Unix

Sistemas Operativos

Escuela de Ingeniería Civil Informática

- Introdución al Sistema Operativo Unix

Luis Gajardo - *lgajardo@ubiobio.cl*

UNIVERSIDAD DEL BÍO-BÍO

HISTORIA DE UNIX

- Unix es un sistema operativo, multitarea y multiusuario.
- Existen versiones para máquinas de un procesador hasta para multiprocesadores.
- Unix es una nueva implementación de un proyecto que fracasó, llamado MULTICS (*Multiplexed Information and Computing Service*), entre 1965 y 1973.
- Fue desarrollado en principio por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie.
- Luego se expande a las Universidades norteamericanas, donde la Universidad de Berkeley crea su propia versión (BSD).

HISTORIA DE UNIX

- Tanto Dennis Ritchie como Ken Thompson han sido premiados por sus aportes en el campo de los sistemas operativos con:
  - El Premio *NEC C&C* en 1979
  - El Premio Turing de la ACM en 1983
  - La Medalla Nacional de Tecnología de los Estados Unidos en 1998

HISTORIA DE UNIX

- Hoy en día todas las versiones de Unix se basan en una de dos versiones principales:
  - Unix BSD (Universidad de Berkeley - USA)
  - Unix System V (la original de AT&T)
- Distribuciones que podemos encontrar actualmente de Unix son:

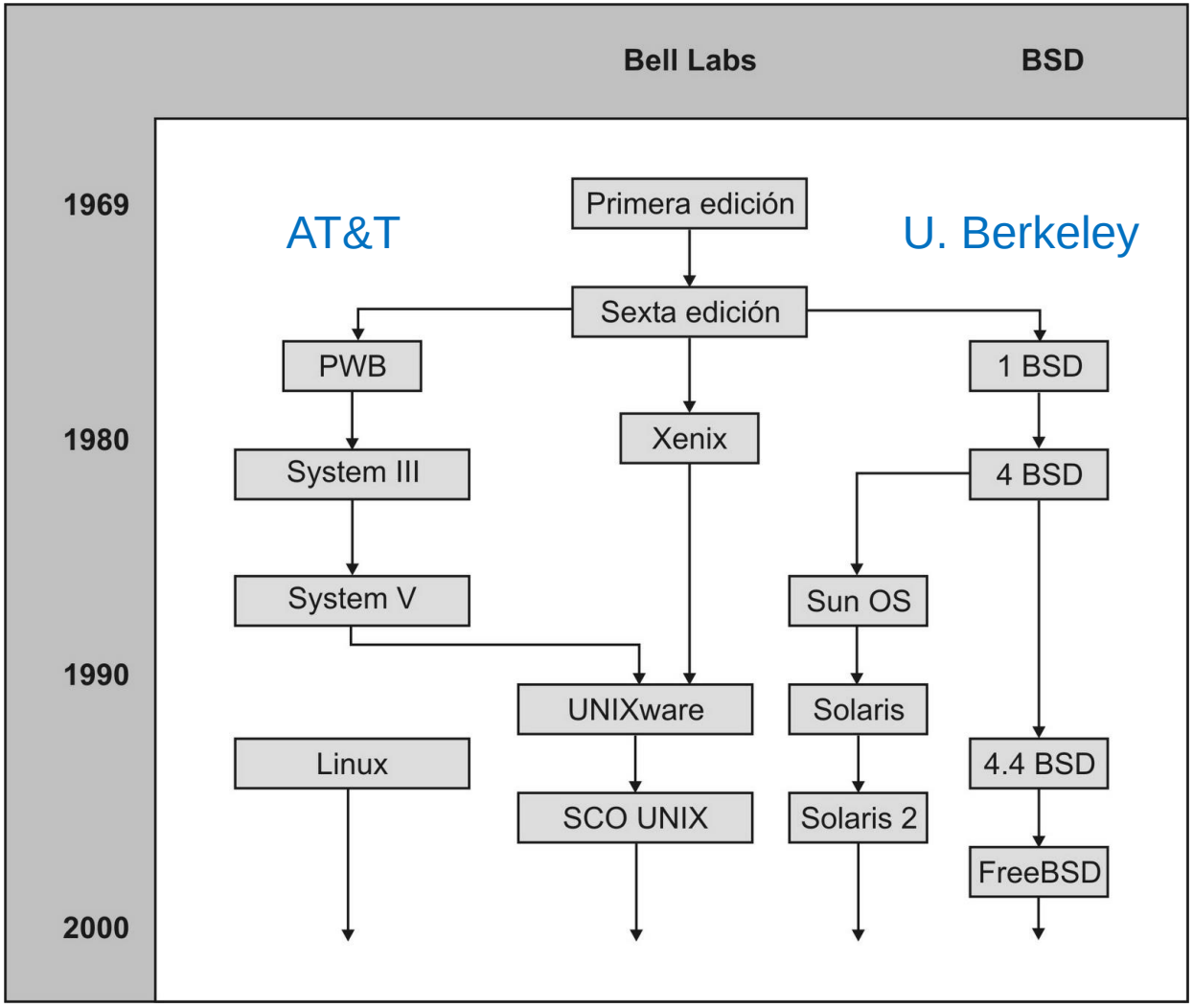
SUN Microsystems

IBM

GNU

Digital, luego Compaq, Ahora HP

EVOLUCIÓN DE UNIX



HISTORIA DE LINUX

- Linux evoluciona de Unix como la primera versión de libre distribución (GNU) para PC.

- La primera versión fue desarrollada en 1991 por un alumno de la Universidad de Helsinki, llamado **Linus Torvalds**.

- Torvalds cursaba la asignatura de sistemas operativos dictado por el profesor Andrew Tanenbaum, cuando molesto por las restricciones, en materia de licencias, impuestas por Minix para expandir sus comandos, crea su propio sistema operativo para PC.

ESTANDARIZACIÓN DE UNIX

- Debido a las múltiples versiones en el mercado de UNIX, se comenzaron a publicar estándares para que todas las versiones fuesen compatibles.

- La primera de ellas la lanzó AT&T llamada SVID (*System V Interface Definition*) que definía:

  - Cómo deberían ser las llamadas al sistema (System Call)
  - El formato de los archivos
  - etc.

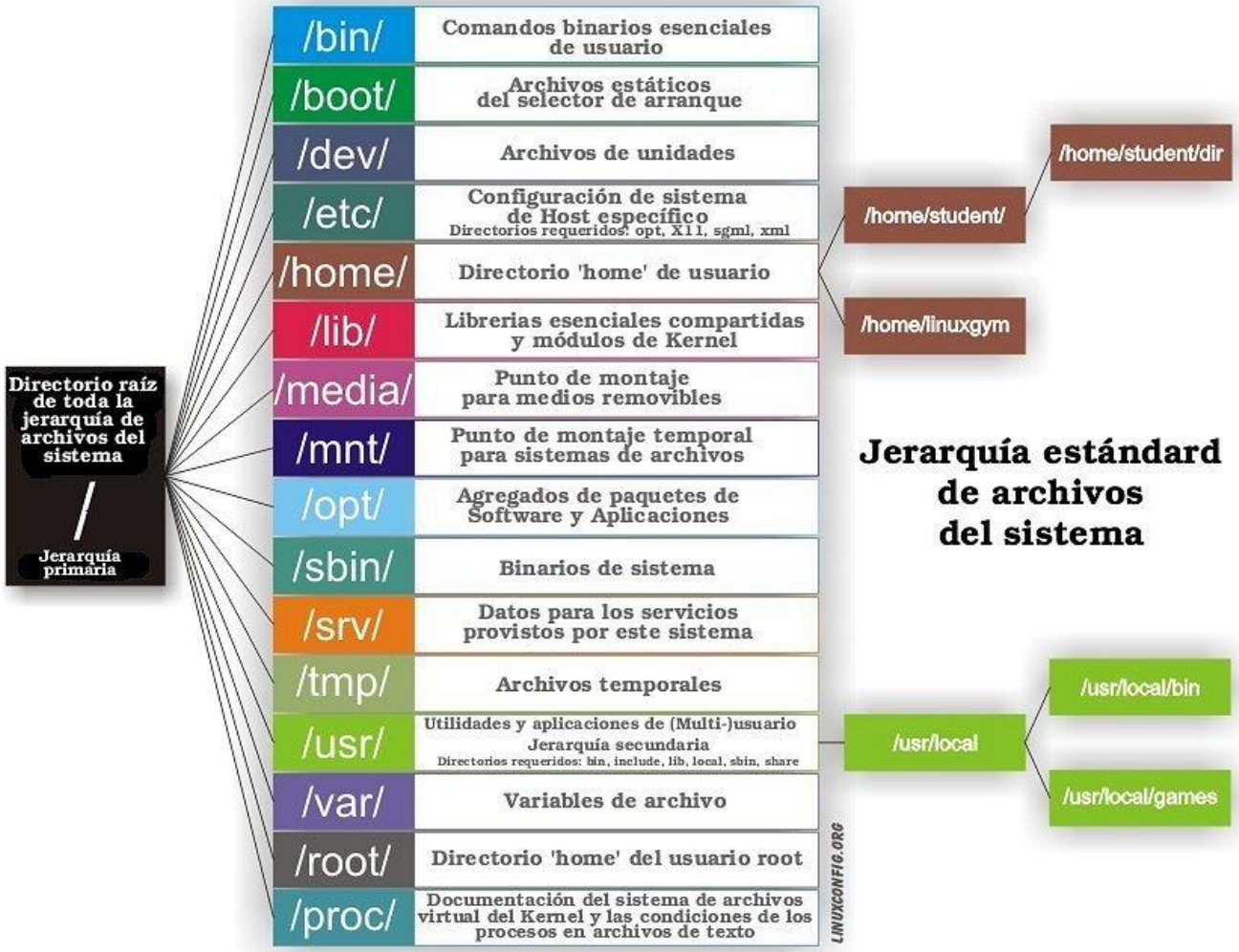
- Otra versión importante fue la de la Universidad de Berkeley (Berkeley Software Distribution o BSD) que simplemente las ignoró.

- Posteriormente la IEEE usó un algoritmo que revisó ambas versiones y buscó las similitudes, a las cuales definió como estándar y llamó **POSIX** (*Portable Operating System for Unix*) la que fue adoptada por muchos fabricantes.
  IEEE: Instituto de Ingenieros Eléctricos y Electrónicos

ESTANDARIZACIÓN DE UNIX

- El estándar de POSIX se llama 1003.
- Algunas de las definiciones del estándar POSIX son las siguientes:
  - 1003.0 Introducción y repaso
  - 1003.1 Llamadas al sistema
  - 1003.2 Intérprete de comandos
  - 1003.3 Métodos de prueba
  - 1003.4 Extensiones para tiempo real
  - 1003.5 Lenguaje Ada
  - 1003.6 Extensiones para la seguridad
  - 1003.7 Administración del Sistema
  - 1003.8 Acceso transparente a archivos
  - 1003.9 Lenguaje Fortran
  - 1003.10 Supercómputo

SISTEMA DE ARCHIVOS



SISTEMA DE ARCHIVOS

Comando cd

(change directory) Cambiar el directorio

Navegación por el árbol de directorios

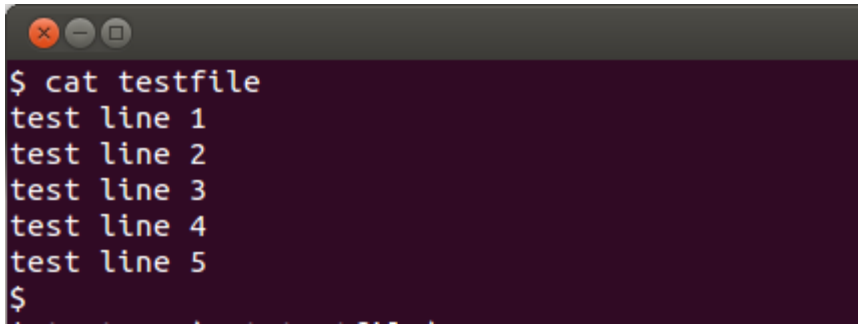
`cd /path`	lleva al directorio cuya trayectoria completa es /path
`cd.`	lleva al directorio actual
`cd..`	sube al directorio padre
`cd (sin parametros)`	lleva al home de tu usuario
`cd ~ nombreusuario`	lleva al home de nombreusuario
`cd ~`	lleva al home de tu usuario
`cd /`	lleva al directorio raíz
`cd -`	lleva al último directorio visitado.

INTÉRPRETE DE COMANDOS

- Un shell o intérprete de comandos es el programa encargado de traducir los comandos que los usuarios introducen a instrucciones que el sistema operativo entiende.

- El prompt es un símbolo que indica que el sistema está listo para recibir un comando.

- El prompt del sistema es el carácter $ o el carácter % para los usuarios y el carácter # para el administrador del sistema (root).



INTÉRPRETE DE COMANDOS

Tareas de la Shell

- Lee y analiza la entrada de la línea de comandos.
- Maneja caracteres especiales, redirecciones, tuberías y control de trabajos (en primero o segundo plano).
- Busca el comando en el disco y si lo encuentra, lo ejecuta. Esto se llama utilizar la shell interactivamente.
- Maneja señales.
- Prepara la ejecución de programas.

INTÉRPRETE DE COMANDOS

Principales shells de Unix/Linux

- **Shell de Bourne (bsh, de ATT)**
  - Es la shell estándar en modo de superusuario.
  - Se usa para para administrar los sistemas Unix.
  - En ella está escrita la mayoría de los scripts de administración.
  - Se arranca con el comando `/bin/bsh`.
  - El símbolo que la acompaña es '$'.
- **C shell (csh, de Berkeley)**
  - Añade más características: historia de los comandos ejecutados, aritmética, etc.
  - Es más lenta para los mismos scripts escritos en la shell de Bourne.
  - Se arranca con el comando `/bin/csh`.
  - El símbolo que la acompaña es '%'.
- **Shell de Korn (ksh, extensión de la shell de Bourne).**
  - Es un superconjunto de la shell de Bourne.
  - Dispone de características extras de la C Shell: funciones, etc.
  - Se arranca con el comando `/bin/ksh`.
  - El símbolo que la acompaña es '$'.

METACARACTERES

Símbolo	Descripción
*	Sustituye a cualquier número de caracteres dentro de un texto.
?	Sustituye a un único carácter dentro de un texto.
|	Tubería o pipe. Utiliza la salida de un comando como entrada a otro.
>	Redirecciona la salida estándar hacia un archivo, creándolo si no existe o sustituyendo su contenido si es que ya existe.
>>	Redirecciona la salida estándar hacia un archivo, creándolo si no existe o añadiendo nueva información si es que ya existe.
2>	Idéntico a > pero redireccionando hacia la salida estándar de errores
2>>	Idéntico a >> pero redireccionando hacia la salida estándar de error
&	Ejecuta un proceso en segundo plano o background
\	Carácter de escape. El siguiente carácter posterior a éste se ignora
[…]	Sustituye cual valor incluido entre los corchetes.

METACARACTERES

Ejemplos

- `c?` : incluye c1, c2, cb, ck, c_, etc.
- `c?b??` : incluye c1b12, chbk2, etc
- `a*` : incluye todos los términos que empiezan por a.
- `*a*` : incluye todos los términos que contienen el carácter a.
- `c[12a]` : incluye a c1, c2, ca.
- `c[1-4]` : incluye c1, c2, c3 y c4.
- `c[!xy]` : incluye todos los términos que empiezan por c y su segundo carácter no es ni x ni y.

FLUJOS DE COMUNICACIÓN DE DATOS

Unix/Linux dispone de tres formas para comunicarse con el exterior:

- **Entrada estándar**
  - Se utiliza para introducir datos en la shell.
  - Abre el descriptor 0 (stdin).
- **Salida estándar**
  - Se utiliza para mostrar datos al ejecutar órdenes o procesos.
  - Abre el descriptor 1 (stdout).
- **Errores estándar**
  - Se utiliza para mostrar errores al ejecutar órdenes o procesos.
  - Abre el descriptor 2 (stderr).
  - Por defecto estos errores aparecen por la salida estándar.

En Unix/Linux tanto los archivos como directorios y todo tipo de dispositivos de E/S, son tratados como archivos.

REDIRECCIÓN

Es el mecanismo por el cual se dirige la entrada o la salida estándar de un comando desde o hacia un archivo:

- Para redirigir la entrada estándar:
  `orden < archivo` (orden “lee” desde archivo)

- Para redirigir la salida estándar:
  `orden > archivo` (orden “escribe/sobreescribe” en archivo)

- Si se utiliza el operador ‘>>’, la salida del comando se añade al final del archivo:
  `orden >> archivo` (orden “añade datos” a archivo)

REDIRECCIÓN

Ejemplos

- `$ ls *.mp3 > listaDeFotosyMusica.txt`
- `$ ls *.jpg > listaDeFotosyMusica.txt`

TUBERÍAS (PIPES)

- La tubería (el carácter '|') permite utilizar la salida de un comando para servir como entrada de otro.
  - `ls –l | more`
  - `ls -l | grep txt`

- En estos dos ejemplos `ls –l` es un comando que muestra una relación de los archivos del directorio actual.

- El comando `more` detiene la salida cuando la pantalla se llena y se queda a la espera de teclear algo.

- El comando `grep` con un parámetro busca dentro de un archivo si existe el patrón indicado en el parámetro.

- Luego `ls –l | grep txt` presentará por pantalla aquellos archivos que contengan en su interior la cadena de caracteres "txt".

TUBERÍAS (PIPES)

- El siguiente ejemplo muestra una orden compuesta que ordena todos los archivos con extensión "`.txt`", elimina las líneas duplicadas y guarda los datos en el archivo "`resultado.sal`".

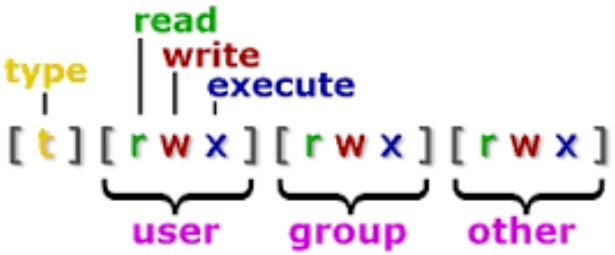
```
cat *.txt | sort | uniq > resultado.sal
```

- Este ejemplo realiza una copia de un archivo convirtiendo a mayúsculas todos los caracteres del archivo original.

```
cat arch | tr 'a-zñáéíóúü' 'A-ZÑÁÉÍÓÚÜ‘ > arch.sal
```

PERMISOS

- Un sistema Unix/Linux es multiusuario, por lo que los archivos de cada usuario deben estar protegidos del resto de usuarios.
- Unix/Linux dispone de tres tipos de permisos (read.4, write:2, execute:1) y tres tipos de usuarios (user, group, other).



- El comando `chmod` permite cambiar los permisos de los archivos.

PERMISOS

Ejemplos

- `chmod u+w hola.c` : añade permiso de escritura sobre el archivo hola.c al propietario.
- `chmod o-r hola.c` : suprime el permiso de lectura de hola.c al resto de usuarios.
- `chmod rw hola.c` : añade permiso de lectura y escritura sobre el archivo hola.c a todos los usuarios.
- `chmod rw *.c` : añade permiso de lectura y escritura sobre todos los archivos con extensión .c a todos los usuarios.
- `chmod 644 hola.c` : establece el permiso de lectura y escritura para el propietario y de lectura para el grupo y resto de usuarios.
- `chmod = hola.c` : desactiva todos los permisos de hola.c
- `chmod 000 hola.c` : idéntico a chmod = hola.c

SHELL SCRIPT

- Un shell-Script es un archivo de texto que automatiza tareas al estilo de ficheros batch para DOS.
- Pueden crearse en cualquier editor de texto o bien con el comando cat.
- La primera línea de este archivo, es la ruta del intérprete de comandos que usará nuestro script, precedido de un #!
- Luego se escriben todos los comandos que se necesiten.
- También es posible escribir comentarios, comenzando la línea con un #

SHELL SCRIPT

Variables

- •Las variables en el Shell son por defecto alfanuméricas.
  •Asignación de un valor a una variable:
  Nombre=valor
  •Acceso a su contenido:
  $Nombre
  •*El comando **read** se usa para leer variables desde teclado*
  •*El comando **echo** se usa para ver el contenido de una variable.*

SHELL SCRIPT

Variables Especiales

- •$0: Nombre del Shell-Script que se está ejecutando.
  •$n: Parámetro o argumento pasado al Shell-Script en la posición n, n=1,2,...
  •$#: Número de argumentos.
  •$*: Lista de todos los argumentos.
  •$$: PID del proceso que se está ejecutando.
  •$!: PID del último proceso ejecutado.
  •$?: Salida del último proceso ejecutado.
  •$PATH :
  $PS1 : Prompt

SHELL SCRIPT – CONTROL DEL FLUJO

Estructura IF

- if condicion1
  then
  comandos si la condicion1 se cumple
  elif condicion2
  then
  comandos si la condición2 se cumple
  else
  comandos si no se cumplen 1 y 2
  fi

SHELL SCRIPT – CONTROL DEL FLUJO

Estructura WHILE

- while condicion
  do
  comandos
  done
  •Estructura Until
  until condicion
  do
  comandos
  done

SHELL SCRIPT – CONTROL DEL FLUJO

Estructura FOR

- for variable in [lista de valores]
  do
  Comandos
  done

Estructura CASE

- Case
  case variable in
  patron1) comandos condicion1;;
  patron2) comandos condicion2;;
  ...
  patron n) comandos condicion n;;
  *) comandos por defecto;;
  esac

SHELL SCRIPT – CONDICIONES

Archivos

- •-f = true si fichero existe
  •-r = true si fichero existe y con derecho de lectura
  •-w = true si fichero existe y con derecho de escritura
  •-x = true si fichero existe y con derecho de ejecución
  •-s = true si fichero existe y no es vacío
  •-d = true si directorio existe

- **Ejemplo:**
  if [! –f fichero]
  true si el fichero no existe

SHELL SCRIPT – CONDICIONES

Cadenas

- -n = true si longitud cadena distinta 0
  •-z = true si longitud cadena es 0
  •= = true si son iguales
  •!= = true si son distintas

- **Ejemplos**
  if [-n cadena1]
  true Si la longitud de la cadena es distinta de 0
  if [cadena1 = cadena2]
  true si las cadenas cadena1 y cadena2 son iguales

SHELL SCRIPT – CONDICIONES

Enteros

- -eq = true si iguales
  -ne = true si distintos
  -gt = true dato1 mayor que dato2
  -ge = true dato1 mayor o igual que dato2
  -lt = true dato1 menor que dato2
  -le = true dato1 mejor o igual que dato2

- **Ejemplos:**
  if [valor –eq 20]
  true si el valor numérico es un determinado escalar
  if [valor1 –gt valor2]
  true si el valor1 es mayor que el valor2

SHELL SCRIPT – EXP. NUMÉRICAS

- Las variables en el Shell son por defecto alfanuméricas. Para darles tratamiento numérico debemos recurrir al comando expr que evalúa expresiones aritméticas.

- **Ejemplo:**
  expr 3 + 4 = 7

- •Utilizando comillas simples inversas podemos asignar comandos a variables.

- **Ejemplo:**
  ocho = `expr 3 + 5 `
  echo $ocho

- •De esta forma aplicando la sustitución de un comando por su resultado, se puede dar tratamiento numérico al contenido de una variable.

SHELL SCRIPT – EXP. NUMÉRICAS

- Las variables en el Shell son por defecto alfanuméricas. Para darles tratamiento numérico debemos recurrir al comando expr que evalúa expresiones aritméticas.

- **Ejemplo:**
  expr 3 + 4 = 7

- •Utilizando comillas simples inversas podemos asignar comandos a variables.

- **Ejemplo:**
  ocho = `expr 3 + 5 `
  echo $ocho

- •De esta forma aplicando la sustitución de un comando por su resultado, se puede dar tratamiento numérico al contenido de una variable.

OPERACIONES SOBRE ARCHIVOS

- **Cat**
- crea, visualiza y concatena un fichero de texto
  •cat > fichero
  crea el fichero de texto. Para terminar con la entrada pulsar Ctrol+D
  •cat fichero
  visualiza el fichero
  •cat fichero1 fichero2 > ficheros1y2
  concatena el fichero “fichero” y lo guarda en el fichero f2.
  split archivo --bytes=1m prefijo

OPERACIONES SOBRE ARCHIVOS

- **Sort fichero**
  Para ordenar un fichero de texto
  •sort –n lo mostrará en orden numérico
  •sort –r en orden inverso.

- **wc fichero**
  Muestra número de líneas, número de palabras y número de caracteres.
  •wc –l fichero, nº de líneas
  •wc –c fichero, nº de palabras
  •wc –w fichero, nº de caracteres.

COMANDOS GREP

- Busca una cadena de caracteres en una serie de ficheros que especificamos como parámetros. Sus opciones serán:
  -c : cuenta las líneas en las que aparece la cadena.
  -l : muestra los nombres de los ficheros en los que aparece la cadena.
  -i : no diferencia entre mayúsculas y minúsculas.
  • filtro de primer carácter.
  ^ principio de línea.
  $ final de línea.
  Ejemplo: mostrar el nombre de los ficheros en el directorio de trabajo en los que aparece alguna línea que comience por c.
  grep –l ^c. $home/*