Compilación del kernel Linux

Explicación de práctica 1

Sistemas Operativos

Facultad de Informática
Universidad Nacional de La Plata

2024

Agenda

Contenido:

- Fundamentos
- Historia
- Versionado
- Compilación
- Apéndice
  - Debian way

Agenda

Contenido:

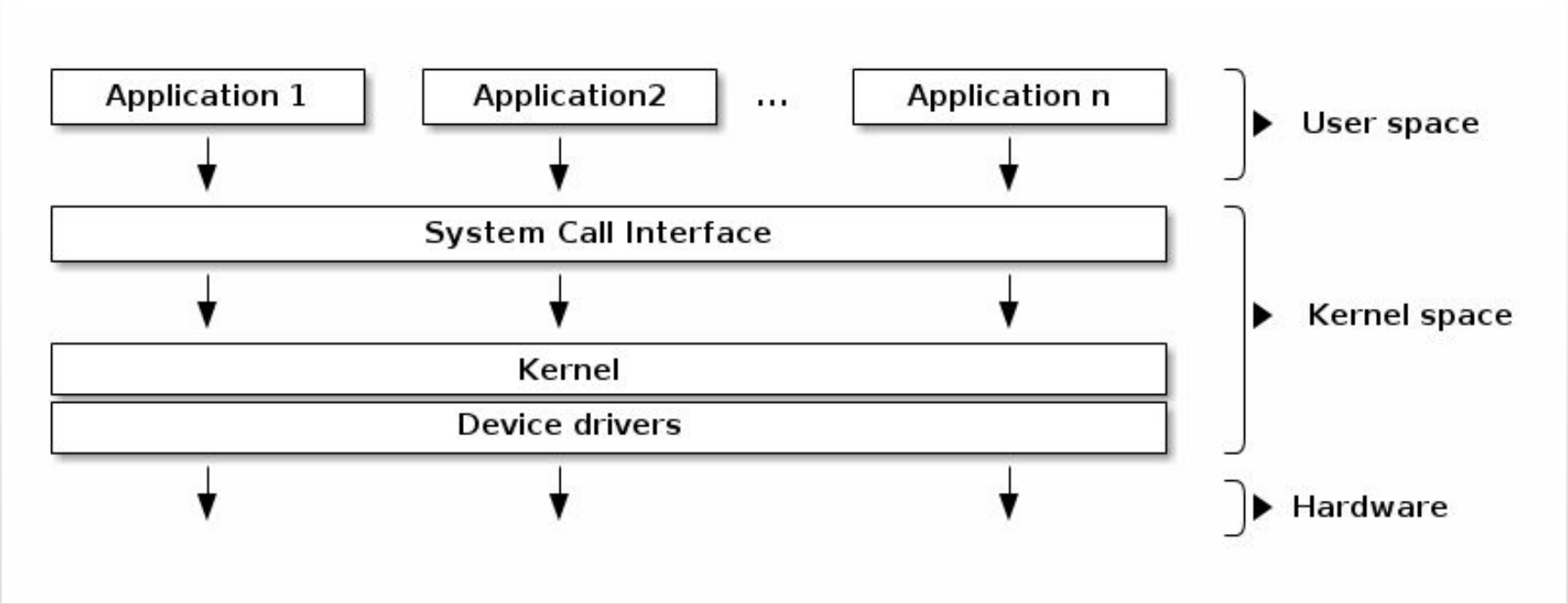
- Fundamentos
- Historia
- Versionado
- Compilación
- Apéndice
  - Debian way

¿Qué es el kernel Linux?

- Programa que ejecuta programas y gestiona dispositivos de hardware
- Encargado de que el software y el hardware puedan trabajar juntos
- Principales funciones
  - Administración de memoria principal
  - Administración de uso de la CPU
- Es de código abierto a los usuarios (kernel/sched.c)
- En una misma estructura de código fuente se da soporte a todas las arquitecturas
- Liberado bajo licencia GPLv2
- En un sentido estricto es el Sistema Operativo

Funcionalidad del kernel

Arquitectura típica:



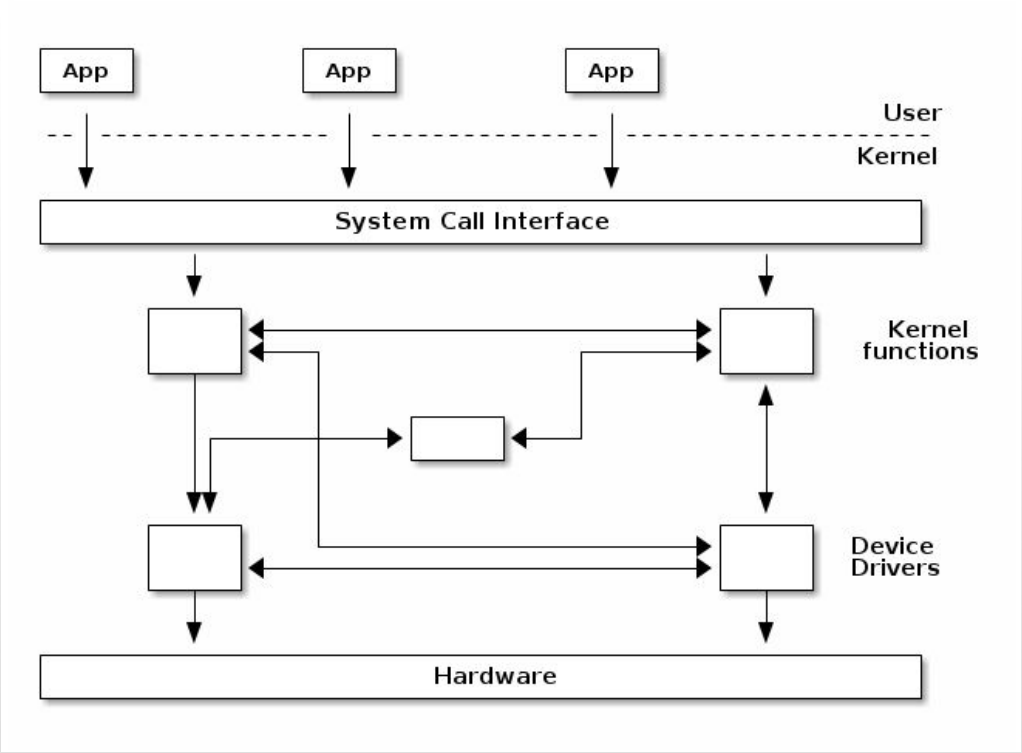
Fuente: https://linux-kernel-labs.github.io

Tipos de kernel - Monolítico

A monolithic kernel is one where there is no access protection between the various kernel subsystems and where public functions can be directly called between various subsystems.

Fuente: https://linux-kernel-labs.github.io

Tipos de kernel - Monolítico



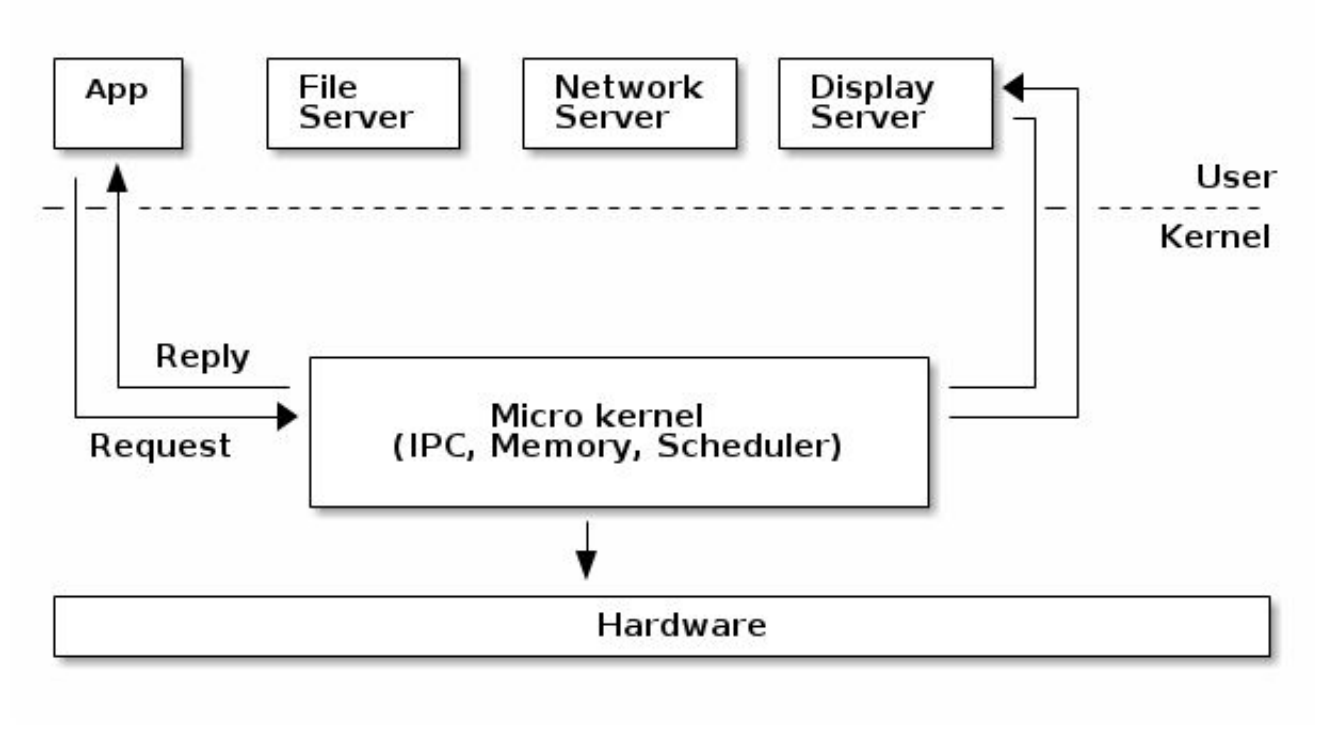
Fuente: https://linux-kernel-labs.github.io

Tipos de kernel - Micro kernel

A micro-kernel is one where large parts of the kernel are protected from each-other, usually running as services in user space. Because significant parts of the kernel are now running in user mode, the remaining code that runs in kernel mode is significantly smaller, hence micro-kernel term.
In a micro-kernel architecture the kernel contains just enough code that allows for message passing between different running processes. Practically that means implement the scheduler and an IPC mechanism in the kernel, as well as basic memory management to setup the protection between applications and services.

Fuente: https://linux-kernel-labs.github.io

Tipos de kernel - Micro kernel



Fuente: https://linux-kernel-labs.github.io

¿Qué es el kernel Linux?

- Es un núcleo monolítico híbrido
  - Los drivers y el código del Kernel se ejecutan en modo privilegiado
  - Lo que lo hace híbrido es la posibilidad de cargar y descargar funcionalidad a través de módulos

Software Libre
Se dispone la libertad de: 1. Usar 2. Estudiar 3. Distribuir 4. Mejorar y publicar

Agenda

Contenido:

- Fundamentos
- Historia
- Versionado
- Compilación
- Apéndice
  - Debian way

Linux, un poco de historia

En 1991 Linus Torvalds inicia la programación del kernel Linux basado en Minix[2]

(Clon de Unix desarrollado por Tanembaum en 1987 con el fin de crear un SO de uso educativo).

Primer anuncio (1991)

Hello everybody out there using minix -
I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones.
[. . .]
It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Linus Torvalds, 25/08/1991, comp.os.minix[3]

Linux, un poco de historia

El 5 de octubre de 1991, se anuncia la primera versión “oficial” de Linux (0.02).

Primera release (1991)

[. . .]
As I mentioned a month ago, I’m working on a free version of a minix-lookalike for AT-386 computers. It has finally reached the stage where it’s even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02 (+1 (very small) patch already), but I’ve successfully run bash/gcc/gnu-make/gnu-sed/compress etc under it.
[. . .]

Linus Torvalds, 05/10/1991, comp.os.minix[3]

En 1992, con la release de la versión 0.12, se decide cambiar a una licencia GNU.

Licencia GNU (1992)

[. . .]
The Linux copyright will change: I've had a couple of requests to make it compatible with the GNU copyleft, removing the "you may not distribute it for money" condition. I agree. I propose that the copyright be changed so that it confirms to GNU - pending approval of the persons who have helped write code. I assume this is going to be no problem for anybody: If you have grievances ("I wrote that code assuming the copyright would stay the same") mail me.
Otherwise The GNU copyleft takes effect as of the first of February. If you do not know the gist of the GNU copyright - read it.
[. . .]

Linus Torvalds, 1992 [4]

Linux, un poco de historia

Fechas relevantes:

- En marzo de 1994 Torvalds considera que todos los componentes del kernel estaban suficientemente maduros y lanza la versión 1.0.
- En el año 1995 Linux se porta a arquitecturas DEC Alpha y Sun SPARC. Con el correr de los años se portó a otra decena de arquitecturas.
- En mayo de 1996 se decide adoptar a Tux como mascota oficial de Linux.

Linux, un poco de historia

Fechas relevantes:

- En julio de 1996 se lanza la versión 2.0 y se define un sistema de nomenclatura.
- Se desarrolló hasta febrero de 2004 y terminó con la versión 2.0.40. Esta versión comenzó a brindar soporte a sistemas multiprocesadores.
- En 2001 se lanza la versión 2.4 y se deja de desarrollar a fines del 2010 con la 2.4.37.11. La versión 2.4 fue la que catapultó a GNU/Linux como un sistema operativo estable y robusto.

Linux, un poco de historia

Fechas relevantes:

- A fines del año 2003 se lanza la versión 2.6. Esta versión ha tenido muchas mejoras para el *kernel* dentro de las que se destacan soporte de *threads*, mejoras en la planificación y soporte de nuevo hardware.
- El 3 de agosto de 2011 se lanza la versión 2.6.39.4 anunciándose la misma desde meses previos como la última en su revisión.
- El 17 Julio de 2011 se lanza la versión 3.0.
  - No agrega mayores cambios. La decisión del cambio son los 20 años del SO y no superar los 40 números de revisión.
  - Totalmente compatible con Kernel 2.6.

Linux, un poco de historia

- El 17 Julio de 2011 se lanza la versión 3.0
  - Termina con la versión 3.8.30
  - Provee mejoras en Virtualización y FileSystems

Anuncio de Linux 3.0 (2011)

[. . .]
I decided to just bite the bullet, and call the next version 3.0. It will get released close enough to the 20-year mark, which is excuse enough for me, although honestly, the real reason is just that I can no longer comfortably count as high as 40.
[. . .]

Linus Torvalds, 2011 [1]

Linux, un poco de historia

- El 12 de Abril de 2015 se lanza la versión 4.0
  - Una de sus principales mejoras es la posibilidad de aplicar parches y actualizaciones sin necesidad de reiniciar el SO.
  - Soporte para nuevas CPU

Anuncio de Linux 4.0 (2015)

[. . .]
.. Because the people have spoken, and while most of it was complete gibberish, numbers don't lie. People preferred 4.0, and 4.0 it shall be. Unless somebody can come up with a good argument against it.
[. . .]

Linus Torvalds, 2015

Linux, un poco de historia

- El 3 de Marzo de 2019 se lanza la versión 5.0
  - Soporte de energy-aware scheduling que despierta tareas en las CPUs de menor consumo energético en smartphones.
  - Soporte de namespaces para binderfs que permite ejecutar múltiples instancias de Android.
  - Soporte de archivos swap en BTRFS.
  - Kernel Lockdown Mode previene el acceso de procesos de usuario (incluso root) a memoria del kernel. (Habilitado por defecto en modo Secure Boot de EFI) - 5.4 (24 Noviembre 2019)
  - Soporte inicial de USB 4 - 5.6 (29 de Marzo 2020)
  - Nuevo mecanismo para manejar systemcalls de Windows para programas como Wine - 5.11 (14 de Febrero 2021)
  - Landlock security module: Permite restringir las acciones que un conjunto de programas puede ejecutar en un filesystem de forma simple - 5.13 (27 de Junio de 2021)
  - Nueva system call para crear áreas de memoria secretas que no pueden ser accedidas ni por el usuario root (por ejemplo para guardar claves) - 5.14 (29 de Agosto de 2021)
  - Soporte opcional para paquetes IPv6 de más de 64KB - 5.19 (31 de Julio de 2021)

Linux, un poco de historia

- El 2 de Octubre de 2022 se lanza la versión 6.0
  - Primer release que soporta módulos escritos en Rust - 6.1 (11 de Diciembre de 2022)
  - Mejoras en la confiabilidad de la implementación de RAID5/6 en Btrfs - 6.2 (19 de Febrero 2023)
  - Mejoras de rendimiento y fragmentación para Btrfs 6.3 (23 de Abril 2023)
  - Soporte inicial para USB4.0 v2 - 6.5 (27 de Agosto 2023)
  - Nuevo scheduler de tareas (EEVDF) que mejora el fairness respecto al algoritmo anterior (CFS) Linux 6.6 (29 de Octubre de 2023)
  - Soporte para un nuevo Filesystem BCachefs con características similares a ZFS y Btrfs - 6.7 (7 de Enero de 2024)
  - Optimización de las estructuras de datos usadas para networking que mejora la performance de TCP cuando hay muchas conexiones concurrentes en hasta un 40% - 6.8 (10 de Marzo de 2023)

Linux, un poco de historia

Fuentes para los datos de 5.x y 6.x:

- https://kernelnewbies.org/LinuxVersions
- https://man7.org/linux/man-pages/man7/kernel_lockdown.7.html
- https://docs.kernel.org/userspace-api/landlock.html
- https://lwn.net/Articles/826313/

Agenda

Contenido:

- Fundamentos
- Historia
- Versionado
- Compilación
- Apéndice
  - Debian way

Versionado previo a la versión 2.6

Anatomía de una versión de Linux < 2.6:

X.Y.Z
● **X** Indicaba la serie principal. Cambiaba al agregar/quitar una funcionalidad muy importante. ● **Y** Indicaba si era una versión de producción o desarrollo. ● **Z** Bugfixes. Existían dos versiones del kernel: ● Números **Y** pares indicaban una versión en Producción (estable) ● Números **Y** impares indicaban una versión en Desarrollo

Versionado en Linux 2.6 a 3.0

- **Anatomía de una versión de Linux >= 2.6 y < 3.0:**

A.B.C.[D]
• **A** Denota Versión. Cambia con menor Frecuencia (cada varios años). • **B** Denota revisión mayor. • **C** Denota revisión menor. Solo cambia cuando hay nuevos drivers o características. • **D** Se utiliza cuando se corrige un grave error sin agregar nueva funcionalidad.

Versionado en 3.0 y superiores

Anatomía de una versión de >= 3.0:

- En el año 2011, cuando se pasó de la versión 2.6.39 a la 3.0 se volvió a un esquema de 3 números:

A.B.C[-rcX]
• **A** Denota revisión mayor. Cambia con menor Frecuencia (cada varios años). • **B** Denota revisión menor. Solo cambia cuando hay nuevos drivers o características. • **C** Número de revisión • **rcX** Versiones de prueba

Agenda

Contenido:

- Fundamentos
- Historia
- Versionado
- Compilación
- Apéndice
  - Debian way

¿Por qué recompilarlo?

- Soportar nuevos dispositivos como, por ejemplo, una placa de video
- Agregar mayor funcionalidad (soporte de nuevos filesystems)
- Optimizar funcionamiento de acuerdo al sistema en el que corre
- Adaptarlo al sistema donde corre (quitar soporte de hardware no utilizado)
- Corrección de bugs (problemas de seguridad o errores de programación)

¿Qué necesitamos?

- gcc: Compilador de C
- make: ejecuta las directivas definidas en los Makefiles
- binutils: assembler, linker
- libc6: Archivos de encabezados y bibliotecas de desarrollo
- ncurses: bibliotecas de menú de ventanas (solo si usamos menuconfig)
- initrd-tools: Herramientas para crear discos RAM

Tip

En Debian y distribuciones derivadas, todo este software se encuentra empaquetado.
Por ejemplo, para instalar el software requerido para hacer la práctica:

```
# apt-get install build-essential libncurses-dev
kbuild flex bison libelf-dev bc
```

Proceso de compilación de Linux

1. Obtener el código fuente.
2. Preparar el árbol de archivos del código fuente.
3. Configurar el kernel
4. Construir el kernel a partir del código fuente e instalar los módulos.
5. Reubicar el kernel.
6. Creación del initramfs
7. Configurar y ejecutar el gestor de arranque (GRUB en general).
8. Reiniciar el sistema y probar el nuevo kernel.

Obtener el código fuente

http://www.kernel.org

Proceso de compilación de Linux

Para descargarlo desde la terminal:

```
$ cd /usr/src
$ sudo wget https://kernel.org/pub/linux/kernel/v6.x/linux-6.7.tar.xz
```

Opcional: Por cuestiones de seguridad, opcionalmente se puede descargar un archivo con una firma criptográfica del kernel descargado. Esta firma nos permite comprobar que el archivo que descargamos es exactamente el mismo que fue subido, y que no hubo modificaciones maliciosas de por medio.

```
$ gpg --locate-keys torvalds@kernel.org gregkh@kernel.org
$ wget https://kernel.org/pub/linux/kernel/v6.x/linux-6.7.tar.sign
$ xz -cd linux-6.7.tar.xz | gpg --verify linux-6.7.tar.sign -
gpg: Firmado el lun 08 ene 2024 02:47:49 -03
gpg:                usando RSA clave 647F28654894E3BD457199BE38DBBDC86092693E
gpg: Firma correcta de "Greg Kroah-Hartman <gregkh@kernel.org>" [desconocido]
gpg: ATENCIÓN: ¡Esta clave no está certificada por una firma de confianza!
gpg:          No hay indicios de que la firma pertenezca al propietario.
Huellas dactilares de la clave primaria: 647F 2865 4894 E3BD 4571  99BE 38DB BDC8
6092 693E
```

Preparar el árbol de archivos

Por convención el código fuente del kernel se guarda en /usr/src. Sin embargo, como dicho directorio generalmente no tiene permisos de escritura para usuarios no privilegiados, el archivo se puede descomprimir en un directorio donde tengamos permisos, como el $HOME del usuario actual.

```
$ mkdir $HOME/kernel
$ cd $HOME/kernel
$ tar xvf /usr/src/linux-6.7.tar.xz
```

Generalmente se crea un enlace simbólico llamado linux apuntando al directorio del código fuente que actualmente se está configurando

```
$ ln -s /usr/src/linux-6.7 /usr/src/linux
```

Configurar el kernel

El kernel Linux se configura mediante el archivo .config. Este archivo, que reside en el directorio de código fuente del kernel, contiene las instrucciones de qué es lo que el kernel debe compilar. También se puede ver la configuración de los kernels instalados en /boot:

```
$ cat /boot/config-$(uname -r) | tail -n5
CONFIG UCS2 STRING=y
CONFIG FONT SUPPORT=y
# CONFIG FONTS is not set
CONFIG FONT 8x8=y
CONFIG FONT 8x16=y
```

Existen tres interfaces que permiten generar este archivo:

- make config: modo texto y secuencial. Tedioso.
- make xconfig: interfaz gráfica utilizando un sistema de ventanas.No todos los sistemas tienen instalado X.
- make menuconfig: este modo utiliza ncurses, una librería que permite generar una interfaz con paneles desde la terminal. Generalmente el más utilizado.

Configurar el kernel

Las herramientas mencionadas permiten:

- Crear un archivo .config con las directivas de compilación
- Configurar un kernel desde cero es una tarea tediosa y propensa a errores (kernels que no arranquen). Estas herramientas automatizan el proceso por nosotros.
- Consejos
  - Lo ideal es ir manteniendo el .config para no tener que configurar todo de cero
  - Cada nueva versión, puede valerse de un .config anterior.
  - Por convención, es recomendable almacenar en el directorio /boot la imagen compilada del kernel junto con su .config.

Configurar el kernel (Módulos)

¿Qué es un módulo del Kernel?

- Un fragmento de código que puede cargarse/descargarse en el mapa de memoria del SO (Kernel) bajo demanda
- Permiten extender la funcionalidad del Kernel en “caliente” (sin necesidad de reiniciar el sistema)
- Todo su código se ejecuta en modo Kernel (privilegiado)
- Cualquier error en el módulo, puede colgar el SO
- Permiten que el kernel se desarrolle bajo un diseño más modular
- Los módulos disponibles se ubican en /lib/modules/version del kernel
- Con el comando lsmod es posible ver qué módulos están cargados
- Vamos a ampliar en las próximas prácticas

Configurar el kernel (Módulos)

¿El soporte lo damos como módulo o built-in?

- Si es built-in, el kernel crece. Mayor uso de memoria. Puede incrementar el tiempo de arranque.
- Si es built-in, es más eficiente su utilización. No hay que cargar un adicional en memoria, acceso directo.
- En el soporte como módulo:
  - Si se quiere dar soporte a algún dispositivo, se carga el módulo y no es necesario recompilar (el soporte debe estar en el kernel).
  - Si hay una modificación de un driver, solo se modifica el módulo y no todo el código del kernel.
  - Los módulos se cargan bajo demanda, con lo cual la utilización de memoria es menor.

Parcheando el kernel

- Patch es un mecanismo que permite aplicar actualizaciones sobre una versión base.
- Se basa en archivos diff (archivos de diferencia), que indican qué agregar y qué quitar.
- En kernel.org hay de 2 tipos:
  - No incrementales: Se aplican sobre la versión mainline anterior (X.0.0). Cómo el que usarán en la práctica.
  - Incrementales: Se aplican sobre la versión inmediatamente anterior.
- Permiten agregar funcionalidad (nuevos drivers, correcciones menores, etc.)
- A veces puede resultar más sencillo descargar el archivo de diferencia y aplicarlo en vez de descargar todo el código de la nueva versión.

```
$ cd linux; zcat ../patch-6.8.gz | patch -p1
```

Tip

Parámetro útil: --dry-run

Configurar el kernel

En el trabajo práctico debemos dar el siguiente soporte adicional:

• Soporte para FS BTRFS: En la opción File Systems, tendremos que seleccionar Btrfs filesystem

• Soporte para dispositivos de Loopback: En la opción Device Drivers → Block Devices, tendremos que seleccionar Loopback device support.

Compilación del kernel

```
$ make
```

El comando make busca el archivo Makefile, interpreta sus directivas y compila el kernel. Este proceso puede durar mucho tiempo dependiendo del procesador que tengamos.

Verificar que este proceso no arroje errores al concluir.

Tip

```
$ make -jX # X es el número de threads
```

Compilación de módulos

Compilando

```
$ make modules
```

El comando make modules compila todos los módulos necesarios para satisfacer las opciones que hayan sido seleccionadas como módulo.

Tip
Generalmente la tarea anterior se encuentra incluida en la compilación del kernel con el comando make

Instalación del kernel

Recién en este paso es necesario convertirse en root.
Al terminar el proceso de compilación, la imagen del kernel quedará ubicada en directorio-del-código/arch/arquitectura/boot/.
El próximo paso, entonces, es instalar el kernel y otros archivos en el directorio /boot.
Ejemplo con arquitectura i386

```
$ cd $HOME/kernel/linux-6.7
$ sudo su
# cp arch/i386/boot/bzImage /boot/vmlinuz-6.8
# cp System.map /boot/System.map-6.8
# cp .config /boot/config-6.8
```

Por supuesto, también existe una regla en el Makefile que realiza esto mismo automáticamente.

```
$ sudo make install
```

System.map: http://rlworkman.net/system.map/

Instalar los módulos

Los módulos compilados deben residir en el directorio /lib/modules/version-del-kernel.
Al igual que en el paso anterior, el archivo Makefile tiene una regla para instalar los módulos.

```
$ sudo make modules install
```

El parámetro modules install es una regla del Makefile que ubica los módulos del kernel recién compilado en el directorio correspondiente.

Creación del initramfs

- Un initramfs es un sistema de archivos temporal que se monta durante el arranque del sistema.
- Contiene ejecutables, drivers y módulos necesarios para lograr iniciar el sistema.
- Luego del proceso de arranque el disco se desmonta.

```
# mkinitramfs -o /boot/initrd.img-6.8 6.8
```

Configuración del gestor de arranque

- En el trabajo práctico utilizaremos la versión 2 de GRUB.
- Luego de instalar el kernel, para que el gestor de arranque lo reconozca simplemente deberemos ejecutar, como usuario privilegiado, el siguiente comando:

```
# update-grub2
```

Finalización del proceso

- Verificar que el kernel esté instalado en /boot:

```
$ test -f /boot/vmlinuz-6.8 && echo ‘‘Kernel instalado’’
```

- Verificar que los módulos estén instalados en /lib/modules/6.8

```
$ test -d /lib/modules/6.8-arquitectura && echo ‘‘Modulos
aparentemente instalados’’
```

- Verificar que el gestor de arranque haya indexado el kernel. Si el gestor de arranque utilizado es GRUB 2, entonces el siguiente comando debería generar un par de líneas de salida.

```
$ cat /boot/grub/grub.cfg | grep --color 6.8
```

- Paso final: ¡reiniciar y probar!

Agenda

Contenido:

- Fundamentos
- Historia
- Versionado
- Compilación
- Apéndice
  - Debian way

Debian way

Instalar un kernel de forma empaquetada ofrece muchas ventajas:
- Integración con el sistema operativo
- Bugfixes
- Actualizaciones de seguridad
- LTS (Long Term Support)

Debian designa una versión de Linux a la cual dar soporte durante todo el ciclo de vida de su versión estable, actualmente Debian 12 bookworm. Esto implica portar actualizaciones críticas y bugfixes. La versión de Linux que está en la actual Debian estable es la 6.1.

```
# apt-get install linux-source-6.1
$ tar xjf /usr/src/linux-source-6.1.tar.bz2
$ cd linux-source-6.1
$ make menuconfig
$scripts/config --disable DEBUG INFO
$ make clean
$ make deb-pkg
# dpkg -i ../linux-image-6.1.*.deb
```

Referencias

Linus Torvalds
Linux 3.0-rc1 https://lkml.org/lkml/2011/5/29/204, 2011
Andrew Tanembaum
Minix http://www.minix3.org/
Linus Torvalds
Linux History https://www.cs.cmu.edu/~awb/linux.history.html, 1992
Linus Torvalds
Release notes for Linux v0.12
http://ftp.funet.fi/pub/linux/historical/kernel/old-versions/RELNOTES-0.12, 1992
Debian Linux Kernel Handbook
http://kernel-handbook.alioth.debian.org/

¿Preguntas?