CSO Resumen Practica Mod1

Age	2024

Practica 1

Sistema operativo

Un SO es un programa que actua en principio como intermediario entre el usuario y el hardware. Su proposito es crear un entorno como y eficiente para la ejecucion de programas. Su obligacion es garantizar el correcto funcionamiento del sistema. Funciones principales: administrar memoria, cpu y dispositivos.

GNU/Linux

“GNU's Not Unix”

Es un sistema operativo tipo Unix, pero libre, es gratuito, de lobre distribucion y de codigo abierto, lo que nos permite estudiarlo y personalizarlo. Caracteristicas generales:

- Es multiusuario, multitarea y multiprocesador y altamente portable
- Posee diversos interpretes de comandos, de los cuales algunos son programables
- Permite el manejo de usuarios y permisos
- Todo es un archivo (hasta los dispositivos y directorios)
- Cada directorio puede estar en una particion diferente
- Es case sensitive y de codigo abierto

Los componentes fundamentales de GNU/Linux son el kernel, la shell y el filesystem.

Nucleo/Kernel

El kernel es el encargado de ejecutar programas y gestionar dispositivos de HW. Tambien hace posible el enlace con el SW. Sus funciones mas importantes son la administracion de memoria, cpu y E/S. En un sentido estricto, es el SO.

Es un nucleo monolitico hibrido. Monolitico significa que los drivers y codigo se ejecutan en modo privilegiado. Lo que lo hace hibrido es la capacidad de cargar y descargar funcionalidad a traves de modulos.

- A: denota version
- B: denota mayor revision
- C: denota menor revision, solo cambia cuando hay nuevos drivers o caracteristicas
- D: cambia cuando se corrige un grave error sin agregar nueva funcionalidad (desuso)

▼ Shell

Tambien conocido como CLI (command line interface), es el modo de comunicacion entre el usuario y el SO. Ejecuta programas a partir del ingreso de comandos, donde cada usuario puede tener una interfaz o shell. Los distintos tipos de shell se diferencian en caracteristicas y usabilidad.

▼ FileSystems

Organiza la forma en la que se almacenan los archivos en dispositivos de almacenamiento (fat,ntfs ext2, ext3, etc). El adoptado por GNU/Linux es Extended(v,v3,v4). Los directorios mas importantes son:

- / tope de estructura de directorios (C:\)
- /home se almacenan archivos de usuarios (documents)
- /var informacion que varia de tamaño (logs, BD, spools)
- /etc archivos de configuracion
- /bin archivos binarios y ejecutables
- /dev enlace a dispositivos
- /usr aplicaciones de usuarios

▼ Distribuciones

Una distribucion es una customizacion de GNU/Linux formada por una version de kernel y determinados programas con sus configuraciones

▼ MBR, MBC y Particiones

El MBR es un sector reservado del disco fisico, existe uno en todos los discos y en caso de que exista mas de uno, hay uno designado que es el Primary Master Disk, solo se puede designar un PMD. Se encuentra en el cilindro 0, cabeza 0, sector 1.

El tamaño del MBR es de 512 bytes, donde los primeros bytes corresponden al MBC (master boot code), que es un pequeño codigo que permite arrancar el SO. Mientras que la tabla de particiones es de 64 bytes.

La ultima accion del BIOS es leer el MBC. Lo lleva a memoria y lo ejecuta. Si se tiene un sistema instalado → bootloader de MBC tipico, sino → uno diferente (multietapa).

Debido al tamaño acotado de MBR, se restringe a 4 la cantidad de particiones primarias. Una de las 4 puede ser extendida, la cual se subdivide en volumenes logicos.

- Primaria: division cruda del disco (max 4 por disco). Se almacena informacion de la misma en el MBR.
- Extendida: sirve para contener unidades logicas en su interior. solo puede existir una particion de este tipo por disco. no se define un tipo de FS directamente sobre ella.
- Logica: ocupa la totalidad o parte de la particion extendida y se le define un tipo de FS. las particiones de este tipo se conectan como una lista enlazada.

Como minimo es necesario una particion (para el /). Es recomendable al menos crear 2 (/ y Swap). Para crearlas se utiliza el particionador, existen dos tipos:

- Destructivos: permiten crear y eliminar particiones (fdisk)
- No destructivo: permiten crear, eliminar y modificar particiones(fips, gparted)

▼ Emuladores/virtualizadores

- Emulacion: Emulan HW. Tienen que implementar todas las instrucciones de la CPU, es muy costosa y poco eficiente. Ademas, permite ejecutar arquitecturas diferentes a las soportadas por el HW.
- Virtualizacion completa: Permite ejecutar SO huespedes en un sostema anfitrion (host), utilizando en el medio un hypervisor o monitor de maquina virtual. El SO huesped debe estar soportado en la arquitectura anfitriona. Es mas eficiente que la emulacion.
- Paravirtualizacion: Permite correr SO modificado exclusivamente para actuar en entornos virtualizados, permitiendo mayor eficiencia que la virtualizacion.

La principal diferencia entre ellos son:

- Los virtualizadores aprovechan el CPU sobre la que estan trabajando, lo cual los hace mas veloces.
- En un emulador se puede correr cualquier arquitectura. En un virtualizador solo se puede correr la arquitectura virtualizada.

▼ Gestor y Proceso de arranque

La finalidad del bootloader es la de cargar una imagen de Kernel (SO) de alguna particion para su ejecucion. Se ejecuta luego del codigo del BIOS, y hay dos modos de instalacion: en el MBR y en el sector de arranque de la particion raiz o activa (Volume Boot Record)

El proceso de arranque se encarga de iniciar y cargar el sistema en la maquina y se denomina bootstrap. En las arquitecturas x86, el BIOS es el resposable de iniciar la carga del SO a traves del MBC, cargando el programa de booteo desde el MBR.

Se carga el programa de booteo desde el MBR. Luego, el gestor de arranque lanzado desde el MBC carga el kernel, prueba y hace disponibles los dispositivos (a traves de POST), luego pasa el control al proceso INIT. El proceso de arranque se ve como una serie de pequeños programas de ejecucion encadenada.

▼ GPT

Se mantiene un MBR para tener compatibilidad con el esquema BIOS, usando un modo de direccionamiento logico. La cabecera GPT y la tabla de particiones estan escritas al principio y al final del disco.

▼ UEFI

Define la ubicacion de gestor de arranque y la interfaz entre este y el firmware. Expone informacion para los gestores de arranque con informacion del HW, punteros, provee un BootManager para cargar aplicaciones UEFI (bootloader es un tipo de aplicacion ahora).

▼ Secure boot

Propone mecanismos para un arranque libre de codigo malicioso. Las aplicacion y drivers UEFI son validadas para verificar que no fueron alteradas, utilizando pares de claves asimetricas. Se almacenan en el firmware una serie de claves publicas que sirven para validar que las imagenes esten firmadas por un proveedor autorizado. Si la clave esta vencida o fue revocada, la verificacion puede fallar.

▼ Practica 2

▼ Configuracion de discos

- Discos IDE (Integrated Device Electronics): los discos pueden configurarse como Master o Slave y conectarse a diferentes buses. Ejemplos de nomenclatura:
  - /dev/hda: Master en el primer bus IDE
  - /dev/hdb: Slave en el primer bus IDE
  - /dev/hdc: Master en el segundo IDE
  - /dev/hdd: Slave en el segundo IDE

  Las particiones primarias son del 1-4. Las logicas del 5 en adelante.

  - Discos SCSI y SATA: se usa una denominacion similar, usando /dev/sda, etc
    - Nomenclatura Udev: permite la asignacion dinamica de nombres basados en UUID (ID universal) o labels (etiquetas). Evita problemas al cambiar de HW.

▼ Herramienta de particionado

La diferencia entre fdisk (destructiva) y gparted (no destructiva) es que fdisk sobrescribe los datos del disco.

▼ Editor de textos Vim

Modo insert (Ins, I). Modo visual (v). Modo de ordenes o normal (Esc)

▼ Gestion de usuarios

Todo usuario debe poseer credenciales para acceder al sistema. Root es el administrador y los demas tienen permisos limitados. Los archivos importantes incluyen:

- /etc/passwd: contienen informacion sobre los usuarios
- /etc/group: gestiona los grupos del sistema
- /etc/shadow: almacena las contraseñas cifradas

No puede haber más de un usuario con el nombre **root**, pero **sí** pueden existir otros usuarios que compartan los mismos **privilegios** que root si se les asigna el **UID 0** superusuario.

▼ Permisos

Los permisos se asignan a archivos y directorios usando una notacion octal.

- Lectura (R): valor octal 4
- Escritura (W): valor octal 2
- Ejecucion (X): valor octal 1

Estos permisos se aplican a tres tipos de usuarios. U (dueño), G (grupo), O (otros). Para modificarlos se usa `chmod 755 archivo`

▼ Bootloader

Es un programa que permite cargar el SO pudiendo llegar a cargar un entorno previo a la carga del sistema. Generalmente se utilizan los cargadores multietapas, en los que varios programas pequeños se van invocando hasta lograr la carga del SO. El codigo BIOS/UEFI forma parte del bootloader, pero el concepto esta mas orientado al codigo que reside

en el MBR, que esta formado por el MBC y la tabla de particiones, aunque solo el MBC es tenido en cuenta.

▼ Proceso de arranque SystemV

1. Se empieza a ejecutar el codigo del BIOS
2. El BIOS ejecuta el POST
3. El BIOS lee el sector de arranque (MBR)
4. Se carga el gestor de arranque (MBC)
5. El bootloader carga el kernel y el initrd (El initrd funciona como un sistema de archivos raíz provisional hasta que se monte el sistema de archivos raíz real)
6. Se monta el initrd como filesystem raiz y se inicializan componentes esenciales
7. El kernel ejecuta el proceso init y desmonta el initrd
8. Se lee el archivo /etc/inittab
9. Se ejecutan los scripts correspondientes al runlevel 1
10. El sistema cambia al runlevel por defecto
11. Se ejecutan los scripts del runleven por defecto
12. El sistema esta listo para usar

▼ Proceso Init

Es ejecutado directamente por el kernel. Su funcion es cargar todos los subprocesos necesarios para el correcto funcionamiento del SO. Posee el PID 1 (proceso padre del sistema pstree). En SystemV se configura con /etc/inittab. Siendo el encargado de montar los filesystems y de hacer disponible los dispositivos.

▼ Runlevels en SystemV

Son modos operativos predefinidos en los que el sistema puede arrancar o detenerse. Se encuentran definidos en /etc/inittab. Existen 7, y permiten iniciar un conjunto de procesos al arranque o apagado del sistema.

- **0**: Apagar el sistema.
- **1**: Modo monousuario (para mantenimiento).
- **2**: Multiusuario sin servicios de red.
- **3**: Multiusuario con red (modo texto).
- **4**: No utilizado.
- **5**: Multiusuario con entorno gráfico (X11).
- **6**: Reiniciar el sistema.

Los scripts que se ejecutan en cada runlevel están ubicados en **/etc/init.d/** y los enlaces simbólicos en **/etc/rcX.d**.

▼ Proceso de arranque Upstart

Es el primer reemplazo propuesto para SystemV, permite la ejecucion de jobs en forma asincronica a traves de eventos, como principal diferencia con sysVinit que es estrictamente sincronico. Los jobs se configuran en /etc/init y pueden ser:

- Task: ejecucion finita → not respawning
- Service: ejecucion indeterminada → respawning

/etc/inittab no existe mas

▼ Proceso de arranque Systemd

Es un sistema que centraliza la administracion de demonios y librerias del sistema, mejorando el paralelismo de booteo. Es compatible con SysV si es llamado como init. Sino, el demonio systemd reemplaza el proceso init, donde este pasa a tener PID 1. Mientras tanto, los runlevels son reemplazados por targets, que son mas flexibles. Al igual que en Upstart /etc/inittab no existe mas.

▼ Fstab

El archivo **/etc/fstab** define qué particiones del disco se montan automáticamente al inicio del sistema. Cada línea especifica el **sistema de archivos**, el **punto de montaje**, y las **opciones** (como `rw` para lectura-escritura, `ro` para solo lectura).

▼ Redirecciones

Al utilizar redirecciones mediantes > (destructiva), si el archivo no existe lo crea, y si existe se lo trunca y se escribe el nuevo contenido

Si se usa mediante >> (no destructiva), si el archivo de destino no existe se lo crea. y si existe se agrega la informacion al final.

El uso del `|` nos permite comunicar dos procesos por medio de unpipe o tuberia desde la shell. El pipe conecta stdout del primer comando con la stdln del segundo.

Se pueden añadir tantos pipes como se desee

▼ Comandos Varios

▼ Uso de comillas, parentesis, etc

```
# Dobles
mensaje="Hola, mundo"
echo "$mensaje"   # Muestra: Hola, mundo
# Simples
mensaje='Hola, mundo'
echo '$mensaje'   # Muestra: $mensaje (no evalúa la variable)
# Llaves
nombre="Juan"
echo "Hola, ${nombre}!"   # Muestra: Hola, Juan!
# Sin llaves tambien se puede concatenar
echo "Hola, $nombre!"  # Para evitar errores usar {}

# En arrays
echo "${frutas[0]}"  # Muestra el contenido en 0 y 2
echo "${frutas[2]}"
echo "${frutas[@]}" o echo "${frutas[*]}" # Muestra todo el contenido
echo "${#frutas[@]}"   # Muestra el numero de elementos
```

▼ Usuarios

- `useradd` o `adduser`: Crea un nuevo usuario en el sistema.

    - Parámetros comunes:
        - `m`: Crea el directorio home del usuario.
        - `d`: Especifica el directorio home.
        - `g`: Especifica el grupo primario del usuario.
        - `s`: Define el shell predeterminado para el usuario.
- `usermod`: Modifica las propiedades de un usuario existente.
    - Parámetros comunes:
        - `g`: Cambia el grupo principal del usuario.
        - `G`: Añade el usuario a grupos adicionales.
        - `d`: Cambia el directorio home del usuario.
        - `s`: Cambia el shell predeterminado.
- `userdel -r`: Elimina un usuario y también el directorio home del usuario.
- `groupadd`: crea un nuevo grupo
- `who`: muestra los usuarios conectados al sistema
- `groupdel`: Elimina un grupo del sistema.
- `passwd`: Cambia la contraseña de un usuario.

▼ Filesystem

- `chmod`: Cambia los permisos de archivos o directorios.
- `chown`: Cambia el propietario de un archivo o directorio.
- `chgrp`: Cambia el grupo propietario de un archivo o directorio.

▼ Procesos

- `ps`: Muestra los procesos en ejecución de la sesión actual..
- `ps aux`: Muestra todos los procesos del sistema, incluyendo los de otros usuarios.
- `ps -aux | grep [nombre_del_proceso]` muestra todos los procesos del sistema de todos los usuarios y filtra por el nombre

- `kill PID` termina un proceso
- `kill -9 PID` lo fuerza a cerrar
- `killall nombre` termina todos los procesos que se llamen nombre

▼ Busqueda y listar

- `ls -l` : muestra los archivos en forma de lista detallada. `ls -a` incluye los ocultos. se puede usar `ls -la o ls -al` para combinar. `ls -lh` muestra una lista con los tamaños de los archivos.
- **`pwd`**: Muestra el directorio actual
- `tar -cvf archivos.tar archivo1 archivo2 archivo3 archivo4`: Empaquetar archivos. Comprimir con `gzip archivos.tar` y descomprimir con `gzip -d archivos.tar.gz`
- `tar -cvzf misLogs.tar.gz -C /tmp logs` empaqueta y comprime un directorio /tmp/logs
- `grep "texto" archivo.txt`: Busca patrones dentro de archivos.
  - `grep -q "patron"` : busca patrones en arreglos junto con `| echo "$nombre"`
  - `grep -c "texto" archivo`: Cuenta las ocurrencias de "texto" en el archivo y muestra solo el número de coincidencias.
  - `grep -l "texto" *`: Muestra los nombres de los archivos en el directorio actual que contienen dentro "texto".
  - `grep -w "texto" archivo`: Busca solo coincidencias completas de la palabra "texto".
  - `grep name /dir/*` : buscar todos los archivos que contengan name en el dir
- `cat archivo` : imprimir el contenido de un archivo
- `cut -d: -f1` : quedarme con la primer columna de un texto separado por ":"
- `wc -1` : contar cantidad de lineas que se leen
- **`find / -name "*.conf"`**: Busca en el directorio raíz `/` todos los archivos cuyo nombre termine en **.conf**

▼ Directorios y disco

- `mount`: Monta un sistema de archivos
- `umount`: Desmonta un sistema de archivos
- `du`: Muestra el uso del espacio en disco
- `df`: Muestra información del uso de los sistemas de archivos
- `mkdir`: Crea un nuevo directorio
- `rmdir`: Elimina un directorio vacío
- `shutdown`: apagar de forma segura en root

▼ Operaciones con archivo

- `cp`: Copia archivos o directorios
  - `cp /carpeta/* /home`: Copia todos los archivos y directorios desde el directorio actual a /home
- `mv`: Mueve o renombra archivos o directorios
- `rm archivo`: Borra el archivo
- `wc archivo`: Cuenta palabras, líneas y bytes, en ese orden

▼ Estructuras de control

```
if [ condición ]; then
  # comandos si la condición es verdadera
elif [ otra_condición ]; then
  # comandos si la otra condición es verdadera
else
  # comandos si ninguna condición es verdadera
fi
if [ $edad -ge 18 ]; then
  echo "Eres mayor de edad."
else
  echo "Eres menor de edad."
fi
```

```
case $variable in
  valor1)
    # comandos si variable es valor1
    ;;
  valor2)
    # comandos si variable es valor2
    ;;
  *)
    # comandos para cualquier otro valor
    ;;
esac
read -p "Introduce una letra: " letra
case $letra in
  a) echo "Es una vocal." ;;
  e) echo "Es una vocal." ;;
  i) echo "Es una vocal." ;;
  o) echo "Es una vocal." ;;
  u) echo "Es una vocal." ;;
  *) echo "Es una consonante." ;;
esac
```

```
for variable in lista; do
  # comandos que usan la variable
done
for nombre in Ana Luis Carla; do
  echo "Hola $nombre"
done
```

```
while [ condición ]; do
  # comandos mientras la condición sea verdadera
done
contador=0
while [ $contador -lt 5 ]; do
  echo "Contador: $contador"
```

```
  contador=$((contador + 1))
done
```

```
until [ condición ]; do
  # comandos hasta que la condición sea verdadera
done
contador=0
until [ $contador -ge 5 ]; do
  echo "Contador: $contador"
  contador=$((contador + 1))
done
```

```
nombre_de_funcion() {
  # comandos
}
saludar() {
  echo "Hola, $1!"
}

saludar "Mundo"
```

```
echo "Hola Mundo"
------------------
read -p "Introduce tu nombre: " nombre
----------------------------------------
if [ $edad -ge 18 ]; then
  echo "Mayor de edad"
fi
---------------------------
exit 0
-------
for i in {1..10}; do
   if [ $i -eq 5 ]; then
      break
```

```
    fi
    echo $i
done
------------------------
for i in {1..5}; do
    if [ $i -eq 3 ]; then
        continue
    fi
    echo $i
done
```

▼ Operadores

- `eq`: igual a.
- `ne`: no igual a.
- `lt`: menor que.
- `le`: menor o igual que.
- `gt`: mayor que.
- `ge`: mayor o igual que.
- **`e archivo`**: Verdadero si el archivo existe.
- **`d archivo`**: Verdadero si el archivo es un directorio.
- **`f archivo`**: Verdadero si el archivo es un archivo regular.
- **`r archivo`**: Verdadero si el archivo es legible.
- **`w archivo`**: Verdadero si el archivo es escribible.
- **`x archivo`**: Verdadero si el archivo es ejecutable.

Para Cadenas:

- `==`: igual a.
- `!=`: no igual a.
- `-z`: cadena vacía.
- `-n`: cadena no vacía.

▼ Varios

```
echo "Texto" > archivo.txt  # Sobrescribe archivo.txt
echo "Texto adicional" >> archivo.txt  # Agrega al archivo.txt
sort < archivo.txt # Salida estandar
comando 2> errores.txt # Redirige los errores a un archivo
comando &> todo.txt # Redirige tanto la salida como los errores
```

```
#!/bin/bash

echo "Número de argumentos: $#"
echo "Todos los argumentos: $*"
echo "Código de salida del último comando: $?"
echo "Directorio home del usuario: $HOME"
./program.sh arg1 arg2
```

Resumen de las salidas:

Comando	Salida
`find`	Lista de rutas completas de archivos que coinciden.
`locate`	Lista rápida de archivos, si están en la base de datos.
`ls` + `grep`	Nombres de archivos filtrados por el patrón.
`find` + `exec`	Lista de archivos más la acción ejecutada (por ejemplo, `cat`).
`whereis`	Ruta de los binarios y archivos asociados a comandos.
`which`	Ruta del ejecutable de un comando.
`tree` + `grep`	Visualización en forma de árbol filtrada por nombre.