CSO TP2

1)

a) Vi, Vim, Nano

b) La diferencia de los editores de texto con respecto a dichos comandos, es que los comandos imprimen en la terminal el contenido de un script, mientras que en el editor de texto no solamente lo podes ver en pantalla, sino que también es posible editarlos.

c) Algunos de los comandos mas comunes son:

* w: para escribir cambios
* q o q!: salir del editor
* dd: cortar
* y: copiar al portapapeles
* p: pegar desde el portapapeles
* u: deshacer
* /: buscar frase

2)

a) Los pasos son los siguientes:

* Encendido y POST (Power-On Self Test): Al encender la PC, la bios o uefi hace una serie de pruebas básicas de HW conocidas como POST. Estas pruebas verifican el correcto funcionamiento de los componentes.
* Carga del gestor de arranque (bootloader): Luego del POST, la BIOS o UEFI busca un dispositivo de almacenamiento que contenga un sector de arranque valido. Una vez encontrado, el gestor se carga en memoria.
* Carga del Kernel de Linux: El bootloader carga el kernel de Linux seleccionado en la memoria. Junto con el kernel, también se carga un disco RAM inicial (initramfs o initrd) que contiene los controladores y scripts necesarios para montar el sistema de archivos raíz.
* Inicializacion del kernel: El kernel comienza su proceso de inicializacion, detectando y configurando los dispositivos de HW disponibles. Luego monta el sistema de archivos raíz y ejecuta el primer proceso del sistema, generalmente llamada init.
* Ejecución del proceso ‘init’ (o su reemplazo): Este proceso (o su versión moderna ‘systemd’ o ‘OpenRC’) se ejecuta como proceso principal y se encarga de iniciar otros procesos esenciales para el funcionamiento del sistema. ‘init’ lee sus archivos de configuración para determinar que servicios deben iniciarse.
* Lanzamiento de servicios y Daemons: Estos servicios son fundamentales para el correcto funcionamiento del entorno del usuario y otros aspectos del sistema.
* Presentación del gestor login: finalmente, cuando todo lo anterior esta cargado, el sistema presenta el gestor de login el cual permite al usuario autenticarse y acceder a su sesión de usuario.

b) El proceso de ‘init’ es ejecutado por el propio kernel de Linux. El proceso se ejecuta como primer proceso en el sistema, asignándole el identificador de proceso (PID) 1. Este proceso se convierte en ‘padre’ de todos los demás procesos que se ejecutan posteriormente en el sistema.

c) Los RunLevels son niveles de funcionamiento predefinidos en los sistemas Unix y GNU/Linux tradicionales que determinan qué servicios y procesos se deben ejecutar. Cada uno está asociado con un conjunto especifico de servicios o estados del sistema, como el modo de rescate, el modo multiusuario sin red, o el modo gráfico completo. Estos son gestionados por el proceso ‘init’.

d) Los RunLevel tradicionales hacen referencia:

* Runlevel 0: Apagado. Este nivel detiene todos los procesos y apaga el sistema.
* Runlevel 1: Modo de usuario único. Se inicia el sistema en un estado mínimo con acceso sólo para el usuario root, sin servicios de red ni otros usuarios. Se utiliza generalmente para tareas de mantenimiento.
* Runlevel 2: Modo multiusuario sin red. Se permite el acceso a múltiples usuarios, pero no se inicia la red.
* Runlevel 3: Modo multiusuario con red. Se permite el acceso a múltiples usuarios y se inician todos los servicios necesarios, incluyendo la red, pero sin interfaz gráfica.
* Runlevel 4: No definido/Reservado. En muchos sistemas, este nivel no tiene un propósito definido y se deja para uso personalizado.
* Runlevel 5: Modo multiusuario con red y entorno gráfico. Similar al runlevel 3, pero además se inicia un gestor de ventanas o entorno gráfico.
* Runlevel 6: Reinicio. Este nivel detiene todos los procesos y reinicia el sistema.

Los runlevel se definen en el archivo /etc/inittab, este especifica el runlevel por defecto que debe ser ejecutado al inicio del sistema, entre otras configuraciones relacionadas con el proceso de arranque. No todos los sistemas respetan estos estándares, en sistemas mas modernos al utilizar el ‘systemd’ en lugar del ‘init’, el conceptos de runlevels fue reemplazado por “Targets”. Aunque ‘systemd’ aun mantiene compatibilidad con los runlevels tradicionales.

e) El archivo /etc/inittab es un archivo de configuración utilizado en sistemas Unix y GNU/Linux que emplean el sistema de inicio SysVinit. Su finalidad principal es definir cómo se comporta el sistema durante el proceso de arranque y qué acciones deben ejecutarse en los diferentes runlevels (niveles de ejecución). Este archivo controla aspectos como el runlevel por defecto, los procesos que deben iniciarse en cada runlevel, y cómo debe manejarse la consola del sistema.

Su finalidad:

* Definir el runlevel por defecto: Indica en qué runlevel debe iniciar el sistema después del arranque.
* Gestionar procesos esenciales: Especifica qué procesos deben iniciarse o detenerse cuando el sistema cambia de un runlevel a otro.
* Configurar la consola: Define cómo deben comportarse los terminales virtuales y las consolas de inicio de sesión.

Tipo de informacion que almacena:

* Runlevel predeterminado: Especifica el nivel de ejecución en el que el sistema debería iniciar.
* Inicio de procesos específicos: Define comandos o scripts que deben ejecutarse cuando el sistema entra en un runlevel particular.
* Gestión de consolas y terminales: Configura cómo deben iniciarse los procesos de consola y gestión de sesiones (como getty) en diferentes runlevels.
* Reinicio y apagado: Controla los comandos o scripts que deben ejecutarse durante el reinicio o apagado del sistema.

La estructura en el ‘inittab’ esta organizada entre lineas, donde cada linea sigue una estructura especifica: ‘id:runlevels:action:process’, donde:

* id: Es un identificador único de 1 a 4 caracteres que identifica la entrada. Se usa principalmente para identificar procesos que se gestionan en las consolas.
* runlevels: Especifica los runlevels en los que debe ejecutarse esta entrada. Se indica como una cadena de números, por ejemplo, 2345.
* action: Define qué acción debe realizarse. Puede ser uno de varios comandos predefinidos como respawn, wait, once, ctrlaltdel, initdefault, etc.
* process: Es el comando o script que debe ejecutarse cuando se cumple la condición del runlevel y la acción especificada.

3)

a) Systemd es un sistema de gestión de servicios y un sistema de inicialización utilizado en la mayoria de distribuciones modernas de Linux. Reemplaza al sistema ‘SysVinit’ o ‘init’, ofreciendo una gestion mas eficiente y flexible de los servicios y procesos del sistema.

b) Las unidades o ‘Units’ son archivos de configuracion que describen como gestionar los servicios, dispositivos, puntos de montaje, sockets, y otros aspectos del sistema.

c) El comando ‘Systemctl’ sirve para gestionar servicios. Con este comando se puede inciar, detener, reiniciar y verificar el estado de los servicios. Ademas, permite habilitar o deshabilitar servicios para que se inicien automaticamente.

d) Los ‘Targets’ son equivalentes modernos a los runlevel tradicionales, pero son mucho mas flexibles y modulares.

e) Este comando imprime en pantalla un árbol de servicios que se están ejecutando en el sistema.

4)

a) La informacion del usuario se almacena en varios archivos clave dentro del sistema de archivos. Estos archivos contienen detalles sobre las cuentas de usuarios, como nombres, contraseñas encriptadas, Ids de usuario, ids de grupo, directorios de inicio, y shells predeterminados.

b) UID: ID de usuario. GID: ID de grupo. No pueden coexistir UIDs iguales ya que cada usuario tiene un identificador unico.

c) El usuario root es el superusuario del sistema, este tiene el permiso de ejecutar comandos o modificar partes sensibles del SO, como por ejemplo editar el archivo /etc/enviroment, o instalar paquetes en el sistema. No existe otro usuario root, pero siendo un usuario común se pueden ejecutar comandos bajo el usuario root poniendo la palabra ‘sudo’ adelante del comando que se quiere ejecutar. El UID/GID de root es siempre 0.

5)

a) En un sistema GNU/Linux, los permisos sobre archivos y directorios se definen mediante un sistema de control de acceso basado en tres atributos principales: el propietario del archivo (usuario), el grupo al que pertenece, y otros (todos los demás usuarios). Cada archivo y directorio tiene permisos asociados que determinan qué acciones pueden realizar el propietario, los miembros del grupo, y otros usuarios en el sistema.

b)

* chmod: cambia los bits del modo del archivo. Parámetros: [OPCIÓN] … MODO[,MODO] … ARCHIVO …
* chown: cambia el dueño del archivo y su grupo. Parámetros: [OPCION] … [DUEÑO][:[GRUPO]] [ARCHIVO]
* chgrp: cambia el dueño del grupo. Parametros: [OPCION] … GRUPO ARCHIVO

c) La notación octal es una manera compacta de representar los permisos de archivos y directorios. Esta confirmada por 3 caracteres, el primero es el usuario, el segundo el grupo y el último son los otros. Los permisos van del 0 al 7, donde 0 es que no tiene ningún permiso y 7 que tiene permiso de lectura, escritura y ejecución.

D) Si, existe dicha posibilidad y esto es posible gracias a:

* errores de configuración
* vulnerabilidad de la seguridad
* escalada de privilegios
* errores de software
* manipulación de permisos

e) Full Path Name: especifica la ubicación de un archivo o directorio en el sistema de archivos a partir del directorio raíz del sistema. Incluye todos los directorios necesarios para llegar al archivo o directorio en cuestión. Es absoluta y no depende del directorio actual.

Relative Path Name: especifica la ubicación de un archivo o directorio a partir del directorio de trabajo actual.

f) El comando para saber en que directorio se encuentra actualmente es ‘pwd’. Para acceder al directorio personal sin la necesidad de escribir todo la ruta, se escribe ‘cd’ a secas y te ubica en tu directorio personal. También, para acortar el acceso al directorio personal se utiliza el signo ‘~’, esto puede ayudar mucho si se quiere acceder a una carpeta interna del home. Por ejmplo: quiero acceder a la carpeta ‘Documentos/CSO2024’, puedo en la terminal, desde cualquier directorio, introducir ‘cd ~/Documentos/CSO2024’ en vez de ‘cd /home/user/Documentos/CSO2024’.

6)

a) Un proceso es una instancia de ejecución de un programa. Un proceso consta de varios componentes, como código ejecutable, datos y recursos del sistema, incluyendo espacio en memoria, descriptores de archivos, etc.

PID (Process ID): es un identificador único que se le da a un proceso para identificarlos entre los demás.

PPID (Parent Process ID): es el identificador del proceso padre que creo el proceso actual.

Atributos: en GNU/Linux existen otros atributos:

* Estado: el estado del proceso indica si esta en ejecucion, detenido, en espera, etc.
* Prioridad: determina la cantidad de tiempo de CPU que se le asigna en comparación con otros procesos.
* Recursos: Incluye los recursos asignados al proceso como memoria, descriptores de archivos y otros recursos del sistema.
* Usuario y grupo: estos atributos determinan los permisos y derechos de acceso del proceso.
* Ruta del ejecutable
* Señales: son mensajes que se pueden enviar a un proceso para que realice ciertas acciones, como detenerse, reanudarse, o terminar.
* Memoria: indica cuanta memoria utiliza el proceso.
* Hijos y padres

b) Los comandos (o algunos) que sirven para ver los procesos son ‘top’, que viene incluido en todas las distros de Linux, y ‘htop’ que es una versión mejorada del anterior pero no viene pre-instalada en todas las distros.

c) Un proceso que se ejecuta en Background significa que esta siendo ejecutado en segundo plano y que no es una ventana activa en el entorno gráfico o no esta siendo ejecutado en la terminal actual. Foreground es lo contrario, es en primer plano.

d) Para ejecutar un proceso en segundo plano, se debe añadir ‘&’ al final del comando, por ejemplo ‘sleep 60 &’, el cual pone a la computadora en suspensión en 60 segundos.

Para pasar un proceso de segundo a primer plano se utiliza el comando ‘fg’, y viceversa ‘bg’.

e) El pipe (|) es un operador en la línea de comandos de Unix/Linux que se utiliza para redirigir la salida de un comando como la entrada de otro comando. Esto permite encadenar varios comandos para procesar datos de manera eficiente. Se utiliza para encadenar comandos, procesar datos de forma eficiente y simplificar tareas complejas. Ejemplos: ‘ls | wc -l’ o ‘ls | grep “archivo”.

f) En Linux, la redirección se utiliza para controlar el flujo de entrada y salida de los comandos en la línea de comandos. Permite enviar la salida de un comando a un archivo, como entrada para otro comando, o manejar errores.

g) El comando ‘kill’ sirve para terminar un proceso forzosamente. Para indicar el proceso a terminar, se debe introducir su PID. Por ejemplo, se tilda firefox y su PID es 1800, entonces en la terminar introduzco ‘kill 1800’ y termina forzosamente el proceso de firefox.