A - Introducción

1. ¿Qué es GCC?

Es el compilador estándar en muchas distribuciones de GNU/Linux y es ampliamente utilizado para la compilación de software de código abierto.

2. ¿Qué es make y para que se usa?

make es una herramienta de automatización que se usa para compilar y construir programas de manera eficiente. Permite definir reglas en un archivo llamado Makefile, que describe cómo generar ejecutables a partir de código fuente. make verifica qué archivos han cambiado y recompila solo lo necesario, optimizando el proceso de compilación.

3. La carpeta /home/so/practica1/ejemplos/01-make de la VM contiene ejemplos de uso de make. Analice los ejemplos, en cada caso ejecute `make` y luego `make run` (es opcional ejecutar el ejemplo 4, el mismo requiere otras dependencias que no vienen preinstaladas en la VM):

el 01-helloworld imprime “Hello world!”
el 02-multiplefiles imprime “Value 9… Value0”
el 03-htmlotxt elimino los tags de los archivos html
el 04- lero lero

a. Vuelva a ejecutar el comando `make`. ¿Se volvieron a compilar los programas? ¿Por qué?

No porque no cambiamos nada :)

b. Cambie la fecha de modificación de un archivo con el comando `touch` o editando el archivo y ejecute `make`. ¿Se volvieron a compilar los programas? ¿Por qué?

Si, vuelve a compilar

c. ¿Por qué “run” es un target “phony”?

Un target "phony" en un Makefile es un objetivo que no corresponde a un archivo real. Se usa para ejecutar comandos sin que make intente buscar o comparar archivos con ese nombre.

El target run no crea un archivo, solo ejecuta un comando (como ejecutar un programa). Si no se marca como .PHONY, make podría pensar que run es un archivo y no ejecutarlo si ya existe un archivo con ese nombre.

d. En el ejemplo 2 la regla para el target `dlinkedlist.o` no define cómo generar el target, sin embargo el programa se compila correctamente. ¿Por qué es esto?

Porque make tiene reglas implícitas para compilar archivos .c en .o mediante el compilador de C (como gcc).

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

El kernel de GNU/Linux es el núcleo del sistema operativo, encargado de gestionar el hardware, la memoria, los procesos y la seguridad.

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

Tipo: Kernel monolítico (porque está todo en un solo lugar) híbrido (porque podemos sacar y agregar módulos)
Módulos: Permiten cargar y descargar funciones en tiempo real (e.g., controladores).
Portabilidad: Es compatible con múltiples arquitecturas de hardware gracias a una capa de abstracción de hardware y una interfaz de programación de aplicaciones.

Componentes principales: Kernel Space, User Space, planificador de procesos, gestion de memoria, Sistema de archivos virtuales, controladores de dispositivos

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

7. ¿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)

8. ¿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.

make menuconfig: Interfaz gráfica basada en consola. Fácil de usar, pero requiere acceso a terminal.
make xconfig: Interfaz gráfica GUI (requiere librerías X).interfaz gráfica avanzada, necesita X.
make oldconfig: Usar configuración anterior y agregar nuevas opciones. Rápido, pero requiere conocimientos previos de configuración

make config: ofrece una interfaz primitiva basada en modo texto sin ayudas ni menus

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

a. make menuconfig:
Abre una interfaz de configuración basada en texto (ncurses) para seleccionar y configurar las opciones del kernel.

b. make clean:
Elimina los archivos generados previamente (como los objetos y binarios) para realizar una compilación limpia desde cero.

c. make -j:
Compila el kernel y sus módulos. El parámetro -j permite especificar el número de trabajos paralelos para acelerar la compilación (por ejemplo, -j4 usa 4 núcleos de CPU).

d. make modules:
Compila solo los módulos del kernel. Aunque en versiones recientes no siempre es necesario, aún puede usarse para compilar módulos específicos.

e. make modules_install:
Instala los módulos compilados en el directorio de módulos del sistema (/lib/modules/$(uname -r)).

f. make install:

Instala el kernel compilado en el sistema, copiando el kernel y los archivos de configuración al directorio adecuado y actualizando el gestor de arranque (GRUB).

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

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

Es una imagen comprimida que contiene los archivos y controladores necesarios para arrancar el sistema. Proporciona los controladores y scripts esenciales para montar el sistema de archivos raíz durante el arranque.No es necesario si el sistema de archivos raíz es simple y el kernel puede acceder directamente sin controladores adicionales.

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

Tiene que ser reconfigurado para reconocer la nueva imagen del kernel.
Si no lo haces capaz arranca con el kernel viejo

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

Un módulo del kernel es una porcion de codigo especializada en realizar una acción específica. se pueden cargar y descargar dinámicamente en el kernel en tiempo de ejecución.

- lsmod: Muestra los módulos cargados actualmente en el kernel.
- modprobe: Carga o descarga módulos en el kernel. Por ejemplo, modprobe <módulo> carga un módulo, y modprobe -r <módulo> lo descarga.
- insmod: Inserta un módulo directamente en el kernel. No resuelve dependencias automáticamente.
- rmmod: Elimina un módulo cargado del kernel.
- depmod: Actualiza las dependencias de los módulos, lo que es útil cuando se añaden nuevos módulos o se cambian las configuraciones.

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?

- Corrección de errores: Arreglar vulnerabilidades o fallos en el código.
- Mejoras de seguridad: Resolver brechas de seguridad y proteger el sistema.
- Optimización de rendimiento: Mejorar la eficiencia del kernel o la compatibilidad con hardware.
- Compatibilidad con nuevos dispositivos: Añadir soporte para nuevos componentes de hardware.

El comando que se usa es patch.

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

a. Característica principal de un procesador ARM big.LITTLE: Un procesador ARM big.LITTLE combina dos tipos de núcleos:
Núcleos "big": Potentes y de alto rendimiento.
Núcleos "LITTLE": Más pequeños y eficientes en términos de energía.
La idea es que los núcleos LITTLE manejen tareas ligeras o de baja demanda energética, mientras que los núcleos big se usan para tareas de alto rendimiento.

b. ¿Qué procesador asigna el scheduler cuando se despierta un proceso? Con Energy-aware Scheduling (EAS) habilitado, el scheduler asigna un proceso al núcleo más adecuado dependiendo de la carga del sistema y las necesidades de energía. Si el proceso es ligero o inactivo, probablemente será asignado a un núcleo LITTLE para ahorrar energía. Si el proceso requiere más potencia, se asignará a un núcleo big.

c. ¿A qué tipo de dispositivos beneficia más esta característica? La característica Energy-aware Scheduling beneficia principalmente a dispositivos móviles, como smartphones y tabletas, donde la eficiencia energética es crucial. También beneficia a dispositivos embebidos y portátiles que necesitan equilibrar el rendimiento y el consumo de energía para maximizar la duración de la batería.

16.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 criptograficas 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?
Si, a vecesi

B - Ejercicio taller: Compilación del kernel Linux