Laboratório 1.8: Kernel do Linux e Módulos - Parte I

Introdução

O kernel é o principal responsável por fazer o hardware e o software funcionarem juntos em um sistema operacional, usando endereços físicos e acessando o hardware em nome dos processos do usuário enquanto aplica as permissões de acesso. O carregador de inicialização faz o que é necessário para extrair e descompactar o kernel na RAM e depois passar o controle para ele.

O kernel encontra-se em um estado de sistema elevado, que inclui um espaço de memória protegido e acesso total ao hardware do dispositivo. Esse estado do sistema e o espaço de memória são chamados de espaço do kernel. Neste espaço, o acesso principal ao hardware e aos serviços do sistema são gerenciados e fornecidos como um serviço para o restante do sistema. Por outro lado, as aplicações do usuário são realizadas no chamado espaço do usuário, onde podem alcançar um subconjunto dos recursos disponíveis da máquina por meio de chamadas de sistema do kernel, ou seja, os usuários não possuem acesso direto ao hardware.

O kernel tem quatro principais funções:

- Gerenciamento da memória: monitora o volume de memória utilizado para armazenar o que (arquivos, dados etc.) e o local (ambiente).
- Gerenciamento de processos : determina quais processos podem usar a unidade central de processamento (CPU), quando e por quanto tempo.
- Drivers de dispositivos: atua como intermediário/intérprete entre o hardware e os processos.
- Chamadas do sistema e segurança: recebe solicitações dos processos para a execução de serviços.

Ao modificar um kernel, podemos editar o suporte de hardware como nativo ou podemos carregar o suporte como um módulo carregável conforme a necessidade.

1. Explorando o Kernel

Para verificar a versão do seu kernel instalado, execute o comando abaixo:

```
$ uname -r
```

```
5.10.16.3-microsoft-standard-WSL2
```

Também é possível obter a versão do kernel utilizando as informações contidas em /proc:

```
$ cat /proc/version
```

```
Linux version 5.10.16.3-microsoft-standard-WSL2 (oe-user@oe-host)
```

Dependendo da distribuição, se você tiver vários kernels instalados, poderá ver vários tipos de arquivo com a versão como parte de seus nomes de arquivo e, em seguida, um link simbólico apontando para o mais recente instalado por meio das atualizações da distribuição.

O arquivo do Kernel que é descompactado e carregado na memória pelo carregador de inicialização encontra-se em /boot/vmlinuz-<kernel-versão>. Possivelmente várias versões diferentes serão mostradas, além de um link simbólico chamado vmlinuz apontando para o arquivo mais recente. Acesse o diretório /boot e liste os arquivos e pastas neste diretório:

```
$ cd /boot
$ ls
```

```
config-5.4.0-109-generic        lost+found
```

Qual versão do kernel possui um link simbólico apontado pelo vmlinuz? Utilize o comando ls -l para verificar:

```
$ ls -l vmlinuz
```

```
lrwxrwxrwx 1 root root 25 Apr 19 21:17 vmlinuz -> vmlinuz-5.4.0-
```

Este é um arquivo grande com três componentes:

- Bloco de inicialização de 512 bytes
- Carregador de inicialização secundário
- Kernel compactado com Gzip

De onde vem esse nome? O Unix tradicionalmente inicializa a partir de um arquivo chamado `/unix`. Assim, os desenvolvedores do Linux usaram `/linux`. Com o uso de memória virtual, o arquivo passou a ser `/vmlinux`. Por fim, a última letra foi trocada por `z` para indicar que é um kernel compactado (zipped).

Configuração de Compilação do Kernel

O arquivo de configuração do kernel atual encontra-se em /boot/config-. Este arquivo permite algumas informações caso enfrente problemas de kernel, pois fornece a

configuração exata usada durante a compilação. Para verificar o conteúdo do arquivo, execute o comando:

```
$ cat /boot/config-$(uname -r)
```

Copie do terminal e cole aqui as 8 últimas linhas do arquivo acima:

```
# CONFIG_DEBUG_ENTRY is not set
# CONFIG_DEBUG_NMI_SELFTEST is not set
CONFIG_X86_DEBUG_FPU=y
CONFIG_PUNIT_ATOM_DEBUG=m
# CONFIG_UNWINDER_ORC is not set
CONFIG_UNWINDER_FRAME_POINTER=y
# CONFIG_UNWINDER_GUESS is not set
# end of Kernel hacking
```

2. Detecção de Hardware do Kernel

O sistema de arquivos /proc é realmente uma grande coleção de estruturas de dados do kernel apresentadas em um formato razoavelmente amigável, utilizado para obter as informações de hardware da máquina. Parece ser uma hierarquia de diretórios e arquivos, que você pode explorar com cd e ls e investigar em muitos casos com cat.

Para saber o que o kernel detectou sobre a CPU, execute o comando: `cat /proc/cpuinfo`. Com base nesse comando, podemos obter quantos processadores estão sendo identificados pelo kernel:

```
$ cat /proc/cpuinfo | grep processor | wc -l
```

```
2
```

Para saber o que o kernel detectou sobre o estado de memória atual, execute o comando: cat /proc/meminfo. Com base nesse comando, qual a memória total da máquina identificada pelo kernel?

```
$ cat /proc/meminfo | grep MemTotal
```

```
MemTotal:        2023564 kB
```

O kernel também pode fornecer para o espaço de usuário informações sobre as partições disponíveis. Para verificar, execute o comando:

```
$ cat /proc/partitions
```

```
major minor  #blocks  name
```

Podemos listar também quais dispositivos estão conectados. Observe que o carregamento de módulos do kernel pode levar à detecção de mais hardware e ao aparecimento automático de mais arquivos especiais de dispositivo em /dev. Acessando o /dev, quantos dispositivos são reconhecidos pelo kernel?

```
$ ls /dev | wc -l
```

```
208
```

O Kernel pode fornecer informações de quais os dispositivos com padrão de barramento PCI estão conectados no momento. O `lspci` é um comando em sistemas operacionais do tipo Unix que imprime informações detalhadas sobre todos os barramentos e dispositivos PCI no sistema. Ele é baseado em uma biblioteca portátil comum `libpci`, que oferece acesso ao espaço de configuração PCI em vários sistemas operacionais.

```
$ lspci | tail -5       # Lista os últimos 5 dispositivos
```

```
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE
[Natoma/Triton II]
00:01.2 USB controller: Intel Corporation 82371SB PIIX3 USB
[Natoma/Triton II] (rev 01)
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev
01)
```

Assim como os dispositivos PCI, o Kernel tem informações sobre os dispositivos USB conectados ao sistema. As informações podem ser obtidas através do comando:

```
$ lsusb
```

```
Bus 001 Device 002: ID 0627:0001 Adomax Technology Co., Ltd QEMU
USB Tablet
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Para obter informações mais detalhadas sobre os dispositivos, execute os comandos acima com o parâmetro -v. Exemplo: `lsusb -v`

As informações acima e muitas outras são fornecidas diretamente pelo kernel para as aplicações do espaço do usuário. Dessa forma, as aplicações não precisam se preocupar em ter acesso diretamente ao hardware da máquina e nem de fazer modificações complexas no sistema. Tudo fica por responsábilidade do kernel em fornecer as informações necessárias de maneira simples.

3. Variaveis do Kernel

O Kernel possui algumas variáveis já definidas e que podem ser encontradas em sysctl. Para verificar quais variáveis estão definidas, use o comando:

```
$ sysctl -a
```

O comando acima reune diversas variáveis para configuração. Por exemplo, a estrutura de dados do kernel net.ipv4.tcp_fin_timeout que define o timeout no tcp é acessível como o arquivo /proc/sys/net/ipv4/tcp_fin_timeout. Em vez desse comando sysctl, você poderia ter alterado para o diretório /proc/sys/net/ipv4 e exibido seu conteúdo com cat:

```
$ cat /proc/sys/net/ipv4/tcp_fin_timeout
```

```
60
```

Observe que o arquivo acima resultou na mesma saída do comando sysctl quando buscamos pelo arquivo tcp_fin_timeout:

```
$ sysctl -a | grep net.ipv4.tcp_fin_timeout
```

```
60
```

Você pode ler os valores atuais desses temporizadores de kernel, contadores e outros campos. Você também pode alterá-los! Agora vamos dizer que sua modificação entusiasmada dos valores do kernel acidentalmente coloca seu kernel em execução em um estado critico - isso não é nada improvável se você não for cuidadoso. Tudo o que você modificou é o kernel em execução na RAM, o arquivo do kernel armazenado no disco permanece inalterado. Então, bastaria reiniciar e você estaria de volta ao estado padrão. Mas por que você iria querer mexer com os valores do kernel? Pois pode ser feito um ajuste nos parâmetros do kernel em execução com finalidade de desempenho ou segurança.

4. Módulos do Kernel

Também conhecido como LKM (*Loadable Kernel Modules*), este é um mecanismo para adicionar ou remover código do kernel Linux em tempo de execução. Eles são ideais para drivers de dispositivo, permitindo que o kernel se comunique com o hardware sem precisar saber como o hardware funciona. A alternativa aos LKMs seria ter que recompilar o kernel para cada novo driver.

Sem essa capacidade modular, o kernel do Linux seria muito grande, pois teria que suportar todos os drivers que seriam necessários. Também seria necessário reconstruir o kernel toda vez que quisesse adicionar um novo hardware ou atualizar um driver de dispositivo.

Os LKMs são carregados em tempo de execução, mas não são executados no espaço do usuário, assim, o código do kernel não pode acessar bibliotecas de código que são escritas para o espaço do usuário Linux. A interface entre o espaço do kernel e o espaço do usuário é claramente definida e controlada.

Assim, a maioria dos drivers são implementados como módulos de kernel do Linux. Quando esses drivers não são necessários, podemos descarregar apenas esse driver específico, o que reduzirá o tamanho da imagem do kernel.

Os módulos do kernel terão uma extensão .ko. Em um sistema linux normal, os módulos do kernel residirão dentro do diretório /lib/modules/kernel_version/kernel/. Mostre o conteúdo desse diretório:

```
$ ls /lib/modules/$(uname -r)/kernel
```

```
arch   crypto   fs      lib  net      sound   virtualbox-guest
zfs
```

Você pode facilmente listar todos os módulos do kernel que estão atualmente carregados no kernel executando o comando `lsmod`, por exemplo:

```
$ lsmod
```

Copie do terminal e cole aqui as 8 últimas linhas do arquivo acima:

```
crypto_simd            16384  1 aesni_intel
cryptd                 24576  2 crypto_simd,ghash_clmulni_intel
glue_helper            16384  1 aesni_intel
drm                   491520  3 drm_kms_helper,cirrus
psmouse               155648  0
i2c_piix4              28672  0
pata_acpi              16384  0
floppy                 81920  0
```

Cada linha de saída `lsmod` especifica o nome de um módulo do kernel atualmente carregado na memória, a quantidade de memória que usa e a soma total dos processos que estão usando o módulo e outros módulos que dependem dele.

Finalmente, observe que a saída `lsmod` é menos detalhada e consideravelmente mais fácil de ler do que o conteúdo do pseudo-arquivo `/proc/modules`.

Você também pode encontrar informações sobre um módulo específico usando o comando $ modinfo nome_do_modulo. Por exemplo, para obter o nome do arquivo do

modulo de video, utilize:

```
$ modinfo video | grep filename
```

```
filename:       /lib/modules/5.4.0-109-
generic/kernel/drivers/acpi/video.ko
```

Assim vemos que o utilitário `modinfo` fornece várias informações sobre este módulo. As informações que você obtém são de responsabilidade dos desenvolvedores desse módulo. Então você pode obter algo muito útil, com uma explicação, uma lista de parâmetros opcionais de tempo de carregamento e o que eles significam, e assim por diante. Ou você pode obter uma tabela enigmática de endereços hexadecimais e uma lista de endereços de barramento PCI e um lembrete de que você sempre pode ler o código-fonte C e descobrir a partir daí.

To infinity... and beyond! — Buzz Lightyear

π