Sistemas operativos

Práctica 2

**1. ¿Qué es el kernel de GNU/Linux? ¿Cuáles son sus funciones principales dentro del Sistema**

**Operativo?**

Es el programa medular que ejecuta aplicaciones y gestiona dispositivos de hardware tales como los discos y las impresoras. Linux, desde un punto estricto es un kernel, no un sistema operativo. El sistema operativo está formado por el kernel junto con todas las herramientas necesarias para que la computadora pueda operar. De poco sirve un kernel sin un shell, ni ambiente gráfico, ni herramientas de administración. El kernel es el encargado de que el software y el hardware de nuestras computadoras puedan trabajar juntos y de acuerdo a lo que el usuario necesite

Las funciones más importantes del Kernel son:

* Administrar de la memoria para todos los programas en ejecución.
* Administrar el tiempo de procesador que los programas en ejecución utilizan.
* Es el encargado de que podamos acceder a los periféricos/elementos de nuestro ordenador de una manera cómoda.

**2. Indique una breve reseña histórica acerca de la evolución del kernel de GNU/Linux**

En 1991 Linus Torvalds inicia la programacion del kernel Linux basado en Minix[2] (Clon de Unix desarrollado por Tanembaum en 1987 con el n de crear un SO de uso didactico).

El 5 de octubre de 1991, se anuncia la primera version \ocial" de

Linux (0.02).

En 1992, con la release de la version 0.12, se decide cambiar a una

licencia GNU.

Fechas relevantes:

* En marzo de 1994 Torvalds considera que todos los componentes del kernel estaban sucientemente maduros y lanza la version 1.0.
* En el a~no 1995 Linux se porta a arquitecturas DEC Alpha y Sun SPARC. Con el correr de los a~nos se porto a otra decena de arquitecturas.
* En mayo de 1996 se decide adoptar a Tux como mascota oficial de Linux.
* En julio de 1996 se lanza la version 2.0 y se dene un sistema de nomenclatura. Se desarrollo hasta febrero de 2004 y termino con la version 2.0.40. Esta version comenzo a brindar soporte a sistemas multiprocesadores.
* En 2001 se lanza la version 2.4 y se deja de desarrollar a fines del 2010 con la 2.4.37.11. La version 2.4 fue la que catapulto a GNU/Linux como un sistema operativo estable y robusto.
* A fines del a~no 2003 se lanza la version 2.6. Esta version ha tenido muchas mejoras para el kernel dentro de las que se destacan soporte de threads, mejoras en la planicacion y soporte de nuevo hardware.
* El 3 de agosto de 2011 se lanza la version 2.6.39.4 anunciandose la misma desde meses previos como la ultima en su revision.
* El 17 Julio de 2011 se lanza la version 3.0
* No agrega mayores cambios. La decision del cambio son los 20 años del SO y no superar los 40 numeros de revision
* Totalmente compatible con Kernel 2.6
* El 17 Julio de 2011 se lanza la version 3.0 Termina con la version 3.8.30. Provee mejoras en Virtualizacion y FileSystems
* El 12 de Abril de 2015 se lanza la version 4.0. Una de sus principales mejoras es la posibilidad de aplicar parches y actualizaciones si necesidad de reiniciar el SO. Soporte para nuevas CPU.
* La version actual es 4.5

**3. Explique brevemente la arquitectura del kernel de GNU/Linux teniendo en cuenta: tipo de**

**kernel, módulos, portabilidad, etc.**

Linux es un kernel monolítico híbrido, se dice híbrido porque a diferencia de los kernel monolíticos tradicionales las extensiones del núcleo y los controladores de dispositivos se pueden cargar y descargar fácilmente como módulos, mientras que el sistema operativo sigue trabajando sin interrupciones. Los módulos son bloques de código que extienden la funcionalidad del kernel y pueden corresponder tanto a extensiones del kernel como a controladores de dispositivos. Estos pueden ser cargados o descargados del kernel sin la necesidad de interrumpir al SO.

Para dar soporte para determinadas características se puede utilizar dar este soporte mediante módulos o embebido, si se utiliza como embebido, se gana performance ya que se realiza un acceso directo a memoria, pero tiene como desventaja que el kernel crece en cuanto a tamaño. En cambio, si este soporte se quiere dar por módulos, tiene la ventaja de que se carga el módulo en tiempo de ejecución y no se precisa recompilar el kernel, pero conlleva a una menor performance debido a que no se realiza acceso directo a memoria para referenciar el módulo.

**4. ¿Cuáles son los cambios que se introdujeron en el kernel a partir de la versión 3.0? ¿Cuál fue la razón por la cual se cambió de la versión 2 a la 3? ¿Y la razón para el cambio de la versión 3 a la 4?**

v3)

* Provee mejoras en Virtualización y FileSystems

v2 a v3)

* 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
* Termina con la versión 2.6.39

v3 a v4)

* Una de sus principales mejoras es la posibilidad de aplicar parches y actualizaciones si necesidad de reiniciar el SO y Soporte para nuevas CPU

**5. ¿Cómo se define el versionado de los kernels de GNU/Linux?**

Hasta que empezó el desarrollo de la serie 2.6 del núcleo, existieron dos tipos de versiones del núcleo:

* Versión de producción: La versión de producción, es la versión estable hasta el momento. Esta versión es el resultado final de las versiones de desarrollo o experimentales. Cuando el equipo de desarrollo del kernel experimental, decide que ha conseguido un kernel estable y con la suficiente calidad, se lanza una nueva versión de producción o estable. Esta versión es la que se debería utilizar para un uso normal del sistema, ya que son las versiones consideradas más estables y libres de fallos en el momento de su lanzamiento.
* Versión de desarrollo: Esta versión es experimental y es la que utilizan los desarrolladores para programar, comprobar y verificar nuevas características, correcciones, etc. Estos núcleos suelen ser inestables y no se deberían usar sin ser completamente consciente de los problemas con los que uno se puede encontrar.

Los números de versiones por debajo del **2.6** se interpretaban de la siguiente manera:,las versiones del núcleo se numeraban con 3 numeros, de la siguiente forma: X.Y.Z :

* *X*: Indicaba la serie/versión principal del núcleo. Solo han existido la 1 y 2. Este número cambiaba cuando la manera de funcionamiento del kernel habia sufrido un cambio muy importante.
* *Y*: Indicaba si la versión era de desarrollo ó de producción. Un número impar, significaba que era de desarrollo, uno par, que era de producción.
* *Z*: Indicaba nuevas revisiones dentro de una versión, en las que lo único que se había modificado eran fallos de programación.

Con la *serie 2.6* del núcleo, el sistema de numeración asi como el modelo de desarrollo han cambiado. Las versiones han pasado a numerarse con 4 digitos y no existen versiones de producción y desarrollo. Las versiones del núcleo se numeran hoy en dia con 4 dígitos, de la siguiente forma: A.B.C.D.

* A: Indica la serie/versión principal del núcleo.
* B: Indica la revisión principal del núcleo. Números pares e impares no tienen ningún significado hoy en dia.
* C: Indica nuevas revisiones menores del núcleo. Cambia cuando nuevas características y drivers son soportados.
* D: Este dígito cambia cuando se corrigen fallos de programación o fallos de seguridad dentro de una revisión.

**A** Denota Version. Cambia con menor Frecuencia. en 1994 (versión 1.0), en 1996 (versión 2.0), en 2011 (versión 3.0) y en en 2015 (versión 4.0).

**B** Denota revision mayor. Antes de la versión 2.6, los números impares indican desarrollo, los pares producción.

**C** Denota revision menor. Solo cambia cuando hay nuevos drivers o caractersticas.

**D** Se utiliza cuando se corrige un grave error sin agregar nueva funcionalidad.

En el año 2011, cuando se pasó de la versión 2.6.39 a la 3.0, se eliminó el número de revisión mayor (B)[1].

**X** Indica serie principal. Cambia cuando su funcionalidad sufre un cambio muy importante.

**Y** Indica si es una version de produccion o desarrollo.

**Z** Nuevas versiones dentro de la actual. Bugxes.

Existan dos versiones del kernel:

 Numeros Y pares indicaban version en Produccion (estable)

 Numeros Y impares indicaban version en Desarrollo

**6. ¿Cuáles son las razones por las cuáles los usuarios de GNU/Linux recompilan sus kernels?**

* + Soportar nuevos dispositivos como, por ejemplo, una placa de video.
  + Agregar mayor funcionalidad (soporte para algun hardware especco).
  + 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)

**7. ¿Cuáles son las distintas opciones para realizar la configuración de opciones de compilación de un kernel? Cite diferencias, necesidades (paquetes adicionales de software que se pueden requerir), pro y contras de cada una de ellas.**

Existen 3 alternativas:

* Generar el archivo de configuración de forma manual , detallando un valor para cada una de las opciones del kernel (esta es la forma más complicada ya que se necesita conocer cada una de las opciones y los valores que pueden tomar.).
* Utilizar una interfaz basada en pregunta-respuesta , donde una aplicación pregunta secuencialmente que valor le queremos dar a cada una de las opciones del kernel y a partir de esto se genera un archivo de configuración (esta forma es un poco mas practica que la primera , pero sigue siendo engorrosa ya que si se confunde en una pregunta va a ser necesario terminar el cuestionario y volver a arrancar de 0).
* La tercera forma es utilizando una interfaz de menues (ya sea basado en texto o ventanas), que nos permite de una forma mas amigable y practica la configuración del kernel (este método es el mas “amigable” para el usuario pero requiere paquetes adicionales para poder generar la interfaz , y en algún caso un servidor de ventanas para poder desplegarlo).

El kernel Linux se configura mediante el archivo .config. Este archivo, que reside en la raíz del directorio del kernel, contiene las instrucciones de que es lo que el kernel debe compilar.

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.

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.

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.

**8. Nombre al menos 5 opciones de las más importantes que encontrará al momento de realizar la configuración de un kernel para su posterior compilación.**

Entre las opciones mas importantes que se pueden encontrar son la de selección de la arquitectura , el menu que respecta la configuración de lo referente a la red , la selección de los filesystms para los que se dara soporte , los drivers de los dispositivos . Si nos referimos a opciones , seria la selección de la sub-arquitectura , la selección de la familia del procesador , las politicas de apropiación del kernel y los algoritmos de planificacion de cpu a utilizar.

**9. Indique que tarea realiza cada uno de los siguientes comandos durante la tarea de configuración/compilación del kernel:**

**(a) make menuconfig**

**(b) make clean**

**(c) make (investigue la funcionalidad del parámetro -j)**

**(d) make modules (utilizado en antiguos kernels, actualmente no es necesario)**

**(e) make modules\_install**

**(f) make install**

* make menuconfig: Con este comando se utiliza una interfaz de grafica para poder configuara los paquetes que se quieren utlizar para el kernel que se esta por compilar.

*Este modo utiliza ncurses, una librerÍa que permite generar una interfaz con paneles desde la terminal.*

* make clean: Se usa para “limpiar” el kernel de compilaciones fallidas anteriores.

Se usa antes de recompilar asegurando de que no quede ningún archivo sobrante de un compilado anterior, se puede hacer después de un make install para liberar espacio. Remueve todos los archivos intermedios de la compilación.

* make: Se utiliza para realizar la compilacion del codigo fuente. Con el parametro -j podemos indicar con cuantos nucleos realizaremos la compilacion.

*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.*

El parámetro -j seguido de un número se utiliza para indicar la cantidad de threads concurrentes queremos establecer para hacer este proceso.

* make modules: Se utiliza para realizar la compilacion de los modulos seleccionados.

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

Generalmente esta tarea se encuentra incluida en la compilación del kernel con el comando make.

* make modules\_install: Se realiza la instalacion de los modulos seleccionados anteriormente compilados. Ubica los modulos en /lib/modules/<version\_del\_kernel>

*Es una regla del Makefile que ubica los módulos del kernel recién compilado en el directorio correspondiente.*

* make\_install: Toma todos los binarios de los pasos anteriores y los copia en los directorios correspondientes para que puedan ser accedidos, respetando las opciones de makefile, ejecutando la sección INSTALL de este.

Se utiliza para instalar el kernel y otros archivos en el directorio /boot.

**10. Una vez que el kernel fue compilado, ¿dónde queda ubicada su imagen? ¿Dónde debería ser reubicada? ¿Existe algún comando que realice esta copia en forma automática?**

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.

Por convención, es recomendable almacenar en el directorio /boot la imagen compilada del kernel junto con su .config. El comando que puede ser utilizado es update-grub2

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 debe descomprimir en un directorio donde tengamos permisos, como el $HOME del usuario actual. Generalmente se crea un enlace simbólico llamado linux apuntando al directorio del código fuente que actualmente se está configurando.

$ ln -s /usr/src/linux-4.15 /usr/src/linux

**11. ¿A qué hace referencia el archivo initramfs? ¿Cuál es su funcionalidad? ¿Bajo qué condiciones puede no ser necesario?**

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

Es un archivo cpio comprimido normalmente en formato gzip que contiene un pequeño sistema de archivos que se cargará en la memoria RAM en el proceso de arranque del núcleo. El kernel lo montará, como una pequeña raíz, pues la necesita para completar algunas tareas relacionadas con módulos y controladores de dispositivos antes de poder arrancar el verdadero sistema de archivos raíz instalado en el disco duro e invocar al proceso init. Más antiguo que initramfs es initrd “disco RAM inicial”, aún ampliamente usado, y cuya función es la misma a pesar de ciertas diferencias técnicas y de funcionamiento.

Recordemos que al compilar el kernel (make menuconfig) decidimos qué controladores se integrarán dentro de kernel mismo (en el bzImage) y cuales irán fuera en forma de módulos, que puedan ser invocados desde éste si son necesarios. Pues estos módulos se integrarán precisamente en la imagen initramfs (o initrd) para que estén a disposición del kernel en el proceso de arranque del núcleo.

La creación tradicional del ramdisk initrd se hacía mediante el comando mkinitrd pero hoy en día generalmente ha sido sustituido por la creación de un initramfs mediante el comando mkinitramfs.

For most distributions, kernel modules are the biggest reason to have an initramfs. In a general distribution, there are many unknowns such as file system types and disk layouts. In a way, this is the opposite of LFS where the system capabilities and layout are known and a custom kernel is normally built. In this situation, an initramfs is rarely needed.

**12. ¿Cuál es la razón por la que una vez compilado el nuevo kernel, es necesario reconfigurar el gestor de arranque que tengamos instalado?**

# *Para que reconozca el nuevo kernel y poder arrancarlo.*

Es necesario configurar el gestor de arranque nuevamente para que la imagen que acabamos de compilar del Kernel sea apuntada por el gestor de arranque para asi poder iniciar con el nuevo Kernel.

Luego de instalar el kernel, para que el gestor de arranque lo reconozca simplemente deberemos ejecutar, como usuario privilegiado, el siguiente comando: update-grub2.

**13. ¿Qué es un módulo del kernel? ¿Cuáles son los comandos principales para el manejo de módulos del kernel?**

Un modulo del kernel es una sección de código escrito principalmente en C , el cual permite extender la funcionalidad del kernel , pero que tiene como característica que puede ser cargado o descargado a voluntad durante la ejecución sin la necesidad de para el sistema.

Un módulo del kernel es 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 que módulos están cargados.

Algunos de los comandos básicos son:

* lsmod: Lista los módulos cargados en el sistema
* rmmod: Descarga módulos del sistema si es que no están siendo usados. Algunos parámetros pueden ser “-r” que intenta descargar recursivamente módulos de los cuales un modulo dependa. Otra opción puede ser “-a” que descarga todos los módulos que no están siendo usados.
* Insmod: Trata de cargar modulos al kernel. Entre las opciones esta “-f” que permite evitar el chequeo de version (kernel y modulo deben tener igual version)
* depmod: Calcula las dependencias de uno o varios modulos y las alamacena para que la momento de carga , estas sean cargadas. Con “-a” calculas las dependencias entre todos los modulos.
* Modprobe: Permite utilizar la info generada por el anterior para poder cargar un modulo , antes habiendo cargado todos sus dependientes.

**14. ¿Qué es un parche del kernel? ¿Cuáles son las razones principales por las cuáles se deberían aplicar parches en el kernel? ¿A través de qué comando se realiza la aplicación de parches en el kernel?**

Es un mecanismo que permite aplicar actualizaciones NO incrementales sobre la version base. Se basa en archivos diff (archivos de diferencia), que indican que agregar y que quitar. Se aplican sobre la versión base. 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 version.

*El comando con el que se realiza la aplicación de parches es “patch”  
$ cd linux; zcat ../patch-4.15.15.gz | patch -p1*

Es importante mencionar que patch no puede recibir el archivo comprimido (.gz) por lo cual se utiliza previamente zcat. El cual funciona de manera similar al cat con la distinción de que descomprime lo que recibe por su entrada estándar enviando el resultado por su salida estándar.