Gestión y depuración del núcleo

Este tema es relativo al seguimiento y a la configuración dinámica del núcleo y sus módulos dinámicos. Estudiaremos diferentes comandos y herramientas que permiten listar, cargar y descargar los módulos LKM, visualizar y modificar dinámicamente los parámetros de los módulos y del núcleo, y forzar esos parámetros para aplicarlos durante el arranque del sistema.

El servicio `udev` y los comandos que permiten seguir el estado de los dispositivos detectados por el sistema son tratados en la sección Gestión de los discos duros locales del capítulo Administración avanzada de dispositivos de almacenamiento.

1. Gestión de los módulos de núcleo LKM

Los módulos de núcleo dinámicos (LKM, *Loadable Kernel Module*), definidos en la compilación del núcleo, son gestionados automáticamente. Algunos, necesarios durante la carga y la inicialización del núcleo, se cargan en memoria mediante el archivo de disco virtual `initramfs`. El núcleo carga también sistemáticamente durante el arranque todos los módulos específicados en un archivo de configuración.

Otros módulos se cargan en el momento que sean solicitados, la primera vez que un programa hace una llamada al sistema cuando necesita su uso. Por otro lado, se pueden descargar de la memoria los módulos que ya no estén siendo utilizados.

Si una aplicación necesita la carga de un módulo que no ha sido tomado en cuenta por el núcleo, éste generará un error en respuesta a su solicitud. Para que la aplicación pueda funcionar, habrá que copiar el archivo del módulo en el sistema (si todavía no se ha hecho), cargarlo manualmente y configurarlo en carga automática.

Distintos comandos y archivos de configuración permiten controlar el funcionamiento de los módulos, gestionar su carga y descarga, así como sus parámetros.

a. Ubicacion de los módulos

Los archivos que contienen el código ejecutable de los módulos están almacenados en la arborescencia en `/lib/modules`. Cada versión del núcleo instalado en el sistema tiene su propio directorio de almacenamiento de los módulos.

Ejemplo

Version del núcleo:

```
uname -r
4.18.0-147.5.1.el8_1.x86_64
```

Directorio de los módulos de esta versión del núcleo:

```
ls -l /lib/modules/4.18.0-147.5.1.el8_1.x86_64
total 15496
-rw-r--r--. 1 root root     325  5 oct 02:07 bls.conf
lrwxrwxrwx. 1 root root      44  5 oct 02:07 build ->
/usr/src/kernels/4.18.0-147.5.1.el8_1.x86_64
-rw-r--r--. 1 root root  184825  5 oct 02:07 config
drwxr-xr-x. 12 root root     128 29 oct 15:44 kernel
-rw-r--r--. 1 root root  849682 29 oct 16:00 modules.alias
-rw-r--r--. 1 root root  813918 29 oct 16:00 modules.alias.bin
-rw-r--r--. 1 root root     488  5 oct 02:07 modules.block
-rw-r--r--. 1 root root    7534  5 oct 02:07 modules.builtin
-rw-r--r--. 1 root root    9748 29 oct 16:00 modules.builtin.bin
-rw-r--r--. 1 root root  277921 29 oct 16:00 modules.dep
-rw-r--r--. 1 root root  386173 29 oct 16:00 modules.dep.bin
-rw-r--r--. 1 root root     365 29 oct 16:00 modules.devname
-rw-r--r--. 1 root root     140  5 oct 02:07 modules.drm
-rw-r--r--. 1 root root      59  5 oct 02:07 modules.modesyting
-rw-r--r--. 1 root root    1595  5 oct 02:07 modules.networking
-rw-r--r--. 1 root root   98346  5 oct 02:07 modules.order
-rw-r--r--. 1 root root     591 29 oct 16:00 modules.softdep
-rw-r--r--. 1 root root  404630 29 oct 16:00 modules.symbols
-rw-r--r--. 1 root root  494618 29 oct 16:00 modules.symbols.bin
lrwxrwxrwx. 1 root root       5  5 oct 02:07 source -> build
-rw-r--r--. 1 root root  339854  5 oct 02:07 symvers.gz
```

```
-rw-------.  1 root root 3841454  5 oct 02:07 System.map
drwxr-xr-x.  2 root root       6  5 oct 02:07 updates
drwxr-xr-x.  2 root root      40 29 oct 15:44 vdso
-rwxr-xr-x.  1 root root 8106744  5 oct 02:07 vmlinuz
drwxr-xr-x.  3 root root      23 29 oct 15:57 weak-updates
```

Los archivos objeto de los módulos se encuentran en el subdirectorio `kernel`, estructurado en subdirectorios que corresponden al tipo de módulo:

```
ls -l /lib/modules/4.18.0-147.5.1.el8_1.x86_64/kernel
total 16
drwxr-xr-x.  3 root root   17 29 oct 15:44 arch
drwxr-xr-x.  3 root root 4096 29 oct 15:44 crypto
drwxr-xr-x. 66 root root 4096 29 oct 15:44 drivers
drwxr-xr-x. 24 root root 4096 29 oct 15:44 fs
drwxr-xr-x.  3 root root   19 29 oct 15:44 kernel
drwxr-xr-x.  4 root root  240 29 oct 15:44 lib
drwxr-xr-x.  2 root root   35 29 oct 15:45 mm
drwxr-xr-x. 36 root root 4096 29 oct 15:44 net
drwxr-xr-x. 13 root root  184 29 oct 15:45 sound
drwxr-xr-x.  3 root root   17 29 oct 15:44 virt
```

Los módulos son archivos que presentan una extensión `.ko` (`.o` en las antiguas versiones del núcleo), eventualmente comprimidos en formato `xz`.

Ejemplo

Módulo LKM del driver del sistema de archivos de tipo ext4.

En una distribución Debian 10:

```
ls -l /lib/modules/4.19.0-8-amd64/kernel/fs/ext4
total 1416
-rw-r--r-- 1 root root 1448540 oct. 26 21:01 ext4.ko
```

En una distribución CentOS 8:

```
ls -l /lib/modules/4.18.0-147.5.1.el8_1.x86_64/kernel/fs/ext4
total 256
-rw-r--r--. 1 root root 259992  5 oct 02:16 ext4.ko.xz
```

b. Lista de los módulos LKM cargados en memoria

El comando `lsmod` muestra la lista de los módulos cargados en memoria. Este comando no presenta ninguna opción.

La primera columna indica el nombre del módulo, la segunda su tamaño en bytes.

La última columna indica el número de elementos que utilizan este módulo. Puede tratarse de procesos, del núcleo o de otros módulos. En el último caso, el nombre de los módulos dependientes está indicado. Un módulo no puede ser descargado de la memoria si su número es igual a cero.

Ejemplo

```
lsmod
Module                  Size  Used by
wl                   6463488  0
powernow_k8            36864  0
uvcvideo              118784  0
edac_mce_amd           28672  0
[...]
ext4                  741376  5
[...]
```

También se puede obtener la lista de los módulos del núcleo cargados actualmente en memoria, usando el pseudoarchivo `/proc/modules`.

Ejemplo

```
more /proc/modules
ath3k 20480 0 - Live 0xffffffffc0764000
sr_mod 28672 0 - Live 0xffffffffc0f6b000
```

```
cdrom 65536 1 sr_mod, Live 0xffffffffc0753000
binfmt_misc 20480 1 - Live 0xffffffffc0f65000
joydev 24576 0 - Live 0xffffffffc0723000
uas 28672 0 - Live 0xffffffffc1114000
usb_storage 73728 1 uas, Live 0xffffffffc10ee000
[...]
```

c. Descarga de un módulo

El comando `rmmod NombreMódulo` descarga el módulo LKM `NombreMódulo` de la memoria, siempre y cuando no sea necesario para un proceso o para el núcleo, y que no esté vinculado a otro módulo.

El comando `modprobe -r NombreMódulo` descarga el módulo LKM `NombreMódulo` de la memoria así como los módulos de los que depende, siempre y cuando ninguno esté siendo usado por un proceso o por el núcleo.

Ejemplo

Buscamos los módulos cargados y no usados:

```
lsmod | grep '0$'
uas                28672  0
nf_conntrack_ipv6     20480  10
nf_conntrack_ipv4     16384  10
iscsi_tcp             20480  0
ip6_tables             32768  0
ip_tables             28672  0
nft_compat             20480  0
ip_sy                49152  0
ath9k               155648  0
hp_wmi                16384  0
[...]
joydev              24576  0
[...]
```

El módulo `joydev` (driver de joystick) no parece que sea indispensable por el momento, lo

descargaremos de la memoria viva:

```
rmmod joydev
lsmod | grep joydev
```

El módulo se ha descargado.

Intentamos descargar el módulo `cdrom`*:*

```
rmmod cdrom
rmmod: ERROR: Module cdrom is in use by: sr_mod
```

El módulo es una dependencia de otro módulo, el comando falla.

Y, sin embargo, el módulo `sr_mod` *no está siendo usado:*

```
lsmod | grep sr_mod
sr_mod          28672  0
cdrom           65536  1 sr_mod
```

Usamos el comando `modprobe -r` *para descargar el módulo* `sr_mod`*:*

```
modprobe -r sr_mod
lsmod | grep sr_mod
lsmod | grep cdrom
```

El comando ha descargado de la memoria los dos módulos.

d. Carga de un módulo

El comando `insmod ArchivoMód` carga en memoria el módulo LKM cuyo camino de acceso es `ArchivoMód`, siempre y cuando los módulos de los que depende estén cargados.

Hay que cargar sucesivamente los módulos que tengan una relación de dependencia con otros módulos, esto puede ser difícil de gestionar.

El comando `modprobe NombreMódulo` carga en memoria el módulo LKM `NombreMódulo` así como los módulos de los que depende, si no han sido cargados todavía. Este comando tiene un uso más fácil que el anterior.

Ejemplo

Para comprobar una nueva versión del juegos desarrollado por la empresa, necesitamos recargar el driver `joydev` descargado anteriormente de la memoria.

Primero lo intentamos con el comando `insmod`:

```
insmod joydev
insmod: ERROR: could not load module joydev: No such file or directory
```

El comando falla porque necesita el camino de acceso al archivo del módulo:

```
find /lib/modules -name 'joydev*'
/lib/modules/4.18.0-147.5.1.el8_1.x86_64/kernel/drivers/input/joydev.ko.xz
insmod /lib/modules/4.18.0-147.5.1.el8_1.x86_64/kernel/drivers/input/joydev.ko.xz
lsmod | grep joydev
joydev             24576  0
```

La carga ha funcionado correctamente.

Se carga el módulo `sr_mod`, que depende del módulo `cdrom`.

Comprobamos que no estén cargados:

```
lsmod | grep sr_mod
lsmod | grep cdrom
```

Usamos el comando `modprobe` para cargar el módulo `sr_mod`:

```
modprobe sr_mod
lsmod | grep sr_mod
sr_mod             28672  0
cdrom              65536  1 sr_mod
```

El comando ha cargado los dos módulos.

e. Carga de los módulos durante el arranque del sistema

La mayoría de los módulos LKM se cargan durante el arranque del sistema, vinculados a la detección del hardware.

Se puede forzar la carga de un módulo durante el arranque del sistema, declarándolo en el archivo `/etc/modules` .

Ejemplo

En una distribución Debian 10:

```
cat /etc/modules
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
firewire-sbp2
```

En una distribución CentOS 8, ese archivo no existe.

f. Configuración de la carga de los módulos

Los archivos que se encuentran en el directorio `/etc/modprobe.d` definen las características de carga/descarga de los módulos.

También permiten definir **alias** para los nombres de los módulos, alias que pueden ser utilizados por los comandos de carga/descarga de los módulos (ver `man modprobe.d`).

La sintaxis de declaración de un alias es la siguiente:

```
alias NombreAlias NombreMódulo
```

Donde:

`NombreAlias` : alias que se puede usar en los comandos de gestión de módulos.

`NombreMódulo` : nombre del módulo.

g. Instalación manual de un módulo LKM

La mayor parte de las veces, los módulos nuevos se instalan y se configuran automáticamente gracias al paquete de software de una aplicación o de un driver de dispositivo.

Si fuera necesario hacer una instalación manual, habría que copiar el archivo que contiene el código ejecutable del módulo en uno de los directorios del directorio `/lib/modules/VersionNœcleo /kernel/` de la versión actual del núcleo.

Después habría que ejecutar el comando `depmod` , que va a controlar las dependencias entre todos los módulos y actualizar el archivo de gestión de esas dependencias, `modules.dep` , y su versión binaria `modules.dep.bin` .

h. Configuración de los módulos: modinfo, modprobe

El comando `modinfo NombreMódulo` muestra la información y la configuración autorizadas para el módulo `NombreMódulo` .

Ejemplo

Características y parámetros del módulo `cdrom`:

```
modinfo cdrom
filename:       /lib/modules/4.18.0-147.5.1.el8_1.x86_64/kernel/drivers/
cdrom/cdrom.ko.xz
```

```
license:        GPL
rhelversion:    8.1
srcversion:     E5A2049F635B552113240F5
depends:
intree:         Y
name:           cdrom
vermagic:       4.18.0-147.5.1.el8_1.x86_64 SMP mod_unload modversions
sig_id:         PKCS#7
signer:         CentOS Linux kernel signing key
sig_key:        6C:E4:44:06:AD:56:56:1C:FE:E9:7E:99:45:F8:69:0F:DF:1E:EA:FA
sig_hashalgo:   sha256
signature:      D2:28:97:3F:8D:4D:68:43:40:BD:EA:6B:D3:26:1C:C8:FC:90:E1:77:
                74:12:3D:9D:55:9B:73:AB:7C:2D:A9:FE:53:05:C8:16:CA:FA:86:4C:
[...]
parm:           debug:bool
parm:           autoclose:bool
parm:           autoeject:bool
parm:           lockdoor:bool
parm:           check_media_type:bool
parm:           mrw_format_restart:bool
```

El módulo acepta seis parámetros, todos ellos de tipo booleano (0 o 1).

Se pueden modificar dinámicamente los parámetros de un módulo LKM, usando el comando `modprobe`. Para ello es necesario descargarlo antes, y cargarlo de nuevo con el comando `modprobe` pasándole como argumentos los parámetros con su valor, según la sintaxis siguiente:

```
modprobe NombreMódulo parám1=Val1 ... parámN=ValN
```

Esta configuración es temporal, y será conservada hasta la siguiente carga del módulo.

2. Configuración dinámica usando /proc/sys

El sistema de archivos virtual proc, normalmente montado en el directorio `/proc`, permite comunicar dinámicamente con el núcleo y con los módulos de núcleo.

El directorio `/proc/sys` presenta una arborescencia de gestión de los parámetros del núcleo y de los módulos cargados en memoria. A través de diferentes pseudoarchivos, podemos leer o modificar estos parámetros dinámicamente. Estas modificaciones se hacen efectivas inmediatamente y son conservadas hasta la próxima modificación o hasta la próxima carga de los módulos y/o del núcleo.

Hay que ser prudente en caso de modificación, ya que un valor incorrecto puede comprometer el buen funcionamiento del núcleo o de los módulos que se estén utilizando actualmente.

Ejemplos

En una distribución CentOS 8, vamos a explorar el directorio `/proc/sys` y modificar algunos parámetros.

Directorio presentes en `/proc/sys` :

```
cd /proc/sys
ls -l
total 0
dr-xr-xr-x. 1 root root 0 23 oct 11:34 abi
dr-xr-xr-x. 1 root root 0 22 oct 17:51 crypto
dr-xr-xr-x. 1 root root 0 23 oct 11:34 debug
dr-xr-xr-x. 1 root root 0 23 oct 11:34 dev
dr-xr-xr-x. 1 root root 0 22 oct 17:51 fs
dr-xr-xr-x. 1 root root 0 22 oct 17:51 kernel
dr-xr-xr-x. 1 root root 0 22 oct 17:51 net
dr-xr-xr-x. 1 root root 0 23 oct 11:34 sunrpc
dr-xr-xr-x. 1 root root 0 23 oct 11:34 user
dr-xr-xr-x. 1 root root 0 22 oct 16:52 vm
```

El directorio

`/proc/sys/kernel` *da acceso a la configuración de diferentes parámetros del núcleo:*

cd kernel
ls
acct modprobe
perf_event_max_stack sched_wakeup_granularity_ns
acpi_video_flags modules_disabled
perf_event_mlock_kb seccomp
auto_msgmni msgmax
perf_event_paranoid sem
bootloader_type msgmnb
pid_max sem_next_id
bootloader_version msgmni
poweroff_cmd shmall
cad_pid msg_next_id
print-fatal-signals shmmax
cap_last_cap ngroups_max
printk shmmni
core_pattern nmi_watchdog
printk_delay shm_next_id
core_pipe_limit ns_last_pid
printk_devkmsg shm_rmid_forced
core_uses_pid numa_balancing
printk_ratelimit softlockup_all_cpu_backtrace
ctrl-alt-del numa_balancing_scan_delay_ms
printk_ratelimit_burst softlockup_panic
dmesg_restrict numa_balancing_scan_period_max_ms
pty soft_watchdog
domainname numa_balancing_scan_period_min_ms
random stack_tracer_enabled
firmware_config numa_balancing_scan_size_mb
randomize_va_space sysctl_writes_strict
ftrace_dump_on_oops osrelease
real-root-dev sysrq
ftrace_enabled ostype
sched_autogroup_enabled tainted
hardlockup_all_cpu_backtrace overflowgid
sched_cfs_bandwidth_slice_us threads-max
hardlockup_panic overflowuid
sched_child_runs_first timer_migration

```
hostname                panic
sched_domain            traceoff_on_warning
hung_task_check_count   panic_on_io_nmi
sched_latency_ns        tracepoint_printk
hung_task_panic         panic_on_oops
sched_migration_cost_ns unknown_nmi_panic
hung_task_timeout_secs  panic_on_rcu_stall
sched_min_granularity_ns unprivileged_bpf_disabled
hung_task_warnings      panic_on_stackoverflow
sched_nr_migrate        usermodehelper
io_delay_type           panic_on_unrecovered_nmi
sched_rr_timeslice_ms   version
kexec_load_disabled     panic_on_warn
sched_rt_period_us      watchdog
keys                    perf_cpu_time_max_percent
sched_rt_runtime_us     watchdog_cpumask
kptr_restrict           perf_event_max_contexts_per_stack
sched_schedstats        watchdog_thresh
max_lock_depth          perf_event_max_sample_rate
sched_tunable_scaling   yama
```

La mayoría de estos elementos son pseudoarchivos, accesibles para lectura y escritura.

Nombre de red del sistema local:

```
cat hostname
centos8
```

Versión del núcleo:

```
cat osrelease
4.18.0-147.5.1.el8_1.x86_64
```

Valor máximo de una identificador de proceso:

```
cat pid_max
```

```
32768
```

El directorio `/proc/sys/net/` da acceso a numerosos parámetros de la gestión de red.

Por defecto, el sistema no responde a las solicitudes ICMP en broadcast, solicitadas por el comando `ping -b`:

```
ping -b -c1 127.0.0.0
WARNING: pinging broadcast address
PING 127.0.0.0 (127.0.0.0) 56(84) bytes of data.
--- 127.0.0.0 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

El sistema no responde, aunque su dirección de loopback, `127.0.0.1`, pertenezca a la red solicitada.

Efectivamente, el parámetro `icmp_echo_ignore_broadcasts` está posicionado a `1`:

```
cat /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts 1
```

Si pasamos este parámetro a `0` y volvemos a lanzar el comando:

```
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
ping -b -c1 127.0.0.0
WARNING: pinging broadcast address
PING 127.0.0.0 (127.0.0.0) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.073 ms
--- 127.0.0.0 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.073/0.073/0.073/0.000 ms
```

El sistema responde ahora a las solicitudes ICMP en broadcast.

3. El comando sysctl

Un método más seguro para gestionar la configuración dinámica del núcleo y de los módulos consiste en usar el comando `sysctl`. Este controla los parámetros antes de modificarlos a través del sistema de archivos virtual proc.

a. Visualización de los parámetros del núcleo y de los módulos

La opción `-a` del comando `sysctl` muestra el conjunto de los parámetros del núcleo y de los módulos LKM cargados. Muestra un camino de acceso para cada elemento en la raíz `/proc/sys`, reemplazando el / separador por un punto.

Ejemplo

```
sysctl -a | head
abi.vsyscall32 = 1
crypto.fips_enabled = 0
debug.exception-trace = 1
debug.kprobes-optimization = 1
dev.cdrom.autoclose = 1
dev.cdrom.autoeject = 0
```

El elemento `dev.cdrom.autoeject` corresponde al archivo `/proc/sys/dev/cdrom/autoeject`:

```
cat /proc/sys/dev/cdrom/autoeject
0
```

El comando `sysctl CaminoParÆm` muestra la información con respecto al objeto cuyo camino se ha pasado como argumento. Si se trata de un parámetro, mostrará su valor; si se trata de un módulo, mostrará todos sus parámetros y sus valores. El camino usa el formato presentado más adelante.

Ejemplo

Un parámetro del driver de CD-ROM:

```
sysctl dev.cdrom.autoclose
dev.cdrom.autoclose = 1
```

Parámetros de red:

```
sysctl net
net.core.bpf_jit_enable = 1
net.core.bpf_jit_harden = 1
net.core.bpf_jit_kallsyms = 1
net.core.bpf_jit_limit = 264241152
net.core.busy_poll = 0
net.core.busy_read = 0
net.core.default_qdisc = fq_codel
net.core.dev_weight = 64
net.core.dev_weight_rx_bias = 1
net.core.dev_weight_tx_bias = 1
net.core.fb_tunnels_only_for_init_net = 0
[...]
net.ipv4.icmp_echo_ignore_broadcasts = 0
[...]
```

b. Modificación de los parámetros del núcleo y de los módulos

La sintaxis siguiente permite asignar dinámicamente un nuevo valor a uno o distintos parámetros:

```
sysctl CaminoParám1=Val1 ... CaminoParámN=ValN
```

Estas modificaciones son inmediatas pero temporales. Para que sean aplicadas durante el arranque del sistema y por lo tanto hacer que sean permanentes, habrá que declararlas usando el archivo de configuración `/etc/sysctl.conf` .

Los parámetros que se tendrán que forzar, tienen que estar declarados con la sintaxis siguiente:

```
CaminoParám = Valor
```

Donde `CaminoParÆm` respeta el formato usado por el comando `sysctl`.

Dependiendo de las distribuciones, hay que declarar los parámetros y su valor directamente en el archivo `/etc/sysctl.conf` o en un archivo del directorio `/etc/sysctl.d` (ver `man sysctl.d`).

Ejemplo

En una distribución CentOS 8, las modificaciones permanentes de los parámetros tienen que ser declaradas en un archivo dentro del directorio `/etc/sysctl.d`:

Forzar el parámetro `icmp_echo_ignore_broadcasts` a `0`:

```
vi /etc/sysctl.d/99-sysctl.conf
# Responder a las solicitudes icmp broadcast
net.ipv4.icmp_echo_ignore_broadcasts=0
```

Después de haber reiniciado el sistema:

```
sysctl net.ipv4.icmp_echo_ignore_broadcasts
net.ipv4.icmp_echo_ignore_broadcasts = 0
ping -b -c1 127.0.0.0
WARNING: pinging broadcast address
PING 127.0.0.0 (127.0.0.0) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.082 ms
--- 127.0.0.0 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.082/0.082/0.082/0.000 ms
```

El sistema responde a las solicitudes ICMP en broadcast.