Trabajo Práctico N° 1 - Conceptos de Sistemas Operativos

# 1. Características de GNU/Linux

a. Características más relevantes de GNU/Linux:

- Software Libre: GNU/Linux se distribuye bajo licencias libres, principalmente la GPL (Licencia Pública General de GNU), lo que permite el uso, estudio, modificación y redistribución del sistema.  
- Multitarea: permite ejecutar múltiples procesos de manera simultánea.  
- Multiusuario: varios usuarios pueden trabajar en el sistema al mismo tiempo, sin interferir entre sí.  
- Estabilidad y seguridad: es reconocido por su robustez y menor exposición a malware en comparación con otros sistemas operativos.  
- Portabilidad: GNU/Linux puede instalarse en múltiples arquitecturas (x86, ARM, PowerPC, etc.).  
- Gran variedad de distribuciones: existen cientos de distribuciones adaptadas a diferentes necesidades.  
- Comunidad activa: el soporte y desarrollo está respaldado por una amplia comunidad.

b. Comparación con otros sistemas operativos:

- Windows: es propietario, orientado a la facilidad de uso, pero menos personalizable que GNU/Linux. Su estabilidad puede verse afectada con el tiempo y requiere licencias pagas.  
- macOS: también es propietario, muy optimizado para el hardware de Apple. Se basa en UNIX como GNU/Linux, pero limita la personalización.  
- BSD: similar a GNU/Linux en su núcleo UNIX, muy seguro, aunque con menor adopción en escritorio.

c. GNU: Es un proyecto iniciado por Richard Stallman en 1983 cuyo objetivo fue crear un sistema operativo libre, completamente compatible con UNIX.

d. Historia del proyecto GNU: El proyecto GNU comenzó en 1983, impulsado por la Free Software Foundation. Su propósito era ofrecer software libre y reemplazar programas privativos. Desarrolló compiladores, bibliotecas y utilidades. La pieza que faltaba era el kernel, por lo cual se unió con Linux, creado por Linus Torvalds en 1991.

e. Multitarea: Es la capacidad de ejecutar varios procesos de manera simultánea. GNU/Linux implementa multitarea preventiva, asignando CPU a cada proceso según prioridades.

f. POSIX: Es un estándar definido por IEEE para garantizar la compatibilidad entre sistemas operativos de tipo UNIX, asegurando que los programas sean portables.

# 2. Distribuciones de GNU/Linux

a. Una distribución de GNU/Linux es una compilación que incluye el kernel de Linux, herramientas GNU, gestores de paquetes, entornos gráficos y aplicaciones adicionales.  
Ejemplos:  
- Debian: estable, orientada a servidores.  
- Ubuntu: basada en Debian, enfocada en facilidad de uso.  
- Fedora: patrocinada por Red Hat, con software de última generación.  
- Arch Linux: minimalista, orientada a usuarios avanzados.

b. Diferencias: se distinguen por gestor de paquetes, filosofía (estabilidad vs. última versión), entornos gráficos incluidos, soporte comunitario o empresarial.

c. Debian: Proyecto iniciado en 1993 por Ian Murdock. Sus objetivos son ofrecer un sistema operativo universal, libre, estable y portable. Cronología: primera versión en 1996, y desde entonces ha mantenido ciclos de desarrollo organizados con ramas stable, testing y unstable.

# 3. Estructura de GNU/Linux

a. Componentes fundamentales:  
- Kernel: núcleo que gestiona hardware y procesos.  
- Herramientas del sistema (GNU): comandos básicos, compiladores, librerías.  
- Aplicaciones: programas de usuario como navegadores, suites ofimáticas, etc.

b. Estructura básica: GNU/Linux se organiza con un kernel central, capas de librerías y utilidades, entornos gráficos (como GNOME o KDE) y aplicaciones. El sistema de archivos jerárquico comienza en '/'.

# 4. Kernel

a. Funciones principales: administración de memoria, manejo de procesos, comunicación entre hardware y software, gestión de dispositivos, control de seguridad.

b. Versión actual: cambia con frecuencia, se puede consultar con 'uname -r'.  
Esquema de versionado:  
- Antes de 2.4: versiones pares eran estables, impares en desarrollo.  
- Desde 2.6: se adoptó un esquema de numeración continua sin esa distinción.

c. Sí, se puede tener más de un kernel instalado. Al iniciar, el gestor de arranque (GRUB) permite elegir qué kernel usar.

d. El kernel se encuentra en /boot, generalmente con nombres como vmlinuz-\*

# 5. Intérprete de comandos (Shell)

a. Es un programa que interpreta comandos que el usuario ingresa y los envía al sistema.

b. Funciones: ejecutar programas, automatizar tareas con scripts, gestionar entrada/salida.

c. Ejemplos de intérpretes:  
- Bash: el más común, completo y con scripting.  
- Zsh: más personalizable, con autocompletado avanzado.  
- Dash: ligero, enfocado en velocidad.

d. Los comandos internos se encuentran en el propio Shell, los externos en directorios como /bin, /usr/bin.

e. No forma parte del kernel porque es una capa de interacción, no de administración del hardware.

f. Sí, cada usuario puede definir su shell en /etc/passwd, pero requiere permisos de administrador para modificarlo.

# 6. Sistema de Archivos en Linux

a. Es la estructura organizada para almacenar y acceder a archivos en un disco.

b. Estructura básica: basada en FHS (Filesystem Hierarchy Standard).  
- / (raíz)  
- /home: directorios de usuarios  
- /etc: configuraciones  
- /bin y /usr/bin: binarios ejecutables  
- /var: archivos variables  
- /tmp: temporales

c. Sistemas soportados: ext2, ext3, ext4, xfs, btrfs, reiserfs, entre otros.

d. Sí, Linux soporta FAT y NTFS mediante controladores especiales.

# 7. Particiones

a. Una partición es una división lógica de un disco duro. Tipos: primaria, extendida y lógica.  
Ventajas: mejor organización, seguridad, instalación de varios SO.  
Desventajas: espacio fijo y limitado si se particiona mal.

b. Identificación en Linux:  
- IDE: /dev/hd[a-d]  
- SCSI/SATA: /dev/sd[a-z]  
- Particiones numeradas: /dev/sda1, /dev/sda2, etc.

c. Particiones mínimas:  
- / (raíz)  
- swap  
Opcional pero recomendable: /home

d. Ejemplos: servidores requieren /var y /srv; estaciones de trabajo requieren /home grande.

e. Software: fdisk, parted, gparted. Diferencias: fdisk es CLI, parted más moderno, gparted con GUI.

# 8. Arranque del Sistema Operativo

a. BIOS: firmware básico que inicializa el hardware.

b. UEFI: reemplazo moderno de BIOS, más seguro y rápido.

c. MBR: Master Boot Record, sector de arranque en discos antiguos.  
MBC no es un término común, probablemente refiere al código de arranque en MBR.

d. GPT: GUID Partition Table, reemplazo de MBR, soporta discos grandes y más particiones.

e. Gestor de arranque: carga el sistema operativo. Ejemplos: GRUB, LILO, systemd-boot.

f. Pasos de bootstrap: encendido → POST → BIOS/UEFI → búsqueda de cargador → carga del kernel.

g. En Linux: BIOS/UEFI → GRUB → Kernel → init/systemd → login.

h. Proceso de parada: systemctl poweroff → envía señales a procesos → desmonta FS → apaga.

i. Sí, se puede tener dual boot, Linux junto a Windows/macOS.

# 9. Archivos y editores

a. Los archivos en GNU/Linux se identifican por su nombre, que distingue mayúsculas y minúsculas. No dependen de la extensión para definir su tipo.

b. Editores:  
- vim: potente, basado en modos de edición.  
- nano: sencillo, accesible desde consola.  
- mcedit: parte de Midnight Commander, interfaz más amigable.  
Comandos de visualización:  
- cat: muestra contenido completo.  
- more: muestra página por página.  
- less: similar a more, pero más flexible (permite retroceder).

c. Para crear archivo 'prueba.exe' en vim:  
$ vim prueba.exe  
Escribir número de alumno y nombre, guardar con ':wq'.

d. Comando 'file': identifica el tipo real de un archivo independientemente de su extensión.

e. Comandos relacionados con archivos:  
- cd: cambiar directorio.  
- mkdir: crear directorio.  
- rmdir: eliminar directorio vacío.  
- ln: crear enlaces (duros o simbólicos).  
- tail: ver últimas líneas de un archivo.  
- locate: buscar archivos rápidamente.  
- ls: listar archivos.  
- pwd: mostrar directorio actual.  
- cp: copiar archivos.  
- mv: mover o renombrar archivos.  
- find: buscar archivos con criterios avanzados.

**10. Comandos y argumentos**

a.Crear la carpeta ISOCSO -> mkdir “nombreCarpeta”.

b. Accede a la carpeta -> cd “nombreCarpeta” o “path”.

c. Crear dos archivos, para crear uno se usa > “nombreArhivo.extension” o para crear varios touch “archivo1.ext archivoN.ext”.

d. Listar el contenido del directorio actual ls.

e. Visualizar la ruta donde estoy situado pwd (print working directory).

f. Para buscar los archivos que contengan iso\* se usa sudo (para evitar que se detenga en caso de no tener permiso de lectura o acceso) find / (la / es para buscar desde la raíz) –name “\*iso”. (con –iname se ignora el case sensitive).

g. La cantidad de espacio libre en disco se puede ver con el comando df (disk free) –h (para que sea legible para humanos GB/MB).

h. Para verificar los usuarios conectados al sistema se utiliza el comando who.

i. Para editar existen varias alternativas:

1. nano archivo.txt -> abre el archivo y nos permite editarlo para luego cerrar con Ctrl + O o con Ctrl + X.

2. vim archivo.txt -> abre el archivo y puedo utilizar i para insertar al inicio, a para agregar despues del cursor, A agrega al final de la linea, o abre una linea debajo y entra en modo inserción, y A abre una nueva linea arriba. Con esto se entra en modo inserción y se escribe lo que se desea insertar, luego con Esc se sale del modo inserción y se utiliza para guardar y salir :wq + Enter, solo guardar :w + Enter y salir sin guardar :q! + Enter. Con echo se puede escribir de dos maneras con, echo “cadena” > archivo.txt para sobrescribir el archivo, o escribiendo echo >> archivo.txt para agregar una linea al final del archivo. Con cat se puede escribir varias líneas interactivamente, cat >> archivo.txt, para salir se usa Ctrl + D.

j. Con el comando tail se puede mostrar las 10 últimas líneas del archivo, haciendo tail archivo.txt, con tail –n 20 archivo.txt se muestran las 20 últimas líneas del archivo, con tail –f archivo.txt el archivo queda abierto para poder ver los cambios en vivo, por último, se puede combinar tail –n 30 –f archivo.txt para poder combinar ambas opciones al mismo tiempo.

**11. Investigación de comandos**

Man (manual) - muestra las páginas de manual de los comandos y programas instalados en el sistema.  
Es la primera fuente de consulta en Linux para aprender qué hace un comando, qué parámetros acepta y ejemplos de uso.  
Ej: man <comando> . man ls -> se muestran los comandos de ls.

Estructura de una pagina de manual:

* Name: nombre y breve descripcion.
* synopsis: sintaxis del comando y parámetros.
* Description: descripción detallada de lo que hace.
* Options: lista de parámetros con su explicación.
* Examples: (a veces) ejemplos de uso.
* See also: comandos relacionados.

Parametros importantes:

* Man –k <palabra> -> busca comandos relacionados con una palabra clave (similar a apropos).
* Man –f <comando> -> muestra en que secciones del manual aparece el comando (similar a whatis).
* Man n <comando> -> abre una sección especifica (ej: man 5 passwd muestra el formato del archivo /etc/passwd).

Ubicación en el sistema:

* El binario suele estar en /usr/bin/man.
* Las paginas del manual estan en /usr/share/man/.

Shutdown (apagar) – se usa para apagar, reiniciar o suspender de forma segura la mauqina.  
Envía señales a todos los procesos, evita pérdida de datos y desmonta los sistemas de archivos correctamente antes de apagar.

Sintaxis:  
shutdown [opciones] [tiempo] [mensaje]

Parámetros importantes:

* Shutdown –h now -> apaga inmediatamente.
* Shutdown –r now -> reinicia inmediatamente.
* Shutdown –h +10 -> apaga el sistema en 10 minutos.
* Shutdown –c -> cancela un apagado/reinicio programado.
* Shutdown –r +5 “Reinicio por mantenimiento” -> reinicio en 5 minutos con mensaje para los usuarios.

Detalles:

* Requiere permisos de root o de un usuario con privilegios de administración.
* El comando envía un aviso a los usuarios conectados cuando se programa con tiempo.
* Puede ser reemplazado en sistemas modernos por “systemctl poweroff” o “systemctl reboot”, ya que muchas distribuciones actuales usan **systemd**.

Ubicacion en el sistema:

* El binario se encuentra en /sbin/shutdown o /usr/bin/shutdown.

Ejemplo:  
sudo shutdown -h 23:00 "El sistema se apagará a las 23 hs"

Reboot – se utiliza para reiniciar el sistema.  
Es una alternativa mas directa a shutdown –r now.  
Internamente llama a funciones del kernel para reiniciar el equipo.

Sintaxis:  
reboot [opciones]

Parametros importantes:

* Reboot -> reinicia inmediatamente.
* Reboot –f -> fuerza a reiniciar sin sincronizar ni desmontar el sistema de archivos (peligro de perdida de datos).
* Reboot –p -> apaga la maquina en lugar de reiniciarla (algunos distros la soportan).
* Reboot --halt -> detiene el sistema sin reiniciarlo ni apagarlo (similar al halt).

Detalles importantes:

* Requiere permisos de root o sudo.
* En sistemas con systemd, se recomienda usar:  
  systemctl reboot

Ubicación en el sistema:

* El binario suele estar en /sbin/reboot o /usr/sbin/reboot.

Ejemplo:

Sudo reboot

Halt- detiene el sistema inmediatamente, terminando todos los procesos y deteniendo la CPU.  
A diferencia de shutdown o reboot, no siempre apaga físicamente el equipo (depende de la configuración).

Sintaxis básica:

halt [opciones]

Parametros mas importantes:

* Halt -> detiene el sistema.
* Halt –p -> apaga el sistema despues de detenerlo.
* Halt –f -> fuerza el apagado sin llamar a shutdown.
* Halt --reboot -> reinicia despues de detener (similar a reboot).

Detalles importantes:

* Se necesita permisos de root o sudo.
* En sistemas modernos como systemd, este comando esta redirigido a:  
  systemctl halt
* La diferencia con shutdown –h now es que halt detiene de forma mas inmediata y no permite programar la acción.

Ubicación en el sistema:

* Generalmente en /sbin/halt o /usr/sbin/halt.

Ejemplo:  
sudo halt –p  
Detiene y apaga el sistema inmediatamente.

At – permite programar la ejecución de un comando o tarea en un momento específico en el futuro (ejecuta una sola vez, no repetitivo como cron).  
Sintaxis básica:  
at [hora] [opciones]  
  
Parámetros principales:  
- at <hora> → programa un comando para esa hora específica.  
- atq → lista los trabajos pendientes en cola.  
- atrm <id> → elimina un trabajo por su ID.  
- at -l → igual que atq, lista trabajos.  
- at -c <id> → muestra el contenido del trabajo programado.  
  
Formatos de hora admitidos:  
- at 14:30 → hoy a las 14:30  
- at 10pm → hoy a las 22:00  
- at tomorrow → mañana  
- at now + 1 hour → dentro de una hora  
- at now + 2 days → dentro de 2 días  
  
Ejemplo:  
echo "echo 'Hola Mundo' > /tmp/saludo.txt" | at now + 1 minute  
  
Archivos relacionados:  
- /var/spool/at/ → almacena los trabajos programados.  
- /etc/at.allow y /etc/at.deny → controlan qué usuarios pueden usar at.  
  
  
Netstat – muestra información sobre las conexiones de red, tablas de enrutamiento, estadísticas de interfaces, etc.  
Sintaxis:  
netstat [opciones]  
  
Parámetros principales:  
- netstat -a → todas las conexiones y puertos en escucha.  
- netstat -t → solo conexiones TCP.  
- netstat -u → solo conexiones UDP.  
- netstat -l → sockets en escucha.  
- netstat -n → direcciones y puertos en formato numérico.  
- netstat -p → muestra el PID/programa asociado a cada conexión.  
- netstat -r → tabla de enrutamiento.  
  
Ejemplo:  
netstat -tuln → muestra puertos TCP/UDP en escucha de forma numérica.  
  
Ubicación: /bin/netstat (aunque en distros modernas fue reemplazado por ss).  
  
  
Head – muestra las primeras líneas de un archivo.  
Sintaxis:  
head [opciones] [archivo]  
  
Parámetros:  
- head archivo.txt → primeras 10 líneas por defecto.  
- head -n 20 archivo.txt → primeras 20 líneas.  
- head -c 50 archivo.txt → primeros 50 caracteres.  
  
Ejemplo:  
head -n 5 /etc/passwd  
  
  
Tail – muestra las últimas líneas de un archivo.  
Sintaxis:  
tail [opciones] [archivo]  
  
Parámetros:  
- tail archivo.txt → últimas 10 líneas.  
- tail -n 20 archivo.txt → últimas 20 líneas.  
- tail -f archivo.txt → seguimiento en tiempo real (logs en vivo).  
  
Ejemplo:  
tail -f /var/log/syslog  
  
  
Procesos – un proceso es un programa en ejecución. Cada proceso tiene un identificador único llamado PID y su proceso padre con PPID.  
Atributos: PID, PPID, usuario, estado, prioridad, consumo de CPU/memoria.  
  
Comandos relacionados:  
- top → monitor de procesos en tiempo real.  
- htop → versión mejorada de top (interactiva).  
- ps → lista de procesos.  
- pstree → muestra procesos en forma de árbol.  
- kill <pid> → envía señales (por defecto SIGTERM) a un proceso.  
- pgrep <nombre> → busca procesos por nombre.  
- pkill <nombre> → mata procesos por nombre.  
- killall <nombre> → termina todos los procesos con ese nombre.  
- nice → ejecuta procesos con prioridad modificada.  
- renice → cambia la prioridad de procesos ya en ejecución.  
- xkill → permite matar procesos clickeando ventanas gráficas.  
- atop → monitor de rendimiento avanzado.  
  
Ubicaciones: /bin o /usr/bin según el comando.  
  
  
Proceso de Arranque SystemV  
- Pasos de inicio: encendido → BIOS/UEFI → gestor de arranque → kernel → proceso init → login.  
- INIT: primer proceso en ejecutarse, encargado de inicializar el sistema.  
- Runlevels: niveles de ejecución (0 = apagado, 1 = monousuario, 3 = multiusuario sin GUI, 5 = multiusuario con GUI, 6 = reinicio).  
- Archivo /etc/inittab: define el runlevel por defecto y qué procesos lanzar.  
- Scripts RC: guiones de arranque almacenados en /etc/rc.d o /etc/init.d, ejecutados en orden para iniciar servicios.  
  
  
SystemD  
- Systemd: reemplazo moderno de SystemV init, sistema de inicio y gestión de servicios.  
- Units: archivos que definen servicios, sockets, dispositivos, etc. (ej: ssh.service).  
- Targets: equivalentes modernos de los runlevels (ej: graphical.target, multi-user.target).  
- Comando systemctl: administra servicios y unidades.  
Ejemplos:  
systemctl status ssh  
systemctl start nginx  
systemctl enable apache2  
  
pstree: muestra jerarquía de procesos en árbol.  
  
  
Usuarios  
- Información se guarda en: /etc/passwd, /etc/shadow, /etc/group.  
- UID: identificador único de usuario. GID: identificador único de grupo.  
- Root: superusuario con UID 0.  
- Archivos relacionados: /etc/passwd guarda usuarios, /etc/group grupos.  
  
Comandos principales:  
- useradd / adduser → agregar usuarios.  
- usermod → modificar usuarios.  
- userdel → eliminar usuarios.  
- groupadd → crear grupos.  
- groupdel → eliminar grupos.  
- su → cambiar de usuario.  
- who → muestra usuarios conectados.  
- passwd → cambia contraseña.  
  
  
FileSystem y Permisos  
- Permisos: lectura (r), escritura (w), ejecución (x) aplicables a dueño, grupo y otros.  
- chmod: cambia permisos (ej: chmod 755 archivo).  
- chown: cambia propietario.  
- chgrp: cambia grupo propietario.  
  
Notación octal:  
r=4, w=2, x=1 → se suman (ej: 7 = rwx).  
  
Otros comandos relacionados:  
- mount / umount → montar/desmontar sistemas de archivos.  
- df → muestra espacio libre en discos.  
- du → muestra uso de disco por directorio/archivo.  
- mkfs → formatea una partición.  
- fdisk → particionado de discos.  
- stat → información detallada de un archivo.  
- losetup → administra dispositivos loop.  
- write → enviar mensajes entre usuarios.  
  
  
Procesos Foreground y Background  
- Foreground: proceso que se ejecuta en primer plano (bloquea la terminal).  
- Background: proceso ejecutado con & al final (no bloquea la terminal).  
Ejemplo: sleep 60 &  
- fg → traer un proceso al foreground.  
- bg → enviar un proceso suspendido al background.  
- jobs → lista procesos en segundo plano.  
  
Pipe (|): redirige la salida de un comando a la entrada de otro.  
Ejemplo: ps aux | grep firefox  
  
Redirecciones:  
- > sobrescribe archivo.  
- >> agrega al final.  
- < usa un archivo como entrada.  
- 2> redirige errores.  
  
  
Otros comandos  
- tar: empaquetar y extraer archivos.  
Ejemplo: tar -cvf archivos.tar file1 file2  
- grep: buscar patrones en archivos.  
Ejemplo: grep "root" /etc/passwd  
- gzip: comprimir archivos.  
Ejemplo: gzip archivo.txt  
- zgrep: buscar dentro de archivos comprimidos .gz.  
Ejemplo: zgrep "error" archivo.log.gz  
- wc: contar líneas, palabras y caracteres.  
Ejemplo: wc -l archivo.txt