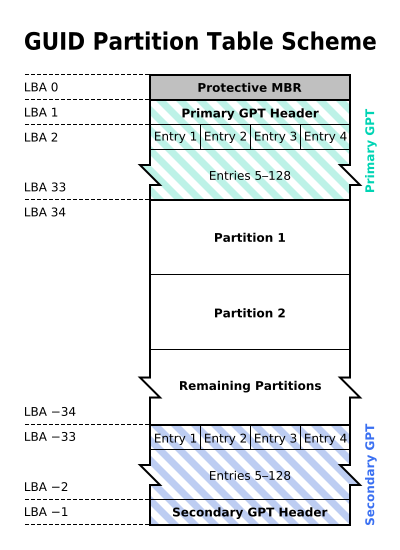
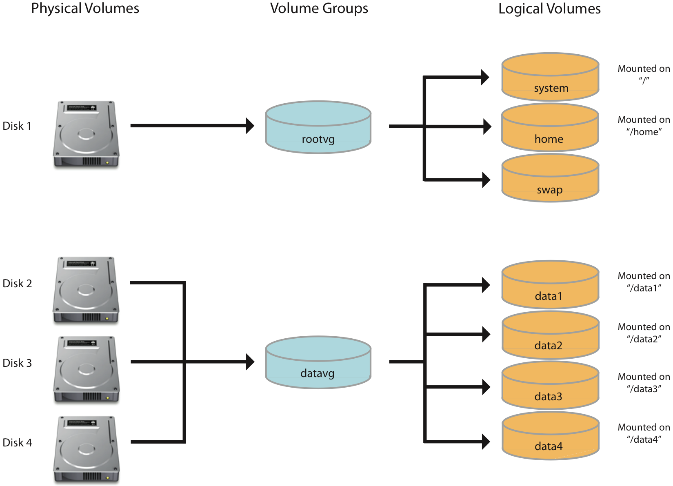
**MATERIA**

**SECCIÓN TEÓRICA**

* Particiones de disco: es un método en el que se crean diferentes espacios dentro de un disco para que quede como si en realidad hubiera más de un disco duro. La tabla de las particiones está repartida en múltiples secciones. Cada sección puede contener la información necesaria para definir una partición. Cada elemento de la tabla de las particiones contiene importantes características relativas a la partición:
  + Los puntos de comienzo y de fin definen el tamaño de las particiones y su posición en el disco.
  + La opción "activa" es utilizada en el arranque de algunos sistemas operativos.
  + El tipo es un número que define previamente el uso que se hará de la partición.
* MBR o GPT: pequeño sector que es leído al momento del *booteo* en el que se almacena la información de particiones.
* MBR, considerado actualmente como Legacy, es utilizado en equipos que estén basados en BIOS/CMOS.
  + Todavía soportado por equipos/placas nuevas.
  + Soporta hasta discos duros de 2TB. Debido a que funciona con 32bits.
  + Límite de solo 4 Particiones Primarias/Extendidas.
  + Gráfico

    Descripción generada automáticamente
* GPT es un estándar más moderno y utilizado en sistemas basados en UEFI, con múltiples ventajas y manejo de información con respecto a MBR.
  + Soportado en su totalidad en equipos, 2012 en adelante.
  + Soporta discos duros de + 2TB. Debido a que funciona con 64bits.
  + Tope teórico es de 9.44ZB (1ZB = 1 billón de TB)
  + Se pueden realizar hasta 128 Particiones Primarias/Extendidas.
* Las particiones están contenidas dentro de los discos duros del sistema.
* La interfaz tiene la forma de archivos de dispositivos de bloque, que se guardan en el directorio /dev.
* Particiones:
  + Su nombre está formado por 3 partes:
    - Un prefijo => en disco SCSI/SATA es sd
    - Una especificación de dispositivo de la unidad => maestro o esclavo.
    - Un número de partición.
    - Texto

      Descripción generada automáticamente
    - EJEMPLO: /dev/sda1
* Al utilizar particiones del tipo MBR, podemos utilizar el comando fdisk para gestionar y administrar el espacio dentro de nuestra unidad de almacenamiento. Su única limitante es que está en 4 particiones primarias máximo por cada disco, y un número de particiones extendidas o lógicas que cambiará de acuerdo con el tamaño del disco duro.
  + Si deseamos realizar particiones extendidas, no debemos utilizar más de 3 particiones primarias, caso contrario, no nos permitirá la creación de particiones extendidas.
* LVM (Logical Volume Manager) es un avanzado sistema de gestión de volúmenes, soportado por el núcleo Linux, que permite al administrador gestionar el espacio de almacenamiento con mucha más flexibilidad que cuando se utilizan métodos tradicionales.
* Son tres los componentes esenciales del LVM:
  + Physical volume/Volúmenes físicos (PV): es un dispositivo de almacenamiento, o más correctamente expresado un dispositivo de bloque. Puede ser un disco duro, una partición, una tarjeta SD, un floppy, un dispositivo RAID, un dispositivo loop (que convierte un fichero a un dispositivo de bloque), un dispositivo cifrado.
  + Volume group/Grupo de volúmenes (VG): Para poder usar el espacio/almacenamiento de un PV, éste debe pertenecer a un Grupo de volúmenes (en adelante VG). Un VG es un “disco” compuesto de UNO o más PVs y que crece simplemente añadiendo más PVs. A diferencia de un disco real, un VG puede crecer con el tiempo, sólo hay que “darle” un PV más.
  + Logical volume/Volúmenes lógicos (LV): Los volúmenes lógicos (en adelante LV) son “el producto final” del LVM. Son estos dispositivos los que usaremos para crear sistemas de ficheros, swap, discos para máquinas virtuales, etc… Por seguir con la analogía del “disco duro virtual” que es el VG, los LVs serían las particiones. Con los que vamos a trabajar realmente.
    - A diferencia de las particiones tradicionales, los LVs pueden crecer (mientras haya espacio en el VG) independientemente de la posición en la que estén, incluso expandiéndose por diferentes PVs.
    - Un LV de 1G puede estar compuesto de 200MB procedentes de un disco duro, 400MB de un RAID software, y 400MB de una partición en un tercer dispositivo físico. El único requisito es que todo los PVs pertenezcan al mismo VG.
* Particionamiento Avanzado con LVM:
  + **Creación de Physical volumes (PV):**
  + pvcreate /dev/sdx
  + **Creación de Physical volumes (PV) con mas de un disco/partición:**
  + pvcreate /dev/sdx /dev/sdx2
  + **Creación de Grupos de volúmenes(VG):**
  + vgcreate nombre-del-vg /dev/vdaX
  + **Creación de Grupo de volúmenes (VG) con mas de un PV:**
  + vgcreate nombre-del-vg /dev/vdaX /dev/vdaX2
  + **Creación de Volúmenes Lógicos (LV):**
  + lvcreate -L Tamaño -n nombre-del-lv nombre-del-vg
  + **Formateo del Volumen Lógico (LV):**
  + mkfs.sistemadearchivo /dev/nombre-del-vg/nombre-del-lv
* Cron es una herramienta de línea de comandos ideada para programar tareas («cron jobs» en inglés), para ejecutarlas periódicamente y automáticamente, en la fecha y hora programada. Este administrador de tareas se puede usar para muchos fines, como:
* Las tareas se ejecutan en segundo plano, sin alterar las funcionalidades de Linux. Los usuarios pueden configurar una lista infinita de tareas cron para que se ejecuten según lo programado —cada x minuto, hora, día, mes o año—. La lista de tareas se almacena en un archivo llamado crontab.
* Las tareas cron tienen la siguiente sintaxis:
* min h d/m mes d/s <comando a ejecutar>
* Atajos de crontab:
  + @reboot (cuando se reinicia el sistema)
  + @anually
  + @monthly
  + @weekly
  + @daily
  + @hourly
* Existen diversas variaciones del comando crontab, entre ellas:
  + crontab -l: lista las tareas agendadas actualmente.
  + crontab -r: elimina todas las tareas agendadas del usuario actual.
  + crontab -u <username>: permite que el usuario actual edite las tareas agendadas de otro usuario (substituya username por el nombre de usuario deseado).
  + crontab -e -u <username>: permite que el usuario actual edite sus propias tareas agendadas o las tareas agendadas de otro usuario (substituya username por el nombre de usuario deseado).
  + crontab -i: solicita confirmación antes de eliminar todas las tareas agendadas del usuario actual.
* Programe una tarea para que todos los días a las 2:30 am se realice un respaldo de los archivos de la carpeta “/clipsfortnite” en la carpeta “/respaldo” (hint: comando “cp”)
  + R: @daily cp -r /home/usuario/documentos /home/usuario/respaldo
* Programe una tarea que muestre un recordatorio para tomar un descanso cada hora en la terminal. (hint: terminal = /dev/tty1)
  + R: @hourly echo "Es hora de tomar un descanso" > /dev/tty1
* Comando cut: Para extraer determinados caracteres de una o varias líneas delimitándolos por su posición, tenemos el comando cut. Con cut podemos extraer partes de una línea indicando la posición de bytes y caracteres. También, si indicamos un delimitador, podemos extraer los campos del archivo.
  + Para extraer campos/palabras con -f (field) debemos indicar un delimitador con -d.
  + Un delimitador puede ser el espacio en blanco (-d “ “), dos puntos (-d “:”), punto y coma (-d “;”)
  + Con -f indicamos el número del campo a extraer. O de los campos, ya que podemos indicar más de un campo, incluso rangos.
  + Por defecto, nos va a devolver los campos separados por espacios en blanco, pero podemos modificar esto y que nos devuelva el resultado con otro separador. Lo haremos con –output-delimiter y el separador que queramos.
* Utilizando cut, extraiga los usuarios y los directorios de cada usuario, y almacene esta información en un archivo de texto llamado usuarios.txt
  + R: cut -d: -f1,6 /etc/passwd > usuarios.txt
* Utilizando algún editor de texto, abra el archivo usuarios.txt
  + R: vim usuarios.txt
* Detenga el editor de texto (ctr+z):
  + R: Presionar ctrl+z estando en el vim, dejará a vim en segundo plano
* Obtenga el PID del editor de texto
  + R: jobs -l
* Mate el proceso correspondiente al editor de texto.
  + R: kill
* Procesos: Un proceso es una instancia de un programa en ejecución, a los procesos frecuentemente se les refiere como tareas. El contexto de un programa que está en ejecución es lo que se llama un proceso.
* Cuando Linux se ejecuta, el kernel de Linux tiene la primera prioridad de ejecución, conocida como PID 1 (Process ID). En versiones anteriores de Linux, este proceso era conocido como init que está basado en la forma en la que sistemas antiguos de Unix arrancaban el sistema.
* Las versiones modernas de Linux utilizan systemd que intenta coordinar la manera en que los procesos son manejados. Como comentamos PID1 es el proceso padre, todos los demás procesos ejecutados a partir de este son procesos hijos.
* Programas y procesos son entidades distintas. En un sistema operativo multitarea, múltiples instancias de un programa pueden ejecutarse simultáneamente. Cada instancia es un proceso separado. Por ejemplo, si cinco usuarios desde equipos diferentes ejecutan el mismo programa al mismo tiempo, habría cinco instancias del mismo programa, es decir, cinco procesos distintos.
* Cada proceso que se inicia es referenciado con un número de identificación único conocido como Process ID PID, que es siempre un entero positivo. Prácticamente todo lo que se está ejecutando en el sistema en cualquier momento es un proceso, incluyendo el shell, el ambiente gráfico que puede tener múltiples procesos, etc. La excepción a lo anterior es el kernel en si, el cual es un conjunto de rutinas que residen en memoria y a los cuales los procesos a través de llamadas al sistema pueden tener acceso.
* Dependiendo de la forma en que corren estos programas en LINUX se los puede clasificar en tres grandes categorías:
  + Procesos Normales: generalmente son lanzados en una terminal (tty) y corren a nombre de un usuario, es decir, son los programas que utiliza el usuario generalmente y se encuentran conectados a una terminal.
  + Procesos Daemon: corren a nombre de un usuario y no tienen salida directa por una terminal, es decir corren en 2º plano. Generalmente los conocemos como servicios. La gran mayoría de ellos en vez de usar la terminal para escuchar un requerimiento lo hacen a través de un puerto.
  + Procesos Zombie: es un proceso que ha completado su ejecución, pero aún tiene una entrada en la tabla de procesos. Esto se debe a que dicho proceso (proceso hijo) no recibió una señal por parte del proceso de nivel superior (proceso padre) que lo creó informándole que su vida útil ha terminado. Se pueden deber a errores de programación, a situaciones no contempladas por el programador y generalmente provocan lentitud y/o inestabilidad en el Sistema.
* Los principales estados en los que pueden encontrarse los procesos en Linux/Unix son los siguientes:
  + running (R) : Procesos que están en ejecución.
  + sleeping (S) : Procesos que están esperando su turno para ejecutarse.
  + stopped (D) : Procesos que esperan a que se finalice alguna operación de Entrada/Salida.
  + zombie (Z) : Procesos que han terminado pero que siguen apareciendo en la tabla de procesos.
* Para ver el estado de los procesos en el sistema operativo tenemos varios comandos.
  + Tecleando el comando top desde una terminal se nos mostrarán todos estos procesos; este comando monitoriza dinámicamente los procesos del sistema mostrando su estado, uso de CPU, cantidad de memoria, tiempo desde su inicio, nombre, etc.
* Algunos procesos inician otros procesos, de esta forma se convierten en padres de estos. Para poder ver esta jerarquía podemos utilizar el comando ps que muestra en la última columna como la jerarquía de los procesos y subprocesos está organizada por tabuladores.
  + Comando ps aux
* Por defecto ps sólo muestra los procesos con el mismo user id efectivo (EUID) que el del usuario que lo ejecuta. Cuando utilizamos ps con las opciones "aux" el resultado es:
  + a: eliminar la restricción BSD "only yourself" para agregar procesos de otros usuarios
  + u: utilizar el formato orientado al usuario
  + x: eliminar la restricción BSD "must have a tty" para agregar procesos que no tengan una tty asociada
* La salida de "ps aux" es una tabla donde cada fila es un proceso y las columnas contienen la siguiente información:
  + USER: usuario con el que se ejecuta el proceso
  + PID: ID del proceso
  + %CPU: porcentaje de tiempo que el proceso estuvo en ejecución desde que se inició
  + %MEM: porcentaje de memoria física utilizada
  + VSZ: memoria virtual del proceso medida en KiB
  + RSS: "resident set size", es la cantidad de memoria física no swappeada que la tarea a utilizado (en KiB)
  + TT: terminal que controla el proceso (tty)
  + STAT: código de estado del proceso
  + STARTED: fecha de inicio del proceso
  + TIME: tiempo de CPU acumulado
  + COMMAND: comando con todos sus argumentos
* Los códigos de estado del proceso pueden ser:
  + D: suspendido no interrumpible (generalmente esperando E/S)
  + R: en ejecución o listo para ejecutarse (en cola)
  + S: suspendido interrumpible (esperando que se complete un evento)
  + T: detenido, por una señal de control de trabajo o porque está siendo traceado
  + W: paginado (no válido a partir del kernel 2.6.xx)
  + X: muerto (nunca debe ser visto)
  + Z: proceso difunto ("zombie"), terminado, pero no reclamado por el proceso padre
  + <: alta prioridad
  + N: baja prioridad
  + L: tiene páginas bloqueadas en memoria
  + s: es un líder de sesión (se trata de un proceso que ha iniciado una nueva sesión)
  + l: es multi hilado
  + +: está en el grupo de procesos en primer plano
* Podemos también instalar otra utilidad que nos permite ver información en tiempo real de los procesos, esta herramienta se llama htop (esta herramienta no se encuentra en los repositorios por defecto de rocky, por lo tanto, debemos instalar algún repositorio extra, como remi o epel).
  + #dnf -y install epel-release
  + #dnf makecache
  + #dnf -y install htop
  + #htop

**DEFINICIONES:**

* Formatear: creación de sistemas de archivos en el disco (se escribe la información al disco ordenando el espacio vacío).
* Sistema de archivos:
  + Un sistema de archivos parte el espacio que queda en pequeños segmentos de tamaño consistente. Estos segmentos son conocidos como bloques.
  + No hay un único sistema de archivo. Un disco puede tener varios sistemas de archivos, aunque hay sistemas de archivos que no son compatibles entre ellos. Un sistema operativo no tiene porqué soportar varios sistemas de archivos.
* Fragmentación: los bloques utilizados no tienen necesariamente una región contigua; los bloques utilizados pueden encontrarse en posiciones separadas.
* Particiones de disco: es un método en el que se crean diferentes espacios dentro de un disco para que quede como si en realidad hubiera más de un disco duro.
* Tipos de particiones en Linux: primarias y extendidas.
* MBR: pequeño sector que es leído al momento del *booteo* en el que se almacena la información de particiones. El MBR es utilizado en sistemas BIOS/CMOS. Es considerado actualmente como legado pues soporta un máximo de 4 particiones y soporta discos duros de hasta 2TB debido a que funciona con 32bits.
* GPT: cumple la misma función que el MBR, pero GPT es un estándar más moderno y utilizado en sistemas basados en UEFI. Soporta discos duros de + 2TB, debido a que funciona con 64bits. Se pueden realizar hasta 128 Particiones.
* LVM (Logical Volume Manager) es un avanzado sistema de gestión de volúmenes, soportado por el núcleo Linux, que permite al administrador gestionar el espacio de almacenamiento con mucha más flexibilidad que cuando se utilizan métodos tradicionales. Está compuesto por 3 componentes:
  + Volúmenes físicos (PV): es un dispositivo de almacenamiento, o más correctamente expresado un dispositivo de bloque.
  + Grupo de volúmenes (VG): es un “disco” compuesto de 1 o más PVs y que crece simplemente añadiendo más PVs. A diferencia de un disco real, un VG puede crecer con el tiempo, sólo hay que “darle” un PV más.
  + Volúmenes lógicos (LV): son “el producto final” del LVM, equivalente a las particiones del “disco duro virtual”.
* Cron: es una herramienta de línea de comandos ideada para programar tareas.
  + Las tareas cron tienen la siguiente sintaxis:
  + min h d/m mes d/s <comando a ejecutar>
* Tareas: Las tareas se ejecutan en segundo plano, sin alterar las funcionalidades de Linux. Los usuarios pueden configurar una lista infinita de tareas cron para que se ejecuten según lo programado.
* Procesos: Un proceso es una instancia de un programa en ejecución, a los procesos frecuentemente se les refiere como tareas. El contexto de un programa que está en ejecución es lo que se llama un proceso.
* Diferencia entre un proceso y una tarea: el proceso es prácticamente todo lo que se está ejecutando en el sistema en cualquier momento, mientras que una tarea es todo lo que nosotros ejecutamos para que se mantengan los procesos.
* PID: Cada proceso que se inicia es referenciado con un número de identificación único conocido como Process ID (PID), que es siempre un entero positivo.
* Tipos de proceso:
  + Normales
  + Zombie: terminó pero sigue apareciendo, hay que aplicar kill.
  + Daemon: ocurre en segundo plano.
* Estados de un proceso:
  + Running
  + Sleeping
  + zombie
  + stopped

**SECCIÓN PRÁCTICA**

**#COMANDOS IMPORTANTES**

fdisk -l #listar los discos implementados

kill #cancela una operación, puede ser crt+c también

:w #escribir en vim

:q #salir de vim

**#INIT (precondición)**

su

cd

**#CREAR PARTICIONES**

**##Opción 1: con "fstab"**

fdisk /dev/sdb #llama al segundo disco. RECORDAR: disco1 = sda, disco2 = sdb, disco3 =sdc...

n #crear nueva partición

p #primaria

enter #valor predeterminado

enter #valor predeterminado

+(TAMAÑO)G #por ejemplo: +17G hace una partición de 17 GB de tamaño. OJO: es número+G, no GB.

#se crea una partición de tipo Linux

**##Opción 2: con LVM**

#con el disco creado, se le reconoce

pvcreate /dev/sdb #crea volumen físico

vgcreate test /dev/sdb #se crea el grupo volumen de la forma: comando + nombre del grupo + nombre partición

#es decir, la nueva ubicación de cada partición será: /dev/test/nombre\_partición

lvcreate -L 17G -n unoo test #se asigna el volumen lógico de la forma lvcreate -L TAMAÑO -n NOMBRE\_PARTICIÓN NOMBRE\_GRUPO

lvcreate -L 3G -n doos test #se asigna volumen lógico a la segunda partición

#ambas particiones creadas

**#CAMBIAR TIPO DE PARTICIÓN**

fdisk /dev/sdb #si ya estamos ahí, se salta

t #Orden para editar tipo

#número de la partición a editar

L #ver código hexadecimal del tipo deseado, por ejemplo: SWAP = 86

86 #tipo deseado, en este ejemplo: SWAP

**#GUARDAR PARTICIONES**

w #Orden de guardar cambios

**#FORMATEAR PARTICIONES**

mkfs.xfs /dev/sdb1 #comando para formatear partición Linux + nombre de la partición

mkswap /dev/sdb2 #comando para formatear partición SWAP + nombre de la partición

#OJO: las particiones creadas con LVM tienen una ubicación distinta, del tipo: /dev/NOMBRE\_GRUPO/NOMBREPARTICIÓN

**#CREAR MONTAJE AUTOMÁTICO DE PARTICIÓN**

blkid /dev/sdb1 #comando para ver id + nombre de la partición, copiamos la UUID

vim /etc/fstab #abrimos el vim para montaje automático y cambiamos el modo a "INSERTAR"

#copiamos desde UUID hasta el final y pegamos abajo

#editamos el UUID de abajo por el que copiamos y /boot por el directorio de montaje

#editamos la dirección abajo de UUID por el nombre de la partición restante

:wq #guardar y salir de vim

**PARTICIONAMIENTO AVANZADO CON LVM:**

**#Creación de Physical volumes (PV):**

pvcreate /dev/sdx

**#Creación de Physical volumes (PV) con mas de un disco/partición:**

pvcreate /dev/sdx /dev/sdx2

**#Creación de Grupos de volúmenes(VG):**

vgcreate nombre-del-vg /dev/vdaX

**#Creación de Grupo de volúmenes (VG) con mas de un PV:**

vgcreate nombre-del-vg /dev/vdaX /dev/vdaX2

**#Creación de Volúmenes Lógicos (LV):**

lvcreate -L Tamaño -n nombre-del-lv nombre-del-vg

**#Formateo del Volumen Lógico (LV):**

mkfs.sistemadearchivo /dev/nombre-del-vg/nombre-del-lv