

# Introducción a la Administración de Sistemas Unix/Linux

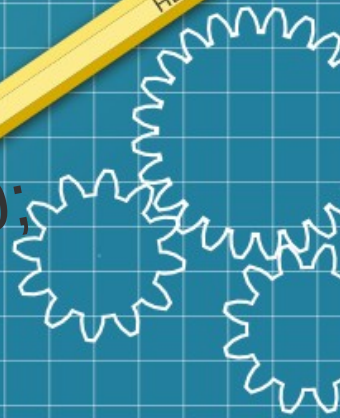
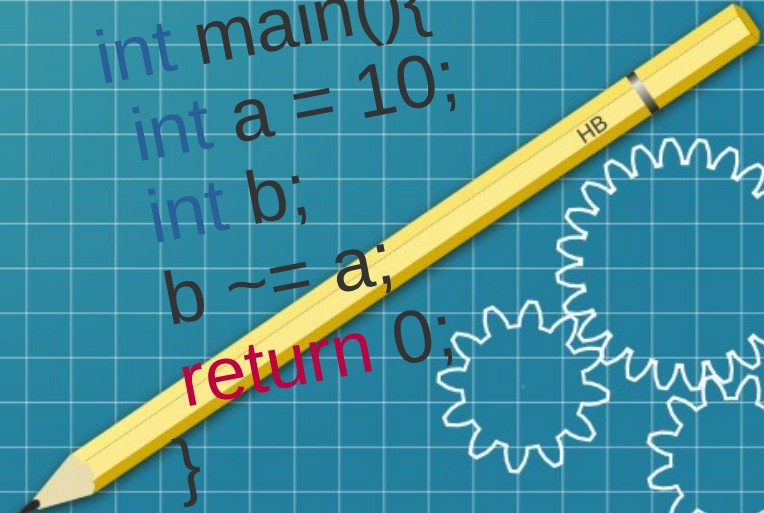
- Grupos y usuarios
- Variables de entorno
- Permisos
- Script



Palomeque Nestor Levi

```
#include <stdio.h>
```

```
int main(){  
    int a = 10;  
    int b;  
    b ~= a;  
    return 0;  
}
```



# Grupos y usuarios

En sistemas Unix/Linux, cada usuario tiene una identificación única (UID, user identifier) y un grupo principal asociado. Los grupos, identificados también en por un número (GID, group identifier), son colecciones de usuarios que comparten ciertos permisos y privilegios en el sistema.

Para que un usuario pueda acceder al puerto serial debe estar en el grupo correspondiente, ***dialout***.

Ejecutar los siguientes comandos:

- *whoami*
- *groups*

# Operadores de redirección y tuberías

Las tuberías (pipes) y las redirecciones son dos conceptos fundamentales en los sistemas Unix y Linux que permiten manipular la entrada y salida de los programas de manera flexible. Mientras que una tubería conecta la salida de un proceso con la entrada de otro proceso, la redirección controla hacia dónde va la entrada, salida o error de un proceso

- **Tuberías (pipes):**

Las tuberías permiten la comunicación entre dos procesos. Se utilizan para pasar la salida de un programa como entrada a otro programa.

En la línea de comandos, una tubería se representa mediante el símbolo |. El flujo es comando1 | comando2, La salida de comando1 se convierte en la entrada de comando2.

Por ejemplo, para buscar un proceso que se está ejecutando podemos hacer:

**top | grep "nombre\_proceso"**

- **Redirecciones de la salida/entrada estandar:**

Las redirecciones permiten cambiar de dónde proviene la entrada o hacia dónde va la salida de un programa.

En la línea de comandos, se utilizan los operadores <, >, >> para redireccionar la entrada o salida de un programa. Y 2> o 2>> para la salida de error estandar.

- Por ejemplo, para guardar la salida de un programa en un archivo, puedes usar el operador >:

**programa > salida.txt**

- Para agregar la salida de un programa a un archivo existente, puedes usar el operador >>:

**programa >> archivo\_existente.txt**

- Para proporcionar un archivo como entrada a un programa, puedes usar el operador <:

**programa < entrada.txt**

- Por ejemplo, para guardar la salida de error de un programa en un archivo, puedes usar el operador 2>:

- **programa 2> salida\_error.txt**

# Script

Los scripts en Bash son archivos de texto que contienen una secuencia de comandos. Se pueden ejecutar utilizando el intérprete de Bash. El uso práctico suele darse en la automatización de tareas comunes.

A continuación veremos como automatizar la creación de un archivo **main.c** para nuevos proyectos.

```
#!/bin/bash
```

```
# Se crea una variable con el primer argumento como nombre de la carpeta, o con un nombre por defecto
```

```
#Para ello se utiliza la expansión de parámetros ${parametro:-valor_predeterminado}
```

```
nombre_carpeta="${1:-mi_proyecto}"
```

```
# Crear la carpeta con el nombre proporcionado
```

```
mkdir "$nombre_carpeta" || exit
```

```
# Cambiar al directorio de la carpeta. El operador || se utiliza para manejar errores. Si el comando a la izquierda del || falla (si no existe el directorio), se ejecuta el comando a la derecha (en este caso, exit).
```

```
cd "$nombre_carpeta" || exit
```

```
# Crear y editar el archivo 'main.c'. Este bloque de código utiliza la funcionalidad de redireccionamiento aquí-documento (<<EOF), que permite pasar múltiples líneas de texto al comando anterior.
```

```
cat > main.c <<EOF
```

```
#include <stdio.h>
```

```
int main() {  
    printf("¡Hola, mundo!\n");  
    return 0;  
}
```

```
EOF
```

```
# Informar al usuario que el script ha terminado
```

```
echo "Se ha creado la carpeta '$nombre_carpeta' "
```

# Ejecución del script

Para ejecutar el script debemos emplear alguno de estos comandos:

a) **bash mi\_script.sh**

b) **./mi\_script.sh**

‘./’ se refiere al directorio actual en el sistema de archivos.

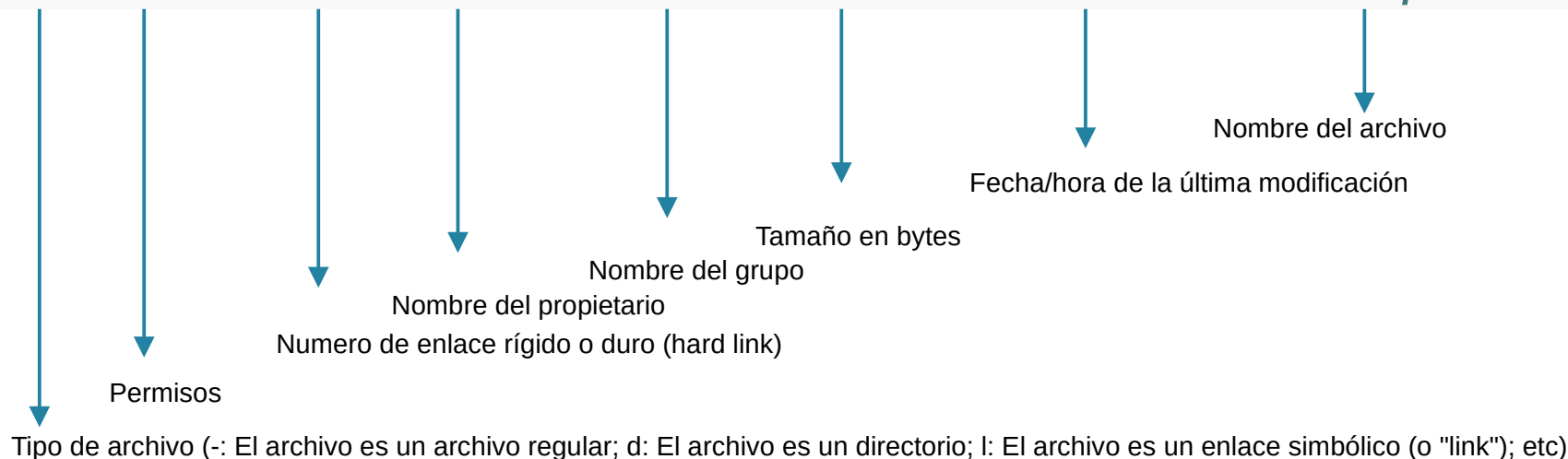
Cuando escribes un comando en la terminal sin una ruta completa (por ejemplo, **mi\_script.sh** en lugar de **./mi\_script.sh**), el sistema busca ese comando en los directorios listados en la variable de entorno PATH. Sin embargo, si el comando no está en ninguno de esos directorios, el sistema no lo encontrará y mostrará un mensaje de error como "command not found".



# Atributos de archivos

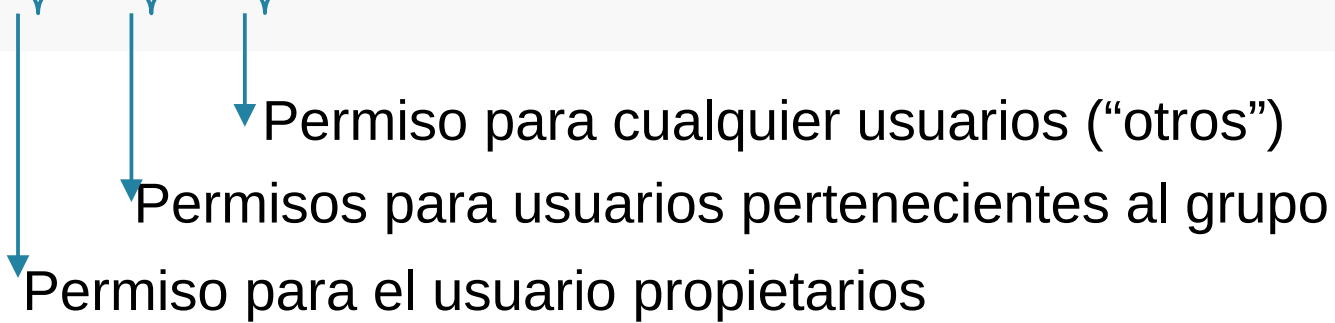
El comando `ls -l` se utiliza para listar el contenido de un directorio en formato detallado en sistemas Unix y Linux. Proporciona una salida que incluye información sobre cada archivo y directorio en el directorio especificado, incluyendo permisos, número de enlaces, propietario, grupo, tamaño, fecha de modificación y nombre del archivo.

```
-rw-rw-r-- 1 kubuntu kubuntu 555 abr 23 13:34 new_p.sh
```



# Permisos

- ***rw-rw-r--*** 1 *kubuntu kubuntu 555 abr 23 13:34 new\_p.sh*



- **r**: Permiso de lectura.
- **w**: Permiso de escritura.
- **x**: Permiso de ejecución.
- **-**: Indica la ausencia de permiso.

# Gestión de permisos

Comandos para cambiar permisos: Los comandos **chmod**, **chown** y **chgrp** se utilizan para cambiar los permisos, el propietario y el grupo de un archivo o directorio, respectivamente.

- 1) Para modificar los permisos de un archivo o directorio, utiliza el comando **chmod**:

El formato general es: **chmod opciones\_permisos archivo\_o\_directorio**. Para asignar los permisos se puede utilizar la notación Octal o la Simbólica.

- **Notación Octal:** Utiliza números para representar los permisos, donde r=4, w=2 y x=1. Se suman estos valores para obtener el valor total del permiso. Por ejemplo, **chmod 755 archivo** establece permisos de lectura, escritura y ejecución para el propietario del archivo (7), y permisos de solo lectura y ejecución para el grupo y otros (5).
- **Notación Simbólica:** Utiliza letras (u, g, o, a para usuario, grupo, otros y todos, respectivamente) junto con operadores (+, -, =) para cambiar los permisos. Por ejemplo, **chmod u+x archivo** agrega el permiso de ejecución para el propietario del archivo.

- 2) El comando **chown** se utiliza para cambiar el propietario de un archivo o directorio.

El formato general es: **chown nuevo\_propietario archivo\_o\_directorio**.

Ejemplo: Para asignar el propietario **usuario1** al archivo **documento.txt**, puedes ejecutar: **chown usuario1 documento.txt**

- 3) El comando **chgrp** cambia el grupo al que pertenece un archivo o directorio.

El formato general es: **chgrp nuevo\_grupo archivo\_o\_directorio**.

Ejemplo: Para asignar el grupo **grupo1** al archivo **archivo.txt**, puedes ejecutar: **chgrp grupo1 archivo.txt**



# Ejemplos de modificación de permisos

*Otros ejemplos:*

*Quitar el permiso de escritura a los miembros del grupo del archivo:*  
***chmod g-w archivo***

*Da permisos de lectura y ejecución a otros usuarios:*  
***chmod o+rx archivo***

*Para dar permisos de lectura y escritura a todos los archivos y subdirectorios dentro de un directorio:*  
***chmod -R u+rwX directorio***

# Variables de entorno

Las variables de entorno son elementos fundamentales en los sistemas operativos Unix y Unix-like, incluyendo Linux. Son variables de configuración que se pueden establecer a nivel del sistema o del usuario y que están disponibles para todos los procesos en el sistema.

- Definición:

Una variable de entorno es una variable que contiene datos específicos del sistema, como directorios de búsqueda, configuración regional, preferencias de usuario, etc.

Proporcionan un método para que los programas obtengan información sobre el entorno en el que se están ejecutando.

- Alcance:

Las variables de entorno pueden ser globales (a nivel del sistema) o locales (a nivel del usuario).

Las variables globales se aplican a todos los usuarios y procesos en el sistema. Las variables locales se aplican solo al usuario que las definió y tienen prioridad sobre las variables globales.

- Algunas variables de entorno:

- PATH: Una de las variables de entorno más comunes. Define los directorios en los que el sistema buscará ejecutables cuando se emite un comando en la terminal.
- HOME: Especifica el directorio de inicio del usuario.
- LANG, LC\_\*: Configuran la configuración regional y de idioma del sistema.
- SHELL: Define la shell predeterminada para el usuario.
- DISPLAY: Especifica el servidor de visualización X para aplicaciones gráficas.
- LD\_LIBRARY\_PATH: Define los directorios donde se buscarán las bibliotecas compartidas durante la ejecución de programas.

# Agregando el script a PATH

Para ejecutar un script desde cualquier directorio en Unix o Linux, puedes agregar la ubicación del directorio que contiene el script al PATH del sistema. De esta manera, el sistema podrá encontrar y ejecutar el script desde cualquier ubicación.

Aquí hay dos enfoques para lograr esto:

- Agregar el directorio actual al PATH temporalmente (Cuando cierres la sesión o reinicies el sistema, el PATH volverá a su valor original):

Puedes agregar el directorio actual al PATH temporalmente utilizando el siguiente comando en tu terminal:

```
export PATH="$PATH:$(pwd)"
```

- Agregar el directorio a PATH de forma permanente:

Para que el directorio esté disponible permanentemente en el PATH, debes agregarlo a tu archivo de configuración de inicio de sesión, como .bashrc, .bash\_profile o .profile, dependiendo de tu sistema operativo y configuración. Por ejemplo, puedes editar el archivo .bashrc en tu directorio de inicio y agregar la siguiente línea al final del archivo:

```
# Otras configuraciones
```

```
export PATH="$PATH:/ruta/al/directorio"
```

Reemplaza /ruta/al/directorio con la ruta completa del directorio que contiene tu script. Guarda el archivo después de hacer la modificación y luego recarga el archivo de configuración de inicio de sesión para que los cambios surtan efecto sin tener que reiniciar sesión:

```
source ~/.bashrc
```

# Agregando el script a PATH



Para agregar tu script a la variable de entorno PATH, de modo que puedas ejecutarlo desde cualquier ubicación en tu sistema, necesitas seguir estos pasos:

- 1) Crea un directorio para tus scripts:

```
mkdir ~/Documentos/Script
```

- 2) Mueve o crea el script en el directorio anterior.

- 3) Agrega el directorio a tu variable de entorno PATH: Abre el archivo de configuración de inicio de sesión de tu shell (por ejemplo, ~/.bashrc, ~/.bash\_profile, ~/.zshrc, etc.) con un editor de texto.

```
nano ~/.bashrc
```

- 4) Agrega la siguiente línea al archivo de configuración de inicio de sesión: Añade la ruta completa del directorio que contiene tus scripts al PATH:

```
export PATH="$PATH:$HOME/scripts"
```

- 5) Guarda y cierra el archivo de configuración de inicio de sesión.

- 6) Recarga la configuración de tu shell:

```
source ~/.bashrc
```

# Ejecicios

- 1) Modificar el script creado para que ejecute **gedit** para editar el archivo **main.c** creado.
- 2) Crear un script que automatice la compilación y ejecución del archivo **main.c**



