



## Introducción al Linux

---

**Ing Florencia Ferrigno**

Abril 2016

UTN - FRBA

# Que es un sistema operativo?

# Que es un sistema operativo?

- Es un programa que controla la ejecución de otros programas.

# Que es un sistema operativo?

- Es un programa que controla la ejecución de otros programas.
- Actúa como interprete entre el usuario y el hardware. Por lo tanto para el usuario es transparente la arquitectura de la computadora

# Que es un sistema operativo?

- Es un programa que controla la ejecución de otros programas.
- Actúa como interprete entre el usuario y el hardware. Por lo tanto para el usuario es transparente la arquitectura de la computadora
- Es un programa que permite usar los recursos de la máquina de manera eficiente.

# Que es un sistema operativo?

- Es un programa que controla la ejecución de otros programas.
- Actúa como interprete entre el usuario y el hardware. Por lo tanto para el usuario es transparente la arquitectura de la computadora
- Es un programa que permite usar los recursos de la máquina de manera eficiente.

## Que es un sistema operativo?

- Cuenta con un set de programas para facilitar la tarea de quienes desarrollamos aplicaciones. Son programas que se ocupan del acceso a disco y el control de los dispositivos de entrada y salida.
- Ese set de programas se los conoce como **API (Application Programming Interface)**.

## Que es un sistema operativo?

- Cuenta con un set de programas para facilitar la tarea de quienes desarrollamos aplicaciones. Son programas que se ocupan del acceso a disco y el control de los dispositivos de entrada y salida.
- Ese set de programas se los conoce como **API (Application Programming Interface)**.
- La implementación de las API dentro del sistema operativo, es decir la forma en que resuelve la llamada efectuada desde un programa de aplicación, se conoce como **System Calls**.



- UNIX es un sistema operativo desarrollado por el Bell Labs (EEUU)
- Uno de sus creadores fue Dennis Ritchie quien también desarrolló el lenguaje C
- UNIX fue el primer sistema operativo desarrollado en C. Hasta entonces se creía que algo tan complejo como un sistema operativo debía ser desarrollado en lenguaje assembler.
- Actualmente todos los sistemas UNIX (y sus derivados, entre ellos Linux) están desarrollados en C
- La arquitectura UNIX es quien denomina al sistema operativo como **kernel** para enfatizar la diferencia entre las aplicaciones de usuario y el intérprete del hardware.

- A qué llamamos **kernel**? Será el conjunto de programas que administran los recursos de la computadora más la implementación de las system calls.

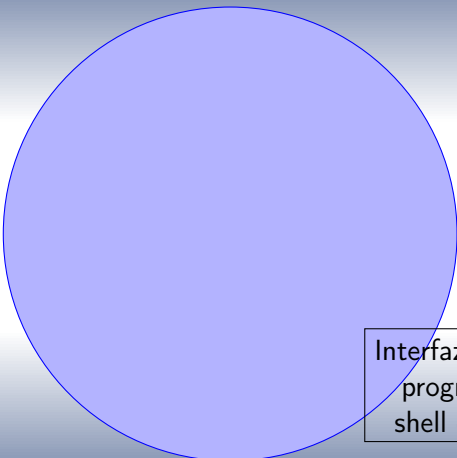
- A qué llamamos **kernel**? Será el conjunto de programas que administran los recursos de la computadora más la implementación de las system calls.
- Que son los **Device Drivers**? Son parte del kernel, su función es acceder al hardware de la máquina de manera directa.

- A qué llamamos **kernel**? Será el conjunto de programas que administran los recursos de la computadora más la implementación de las system calls.
- Que son los **Device Drivers**? Son parte del kernel, su función es acceder al hardware de la máquina de manera directa.

### En resumen:

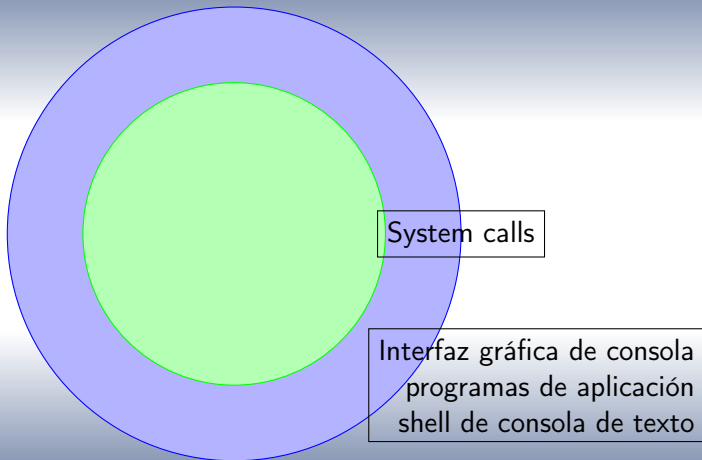
En el mundo **UNIX** cuando se habla de kernel se habla de Sistema Operativo y viceversa.

# Arquitectura

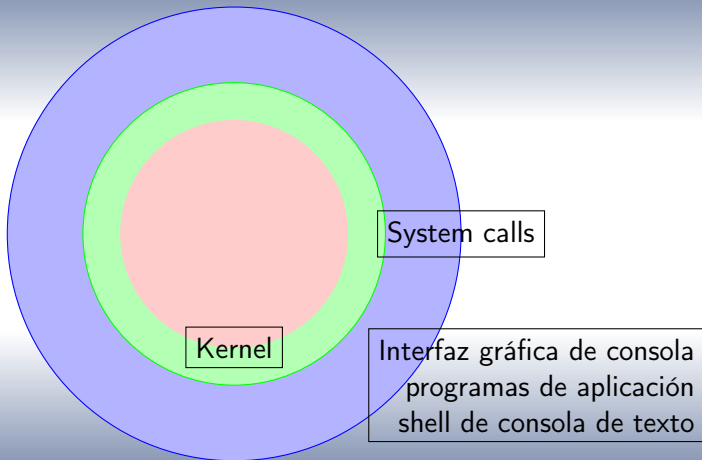
A large, light blue circle with a thin blue outline, positioned on the left side of the slide, partially overlapping the text box.

Interfaz gráfica de consola  
programas de aplicación  
shell de consola de texto

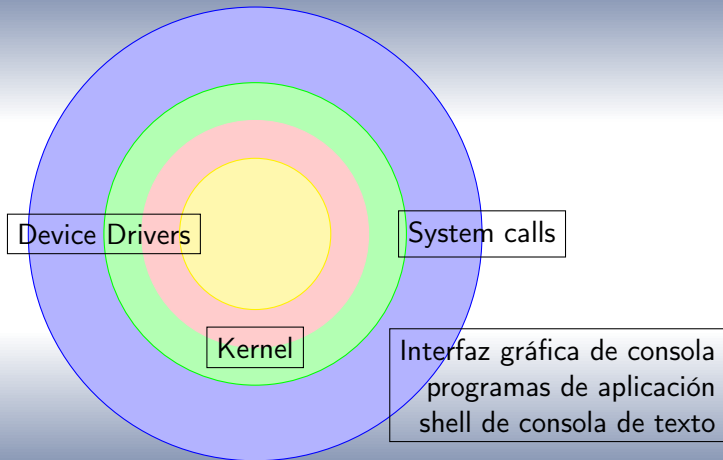
# Arquitectura



# Arquitectura

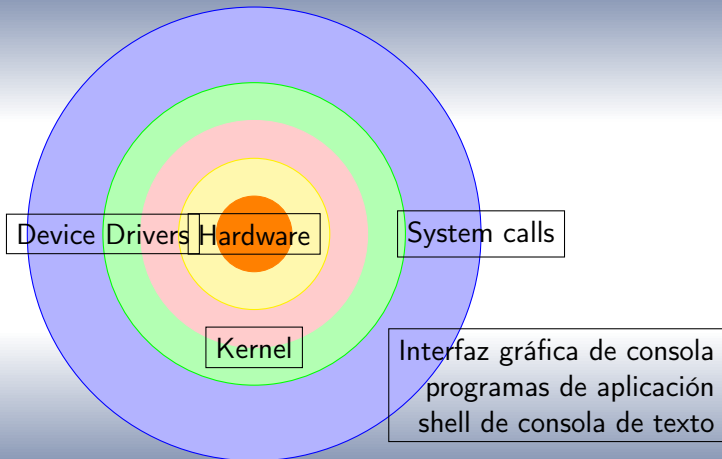


# Arquitectura





# Arquitectura



# Introducción

- Desarrollado por Linus Torvalds en la Universidad de Helsinki como trabajo de tesis final.
- Es un kernel basado en UNIX bajo licencia GPL (General Public Licence).
- Es de **código abierto**. Es decir que aquellos interesados pueden ver cómo está hecho y modificarlo para hacer su propio kernel.
- Cualquier programa escrito y compilado en UNIX corre en Linux.

# Introducción

- Es un sistema multiusuario, es decir que puede existir más de un usuario logueado a la vez.
- Existe un usuario muy potente, que tiene permitido hacer de todo dentro del sistema. **root**.
- El usuario **root**, también conocido como super usuario, es quien se encarga de administrar el sistema.
- El resto de los usuarios, no cuentan con los mismos privilegios. Sin embargo veremos que existen formas para "simular" ser el root.

## Organización jerárquica del disco

- Cada usuario que se cree tendrá asociado un **home directory** donde se nos ubica una vez que nos logueamos
- Cada usuario tendrá los permisos para leer, escribir y ejecutar sobre su home directory, no así sobre los home directory de otros usuarios.
- La ruta será: **/home/nombreDeUsuario** y su forma abreviada es **~**

## Algunas carpetas a tener en cuenta

- **/:** **Root File System:** Es la raíz de toda estructura de directorios.
- **/boot:** **Boot loader.** Son los archivos necesarios para realizar el booteo de la máquina. Los archivos por excelencia serán:
  - **GRUB**, que será quien administra con que sistema operativo iniciará la máquina
  - **initrd.** Es el inicializador general del sistema.
- **/sys:** Aquí se encuentran todos los archivos relacionados con el kernel y el sistema en general.
- **/sbin:** son los binarios esenciales para la administración del sistema.

## Algunas carpetas a tener en cuenta

- **/bin**: binarios para uso del sistema por parte de los usuarios, aquí se encuentran los binarios que se corren cuando ejecutamos comandos desde la consola.
- **/lib**: librerías para todos los binarios contenidos en los directorios **/sbin** y **/bin**
- **/etc**: archivos de configuración del sistema operativo.
- **/home**: todos los directorios de los usuarios están dentro de esta carpeta, a excepción del usuario **root**, cuyo home está en **/root**

# El Shell

- Es una aplicación
- Interfaz de usuario que oficia de intérprete de comandos => Traduce los pedidos del usuario en system calls.
- Cada vez que se escribe un comando en el shell, el mismo se lo interpreta como un ejecutable binario o script.

## Aclaración

- **Ejecutable binario:** surge del resultado de la edición, compilación y linkeo de un programa determinado.
- **Script:** archivo de texto que contiene comandos binarios u otros scripts para ejecutar.

# El Shell. Primer vistazo a la terminal

- Que es ~ ? A este símbolo se lo denomina prompt y nos indica que el shell esta listo para recibir comandos.



## El Shell. Primer vistazo a la terminal

- Que es ~ ? A este símbolo se lo denomina prompt y nos indica que el shell esta listo para recibir comandos.
- Cuando nos logueamos en el terminal, veremos algo similar a esto:  
**nombre\_de\_usuario@nombre\_de\_máquina:~\$**

# El Shell. Primer vistazo a la terminal

- Que es ~ ? A este símbolo se lo denomina prompt y nos indica que el shell esta listo para recibir comandos.
- Cuando nos logueamos en el terminal, veremos algo similar a esto:  
**nombre\_de\_usuario@nombre\_de\_máquina:~\$**
- **nombre\_de\_usuario**: El nombre del usuario logeado en la terminal.

# El Shell. Primer vistazo a la terminal

- Que es ~ ? A este símbolo se lo denomina prompt y nos indica que el shell esta listo para recibir comandos.
- Cuando nos logueamos en el terminal, veremos algo similar a esto:  
**nombre\_de\_usuario@nombre\_de\_máquina:~\$**
- **nombre\_de\_usuario**: El nombre del usuario logueado en la terminal.
- **nombre\_de\_máquina**: El nombre que le hayan puesto a su máquina en el momento de la instalación.

# El Shell. Primer vistazo a la terminal

- Que es ~ ? A este símbolo se lo denomina prompt y nos indica que el shell esta listo para recibir comandos.
- Cuando nos logueamos en el terminal, veremos algo similar a esto:  
**nombre\_de\_usuario@nombre\_de\_máquina:~\$**
- **nombre\_de\_usuario**: El nombre del usuario logueado en la terminal.
- **nombre\_de\_máquina**: El nombre que le hayan puesto a su máquina en el momento de la instalación.
- Luego de los :
- ~: Es la forma abreviada del /home/nombreDeUsuario (home directory). Si cambiamos de directorio, aquí nos mostrará donde estamos trabajando.

# El Shell. Primer vistazo a la terminal

- Que es ~ ? A este símbolo se lo denomina prompt y nos indica que el shell esta listo para recibir comandos.
- Cuando nos logueamos en el terminal, veremos algo similar a esto:  
**nombre\_de\_usuario@nombre\_de\_máquina:~\$**
- **nombre\_de\_usuario**: El nombre del usuario logueado en la terminal.
- **nombre\_de\_máquina**: El nombre que le hayan puesto a su máquina en el momento de la instalación.
- Luego de los :
- ~: Es la forma abreviada del /home/nombreDeUsuario (home directory). Si cambiamos de directorio, aquí nos mostrará donde estamos trabajando.
- \$: indica que el usuario logueado es un usuario "común", si el usuario logueado es root, el prompt cambiará a: #

## Comandos útiles: man

- **man**: Es el comando de ayuda. Cada vez que surja una duda de como usar un comando podemos llamar al **man** para que nos indique su uso, de la siguiente manera
- **man nombre\_del\_comando**, por ejemplo: **man gcc** nos dirá cómo usar el comando para compilar nuestro código

## Comandos útiles: sudo

- Existen acciones que sólo están habilitadas para **root**, por lo tanto si las queremos ejecutar como un usuario común del sistema no nos será posible hacerlo. Por lo tanto vamos a necesitar hacerlo "como si fuéramos" root
- **sudo**: Este comando significa super usuario hace (Super User Do). Este comando nos permitirá ejecutar determinados comandos que sólo pueden ser hechos por root.
- Por ejemplo cada vez que necesitemos instalar "algo" vamos a tener que hacerlo como si fuéramos un super usuario, de la siguiente manera:
- **sudo apt-get install nombre\_del\_paquete**

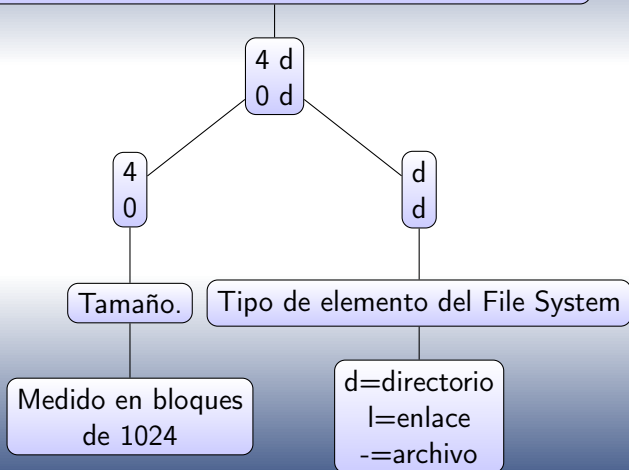
## Comandos útiles: List directory

- Este comando lista el contenido del directorio en el cual estamos "parados" al momento de ejecutarlo.
- Algunas opciones más usadas de este comando son:
  - -l = long listing (listado detallado de cada uno de los archivos contenidos por el directorio)
  - -a = all (lista todos los archivos que estan dentro del directorio, inclusive los archivos ocultos)
  - -s = size (muestra el tamaño de cada uno de los archivos)
  - -t = time (ordena los archivos por orden de fecha y hora, empezando por los mas nuevos)



## Comandos útiles: List directory

```
4 drwxr-xr-x 11 user grupo 4096 Mar19 12:30 .
0 dwxr-xr-x 1 user grupo 816 Mar20 09:00 INFO1
```



## Comandos útiles: List directory

```
4 drwxr-xr-x 11 user grupo 4096 Mar19 12:30 .  
0 dwxr-xr-x 1 user grupo 816 Mar20 09:00 INFO1
```

rwX r-x r-x  
wrX r-x r-x

Permisos del elemento separado en 3 grupos de 3 campos cada uno.  
Los 3 primeros para el dueño del elemento.  
Los 3 segundos para el grupo al que pertenece el dueño.  
Los 3 últimos para el resto de los usuarios

## Comandos útiles: List directory

```
4 drwxr-xr-x 11 user grupo 4096 Mar19 12:30 .  
0 dwxr-xr-x 1 user grupo 816 Mar20 09:00 INFO1
```

11 user grupo  
1 user grupo

Número de enlaces

Nombre del usuario  
propietario del elemento

Grupo al que  
pertenece el usuario

## Comandos útiles: List directory

```
4 drwxr-xr-x 11 user grupo 4096 Mar19 12:30 .  
0 dwxr-xr-x 1 user grupo 816 Mar20 09:00 INFO1
```

```
4096 Mar19 12:30 .  
816 Mar20 09:00 INFO1
```

Tamaño en bytes

Timestamp  
de la última  
modificación

Nombre del elemento

## Archivos:

- En Linux **todo es un archivo**. Es decir que el kernel interpreta todo como si fuera un archivo: documentos, directorios, discos, el teclado, TODO es un archivo.
- El nombre de un archivo es solo eso, sin importar si hay un punto que separa el nombre en dos. Es decir para Linux no existen las extensiones.
- Como hace entonces el sistema para saber de que tipo de archivo se trata?

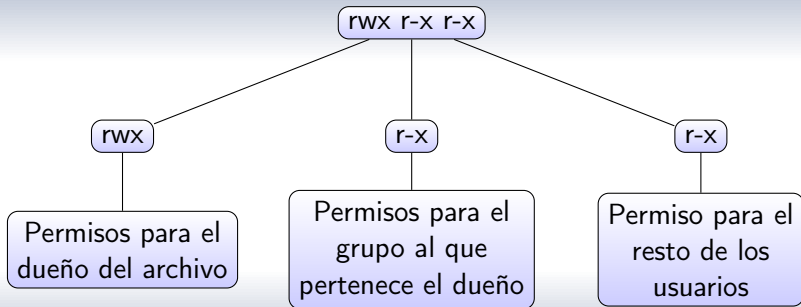
## Archivos:

- En Linux **todo es un archivo**. Es decir que el kernel interpreta todo como si fuera un archivo: documentos, directorios, discos, el teclado, TODO es un archivo.
- El nombre de un archivo es solo eso, sin importar si hay un punto que separa el nombre en dos. Es decir para Linux no existen las extensiones.
- Como hace entonces el sistema para saber de que tipo de archivo se trata?
- Linux maneja los archivos segun permisos (como vimos en los slides anteriores). Es decir si el archivo es un ejecutable es por que tiene permisos de ejecución

## Archivos: Permisos

- Volvamos a la parte de los permisos de un elemento.
- Cada elemento es propiedad de un usuario y esta relacionado con el grupo al que pertenece.
- Para cada elemento existe un conjunto de 9 modificaciones que controlan quien puede leer, escribir o ejecutar dicho elemento.
- Estas tres acciones se identifican con una letra:
  - Lectura: r (read)
  - Escritura: w (write)
  - Ejecución: x (executable)

# Archivos: Permisos





## Archivos: Permisos

- Existe un comando que nos permite cambiar los permisos asignados a un elemento.
- Esta acción sólo la puede ejecutar el dueño del mismo o root.
- Existen archivos sensibles, con los de configuración, los cuales sólo podrán ser modificados por root.
- El comando a ejecutar es **chmod** (Change Mode)

## Archivos: Permisos CHMOD

- Este comando necesita de dos parámetros:
  - Los permisos modificados
  - El nombre del elemento que queremos modificar
- Los permisos se expresan de manera simbólica ya sea de manera numérica o con símbolos.
- Numérica:
  - 4: permiso de lectura
  - 2: permiso de escritura
  - 1: permiso de ejecución
  - 0: ausencia de permisos.
  - Si se desea habilitar permisos de lectura y escritura, debemos sumar los valores que cada uno de ellos representa, es decir:  $4+2=6$
- Supongamos que queremos cambiar los permisos del archivo **HolaMundo.c** de manera tal que solo lo pueda leer y escribir su dueño y nadie más:
  - **chmod 600 HolaMundo.c**

# Archivos: Permisos CHMOD

- Simbólica:
  - Establecer los permisos con símbolos, permiten configurar los permisos para cada una de las situaciones por separado, sin necesidad de conocer los permisos preexistentes.
  - Dichos símbolos son:
    - **u**: propietario o usuario
    - **g**: grupo
    - **o**: otros
    - **r**: lectura
    - **w**: escritura
    - **x**: ejecución

# Archivos: Permisos CHMOD

- Ejemplos de códigos:
  - **ugo+w**: permisos de lectura, escritura y ejecución otorgados al dueño, grupo y demás usuarios.
  - **g+rx**: permisos de lectura y ejecución otorgados al grupo.
  - **ugo-rwx**: permisos de lectura, escritura y ejecución denegados para dueño, grupo y demás usuarios.
  - **o-rw**: permisos de lectura y escritura denegados a los demás usuarios.
- Vamos a hacer el mismo cambio de antes pero de la manera simbólica:
  - **chmod go-rwx HolaMundo.c**

## Otros comandos útiles

- **mkdir**: (Make directory) Para crear un directorio (o carpeta). Ejemplo: **mkdir info1**
- **pwd**: (Print working directory) Muestra el directorio en el cual se esta "parado". Ejemplo: **pwd**
- **cd**: (Change directory) Este comando nos permite cambiar de directorio. Por ejemplo si queremos acceder a la carpeta **info1**: **cd info1**, en caso que querramos subir un nivel desde donde estamos: **cd ..**

## Otros comandos útiles

- **rm:** (Remove) Comando para borrar archivos y directorios. Hay que tener cuidado con este comando, ya que Linux no pregunta si "estamos seguros", simplemente lo borra. Al menos que lo ejecutemos con la opción **-i**, en este caso pedirá confirmación. Para poder borrar un directorio necesitaremos ejecutar este comando con la opción **-r** de recursivo.
- Algunos ejemplos de como ejecutarlos:
  - Borrar un archivo sin confirmación: **rm nombreDelArchivo**
  - Borrar un archivo con confirmación: **rm -i nombreDelArchivo**
  - Borrar un directorio: **rm -r nombreDelDirectorio**