

LINUX
MAGAZINE

COLEÇÃO
Linux Pro

Este livro é
recomendado por



Certificação LPI-2

201 - 202

Luciano Antonio Siqueira



Curso completo para LPIC-2

2ª edição revisada e ampliada.

Exercícios em todos os tópicos.

Livro preparado para a nova
prova válida a partir de 2009.

Luciano Antonio Siqueira

Certificação LPI-2



2ª edição

São Paulo
2009

Certificação LPI-2

por Luciano Antonio Siqueira

Direitos autorais e marcas registradas © 2004 - 2009:

Linux New Media do Brasil Editora Ltda.

Nenhum material pode ser reproduzido em qualquer meio, em parte ou no todo, sem permissão expressa da editora. Assume-se que qualquer correspondência recebida, tal como cartas, emails, faxes, fotografias, artigos e desenhos, é fornecida para publicação ou licenciamento a terceiros de forma mundial não-exclusiva pela Linux New Media do Brasil, a menos que explicitamente indicado. Linux é uma marca registrada de Linus Torvalds.

Revisão: Aileen Nakamura

Projeto gráfico e diagramação: Paola Viveiros

Capa: Paola Viveiros

Contatos

Autor: lsiqueira@linuxnewmedia.com.br

Comercial: vendas@linuxnewmedia.com.br

Redação: redacao@linuxnewmedia.com.br

Linux New Media do Brasil Editora Ltda.

Av. Fagundes Filho, 134

Conj. 53 – Saúde

CEP: 04304-000

São Paulo – SP – Brasil

Tel: +55 (0)11 4082-1300


Fax: +55 (0)11 4082-1302

Siqueira, Luciano Antonio

Certificação LPI-2 / Luciano Antonio Siqueira. - São Paulo: Linux New Media do Brasil Editora Ltda, 2009.

ISBN: 978-85-61024-14-7

1. Linux 2. Informática 3. LPI 4. Certificação 5. Redes 6. Computação



Suba e me veja às vezes
Então tentarei te alcançar
Bem do fundo do meu mar
Pois você vê eu aprendo

(A.B.)



Prefácios	7
Introdução	11
Visão geral das mudanças nos exames LPIC nível 2	11
Tópico 201: O kernel Linux	15
201.1 Componentes do kernel	16
201.2 Compilando um kernel	19
201.3 Aplicar um patch ao kernel	25
201.4 Configurar, compilar e instalar um kernel personalizado e seus módulos	27
201.5 Controlar/investigar o kernel e seus módulos durante sua execução	28
Tópico 202: Início do sistema	41
202.1 Personalizar o início do sistema e o processo de boot	42
202.2 Recuperação do sistema	46
Tópico 203: Sistema de arquivos e dispositivos	55
203.1 Trabalhando com o sistema de arquivos Linux	56
203.2 Manutenção de um sistema de arquivo Linux	63
203.3 Criando e configurando opções para sistemas de arquivos	69
203.4 Gerenciamento de dispositivos udev	74
Tópico 204: Hardware	83
204.1 Configuração de RAID	84
204.2 Ajustar o acesso a dispositivos de armazenamento	87
204.3 Gerenciamento de volumes lógicos (LVM)	91
Tópico 205: Configuração de rede	99
205.1 Configurações básicas de rede	100
205.2 Configuração avançada de rede e resolução de problemas	106
205.3 Soluções de problemas de rede	120
205.4 Informar usuários sobre questões relativas ao sistema	122
Tópico 206: Manutenção do sistema	129
206.1 Compilar e instalar programas a partir do código fonte	130
206.2 Operações de backup	133

Tópico 207: Domain Name Server	143
207.1 Configuração básica de um servidor DNS	144
207.2 Criação e manutenção de zonas de DNS	146
207.3 Segurança de DNS	155
Tópico 208: Web Services	163
208.1 Implementando um servidor Web	164
208.2 Manutenção de servidor Web	171
208.3 Implementando um servidor Proxy	175
Tópico 209: Compartilhamento de arquivos	181
209.1 Configurar um servidor Samba	182
209.2 Configurar um servidor NFS	188
Tópico 210: Administração dos clientes de rede	197
210.1 Configuração DHCP	198
210.2 Autenticação por PAM	201
210.3 Uso de cliente LDAP	205
Tópico 211: Serviços de email	217
211.1 Utilização de servidores de email	218
211.2 Administração da entrega local de email	223
211.3 Administrar entrega de email remoto	225
Tópico 212: Segurança do Sistema	233
212.1 Configuração de roteador	234
212.2 Segurança de servidores FTP	242
212.3 Shell seguro (SSH)	243
212.4 TCP Wrappers	248
212.5 Tarefas de segurança	251
Tópico 213: Solução de problemas	257
213.1 Identificar estágios de boot e consertar carregadores de boot	258
213.2 Solução geral de problemas	263
213.3 Problemas em recursos do sistema	265
213.4 Problemas em configurações de ambiente	267
Apêndice	273
Respostas	301



Este livro destina-se a candidatos em busca da certificação LPI nível 2 (LPIC-2) e atende tanto às necessidades de profissionais que já trabalham com outros sistemas operacionais como as de profissionais de Linux em geral. Este projeto da Linux New Media, tão bem conduzida por Rafael Peregrino e pelo Claudio Bazzoli, foi desenvolvido pelo Luciano Siqueira, um profissional raro, capaz de aliar conhecimentos técnicos profundos com uma impressionante capacidade de comunicação. Como resultado, temos esta obra completa, abrangente e, ao mesmo tempo, que dá todas as condições para que um candidato se prepare para as provas de certificação LPIC-2.

O LPI – Linux Professional Institute Linux (www.lpi.org) promove e certifica habilidades essenciais em Linux e em tecnologias de Open Source por meio de provas abrangentes, de alta qualidade e independentes de distribuições Linux. O LPI foi criado em 1999 pela comunidade Linux como uma organização internacional sem fins lucrativos, com o objetivo ser reconhecido como o líder global na certificação de profissionais de Linux, promovendo o Linux e o movimento de Open Source.

O programa de certificação profissional LPI é composto de três níveis de certificação (LPIC-1, LPIC-2 e LPIC-3), sendo que a certificação LPIC-1 tem como alvo profissionais junior e plenos, ao passo que a certificação LPIC-2 é orientada a profissionais mais experientes e líderes de equipes. Para que um candidato seja certificado no nível 2 já tenha obtido a certificação no nível 1.

No mundo de tecnologia, cada vez mais a certificação profissional é necessário que ele tem se tornado, uma vez é um indicativo claro e objetivo do conhecimento de uma pessoa a respeito de um determinado assunto, no nosso caso, o Linux. Obviamente, na hora de uma contratação, por exemplo, outros fatores também contam, mas o mais difícil de se avaliar é o conhecimento, já que as características pessoais e a experiência podem ser facilmente avaliadas com entrevistas, testes, referências, etc.

Assim, pode-se dizer que a certificação profissional acaba sendo uma ferramenta essencial tanto para quem contrata como para quem é contratado, garantindo que os candidatos tenham as habilidades necessárias e, conseqüentemente, sejam capazes de executar o que se espera deles. Dessa forma, garante-se um padrão de qualidade e facilita tanto a contratação como futuras promoções.

José Carlos Gouveia

José Carlos Gouveia é Diretor Geral do Linux Professional Institute – LPI – da América Latina. Anteriormente, trabalhou por cinco anos para a SGI – Silicon Graphics – como Diretor Geral da América Latina, além de ter sido diretor geral da Novell, Platinum Technology, PeopleSoft e JDEdwards, tendo sido também diretor da Anderson Consulting (Accenture) e da Dun&Bradstreet Software e gerente da EDS. Gouveia é formado em Ciência da Computação pela Unicamp, com pós-graduação pela Unicamp e pela PUC-RJ.



A certificação do Linux Professional Institute (LPI) é uma ótima maneira de demonstrar a empregadores e a potenciais clientes que você possui as habilidades e os conhecimentos necessários para trabalhar num ambiente Linux e de Código Aberto, além de ser uma excelente ferramenta para seu desenvolvimento profissional. O programa de certificação LPI é globalmente reconhecido por profissionais de TI, empresas e governos como a certificação Linux número um em todo o mundo. Esse programa vem continuamente sendo reconhecido por fabricantes como pré-requisito para seus próprios programas específicos de certificação em alto nível. São exemplos os programas de certificação do Ubuntu pela Canonical, HP, IBM, Novell e Oracle.

O Linux Professional Institute completa 10 anos em 2009, ano em que nós vamos ampliar nosso programa de certificação para satisfazer as necessidades de um mercado de TI em contínua transformação. Nós encontramos muitos desafios nos últimos dez anos e, apesar das reviravoltas da economia em 2008, vemos o aumento do número de exames aplicados ao redor do mundo. Esperamos que essa tendência positiva de certificações emitidas e exames realizados continue ao longo de 2009 e 2010, ao passo que profissionais de TI buscam atualizar seus conhecimentos e habilidades para permanecerem competitivos nesse período desafiador da economia.

Além disso, o panorama para os próximos anos no setor de Software Open Source como um todo é muito otimista. Grandes, pequenas e médias empresas, governos e outras organizações do setor público procuram novas oportunidades de negócios enquanto valorizam seu importante patrimônio pessoal. Nesse contexto, a tecnologia de Código Aberto oferece um instigante potencial em função das soluções versáteis disponibilizadas às revendas e do estável ciclo de atualizações de produtos.

Investir em tecnologia de Código Aberto é investir em pessoas, afirmação essa que sempre temos que ter em mente no Linux Professional Institute. É por isso que nossa mensagem de profissionalismo continua a ganhar apoio em todo mundo.

No LPI, nós nos guiamos pelas lições bem sucedidas do passado. Reduzindo a falta de capacitação que desafia a indústria, nosso instituto continuará a desempenhar um papel fundamental no futuro do Código Aberto. Os próximos anos serão cruciais para o LPI e nós queremos que você faça parte desse esforço em promover a contínua adoção do Linux e do Código Aberto por todo mundo!

Jim Lacey

Jim Lacey é Presidente e CEO do Linux Professional Institute.

Introdução



Muita coisa aconteceu desde a última vez que a Certificação LPI foi alterada. Alguns dos conteúdos que eram abordados encontravam pouca aplicação prática. Além disso, a organização dos tópicos não obedecia a uma ordenação lógica e em alguns pontos não havia distinção entre as provas da certificação nível 1 e as provas da certificação nível 2.

A revisão 3.0, além de eliminar alguns conteúdos ultrapassados e incluir novos conteúdos atualmente mais relevantes, procurou estabelecer focos ainda mais distintos entre a certificação nível 1 e a certificação nível 2.

A certificação nível 1 procura abordar todos os aspectos que envolvem a configuração e a manutenção de uma máquina local conectada à rede. Já a certificação nível 2 tem por objetivo geral a configuração e a manutenção de um ambiente de servidor. Apesar das mudanças, prevalece a política do LPI de abordar somente as ferramentas tradicionais de um sistema GNU/Linux, independente de distribuição. A seguir, está a visão geral das modificações nessa nova revisão da prova, fornecida pelo próprio LPI. ●

Visão geral das mudanças nos exames LPIC nível 2



A nova revisão dos objetivos para as provas LPIC nível 2, válida a partir de abril de 2009, levou as provas para a versão 3.0. Essa é a segunda revisão completa dos objetivos, que padroniza a versão para o mundo todo. No âmbito geral, o LPI antecipou o ciclo de cinco anos para revisões completas. Por volta de cada dois anos e meio, os objetivos serão modificados para refletir as possíveis mudanças do Linux. A próxima versão do LPIC-1 será a 3.5, que refletirá essa revisão parcial.

Além dessas revisões principais, haverá adendos incluídos numa média trimestral, com o intuito de esclarecer pontos e detalhes dos exames. Esses adendos não alteram a versão da prova, pois têm apenas o intuito de esclarecer a cobertura da prova para organizadores de cursos e livros.

Os novos pesos

O peso total de cada prova foi estabelecido em 60. Isso significa que, salvo em provas com perguntas “beta” para fins de desenvolvimento do exame, cada prova terá exatamente 60 questões. Portanto, a indicação de peso 3 em um determinado objetivo

indica que haverá três questões sobre o tema na prova (exceto, novamente, no caso de haver questões beta para fins de desenvolvimento dos exames).

Numeração dos objetivos

A numeração dos objetivos é passível de dúvida em função de sua falta de linearidade. Por isso, os prefixos *1.* e *2.* foram descartados nessa revisão. Em todos os momentos em que numerações como *1.xxx.y* ou *2.xxx.y* aparecem, o fazem para citar os objetivos antigos.

Redução de conteúdo duplicado

Em versões anteriores dos objetivos da certificação LPI, alguns tópicos eram abordados tanto nos exames do nível 1 quanto nos exames do nível 2. Em alguns casos, o mesmo conteúdo aparecia em diferentes provas dentro do mesmo nível de certificação. A atualização dos objetivos buscou reduzir as ocorrências de conteúdo duplicado em diferentes provas ou objetivos.

Contudo, algumas tecnologias – como DNS – são importantes nos dois níveis de certificação e estão distribuídos nos locais apropriados. Por exemplo, na certificação nível 1, a abordagem sobre o DNS está restrita à configuração do cliente do serviço. Na certificação nível 2, a abordagem passa para configuração e segurança de servidores DNS.

Versões de programas

Quando apropriado, as versões específicas de programas são mostradas nos objetivos. Por exemplo, a abordagem do Kernel 2.4 foi descartada para priorizar a versão 2.6. As questões relacionadas ao ReiserFS limitam-se à versão 3 do sistema de arquivos e o servidor Bind 8.x não é mais abordado na prova.

Alterações de conteúdo

Enquanto o foco da certificação nível 1 foi mais direcionado para o uso e administração de um sistema Linux local, a maioria dos serviços de rede e demais tarefas administrativas foram movidas para a certificação nível 2.

Além disso, os tópicos que tratam do kernel foram reunidos na certificação nível 2. Além disso, tópicos mais avançados, antes abordados na certificação nível 1, foram trazidos para a certificação nível 2. Por exemplo, a compilação de programas a partir do código fonte agora é abordada na LPIC-2. Temas relativos a segurança e solução de problemas ganharam muito mais foco nesse nível.

Solução de problemas e Segurança

A cobertura dos conteúdos relativos a solução de problemas e segurança aumentou muito. Em muitos casos, pesos originalmente marcados como 1 ou 2 passaram a 5. Com isso, o próprio tópico específico para solução de problemas tornou-se um dos mais importantes na prova.

Gerenciamento de dispositivos udev

O sistema udev foi incluído para garantir que os candidatos compreendam como é a detecção e gerenciamento de dispositivos em sistemas GNU/Linux modernos.

LVM

A abordagem ao LVM – *Logical Volume Management* – foi ampliada. Agora, o tema conta com seu próprio objetivo.

IMAP/POP

A prova 202 agora aborda servidores IMAP/POP. São abordados os mais comuns em ambientes GNU/Linux: o Courier e Dovecot. 



Como usar este livro

Este livro está organizado segundo o programa de conteúdos oficiais para a Certificação LPI-2. Dessa forma, o candidato encontrará exatamente os temas que são abordados nos exames de certificação, na profundidade que é exigida para a prova. Além disso, a sequência dos assuntos corresponde à sequência em que serão abordados na prova. Essa estrutura auxilia o candidato a manter o foco naquilo que é importante para a prova, mas sem deixar de lado a coerência e a consistência do texto.

Ao final de cada tópico foram colocadas 10 perguntas correspondentes aos temas abordados no tópico em questão, totalizando 130 questões no livro. Com a finalidade de familiarizar o candidato, as perguntas foram formuladas com o mesmo formato em que apareceram nos exames de certificação. As respostas para as perguntas de todos os tópicos encontram-se no final do livro.

Professores e escolas podem se beneficiar da adoção desse livro. Conteúdos densos são abordados de forma objetiva e coesa, o que facilita o ensino e preparação de aulas.

A leitura do livro não dispensa a experimentação prática, devendo, assim, ser acompanhada. Dado o grande volume de assuntos abordados, a utilização das ferramentas e conceitos demonstrados são muito importantes para fixação, principalmente para quem os está vendo pela primeira vez. ●



Peso total do tópico
na prova: 7

Tópico 201:

0 kernel Linux

Principais temas abordados:

- Identificação dos componentes do kernel;
- Compilação do kernel;
- Patches e personalização do kernel.

201.1 Componentes do kernel

Peso 2

O kernel é o componente central do sistema. É tão importante que muitas vezes é confundido com o sistema em sua totalidade. Ou seja, apesar de o termo Linux designar apenas o componente central – o kernel –, ele é normalmente utilizado para designar todo o sistema, que é composto por muitos outros programas.

Por isso, muitos desenvolvedores e personagens importantes do mundo do Software Livre preferem nomear o sistema como **GNU/Linux**, dado que a maior parte dos programas que funcionam em conjunto com o kernel Linux fazem parte do projeto GNU, cujo propósito é manter um ambiente de desenvolvimento e ferramentas o mais próximo possível de seus similares do Unix, porém obedecendo ao modelo de desenvolvimento aberto.

O termo código aberto refere-se a um modelo de desenvolvimento de programas de computador no qual o acesso ao código fonte é liberado para consulta, alteração e redistribuição. Isso faz com que um número muito grande de programadores possa analisar e contribuir para o desenvolvimento de uma ferramenta ou sistema, na medida em que podem alterá-lo para satisfazer suas próprias necessidades. Alterações feitas no programa original podem ser tornadas públicas ou enviadas à pessoa ou equipe responsável, que analisará a qualidade da alteração e a incorporará ao produto. Linus Torvalds, o criador e atual administrador do kernel, adotou o modelo de desenvolvimento e as licenças GNU para o Linux.

O kernel Linux é um desses componentes que juntos formam o sistema operacional. O papel do kernel é identificar e controlar a comunicação com o hardware, administrar os processos em execução e a comunicação de rede, entre outras atividades relacionadas.

O kernel e seus módulos

Existem basicamente duas formas de se escrever um kernel para qualquer plataforma: um kernel monolítico ou um micro-kernel. Diferente de um micro-kernel, um kernel monolítico agrega todas as funções dentro de um único processo. Já um micro-kernel delega cada função específica a processos derivados.



O Projeto GNU

Antes mesmo que o kernel Linux existisse, já havia o projeto GNU. O projeto foi iniciado em 1983 por Richard Stallman, desenvolvedor e ativista do software livre. O propósito do projeto GNU é criar um sistema operacional de código aberto que corresponda ao mesmo padrão dos sistemas Unix tradicionais.

Por exemplo, um micro-kernel delega a outro processo o controle das conexões de rede. Dessa forma, possíveis instabilidades na rede não comprometem o funcionamento essencial do sistema. Entretanto, o desenvolvimento de um micro-kernel é muito mais demorado em relação a um kernel monolítico, pois rastrear eventuais falhas e incluir novos recursos é muito mais complicado do que em um kernel monolítico.

O kernel Linux é *monolítico*. Porém, sua arquitetura é chamada *modular*. Isso significa que, mesmo sendo um kernel monolítico, todas as suas funções não precisam necessariamente estar todo o tempo presentes na memória. Por exemplo, o kernel pode estar configurado para trabalhar com dispositivos USB, mas não para manter em memória as funções exigidas para trabalhar com tais dispositivos. Mantidas em módulos, essas funções somente serão carregadas para a memória quando forem necessárias, ou seja, quando for conectado um dispositivo USB.

É importante não confundir um kernel modular com um micro-kernel. Apesar de modular, o kernel Linux é um kernel monolítico. Cada módulo carregado é integrado ao kernel ativo e, apesar de em sua maioria poderem ser descarregados da memória, o kernel continua se comportando como único e centralizado.

Versões do kernel

A versão de um kernel Linux é composta de quatro números. Juntos, eles informam não só quão recente é o kernel, mas também algumas outras características.

Esses quatro números que compõem a versão do kernel são separados por pontos, no formato *A.B.C.D*. O último elemento – *D* – nem sempre é utilizado, mas tem função muito importante.

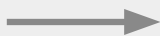
Sufixos do kernel

Além dos quatro números, a versão do kernel pode possuir um sufixo que representa um kernel diferente do oficial – chamado *vanilla* –, que por sua vez representa um recurso adicional, uma versão de testes ou outra diferença em relação à versão oficial estável.

Os sufixos mais comuns são *rc* e *mm*. O sufixo *rc* (*release candidate*, ou candidato à versão) representa um *prepatch*, que é equivalente a uma versão alfa do kernel, e seus arquivos fonte ficam no diretório de teste no servidor do oficial do kernel (www.kernel.org). Essas versões parciais são aplicadas ao kernel completo da versão imediatamente anterior, usando o comando *patch*. Por exemplo, o patch 2.6.31-rc5 deve ser aplicado à versão 2.6.30. Apenas as versões de kernel com três números podem receber um patch *rc*. Portanto, as versões release candidate não devem ser aplicadas a um kernel com quatro números de versão, como 2.6.30.4.

Versão do Kernel Linux

2

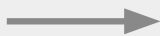


Versão principal

Muda apenas quando ocorrem transformações radicais na estrutura do kernel. Está atualmente na versão 2.

•

6

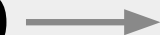


Número de revisão principal

Até a versão 2.4 do kernel, esse segmento, quando par, tratava de uma versão estável. Quando ímpar, tratava de uma versão em desenvolvimento, considerada instável. Da versão 2.6 em diante, não utiliza-se mais essa distinção entre par e ímpar.

•

30

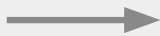


Número de revisão secundário

Nas versões mais antigas do kernel, determinava correções e patches de segurança. Hoje demonstra se houve inclusões de novos recursos, como drivers de dispositivos, por exemplo.

•

4



Correções urgentes

Utilizado quando há revisões de segurança urgentes, que alteram aspectos relativos à revisão secundária.

O sufixo *mm* representa um kernel com as modificações realizadas pelo desenvolvedor Andrew Morton, feitas sobre a versão oficial do kernel. Essas alterações são geralmente mais experimentais do que aquelas feitas nas versões oficiais. Todas essas diferentes versões podem ser obtidas diretamente do site oficial do kernel: www.kernel.org.

Localização do kernel no sistema

O kernel Linux oficial é distribuído como código fonte, ou seja, precisa ser configurado e compilado para ser utilizado pelo sistema. Depois de copiado do site oficial ou usando as ferramentas de instalação da distribuição, o código-fonte do kernel deve ser mantido na máquina local, no diretório `/usr/src/`, num lugar com as informações de versão do kernel, como `/usr/src/linux-2.6.30`. Como é possível possuir mais de uma versão dos códigos-fonte, é importante criar um link simbólico `/usr/src/linux` que aponta para o diretório do código-fonte do kernel atualmente em uso. Esse procedimento é importante para que outros programas possam eventualmente localizar os arquivos do kernel atual.

Ali encontram-se não só os arquivos de código-fonte do kernel, mas também a documentação oficial e onde estará o arquivo imagem do kernel após compilado. O



Kernel original x Kernel modificado

Na grande maioria dos casos, não é necessário fazer qualquer alteração no kernel original para o funcionamento adequado do sistema. Contudo, algumas distribuições fazem pequenas alterações no kernel original principalmente para fins estéticos. Por exemplo, é necessário fazer alterações para exibir uma imagem de progresso durante o início do sistema. Caso essa alteração não seja feita, serão exibidas as informações de controle do sistema, comportamento padrão no kernel original.

arquivo imagem é o kernel em si, que mais tarde será invocado pelo carregador de boot durante o carregamento do sistema.

Imagem e documentação do kernel

A documentação oficial do kernel em questão fica em `/usr/src/linux/Documentation`. Neste diretório encontram-se vários arquivos de texto que documentam aspectos específicos do kernel. Por exemplo, para descobrir quais parâmetros o kernel aceita ao ser carregado, o arquivo `kernel-parameters.txt` pode ser consultado.

Após compilar um novo kernel a partir de seu código-fonte, o arquivo imagem final será encontrado em `/usr/src/linux/arch/x86/boot/`. Note que o subdiretório `x86` varia conforme a arquitetura escolhida durante a configuração do kernel. Se for escolhida a arquitetura PowerPC, por exemplo, a imagem estará em `/usr/src/linux/arch/powerpc/boot/`.

O nome e o tamanho do kernel variam conforme for invocada a compilação. Existem basicamente dois formatos de imagem de kernel: `zImage` e `bzImage`. Ambos são compactados com o método `zlib`, mas o que os difere é o tamanho máximo de arquivo. Um arquivo `zImage` possui tamanho máximo de 512 Kb, tamanho adequado para versões mais antigas do kernel. As versões recentes exigem um tamanho maior, por isso foi instituído o arquivo `bzImage` – *big zImage* –, que não possui essa limitação de tamanho. ●

201.2 Compilando um kernel



Peso 2

Apesar de a maioria das distribuições acompanhar imagens de kernel pré-compiladas, pode ser necessário personalizar o kernel para corresponder a necessidades específicas, como suporte a hardware incomum ou a um sistema de arquivos exótico. Também é comum recompilar o kernel para ocupar menos recursos e operar em máquinas mais antigas.

O código-fonte do kernel pode ser obtido por meio de pacote específico da distribuição ou diretamente do site oficial *www.kernel.org*. No último caso, o código é distribuído como um arquivo Tar compactado, e deve ser extraído no local padrão, */usr/src/linux*, que geralmente é um link simbólico para o diretório criado na extração, como */usr/src/linux-x.x.xx*.

O processo de personalização de um kernel exige três etapas principais: configuração, compilação e, por fim, instalação.

Configuração

A configuração de um novo kernel é feita invocando o comando `make`, de dentro do diretório onde está o código-fonte do kernel. Existem diferentes tipos de interfaces de configuração, mas a mais tradicional é a interface *ncurses*, invocada por meio do comando `make menuconfig`. A configuração é feita no próprio terminal, numa interface simples de menu (**figura 1**).

Existem outras maneiras de configurar o kernel, mas todas elas produzem o mesmo resultado, que é gerar o arquivo `.config` no diretório do código-fonte do kernel. Interfaces alternativas à configuração via terminal são os assistentes de configuração feitos em Qt (**figura 2**) e em Gtk (**figura 3**). Ambos devem ser executados de dentro do ambiente gráfico X.



Figura 1. A interface de configuração ncurses é a mais simples e a mais utilizada para configuração do kernel.

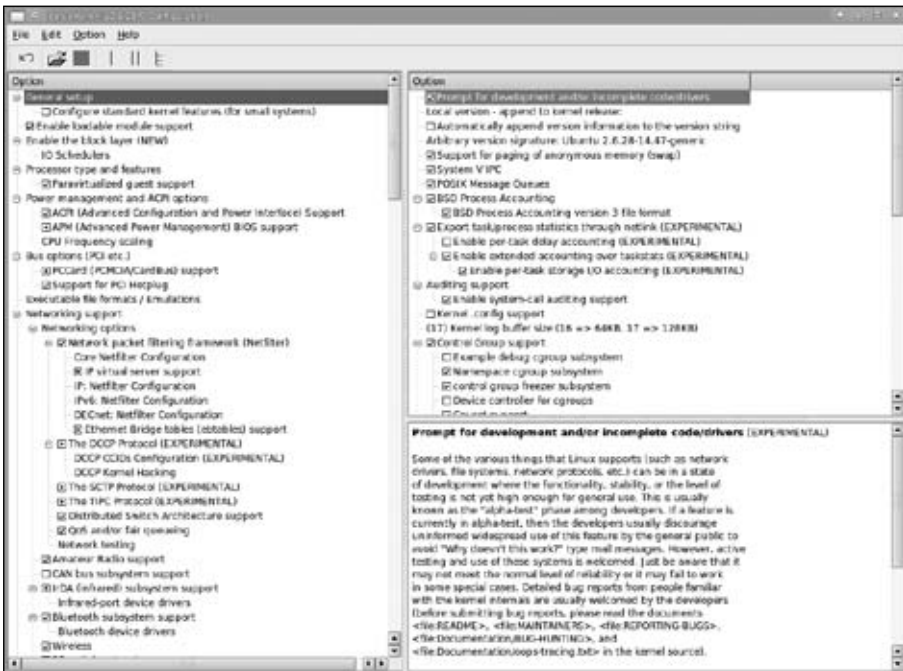


Figura 2. A interface de configuração do kernel feita em Qt.



Figura 3. A interface de configuração do kernel feita em Gtk.



Pré-requisitos para configuração e compilação

Para que seja possível compilar um kernel, é necessário que estejam presentes no computador as ferramentas de desenvolvimento, ou seja, os programas e arquivos necessários para compilar outros programas. Essencialmente, é necessário um compilador da linguagem C – o `gcc` – e alguns acessórios, como o próprio comando `make` e bibliotecas. A instalação dessas ferramentas varia de acordo com a distribuição. Em distribuições Debian e derivados, como a distribuição Ubuntu, basta executar o comando `apt-get install build-essential`, que se encarrega de instalar os programas de desenvolvimento e compilação fundamentais. Em ambientes Red Hat ou derivados, como o Fedora, a instalação pode ser feita com os comandos `yum groupinstall "Development Tools"` e `yum install kernel-devel kernel-headers`. Mesmo as interfaces de configuração do kernel são compiladas antes de serem utilizadas. Por isso, é necessário que estejam instalados os pacotes de desenvolvimento para a interface desejada, sejam eles *ncurses*, *Qt* ou *Gtk*.

O arquivo gerado após a configuração do kernel, `.config`, também pode ser editado num editor de texto convencional. Porém, essa prática não é recomendada, pois além de o arquivo ser demasiado extenso há opções interdependentes que podem ser quebradas. A recomendação é sempre utilizar um meio tradicional de configuração. Outras formas de configurar o kernel são:

- **make config**: Criar ou atualizar a configuração por meio de uma interface orientada a perguntas e respostas na linha de comando;
- **make oldconfig**: Recupera todas as configurações do arquivo `.config` pré-existente e pergunta apenas sobre novas opções.

Compilação

Finalizada a configuração, o kernel já pode ser compilado. Se não for a primeira compilação do kernel, pode ser conveniente executar `make mrproper`, que apaga as configurações e os arquivos gerados durante uma compilação anterior. Dessa forma, se evita que hajam objetos com dependências desencontradas. Para apagar apenas os arquivos compilados anteriormente, mas preservar a configuração, utiliza-se `make clean`.



Acelerando a compilação

Boa parte das máquinas modernas são equipadas com mais de um processador ou processadores de mais de um núcleo. Para aproveitar essa capacidade é recomendável disparar processos de compilação simultâneos. Isso é feito com a opção `-j` do comando `make`. Por exemplo, `make -j4 bzImage` compilará o kernel em quatro processos simultâneos, o que agilizará bastante a compilação.

Para gerar a imagem `zImage` ou `bzImage` do kernel, utiliza-se `make zImage` ou `make bzImage`, respectivamente. Os comandos apenas funcionarão se for obedecida a grafia correta, com o *I* maiúsculo em `zImage` e `bzImage`. A compilação pode demorar de minutos a horas, dependendo do número de recursos escolhidos e do computador utilizado.

Provavelmente o novo kernel será modular, e os módulos precisarão ser compilados separadamente. Isso é feito com o comando `make modules`.

Instalação

Depois que o kernel e os módulos forem compilados, é necessário copiá-los para a localização correta e configurar o carregador de boot para que possa encontrá-lo ao iniciar o sistema.

Num computador de arquitetura x86, o arquivo de imagem do novo kernel estará em `/usr/src/linux/arch/x86/boot/zImage` ou em `/usr/src/linux/arch/x86/boot/bzImage`. Esse arquivo deve ser copiado para o diretório `/boot` e deve ser renomeado para algo mais claro e apropriado, como `vmlinuz-2.6.28-abc`. O termo `vmlinuz` é a nomenclatura padrão para os arquivos de kernel do Linux. Recomenda-se também especificar a versão no nome do kernel em questão e utilizar um sufixo que indique a sua especificidade.

Já os módulos possuem comandos específicos para instalação. Após compilados, são instalados usando o comando `make modules_install`. Será criado um diretório com a numeração do kernel em `/lib/modules`. Portanto, para um kernel 2.6.30 o diretório dos módulos será `/lib/modules/2.6.30`. O comando `make modules_install` também se encarrega de gerar o arquivo `modules.dep` nesse diretório, que armazena as informações de interdependência dos módulos.

Initial Ramdisk

A partir da versão 2.6 do kernel é possível criar um arquivo chamado *Initial Ramdisk*. Um Initial ramdisk é um pequeno conjunto de arquivos que é carregado na memória RAM durante o carregamento do kernel e montado como se fosse uma partição, porém sem que haja um sistema de arquivos. Como o Initial Ramdisk é montado antes da partição raiz, o motivo principal para utilizá-lo é a necessidade de utilizar um módulo essencial para montar a partição raiz, geralmente o módulo que oferece suporte ao sistema de arquivos da partição raiz (xfs, ext3, reiserfs etc.) ou o que oferece suporte para a controladora a qual o disco está conectado (SCSI, RAID etc.). Como há muitas opções desse tipo disponíveis na configuração do kernel, é mais prático compilar um kernel genérico e adicionar esses recursos como módulos.

Dependendo da distribuição Linux utilizada, o comando que gera o Initial Ramdisk pode ser o `mkinitrd` ou o `mkinitramfs`. No caso do **mkinitrd**, as opções mais

importantes são **-c** (eliminar árvore de diretórios usada para criação anterior do Initial Ramdisk), **-k** (versão do kernel), **-m** (lista de módulos a incluir, separados por dois-pontos) e **-o** (nome para o Initial Ramdisk a ser criado). Portanto, para criar um Initial Ramdisk para o Kernel 2.6.30 instalado incluindo o módulo xfs, faça:

```
# mkinitrd -c -k 2.6.30 -m xfs -o /boot/initrd-2.6.30.gz
```

Para incluir outros módulos além do xfs, basta separá-los por dois-pontos:

```
# mkinitrd -c -k 2.6.30 -m xfs:reiserfs:ext3 -o /boot/initrd-2.6.30.gz
```

A utilização do **mkinitramfs** é ligeiramente diferente. Todas as principais configurações são especificadas no arquivo `/etc/initramfs-tools/initramfs.conf`. Ali é possível definir quais módulos devem ser incluídos, em grupos pré-determinados ou individualmente. A principal opção de linha de comando para o **mkinitramfs** é **-o**, que determina o arquivo Initial Ramdisk. Caso não seja informada a versão para qual deve ser gerada a Initial Ramdisk, será assumida a versão atual. Para gerar a Initial Ramdisk para o kernel 2.6.30 instalado, faça:

```
# mkinitramfs -o /boot/initramfs-2.6.30 2.6.30
```

Nas distribuições que utilizam o **mkinitramfs**, também há o comando `update-initramfs`. No lugar de criar uma nova Initial Ramdisk, ele atualiza aquela já existente de acordo com as opções do arquivo `initramfs.conf`.

Atualizando o bootloader

Após compilar e instalar o kernel e seus módulos e criar a Initial Ramdisk, o carregador de boot deve ser atualizado para que o kernel possa ser localizado e carregado após o religamento do sistema.

Essa configuração é feita de diferentes maneiras, dependendo de qual bootloader é utilizado: LILO ou GRUB.

No caso do Lilo, as alterações são feitas no arquivo `/etc/lilo.conf`. Por exemplo:


```
image = /boot/vmlinuz-2.6.30
root = /dev/sda1
initrd = /boot/initrd-2.6.30.gz
label = Linux-2.6.30
read-only
```

Após editar o arquivo `/etc/lilo.conf`, o comando `lilo` deve ser executado para que as alterações tenham efeito.

É recomendável manter o kernel anterior e apenas incluir o novo kernel, a fim de que seja possível iniciar o sistema com o kernel anterior caso haja algum problema.

Se o bootloader utilizado for o Grub, dev-se editar o arquivo `/boot/grub/menu.lst`. Inclua as seguintes linhas, com as opções correspondentes ao novo kernel:

```
title Linux 2.6.30
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.30root=/dev/sda1 ro
    initrd /boot/initrd-2.6.30.gz
```

Para o Grub, não é necessário executar nenhum comando após a alteração no arquivo de configuração. O novo kernel aparecerá automaticamente no menu de início do sistema. 

201.3 Aplicar um patch ao kernel

Peso 1

O kernel Linux é um grande e complexo sistema, com áreas e sub-áreas praticamente independentes. Novos recursos, aprimoramentos e correções são liberados o tempo todo e, em sua maioria, dizem respeito a um aspecto muito específico do kernel.

Portanto, não é prático copiar todo o código-fonte do kernel em função de uma alteração em apenas uma pequena parte dele. É possível utilizar um *patch* – algo como “remendo” numa tradução livre – para que apenas os trechos relevantes de código sejam alterados.

Os patches do kernel são trechos de código que contêm as correções, aprimoramento de código antigo e inclusão de novos recursos. Por exemplo, o arquivo `patch-2.6.30.5.bz2` é o quinto patch corretivo para o o kernel da série 2.6.30.

Obtenção e aplicação de patches

Assim como o código-fonte oficial do kernel, os patches são fornecidos no site oficial do kernel – www.kernel.org –, em arquivos compactados. Por exemplo, `patch-2.6.30.1.bz2`. Um patch deve ser aplicado apenas à versão imediatamente anterior do kernel. Portanto, o patch para o kernel 2.6.31 deve ser aplicado somente ao kernel 2.6.30.

O comando utilizado para aplicar um patch tem o nome cognato *patch*. Um patch deve ser aplicado a partir do diretório raiz do código-fonte. No caso do kernel, em `/usr/src/linux`.

Dentro do arquivo patch existem as localizações, o nome de cada arquivo que precisará ser alterado e quais são as alterações. A maneira mais prática de aplicar um patch é direcionar todo o conteúdo do arquivo para a entrada padrão do comando patch. Isso pode ser feito numa única linha, utilizando o comando `bzcat` para arquivos `.bz2` ou `zcat` para arquivos `.gz`.

No interior dos arquivos de patch oficiais há uma letra ou um termo antes do caminho para o arquivo que deve ser alterado:

```
a/arch/alpha/include/asm/percpu.h
```

Esse caminho indica que uma alteração deverá ser feita no arquivo `arch/alpha/include/asm/percpu.h`. Portanto, a letra “a” deve ser retirada do caminho, o que é feito com a opção `-p1`:

```
cd /usr/src/linux
wget http://kernel.org/pub/linux/kernel/v2.6/patch-2.6.30.3.bz2
bzcat patch-2.6.30.3.bz2 | patch -p1
```

Revertendo um patch

Diferentemente dos patches do kernel com três números de versão (como 2.6.30, 2.6.31, 2.6.31 etc.), os patches com quatro números de versão devem ser removidos antes de aplicar outro patch.

Portanto, antes de aplicar o patch 2.6.30.4 ao kernel 2.6.30.3, é necessário reverter o patch 2.6.30.3:

```
bzcat patch-2.6.30.3.bz2 | patch -p1 -R
```

O próprio patch 2.6.30.3 é usado para realizar a reversão. Somente então o patch 2.6.30.4 pode ser aplicado:

```
bzcat patch-2.6.30.4.bz2 | patch -p1
```

Apesar de os patches com três números de versão não necessitarem serem revertidos, é importante aplicar todos os patches na sequência, sem intervalos. Ou seja, antes de aplicar o patch 2.6.33 ao kernel 2.6.30, é necessário aplicar os patches 2.6.31 e 2.6.32. ●



201.4 Configurar, compilar e instalar um kernel personalizado e seus módulos

Peso 2

A configuração do kernel é uma tarefa que exige conhecimentos sobre cada aspecto do sistema e da arquitetura. Uma configuração mal feita pode provocar mal funcionamento ou mesmo o total travamento do sistema. A melhor recomendação é utilizar como base uma configuração pré-existente e alterar somente aquilo que se tem certeza.

Enquanto alguns recursos precisam ser compilados internamente (embutidos no kernel), a maioria pode ser compilada como dinamicamente carregável (como um módulo carregável e descarregável). Na interface de configuração via terminal – invocada com `make menuconfig` – o item marcado com asterisco (*) será compilado internamente, ao passo que aqueles marcados com a letra M serão compilados como módulos. Itens deixados em branco serão descartados para o kernel em questão. Espaços de escolha com colchetes ([]) indicam que o item só poderá ser compilado internamente, enquanto que espaços de escolha com sinais de maior e menor (< >) indicam que o item poderá ser compilado tanto internamente quanto como módulo.

Principais seções de configuração

A configuração do kernel está dividida nos eixos principais:

- **Code maturity level options:** Mostrar ou não recursos do kernel considerados instáveis;
- **General setup:** Características gerais do kernel. É possível incluir um termo de versão para o kernel personalizado;
- **Loadable module support:** Suporte ao sistema de módulos e definição de algumas características;
- **Processor type and features:** Indica o tipo de processador que o kernel utilizará, e recursos como multiprocessamento;
- **Power management options (ACPI, APM):** Opções relativas ao controle de energia. Indicado especialmente para laptops;
- **Bus options (PCI, PCMCIA, EISA, MCA, ISA):** Suporte aos diferentes tipos de barramentos;
- **Executable file formats:** Tipos de arquivos que o sistema operacional será capaz de executar;
- **Networking:** Suporte e configuração dos diferentes tipos de plataformas de rede;
- **Device Drivers:** Escolha e configuração dos dispositivos de hardware, integrados e periféricos;

- **File systems:** Lista de sistemas de arquivos compatíveis e recursos a eles relacionados;
- **Kernel hacking:** Opções de depuração do kernel.

Essas são as principais categorias de configuração, que podem diferenciar de uma versão do kernel para outra. Praticamente toda opção de configuração do kernel conta com um pequeno texto de ajuda, exibido ao pressionar o ponto de interrogação no teclado. Além de explicar a finalidade da opção, é feita uma recomendação sobre incluir ou não o item na configuração do kernel.

As configurações são salvas no arquivo `/usr/src/linux/.config`, que será usado para guiar a construção do novo kernel e módulos. No arquivo `Makefile` é possível mudar variáveis como **EXTRAVERSION**, que indica ser uma compilação de kernel personalizado.

Empacotamento do kernel

A maneira ideal de distribuir um kernel personalizado é criar pacotes Tar, RPM ou Deb, utilizando os próprios recursos de compilação do kernel:

- `make rpm-pkg`: Gera um pacote RPM compilado e com código-fonte;
- `make binrpm-pkg`: Gera um pacote RPM compilado;
- `make deb-pkg`: Gera um pacote compilado Deb;
- `make tar-pkg`: Gera um arquivo Tar sem compressão;
- `make targz-pkg`: Gera um arquivo Tar com compressão gzip;
- `make tarbz2-pkg`: Gera um arquivo Tar com compressão bzip2.

Criar um pacote com um kernel personalizado pode ser útil especialmente em ambientes de rede com hardware em comum, que eventualmente necessite de algum recurso específico. O empacotamento evita a configuração e compilação individual e a cópia manual de cada arquivo necessário para as demais máquinas. ●

201.5 Controlar/investigar o kernel e seus módulos durante sua execução

Peso 3

Enquanto que as interfaces de configuração do kernel oferecem toda a gama de ajustes para a criação de um novo kernel, os recursos disponíveis no kernel atualmente em execução podem não estar tão evidentes. Saber localizar e interagir com um kernel e seus módulos já instalados pode tornar desnecessária uma possível reconfigu-

ração e compilação. Um dos comandos mais importantes para obter informações do kernel em execução é o comando `uname`. Quando usado apenas com a opção `-a`, mostra todas as informações sobre o kernel atual:

```
# uname -a
Linux themisto 2.6.26-2-686 #1 SMP Sun Jul 26 21:25:33 UTC 2009 i686 GNU/Linux
```

A tabela **Saída do comando `uname`** mostra o que representam cada um dos valores.

O `uname` também pode ser utilizado para exibir apenas alguma dessas informações isoladamente, fornecendo as opções:

- **-i:** Tipo do processador;
- **-m:** Nome da máquina;
- **-n:** Nome da máquina na rede;
- **-p:** Processador;
- **-o:** Sistema operacional;
- **-r:** Versão do código fonte do kernel;
- **-s:** Nome do kernel;
- **-v:** Versão de compilação do kernel.

Eventualmente, alguns desses valores – como *Processador* – pode ser mostrado como *unknown* (desconhecido). Isso acontece quando é um kernel genérico sem informação específica de processador.

O `uname` é bastante utilizado em scripts de compilação, para identificar onde estão localizados os módulos do kernel atual. A saída do comando `uname -r` indica exatamente o nome do diretório em `/lib/modules` que armazena os módulos para o kernel atual. Além disso, o `uname` pode fornecer informações de recursos disponíveis, como SMP (*Symmetric Multi Processing*, para máquinas com mais de um proces-



Saída do comando `uname`

Valor	Descrição
Linux	Nome do kernel
themisto	Nome da máquina
2.6.26-2-686	Versão de kernel utilizada
#1 SMP Sun Jul 26 21:25:33 UTC 2009	Versão de compilação do kernel
i686	Arquitetura/processador
GNU/Linux	Sistema Operacional

sador ou mais de um núcleo por processador), RT (*Realtime*, para sistema com baixa latência de resposta) ou compatibilidade com processadores de 64 bits.

O comando `uname` extrai essas informações de arquivos localizados no diretório `/proc/sys/kernel/`, diretório lógico criado pelo próprio kernel para disponibilizá-las ao `uname` e a outros programas que possam necessitar delas.

Módulos do kernel

A maioria das funcionalidades de um kernel podem existir na forma de módulos externos, ou seja, em arquivos externos que não necessitam estar o tempo todo carregados na memória. Os módulos ficam no diretório `/lib/modules/`, no diretório correspondente à versão do kernel obtida com o comando `uname -r`.

Assim que o kernel identifica um dispositivo ou quando um recurso modular é solicitado, o módulo correspondente é carregado automaticamente, invocado pelo comando `modprobe` ou `insmod*`.

A diferença entre os dois comandos é que o `modprobe` carrega o módulo especificado e os módulos dos quais ele depende, conforme determinado no arquivo `modules.dep` – presente no diretório dos módulos correspondentes ao kernel. Ele automaticamente identifica o kernel em execução e busca o módulo no diretório apropriado, sem necessidade de oferecer o caminho completo. Já o comando `insmod` precisa do caminho completo para o módulo e apenas o carrega sem verificar suas dependências.

O arquivo `modules.dep`, que armazena quais são os módulos disponíveis e quais são as dependências de cada módulo, é gerado ou atualizado pelo comando `depmod -a`. Isso é importante pois a maioria dos módulos dependem de recursos que estão presentes em outros módulos, que podem ainda não estar presentes na memória. Ao analisar as dependências no arquivo `modules.dep`, o comando `modprobe` carrega tanto o módulo solicitado quanto os outros módulos dos quais ele depende.

O arquivo `modules.dep` é gerado automaticamente quando novos módulos são instalados a partir de uma compilação do kernel oficial, pois o comando `depmod -a` é invocado ao término do processo de instalação (`make modules_install`). Quando é instalado um módulo não oficial, o `depmod` pode não ser executado automaticamente. Nesse caso, basta invocar o comando `depmod -a` manualmente, como usuário `root`.

Inspecionando e interagindo com módulos

Todos os módulos atualmente carregados podem ser verificados com o comando `lsmod`. Exemplo de lista de módulos carregados:

Apesar de o comando `insmod` ainda ser aceito, o comando padrão para manejo de módulos nas versões mais recentes do kernel é o `modprobe`.

```
# lsmod
Module                Size  Used by
snd_usb_audio          81440  0
snd_usb_lib            15276  1 snd_usb_audio
ppp_deflate            4652  0
bsd_comp               5228  0
ppp_async              8428  1
crc_ccitt              1612  1 ppp_async
ppp_generic            23136  7 ppp_deflate,bsd_comp,ppp_async
slhc                   5260  1 ppp_generic
option                 21296  1
usbserial              30568  3 option
...
ppdev                  7184  0
v4l2_common            13164  2 tuner,saa7134
snd                    47668  19 snd_usb_audio,snd_usb_lib,snd_seq,snd_hda_codec_
↳ analog,snd_ice1712,snd_ak4xxx_adda,snd_cs8427,snd_ac97_codec,snd_hda_intel,snd_
↳ i2c,snd_hda_codec,snd_mpu401_uart,snd_rawmidi,snd_seq_device,snd_hwdep,snd_
↳ pcm,snd_timer
videodev               36160  3 tuner,saa7134,v4l2_common
v4l1_compat            14192  1 videodev
videobuf_dma_sg        10960  1 saa7134
videobuf_core          15056  2 saa7134,videobuf_dma_sg
tveeprom               11568  1 saa7134
cdrom                   33888  1 sr_mod
r8169                   32624  0
parport_pc             24932  1
rtc_cmos                10156  0
thermal                12936  0
parport                 31852  3 lp,ppdev,parport_pc
mii                     4236  1 r8169
i2c_core                20480  8 tuner_simple,tea5767,tuner,nvidia,saa7134,i2c_
↳ i801,v4l2_common,tveeprom
soundcore               1200  1 snd
rtc_core                15976  1 rtc_cmos
snd_page_alloc         8372  2 snd_hda_intel,snd_pcm
rtc_lib                 2316  1 rtc_core
processor               34240  1 acpi_cpufreq
```

button	5212	0
ns558	4492	0
gameport	10360	2 ns558
evdev	9440	7

Devido ao grande número de módulos carregados no sistema do exemplo, a lista foi cortada para exibir apenas parte dos módulos. A primeira coluna mostra todos os módulos atualmente carregados, seguido do tamanho em bytes que ocupa na memória. A terceira coluna mostra o número de módulos ou outros recursos que estão usando o módulo em questão, seguida dos nomes desses módulos. Quando não há módulos dependentes listados, é porque eles foram compilados internamente no kernel.

Um módulo não poderá ser descarregado se estiver sendo utilizado, seja por um dispositivo ou por um programa. Se o módulo estiver ocioso, pode ser descarregado com o comando `rmmod nome_do_módulo`. Seguindo o mesmo princípio do comando `insmod`, o comando `rmmod` somente descarregará o módulo indicado, mas outros módulos presentes apenas para satisfazer as dependências deste permanecerão carregados. Ainda, se o módulo que se deseja descarregar é pré-requisito para outro recurso, não poderá ser descarregado.

Para descarregar um módulo e os módulos diretamente relacionados – desde que não estejam sendo utilizados por outros recursos – é novamente usado o comando `modprobe`, com a opção `-r`. Por exemplo, para descarregar o módulo `snd-hda-intel` (dispositivo de som HDA Intel) e demais módulos relacionados ao sistema de som (caso não estejam sendo utilizados por outros recursos), faça:

```
modprobe -r snd-hda-intel
```

Além da possibilidade de carregar e descarregar módulos do kernel com o sistema em funcionamento, é possível passar opções para um módulo durante seu carregamento de maneira semelhante a passar opções para um comando.

Cada módulo aceita opções e valores específicos, mas que na maioria das vezes não precisam ser informados. Contudo, em alguns casos pode ser necessário passar parâmetros ao módulo para alterar seu comportamento e operar como desejado.

O comando `modinfo` mostra a descrição, o arquivo, o autor, a licença, a identificação, as dependências e os parâmetros para o módulo solicitado. Usando como argumento apenas o nome do módulo, mostra todas as informações disponíveis:

```
# modinfo snd-hda-intel
filename:      /lib/modules/2.6.30.4-smp/kernel/sound/pci/hda/snd-hda-intel.ko
```

```

description:   Intel HDA driver
license:      GPL
alias:        pci:v00001002d*sv*sd*bc04sc03i00*
alias:        pci:v00006549d00001200sv*sd*bc*sc*i*
...
alias:        pci:v00008086d000027D8sv*sd*bc*sc*i*
alias:        pci:v00008086d00002668sv*sd*bc*sc*i*
depends:       snd-pcm,snd,snd-page-alloc,snd-hda-codec
vermagic:     2.6.30.4-smp SMP preempt mod_unload PENTIUM4
parm:         index:Index value for Intel HD audio interface. (array of int)
parm:         id:ID string for Intel HD audio interface. (array of charp)
parm:         enable:Enable Intel HD audio interface. (array of bool)
parm:         model:Use the given board model. (array of charp)
parm:         position_fix:Fix DMA pointer (0 = auto, 1 = none, 2 = POSBUF).
↳ (array of int)
parm:         bdl_pos_adj:BDL position adjustment offset. (array of int)
parm:         probe_mask:Bitmask to probe codecs (default = -1). (array of int)
parm:         probe_only:Only probing and no codec initialization. (array of bool)
parm:         single_cmd:Use single command to communicate with codecs (for
↳ debugging only). (bool)
parm:         enable_msi:Enable Message Signaled Interrupt (MSI) (int)
parm:         power_save:Automatic power-saving timeout (in second, 0 =
↳ disable). (int)
parm:         power_save_controller:Reset controller in power save mode. (bool)

```

As linhas começadas com *alias* mostram todas as identificações de hardware correspondentes ao módulo em questão. Ou seja, todo hardware identificado com um desses valores provocará o carregamento desse módulo.

Para listar apenas as opções que o módulo aceita, o comando `modinfo` deve receber a opção **-p**:

```

# modinfo -p snd-hda-intel
power_save_controller:Reset controller in power save mode.
power_save:Automatic power-saving timeout (in second, 0 = disable).
enable_msi:Enable Message Signaled Interrupt (MSI)
single_cmd:Use single command to communicate with codecs (for debugging only).
probe_only:Only probing and no codec initialization.
probe_mask:Bitmask to probe codecs (default = -1).

```



```

bdl_pos_adj:BDL position adjustment offset.
position_fix:Fix DMA pointer (0 = auto, 1 = none, 2 = POSBUF).
model:Use the given board model.
enable:Enable Intel HD audio interface.
id:ID string for Intel HD audio interface.
index:Index value for Intel HD audio interface.

```

Cada uma dessas opções pode receber parâmetros diferentes dos valores padrão. Por exemplo, usar a opção *model* para determinar um modelo do dispositivo específico:

```
modprobe snd-hda-intel model=3stack
```

Instruções sobre os parâmetros dos módulos podem ser encontrados na própria documentação do kernel, no diretório *Documentation* do código fonte.

Esses parâmetros personalizados são passados no momento em que o módulo é carregado. No entanto, o mais comum é que os módulos sejam carregados automaticamente pelo sistema. Por isso, os parâmetros personalizados devem ser armazenados no arquivo `/etc/modprobe.d/modprobe.conf` (ou dentro de um arquivo individual nesse mesmo diretório). O mesmo parâmetro do exemplo poderia estar no arquivo `modules.conf` na forma:

```
options snd-hda-intel model=3stack
```

No mesmo arquivo `modprobe.conf`, é possível criar nomes alternativos para os módulos, chamados *aliases*. Dessa forma, o módulo pode ser invocado por um nome mais genérico. Por exemplo, um alias chamado *sound-slot-0* para o módulo `snd-hda-intel`:


```
alias sound-slot-0 snd_hda_intel
```

Outra possibilidade do `modprobe.conf` é definir as ações que devem ser executadas antes e depois do carregamento e descarregamento de módulos, como a execução de um programa. Por exemplo, carregar primeiro o módulo `snd-ice1712` quando o módulo `snd-hda-intel` for carregado:

```
install snd-hda-intel modprobe snd-ice1712; modprobe --ignore-install snd-hda-
↳intel
```


A opção `--ignore-install` é necessária para que o `modprobe` não execute novamente a instrução `install` referente ao módulo `snd-hda-intel` no arquivo `modprobe.conf`. É possível usar o mesmo procedimento ao descarregar módulos:

```
remove snd-hda-intel modprobe --ignore-remove -r snd-hda-intel; modprobe -r  
↳ snd-ice1712
```

Dessa vez, a opção `--ignore-remove` aparece para impedir a execução cíclica da instrução `remove`, referente ao módulo `snd-hda-intel` no arquivo `modprobe.conf`. 



Exercícios



Questões Tópico 201

1. A estrutura do kernel Linux oficial pode ser classificada como:
 - a. Monolítica
 - b. Micro-kernel
 - c. Estável
 - d. Multitarefa

2. Qual é o formato correto de numeração de versão do kernel Linux?
 - a. 9.04
 - b. 2009
 - c. 2.6.31
 - d. 2008 Server

3. O que representam os sufixos adicionados ao número de versão do kernel?
 - a. As iniciais do nome do autor daquele kernel.
 - b. Um kernel específico, alterado em relação à versão correspondente oficial.
 - c. Uma versão instável do kernel.
 - d. Um kernel que não é fornecido pelo site oficial *www.kernel.org*.

4. Onde pode ser encontrada a documentação oficial específica de uma versão do kernel?
 - a. Em grupos de discussão de Linux.
 - b. No código fonte dos módulos.
 - c. No diretório Documentation onde foi colocado o código-fonte do kernel.
 - d. Nos comentários deixados pelos desenvolvedores.

5. O comando utilizado para aplicar patches ao kernel é o _____ .

6. Quais comandos podem ser utilizados para gerar uma Initial Ramdisk? Marque todas as opções corretas.
- a. mkisofs
 - b. fdisk
 - c. mkinitramfs
 - d. mkinitrd
7. Em qual arquivo ficam armazenadas as informações sobre a disponibilidade de módulos do kernel e suas dependências? Informe somente o nome do arquivo, sem o diretório.
8. Qual comando pode ser utilizado para configurar o kernel Linux? Marque todos os corretos.
- a. make config
 - b. make xconfig
 - c. make menuconfig
 - d. make gconfig
9. O comando `uname -r` informa:
- a. os recursos disponíveis no kernel em execução.
 - b. os módulos carregados na memória para o kernel em execução.
 - c. os módulos não carregados na memória para o kernel em execução.
 - d. a versão do código fonte do kernel em execução.
10. A maneira correta de carregar o módulo `ehci_hcd` e suas dependências é:
- a. make modules_install
 - b. make modules_install ehci_hcd
 - c. modprobe ehci_hcd
 - d. insmod ehci_hcd



Apêndice

Objetivos detalhados para a prova 201

Primeira prova para a certificação LPI nível 2.



Tópico 201: O Kernel Linux

201.1 Componentes do Kernel

Peso 2

O candidato deve ser capaz de operar componentes do kernel necessários a um hardware específico, drivers de hardware, recursos e necessidades do sistema. Este objetivo contempla a implementação de diferentes tipos de imagens de kernel, identificação de kernels e patches estáveis e de testes e a operação dos módulos do kernel.

Conhecimentos chave

- Documentação do kernel 2.6.x

Lista parcial dos arquivos, termos e ferramentas utilizados

- /usr/src/linux
- /usr/src/linux/Documentation
- zImage
- bzImage

201.2 Compilando um kernel

Peso 2

O candidato deve ser capaz de configurar adequadamente um kernel de modo a incluir ou excluir um recurso específico, conforme necessário. Este objetivo inclui compilar e recompilar o kernel Linux quando necessário, atualizando e marcando as mudanças no kernel novo, além de criar uma imagem initrd e instalar novos kernels.

Conhecimentos chave

- /usr/src/linux
- Arquivos de configuração do GRUB
- Alvos do make para o kernel 2.6.x

Lista parcial dos arquivos, termos e ferramentas utilizados

- mkinitrd
- mkinitramfs
- make
- Alvos do make (config, xconfig, menuconfig, oldconfig, mrproper, zImage, bzImage, modules, modules_install)

201.3 Aplicar um patch ao kernel**Peso 1**

O candidato deve ser capaz de aplicar um patch ao kernel para incluir suporte a um novo hardware. Este objetivo inclui ser capaz de remover patches previamente aplicados ao kernel.

Conhecimentos chave

- Arquivos Makefile do kernel

Lista parcial dos arquivos, termos e ferramentas utilizados

- patch
- gzip
- bzip2

201.4 Personalizar, compilar e instalar um kernel personalizado e seus módulos**Peso 2**

O candidato deve ser capaz de personalizar, compilar e instalar um kernel 2.6 para necessidades específicas do sistema, seja aplicando um patch, compilando ou editando sua configuração. Este objetivo inclui ser capaz de identificar as exigências para uma compilação de kernel, assim como configurar e compilar seus módulos.

Conhecimentos chave

- Personalizar a configuração atual do kernel;
- Compilar um novo kernel e os módulos apropriados;
- Instalar um novo kernel e qualquer módulo;
- Assegurar que o gerenciador de boot conhece a localização do novo kernel e dos arquivos relacionados;
- /usr/src/linux/;
- Arquivos de configuração dos módulos;

- Lista parcial dos arquivos, termos e ferramentas utilizados;
- patch;
- make;
- Ferramentas de módulos;
- /usr/src/linux/*;
- /usr/src/linux/.config;
- /lib/modules/versão-do-kernel/*;
- /boot/*;
- Alvos do make: all, config, menuconfig, xconfig, gconfig, oldconfig, modules, install. modules_install, depmod, rpm-pkg, binrpm-pkg, deb-pkg.

201.5 Controlar/investigar o kernel e seus módulos durante sua execução

Peso 3

O candidato deve ser capaz de controlar e investigar um kernel 2.6.x e seus módulos disponíveis.

Conhecimentos chave

- Usar comandos para obter informações sobre o kernel em execução e seus módulos;
- Carregar e descarregar módulos manualmente;
- Identificar quando um módulo pode ser descarregado;
- Identificar quais opções um módulo aceita;
- Configurar o sistema para carregar um módulo por um nome diferente de seu nome de arquivo.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /lib/modules/versão-do-kernel/modules.dep
- Arquivos de configuração de módulos em /etc
- /proc/sys/kernel/
- depmod
- insmod
- lsmod
- rmmod
- modinfo
- modprobe
- uname



Tópico 202: Início do sistema

202.1 Personalizar o início do sistema e o processo de boot

Peso 4

O candidato deve ser capaz de analisar e alterar o comportamento dos serviços do sistema em qualquer nível de execução. É exigido sólido conhecimento sobre a estrutura do init e do processo de boot. Este objetivo inclui a interação com os níveis de execução.

Conhecimentos chave

- Linux Standard Base Specification (LSB)

Lista parcial dos arquivos, termos e ferramentas utilizados

- /etc/inittab
- /etc/init.d/
- /etc/rc.d/
- chkconfig
- update-rc.d

202.2 Recuperação do sistema

Peso 4

O candidato deve ser capaz de operar adequadamente um sistema Linux durante tanto o processo de boot quanto o modo de recuperação. Este objetivo inclui a utilização do init e as opções do kernel relacionadas a ele.

Conhecimentos chave

- inittab
- GRUB
- shell do grub

Lista parcial dos arquivos, termos e ferramentas utilizados

- init
- mount
- fsck
- telinit



Tópico 203: Sistemas de arquivos e Dispositivos

203.1 Trabalhando com o sistema de arquivos Linux

Peso 4

O candidato deve ser capaz de navegar e configurar um por um, sistema de arquivos Linux padrão. Este objetivo inclui saber como configurar e montar diversos tipos de sistemas de arquivos.

Conhecimentos chave

- Conceito de configuração do fstab;
- Ferramentas de manipulação de partições e arquivos SWAP;
- Utilização de UUIDs.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /etc/fstab
- /etc/mtab
- /proc/mounts
- mount e umount
- sync
- swapon
- swapoff

203.2 Manutenção de sistemas de arquivos Linux

Peso 3

O candidato deve ser capaz de fazer a manutenção adequada de um sistema de arquivos Linux usando as ferramentas de sistema. Este objetivo inclui a manipulação de sistemas de arquivos padrão.

Conhecimentos chave

- Ferramentas de manipulação de ext2 e ext3;
- Ferramentas de manipulação reiserfs V3;
- Ferramentas de manipulação de xfs.

Lista parcial dos arquivos, termos e ferramentas utilizados

- fsck (fsck.*)
- badblocks

- mkfs (mkfs.*)
- dumpe2fs
- debugfs, debugreiserfs
- tune2fs, reiserfstune
- mkswap
- xfs_info, xfs_check e xfs_repair

203.3 Criando e configurando opções de sistemas de arquivos

Peso 2

O candidato deve ser capaz de configurar a montagem automática de sistemas de arquivos utilizando o AutoFS. Este objetivo inclui a configuração de montagem automática para sistemas de arquivos em dispositivos locais e na rede. Também inclui a criação de sistemas de arquivos para dispositivos como CD-ROMs.

Conhecimentos chave

- Arquivos de configuração do autofs;
- Ferramentas UDF e ISO9660;
- Conhecimento sobre sistemas de arquivo de CD-ROM (UDF, ISO9660, HFS);
- Conhecimento sobre extensões de sistemas de arquivos de CD-ROM (Joliet, Rock Ridge, El Torito).

Lista parcial dos arquivos, termos e ferramentas utilizados

- /etc/auto.master
- /etc/auto.[dir]
- mkisofs
- dd
- mke2fs

203.4 Gerenciamento de dispositivos udev

Peso 1

O candidato deve entender como o udev identifica e controla dispositivos. Este objetivo inclui a correção de regras do udev.

Conhecimentos chave

- Regras do udev;

- Interface do kernel.

Lista parcial dos arquivos, termos e ferramentas utilizados

- udevmonitor
- /etc/udev



Tópico 204: Administração avançada de dispositivos de armazenamento

204.1 Configuração de RAID

Peso 2

O candidato deve ser capaz de configurar e implementar um RAID por software. Este objetivo contempla a configuração de RAID 0, 1 e 5.

Conhecimentos chave

- Ferramentas e arquivos de configuração de RAID por software.

Lista parcial dos arquivos, termos e ferramentas utilizados

- mdadm.conf
- mdadm
- /prc/mdstat
- fdisk

204.2 Ajustar o acesso a dispositivos de armazenamento

Peso 1

O candidato deve ser capaz de configurar opções do kernel para diversos dispositivos. Este objetivo inclui saber como utilizar ferramentas de software para verificar e alterar configurações de discos rígidos.

Conhecimentos chave

- Ferramentas para configurar DMA para dispositivos IDE, incluindo ATAPI e SATA;
- Ferramentas para analisar e manipular recursos do sistema (como interrupções);
- Conhecimento sobre o sdparm e suas finalidades.

Lista parcial dos arquivos, termos e ferramentas utilizados

- hdparm
- sdparm
- tune2fs
- sysctl
- /dev/hd* e /dev/sd*

204.3 Gerenciamento de volumes lógicos (LVM)**Peso 3**

O candidato deve ser capaz de criar e apagar volumes lógicos, grupos de volumes e volumes físicos. Este objetivo inclui saber como realizar snapshots e redimensionar volumes lógicos.

Conhecimentos chave

- Ferramentas do pacote LVM;
- Redimensionar, renomear, criar e apagar volumes lógicos, grupos de volumes e volumes físicos;
- Criação e manutenção de snapshots;
- Ativação de grupos de volumes.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /sbin/pv*
- /sbin/lv*
- /sbin/vg*
- mount
- /dev/mapper/

Tópico 205: Configuração de rede**205.1 Configuração básica de rede****Peso 3**

O candidato deve ser capaz de configurar um dispositivo de rede para conexão local, com ou sem fio, ou para uma WAN (Wide Area Network). Este objetivo inclui a comunicação entre diversas sub-redes dentro de uma rede.

Conhecimentos chave

- Ferramentas para configuração e manipulação de interfaces de rede ethernet;
- Configuração de redes sem fio.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /sbin/route
- /sbin/ifconfig
- /sbin/ip
- /usr/sbin/arp
- /sbin/iwconfig
- /sbin/iwlist

205.2 Configuração avançada de rede e resolução de problemas

Peso 4

O candidato deve ser capaz de configurar um dispositivo de rede para aceitar diversos tipos de sistemas de autenticação. Este objetivo inclui a configuração de um dispositivo de rede em trânsito, configurar um cliente VPN e solucionar problemas de comunicação.

Conhecimentos chave

- Ferramentas para manipular tabelas de rotas;
- Ferramentas para manipular e configurar interfaces de rede ethernet;
- Ferramentas para analisar o estado dos dispositivos de rede;
- Ferramentas para monitorar e analisar o tráfego TCP/IP;
- OpenVPN.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /sbin/route
- /sbin/ifconfig
- /bin/netstat
- /bin/ping
- /usr/sbin/arp
- /usr/sbin/tcpdump
- /usr/sbin/lsof
- /usr/bin/nc
- /sbin/ip
- /etc/openvpn/*

- openvpn
- nmap
- wireshark

205.3 Solução de problemas de rede

Peso 5

O candidato deve ser capaz de identificar e corrigir problemas comuns de configuração de redes e conhecer a localização de arquivos e comandos básicos de configuração.

Conhecimentos chave

- Localização e conteúdo dos arquivos de restrição de acesso;
- Ferramentas para configurar e manipular interfaces de rede ethernet;
- Ferramentas para manipular tabelas de rotas;
- Ferramentas para exibir o estado da rede;
- Ferramentas para obter informações sobre a configuração da rede;
- Maneiras de obter informações a respeito dos dispositivos reconhecidos e utilizados;
- Arquivos de inicialização do sistema e seus conteúdos (processo de inicialização SysV).

Lista parcial dos arquivos, termos e ferramentas utilizados

- /sbin/ifconfig
- /sbin/route
- /bin/netstat
- /etc/network || /etc/sysconfig/network-scripts/
- Arquivos de log como /var/log/syslog e /var/log/messages
- /bin/ping
- /etc/resolv.conf
- /etc/hosts
- /etc/hosts.allow e /etc/hosts.deny
- /etc/hostname | /etc/HOSTNAME
- /bin/hostname
- /usr/sbin/traceroute
- /usr/bin/dig
- /bin/dmesg
- /usr/bin/host

205.4 Informar usuários sobre questões relativas ao sistema

Peso 1

O candidato deve ser capaz de informar os usuários sobre os acontecimentos que atualmente afetam o sistema.

Conhecimentos chave

- Automatizar a comunicação com os usuários usando as mensagens de logon;
- Informar os usuário no sistema sobre manutenções no sistema.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /etc/issue
- /etc/issue.net
- /etc/motd
- wall
- /sbin/shutdown



Tópico 206: Manutenção do Sistema

206.1 Compilar e instalar programas a partir do código fonte

Peso 4

O candidato deve ser capaz de compilar e instalar programas a partir do código fonte, o que significa saber como descompactar um arquivo de código fonte.

Conhecimentos chave

- Descompactar arquivos de código usando as ferramentas comuns de compressão;
- Entender o básico de como utilizar o make para compilar programas;
- Usar parâmetros em um script configure;
- Saber a localização padrão dos códigos fonte.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /usr/src/
- gunzip
- gzip
- bzip2

- tar
- configure
- make
- uname
- install

206.2 Operações de Backup

Peso 3

O candidato deve ser capaz de usar as ferramentas do sistema para criar cópias de segurança dos dados importantes.

Conhecimentos chave

- Conhecimento sobre quais diretórios devem ser incluídos num backup;
- Conhecer as soluções de backup em rede como Amanda, Bacula e BackupPC;
- Conhecer os prós e contras de fitas, CDR, discos ou outros meios de backup;
- Realizar backups manuais e parciais;
- Verificar a integridades de arquivos de backup;
- Restaurar backups parcialmente e completamente.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /bin/sh
- cpio
- tar
- /dev/st* e /dev/nst*
- mt
- rsync

Tópico 207: Domain Name Server



207.1 Configuração básica de um servidor DNS

Peso 2

O candidato deve ser capaz de configurar o BIND para funcionar como um servidor DNS apenas de cache. Este objetivo contempla saber como converter configurações antigas do BIND para o novo formato, administrar um servidor em execução e configurar o registro de log.

Conhecimentos chave

- Arquivos de configuração, termos e ferramentas do BIND 9.x;
- Definição de localização dos arquivos de zona do BIND nos arquivos de configuração do BIND;
- Recarregar configurações e arquivos de zona alterados.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /etc/named.conf
- /var/named/*
- /usr/sbin/rndc
- kill

207.2 Criar e manter zonas de DNS

Peso 2

O candidato deve ser capaz de criar um arquivo de zona para redirecionamento, zona reversa ou servidor raiz. Este objetivo inclui definir apropriadamente valores de registros, incluindo hosts em zonas e adicionando as zonas ao DNS. Também deve ser capaz de delegar uma zona a outro servidor DNS.

Conhecimentos chave

- Arquivos de configuração, termos e ferramentas do BIND 9;
- Ferramentas para solicitar informações de um servidor DNS;
- Formato, conteúdo e localização dos arquivos de zona do BIND;
- Métodos para incluir um novo host à zona, incluindo zonas reversas.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /var/named/*
- Sintaxe de arquivos de zona
- Formato dos registros
- dig
- nslookup
- host

207.3 Segurança de servidor DNS

Peso 2

O candidato deve ser capaz de configurar um servidor DNS para operar sob um usuário diferente do root e dentro de uma ambiente chroot. Este objetivo inclui realizar a troca segura de dados entre servidores DNS.

Conhecimentos chave

- Arquivos de configuração do BIND 9;
- Configuração do BIND para operar num ambiente chroot;
- Dividir a configuração do BIND usando redirecionamentos.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /etc/named.conf
- /etc/passwd
- DNSSEC
- dnssec-keygen

Objetivos detalhados para a prova 202

Primeira prova para a certificação LPI nível 2.



Tópico 208: Serviços Web

208.1 Implementar um servidor Web

Peso 3

O candidato deve ser capaz de instalar e configurar um servidor web. Este objetivo inclui monitorar a carga e performance do servidor, restringindo o acesso, suporte aos módulos de linguagens de script e configuração de restrição de acesso por cliente. Inclui saber como configurar o servidor para restringir o uso de recursos.

Conhecimentos chave

- Arquivos de configuração, termos e ferramentas do Apache 2.x;
- Conteúdo e configuração dos arquivos de log do Apache;
- Arquivos e métodos de restrição de acesso;
- Configuração do PHP e mod_perl;
- Arquivos e ferramentas para autenticação por usuário;
- Configuração de requisições máximas, mínimo e máximo e de servidores e clientes.

Lista parcial dos arquivos, termos e ferramentas utilizados

- Logs de erro e logs de acesso
- .htaccess
- httpd.conf
- mod_auth
- httpasswd
- htgroup
- apache2ctl
- httpd

208.2 Manutenção de servidor web

Peso 2

O candidato deve ser capaz de configurar um servidor web para usar domínios virtuais, Secure Sockets Layer (SSL) e personalizar o acesso a arquivos.

Conhecimentos chave

- Arquivos de configuração e ferramentas do SSL;
- Manuseio de certificados SSL;
- Implementação de domínios virtuais no Apache 2.x;
- Uso de redirecionamento nos arquivos de configuração do Apache para personalizar o acesso a arquivos.

Lista parcial dos arquivos, termos e ferramentas utilizados

- Arquivos de configuração do Apache
- /etc/ssl/*
- openssl

208.3 Implementação de um servidor proxy**Peso 1**

O candidato deve ser capaz de instalar e configurar um servidor de proxy, incluindo políticas de acesso, autenticação e consumo de recursos.

Conhecimentos chave

- Arquivos de configuração, termos e ferramentas do Squid 2.x;
- Métodos de restrição de acesso;
- Métodos de autenticação de usuários;
- Formato e conteúdo de ACL nos arquivos de configuração do Squid.

Lista parcial dos arquivos, termos e ferramentas utilizados

- squid.conf
- acl
- http_access

Tópico 209: Compartilhamento de arquivos**209.1 Configuração de servidor SAMBA****Peso 4**

O candidato deve ser capaz de configurar um servidor Samba para diversos clientes. Este objetivo inclui configurá-lo para login dos clientes, configurar o grupo de trabalho do servidor e definição das impressoras e diretórios compartilhados. Também

é necessário configurar um cliente Linux para usar um servidor Samba e testar e solucionar problemas de instalação.

Conhecimentos chave

- Documentação do Samba 3;
- Arquivos de configuração do Samba;
- Ferramentas do Samba;
- Montar compartilhamentos Samba no Linux;
- Daemons do Samba;
- Mapear nomes de usuário do Windows para nomes de usuário do Linux;
- Segurança em nível de usuário e em nível de compartilhamento.

Lista parcial dos arquivos, termos e ferramentas utilizados

- `smbd`, `nmbd`
- `smbstatus`, `testparm`, `smbpasswd`, `nmblookup`
- `smbclient`
- `net`
- `/etc/smb/*`
- `/var/log/samba/`

209.2 Configuração de servidor NFS

Peso 4

O candidato deve ser capaz exportar um sistema de arquivos usando o NFS. Este objetivo inclui restringir o acesso, montar um sistema de arquivos NFS no cliente e segurança NFS.

Conhecimentos chave

- Arquivos de configuração NFS;
- Ferramentas do NFS;
- Restrição de acesso para hosts ou sub-redes específicos;
- Opções de montagem no servidor e no cliente;
- `tcpwrappers`.

Lista parcial dos arquivos, termos e ferramentas utilizados

- `/etc/exports`
- `exportfs`
- `showmount`

- nfsstat
- /proc/mounts
- /etc/fstab
- rpcinfo
- mntd
- portmapper

Tópico 210: Administração dos clientes de rede



210.1 Configuração DHCP

Peso 2

O candidato deve ser capaz de configurar um servidor DHCP. Este objetivo inclui configurar opções padrão e por cliente, incluindo hosts estáticos e hosts BOOTP. Também inclui configurar um agente de redirecionamento DHCP e manutenção de um servidor DHCP.

Conhecimentos chave

- Arquivos de configuração, termos e ferramentas DHCP;
- Configuração de sub-rede e faixas atribuídas dinamicamente.

Lista parcial dos arquivos, termos e ferramentas utilizados

- dhcpd.conf
- dhcpd.leases
- /var/log/daemon.log e /var/log/messages
- arp
- dhcpd

210.2 Autenticação por PAM

Peso 3

O candidato deve ser capaz de configurar o PAM para trabalhar autenticação com os diferentes métodos disponíveis.

Conhecimentos chave

- Arquivos de configuração, termos e ferramentas do PAM;
- Senhas passwd e shadow.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /etc/pam.d
- pam.conf
- nsswitch.conf
- pam_unix, pam_cracklib, pam_limits, pam_listfile

210.3 Uso de cliente LDAP

Peso 2

O candidato deve ser capaz de realizar consultas e atualizações em um servidor LDAP. Inclui também importar e incluir itens, assim como incluir e administrar usuários.

Conhecimentos chave

- Ferramentas do LDAP para administração de dados e consultas;
- Alterar senhas de usuários;
- Consultar o diretório LDAP.

Lista parcial dos arquivos, termos e ferramentas utilizados

- ldapsearch
- ldappasswd
- ldapadd
- ldapdelete



Tópico 211: Serviços de email

211.1 Utilização de servidores de email

Peso 3

O candidato deve ser capaz de administrar um servidor de email, incluindo a configuração de aliases de email, cotas e domínios de email virtuais, que inclui saber configurar redirecionamento interno de emails.

Conhecimentos chave

- Arquivos de configuração do Postfix;
- Conhecimento básico do protocolo SMTP, sendmail e Exim.

Lista parcial dos arquivos, termos e ferramentas utilizados

- Arquivos de configuração e comandos do Postfix;
- Configuração básica do sendmail;
- /etc/aliases;
- /etc/mail/*;
- /etc/postfix/*;
- Comandos da camada de emulação do sendmail;
- /var/spool/mail;
- Arquivos de log relacionados a email em /var/log/.

211.2 Administração da entrega local de email**Peso 2**

O candidato deve ser capaz de implementar programas de gerenciamento de email cliente, classificar e monitorar a entrada de email.

Conhecimentos chave

- Arquivos de configuração e ferramentas do procmail
- Uso do procmail no cliente e no servidor

Lista parcial dos arquivos, termos e ferramentas utilizados

- ~/.procmail
- /etc/procmailrc
- procmail
- Formatos mbox e Maildir

211.3 Administrar entrega de email remoto**Peso 2**

O candidato deve ser capaz de instalar e configurar serviços POP e IMAP

Conhecimentos chave

- Configuração Courier IMAP e Courier POP;
- Configuração Dovecot.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /etc/courier/*
- dovecot.conf



Tópico 212: Segurança do Sistema

212.1 Configuração de roteador

Peso 3

O candidato deve ser capaz de configurar um sistema para realizar tradução de endereço de rede (NAT, Mascaramento de IP) e determinar sua importância na proteção da rede. Este objetivo inclui configurar redirecionamento de portas, regras de filtros e prevenção de ataques.

Conhecimentos chave

- Arquivos de configuração e ferramentas do iptables;
- Ferramentas e comandos para administrar tabelas de rota;
- Faixas de endereços privados;
- Redirecionamento de portas e de endereços IP;
- Exibir e criar filtros e regras que aceitem ou bloqueiem datagramas baseados na origem, destino, protocolo, porta ou endereço;
- Salvar e recuperar configurações de filtragem.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /proc/sys/net/ipv4
- /etc/services
- iptables
- routed

212.2 Segurança de servidores FTP

Peso 2

O candidato deve ser capaz de configurar um servidor FTP para download e upload anônimo. Este objetivo inclui as precauções necessárias quando upload anônimo é permitido e com as configurações de acesso de usuários.

Conhecimentos chave

- Arquivos de configuração e ferramentas do Pure-FTPd e vsftpd;
- Conhecimento do ProFTPD;
- Entendimento de conexões FTP passivas e ativas.

Lista parcial dos arquivos, termos e ferramentas utilizados

- vsftpd.conf
- Opções de linha de comando importantes do Pure-FTPd

212.3 Shell seguro (SSH)**Peso 4**

O candidato deve ser capaz de configurar e tornar seguro um daemon SSH. Este objetivo inclui administrar chaves e configurar o SSH para os usuários. Além disso, deve ser capaz de redirecionar um outro protocolo através do SSH e controlar o login SSH.

Conhecimentos chave

- Arquivos de configuração e ferramentas do OpenSSH;
- Restrições de login para o superusuário e para usuários normais;
- Controlar e usar chaves de cliente e servidor para realizar login com e sem senha;
- Uso do XWindow e outros protocolos através de túneis SSH;
- Configuração do ssh-agent;
- Uso de várias conexões de diversos hosts para garantir a manutenção da conexão em caso de perda de conexão com um host remoto.

Lista parcial dos arquivos, termos e ferramentas utilizados

- ssh
- sshd
- /etc/ssh/sshd_config
- Arquivos de chaves públicas e privadas
- ~/.ssh/authorized_keys
- PermitRootLogin, PubKeyAuthentication, AllowUsers, PasswordAuthentication. Protocol

212.4 TCP Wrapper**Peso 1**

O candidato deve ser capaz de configurar o recurso TCP Wrapper para liberar conexões para servidores específicos apenas se vindas de hosts ou sub-redes específicos.

Conhecimentos chave

- Arquivos de configuração e ferramentas de TCP Wrapper;
- Arquivos de configuração e ferramentas do inetd.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /etc/inetd.conf
- /etc/hosts.allow
- /etc/hosts.deny
- libwrap
- tcpd

212.5 Tarefas de segurança

Peso 3

O candidato deve ser capaz de buscar alertas de segurança de diversas fontes, instalar, configurar e operar sistemas de detecção de intrusos, aplicar patches de segurança e corrigir erros.

Conhecimentos chave

- Ferramentas para varrer e testar portas em um servidor;
- Endereços e organizações que divulgam alertas de segurança, como Bugtraq, CERT, CIAC ou outras fontes;
- Ferramentas de implementação de sistemas de detecção de intrusos (IDS);
- Conhecimento de OpenVAS.

Lista parcial dos arquivos, termos e ferramentas utilizados

- telnet
- nmap
- snort
- fail2ban
- nc
- iptables

Tópico 213: Solução de problemas

213.1 Identificar estágios de boot e consertar carregadores de boot

Peso 4

O candidato deve ser capaz de determinar as causas de erros ocorridos durante o carregamento e uso dos carregadores de boot. Os carregadores de boot em questão são o GRUB e o LILO.

Conhecimentos chave

- Início da operação do carregador de boot e liberação para o kernel;
- Carregamento do kernel;
- Inicialização e configuração do hardware;
- Inicialização e configuração dos serviços/daemons;
- Conhecer as diferentes localizações de instalação dos carregadores de boot no disco ou em mídias removíveis;
- Sobrepor opções padrão do carregador de boot e usar o shell do carregador de boot.

Lista parcial dos arquivos, termos e ferramentas utilizados

- Conteúdos de /boot/ e /boot/grub/
- GRUB
- grub-install
- initrd, initramfs
- Master Boot Record
- /etc/init.d
- lilo
- /etc/lilo.conf

213.2 Solução geral de problemas**Peso 5**

O candidato deve ser capaz de identificar e corrigir problemas comuns de boot e execução.

Conhecimentos chave

- Sistema de arquivos /proc;
- Arquivos de log diversos dos daemons e do sistema;
- Conteúdo de /, /boot e /lib/modules;
- Informações exibidas durante o boot;
- Entradas do syslog correspondentes ao kernel nos logs do sistema;
- Ferramentas para analisar as informações do hardware utilizado;
- Ferramentas para rastrear programas e suas chamadas de sistemas e bibliotecas.

Lista parcial dos arquivos, termos e ferramentas utilizados

- dmesg
- /sbin/lspci

- /usr/bin/lsdev
- /sbin/lsmmod
- /sbin/modprobe
- /sbin/insmod
- /bin/uname
- strace
- strings
- ltrace
- lsof
- lsusb

213.3 Problemas em recursos do sistema

Peso 5

O candidato deve ser capaz de identificar, diagnosticar e consertar problemas no sistema local usando programas da linha de comando.

Conhecimentos chave

- Principais variáveis do sistema
- /etc/profile && /etc/profile.d/
- /etc/init.d/
- /etc/rc.*
- /etc/sysctl.conf
- /etc/bashrc
- /etc/ld.so.conf
- Outros arquivos relacionados à configuração global do shell
- Qualquer editor padrão
- Ferramentas e comandos padrão para edição dos arquivos e variáveis mencionados

Lista parcial dos arquivos, termos e ferramentas utilizados

- /bin/ln
- /bin/rm
- /sbin/ldconfig
- /sbin/sysctl

213.4 Problemas em configurações de ambiente

Peso 5

O candidato deve ser capaz de identificar problemas comuns na configuração do sistema e do ambiente e maneiras comuns de corrigi-los.

Conhecimentos chave

- Variáveis principais do sistema;
- Arquivos de configuração do init;
- Processo de inicialização do init;
- Arquivos de configuração do cron;
- Processo de login;
- Arquivos de armazenamento das senhas e contas de usuários;
- Determinar associações de usuários e grupos;
- Arquivos de configuração SHELL do bash;
- Análise de quais processos ou daemons estão ativos.

Lista parcial dos arquivos, termos e ferramentas utilizados

- /etc/inittab
- /etc/rc.local
- /etc/rc.boot
- /var/spool/cron/crontabs/
- Arquivos de configuração padrão do shell em /etc/
- /etc/login.defs
- /etc/syslog.conf
- /etc/passwd
- /etc/shadow
- /etc/group
- /sbin/init
- /usr/sbin/cron
- /usr/bin/crontab