Linux Compilando o Kernel

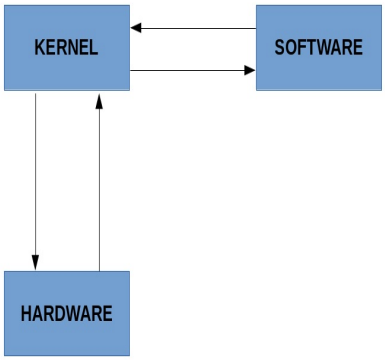
8 de agosto de 2015 / daher

Hoje trataremos de um assunto polêmico discutido entre os usuários Linux: a **compilação do Kernel**.

Muitos usuários pensam que a compilação de um Kernel é algo “de outro mundo”. Provaremos o contrário neste post.

O que é o Kernel?

O Kernel nada mais é que uma parte do sistema operacional que permite a comunicação entre o hardware e o software. Podemos utilizar a imagem abaixo como referência:



Download do Kernel

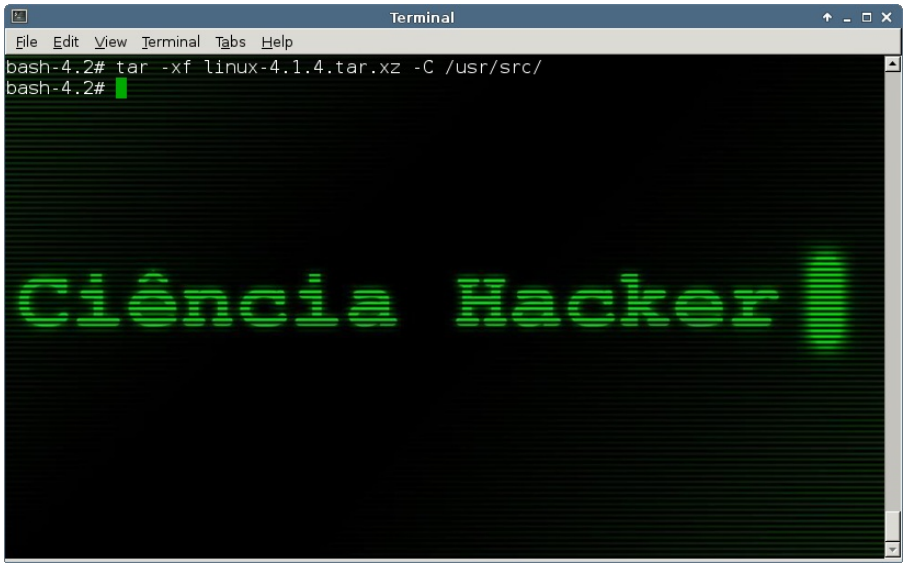
Para realizarmos o download da última versão do Kernel, poderemos utilizar o site kernel.org.

Descompactando no diretório /usr/src/

O diretório **/usr/src/** armazena o código-fonte do Kernel. Nele, descompactaremos o arquivo que baixamos no site utilizando o comando:

tar -xf arquivo_baixado.tar.xz -C /usr/src

Podemos – observando atentamente na versão do Kernel – seguir o exemplo da imagem abaixo:



Arquivo /proc/config.gz

O arquivo **config.gz**, localizado no diretório **/proc/** possui a finalidade de armazenar a versão de configuração do Kernel que está sendo utilizado no sistema. Precisaremos do conteúdo desse arquivo dentro da pasta onde descompactamos o Kernel baixado (**/usr/src/**) com o nome **.config**. Para isso, utilizaremos o comando `zcat` com a seguinte sintaxe:

zcat /proc/config.gz > /usr/src/pasta_descompactada/.config

Realizando a compilação

Para configurarmos esta parte, é necessário que estejamos no diretório **/usr/src/kernel_baixado**.

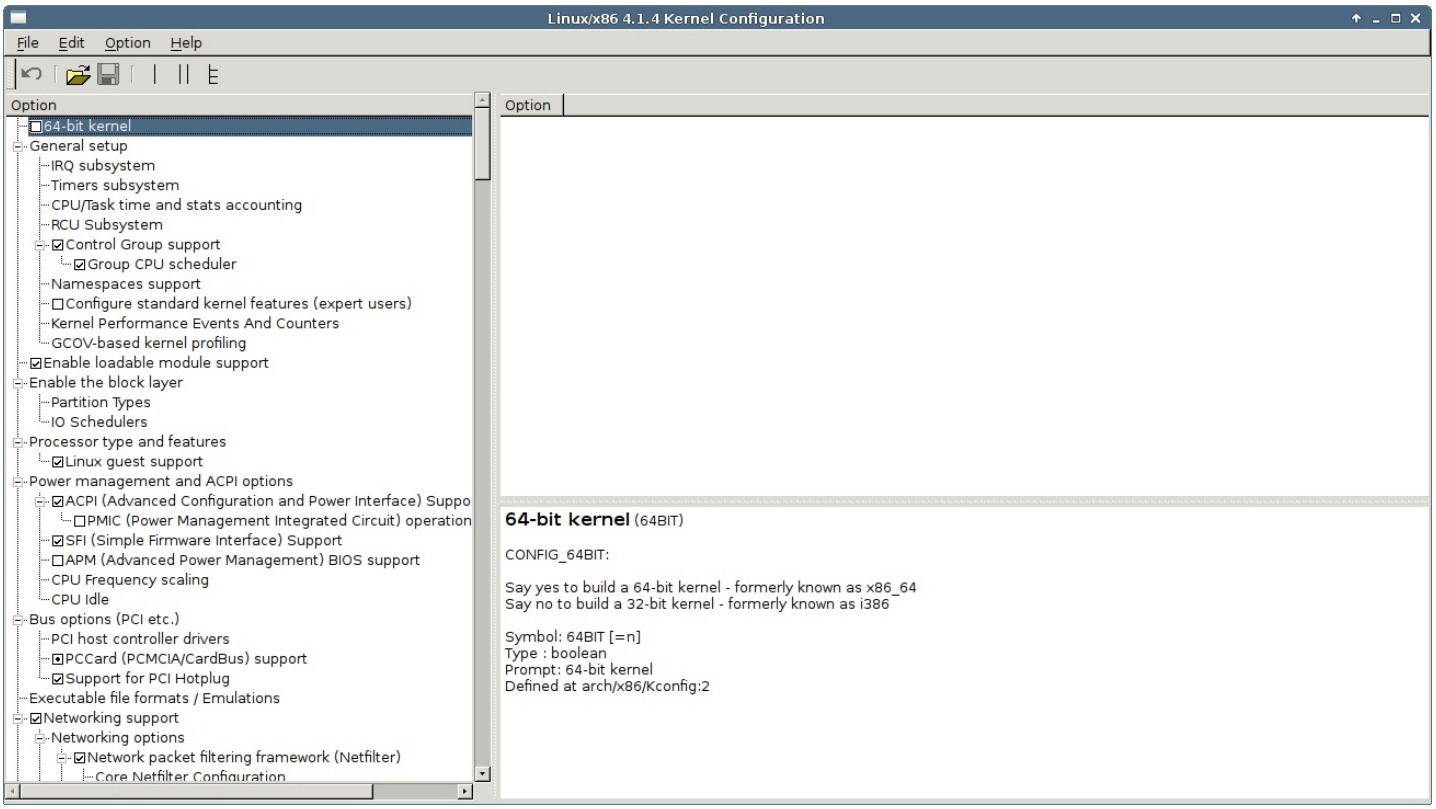
Definindo suporte

Definiremos para quais configurações nosso Kernel fornecerá suporte. Porém, isso exige um certo cuidado. Não se deve, de maneira alguma, modificar algum parâmetro ou opção que não seja de total conhecimento do usuário.

Para esta parte, contaremos com uma interface gráfica que será exibida após a execução do comando:

make xconfig

Após a execução, nos depararemos com a seguinte janela:



Após, você deve marcar e desmarcar as opções de acordo com sua necessidade. Atenção: Se seu sistema for x64, certifique-se de deixar a opção **64-bit kernel** marcada. Caso contrário, desmarque-a.

Compilando imagem

Compilaremos a imagem que será reconhecida pelo nosso gerenciador de inicializações (LILO, por exemplo). Para isso, executaremos o comando:

```
# make bzImage
```

Ativando módulos

Para ativarmos os módulos do Kernel, utilizaremos o comando abaixo:

```
# make modules
```

Instalando módulos

Instalaremos os módulos compilados no último passo utilizando o comando:

```
# make modules_install
```

Copiando arquivos

Com tudo compilado, copiaremos alguns dos arquivos necessários para seus diretórios. Obs.: Nosso diretório corrente ainda é o **/usr/src/kernel_baixado**. Copiaremos cada um dos arquivos para o diretório necessário. Para isso, utilizaremos os três comandos:

```
# cp arch/x86/bzImage /boot/vmlinuz-huge-smp-versão-smp
# cp System.map /boot/System.map-huge-smp-versão-smp
# cp .config /boot/config-huge-smp-versão-smp
```

Configurando inicialização

Para configurarmos a inicialização do Kernel, utilizaremos como exemplo o **LILO**. Porém, a mesma configuração pode ser feita no **GRUB**. Configuraremos, então, o arquivo **/etc/lilo.conf**. E modificaremos as seguintes linhas:

```
Linux bootable partition config begins
image =
```

Fazendo-as ficarem como abaixo:

```
Linux bootable partition config begins
image = vmlinuz-huge-smp-versão-smp
```

Se observarmos, perceberemos que a imagem de boot indicada é aquela que criamos utilizando o comando `# make bzImage`.

Conclusão

Se todos os passos foram seguidos corretamente, o Kernel compilado será ativado na próxima inicialização do sistema.