Sistemas Operativos - Práctica 1

A - Introducción

1. ¿Qué es GCC?
   GCC es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran; es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde ha de correr. La sigla GCC significa "GNU Compiler Collection".

```
gcc hola.c
# Compila el programa en C hola.c, genera un archivo ejecutable a.out
```

2. ¿Qué es make y para qué se usa?
   `Make`es un programa de automatización de compilación. Gestiona la construcción de programas a partir de su código fuente, definiendo las reglas y dependencias necesarias para compilar cada archivo. Si un archivo fuente se modifica, `make`recompila solo los archivos afectados. Lee una descripción de un proyecto desde un archivo conocido por `makefile`donde se automatiza el proceso sin tener que escribir manualmente cada comando de compilación.

```
programa: main.o funciones.o
    gcc -o programa main.o funciones.o

main.o: main.c funciones.h
    gcc -c main.c

funciones.o: funciones.c funciones.h
    gcc -c funciones.c

clean:
    rm -f *.o programa
```

y luego para compilar el programa, que ejecutará la primera regla y sus dependencias:

```
make
```

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):
   - a. Vuelva a ejecutar el comando `make`. ¿Se volvieron a compilar los programas? ¿Por qué?
   - 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?
   - c. ¿Por qué “run” es un target “phony”?

    - 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?

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, es decir, la capa de software que se encarga de la comunicación entre el hardware y las aplicaciones. Funciones principales del kernel:

- **Gestión de procesos**: manejo de prioridad de procesos y planificación. Asignación de tiempo de CPU a los procesos.
- **Gestión de memoria**: asignación y liberación de memoria para los procesos en ejecución. Administra la memoria RAM y la virtual.
- **Gestión del sistema de archivos**
- **Gestión de dispositivos**: comunicación con el hardware mediante drivers.
- **Gestión de seguridad y permisos**: proporciona aislamiento entre procesos para evitar accesos no autorizados.
- **Gestión de redes**

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

- **Tipo de kernel**: es un **kernel monolítico modular**, lo que significa que todo el sistema central (gestión de procesos, memoria, dispositivos, etc.) opera en el mismo espacio de memoria (monolítico), pero con soporte para cargar y descargar módulos de manera dinámica (modularidad).
- **Módulos del kernel**: permite cargar y descargar módulos *(kernel modules o LKM - Loadable Kernel Modules)*, lo que evita la necesidad de recompilar el kernel completo cuando se requiere soporte para nuevos dispositivos o funciones.
- **Portabilidad**: es altamente portable y funciona en diferentes arquitecturas de hardware, desde computadoras personales (x86, x86_64) hasta dispositivos móviles y embebidos. Se divide en varias capas:
- Espacio de usuario: contiene aplicaciones y herramientas. Utiliza syscalls para comunicarse con el kernel.
- Llamadas al sistema (System Calls): funciones del kernel que permiten a los programas solicitar servicios como abrir archivos, asignar memoria, etc.
- Capa del Kernel: gestion de procesos, gestión de memoria, etc.
- Capa de abstracción de hardware: el kernel interactúa con diferentes arquitecturas de hardware de forma genérica.
- Hardware: CPU, RAM, etc.

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

```
<Major>.<Minor>.<Patch>
```

Donde:

- Major (Mayor): Cambia en eventos excepcionales (muy raro). Indica cambios profundos en la arquitectura del kernel.
- Minor (Menor): Se incrementa con cada nueva versión estable del kernel.

- Patch (Corrección): Se usa para actualizaciones de mantenimiento o seguridad dentro de una versión estable.

7. ¿Cuáles son los motivos por los que un usuario/a GNU/Linux puede querer recompilar el kernel?
   Para optimizar el rendimiento (eliminar módulos innecesarios), agregar soporte para hardware nuevo, habilitar o deshabilitar características del kernel, agregar parches de seguridad, etc.

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), pros y contras de cada una de ellas.

9. Indique qué tarea realiza cada uno de los siguientes comandos durante la 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`

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?

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

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

- El nuevo kernel debe ser detectado: El gestor de arranque carga el sistema operativo, pero solo reconoce los kernels configurados en su archivo de configuración. Si no se actualiza, el sistema seguirá arrancando con el kernel antiguo.
- Ubicación de la imagen del kernel: cuando se instala el nuevo kernel, su imagen `(vmlinuz-*)` se guarda en `/boot/`. Se debe actualizar GRUB para que apunte a la nueva versión.
- Actualización del `initramfs`: `initramfs` es una imagen de disco temporal necesaria para que el kernel acceda a los dispositivos antes de montar el sistema de archivos principal. Debe regenerarse para que el nuevo kernel tenga los módulos correctos.

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

14. ¿Qué es un parche del kernel?

    - ¿Cuáles son las razones principales por las que se deberían aplicar parches en el kernel?
    - ¿A través de qué comando se realiza la aplicación de parches en el kernel?

15. Investigue la característica *Energy-aware Scheduling* incorporada en el kernel 5.0 y explique brevemente:
    - a. ¿Qué característica principal tiene un procesador ARM *big.LITTLE*?
    - b. En un procesador ARM *big.LITTLE* y con esta característica habilitada, cuando se despierta un proceso, ¿a qué procesador lo asigna el *scheduler*?
    - c. ¿A qué tipo de dispositivos cree que beneficia más esta característica?
16. Investigue la *system call* `memfd_secret()` incorporada en el kernel 5.14 y explique brevemente:
    - a. ¿Cuál es su propósito?
    - b. ¿Para qué puede ser utilizada?
    - c. ¿El kernel puede acceder al contenido de regiones de memoria creadas con esta *system call*?