

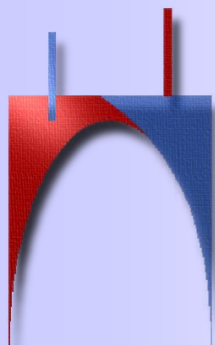
C.F.G.S.

Desarrollo de Aplicaciones Multiplataforma

Desarrollo de Aplicaciones Web

UD 5.1

Sistemas Operativos Multiusuario. Operación



Instituto de Educación Secundaria
Santiago Hernández
Informática

Historia de linux

➤ unix

- 1970: Ken Thomson, Dennis Ritchie y otros en los laboratorios Bell de AT&T
- Desarrollado como Sistema de Tiempo compartido más simple que multics
- Idea nueva: Soporte para varios Hardware diferentes:
 - Necesario dejar de trabajar en ensamblador
 - Desarrollan un lenguaje de programación específico y los compiladores adecuados a cada hardware
 - Lenguaje de programación: B
 - Lenguaje Mejorado por Dennis Ritchie: C

➤ Problemas de Nacimiento

- Al no usar ensamblador no se aprovechan exactamente los recursos de la máquina
- Las máquinas necesitan más recursos (Procesadores más potentes y memoria): Muy Caro para la época
- Años 90: Bajada de precios → dominio de UNIX

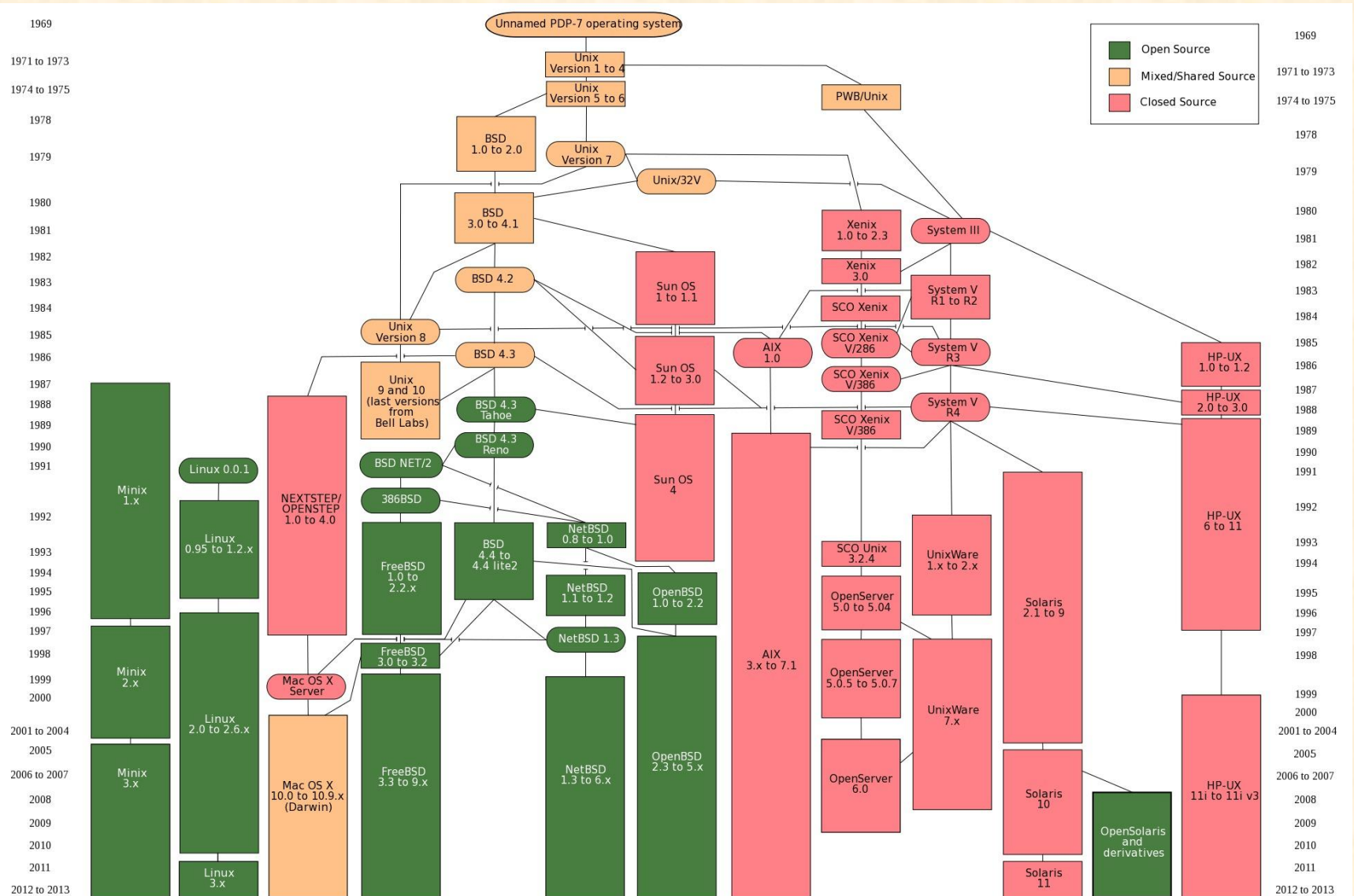
➤ Diversidad

- Sistema Abierto: Puede correr en cualquier máquina
- Varias empresas desarrollan su propio UNIX (HP-UX, AIX, Solaris, Xenix, OpenServer, Linux)
- Distribuciones libres: GNU
- “Pequeñas” diferencias de manejo entre marcas
- Serias diferencias en funcionamiento interno
- Grandes diferencias en administración

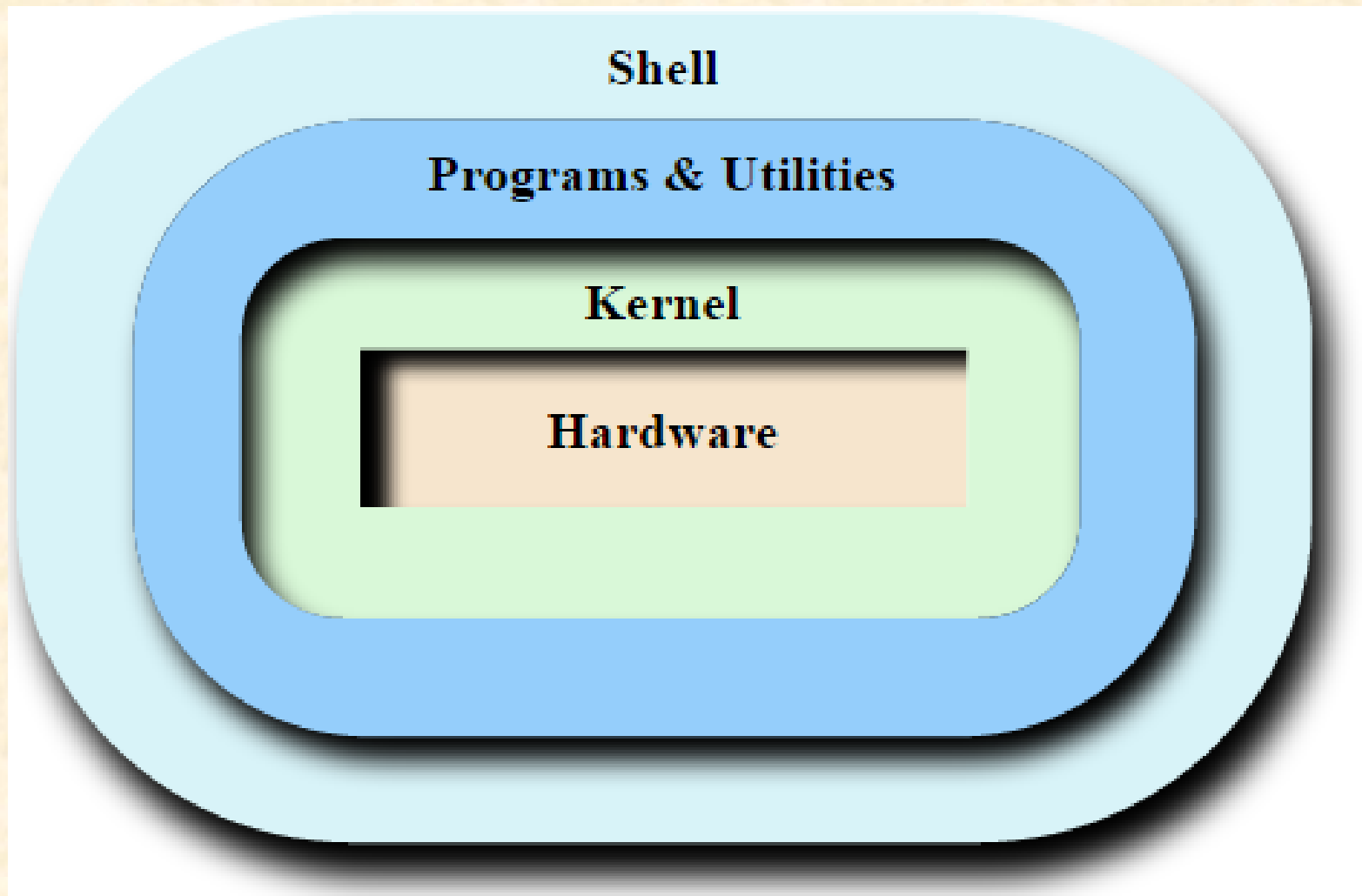
➤ linux

- 1983: Richar Stallman: Proyecto GNU
 - Desarrolló parte del software (librerías, compiladores,...).
 - La parte de bajo nivel (Drivers, kernel) no fueron terminadas.
- 1991: Linus Torvalds: linux
 - Inicio: Solo desarrollo de un kernel.
 - Desarrollado en minix utilizando el compilador C de GNU.
 - Distribuido bajo licencia libre GNU.
- GNU/linux
 - Kernel linux con aplicaciones del proyecto GNU.
 - Se suele hacer referencia a esto llamándolo, simplemente, linux.
- Problemas legales
 - Siempre en litigios con los poseedores de la licencia de UNIX

UNIX Timeline

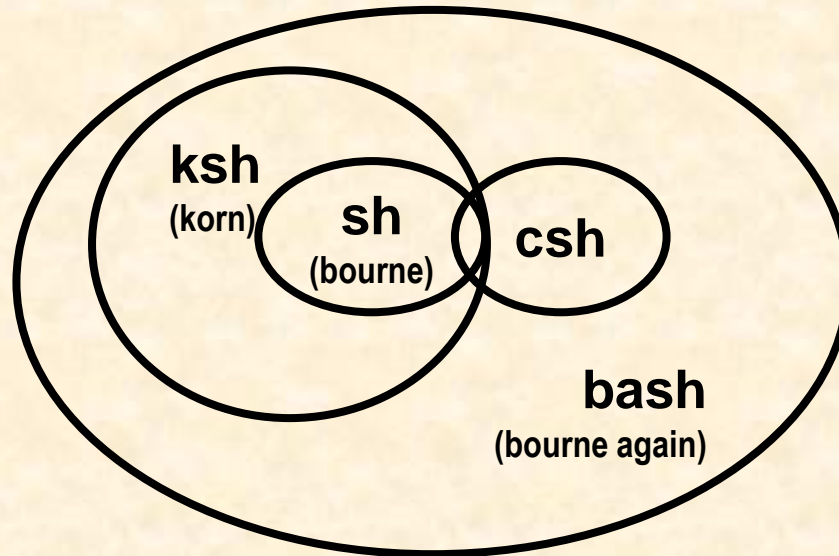


Arquitectura



Shell

➤ Tipos



➤ Prompt

- Indicador del shell
- Originales:
 - \$: usuarios normales
 - #: root (superusuario)
- Actuales: Múltiples variaciones y configurable (nombre de máquina, de usuario, de terminal, directorio actual, colores, ...)

Los escritorios gráficos

➤ El servidor gráfico

- X Window

➤ Entornos de Escritorio (Desktop Environment)

- Pesados
 - gnome
 - kde plasma
- ligeros
 - lxde
 - xfce
- Otros y comparación
 - https://wiki.archlinux.org/index.php/Desktop_environment

➤ Gestores de escritorio (Display Manager)

- Permiten realizar el login y elegir el Entorno de escritorio
 - https://wiki.archlinux.org/index.php/Display_manager

Conceptos importantes

➤ Usuarios y grupos

- Base de la seguridad
- root (superusuario): no le afecta la seguridad
- Usuarios y grupos especiales
- Usuarios “normales”

➤ Entorno

- Variables del shell
- \$NOMBRE; \${NOMBRE}: valor
- Algunas variables:
- HOME, LOGNAME, TERM
- PS1
- Mantenimiento del entorno: creación de nuevos shell
- Variables exportables
- Alias
- Entrada, salida y error estándar (stdin, stdout, stderr) redireccionables

➤ “Estilo” UNIX

- Si no hay noticias: Buenas noticias
 - Menos datos a transmitir
 - Muchos comandos no dicen nada si lo hacen bien
 - Si dicen algo suelen ser errores
- Construcción de comandos como bloques de edificios: pipelines
- Filtros en lugar de complicar los comandos
- Pulsaciones mínimas (menos datos a transmitir):
 - Comandos cortos
 - Opciones cortas (la mayoría de una letra)

Conexión y login

➤ Conexión local

- Terminal gráfico
- Terminales de texto

➤ Herramientas de conexión remota

- telnet
- ssh
- putty

➤ Login: Autenticación

- Usuario y contraseña (sin asteriscos)
- Mayúsculas y minúsculas
- Usuario validado en terminal de texto: prompt del shell

➤ Logout: Salida

\$ **exit**

cerrar el shell

- Al cerrar el último shell se sale al login (conexión local) o se cierra la conexión (conexión remota)



Primeros Comandos

Identificando el sistema

➤ Identificación de usuario y máquina

```
$ logname  
$ id  
$ uname [-a]  
$ who
```

➤ fecha y hora

```
$ date (como usuario normal y como root)  
$ cal
```

➤ directorio

```
$ pwd  
$ cd  
$ ls [-l a ]
```

➤ variables

```
$ set  
$ alias  
$ export
```

Comandos básicos

➤ manejo de la pantalla

\$ clear

- filtros paginadores (more, less, pg)

➤ mostrar información

\$ echo

\$ cat

➤ introducción de comandos (bash)

- Revisión del historial (arriba, abajo)
- Tecla de tabulación

➤ Documentación

- *man comando*
- *comando --help*



El árbol de directorios

Nombres

➤ El directorio Raiz

➤ Nombres completos

- / como inicio del nombre
- / como separador

➤ Nombres relativos

- “.” Identificador de Directorio Actual
- “..” Identificador de Directorio Padre

➤ Nombrando ficheros

- Caracteres: letras, números y caracteres especiales (limitaremos a “.” y “-”)
- Diferencia ente mayúsculas y minúsculas
- Si empieza por “.” → Oculto

Comando ls

- **Muestra el contenido de un directorio**
- **Sintaxis:**
 - \$ ls [opciones] [fichero | directorio ...]
- **Ayuda:**
 - \$ ls --help
 - \$ man ls
- **Opciones más importantes**
 - -l: listado largo
 - -a: todos los ficheros y directorios, incluidos ocultos
 - -R: Recursivo (incluye subdirectorios)
 - -d: muestra el directorio en lugar de su contenido
 - -h: muestra las cantidades de forma legible por el ser humano

Metacaracteres

➤ Para generar listas de ficheros o directorios

- Expande todos los nombres del directorio actual o el indicado en la ruta

➤ Significado

- ?: un carácter cualquiera
- *: una cadena cualquiera de caracteres (incluida la cadena vacía)
- [lista]: un carácter cualquiera de los de la lista. Admite rangos
- [!lista]: un carácter cualquiera menos los de la lista

➤ Ejemplos

- f???: incluye “file” “f000” “f.sh” pero no “f1” ni “f1234”.
- f*: todos los nombres que empiecen por “f”. Incluye el nombre “f”.
- *s: todos los nombres que acaben en s. Incluye el nombre “s”. No incluye los ocultos
- *: todos los nombres. No incluye los ocultos
- .*: todos los ocultos
- file[123]: incluye file1, file2 y file3
- file[!123]: incluye file0, files, file4, ... pero no file, file1, file2 ni file3

Comandos de manejo de ficheros

➤ Nota

- En todos los comandos en que puede utilizarse una lista de nombres esta lista puede ser generada por medio de metacaracteres

\$ **touch fichero [fichero2 fichero3 ...]**

Si el fichero no existe lo crea en blanco

Si el fichero modifica la fecha de ultima actualización

\$ **stat fichero [fichero2 fichero3 ...]**

muestra información sobre los ficheros de la lista

también pueden especificarse directorios

\$ **cat fichero [fichero2 fichero3 ...]**

Muestra el contenido de los ficheros de la lista

uso especial:

\$ cat > fichero

para crear un fichero con contenido. Se finaliza con CTRL-D

\$ **rm fichero [fichero2 fichero3 ...]**

Borra los ficheros de la lista

Comandos de manejo de directorios

\$ **pwd**

muestra el directorio de trabajo (actual)

\$ **cd [directorio]**

moverse al directorio especificado.

Si no se escribe ninguno se mueve al directorio almacenado por la variable \$HOME

\$ **mkdir [-p] directorio [directorio2 directorio3 ...]**

Crea el directorio o directorios de la lista.

El directorio padre del que se quiere crear debe existir.

Si no existe puede crearse la rama completa con la opción -p

\$ **rmdir [-p] directorio [directorio2 directorio 3 ...]**

Borra el directorio o directorios de la lista

El directorio a borrar debe encontrarse vacío

la opción -p permite borrar ramas vacías

p.e. rmdir -p d1/d11 borrará d11 y si d1 queda vacío también lo borrará

\$ **rm -r directorio [directorio2 directorio 3 ...]**

Borra el directorio o directorios de la lista

Comandos de manejo de ficheros (2)

\$ **mv fichero_existente nuevo_nombre**

Cambia el nombre de fichero_existente
nuevo_nombre no debe existir

\$ **mv fichero_existente [fich_existente2 ...] directorio_destino**

mueve los ficheros de la lista al directorio_destino
directorio_destino debe existir

\$ **cp [-p] fichero_existente nuevo_fichero**

copia el contenido de fichero_existente creando un nuevo_fichero
-p: mantiene todas las propiedades del fichero original (propietarios, fechas, tipo, ...)
solo para root

\$ **cp [-p] fichero_existente [fich_existente2 ...] directorio_destino**

copia el contenido de los ficheros de la lista en ficheros de nueva creación bajo directorio_destino
los ficheros de nueva creación mantendrán el mismo nombre (terminal)

Comandos de manejo de directorios (2)

\$ **mv directorio_existente nuevo_nombre**

Cambia el nombre de directorio_existente
nuevo_nombre no debe existir

\$ **mv directorio_existente [dir_existente2 ...] directorio_destino**

mueve los directorios de la lista y todo su contenido al directorio_destino
directorio_destino debe existir

\$ **cp -R directorio_existente nuevo_directorio**

copia el contenido de directorio_existente creando nuevo_directorio
nuevo_directorio no debe existir
-a: equivalente a -Rp

\$ **cp -R directorio_existente [dir_existente2 ...] directorio_destino**

copia los directorios de la lista y todo su contenido bajo directorio_destino
directorio_destino debe existir

Enlaces

➤ Enlace duro

- Dos o más ficheros que apuntan a la misma zona de datos
- Con `ls -l` se ve el número de enlaces duros
- Con `ls -li` se ve el número de ínodo (zona a la que apuntan)
- La información solo se borra cuando se borra el último enlace duro

➤ Enlace Simbólico

- Nombre que apunta al nombre de un fichero o directorio
- Con `ls -l` se ve el nombre al que apunta
- Si se borra o renombra el fichero o directorio el enlace queda huérfano
- Dependiendo si el enlace apunta a un nombre completo o relativo el enlace puede quedar huérfano al mover alguno de los elementos implicados

➤ Comandos

\$ `ln fichero_existente nuevo_fichero`

Crea un nuevo fichero que es enlace duro del existente

\$ `ln -s fichero_existente | directorio_existente nuevo_enlace`

Crea un enlace simbólico al fichero o directorio existente indicado

➤ Notas sobre los enlaces simbólicos

- Las operaciones que afectan a la información (`cp`, `cat`) actúan sobre el fichero
- Las operaciones que afectan al nombre (`mv`, `rm`) actúan sobre el enlace



vi

El editor vi

➤ **vi vs vim**

- vi: editor de páginas
- vim: vi improved (añade, entre otras cosas, un analizador sintáctico: colores)
- vi es, normalmente, un enlace simbólico a vim

➤ **vi en Ubuntu**

- Se instala vim.tiny (falla el movimiento del cursor)
- Para instalar el vim completo:
 - `sudo apt-get install vim`

➤ **vi en Fedora26**

- No se instala vim, solo vi.
- para instalar el vim completo:
 - `sudo dnf install vim`

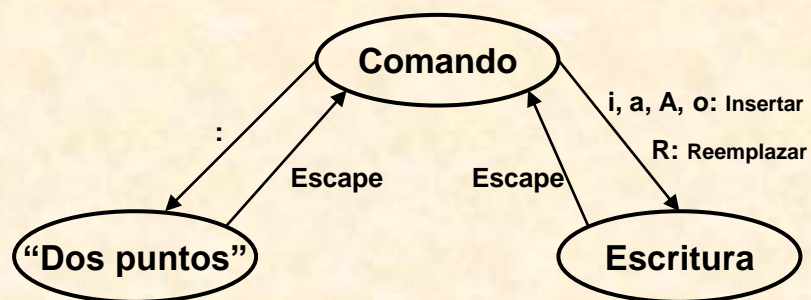
➤ **Invocación**

- `vi`
- `vi nombre`
- si vi y vim son diferentes, para invocar el mejorado:
 - `vim`
 - `vim nombre`

➤ Modos

- Modo comando
- Modo “dos puntos”
- Modos de escritura:
 - Insertar
 - Reemplazar

➤ Transición entre modos

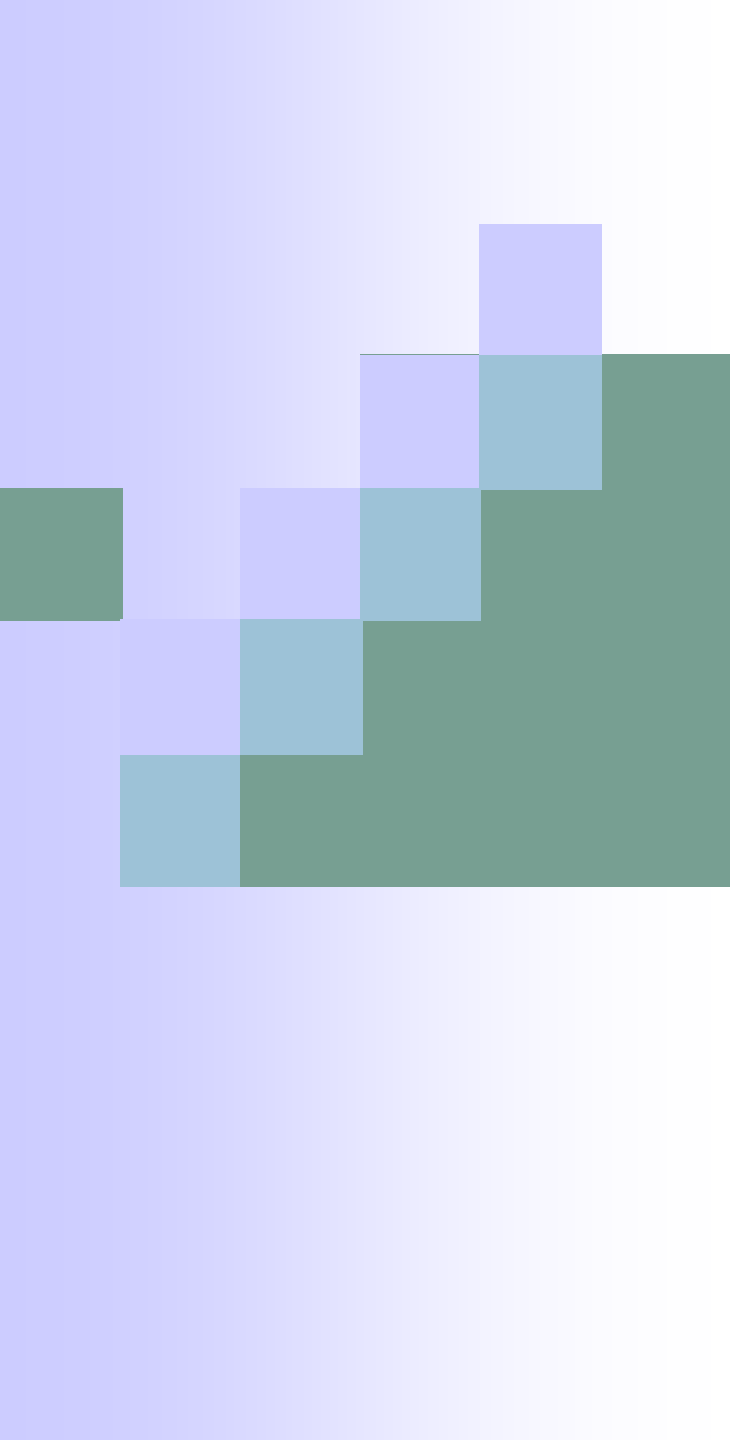


➤ Comandos

- Movimiento del cursor
 - h, j, k, l (o flechas de movimiento del cursor)
 - AvPag, Repag
 - nG: ir a línea *n*
- Entrada en modo de escritura
 - i: insertar en la posición del cursor
 - a: añadir después del cursor
 - A: Añadir al final de la línea
 - o: añadir línea debajo del cursor
 - R: Reemplazar a partir de la posición del cursor
- Borrado
 - x: Borra el carácter del cursor
 - dd: borra la línea actual
- Miscelánea
 - r: reemplazar carácter en el cursor
 - u: deshacer última operación realizada
 - U: deshacer todas las operaciones realizadas en la línea actual
 - J: unir la línea siguiente con la actual
- Nota
 - Repetición de comandos al escribir un número antes del comando (ojo, el número no se ve)

➤ Comandos en modo “dos puntos”

- Grabado
 - w: graba el fichero actual
 - w *nombre*: crea un nuevo fichero
- Salida
 - q: sale sin grabar (solo si el fichero no ha sido modificado)
 - q!: sale sin grabar
 - wq: graba y sale
 - x: graba y sale
- Búsqueda
 - / *cadena*: busca *cadena*
 - /: Repite búsqueda
- Miscelánea
 - help: entra en la ayuda
 - set all: muestra variables
 - set *variable*: activa variable (p.e. set number)
 - set *novariable*: desactiva variable (p.e. set nonumber)



Seguridad de ficheros y directorios: **Permisos**

Permisos

➤ Grupos de permisos

- Se ven al ejecutar `ls -l`
- Tres bloques de tres letras
- Primer bloque se aplica al usuario propietario
 - Si No: Segundo Bloque se aplica al grupo propietario
 - Si No: se aplica el Tercer Bloque

➤ Significado de cada bloque

- Siempre mismo orden: rwx (lectura, escritura, ejecución)
- letra: indica que hay permiso
- -: indica que no hay permiso
- La tercera letra puede contener otros caracteres (ver permisos especiales más adelante)

➤ Los permisos y root

- Los permisos no afectan a root, este siempre tiene acceso total
- Una excepción es el permiso de ejecución, para que root pueda ejecutar un fichero este tiene que tener, al menos, un permiso de ejecución activado

Permisos de ficheros

➤ **r: Lectura**

- Se permite examinar el contenido

➤ **w: Escritura**

- Se permite modificar el contenido

➤ **x: Ejecución**

- Se puede ejecutar el fichero al escribir su nombre en el prompt del sistema
- Nota: El fichero debe tener un contenido adecuado para evitar errores del sistema

Permisos de directorios

➤ ls -ld

- Se ve el directorio en lugar de su contenido
- Permite ver los permisos y propietarios de un directorio

➤ r: Lectura

- Se permite examinar el contenido
- Si el permiso no está activo se puede acceder al contenido del directorio si se conoce su nombre

➤ w: Control Administrativo

- Se permite crear ficheros y directorio, renombrarlos o eliminarlos
- Para eliminar directorios dependerá de los permisos que se tengan sobre su contenido

➤ x: Permiso de Paso

- Si no se tiene no se tiene posibilidad de acceder o modificar el contenido del directorio ni de ninguno de sus subdirectorios

Permisos Especiales

➤ ficheros

Para ficheros binarios ejecutables

Permiten acceder a los ficheros con los permisos del propietario del ejecutable.

- **setuid**
 - Se accede con los permisos del usuario propietario del programa
- **setgid**
 - Se accede con los permisos del grupo propietario del programa

➤ directorios

- **setgid**
 - Los elementos de nueva creación tendrán como grupo propietario al grupo propietario del directorio en el que se crean en lugar del grupo principal del usuario que lo crea
- **sticky bit**
 - Solo el usuario o el grupo propietario podrán renombrar o borrar elementos aunque “otros” tenga permiso de escritura sobre el directorio (“otros” pierde el control administrativo)

➤ **visualización de los permisos**

se integran en el permiso de ejecución:

- **setuid: s o S en usuario**
- **setgid: s o S en grupo**
- **sticky bit: t o T en otros**

minúscula indica que el permiso de ejecución está activado

mayúscula indica que el permiso de ejecución está desactivado

p.e.

rws rwS rwx:

- setuid activado; ejecución para usuario activado
- setgid activado; ejecución para grupo desactivado

rw- rws rwT:

- setgid activado; ejecución para grupo activado
- sticky bit activado; ejecución para otros desactivado

Propietarios

- Solo root puede cambiar el usuario o el grupo propietario de un elemento.
- Antiguamente podía hacerlo, además, el usuario propietario (esta capacidad es modificable en algunos sistemas)

➤ Cambio de grupo propietario

- **chgrp [-R] nuevo_grupo_propietario elemento [...]**
 - -R: Recursivo

➤ Cambio de usuario propietario

- **chown [-R] nuevo_usuario_propietario elemento [...]**
 - -R: Recursivo
 - Actualmente existe la posibilidad de cambiar el grupo con este comando
chown [-R] usuario:grupo elemento [...]
o bien
chown [-R] usuario.grupo elemento [...]

Modificar permisos

- **chmod [-R] *nuevos_permisos* elemento [...]**
 - -R: Recursivo
- ***nuevos_permisos* en Modo Absoluto (o numérico u octal)**
 - hasta 4 dígitos
 - **De mayor a menor peso:** Especiales – usuario – grupo – otros
 - **Cada dígito octal se convierte en binario de tres dígitos**
 - 0 indicará permiso desactivado
 - 1 indicará permiso activado

p.e. (sin contar los especiales):

 - 644 = 110 100 100 = r w - r - - r - -
 - 51 = 051 = 000 101 001 = - - - r - x - - x
- **Se utiliza para establecer permisos sin tener en cuenta lo existente**

➤ **nuevos_permisos** en modo Simbólico

- **formato** <quien><operador><permisos>[,<><><>,...]
- **<quien>**
 - u: usuario
 - g: grupo
 - o: otros
 - a: todos
 - mezclas: ug, uo, go
- **<operador>**
 - +: poner
 - -: quitar
- **<permisos>**
 - r: lectura
 - w: escritura
 - x: ejecución
 - s: setuid o setgid
 - t: sticky bit
 - mezclas: rw, rx, rwx,
- **un caso especial:**
 - <quien>=<quien>

Permisos en elementos de nueva creación

➤ permisos máximos

- ficheros: rw- rw- rw-
- directorios: rwx rwx rwx

➤ *máscara*

- umask [valor]
- Restará, bit a bit, sobre los permisos máximos
- Mismo formato que los permisos en octal
- La misma máscara para ficheros o directorios
- Se establece en el login en el fichero `/etc/profile`
- valores habituales:
 - 022 para sistemas estándar
 - 077 para sistemas de alta seguridad



Reglas de Acotación

Reglas de Acotación

Para eliminar el significado especial de los caracteres

➤ \ (Barra invertida)

- Elimina el significado especial del siguiente carácter

➤ '....' (Comilla sencilla)

- Elimina el significado especial de todos los caracteres encerrados entre ellas
- No puede utilizarse para acotar una comilla sencilla.

➤ "...." (Comillas dobles)

- Elimina el significado especial de todos los caracteres encerrados entre ellas excepto de:

\$ ` \\$ ` \" \\

➤ `....` (Comillas inversas)

- No es, estrictamente, una regla de acotación.
- Ejecuta el comando (o pipeline) entre ellas y deja el resultado para ser utilizado por el shell.

➤ **Prompt Secundario (PS2)**

- Aparece cuando el shell espera que se cierre una pareja de comillas
- El valor habitual es ">"
- Las pulsaciones de la tecla "Return" las interpretará el shell como saltos de línea dentro del comando



Redireccionamientos

Flujo de información



➤ Redireccionamiento de entrada

- `comando < fichero`

➤ Redireccionamiento de salida

- `comando > fichero` Crea el fichero. Si existe contenido, lo borra.
- `comando >> fichero` Crea el fichero. Si existe contenido, lo mantiene y añade lo nuevo al final.

➤ pipelines

- `comando1 | comando2`

➤ Redireccionamiento de errores

- `comando 2> fichero`
- `comando 2>> fichero`
- Si se quieren despreciar los errores se utiliza como fichero el dispositivo nulo (`/dev/null`)

➤ **filtro tee**

- **comando1 | tee [-a] fichero | comando2**

➤ **Guardar error y salida en el mismo fichero**

- **2>&1** Primero debe redireccionarse la salida



Búsqueda

whereis

- Búsqueda de ejecutables, código fuente o manuales

➤ **whereis [opciones] fichero [...]**

- opciones:
 - **-b** busca ejecutables (binarios)
 - **-m** busca manuales
 - **-s** busca códigos fuente

El comando find

- Búsqueda de ficheros o directorios en todo el árbol
- Suele ser recomendable eliminar los errores con 2>/dev/null

➤ **find dir_inicio [opciones]**

- **dir_inicio**: punto de inicio de la búsqueda
- **opciones**: para restringir la búsqueda. Se unen por "y lógico" salvo que se indique de forma especial.
- principales opciones:
 - **-name *nombre*** (pueden utilizarse metacaracteres pero es necesario acotar el nombre si se hace)
 - **-type *tipo*** (f, d, l, c o b)
 - **-user *usuario*** (nombre o número)
 - **-group *grupo*** (nombre o número)
 - **-inum *número***
 - **-links [**+**|-|] *número*** (enlaces duros)
 - **-follow** (sigue los enlaces simbólicos)
 - **-size [**+**|-|] *número* [**c**]** (Tamaño en bloques o bytes (caracteres))
 - **-maxdepth *número*** (límite de profundidad de subdirectorios)
 - **-perm *permisos*** (coincidencia exacta en octal)
 - **-perm -*permisos*** (coincidencia por máscara en octal: todos los indicados)
 - **-perm /*permisos*** (coincidencia por máscara en octal: cualquiera de los indicados)
 - **-print** (muestra el nombre encontrado)

- Creación de condiciones complejas

Se siguen las reglas de precedencia (not - and - or)

Si se quieren cambiar se debe encerrar entre paréntesis. Ojo, deben ir acotados: `\(` y `\)`

- `!` Negación. Con cualquier opción. Se usa antecediéndola a la opción pero separada por un espacio (p.e. `! -user root`)
- `-a` y lógico
- `-o` o lógico

- Ejecución de comandos

Se ejecutará por cada uno de los elementos encontrados

Si se quiere utilizar el nombre dentro del comando se pondrá `}` en el lugar correspondiente

- `-exec comando \;` (Ejecuta sin pedir permiso)
- `-ok comando \;` (Pide confirmación por cada elemento encontrado)



Filtros

cat

muestra fichero

➤ **cat [opciones] fichero [...]**

➤ **... | cat [opciones] [fichero ...]**

- en la segunda sintaxis un '-' representa a stdin
- principales opciones:
 - **-tv** (muestra tabulaciones como ^I)
 - **-ev** (muestra finales de línea como \$)

od / hexdump

Volcado de ficheros

➤ **od [opciones] fichero [...]**

➤ **... | od [opciones]**

- -b: octal
- -c: ASCII. los caracteres no imprimibles los muestra por su símbolo unix \t, \n...)
- -x: hexadecimal
- sin opciones: octal de dos bytes

➤ **hexdump [opciones] fichero [...]**

➤ **... | hexdump [opciones]**

- sin opciones: hexadecimal
- -b: octal
- -c: ASCII. los caracteres no imprimibles los muestra por su símbolo unix \t, \n...)
- -C: hexadecimal + imprimibles en paralelo

Paginadores

Depende del sistema puede haber uno o varios

➤ **comando_paginador [opciones] fichero [...]**

➤ **... | comando_paginador [opciones]**

- principales comandos:
 - **more** (existe siempre)
 - **less** (el habitual en los sistemas linux)
 - **pg** (en sistemas unix)
- Cada uno tiene sus características, pero habitualmente es común:
 - **h** (ayuda)
 - **q** (salir)

WC

Cuenta caracteres, palabras y líneas

➤ **wc [opciones] fichero [...]**

➤ **... | wc [opciones]**

- principales opciones:
 - **-c** (cuenta caracteres)
 - **-w** (cuenta palabras)
 - **-l** (cuenta líneas)
 - sin opciones cuenta todo
- Si se especifica algún nombre del fichero, este aparece como cuarto campo de la salida
- Si se especifican varios ficheros se añadirá una línea con el total

nl

numera líneas

➤ **nl [opciones] fichero [...]**

➤ **... | nl [opciones]**

- principales opciones:
 - **-ba** (numera las líneas en blanco)

tr

Modifica el contenido de las líneas:

- Eliminando o sustituyendo caracteres
- Eliminando repeticiones

➤ ... | **tr [opciones] *lista1* [*lista2*]**

- lista1 o lista2 puede ser un rango o utilizar clases
- Usos:
 - `tr lista1 lista2`
 - Sustituye los caracteres de lista1 por los de lista 2 (en el orden en el que se encuentran en cada una de las listas. Si hay más en lista1 se usa el último de lista2 para las sobrantes)
 - `tr -d lista1`
 - Elimina los caracteres de lista1
 - `tr -s lista1`
 - Comprime los caracteres de lista1 (elimina repeticiones)
 - `tr -s lista1 lista2`
 - Comprime las apariciones de los caracteres de lista1 y los sustituye por los de lista2
 - `tr -ds lista1 lista2`
 - Elimina los caracteres de lista1 y comprime los de lista2
- Ejemplo
 - `tr [:lower:] [:upper:]` convierte todo a mayúsculas

sed

Modifica el contenido de las líneas:

- Eliminando o sustituyendo cadenas

➤ ... | **sed 's/cadena1/[cadena2]/[g]'**

- cadena1 puede ser una expresión regular (ver grep)
- Si se cadena2 se deja vacía se eliminará cadena1
- Si no se especifica la "g" final solo se cambiará la primera aparición de cadena1 en la línea. Si se especifica las cambiará todas
- Puede utilizarse el símbolo "/" o cualquier otro con la condición de que no se encuentre ni en cadena1 ni en cadena2.

- Ejemplos

sed 's/hola/adios/g' sustituye todos los "hola" por "adios"

sed 's/:-:g' sustituye todos los "/" por "-"

sed 's/ \{1,\}/ /g' comprime los espacios

sed 's/^ //' elimina un espacio al principio de la línea

sed 's/[[:blank:]]*//g' elimina todos los blancos (espacios y tabulaciones) de la línea

sed 's/^ *//' elimina todos los espacios al principio de la línea

Ordenación

- **sort [-c] [opciones] [-tx] [campos_ord] fichero ...**
- **... | sort [-c] [opciones] [-tx] [campos_ord]**
 - Si se utiliza la entrada estándar con más ficheros se pondrá - en el lugar que corresponda de la lista de ficheros
 - **-c** comprueba la ordenación (se verifica con el valor de status: \$?)
 - Opciones que cambian el comportamiento
 - **-u** (suprime las líneas duplicadas en la salida)
 - Opciones que cambian la ordenación
 - **-r** (ordenación inversa)
 - **-n** (ordenación numérica)
 - **-d** (ordenación por diccionario: LOCALE)
 - **-f** (ignora mayúsculas y minúsculas: LOCALE)
 - **-b** (ignora "blancos" al principio: LOCALE)

- Ordenación por campos
 - **-tx** (hace que x sea el carácter separador de campos. Puede ser necesario acotarlo. Por defecto utiliza "blancos")
- Sistema 1: **-*inicio*[,*final*] [...]**
 - inicio y final indican los campos que quieren utilizarse para ordenar
 - El primer campo de la línea toma el número 1
 - Si no se indica final tendrá en cuenta desde el inicio del campo hasta el final de la línea
 - Se pueden utilizar flags particulares para cada campo (bdfnr)
 - Ejemplos:
 - -k3
 - -k3,3
 - -k1,1 -k4,4
 - -k1,2 -k3r,3
- Sistema 2: **+*inicio* -*final* [...]**
 - inicio y final están compuestos de parte entera y fraccionaria. La parte entera indica el campo y la fraccionaria el carácter dentro del campo
 - El primer campo de la línea toma el número 0
 - Se pueden utilizar flags particulares para cada campo (bdfnr)
 - Ejemplos:
 - +2 -3
 - +0 -1 +3 -4
 - +0 -2 +2r -3
 - +3.4 -4

head

Selecciona las primeras líneas

➤ **head [opciones] fichero [...]**

➤ **... | head [opciones]**

- normalmente selecciona las 10 primeras líneas
- principales opciones:
 - **-n** (selecciona **n** líneas)
 - **-n -num** (selecciona todas menos las últimas **num** líneas)

tail

Selecciona las últimas líneas

➤ **tail [opciones] fichero [...]**

➤ **... | tail [opciones]**

- normalmente selecciona las 10 últimas líneas
- principales opciones:
 - **-num** (selecciona las **num** últimas líneas) (en algunos linux: **-n num**)
 - **-n +num** (selecciona desde la línea **num** hasta el final)
 - **-F** (Seguir un fichero) (solo con sintaxis tail -F fichero)

grep

Selecciona líneas que cumplen una determinada condición

➤ **grep [opciones] cadena fichero [...]**

➤ **... | grep [opciones] cadena**

- principales opciones:
 - **-i** (ignora mayúsculas/minúsculas)
 - **-v** (selecciona líneas que **no** contienen la cadena)
 - **-n** (muestra el número de línea)
 - **-l** (muestra ficheros que contienen (o no) la cadena)
- cadena:
 - Sin acotar si es palabra sencilla
 - Acotada si lleva caracteres "extraños"

- Expresiones Regulares
 - '^cadena' en el principio de la línea
 - 'cadena\$' en el final
 - '^cadena\$' línea completa
 - '^\$' líneas vacías
 - '.' un carácter cualquiera
 - '[lista]' un carácter de los de la lista
 - rango con '-'
 - '[^lista]' un carácter que no esté en la lista
 - clases
 - '[:lower:]' [:upper:] [:digit:] [:alpha:] [:alnum:]
 - repeticiones el carácter anterior (poner entre comillas)
 - '\{n\}' n apariciones
 - '\{n,\}' ≥ apariciones
 - '\{n,m\}' entre n y m apariciones (incluidos)
 - '*' ≥ 0 apariciones
 - Ejemplos:
 - '[0-9]\{10\}' 10 dígitos
 - '[a-zA-Z]\{5,\}' ≥ 5 letras
 - 'ab*c' ac, abc, abbc, abbbc, ...

cut

Selecciona caracteres o campos de las líneas

➤ **cut [opciones] -c|-f lista fichero [...]**

➤ **... | cut [opciones] -c|-f lista**

lista indicará la información a extraerse

(lista de números separada por comas que puede incluir rangos: i-f, i-, -f)

- opciones de selección:
 - **-c** (caracteres)
 - **-f** (campos)
- opciones:
 - **-dx** (con -f. Carácter delimitador de campos. Puede ser necesario acotarlo. Si no se indica se tomará \t)
 - **-s** (con -f. Suprime las líneas sin delimitador de campos)
- Ejemplos
 - `echo $HOME | cut -d/ -f2`
 - `who | tr -s " " | cut -d" " -f3 | cut -c1-4`

paste

fusiona ficheros y los separa por campos

➤ **paste [opciones] fichero [fichero2 ...]**

➤ **... | paste [opciones] [fichero...]**

se pone - para fusionar con stdin.

- opciones:
 - **-dx** (Carácter delimitador de campos. Puede ser necesario acotarlo. Si no se indica se tomará \t)
 - **-s** (compatibilidad con versión antigua del paste: fusiona todas las líneas de un fichero en una sola separándolas por el carácter elegido. al final añade el carácter de fin de línea. Si hay más de un fichero hará una línea con cada uno de ellos)

join

fusiona dos ficheros ordenados por un campo común (ordenación ASCII ascendente). El campo común solo sale una vez

➤ **join [opciones] fichero1 fichero2**

➤ **... | join [opciones] fichero1 - | - fichero2**

- opciones:
 - **-tx** (Carácter delimitador de campos. Puede ser necesario acotarlo. Si no se indica se tomarán blancos)
 - **-j1n -j2m (-1n -2n)** (utiliza el campo n del fichero 1 y el campo m del fichero 2 como clave para la fusión. Los campos empiezan a numerarse por 1)
 - **-jn** (utiliza el campo n de ambos ficheros como clave)
 - **-a1 | -a2** (se añaden las líneas del fichero indicado que no tengan su correspondiente en el otro)
 - **-v1 | -v2** (se muestran solo las líneas del fichero indicado sin su correspondiente en el otro)

split

divide un fichero en varios con un prefijo común y un sufijo ascendente (de una o más letras).

Si no se indica prefijo, éste será x.

➤ **split [opciones] fichero [prefijo_fichero_salida]**

➤ **... | split [opciones] - [prefijo_fichero_salida]**

- opciones:
 - **-an** (el sufijo tendrá *n* letras. Por defecto, 2)
 - **-ln / -n** (genera ficheros de *n* líneas. Por defecto genera ficheros de 1000 líneas)
 - **-bn** (genera *n* ficheros de igual tamaño)
 - **-bn[k|m]** (genera ficheros de *n* Kb o Mb)

diff

compara dos ficheros y muestra un indicador del lugar de la diferencia y el contenido de ambos ficheros

diff3 sirve para comparar tres ficheros (ver man)

➤ **diff [opciones] fichero1 fichero2**

➤ **... | diff [opciones] - fichero**

- opciones:
 - **-b** (ignora blancos consecutivos)
 - **-c** (muestra contexto: tres líneas anteriores y posteriores a la diferencia)

sdiff

compara dos ficheros y muestra en paralelo el contenido de ambas (mejor no usar con líneas largas)

en medio aparecerá un indicador:

- blanco: líneas iguales
- |: líneas diferentes
- < o > línea en solo un fichero

➤ **sdiff [opciones] fichero1 fichero2**

➤ **... | sdiff [opciones] - fichero**

- opciones:
 - -s (no imprime las líneas iguales)

comm

compara dos ficheros y muestra las líneas que va comparando en un máximo de tres columnas:

- Columna1: si la línea solo está en el fichero1
- Columna2: si la línea solo está en el fichero2
- Columna3: Si está en ambos ficheros

➤ **comm [opciones] fichero1 fichero2**

➤ **... | comm [opciones] - fichero**

- opciones:
 - **-1** (elimina la columna de salida 1)
 - **-2** (elimina la columna de salida 2)
 - **-3** (elimina la columna de salida 3)

cmp

compara dos ficheros y muestra donde encuentra las diferencias (si son iguales no dice nada). Si no se le indica lo contrario, se parará al encontrar la primera diferencia.

➤ **cmp [opciones] fichero1 fichero2**

➤ **... | cmp [opciones] - fichero**

- opciones:
 - **-l** (muestra todas las diferencias)
 - **-s** (no produce salida. Se utiliza para comprobar después el valor de STATUS)