Tutorial

Compilação de um novo *Kernel* Linux

(independente de distribuição)

Procedimento genérico para compilação de um novo *Kernel*

A grande maioria dos usuários que possuem o Ubuntu instalado nunca compilou o *Kernel*. Isto quer dizer que a distribuição estará utilizando uma versão de *Kernel* específica – isto quer dizer modificada - para ela, com chances de ser uma versão LTS (*Long Term Suport*).

Os *Kernels* das distribuições (*distro kernels*) são defasados com relação à última versão estável do *Kernel* Linux, mantida em **kernel.org**. Por exemplo, no momento da escrita deste tutorial, o Ubuntu 14.04.1 utilizava o 3.13.0 enquanto a última versão estável era a 3.17.1.

Existem ainda outros dois tipos de *kernels* lançados: *stable kernels* e *release candidate kernels*.

O *Kernel* candidato a lançamento (*release candidate kernels*) cujo nome termina com a numeração -rcX, é o *Kernel* mais novo e avançado que Linus Torvalds mantém. Todas as características novas são incluídas nele durante a janela de duas semanas.

Kernels estáveis (*stable kernels*) são lançados a cada 2-3 meses quando Linus sente que todas as novas características estão estabilizadas. Um *Kernel* estável (como 3.17) será atualizado com correções de segurança e *bugs*, sendo que as novas versões terão as seguintes designações X.Y.Z como: 3.17.1, 3.17.2 e assim por diante.

Ferramentas necessárias

Para compilar o *Kernel* a partir dos fontes é necessário que várias ferramentas estejam instaladas.

No Ubuntu pode-se instalar tais ferramentas (provavelmente `gcc` e `make` já estejam instalados) executando o comando:

```
sudo apt-get install libncurses5-dev gcc make git exuberant-ctags
```

No Fedora, os comandos abaixo:

```
sudo yum install gcc make ctags ncurses-devel -y

ou

sudo yum groupinstall 'C Development Tools and Libraries'
```

Baixando os fontes da versão estável mais recente

Aqui optamos por baixar os fontes utilizando a ferramenta `wget` utilizando como alvo a última versão estável disponível em **kernel.org** (3.17.1, no momento da escrita deste tutorial). Mas também podemos baixar os fontes do repositório **git** oficial em **kernel.org**. Detalhes em [2].

```
wget https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.17.1.tar.xz
```

Uma dica aqui é visitar o site **kernel.org** no momento em que você estiver lendo este tutorial para baixar a versão mais recente. Para isso, vá até o site, clique com o botão da direita no *link* da versão *Latest*

Stable Kernel (geralmente, uma caisa amarela na página inicial), escolha a opção 'copiar link' e cole na frente do comando `wget`.

Descompactando e extraindo os arquivos

Uma vez concluído o download do pacote dos fontes, você deve descompactar e extrair os arquivos em local adequado. Adotaremos, neste tutorial, o diretório de usuário $HOME/linux. Utilizaremos privilégios de `root` apenas onde for necessário.

Faça a extração com o comando abaixo:

```
tar xvf  linux-3.17.1.tar.xz
```

O comando `tar` gera o diretório linux-3.17.1 onde os fontes do *Kernel* são descompactados.

Configuração do novo *Kernel*

Mude para o diretório recém criado pela extração dos arquivos.

```
cd ./linux-3.17.1
```

OPCIONAL

Se você quiser tomar como base o arquivo de configuração de sua instalação atual, faça uma cópia dele para o diretório atual e execute o comando seguinte, que irá lhe solicitar decisões de configuração relacionadas a novas características encontradas no novo *kernel*. Para cada solicitação responda com uma das opções [Y/n/m/?], significando, respectivamente: Yes, no, module, help. A resposta recomendada aparece com letra maiúscula (nesse exemplo: Y).

```
cp /boot/config-`uname -r`* .config
```

```
make oldconfig
```

Finalmente, começamos a configurar o novo *Kernel* executando uma interface gráfica em modo texto com o comando:

```
make menuconfig
```

Esta é uma interface bastante simples e de fácil navegação. Porém, com uma imensa quantidade de opções de configuração.

Você verá uma lista com várias opções. Nesta lista, para cada característica ou *driver*, o usuário pode responder [Y/n/m/?], significando, respectivamente: Yes, no, module, help. A resposta recomendada aparece com letra maiúscula.

Basicamente, uma boa otimização do *kernel* deve atualizar sua máquina para uma configuração de processador mais “alta” possível, e eliminar recursos indesejados. No caso das versões 386 do *Kernel*, há uma infinidade de processadores, procure um que se adapte exatamente ao que tem na sua máquina. Já no caso das versões x64, simplesmente você opta entre as plataformas AMD, Pentium 4 ou Core2/Xeon. É talvez a configuração mais importante, porque por padrão, o Linux normalmente tenta ser compatível com máquinas 486, abrindo mão de importantes recursos de arquitetura em nome da compatibilidade.

Quanto aos dispositivos que sua máquina possui e que o *Kernel* deverá suportar, é interessante que

você faça antes uma investigação detalhada com ajuda de alguns programas como: lsusb, lspci, lshw, etc. Assim, conhecendo exatamente quais dispositivos você possui fica mais fácil eliminar outros que vêm marcados no arquivo de configuração padrão e que seu computador não possui.

Enfim, configurar este arquivo é um assunto a parte e está fora do escopo deste tutorial. Para aprender mais sobre isso indicamos o livro, disponível online: *Linux Kernel in a Nutshell* [3]

Após fazer as alterações desejadas, volte ao menu principal e lembre de salvar (salve com padrão `.config`)

Compilando o *Kernel*

Agora, está tudo pronto para compilar o *Kernel*.

Para isso, o comando `make` é suficiente. No entanto, algumas opções desse comando são bastante úteis. Veja abaixo. Além disso, saiba que esta etapa consome considerável espeço em disco. Por isso, antes de executar o comando abaixo, certifique-se de que tenha livres no disco, pelo menos, 15 GB, para trabalhar com folga. Exemplo: após instalado o Ubuntu 14.04, o disco está com ocupação de 4,2 GB e depois de terminado o processo de compilação e instação desse tutorial, haviam 16,4 GB ocupados. O processo todo, até a reinicialização do sistema, pode demorar mais de uma hora.

```
make -j4 CONFIG_LOCALVERSION="tutorial"
```

A opção -jX ajuda a distribuir as tarefas de compilação pelos núcleos do processador, caso seja um processador *multicore*. O 'X' indica a quantidade núcleos do processador. A opção `CONFIG_LOCALVERSION` serve para anexar um rótulo ao nome do seu novo *Kernel*. Trata-se de uma das variáveis dentro do arquivo de configuração `.config` – esta também pode ser modificada por edição desse arquivo. Você pode até colocar a data da compilação nesse rótulo usando `CONFIG_LOCALVERSION="tutorial-$(date +%Y%M %d)"`, se assim desejar.

Uma vez terminada a compilação do *kernel*, deve-se complilar e instalar os módulos. O tempo que gasto para cumprir esta etapa depende da quantidade de módulos a serem compilados.

```
make modules

sudo make modules_install
```

Por fim, instale o novo *Kernel*. Este comando instala o *Kernel*, gera a imagem, copia os arquivos para o diretório `/boot` e atualiza o gerenciador de *boot*.

```
sudo make install
```

Assim, com tudo pronto, podemos reiniciar o sistema e escolher o novo *Kernel*.

```
sudo reboot
```

Bibliografia

1. http://kernelnewbies.org/KernelDevProcess
2. http://kernelnewbies.org/KernelBuild
3. http://www.kroah.com/lkn/
4. https://www.banym.de/linux/fedora/custom-kernel-on-fedora-20
5. http://www.vivaolinux.com.br/artigo/kernel-Linux-otimizado-Compilacao-e-teste