXTRAVERSION =

KERNELRELEASE=$(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION)
```

Bastará acrescentarmos o parâmetro na linha do *Extraversion* após o sinal =. Ficará então assim:

```
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 20
EXTRAVERSION = -[NOMENCLATURA_ESPECIAL]

KERNELRELEASE=$(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION)
```

A executar o comando make *modules_install*, será criado o diretório com a nova nomenclatura para armazenamento dos módulos – neste caso, o diretório */lib/modules/2.4.20-[NOMENCLATURA_ESPECIAL]*.

Novamente reforçando, utilizem a *2a.* opção, pois caso algo não dê certo, será necessário reiniciar a máquina com o *kernel* antigo. Se viéssemos à utilizar o método anterior, a inicialização não encontrará os antigos módulos do sistema, já que eles estariam em um diretório diferente (renomeado).

Iniciando a compilação

Para concluir a compilação do novo *Kernel*, basta agora executarmos os comandos necessários para iniciar a compilação dos módulos.

```
# make modules
# make modules_install
```

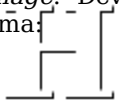
O comando *make modules* construirá os módulos necessários de acordo com as especificações feitas durante a configuração da compilação do *kernel*. Já o comando *make modules_install* instalará os módulos para o diretório */lib/modules/[NOVA_VERSÃO]-[EXTRA_VERSÃO]*.

O processo de compilação dos módulos também é um pouco demorado, dependendo da capacidade do processador da quantidade de memória disponibilizada; haverá necessidade de um pouco de paciência. Então aproveite novamente o momento e vão esticar um pouco as pernas e tomar um bom café! &;-D

Ajustes necessários

Copiando a imagem do kernel

Após a compilação do *kernel*, será criada uma imagem do *kernel* em */usr/src/linux/arch/i386/boot/bzImage*. Devemos copiá-la para o diretório raiz e renomeá-la da seguinte forma:



```
# cd /
# mv /usr/src/linux/arch/i386/boot/bzImage
# mv bzImage vmlinuz-[VERSÃO_ATUAL]
```

Em caso da necessidade de manter múltiplas versões compiladas, deveremos renomeá-lo para a data atual, para que possamos ter um melhor controle sobre as versões personalizadas do *kernel*. Observem que não poderemos utilizar espaço entre os caracteres para a atribuição do novo nome do arquivo.

Utilizando a linha de comando, ficará assim:

```
# mv bzImage vmlinuz-[VERSÃO_ATUAL]-[DATA_ATUAL]
```

Uma boa dica é utilizar o formato *[ANO-MÊS-DIA]* (p. ex. *2003-31-01*) como padrão, para um melhor controle. Muitos utilizam o formato *[MÊS-ANO] (jan-2003)* ou *[DIA-MÊS-ANO]* (*31-01-2003*), mas ao visualizá-los em qualquer gerenciador de arquivos, a tendência é que fique "*fora de ordem cronológica*", podendo enganar os usuários mais despercebidos.

Alterando a configuração do LILO

Edite o arquivo */etc/lilo.conf* e veja a seção "*Linux bootable partition config*":

```
# Linux bootable partition config begins
image = /boot/vmlinuz
  root = /dev/hda5
  label = Linux
  read-only
# Linux bootable partition config ends
```

Após o termo *read-only*, pulem uma linha e adicionem os seguintes códigos para disponibilizarmos uma nova opção de *kernel* para o sistema. No exemplo abaixo, ficaria da seguinte forma:

```
image=/boot/vmlinuz-[DATA_ATUAL]
  root=/dev/hda5
  label=[NOME_DO_KERNEL_PERSONALIZADO]
  read-only
```

O novo escopo do arquivo de configuração do *LILO* deverá ser editado para que fique desta forma:

```
# Linux bootable partition config begins
image = /boot/vmlinuz
  root = /dev/hda5
  label = Linux
  read-only

image=/boot/vmlinuz-[DATA_ATUAL]
  root=/dev/hda5
  label=[NOME_DO_KERNEL_PERSONALIZADO]
  read-only
# Linux bootable partition config ends
```

É importante notar que a nomenclatura adotada para o novo *kernel* (*label = [NOME_DO_KERNEL_PERSONALIZADO]*) não poderá ter espaços.

Se preferirem, o arquivo *boot_messages.txt* poderá ser alterado para que este exiba a nova opção de *kernel* disponibilizada, apesar que as opções estarão presentes no menu principal do utilitário.

```
Welcome to the LILO Boot Loader!

Please enter the name of the partition you would like to boot
at the prompt below.  The choices are:

Linux   - Linux (Linux native partition)
[NOVO]  - [DADOS_SOBRE_O_NOVO_KERNEL_COMPILADO]
```

Basta apenas rodar o *LILO* para que sejam aceitas as alterações efetuadas:

```
# lilo

Added

Linux *
[NOME_DO_KERNEL_PERSONALIZADO]
```

Observem que, ao utilizarmos múltiplas opções de inicialização para *kernels* de mesma versão, estes utilizarão o mesmo diretório */lib/modules/[VERSÃO]*, o que poderá ocasionar problemas tais como incompatibilidade ou falta de módulos. Para isto, consultem o capítulo *Circunstâncias especiais -> Mantendo múltiplas versões personalizadas do kernel* para conhecerem a aplicação de melhores técnicas e procedimentos.

Conclusão

A realização do processo de compilação, apesar da necessidade de ter atenção à alguns aspectos e outras intervenções necessárias, é um processo relativamente simples e fácil de gerenciar. O maior incômodo que poderemos encontrar em sua realização será a possível demora do andamento do processo: a atividade poderá demandar um certo tempo e, dependendo da carga de processamento do computador, exigirá bastante da *CPU*, da memória, além de um bom espaço disponível no disco rígido.

Estejam atentos nas ocorrências e informações exibidas na tela de vídeo, pois a incidência de erros e falhas durante a operação. Tendo em mão estes dados, poderemos analisar uma depuração mais eficiente da ocorrência encontrada e assim corrigir e/ou encontrar a solução desejada. Tendo todas as opções de configuração ajustadas corretamente (no capítulo *Configuração de suporte e parâmetros*), o desenrolar do processo seguirá sem maiores problemas. &;-D

VI. Manipulação de módulos e pendências

Introdução

Os módulos do *kernel* são componentes que auxiliam a interação do *kernel* com os dispositivos do sistema. À grosso modo, poderemos dizer que eles nos sistemas *GNU/Linux* possuem funções eqüivalentes aos *drivers* do tradicional *Windows*. Neste capítulo iremos conhecer as principais ferramentas para a sua manipulação, bem como seus recursos e métodos de utilização para o bom aprendizado e obter melhor aproveitamento nas atividades de administração e manutenção do *kernel* e respectivos módulos.

As ferramentas...

LSMOD

A função do comando *lsmod* é listar os módulos que se encontram carregados no sistema. Para uma utilização simples e rápida basta digitar...

```
# lsmod
Module                  Size  Used by    Not tainted
lp                      6752   0  (unused)
apm                     9608   0  (unused)
usb-storage            65536   0  (unused)
radeon                 96932   0
parport_pc             14724   1
parport                23264   1  [lp parport_pc]
uhci                   24560   0  (unused)
usbcore                58144   1  [uhci]
emu10k1                61288   0
ac97_codec              9512   0  [emu10k1]
usbcore                58400   1  [usb-storage uhci ehci-hcd]
soundcore               3332   4  [emu10k1]
emu10k1-gp              1352   0  (unused)
gameport                1452   0  [emu10k1-gp]
ide-scsi                8048   0
# _
```

Por possuir uma sintaxe básica, sua saída também é bastante simples, onde será apresentada uma listagem de todos os módulos carregados no sistema.

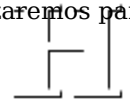
MODPROBE

O responsável pelo carregamento dos módulos disponíveis no sistema.

Sintaxe:

```
# modprobe [MÓDULO]
```

Para este comando existem parâmetros e funcionalidades extras, porém na maioria das vezes apenas o utilizaremos para o carregamento de módulos.



Por exemplo, para carregar os módulos *USB* para habilitar um *mouse USB*...

```
# modprobe usbmouse
```

INSMOD

Da mesma forma que *modprobe*, carrega os módulos disponíveis do sistema.

Sintaxe:

```
# modprobe [MÓDULO]
```

Porém, sua diferença está no fato de que o *modprobe* carrega as pendências necessárias para o funcionamento do módulo, ao passo que o *insmod* tão somente carrega o módulo. Veja um exemplo prático (apenas para demonstração):

```
# insmod usb-storage
Using /lib/modules/2.4.22/kernel/drivers/usb/storage/usb-storage.o.gz
# _
```

Para certificarmos de que foi corretamente carregado...

```
# lsmod | grep usb
usb-storage             65536   0  (unused)
usbcore                 58400   1  [usb-storage uhci ehci-hcd]
# _
```

Em algumas circunstâncias poderá ser necessário um carregamento forçado de um determinado módulo. Para isto utilize...

```
# insmod -f [MÓDULO]
```

Isto ocorre muito quando há necessidade de carregar um módulo compilado de uma versão diferente do *kernel*. Apesar desta operação não ser recomendável, poderá ser a única saída para diversas situações &;-(

DEPMOD

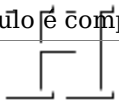
O comando *depmod* é utilizado para checar as pendências dos módulos, onde estas são verificadas em */etc/init.d*.

Sintaxe:

```
# depmod [PARÂMETROS]
```

Onde:

depmod	
-a	Checa as pendências de módulo necessárias pelo *kernel*.
-b	Define o caminho de um módulo específico que não esteja armazenado no diretório padrão (*/lib/modules/[KERNEL]/*).
-e	Verifica se determinado módulo é compatível com o *kernel* em uso.



Para realizarmos uma simples checagem, deveremos utilizar...

```
# depmod -a
```

Os resultados deste comando são armazenados em um arquivo-texto chamado *modules.dep*, situado em seu diretório padrão.

MODINFO

Exibe informações básicas sobre determinados módulos.

Sintaxe:

```
# modinfo [PARÂMETROS] [MÓDULO]
```

Onde:

modinfo	
-a	Exibe o autor.
-d	Exibe um breve resumo.
-l	Exibe a licença.
-p	Exibe parâmetros específicos.

Exemplo:

```
# modinfo -d /lib/modules/2.4.22/kernel/drivers/hotplug/pci_hotplug.o.gz
"PCI Hot Plug PCI Core"
# _
```

Experimentem os demais parâmetros e vejam os resultados.

RMMOD

Sem grandes mistérios, o comando *rmmod* serve unicamente para descarregar um determinado módulo do sistema.

Sintaxe:

```
# rmmod [MÓDULO]
```

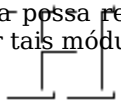
Lembrem-se de que estaremos apenas DESCARREGANDO o módulo da memória, e não EXCLUINDO-O. Poderemos então ficar sossegados... &;-D

Segue um simples exemplo para descarregar o módulo responsável pelo acesso às memórias *flash* (*Pendrive*):

```
# rmmod usb-storage
```

MANIPULAÇÃO MANUAL

Em muitos casos são disponibilizados módulos pré-compilados de determinados dispositivos, onde os mesmos deverão ser ativados manualmente para que o sistema possa reconhecê-lo. Exemplos típicos de periféricos que necessitam de ter tais módulos são os *softmodens*.



Para realizarmos esta operação, deveremos inicialmente obter o módulo (lógico), onde este deverá ter uma extensão *.o* ou *.o.gz* no final de sua nomenclatura. Em seguida, deveremos copiá-lo para o diretório-padrão do sistema. Neste caso, o endereço-destino será...

```
# cp [MÓDULO] /lib/modules/[VERSÃO_DO_KERNEL]/kernel
```

Em seguida, para ativarmos deveremos evocar o comando *modprobe*...

```
# modprobe /lib/modules/[VERSÃO_DO_KERNEL]/kernel/[MÓDULO]
```

Por último, bastará apenas atualizar a listagem de módulos do sistema. Utilizem na linha de comando...

```
# depmod -a
```

Incluam o comando *modprobe* e sua sintaxe no arquivo */etc/rc.modules* para que este módulo esteja sempre disponível a partir da inicialização.

Arquivos de configuração

Localização padrão dos módulos

Os módulos encontram-se localizados por padrão no diretório...

```
$ ls -l /lib/modules/[VERSÃO]/kernel/
total 5
drwxr-xr-x    3 root     root           72 2003-08-28 17:08 arch/
drwxr-xr-x    2 root     root          432 2003-09-02 23:03 crypto/
drwxr-xr-x   24 root     root          592 2003-08-28 17:08 drivers/
drwxr-xr-x   23 root     root          656 2003-09-02 23:03 fs/
drwxr-xr-x    3 root     root          112 2003-09-02 23:03 lib/
drwxr-xr-x   21 root     root          528 2003-08-28 17:08 net/
drwxr-xr-x   11 root     root          264 2003-08-29 04:39 sound/
$ _
```

Cada subdiretório armazena cada um de acordo com sua categoria.

Sobre o arquivo /etc/rc.d/rc.modules

Todos os módulos, para serem carregados diretamente na inicialização do sistema, deverão estar especificados no arquivo */etc/rc.d/rc.modules*, onde normalmente basta apenas descomentar os comandos abaixo descritos para carregá-los.[4] Para obterem maiores informações, consulte a *2a. Parte: Conhecimentos Gerais -> Inicialização*.

Conclusão

Os módulos do *kernel* estão entre as maiores fontes de insucesso na

4 Para facilitar a nossa vida, encontra-se incluso no *kernel* uma grande quantidade de módulos pré-compilados para o suporte à diversos periféricos, bastando apenas descomentar as linhas referentes. Por sua vez, estas linhas estão subdivididas por várias seções, os quais facilitarão muito a nossa procura.

utilização do novo *kernel* personalizado. São vários os possíveis erros de ajustes e configurações que trazem diversos e inconvenientes transtornos. Diversos dispositivos e periféricos passam à ser suportados de forma incorreta e instáveis ou até mesmo, na maioria dos casos, deixam de ser suportados, ocasionando também o travamento geral do sistema. Este é um dos principais motivos pelo qual merecem atenção especial, onde o conhecimento de ferramentas e procedimentos são fundamentais para o sucesso das operações afins realizadas. &;-D

VII. Circunstâncias especiais

Introdução

Enfim, após realizarmos todos os procedimentos necessários para a compilação do *kernel*, enganam-se àqueles que consideram que não há mais nada à fazer. Para provar isto, este capítulo propõe à estudar os necessários e possíveis ajustes para que seja garantido o perfeito funcionamento do novo *kernel* customizado.

Kernel

Mantendo múltiplas versões personalizadas do kernel

Uma das recomendações mais importantes para a compilação o *kernel* é trabalhar com múltiplas versões para um melhor controle das operações realizadas, pois quando utilizamos uma nova versão e/ou correção do *kernel*, os módulos compilados serão armazenados em um novo diretório.

```
# ls -l /lib/modules/
total 2
drwxr-xr-x    4 root     root          416 Mar 13 15:36 2.4.18/
drwxr-xr-x    4 root     root          416 Set 17 17:20 2.4.20/
# _
```

Mas se viermos à utilizar a mesma versão do *kernel* corrente? Neste caso, os módulos compilados irão sobrescrever os antigos já existentes pelo fato de utilizarem a mesma nomenclatura da versão.

```
# ls -l /lib/modules/
total 1
drwxr-xr-x    4 root     root          416 Set 17 17:20 2.4.20/
# _
```

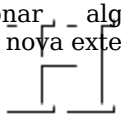
Para evitarmos este tipo de inconveniente, deveremos editar o arquivo */usr/src/linux/Makefiles* gerado após a configuração das opções do *kernel*.

As primeiras linhas deste arquivo fornecem as especificações para a criação de uma nova nomenclatura para o novo *kernel* à ser compilado. Deveremos redefinir uma nova nomenclatura:

```
VERSION = X
PATCHLEVEL = X
SUBLEVEL = XX
EXTRAVERSION =

KERNELRELEASE=$(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION)
```

Levem em conta que nem sempre é bom alterar os valores padrões de versão, pois poderá haver a necessidade do conhecimento desta em futuras ocasiões. Prefiram condicionar alguns caracteres na opção *EXTRAVERSION* para gerar uma nova extensão para a versão já existente.



```
EXTRAVERSION = Darkstar
```

Também poderemos alterar o formato desta nova nomenclatura conforme melhor nos vier. Observem que colocamos um ponto para separar as informações de *SUBLEVEL* para *EXTRAVERSION*, evitando assim que a correção fique "*colada*" com a nomenclatura extra.

```
KERNELRELEASE=$(VERSION).$(PATCHLEVEL).$(SUBLEVEL).$(EXTRAVERSION)
```

Agora deveremos dar continuidade ao processo de compilação e, quando terminarmos de compilar e instalar os módulos, estes estarão armazenados em seu respectivo diretório onde foram definidas a nova nomenclatura.

```
# l /lib/modules/
total 1
drwxr-xr-x    4 root     root          416 Set 17 17:20 2.4.20/
drwxr-xr-x    4 root     root          416 Set 21 18:42 2.4.20.Darkstar/
# _
```

Editem o arquivo */etc/lilo.conf* e reorganizem as imagens de inicialização, porém sempre procurem manter a imagem original (obtida durante a instalação do sistema) e uma entrada no menu para a imagem recém criada.

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=Atual

image=/boot/vmlinuz-2.4.20
        label=Original
        read-only
        root=/dev/hda5

image=/boot/vmlinuz-2.4.20.Darkstar
        label=Darkstar
        read-only
        root=/dev/hda5

image=/boot/bzImage-2.6.0
        label=kernel_2.6.0
        root=/dev/hda5
        read-only

image=/usr/src/linux/arch/i386/boot/bzImage
        label=Experimental
        root=/dev/hda5
        read-only
```

Neste caso batizamos a nova imagem de *Experimental*, onde apenas apontamos para a localização onde esta foi criada – neste caso no diretório *arch/i386/boot/bzImage* do código-fonte do *kernel*. Caso o procedimento tenha dado certo, copiem-na para o diretório padrão – */boot* – e renomeiem-na conforme os termos utilizados para definir a nova nomenclatura.

Observem que utilizamos como exemplo tanto as versões personalizadas do

mesmo *kernel* do sistema – *vmlinuz-2.4.20.Darkstar* – quanto uma nova versão recentemente liberada – *bzImage-2.6.0*.

```
image=/boot/vmlinuz-2.4.20.Darkstar
image=/boot/bzImage-2.6.0
```

Não existe problema ao utilizar os termos *vmlinuz* e *bzImage*. Fica à critério. Realizamos estas modificações apenas para melhor exemplificar.

Após o término da edição do arquivo de configuração, executem novamente o *LILO* para sejam salvas as alterações no setor *MBR*, para que estejam disponíveis na próxima inicialização.

```
# lilo
Added Original *
Added Darkstar
Added Kernel_2.6.0
Added Experimental
```

Aplicando correções ao código-fonte do kernel

Devido ao alto grau de evolução do *kernel*, os erros encontrados são rapidamente corrigidos, onde em pouco tempo se encontram disponíveis novas versões destes programas prontos para baixar. Mas o que fazer com pacotes extremamente grandes, como é o caso do *kernel*?

Ao invés de obtermos a mesma versão do *kernel* com todas as correções necessárias, poderemos baixar apenas um pacote específico com estas correções, conhecidos popularmente como *patches*[5]. Estas correções atualizam o código-fonte do *kernel* corrente, deixando-o em dia como a última versão (correção). Baixem a correção do *kernel* desejado, porém procure, realizar antes uma cópia de segurança do *kernel* original.

Os comandos diff e patch

Os comandos *diff* e *patch* são utilizados respectivamente para comparar arquivos e modificar as diferenças encontradas entre eles. Este último utiliza o *diff* para realizar as devidas mudanças no arquivo original, quando encontradas as diferenças desejadas.

A aplicação do comando *patch* é bem simples, onde sua *sintaxe* básica é:

```
# patch [PARÂMETROS] [ARQUIVO_ORIGINAL] [CORREÇÃO]
```

Existem diversos parâmetros, porém demonstraremos logo abaixo somente aqueles que serão necessários para o nosso aprendizado.

Conforme recomendado anteriormente, deveremos realizar uma cópia de segurança do código-fonte do *kernel*. Para as distribuições que possuem o pacote disponível nos *CDs* de instalação, esta tarefa faz-se desnecessária.

```
# cd /usr/src/linux-[VERSÃO]
# make clean
-//-
```

5 Conhecidos também como *remendo*.

```
# tar -cvf kernel-[VERSÃO].tar *
-//-
# gzip kernel-[VERSÃO].tar
# mv kernel-[VERSÃO].tar.gz /usr/src/
```

Copiem o pacote que contenha o *patch* dentro do diretório */usr/src/*. Após isto, segue a *sintaxe* do comando, necessários para atualizar o antigo *kernel*:

```
# patch -p1 -d [PATCH_DO_KERNEL]
```

Em poucos instantes o comando *patch* irá atualizar o pacote desejado, utilizando a correção fornecida. Lembrem-se de que os pacotes deverão estar dentro do diretório principal onde se encontram os fontes do *kernel*.

```
# cd /usr/src/linux-[VERSÃO]
```

Caso exista no sistema uma versão muito antiga do *kernel*, deveremos aplicar os *patches* de acordo com a seqüência das versões disponíveis. Por exemplo, para atualizar um *kernel* da versão *2.4.20* para *2.4.26*, deveremos atualizar de *2.4.20* -> *2.4.21*; em seguida de *2.4.21* -> *2.4.22*; *2.4.22* -> *2.4.23*... e assim por diante até chegar à versão desejada. Dêem preferências em utilizar o código-fonte da versão mais nova do *kernel* atual ao invés de aplicar um *patch*, pois nestas circunstâncias a ocorrência de erros tenderá a ser maior.

Para remover uma aplicação do *patch*, deveremos utilizar...

```
# patch -R [PATCH_APLICADO]
```

Compilação

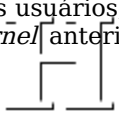
Compilando o código-fonte em outros processadores

Em virtude da lentidão da utilização deste processo em computadores obsoletos, existe a possibilidade da compilação do *kernel* em máquinas mais possantes, bastando apenas certificar-se que que as opções que serão configuradas correspondam à máquina o qual se pretende utilizar o *kernel* compilado. Para isto, após a realização do processo, copiem a imagem do *kernel* compilado obtido de */usr/src/linux/arch/i386/boot/* e seus módulos referentes de */lib/modules/[VERSÃO]* e transfiram-os para a máquina final, bastando copiá-los para os mesmos diretórios e ajustando os arquivos */etc/lilo.conf* com os novos parâmetros de inicialização.

Configuração

Restaurando as definições anteriores

Infelizmente existirão diversos casos onde o processo de compilação poderá não atenderá às necessidades dos usuários, e para piorar, será estritamente necessário a reutilização do *kernel* anterior para disponibilizar o sistema



para uso. Inicialmente recomendamos a manutenção das duas versões do *kernel*, a original e a customizada, pois caso ocorra quaisquer problema que impossibilite utilizar o sistema ou funcionalidade dele, poderemos reinciar o sistema normalmente utilizando a versão original do *kernel*, bastando remover as entradas do novo *kernel* compilado e realizando novamente o processo de compilação. Executem novamente o menu de configuração do sistema que contenha a opção para recarregar as definições anteriores do *kernel* utilizado. Esta opção somente se encontra disponível nos menus habilitados pelo comando *make xconfig / gconfig* (*Load configuration from file*) e *make menuconfig* (*Load an alternative configuration file*). Indiquem corretamente o arquivo de configuração.

Conclusão

Além da aplicação dos métodos tradicionais de compilação do código-fonte, existirão muitas circunstâncias específicas das quais a grande maioria dos tutoriais existem não apresentam, ou apresentam informações incompletas. Nesta seção apenas abordamos as principais necessidades extraordinárias referentes à compilação do *kernel*. Caso existam outras circunstâncias de ocorrência não rara, seremos gratos aos leitores em nos notificar para que possamos disponibilizar nas próximas versões deste guia as instruções necessárias para estes eventos. &;-D

VIII. Problemas mais freqüentes

Introdução

Enfim, chegamos à mais uma (possível) etapa (embora indesejada) do processo de compilação do *kernel*. Trata-se do entendimento e solução dos problemas e questões mais freqüentes referentes ao processo.

Infelizmente, em conseqüência de uma realização incorreta das medidas e intervenções necessárias para a customização do novo *kernel*, poderemos ter diversos inconvenientes que venham à comprometer o uso do sistema operacional. Nesta seção descreveremos as principais anomalias existentes e algumas dicas para a solucioná-las.

Resolvendo problemas...

Antes...

Parâmetros errados sendo definidos na configuração

Em alguma circunstância poderemos ter erros de configuração das opções do arquivo *.config* do *kernel*. Serão exibidas mensagens do tipo...

```
config: line 128: syntax error
```

Felizmente serão apontadas as linhas das quais ocorrem o erro, onde uma simples intervenção corretiva com um editor de textos eliminará o problema.

Erros durante a verificação de pendências com o comando make dep

Para evitar que este erro ocorra, deveremos nos certificar de todas as pendências necessárias para a compilação do *kernel* foram atendidas. Para isto, consultem nesta parte o capítulo *Preparativos iniciais*.

Durante...

Falha na execução dos comandos make bzImage e make modules

Em algumas circunstâncias teremos talvez a possibilidade da falha de compilação do *kernel* ao utilizarmos estes comandos. Isto geralmente ocorre quando o *kernel* é recompilado diversas vezes e o comando *make clean* não realiza uma limpeza eficiente. Caso isto ocorra, repitam os passos realizados para a checagem das pendências (*make dep*) e limpeza dos arquivos temporários (*make clean*); ao final deles utilizem o comando...

```
make mpromper
```

Para se certificarem de que realmente será feita uma eficiente limpeza, onde deveremos lembrar de salvaguardar uma cópia de segurança do arquivo de configuração – o *.config*.

```
# make dep
# make clean
# cp .config /root
# make mpromper
# make bzImage
# make modules
# make modules_install
```

Após isto, deveremos dar continuidade ao processo normalmente.

Depois...

O sistema não inicializa

Isto poderá ocorrer por variados motivos; o mais freqüente está justamente na definição de parâmetros do gerenciador de inicialização – *LILO*. Consultem a configuração do *LILO* e verifique na seção de partições os parâmetros existentes.

```
# Linux bootable partition config begins
image = /boot/vmlinuz
  root = /dev/hda6
  label = Linux
  read-only
# Linux bootable partition config ends
```

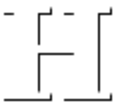
Para estas circunstâncias é recomendada a manutenção do *kernel* original, além do *kernel* personalizado. Consultem o capítulo *Circunstâncias especiais* para obteremo maiores informações para proceder.

O sistema apresenta a mensagens de erro "Unresolved symbols"

Isto ocorre porque algum módulo do *kernel* não foi compilado corretamente. Reiniciem a compilação dos módulos do *kernel* novamente, rodando o comando *tail -f nohup.out* +<ENTER> para a monitoração do processo:

```
# cd /usr/src/linux
# make dep
# make clean
# nohup make bzImage &
# tail -f nohup.out
# make modules
# make modules_install
```

Caso estes problemas ocorram novamente, consultem o arquivo *nohup.out* gerado para verificar quais foram as anomalias ocorridas.



Kernel "enorme"

Nestas situações geralmente o *kernel* foi compilado com suporte à uma série de *drivers* e parâmetros desnecessários. Ao realizarem a configuração do *kernel*, procurem habilitar SOMENTE os recursos de dispositivos suportados (ou que terão necessidade de suportar) pela máquina em uso. Além disso apenas definam as opções como *Y* (*Yes* – embutido) apenas aquelas essenciais e de uso constante do sistema. As demais deverão ficar como módulos. Consultem nesta parte o capítulo *Configuração de suporte e parâmetros* para maiores detalhes.

Comportamento anormal do kernel

Entende-se como comportamento anormal travamentos aleatórios, erros de entrada e saída, performance alterada, entre outras anomalias. Isto pode ocorrer quando o *kernel* é compilado mais de uma vez, sendo que na última não foi utilizado o comando *make clean* para realizar uma limpeza geral nas definições anteriores. Por este motivo recomendamos novamente rodar...

```
# make clean
```

... toda vez que realizar uma nova recompilação do *kernel*.

Algumas aplicações não funcionam corretamente

Isto é bastante comum quando da substituição de uma versão antiga do *kernel* (*2.2.x* -> *2.4.x* ou *2.4.x* -> *2.6.x*). Neste caso recomenda-se consultar a documentação do *kernel* e verificar quais aplicativos e bibliotecas deverão ser atualizados. Geralmente é solicitado uma versão *X* ou superior.

Recomendações gerais

Segue algumas recomendações gerais para evitarmos algumas ocorrências indesejadas ao processo de compilação do *kernel*, além de obtermos os melhores resultados possíveis.

- Mantenham sempre a versão original do *kernel* instalado no sistema. Caso ocorra algum inconveniente, poderemos reutilizá-lo para retornar com as configurações anteriores.
- Utilizem sempre a versão estável e atualizada do *kernel*. Em casos especiais optem por utilizar a versão de testes somente se esta atender à propósitos específicos.
- Evitem baixar diversos *patches* para atualizar *kernels* muito antigos. Além da possível demora, poderá ser um processo não muito seguro. Será mais seguro baixar um novo *kernel*.

Outro questão bastante importante é a criação de disquetes de inicialização para a solução de eventuais problemas. Haverá situações em que a inicialização do sistema estará impossibilitada, seja por algum erro de

configuração do *LILO* ou por qualquer outro.

Conclusão

Dentre as possíveis causas para diversas anomalias, a errônea definição de parâmetros durante o processo de configuração constitui a maior parte das ocorrência de erros durante o processo de compilação do *kernel*. Esse é um dos principais motivos pelo fato de recomendarmos – no capítulo anterior – a realização dessas definições de configuração de forma tranqüila.

Caso ocorra algum erro que impeça do sistema funcionar ou acarrete o seu mau funcionamento, inicialmente recomendamos checar o sistema para descobrir o motivos que o causaram. Será uma ótima oportunidade de aprendizado desvendar as origens do problemas. Não tendo nada mais à fazer, uma boa pesquisa pela *Internet* ajudará. Caso ainda não consigam resolver a questão, redijam uma mensagem para uma lista de discussão especializada no assunto, contando os detalhes pormenores destas ocorrências. Lá com certeza encontrarão apoio de outros usuários que conhecem melhor este problema, onde os mesmos irão propor soluções para este inconveniente. &;-D

ENCERRAMENTO

- ✔ <http://br.kernelnewbies.org/>.

Conforme podemos ver, foi dada uma atenção minuciosa para que este capítulo fosse desenvolvido com o objetivo de prover todas as informações possíveis e necessárias para a realização deste procedimento com eficiência, segurança e riqueza de detalhes. Se por algum acaso existirem informações específicas e/ou especiais que não se encontrem aqui inclusas, favor contacte-nos para que possamos atualizá-las.

A realização de buscas pela *Internet* é interessante e estimula o espírito de investigação dos usuários, facilitando a assimilação de diversas informações. Ao nosso modo de ver, esta é a opção mais recomendada. Este livro foi escrito basicamente utilizando estes recursos: mescla de conhecimentos adquiridos com consultas diversas à endereços eletrônicos especializados em sistemas *GNU/Linux*. Mas para dar um pontapé inicial, recomendamos uma visita ao endereço eletrônico abaixo relacionado. Lá encontraremos dicas e informações diversas que nos serão bastante úteis.

O *kernel* dos sistemas *GNU/Linux* continua sendo ativamente desenvolvido pela sua equipe, com o apoio de milhares de usuários entusiastas e corporações. Graças à isto, poderemos contar mais adiante com melhorias contínuas que por sua vez nos trarão benefícios enormes de performance e estabilidade do sistema. Se no presente o *kernel* atual nos encanta com suas características e qualidades únicas, mal podemos esperar o que ele nos reserva no futuro. &;-D