Módulo Hello, Kernel!

Pergunta 1: Após carregar (insmod), verifique as mensagens de debug do kernel:

$ sudo dmesg

O que apareceu lá?

Resposta:
[1256.521484] hello: loading out-of-tree module taints kernel.
[1256.521516] hello: module verification failed: signature and/or required key missing - tainting kernel
[1256.521612] <1> Hello, kernel!

Pergunta 2: Depois descarregue o módulo (rmmod) e verifique novamente as mensagens de debug do kernel. Há algo novo? Entendeu como funciona e para que serve a função printk?

Resposta:
[1256.521484] hello: loading out-of-tree module taints kernel.
[1256.521516] hello: module verification failed: signature and/or required key missing - tainting kernel
[1256.521612] <1> Hello, kernel!
[2031.635396] <1> Bye, kernel!

A novidade fica por conta da mensagem “Bye, kernel!” que não era exibida anteriormente. Esta mensagem só foi exibida porque o módulo foi descarregado, visto que o módulo de finalização só é chamado no descarregamento (por rmmod).
Como não é recomendado o uso de bibliotecas padrão ou cabeçalhos, utiliza-se o “printk” que é ideal para exibir mensagens impressas no buffer de log do kernel.

Módulo de cópia de memória

Pergunta 1: Qual foi a saída do comando "cat"? Por que?

Resposta: Saída: “e”. Ou seja, saiu apenas o último caractere da mensagem. Isso ocorreu pois o buffer está configurado para armazenar apenas o último caractere da mensagem “hello, module”.

Pergunta 2: O que será preciso modificar no código para que sejam armazenados os 5 últimos caracteres da mensagem?

Resposta: Para ajustar o funcionamento do módulo de forma que sejam obtidos os 5 últimos caracteres, foi necessário ajustar as funções "memory_read", nas seguintes linhas:

linha 98: rv=copy_to_user (buf, memory_buffer, n);
linha 105: return n;

E também na função "memory_write", na linha:

119: rv=copy_from_user (memory_buffer, tmp, n);

Adicionalmente, a função "memory_init" também foi ajustada para alocar a quantidade correta:

58: memset (memory_buffer, 0, n);

A variável "n" foi definida na linha 14 do código e ajusta a quantidade de caracteres a serem armazenados e lidos. Nesse caso, n = 5:

14: #define n 5

O comando usado para realizar o teste foi:

sudo rmmod memory.ko ; make ; sudo insmod memory.ko ; echo -n "hello, MODULE" > /dev/memory ; cat /dev/memory

Saída do comando acima:

```
make -C /lib/modules/4.15.0-29-generic/build M=/home/aluno/sop modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-29-generic'
Makefile:976: "Cannot use CONFIG_STACK_VALIDATION=y, please install libelf-dev, libelf-devel or elfutils-libelf-devel"
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-29-generic'
ODULEaluno@ubuntu:~/sop$
```

No comando, acima, o módulo é descarregado, o arquivo de módulo é regerado, módulo então é carregado novamente, é inserida a frase no "/dev/memory" e depois essa frase é lida.

Para que o dispositivo de memória exista, foi executados os comandos abaixo:

```
$ sudo mknod /dev/memory c 60 0
$ sudo chmod 666 /dev/memory
```

Observação: os ajustes feitos foram devidamente comentados no código do programa “memory.c”.