Practica 1

A - Conceptos teóricos

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

   El kernel de GNU/Linux es un programa que ejecuta programas y gestiona dispositivos de hardware. Es el encargado de que el software y el hardware puedan trabajar juntos.
   Sus principales funciones son:
   - Administración de memoria principal
   - Administración de uso de la CPU

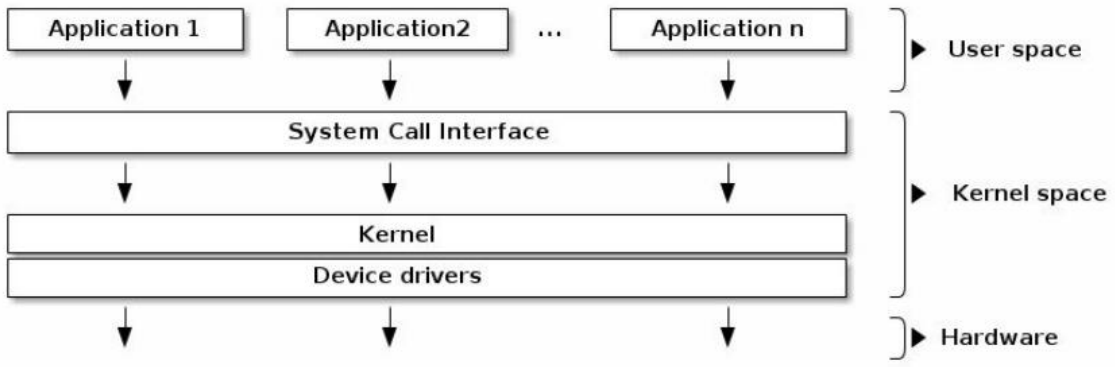
2. Explique brevemente la arquitectura del kernel Linux teniendo en cuenta: tipo de kernel, módulos, portabilidad, etc.

   El kernel de Linux es un kernel monolítico hibrido:
   - 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

   1- Tipo de kernel:

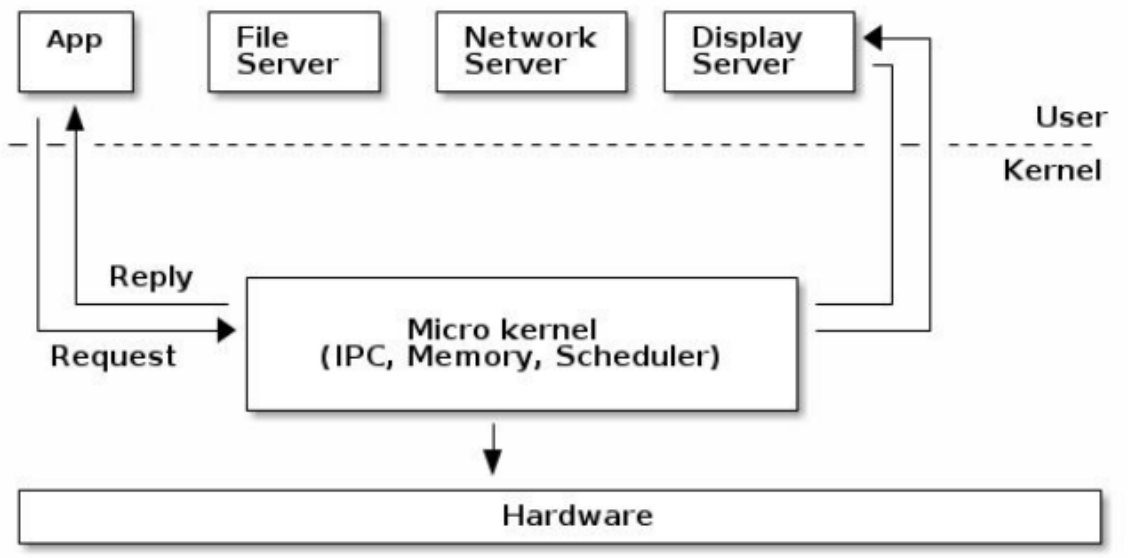
   Kernel monolítico: todas las funciones del sistema operativo se ejecutan en el mismo espacio de memoria y comparten acceso directo a los recursos del hardware. No existe protección de acceso entre los diversos subsistemas del kernel y donde las funciones públicas pueden ser llamadas directamente entre los diferentes subsistemas.

   Como se mencionó antes el kernel de Linux es monolitico.



Micro kernel: las grandes partes del kernel están protegidas entre sí, generalmente ejecutándose como servicios en el espacio de usuario. Debido a que partes significativas del kernel ahora se ejecutan en modo de usuario, el código restante que se ejecuta en modo de kernel es considerablemente más pequeño, de ahí el término “microkernel”. En una arquitectura de microkernel, el kernel contiene justo el código necesario para permitir la comunicación mediante paso de mensajes entre diferentes procesos en ejecución. En la práctica, esto significa implementar el planificador y un mecanismo de comunicación interprocesos (IPC) en el kernel, así como una gestión básica de memoria para establecer la protección entre aplicaciones y servicios. En resumen, solo se ejecuta en modo kernel lo estricto, el resto en modo usuario

No hay muchas implementaciones comerciales de este tipo de kernel.



2- Modulos:

Los módulos son fragmentos de código que se pueden cargar y descargar dinámicamente en el kernel en tiempo de ejecución. Permiten extender o modificar el kernel sin necesidad de reiniciar el sistema completo. Los controladores de dispositivos, sistemas de archivos y otras funcionalidades se implementan como módulos en Linux.

3- Portabilidad:

El kernel de Linux está diseñado para ser altamente portable. Se ha adaptado a una amplia variedad de arquitecturas de hardware, desde computadoras personales hasta servidores, dispositivos móviles, sistemas embebidos y supercomputadoras. La portabilidad se logra mediante una capa de abstracción

de hardware (HAL) y una interfaz de programación de aplicaciones (API) bien definida.

4- Componentes Principales:

- Espacio del núcleo (Kernel Space): Aquí residen las partes críticas del kernel, como la gestión de memoria, la planificación de procesos y los controladores de dispositivos.
- Espacio del usuario (User Space): Contiene las aplicaciones y servicios del sistema.
- Planificador de Procesos: Decide qué proceso se ejecuta en qué momento.
- Gestión de Memoria: Administra la asignación y liberación de memoria.
- Sistema de Archivos Virtual (VFS): Proporciona una interfaz uniforme para acceder a diferentes sistemas de archivos.
- Controladores de Dispositivos: Comunican el kernel con el hardware.

3. ¿Cómo se define el versionado de los kernels Linux en la actualidad?

- Versión < 2.6: X.Y.Z
  - X: serie principal. Funcionalidades importantes.
  - Y: indicaba si era de producción o desarrollo.
    - números Y pares indicaban producción (estable) y los Y impares indicaban una versión en desarrollo.
  - Z: bugfixes.
- Versión ≥ 2.6 y < 3.0: A.B.C.[D]
  - A: Denota versión. Cambia con menor Frecuencia (cada varios años).
  - B: Denotan revisión mayor, cambios importantes.
  - C: Denota revisión menor. Cambia cuando hay nuevos drivers o características.
  - D: Se corrige un error grave sin agregar funcionalidad.
- Versión ≥ 3.0: 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

4. ¿Cuáles son los motivos por los que un usuario/a GNU/Linux puede querer re-compilar el kernel?

- 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)

5. ¿Cuáles son las distintas opciones y menús 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.

   Las alternativas para configurar el nuevo kernel corresponden a los comandos:

   - make config,
   - make menuconfig
   - make xconfig

   El comando make config ofrece un interfaz primitivo basado en modo texto, sin ayuda ni menús. Los comandos make menuconfig y make xconfig conducen a mejores interfaces de usuario, con menús y quizá lo más importante, con una ayuda contextual que facilita en gran medida la configuración.

   Con make menuconfig se requiere un terminal y un sistema de compilación básico (como el conjunto de herramientas GNU, incluyendo gcc y make) y utiliza ncurses, una librería que permite generar una interfaz con paneles desde la terminal. Es robusto y se puede ejecutar en entornos con recursos limitados. Aun así, puede no ser tan intuitivo para usuarios nuevos. La navegación puede ser un poco lenta debido a la interfaz basada en texto.

   Con make xconfig se requiere un sistema con soporte para X Window System, así como las bibliotecas y herramientas necesarias para desarrolladores gráficos. No todos los sistemas tienen instalado X. Es más fácil de usar para usuarios que prefieren interfaces gráficas. Permite una visualización más clara de las opciones del kernel. Puede ser menos eficiente en términos de recursos que las herramientas basadas en texto. Requiere un entorno gráfico instalado y configurado.

   Con make config se necesita un terminal y un sistema de compilación básico, similar a Menuconfig. Este método no requiere una interfaz gráfica ni bibliotecas adicionales. No requiere una interfaz gráfica, lo que lo hace adecuado para sistemas sin soporte para entornos gráficos. Puede ser bastante tedioso y llevar mucho tiempo, especialmente en sistemas con muchas opciones de configuración. No ofrece la misma experiencia visual o la facilidad de navegación que otras herramientas como Menuconfig o Xconfig. Dado que presenta las preguntas de configuración en orden secuencial, puede ser menos eficiente que otras herramientas que permiten una navegación más rápida a través de las opciones.

6. Indique qué tarea realiza cada uno de los siguientes comandos durante la tarea de configuración/compilación del kernel:

    a. make menuconfig
    Inicia la interfaz de configuración menuconfig. Se puede habilitar o deshabilitar características del kernel, ajustar configuraciones de hardware, etc.

    b. make clean
    Limpia el árbol de código fuente del kernel, eliminando los archivos generados durante el proceso de compilación anterior. Se utiliza para asegurar que la compilación se realice desde un estado limpio, sin archivos residuales de compilaciones anteriores.

    c. make (investigue la funcionalidad del parámetro -j)
    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.
    El parámetro -j se utiliza para especificar el número de trabajos paralelos que pueden ejecutarse durante la compilación. Por ejemplo, make -j4 permitiría que hasta 4 tareas se ejecuten simultáneamente durante el proceso de compilación. Esto puede acelerar significativamente el tiempo de compilación en sistemas con múltiples núcleos de CPU.

    d. make modules (utilizado en antiguos kernels, actualmente no es necesario)
    Este comando compila únicamente los módulos del kernel, es decir, las partes del kernel que pueden cargarse y descargarse dinámicamente en tiempo de ejecución.
    En versiones más antiguas del kernel, era necesario compilar los módulos por separado. Sin embargo, en versiones modernas del kernel, este paso generalmente se incluye en el proceso de compilación principal y no se necesita ejecutar por separado.

    e. make modules_install
    Después de compilar los módulos del kernel, este comando instala los módulos compilados en el directorio /lib/modules/<kernel-version>/. Los módulos instalados aca pueden ser cargados por el sistema operativo cuando sea necesario.

    f. make install
    Este comando instala el kernel compilado en el sistema. Copia el kernel y los archivos necesarios al directorio de arranque del sistema (generalmente

/boot) y actualiza la configuración del cargador de arranque para que pueda arrancar el nuevo kernel.
Una vez completado, el nuevo kernel está listo para ser seleccionado y arrancado durante el proceso de inicio del sistema.

7. 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/. Debería ser reubicada en el directorio /boot. Si, existe un comando que realiza esta copia de forma automática: sudo make install

8. ¿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. Proporcionar un entorno mínimo y temporal que permite al sistema operativo cargar los controladores necesarios, montar el sistema de archivos raíz real y completar el proceso de inicio. Contiene ejecutables, drivers y módulos necesarios para lograr iniciar el sistema. Luego del proceso de arranque el disco se desmonta

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

   Es necesario reconfigurar el gestor de arranque para actualizar su configuración con la información correcta sobre la nueva imagen del kernel y sus parámetros de arranque.
   El gestor de arranque no tiene conocimiento automático cuando se tiene un nuevo kernel y, por lo tanto, no lo incluirá en la lista de opciones de arranque del sistema. Por lo tanto, es necesario reconfigurar el gestor de arranque para que se incluya el nuevo kernel.

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

    Es un fragmento de código que puede cargarse/descargarse en el mapa de memoria del SO (Kernel) bajo demanda. Estos módulos permiten extender la funcionalidad del Kernel sin la necesidad de reiniciar el sistema. Los módulos disponibles se ubican en /lib/modules/version del kernel.
    Los comandos principales para el manejo de módulos del kernel son:

1. modprobe: Este comando se utiliza para cargar un módulo del kernel en la memoria en tiempo de ejecución.
2. insmod: Este comando se utiliza para insertar (cargar) un módulo del kernel en la memoria. A diferencia de modprobe, insmod no maneja automáticamente las dependencias del módulo
3. rmmod: Este comando se utiliza para eliminar (descargar) un módulo del kernel de la memoria.
4. lsmod: Este comando muestra una lista de los módulos del kernel que están actualmente cargados en la memoria del sistema.
5. depmod: Este comando se utiliza para generar y manipular los archivos de dependencia de los módulos del kernel.

11. ¿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?

    Patch es un mecanismo que permite aplicar actualizaciones sobre una versión base. Se basa en archivos diff, que indican que 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.

    El comando principal utilizado para aplicar parches en el kernel es patch. Este comando toma un archivo de parche como entrada y aplica los cambios especificados en el código fuente del kernel. La sintaxis básica del comando es la siguiente:

    patch -p<num> < <archivo_parche>

    Donde <num> es el nivel de prefijo de ruta a ser eliminado del nombre del archivo y <archivo_parche> es el archivo que contiene los cambios del parche. Es importante ejecutar el comando patch en el directorio raíz del código fuente del kernel para que pueda aplicar los cambios correctamente en la estructura de directorios correcta.

12. Investigue la característica Energy-aware Scheduling incorporada en el kernel 5.0 y explique brevemente con sus palabras:

a. ¿Qué característica principal tiene un procesador ARM big.LITTLE?

   Un procesador ARM big.LITTLE tiene dos tipos de núcleo: núcleos potentes de alto rendimiento y núcleos menos potentes pero con mucho menor consumo. Esto permite que el sistema balancee la carga de trabajo entre los núcleos de acuerdo con las demandas de rendimiento y energía actuales. Los núcleos big se utilizan para tareas intensivas en recursos y los LITTLE para tareas menos exigentes.

b. En un procesador ARM big.LITTLE y con esta característica habilitada. Cuando se despierta un proceso ¿a qué procesador lo asigna el scheduler?

   Con la característica Energy-aware Scheduling habilitada en un procesador ARM big.LITTLE, cuando se despierta un proceso, el scheduler lo asigna al núcleo que mejor se adapte a las necesidades de rendimiento y consumo de energía en ese momento. Si el proceso requiere un alto rendimiento, podría asignarse a un núcleo big, mientras que si requiere menos recursos, podría asignarse a un núcleo LITTLE para conservar energía.

c. ¿A qué tipo de dispositivos opinás que beneficia más esta característica?

   La característica Energy-aware Scheduling beneficia especialmente a dispositivos móviles para equilibrar el rendimiento y la duración de la batería.

13. Investigue la system call memfd_secret() incorporada en el kernel 5.14 y explique brevemente con sus palabras

    a. ¿Cuál es su propósito?

       Permite crear áreas de memoria secretas que no pueden ser accedidas ni por el usuario root.

    b. ¿Para qué puede ser utilizada?

       Esa memoria se puede utilizar para almacenar claves criptográficas u otros datos que no deben ser expuestos a otros.

    c. ¿El kernel puede acceder al contenido de regiones de memoria creadas con esta system call?

       No, no puede.

B - Ejercicio taller: Compilación del kernel Linux

1. Descargue los siguientes archivos en un sistema GNU/Linux moderno, sugerimos descargarlo en el directorio $HOME/kernel/ (donde $HOME es el directorio del usuario no privilegiado que uses):

    a. El archivo btrfs.image.xz publicado en la página web de la cátedra.
    b. El código fuente del kernel 6.7 (https://mirrors.edge.kernel.org/pub/linux/kernel/v6.x/linux-6.7.tar.xz).
    c. El parche para actualizar ese código fuente a la versión 6.8 (https://cdn.kernel.org/pub/linux/kernel/v6.x/patch-6.8.xz).

    Este ejercicio lo pude hacer con una maquina virtual con Debian 12.5.0 (no con la de la catedra)

    Herramientas que tuve que instalar:

    - sudo apt-get install flex
    - sudo apt-get install bison
    - sudo apt-get install libncurses5-dev
    - sudo apt-get install libelf-dev
    - sudo apt-get install libelf-dev
    - sudo apt-get install libssl-dev

2. Preparación del código fuente:

    a. Posicionarse en el directorio donde está el código fuente y descomprimirlo:

    $ cd $HOME/kernel/
    $ tar xvf linux-6.7.tar.xz

    b. Emparchar el código para actualizarlo a la versión 6.8 usando la herramienta patch:

    $ cd $HOME/kernel/linux-6.7
    $ xzcat $HOME/kernel/patch-6.8.xz | patch -p1

3. Pre-configuración del kernel:

    a. Usaremos como base la configuración del kernel actual, esta configuración por convención se encuentra en el directorio /boot. Copiaremos y renombraremos la configuración actual al directorio del código fuente con el comando:

    $ cp /boot/config-$(uname -r) $HOME/kernel/linux-6.7/.config

b. Generaremos una configuración adecuada para esta versión del kernel con olddefconfig. olddefconfig toma la configuración antigua que acabamos de copiar y la actualiza con valores por defecto para las opciones de configuración nuevas

$ cd $HOME/kernel/linux-6.7/
$ make olddefconfig

c. A fin de construir un kernel a medida para la máquina virtual usaremos a continuación localmodconfig que configura como módulos los módulos del kernel que se encuentran cargados en este momentos deshabilitando los módulos no utilizados. Es probable que make pregunte por determinadas opciones de configuración, si eso sucede presionaremos la tecla Enter en cada opción para que quede el valor por defecto hasta que make finalize.

$ make localmodconfig

4. Configuración personalizada del kernel. Utilizaremos la herramienta menuconfig para configurar otras opciones. Para ello ejecutaremos:

$ make menuconfig

a. Habilitar las siguientes opciones para poder acceder a btrfs.tar.xz:
   i. File Systems -> Btrfs filesystem support.
   ii. Device Drivers -> Block Devices -> Loopback device support.
b. Deshabilitar las siguientes opciones para reducir el tamaño del kernel y los recursos necesarios para compilarlo:
   i. General setup -> Configure standard kernel features (expert users).
   ii. Kernel hacking -> Kernel debugging.

5. Luego de configurar nuestro kernel, realizaremos la compilación del mismo y sus módulos.
   a. Para realizar la compilación deberemos ejecutar:

$ make -j4

6. Finalizado este proceso, debemos reubicar las nuevas imágenes en los directorios correspondientes, instalar los módulos, crear una imagen initramfs y reconfigurar nuestro gestor de arranque. En general todo esto se puede hacer de forma automatizada con los siguientes comandos.

7. Como último paso, a través del comando reboot, reiniciaremos nuestro equipo y probaremos el nuevo kernel recientemente compilado.

a. En el gestor de arranque veremos una nueva entrada que hace referencia al nuevo kernel. Para bootear, seleccionamos esta entrada y verificamos que el sistema funcione correctamente.
b. En caso de que el sistema no arranque con el nuevo kernel, podemos reiniciar el equipo y bootear con nuestro kernel anterior para corregir los errores y realizar una nueva compilación.
c. Para verificar qué kernel se está ejecutando en este momento puede usar el comando:

$ uname -r

```
agusnfr@SistemasOperativos:~$ uname -r
6.8.0
```

C - Ejercicio obligatorio

1. Descomprimir el filesystem con:

$ unxz btrfs.image.xz

```
agusnfr@SistemasOperativos:~$ cd kernel/
agusnfr@SistemasOperativos:~/kernel$ unxz btrfs.image.xz
```

2. Verificaremos que dentro del directorio /mnt exista al menos un directorio donde podamos montar nuestro pseudo dispositivo. Si no existe el directorio, crearlo. Por ejemplo podemos crear el directorio /mnt/btrfs/.

```
agusnfr@SistemasOperativos:~/kernel$ cd /mnt
agusnfr@SistemasOperativos:/mnt$ ls
agusnfr@SistemasOperativos:/mnt$ mkdir btrfs
mkdir: cannot create directory ‘btrfs’: Permission denied
agusnfr@SistemasOperativos:/mnt$ su
Password:
root@SistemasOperativos:/mnt# mkdir btrfs
```

3. A continuación montaremos nuestro dispositivo utilizando los siguientes comandos:

$ su -
 # mount -t btrfs -o loop $HOME/btrfs.image /mnt/btrfs/

```
root@SistemasOperativos:/mnt# mount -t btrfs -o loop /home/agusnfr/kernel
/btrfs.image /mnt/btrfs/
root@SistemasOperativos:/mnt#
```

4. Diríjase a /mnt/btrfs y ejecute el script entregap1.sh completando los datos solicitados

```
root@SistemasOperativos:/mnt/btrfs# ls
entregap1.sh
root@SistemasOperativos:/mnt/btrfs# ./entregap1.sh
Verifying archive integrity...  100%   MD5 checksums are OK. All good.
Uncompressing entregap1  100%
Ingrese su nombre y apellido: Agustina Sol Rojas
Ingrese su número de alumno:
Entrega enviada a la cátedra con éxito
```

5. Si no tiene acceso a Internet el script le dará un texto que deberá enviar a la cátedra para considerar la entrega.