Revisão do Tópico 201 – Kernel Linux

201.1 – Componentes do Kernel

Conceitos

- Linux é o kernel, o núcleo do sistema operacional
- Tipos de Kernel:
  - Monolítico: Arquitetura em que toda a implementação do kernel é feita dentro de um processo único, através de uma imagem única.
  - Micro-Kernel: Nessa arquitetura o kernel é um controlador central que degela funções a outros subprocessos, ou servers.
  - O kernel Linux segue a arquitetura Monolítica, porém de forma modular. As implementações principais de gerenciamento são feitas na imagem do kernel, porém ele também trabalha em conjunto com módulos, que implementam recursos específicos e são carregados em memória apenas quando necessário.

As Versões do Kernel

- Versões disponibilizadas pelo site http://www.kernel.org
- Atualmente a nomenclatura da versão do kernel segue o formato A.B.C, aonde:
  - A é a versão principal do kernel
  - B é a release principal, contendo novas implementações
  - C é a release secundária, ou minor release, indicando principalmente a correções
- Durante a versão 2.6 o padrão possuía 4 números, A.B.C.D, e o 2.6 (A.B) era considerado a versão estável

Os Arquivos e a Localização do Kernel e da Imagem

- Fontes do Kernel: /usr/src/
- Documentação do kernel: /usr/src/linux/Documentation/
- Imagem do Kernel: /boot/ (arquivos vmlinuz*)
- Módulos: /lib/modules/<kernel>/ (arquivos *.ko)
- O arquivo tar com as **fontes** é compactado com o algoritmo **XZ**
- É indicado criar um link simbólico chamado “linux” apontando para a fonte da kernel atual em /usr/src
  - ln -s linux-A.B.C linux
- Há dois tipos de imagem de kernel: zImage ou bzImage
  - zImage possui o tamanho máximo de 512 Kb e é alocada na memória baixa (low memory)

- bzImage (big zImage) não tem limite de tamanho e é alocada na memória alta (high memory)
- Ambas **imagens** são compactadas com o **gzip**.

201.2 – Compilando um kernel

O processo é composto basicamente de 3 fases:

- Configuração
- Compilação
- Instalação

Configuração

O processo de configuração tem por objetivo gerar o **arquivo .config** que vai ser usado durante o processo de compilação. Esse arquivo indica os recursos que devem ser instalados, e quais devem ser construídos dentro do kernel ou implementados via módulo.

As opções disponíveis são:

- make config : são feitas perguntas sobre todos os recursos
- make oldconfig : O .config atual será lido e caso hajam opções não informadas, serão perguntadas.
- make menuconfig : Interface de menus através da lib ncurses
- make xconfig : Interface gráfica através da biblioteca QT
- make gconfig : Interface gráfica através da biblioteca GTK

Compilação

Nesta fase vamos compilar o kernel e os módulos associados, gerando o arquivo de imagem bzImage que será usado durante o boot pelo GRUB.

Os comandos utilizados são:

- make bzImage : Compilar o código fonte e gerar o arquivo de imagem bzImage em /usr/src/linux/arch/x86/boot/bzImage
- make modules_install : Compilar os módulos

Instalação

Para instalar os módulos do novo kernel utilizamos o comando “make modules_install”. O comando criará um novo diretório /lib/modules/<versao-release-kernel> com os módulos da versão atual e os arquivos modules.*, criados pela execução do depmod.

A instalação da nova imagem do kernel pode ser feita de 2 formas:

- Pelo comando “make install” que automaticamente copiará o bzImage para o /boot, e fará o mesmo com os arquivos .config e System.map. Além disso fará a geração do arquivo initrd. E por final executará o update-grub para atualizar as configurações do GRUB.

- Manualmente: Simplesmente copiando o arquivo bzImage para o diretório /boot e renomeando-o. Utilizando o mkinitramfs/mkinitd/update-initramfs para gerar o novo arquivo Initial Ramdisk. E, ao final, executando o update-grub para atualizar o GRUB.

Limpeza

Durante o processo de compilação e instalação de um novo kernel, podem ocorrer erros que exijam que todo o processo seja reiniciado. Nesses casos é importante o uso de comandos que removam os arquivos criados durante o processo. Além disso, depois que o kernel foi compilado e instalado, os objetos criados na árvore /usr/src não são mais necessários e podem ser limpos para liberar espaço. Os principais comandos são:

- **make clean :** Remover os objetos criados durante o processo de compilação, por exemplo arquivos com a extensão .o e .ko.
- **make mrpropper :** Além do que já é limpo pelo “make clean” remove também o arquivo de configuração .config.

Geração do Initial Ramdisk

O Initramfs (ou Initrd) é um arquivo de imagem que será utilizado durante o processo de boot. Essa imagem será montada na memória RAM e utilizada como uma partição raiz (/) temporária, que conterá por exemplo os módulos que precisam ser usados pelo kernel, por exemplo um controlador RAID, de uma interface de rede e etc.

Em sistemas padrão Debian, são utilizados os comandos:

- mkinitramfs
  - # mkinitramfs -o /boot/iniramfs-x.y.z x.y.z
- update-initramfs
  - update-initrafms -u (atualizar um arquivo já existente)
  - update-initramfs -c (criar um novo arquivo)

Em sistemas padrão RedHat, são utilizados os comandos

- mkinitrd (atualmente chama o dracut)
  - # mkinitrd -c -o /boot/initrd.img-x.y.z -k x.y.z
- dracut

- # dracut
- # dracut --force (caso já exista um arquivo)
- # dracut novo-arquivo.img (gerar um novo arquivo)

Empacotamento do kernel

make rpm-pkg : gera um conjunto de pacotes .rpm
make birrpm-pkg : gera apenas o pacote rpm com a imagem compilada
make deb-pkg : gera um conjunto de pacotes .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

DKMS (Dynamic Kernel Module Support)

O DKMS é um framework que permite integrar módulos cujas fontes ficam fora da árvore de fontes do kernel.

A principal função é que esses módulos sejam automaticamente reconstruídos sempre que uma nova versão do kernel for instalada.

Para fazer essa integração, nas fontes do módulo deve ser configurado o arquivo dkms.conf.

201.3 – Gerenciando e Resolvendo Problemas no Kernel em Tempo de Execução

Comando uname

Função: Exibe informações referentes ao kernel em uso.

Uso e Principais Opções:

uname -r : Mostra a release do kernel em uso
uname -a : Mostra todas as informações do kernel em uso

Visualização e Alteração de Parâmetros do Kernel

Pode ser feito diretamente nos arquivos do /proc. Exemplo:

cat /proc/sys/fs/file-max
echo 100000 > /proc/sys/fs/file-max

Ou pelo uso do comando sysctl:

sysctl fs.file-max
sysctl -w fs.file-max=100000

Para que a configuração seja permanente, ele deve ser feita nos seguintes arquivos ou diretórios:

- /etc/sysctl.conf
- /etc/sysctl.d/

Os comandos a seguir também são utilizados para obter informações do sistema, através das referências do /proc:

- lspci – Dispositivos conectados ao barramento PCI
  - # lspci -s 00:11 -v
- lsusb – Dispositivos conectados ao barramento USB
  - # lsusb -d xxx:yyy -v
  - # lsusb -s 00:11 -v
- lsdev – Informações dos hardwares utilizados

O **dmesg** é o comando utilizado para acessar e controlar as informações do “**kernel ring buffer”**, que entre outros dados, contém as informações do boot do sistema.

Módulos

Os módulos utilizados pelo kernel ficam em /lib/modules/`uname -r`.

Principais comandos relacionados ao gerenciamento de módulos:

- lsmod : lista os módulos carregados
- modinfo : Exibe informações detalhadas de um módulo
  - # modinfo psmouse
  - # modinfo -p psmouse (exibe apenas os parâmetros do módulo)
- insmod : carrega um módulo mediante a indicação do nome do arquivo .ko do módulo
- rmmod : descarrega um módulo
- modprobe : pode ser utilizado para carregar e descarregar um módulo a partir do seu nome. O modprobe utiliza as informações dos arquivos /lib/modules/`uname -r`/modules.* para relacionar os nomes aos arquivos, identificar dependências e etc.
  - # modprobe psmouse (carrega o módulo)
  - # modprobe -r psmouse (descarrega o módulo)
  - # modprobe modulo parametro=valor (define um parâmetro de um módulo)

Arquivos de Configuração:

- /etc/modules.conf : Configurações dos módulos
- /etc/modprobe.d/* : Configurações dos módulos
- /etc/modules : Módulos que devem ser carregados no boot do sistema
- /etc/module-load.d/* : Módulos que devem ser carregados no boot do sistema

Principais opções que podem ser utilizadas nos arquivos de configuração:

- options : Especificar uma opção para um módulo
  - # options r8169 debug=16
- alias : Definir um outro nome para um módulo
  - # alias nome-alias nome-modulo
- install : Definir ações que serão executadas para o carregamento do módulo.
  - # install moduloX /sbin/modprobe --ignore-install moduloX && /bin/touch /tmp/moduloX.tmp
- remove : Definir ações que serão executadas para o descarregamento de um módulo
  - # remove moduloX /sbin/rmmod moduloX && /bin/rm /tmp/moduloX.tmp
- * Quando o modprobe é usado nas opções install e remove, devem ser usados os parâmetros --ignore-install e --ignore-remove, para que o modprobe não entre em um loop.

Udev (Dynamic Device Management)

Processo responsável por gerenciar em tempo real as referências em /dev/ e /sys/. O processo roda em background ouvindo as notificações do kernel sobre novos dispositivos de hardware ou remoção deles.

Principais arquivos de configuração:

- /etc/udev/udev.conf : Arquivo de Configuração
- /etc/udev/rules.d/* : Arquivos com as Regras

Principais comandos utilizados:

- udevmonitor (ou)
- udevadm monitor