

Pontifícia Universidade Católica de Campinas Faculdade de Engenharia de Computação - FECOMP

Sistemas Operacionais B – Relatório Atividade 2

Beatriz Morelatto Lorente RA: 18071597

Cesar Marrote Manzano RA: 18051755

Fabricio Silva Cardoso RA: 18023481

Pedro Ignácio Trevisan RA: 18016568

**Sumário**

**1.Introdução.......................................................................................................3**

**2.Conceitos iniciais...........................................................................................4**

**3.Principais passos da compilação de um módulo.......................................5**

**4.Conclusão.......................................................................................................**

**Introdução**

O experimento desenvolvido pretende demonstrar os detalhes da implementação de um módulo do kernel Linux e se familiarizar com a programação dentro do kernel. Todo o processo de instalação de um módulo será explicado ao longo do relatório e é necessário compreender cada passo para obtermos um resultado satisfatório.

**Conceitos iniciais**

Antes de demonstrar o processo de implementação de um módulo do kernel Linux, é importante esclarecer alguns pontos para que o entendimento do experimento fique mais fácil.

Alguns detalhes sobre os módulos

Os módulos que vamos trabalhar são aqueles que são carregados no kernel em tempo de execução, ou seja, estendem a funcionalidade do kernel em execução.

Normalmente os programas feitos em C, começam com a função main(), executando vários passos e encerrando sua execução. Já quando um módulo é carregado para dentro do kernel, ele não é executado imediatamente, ou seja, ele apenas avisa o kernel que está disponível para ser chamado quando precisar. Isto é feito através de uma função de entrada, que mostra sua funcionalidade para o kernel. Os módulos também possuem uma função para desfazer tudo que foi feito pela função de entrada, liberando recursos e afins, quando este for retirado do kernel.

Os módulos que serão apresentados serão básicos e suas funcionalidades não são muito práticas, serão mostrados apenas para compreensão do processo de instalação.

Espaço de usuário e espaço de kernel

Quando nos referimos ao espaço de kernel, falamos de um lugar onde não há restrições, ou seja, uma aplicação rodando nesse espaço tem acesso total à máquina desde a memória até o controle do hardware do computador. Todos os módulos que serão mostrados rodam nesse espaço, sem restrições. Já o espaço de usuário é mais restritivo, menos privilegiado e há recursos que são limitados para as aplicações que rodam nesse espaço. Esses níveis servem para garantir segurança ao sistema operacional, impedindo que programas de terceiros venham a prejudicar o funcionamento normal da máquina.

Programação em espaço de kernel

Ao se programar em espaço de kernel, também há diversas restrições, portanto, ao programar um módulo haverá algumas diferenças no modo de se projetá-lo.

Uma diferença notável são as bibliotecas usadas. Uma vez que o kernel roda por si só, não há como usar as bibliotecas padrão da linguagem C, como a ‘stdlib’, ‘stdio’, por exemplo. Todas as funções usadas em um módulo são aquelas que são vem com o kernel, ou seja, as únicas bibliotecas que podem ser usadas são fornecidas pelo kernel

**Principais passos da compilação de um módulo**

Para iniciarmos o processo de instalação e compilação de um módulo do kernel Linux, é importante preparar o ambiente e já ter conhecimento de alguns aspectos do kernel. Para o experimento foi usado a versão do kernel 4.15.0 e o Ubuntu 16.04. Todo o processo foi feito através do terminal do Linux.

É recomendado que todo o processo seja feito no modo ‘root’ da máquina. Para isso basta abrir um terminal e digitar o comando *‘sudo su’*.

Inicialmente é necessário instalar a biblioteca ‘build-essential’, que conterá alguns comandos essenciais para a compilação de um módulo. Basta digitar o comando *‘sudo apt-get install build-essential kmod’* ou apenas *‘sudo apt-get install build-essential’.*

Após instalar a biblioteca é necessário instalar o ‘header files’ para o o seu sistema. Isso possibilitará que o módulo tenha acesso as funções oferecidas pelo kernel. Antes de instalar é importante atualizar algumas coisas no seu kernel com o comando *‘sudo apt-get update’*. Antes de instalar é necessário pesquisar algumas das versões disponíveis, com o comando *‘apt-cache search linux-headers-$(uname -r)’.* Caso o comando não retorne nada, significa que você já tem uma versão instalada. Caso contrário, basta instalar com o comando *‘sudo apt-get install linux-headers-xxx’* (sendo ‘xxx’ a versão que você deseja).

A partir desse ponto será mostrado a compilação dos módulos, de fato. Para acessar o código fonte dos módulos apresentados, basta acessar o link abaixo:

<https://github.com/cesarmmanzano/Experimentos-SO-B/tree/Atividade2/Atividade%202/modulos>

Primeiro módulo: hello-1

Para qualquer módulo é necessário fazer um arquivo com o nome ‘Makefile’. Isso é necessário para que o módulo possa ser compilado. Para isso, basta abrir um editor de sua preferência e copiar o texto abaixo:

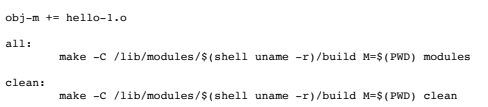
****

Figura 1 - Arquivo Makefile do primeiro módulo

Após fazer o arquivo, basta digitar o comando *‘make’* na pasta na qual o seu módulo está inserido, para a compilação iniciar.

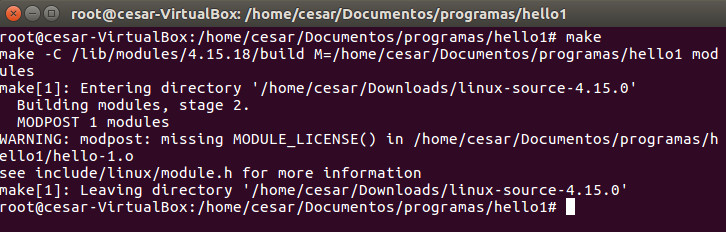


Figura 2 - Print mostrando o comando make para a compilação do primeiro programa

Após a compilação deverá aparecer um arquivo com o final ‘.ko’. Este arquivo é extremamente importante e será usado na continuação da compilação .

Próximo passo é digitar o comando *‘sudo insmod hello-1.ko’.* É bem provável que não mostre nenhuma mensagem no terminal. Para visualizar a mensagem do módulo, abrimos outro terminal e utilizamos o comando *‘tail -f /var/log/kern.log’* para podemos ver se a compilação foi correta. A imagem abaixo mostra que a compilação foi feita com sucesso.

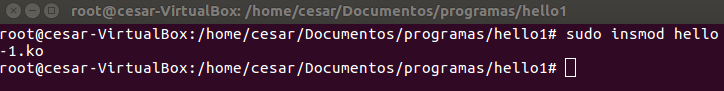


Figura 3 - Print mostrando o comando insmod

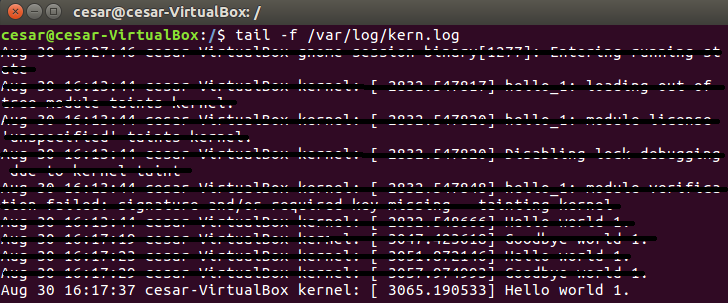


Figura 4 - Print mostrando o resultado da compilação do primeiro módulo

Após a compilação, o módulo pode ser retirado com o comando *‘sudo rmmod hello\_1’.* O comando, no caso do módulo exemplo, também retorna uma mensagem, que pode ser conferida com os comandos: *‘tail -f /var/log/kern.log’* (já citado anteriormente) ou *‘journalctl --since "1 hour ago" | grep kernel’.*

Os passos para a compilação dos próximos programas serão os mesmos portanto, as explicações sobre os comandos será reduzida, focando mais nos resultados.

Segundo módulo: hello-2

A imagem abaixo mostra o arquivo ‘Makefile’ usado para a compilação do módulo:

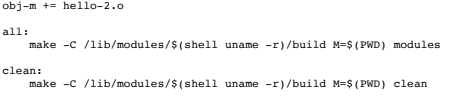


Figura 5 - Arquivo Makefile do segundo módulo

A imagem a seguir mostra o resultado da compilação do módulo.

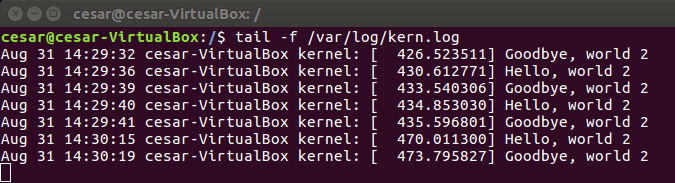


Figura 6 - Print mostrando o resultado da compilação do segundo módulo

Cada ‘Hello, world 2’ é mostrado quando o módulo é inserido no kernel e cada ‘Goodbe world 2’ é mostrado quando o módulo é retirado do kernel.

Terceiro módulo: hello-3

Para o arquivo ‘Makefile’, apenas mudar a primeira linha, substituindo o nome do arquivo pelo desejado.

A imagem a seguir mostra o resultado da compilação do módulo.

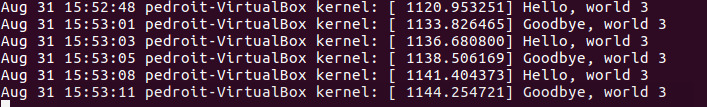


Figura 7 - Print mostrando o resultado da compilação do terceiro módulo

Quarto módulo: hello-4.c

A imagem a seguir mostra o resultado da compilação do módulo.

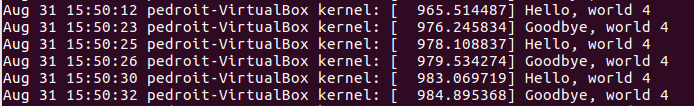


Figura 8 - Print da tela com o resultado da compilação do quarto módulo

Quinto módulo: hello-5

Neste módulo podemos passar diferentes parâmetros para a compilação do mesmo. Como será mostrado nas imagens abaixo, conforme alteramos os parâmetros alteramos o resultado da compilação.



Figura 9 - Compilação do módulo 5 com diferetes parâmetros

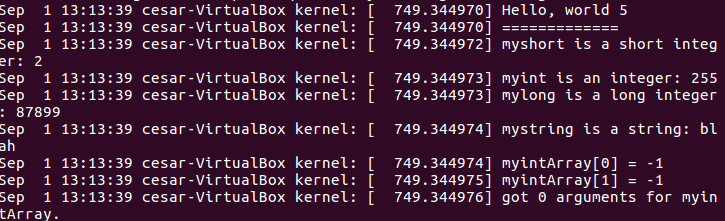


Figura 10 - Print mostrando um dos resultados da compilação do quinto módulo

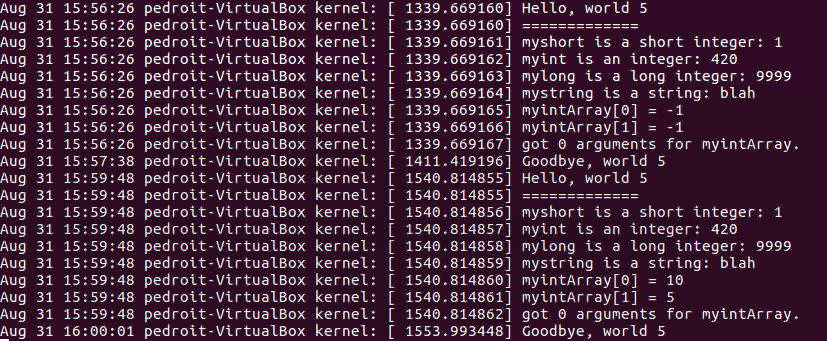


Figura 11 - Print mostrando um dos resultados da compilação do quinto módulo

Sexto módulo: startstop

Neste módulo, podemos ver como funciona a compilação de um módulo, quando este é dividido em dois arquivos diferentes. O primeiro arquivo (start.c) contém apenas a função de inicializar o módulo e já o segundo (stop.c) contém a função de saída do módulo. As imagens abaixo mostram o conteúdo de cada arquivo citado.

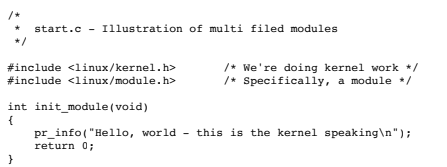


Figura 12 - Programa start.c

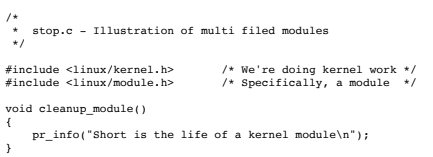


Figura 13 – Programa stop.c

Pelo fato do módulo ser composto por dois arquivos diferentes, o ‘Makefile’ deste também sofrerá mudanças.

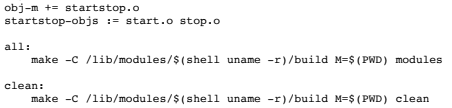


Figura 14 - Arquivo Makefile do sexto módulo

Na primeira linha do arquivo, escolhemos um nome para o módulo e na segunda quais os arquivos que o compõe.

Para a compilação do módulo basta digitar *‘sudo insmod (nome do módulo escolhido)’* e para remove-lo do kernel *‘sudo rmmod (nome do módulo escolhido)’.* Portanto, o processo de compilação de um módulo com mais de um arquivo origem é bem simples, com pequenas alterações no arquivo ‘Makefile’.

As imagens abaixo mostram o resultado da compilção deste módulo.

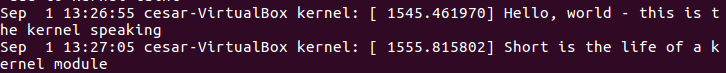
**

Figura 15 - Print mostrando o resultado da compilação do sexto módulo



Figura 16 - Print mostrando o resultado da compilação do sexto módulo

Sétimo módulo: chardev

A imagem abaixo mostra o resultado da compilação do módulo.



Figura 17 - Print dos comandos para compilação do sétimo módulo

Oitavo módulo: procfs1

As imagens abaixo mostram o resultado da compilação do módulo.

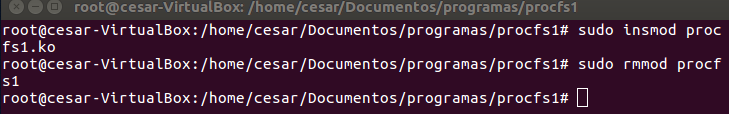


Figura 18 - Print dos comandos para compilação do oitavo módulo

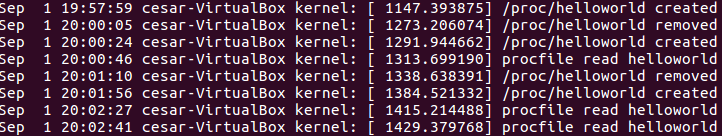


Figura 19 - Print mostrando o resultado da compilação do oitavo módulo

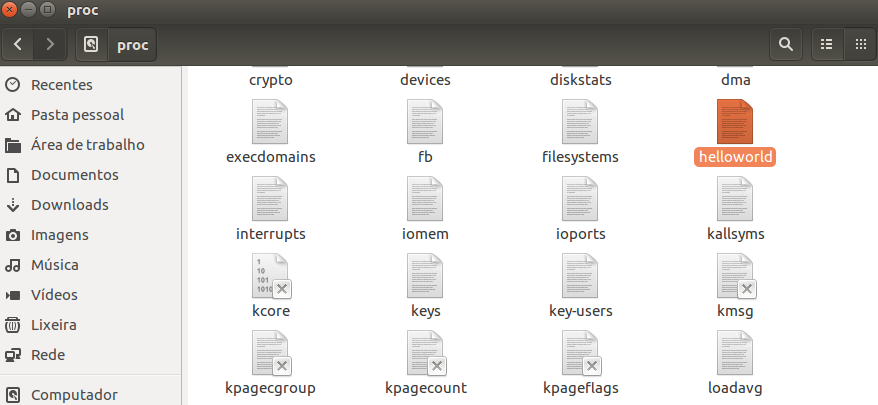


Figura 20 - Print mostrando o arquivo criado pelo módulo

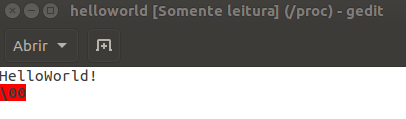


Figura 21 - Print mostrando o conteúdo do arquivo criado pelo módulo

Nono módulo: procfs2

As imagens abaixo mostram o resultado da compilação do módulo.

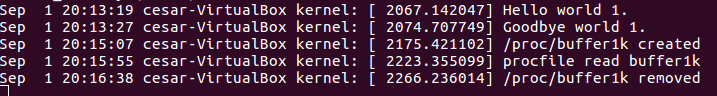


Figura 22 - Print mostrando o resultado da compilação do nono módulo

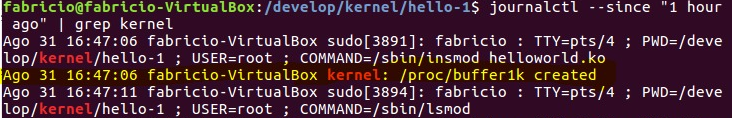


Figura 23 - Print mostrando o resultado da compilação do nono módulo

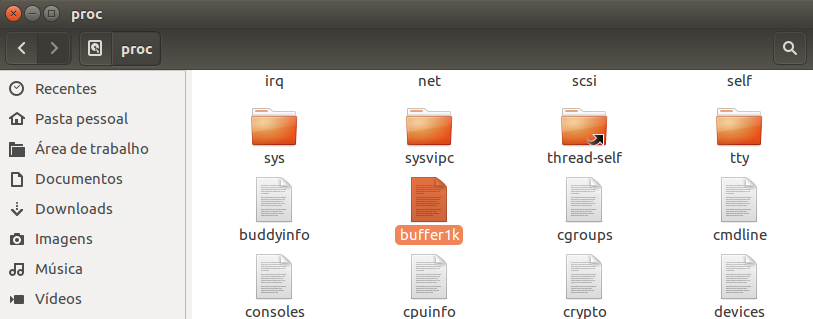


Figura 24 - Print mostrando o arquivo criado pelo módulo

Décimo módulo: procfs3

Décimo primeiro módulo: procfs4

**Conclusão**

Com o experimento foi possível compreender todos os passos da implementação, compilação e instalação de um módulo para o kernel Linux. Mesmo que as utilidades de cada módulo não fossem práticas, foi importante entender a diferença da programação em espaço de kernel, uma vez que as bibliotecas são diferentes e o jeito de programar também. Também foi compreendido o uso de alguns comandos para compilação como: ‘insmod’, ‘rmmod’, ‘journalctl’ e ‘tail -f’.