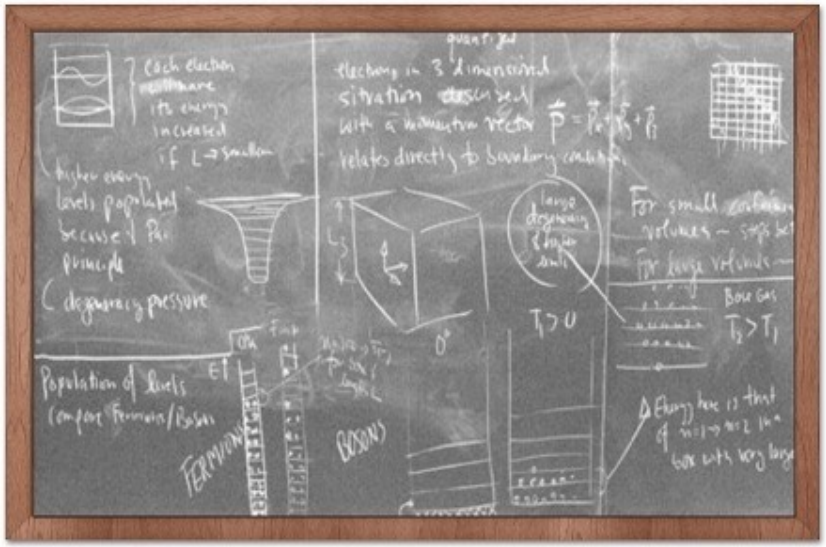
LINUX
Una breve introducción



Física Computacional
Universidad de Granada

Contenidos

- Introducción
- Cómo funciona Linux
- Acceso a Linux
- Algunos comandos sencillos
- Ficheros y directorios
- Permisos, redireccionamientos, encauzamientos
- Procesos
- Variables shell y scripts
- Resumen de comandos útiles

Sistemas operativos

- **Sistema operativo (SO)**: software (programa) encargado de gestionar y usar el hardware (piezas)
- Interfaz amigable para **interactuar** con la compleja red de circuitos y componentes del ordenador
- Ejemplos de SO: Windows, Mac OS X, Linux
- Linux: SO **libre** desarrollado por voluntarios
- Linux: misma funcionalidad que UNIX (privativo)

TOP500 (Nov 2015): LINUX (98.8%), UNIX (1.2%), WINDOWS (0%!!)

Propiedades del LINUX

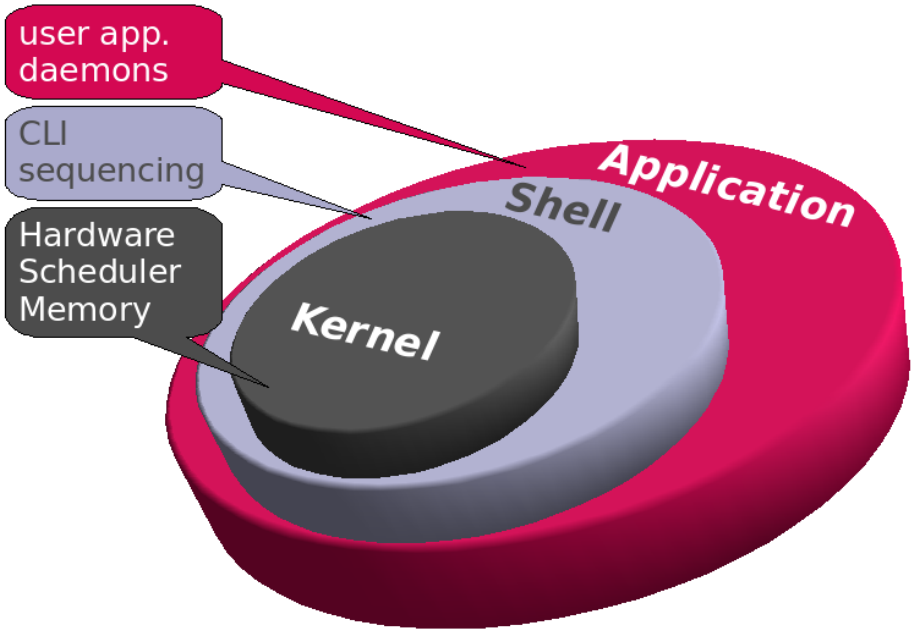
- **Multitarea**: se pueden realizar distintas taréas a la vez
- **Multiusuario**: varios usuarios pueden trabajar concurrentemente en la misma máquina.
- **Multiplataforma**: instalable en multitud de dispositivos (pc's, portátiles, móviles, consolas, etc.)
- **Redes**: permite acceso a recursos remotos y comunicaciones
- **Estabilidad**: puede funcionar correctamente meses, incluso años, sin apagar
- **Libre**: su código fuente se puede usar, modificar y distribuir
- Potente, flexible y versátil

Software libre

- Movimiento iniciado por **Richard Stallman** en 1984 con el proyecto **GNU** (GNU is not Unix)
- **Postulados** (o libertades) del software libre:
  - Libertad de usar el programa, con cualquier propósito
  - Libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.
  - Libertad de distribuir copias del programa, con lo cual puedes ayudar a otros.
  - Libertad de mejorar el programa y hacer públicas esas mejoras a los demás, beneficiando así a la comunidad

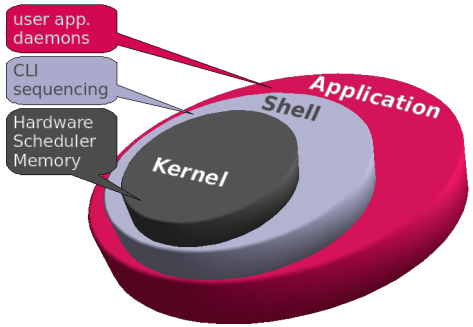
¿Qué es el LINUX?

- Es un Sistema Operativo
- Está formado por:
  - Núcleo (kernel)
  - Shell
  - Sistema de archivos
  - Utilidades



El núcleo (kernel)

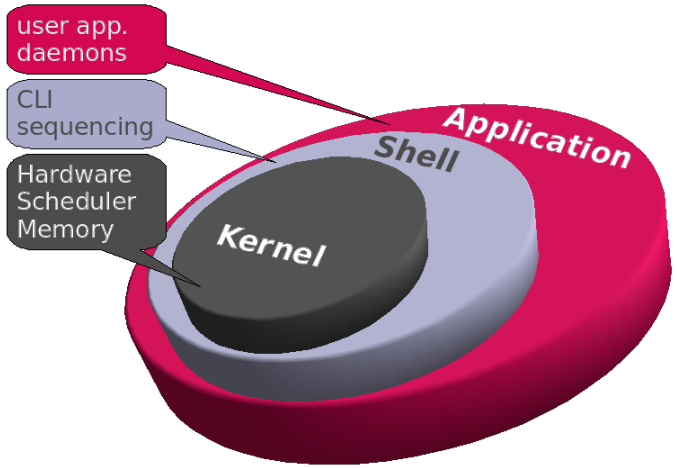
- Interactúa directamente con el **hardware**
- **Funciones**:
  - Gestión memoria
  - Control de acceso al ordenador y permisos
  - Mantenimiento sistema de archivos
  - Manejo interrupciones
  - Manejo Errores
  - Servicios I/O
  - Asignación de recursos entre usuarios
  - Control de procesos y comunicaciones entre procesos



- Todos estos son **procesos de bajo nivel**
- Último núcleo estable (www.kernel.org): 4.4.1 (Enero 2016)

La Shell

- Intérprete de órdenes o comandos (equivale al COMMAND.COM de MS-DOS … ¿os suena?)
- Incluye un lenguaje de programación para procesamiento por lotes
- Existen distintos tipos de shell:
  - bash-shell (LINUX por defecto)
  - Sh
  - C-shell, k-shell, tc-shell



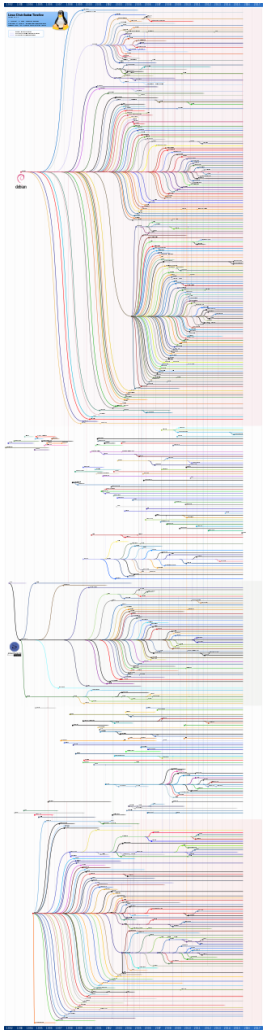
Sistema de archivos

- **Archivo**: unidad básica de organización de la información.
- Sistema de archivos **jerárquico**
- Archivos locales o **en red transparentes** (NFS=Network File System)
- **Particiones** en el disco duro de tipo EXT3 y EXT4, diferentes de los estándares de Windows (i.e. NTFS, FAT32, etc.)
- Método lógico y flexible, sin restricciones sobre nombres, extensiones, tamaño, etc.
- Sistema de archivos **muy estable**!! No es necesario defragmentar, muchos usuarios pueden escribir/leer/borrar a la verz, etc.

Utilidades y aplicaciones

- Diferencias notables entre el LINUX y otros Sistemas Operativos
  - Fácil instalación de nuevos programas
  - La shell conoce dónde debe buscar las órdenes
- Multitud de utilidades y aplicaciones
  - Edición y procesamiento de texto (inc. LaTEX)
  - Paquetes matemáticos y de representación gráfica de datos
  - Lenguajes y entornos de programación: Fortran, C, C++, Python, etc.
  - Bases de datos
  - Hojas de cálculo
  - Comunicaciones electrónicas y para redes
  - Herramientas de diseño gráfico
  - Navegación y edición web
  - Etc.

Distribuciones de LINUX

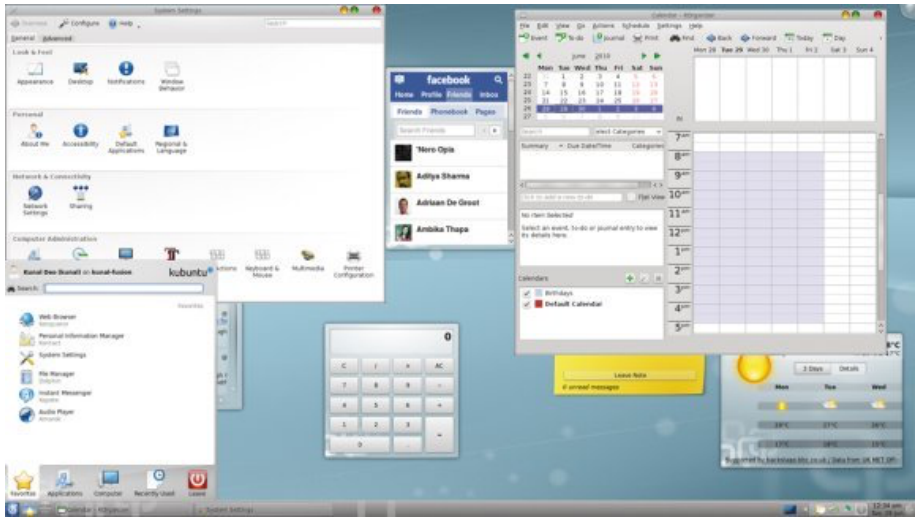


- Distribución de software basado en el núcleo Linux que incluye determinados programas para satisfacer las necesidades de un grupo específico de usuarios

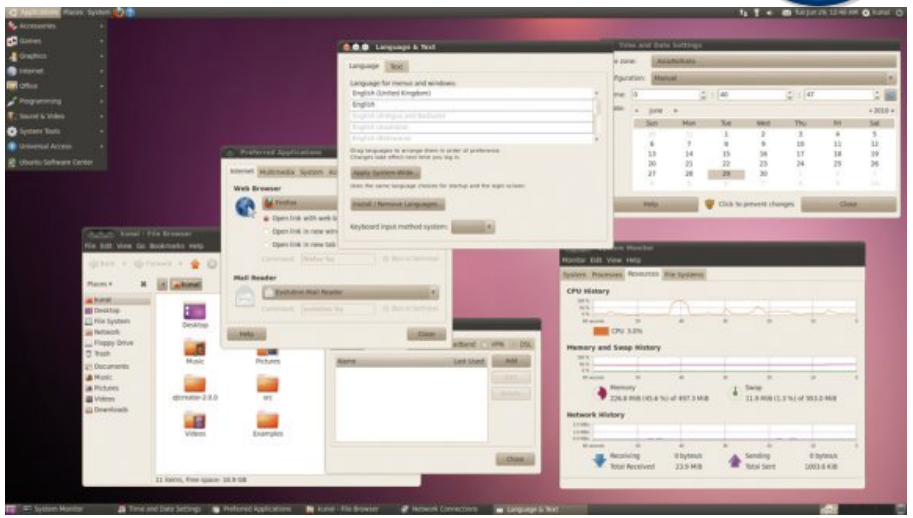
- Algunas distribuciones populares:
  - OpenSUSE
  - Ubuntu
  - Debian
  - Red Hat
  - Arch Linux, Steam OS, elementary OS, Scientific Linux, etc.

Entornos gráficos

- Interfaz gráfica de usuario que ofrece facilidades de acceso y configuración
- Los más populares hoy son **KDE** y **GNOME**



KDE



GNOME

Usuarios en LINUX

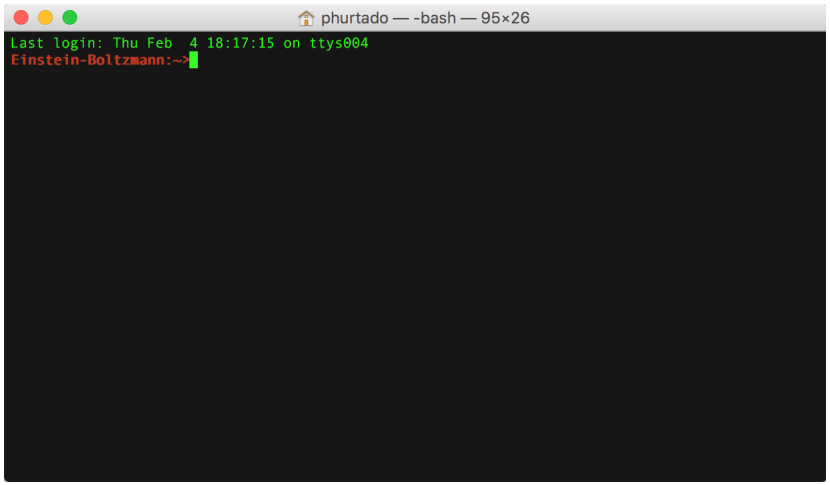
- Identificados por un **nombre de usuario** (hasta 8 caracteres)
- Tienen asociado un **número UID (User ID)**
- ***root*** es el nombre del superusuario, UID=0
- ***root*** tiene todos los privilegios
- Existen usuarios propios del sistema que no pueden ser utilizados por usuarios externos
- Física computacional: usuarios **cphys-*apellido***

Acceso a LINUX

- **Login**: Se introduce el nombre del usuario
- **Password**: Palabra secreta. El sistema sólo tiene en cuenta los 8 primeros caracteres tecleados.
  - Es aconsejable poner al menos unos 6 caracteres y que sea una palabra no usual, pues los hackers tienen métodos de búsqueda de passwords, y lo hacen por búsqueda en diccionarios junto con reglas sencillas de números.

Acceso a LINUX

- Tras la conexión abrimos una **terminal**



- Nos aparece un símbolo $ o ~/acto03:/> … Es el ***prompt***
- El prompt del superusuario o root es #

Algunos comandos sencillos

- ***who*** nos indica los usuarios que están conectados de forma interactiva con el ordenador.
- ***finger*** muestra una información más amplia sobre los usuarios conectados.
- ***whoami*** nos dice que usuario somos ahora
- ***hostname*** nos da el nombre de la máquina a la que estamos conectados
- ***passwd*** nos permite cambiar nuestro *password*

Ejercicio: cambiad ahora vuestro password
Usamos el comando ***passwd***

Algunos comandos sencillos

- ***ssh*** (secure shell) nos permite conectar de manera segura con una **máquina remota**
- **Ejemplo: *ssh cphys-XYZ@acto03***
- ***exit*** cierra la conexión y nos devuelve a la máquina de partida. También sirve para cerrar la terminal

Ejercicio: Conectar via ***ssh*** con otra máquina del aula. Comprobar usuario con ***whoami*** y máquina con ***hostname*****.** Salir usando ***exit***

Archivos

- Estructura básica para almacenar información
- Secuencias de bytes que se almacenan
- Posee un **nombre único** que lo identifica
- Pertenecen a un **propietario** y un **grupo**
- Tienen asociados un **conjunto de permisos: lectura, escritura, ejecución (r,w,x)**

Archivos

- **Reglas para el nombre**: El número máximo de caracteres, si existe, es muy alto
- Es aconsejable utilizar caracteres que no tengan un significado especial para la shell, para evitar confusiones. Ej: $, %, >, etc.
- Los nombres pueden acabar con **cualquier extensión**, o múltiples extensiones.

Archivos especiales

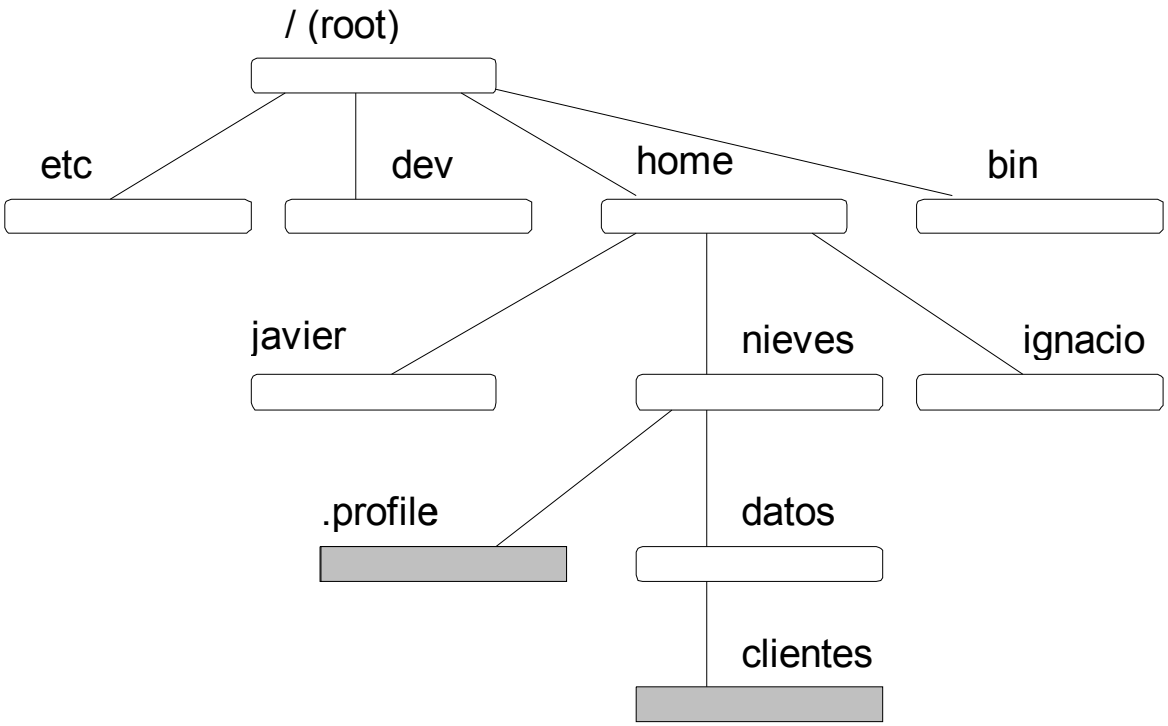
- Los **dispositivos** (cd, usb, hdd) se consideran como “archivos”.
- Se montan como tales en el árbol de directorios
- Se puede **leer y escribir** en un dispositivo como si fuera un archivo.
- Se puede transferir el contenido de un dispositivo a un fichero y viceversa (aunque no todos los dispositivos lo permiten).

Directorios

- Permiten agrupar ficheros
- Poseen una **estructura jerárquica**
- En principio no hay limitación del número de ficheros dentro de un directorio. Sólo estamos limitado por el espacio en disco

Estructura jerárquica de archivos

- Estructura de **árbol**:



Movimiento por archivos y directorios

- Se hace referencia a los nombres de directorio utilizando la / para separar niveles.

  /home/nieves/datos/mi_fichero

- El nombre de un archivo incluye su ***PATH*** completo. El *path* es el camino que lleva al archivo en el árbol de directorios.

- Si no se comienza con / entonces se entiende relativo al directorio actual.

  datos/mi_fichero

- OJO!! En Windows y MS-DOS se utiliza \.

El árbol de directorios de LINUX

- **/** Directorio raíz (inicio del árbol).
- **/home** Contiene los directorios de los usuarios.
- **/bin** Ordenes usuales y utilidades.
- **/usr** Programas, librerías y ficheros de uso normal
- **/dev** Dispositivos del sistema (realmente no contiene ficheros sino referencias a dispositivos)

El árbol de directorios de LINUX

- **/etc** Contiene ficheros de configuración.
- **/sbin** Contiene programas necesarios de inicio del sistema.
- **/tmp** Contiene ficheros temporales.
- **/var** Contiene ficheros de spool de datos, logs....
- **/proc** Información sobre el sistema.
- **/lib** Librerías de ejecución.

Mostrar directorio actual

- La orden que nos dice en cada momento la ruta completa de dónde nos encontramos es ***pwd*** (print working directory).

Ejercicio: averiguar el directorio en el que estás ahora mismo usando el comando ***pwd***

Información contenida en un directorio

- La orden ***ls*** es bastante parecida a la orden DIR de MSDOS.
- ***ls*** nos muestra los archivos del directorio actual.
- Podemos especificar un nombre de directorio o caracteres y comodines (como el *) para seleccionar archivos. Ejemplo:

  *ls fich**

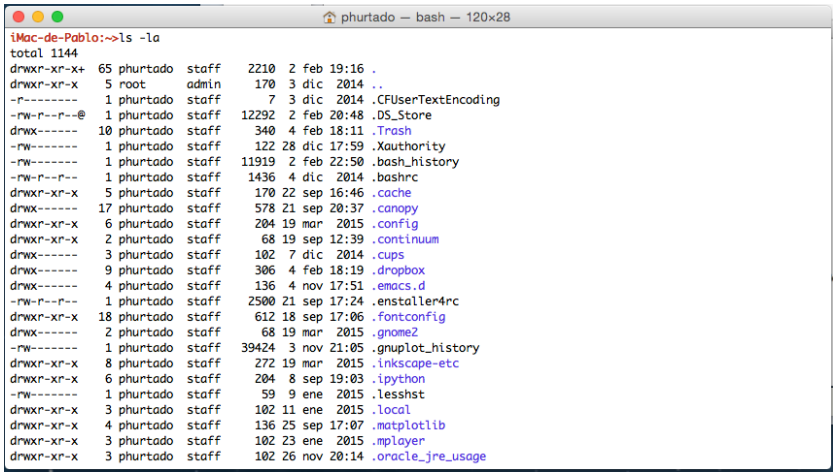
  *ls *.exe*

  *ls *penta**

Información contenida en un directorio

- ***ls –l*** nos muestra información extendida sobre los archivos.
- ***ls –a*** muestra todos los ficheros (también los ocultos), pues aquellos que comienzan por . no aparecen con ls
- Podemos combinar opciones: ***ls –la*** muestra todos los archivos junto con su tamaño, usuario y grupo, fecha de modificación, permisos y número de enlaces "hard"
- ***ls –d*** muestra los directorios.
- ***ls –R*** muestra el directorio actual y los subdirectorios

Ejercicio: usar ***ls -R*** para explorar la estructura de mi home



Directorios

- Comando para crear un directorio

  mkdir nombre_dir

- Podemos crear el directorio en cualquier lugar del árbol de directorios dando el **path**
- Para entrar en un directorio, ***cd nombre_dir***
- Debemos tener **permiso** para poder acceder a dicho directorio, de lo contrario se rechaza.

Ejercicio: crear con ***mkdir*** un directorio local y otro dando un path. Moverse entre directorios con ***cd***.

ATENCIÓN: presionando **TAB** se completa el nombre de un archivo o directorio que ya exista

Cómo eliminar un directorio

- Para borrar un directorio, escribimos

  rmdir nombre_directorio

- También se pueden eliminar múltiples directorios y utilizar comodines

- Un directorio se borra si está totalmente vacío

Ejercicio: eliminar los directorios creados anteriormente usando ***rmdir***.

Crear y editar ficheros

- Podemos crear ficheros de texto con un editor preferido
- Editores para la línea de comandos:
  - vi, vim
  - pico, nano
- Editores para entorno gráfico:
  - emacs, xemacs
  - kate, kwrite
  - gedit

Ejercicio: crear dos ficheros arbitrarios, uno usando ***vi*** (o similar) y otro usando ***emacs*** o equivalente

El editor vi

vi basic commands

Summary of most useful commands

Entering command mode

`[Esc]` Exit editing mode. Keyboard keys now interpreted as commands.

Moving the cursor

`h` (or left arrow key) move the cursor left.
`l` (or right arrow key) move the cursor right.
`j` (or down arrow key) move the cursor down.
`k` (or up arrow key) move the cursor up.
`[Ctrl] f` move the cursor one page **f**orward .
`[Ctrl] b` move the cursor one page **b**ackward.
`^` move cursor to the first non-white character in the current line.
`$` move the cursor to the end of the current line.
`G` **g**o to the last line in the file.
n`G` **g**o to line number *n*.
`[Ctrl] G` display the name of the current file and the cursor position in it.

Entering editing mode

`i` **i**nsert new text before the cursor.
`a` **a**ppend new text after the cursor.
`o` start to edit a new line after the current one.
`O` start to edit a new line before the current one.

Replacing characters, lines and words

`r` **r**eplace the current character (does not enter edit mode).
`s` enter edit mode and **s**ubstitute the current character by several ones.
`cw` enter edit mode and **c**hange the **w**ord after the cursor.
`C` enter edit mode and **c**hange the rest of the line after the cursor.

Copying and pasting

`yy` copy (**y**ank) the current line to the copy/paste buffer.
`p` **p**aste the copy/paste buffer after the current line.
`P` **P**aste the copy/paste buffer before the current line.

Deleting characters, words and lines

All deleted characters, words and lines are copied to the copy/paste buffer.

`x` delete the character at the cursor location.
`dw` **d**elete the current **w**ord.
`D` **d**elete the remainder of the line after the cursor.
`dd` **d**elete the current line.

Repeating commands

`.` repeat the last insertion, replacement or delete command.

Looking for strings

`/`*string* find the first occurrence of *string after the cursor.*
`?`*string* find the first occurrence of *string* before the cursor.
`n` find the **n**ext occurrence in the last search.

Replacing strings

Can also be done manually, searching and replacing once, and then using `n` (next occurrence) and `.` (repeat last edit).

`:`*n*`,`*p*`s/`*str1*`/`*str2*`/g` between line numbers *n* and *p*, **s**ubstitute all (**g**: global) occurrences of *str1* by *str2*.
`:1,$s/`*str1*`/`*str2*`/g` in the whole file (`$`: last line), **s**ubstitute all occurrences of *str1* by *str2*.

Applying a command several times - Examples

`5j` move the cursor 5 lines down.
`30dd` **d**elete 30 lines.
`4cw` **c**hange 4 **w**ords from the cursor.
`1G` **g**o to the first line in the file.

Misc

`[Ctrl] l` redraw the screen.
`J` **j**oin the current line with the next one
`u` **u**ndo the last action

Exiting and saving

`ZZ` save current file and exit vi.
`:w` **w**rite (save) to the current file.
`:w` *file* **w**rite (save) to the *file* file.
`:q!` **q**uit vi without saving changes.

Going further

`vi` has much more flexibility and many more commands for power users!
It can make you extremely productive in editing and creating text.

Learn more by taking the quick tutorial: just type `vimtutor`.

Visualización de un fichero

- ***cat nombre_fichero***
- También muestra varios ficheros, uno tras de otro si se especifican varios nombres.
- Ctrl-S congela la salida. Ctrl-Q la restablece. Ctrl-C cancela la salida.
- ***more nombre_fichero*** vuelca un fichero a la salida estándar de la terminal, **página a página**
- ***head –10 nombre_fichero*** muestra las 10 primeras líneas de un fichero.
- ***tail –100 nombre_fichero*** muestra las 100 últimas líneas de un fichero.

Ejercicio: probar estos comandos

Copiar ficheros

- ***cp nombre_original nuevo_fichero***
- Podemos realizar copias recursivas con la orden:
  - ***cp –r nombre_directorio1 nombre_directorio2***
  - Copiará los archivos y los subdirectorios
- ***cp origen1 origen2 destino***
  - Coge los dos ficheros y los copia al destino

Ejercicio: probar estos comandos

- Algunas otras **opciones**:
  - ***-d*** Copia los **enlaces simbólicos** (por defecto se copia el contenido del original)
  - ***-p*** Preserva intactos propietario, grupo, permisos y fechas
- Se pueden **concatenar opciones**: ***cp -dpr dir_origen dir_destino***
- **Ayuda** sobre un comando: ***man cp***

Mover ficheros y directorios

- ***mv antiguo_nombre nuevo_nombre*** (mv=move)
- Si movemos archivos entre sistemas de ficheros diferentes, automáticamente se realiza una copia física para trasladar los datos, y después, borra los originales.
- En el mismo sistema de ficheros sólo se cambia el nombre, no se desplazan los datos.
- Permite cambiar **archivos y directorios**.

Ejercicio: probar estos comandos

Cómo borrar ficheros

- Borrar uno o múltiples ficheros con ***rm*** (rm=remove)
- También se puede borrar un directorio con todo su contenido mediante un **borrado recursivo**.

rm –r nombre_directorio

- **¡Cuidado! Lo borrado NO se puede recuperar**

Creamos un alias para rm

- Para evitar borrar accidentalmente archivos o directorios, **creamos un *alias* para el comando *rm***
- Para hacer esto, editamos el archivo ***.bashrc*** que hay en nuestro **home** (en algunos casos, ***.bash_profile***)
- Añadimos la siguiente línea

alias rm="rm -i"

- Guardamos, salimos, y ejecutamos en la terminal

source .bashrc

- De esta manera, cada vez que queramos borrar algo, el sistema nos pregunta si estamos seguros …

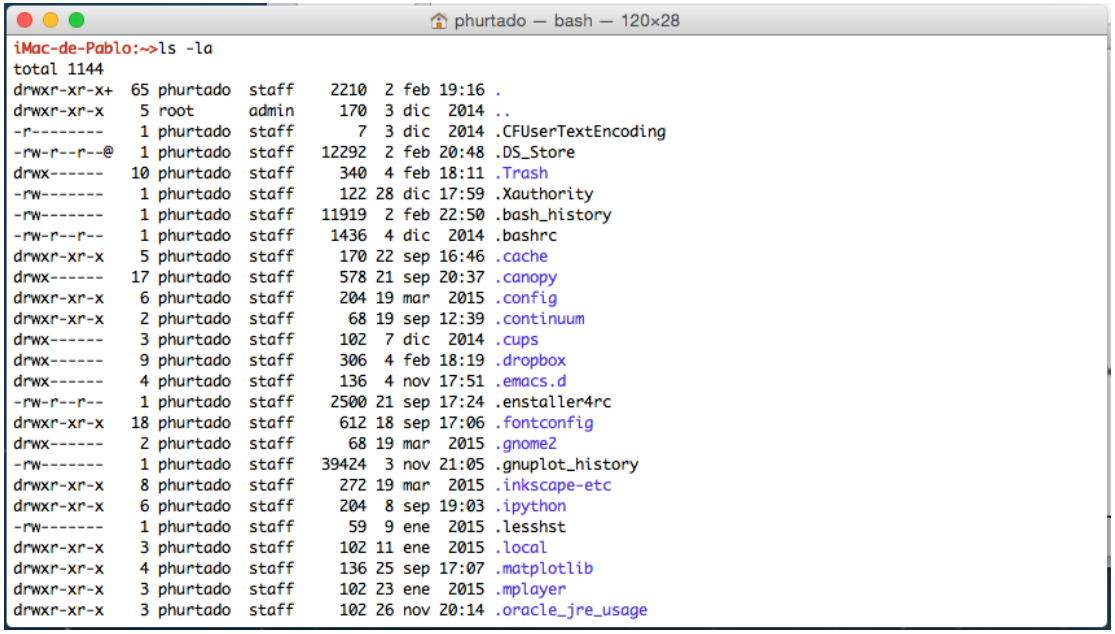
Conceptos: directorio actual, padre y home

- Directorio actual .
- Directorio padre ..
- Directorio home ~
- Ejemplos:
  - ***cp fich1 ..***
  - ***cp fich1 ~***
  - ***cp fich2 ~/datos***

Ejercicio: probar estos comandos

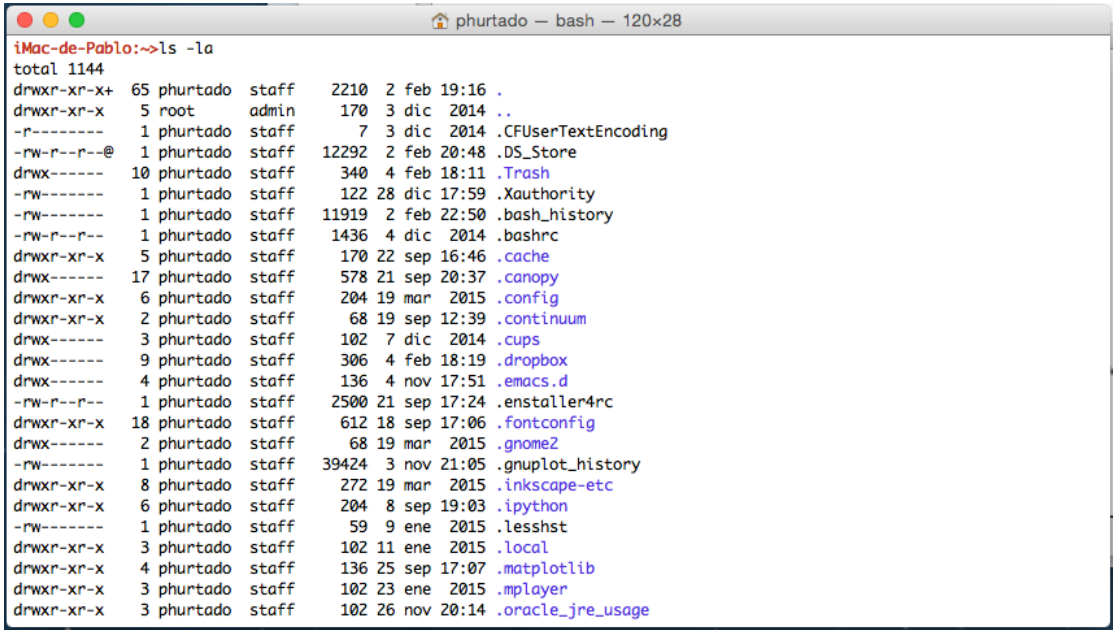
Permisos de ficheros

- Usuarios
  - Del propietario (u)
  - Del grupo (g)
  - Otros (o)
- Propiedad
  - Lectura (r)
  - Escritura (w)
  - Ejecución (x)



Permisos de directorios

- Propiedad
  - Lectura de ficheros (r)
  - Crear, borrar y modificar archivos (w)
  - Ver el contenido del directorio (x)



Cómo cambiar permisos

- Se puede utilizar **+** y **–** para conceder o denegar permisos.
- La orden Linux para cambiar permisos es ***chmod***, a la que hay que añadir una serie de parámetros.
- Ejemplo: ***chmod ug+x fichero_1 datos****
- Da permiso de ejecución (**+x**) para usuario y grupo (***ug***) de los ficheros indicados

Ejercicio: crea un **fichero** y un **directorio**, y da/quita permisos (r,w,x) comprobando que sucede en cada caso

Comodines

- Los símbolos especiales **?** ***** actúan como comodines para uno o múltiples caracteres.
- Podemos especificar un conjunto de caracteres válidos ***[abz]*** o rangos ***[a-m]*** o excluir rangos ***[^a-m]***
- Algunos ejemplos
  - ***ls [a-m]****
  - ***cp [ab]* /home/usuario***
  - ***rm c[^0-4]****

Ejercicio: probar estos comandos

Redirección de la entrada y salida estándar

- Es posible redireccionar la entrada y salida por pantalla de muchos programas a otro alternativo.
- ***comando < archivo_de_entrada***: El contenido del fichero se dirige al comando
- ***comando > fichero_salida***: redirecciona la salida de un comando a un fichero. **Ejemplo**: ***ls –la > salida***
- **Añadir al final** de un fichero: ***comando >> fichero_salida***

  ls b* >>salida
- Combinar entrada y salida: ***sort < fichero_random > salida***

Ejercicio: ordena los **números aleatorios** de un fichero y guárdalos en otro. ¿Qué diferencia hay entre ***sort*** y ***sort -g***?

INCISO: Scripts y la shell

- Un **script** es un programa que se ejecuta bajo la shell. Sirven para automatizar multitud de tareas
- Ejemplos de **scripts**

```
for i in {1..10}
    do
        echo $RANDOM
    done
```

```
for i in {1..10}
    do
        echo $i
        if [ "$i" == "1" ] ; then
            echo "Bienvenido $i vez!"
        else
            echo "Bienvenido $i veces!"
        fi
    done
echo "Hecho"
```

- Guardamos el archivo con **extensión ".sh"**, por ejemplo "**script.sh**", damos permiso de ejecución, i.e. "***chmod +x script.sh***", y **¡*voilá*!**
- ***./script.sh > file*** guarda la salida de script.sh en un archivo llamado file

Salida de errores

- La shell permite redirigir las salidas correspondientes a errores a una salida distinta de la salida estándar con **2>**
- ***rm prueba 2> errores***: Si esta orden provoca un error (porque el fichero no exista o no haya permiso) dicho mensaje se enviará al fichero errores en lugar de la pantalla.
- ***/dev/null*** es una especie de papelera. Actúa como un fichero que siempre está vacío.
- De forma que las salidas que no deseamos que aparezcan por pantalla o en un fichero se pueden enviar a dicho archivo.
- Ejemplo: ***rm datos 2>/dev/null***

Encauzamiento o *pipes*

- Con el ***pipe*** **|** es posible redireccionar la salida estándar de un programa a la entrada de otro programa.
  - ***ls c**** **|** ***more***: La salida de ls se transfiere a more y éste la va mostrando página a página.
  - ***sistema_solar.exe*** **|** ***PlotAtoms.exe***: las coordenadas de cada planeta se envían a un programa que representa gráficamente la dinámica del sistema
- **Otros ejemplos**

ls -la |grep cphys-XYZ

ps aux |grep cphys-XYZ

- El comando ***grep*** permite buscar patrones alfanuméricos en la información de entrada
- La **diferencia entre el pipe | y los redirecionadores > o <** es que estos últimos trabajan con **archivos intermedios**, mientras que | envía los datos de un comando a otro

Procesos

- Cada programa que ejecuta el ordenador es un **proceso**.
- **Multitarea**: El S.O. puede ejecutar varios procesos asignando pequeñas fracciones de tiempo a cada uno de forma que parece que todos funcionan simultáneamente.
- Algunos procesos pueden estar "**congelados**" de forma que se le dedica el tiempo a aquellos procesos realmente prioritarios.

Procesos

- Un proceso puede "crear" otro proceso.
- Relación **padre-hijo.** Si un proceso "padre" muere, también desaparecerán sus procesos hijos.
- Esto puede evitarse con ***nohup comando &*** **.** En este caso es el "abuelo" que hará las veces de padre.
- ***Init*** es el proceso padre de todos.
- A cada proceso se le asigna un número (***PID*** - process ID)
- Un ***daemon*** (demonio) es un proceso residente que está a la espera de realizar alguna función.
  - Ej: ***lpd*** es el *daemon* de impresión.

Ejercicio: lanza ***emacs*** desde terminal y ciérrala. ¿Qué pasa?

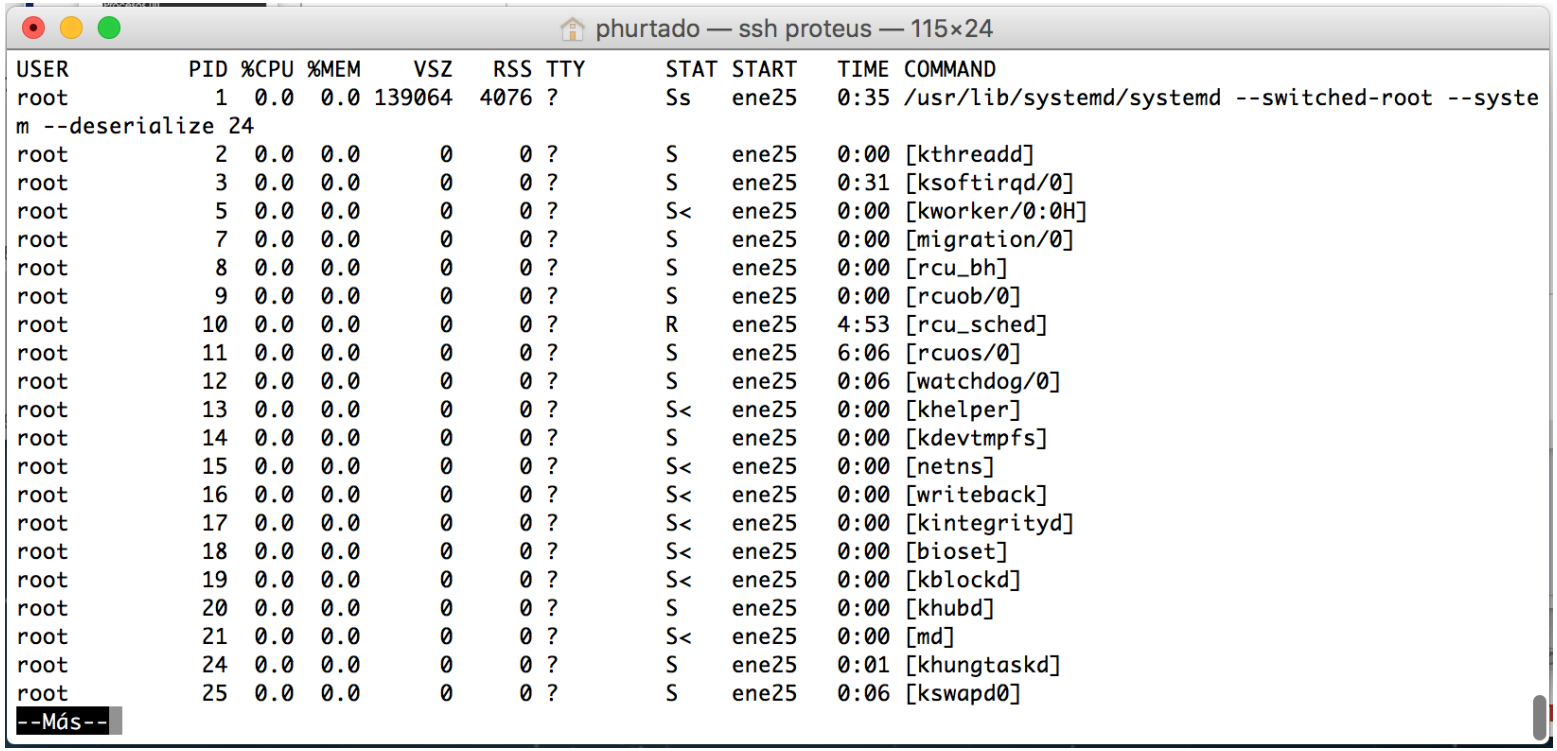
Planificación del procesos

- Estados posibles de un proceso:
  - En ejecución (**Running**)
  - Dormidos (**Sleeping**)
  - En espera de Entrada/Salida (**Waiting**)
  - Zombies (**Z**)
- El S.O. planifica en función de:
  - La prioridad del proceso
  - Los requisitos de CPU en instantes anteriores
  - Si se pueden suspender un proceso por procesos de espera.
  - Si se deben atender interrupciones de periféricos (de disco, red local, puertos serie,...)

Información de procesos

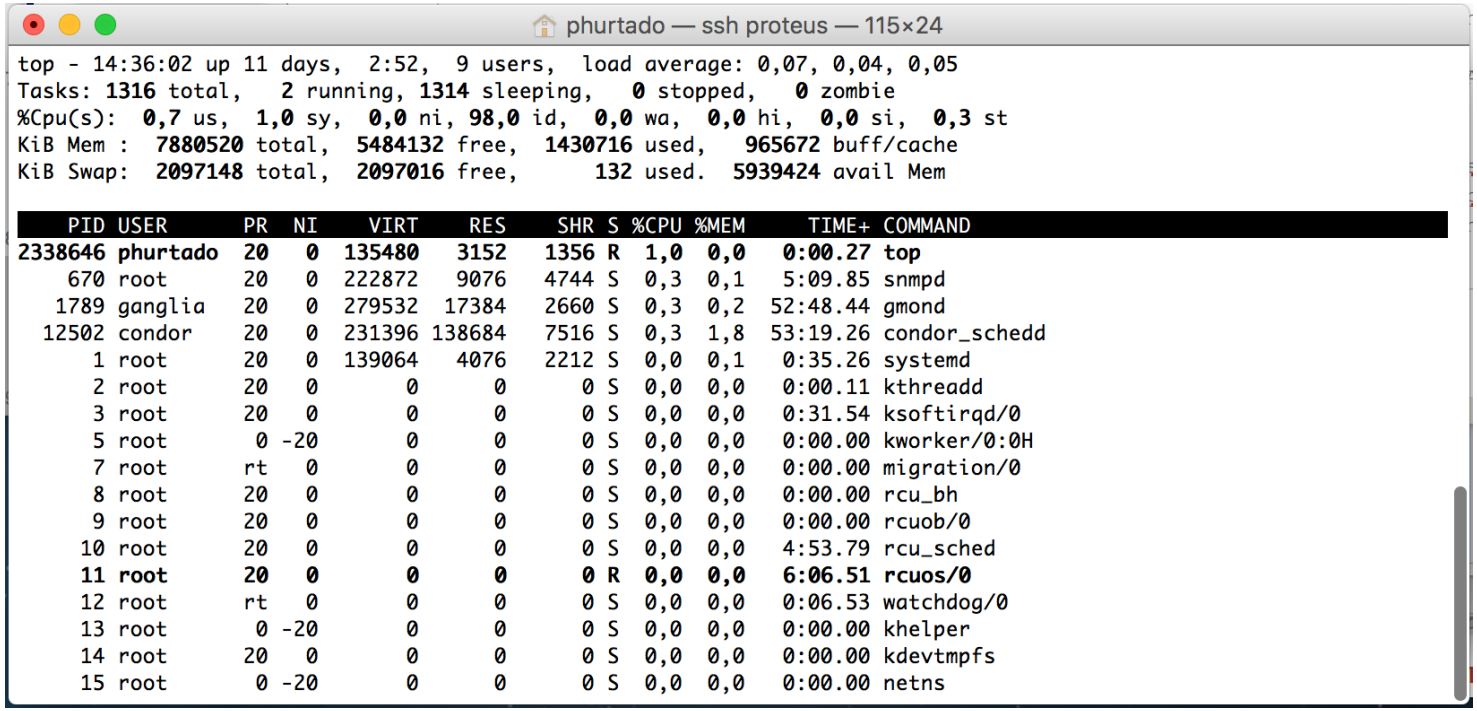
- Para saber qué procesos corren en mi máquina:

ps aux |more



Información de procesos

- ***top*** muestra información de los procesos de forma que se actualiza periódicamente. Aparecen ordenados en función del % de consumo de CPU.



Señales a procesos

- Es un pequeño mensaje de un proceso a otro.
- La señales en realidad son números del **0 al 30**.
- Cada número representa una señal, que entienden muchos procesos. El receptor puede ignorar la señal o atenderla.
- ***kill*** envía una señal a un proceso. Cualquier proceso que reciba la señal 9 (SIGKILL) entiende que debe desaparecer

  kill -9 PID

- Otra señal útil es la ***1*** (SIGHUP) pues suele utilizarse en muchos *daemons* para actualizar sus tablas
  - Ejemplos: ***kill –1 389*** ó ***kill –SIGHUP 389***

Ejercicio: lanza ***emacs*** busca su **PID**, y mata el proceso con ***kill -9 PID***

Variables de la Shell

- Las shell poseen variables que se pueden utilizar para:
  - Configuración personal de nuestro entorno de trabajo.
  - Información.
  - Transferir parámetros entre un proceso padre y otro hijo.
- ***Set*** permite mostrar las variables y asignar valores.

  set VARIABLE=valor
- Para que un hijo herede una variable del padre, previamente el padre debe **exportarla**.
- El comando ***echo*** nos permite ver el valor actual de una variable: Ejemplo: ***echo $USER***

Ejercicio: ejecuta en la terminal el comando ***echo $USER*** ¿Qué comando Linux es equivalente a esta orden?

Variables del Shell

- Algunas **variables** de la shell:
  - ***$HOME***: indica el directorio "home" del usuario.
  - ***$PATH***: directorios donde el sistema busca un comando
  - ***$TERM***: indicamos que tipo terminal
  - ***$USER***: nombre del usuario (login)
  - ***$UID***: número de identificación del usuario
  - ***$PS1***: prompt del sistema
- Con la orden ***unset variable*** eliminamos la variable.

Ejercicio: Usando ***echo***, comprobar estas variables

Ejercicio: Añadir el **directorio actual (.)** al **PATH** del sistema. Para eso editamos el archivo **.bashrc** de nuestro home

Ejecución de órdenes en modo subordinado y control de trabajos

- Linux permite ejecutar procesos de forma que trabajen en un **segundo plano**.
- Estos procesos seguirán en ejecución aunque el usuario que los lanzó cierre su sesión en la máquina
- **Esencial para ejecutar simulaciones** que duren un tiempo largo
- ***comando &*** : lanza el proceso en *background*. Ejemplo: ***sort < entrada > salida &***
- Para **enviar un proceso al *background***, presionamos ***CTRL+z*** y escribimos ***bg*** (background). También lo podemos hacer escribiendo ***bg PID***. La orden ***fg PID*** reactiva un proceso (*fg=foreground*)

Ejercicio: ejecutar ***emacs&*** en la terminal. Compararlo con ***emacs***. Enviarlo al *background*

Ficheros de configuración de sh

- La shell puede ejecutar ciertos comandos automáticamente al iniciar una sesión de usuario
- ***/etc/profile***: definido por root, se ejecutará siempre (se definen variables como TERM,...)
- ***.profile***: en nuestro HOME podemos editarlo y modificarlo
- ***.bashrc***: modifica los valores de la shell bash
- ***.bash_profile***: similar al anterior

Búsqueda de archivos

- El comando find sirve para encontrar archivos en el sistema

find directorio –opciones criterios

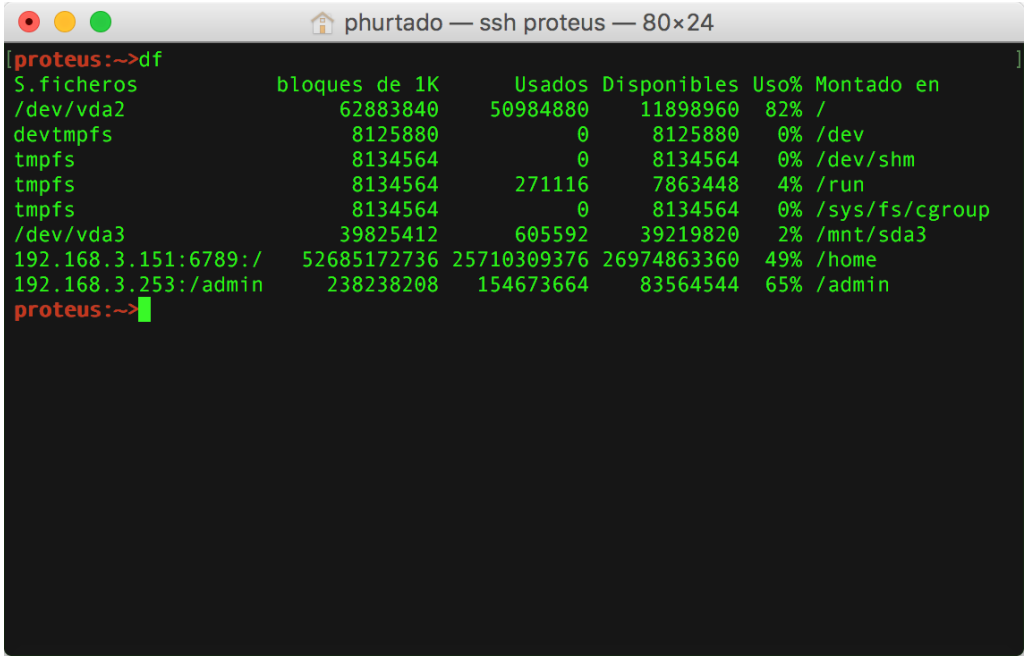
- Entre las opciones más usuales tenemos
  - ***name***: patrón de búsqueda del nombre
  - ***print***: indica que se muestre el nombre
- **Ejemplos**

find / -name "pas*" –print

find .-name datos –print

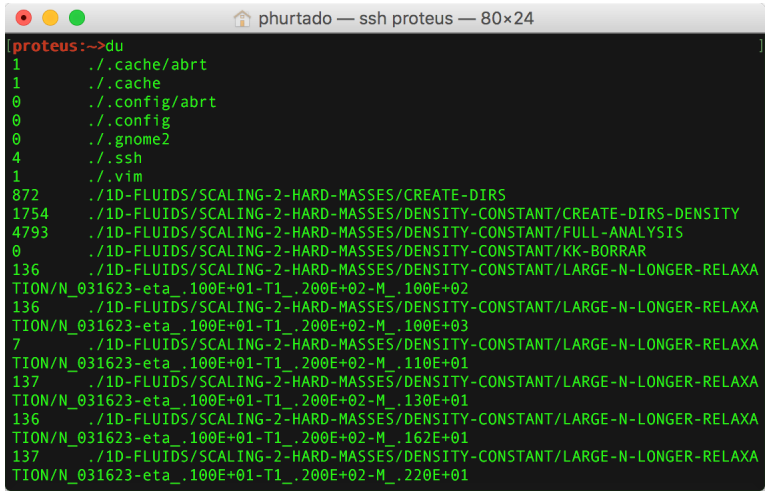
Disco disponible

- Linux almacena toda la información en sistemas de archivos o *filesystems*
- El comando ***df*** muestra la información de cada sistema de archivos (sean locales o remotos)



Disco utilizado

- El comando ***du*** nos informa del espacio total ocupado por un conjunto de ficheros o subdirectorios.



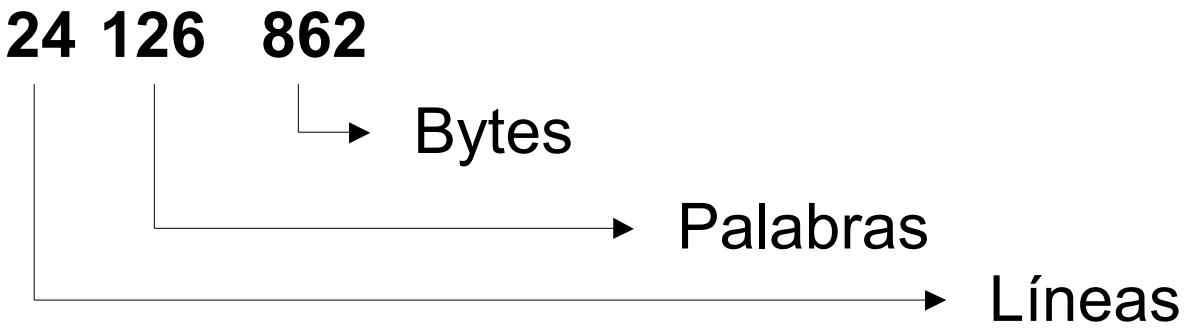
`Desglosa por defecto el tamaño de cada subdirectorio en bytes. Al final indica el tamaño total`

Ejercicio: Usar ***df***, ***du -m*** y ***quota*** en nuestro home

- La opción ***–s*** muestra el total sin desglosar los subdirectorios. Con ***-m*** nos da el resultado en **Mb**
- **ATENCIÓN**: los usuarios del **Aula de Física Estadística y Computacional** tienen una **cuota de disco duro**. Para averiguar cual es, ejecutar la orden ***quota***.

Contar elementos en ficheros

- El comando ***wc*** (*word count*) nos permite **contar elementos en un fichero**.
- Ejemplo con un fichero ***hosts***: ***wc hosts***



- ***wc –c*** sólo cuenta caracteres
- ***wc –l*** sólo cuenta líneas
- ***wc –w*** sólo cuenta palabras

Almacenamiento

- El comando ***tar*** permite empaquetar en un solo fichero varios ficheros diferentes

  tar -opciones lista

- Esto es muy práctico para procesarlos (envío, copias de seguridad, etc.)
- **Opciones** usuales:
  - **-x**: extraer
  - **-c**: crear
  - **-v**: ver los ficheros que se procesan
  - **-f**: indicar el nombre del fichero
  - **-z**: comprimir
- Se pueden controlar tamaños de bloque, permisos, etc.
- Para **comprimir o descomprimir** un archivo, podemos usar los comando ***gzip*** y ***gunzip***

Almacenamiento

- Es aconsejable que el fichero empaquetado tenga extensión .tar para recordar el formato

  tar -cvf fich.tar c*

- Algunos **ejemplos**:
  - ***tar -cvf fich.tar .*** : almacenamiento relativo
  - ***tar -cvf fich.tar /home/paco*** : almac. absoluto
  - ***tar -xcf fich.tar*** : extrae fichero
  - ***tar -tvf fich.tar*** : muestra el contenido

Ejercicio: Crear dos ficheros con nuestro editor preferido, y empaquetarlos y comprimirlos usando ***tar***

Tareas periódicas con *cron*

- El comando ***crontab*** nos permite definir tareas periódicas
- ***crond*** es el daemon de control de tareas
- ***crontab -e*** nos permite editar la tareas periódicas (por defecto usa el editor vi)

Ejecuta *copiasemanal.sh* cada domingo a las 23:55

Linux Command Cheat Sheet

Share This Cheat Sheet

Basic commands

|	**Pipe (redirect) output**
sudo [command]	run < command> in superuser mode
nohup [command]	**run < command> immune to hangup signal**
man [command]	display help pages of < command>
[command] &	**run < command> and send task to background**
>> [fileA]	append to fileA, preserving existing contents
> [fileA]	**output to fileA, overwriting contents**
echo -n	display a line of text
xargs	**build command line from previous output**
1>2&	Redirect stdout to stderr
fg %N	**go to task N**
jobs	list task
ctrl-z	**suspend current task**

File permission

chmod -c -R	**chmod file read, write and executable permission**
touch -a -t	modify (or create) file timestamp
chown -c -R	**change file ownership**
chgrp -c -R	change file group permission
touch -a -t	**modify (or create) file timestamp**

Network

netstat -r -v	**print network information, routing and connections**
telnet	user interface to the TELNET protocol
tcpdump	**dump network traffic**
ssh -i	openSSH client
ping -c	**print routing packet trace to host network**

File management

find	**search for a file**
ls -a -C -h	list content of directory
rm -r -f	**remove files and directory**
locate -i	find file, using updatedb(8) database
cp -a -R -i	**copy files or directory**
du -s	disk usage
file -b -i	**identify the file type**
mv -f -i	move files or directory
grep, egrep, fgrep -i -v	**print lines matching pattern**

File compression

tar xvfz	**create or extract .tar or .tgz files**
gzip, gunzip, zcat	create, extract or view .gz files
uuencode, uudecode	**create or extract .Z files**
zip, unzip -v	create or extract .ZIP files
rpm	**create or extract .rpm files**
bzip2, bunzip2	create or extract .bz2 files
rar	**create or extract .rar files**

File Editor

ex	**basic editor**
vi	visual editor
nano	**pico clone**
view	view file only
emacs	**extensible, customizable editor**
sublime	yet another text editor
sed	**stream editor**
pico	simple editor

Directory Utilities

mkdir	**create a directory**
rmdir	remove a directory

File Utilities

tr -d	**translate or delete character**
uniq -c -u	report or omit repeated lines
split -l	**split file into pieces**
wc -w	print newline, word, and byte counts for each file
head -n	**output the first part of files**
cut -s	remove section from file
diff -q	**file compare, line by line**
join -i	join lines of two files on a common field
more, less	**view file content, one page at a time**
sort -n	sort lines in text file
comm -3	**compare two sorted files, line by line**
cat -s	concatenate files to the standard output
tail -f	**output last part of the file**

Scripting

awk, gawk	**pattern scanning**
tsh	tiny shell
" "	**anything within double quotes is unchanged except \ and $**
' '	anything within single quote is unchanged
python	**"object-oriented programming language"**
bash	GNU bourne-again SHell
ksh	**korn shell**
php	general-purpose scripting language
csh, tcsh	**C shell**
perl	Practical Extraction and Report Language
source [file]	**load any functions file into the current shell, requires the file to be executable**

Memory & Processes

free -m	**display free and used system memory**
killall	stop all process by name
sensors	**CPU temperature**
top	display current processes, real time monitoring
kill -1 -9	**send signal to process**
service [start|stop|restart]	manage or run sysV init script
ps aux	**display current processes, snapshot**
dmesg -k	display system messages

Disk Utilities

df -h, -i	**File system usage**
mkfs -t -V	create file system
resize2fs	**update a filesystem, after lvextend***
fsck -A -N	file system check & repair
pvcreate	**create physical volume**
mount -a -t	mount a filesystem
fdisk -l	**edit disk partition**
lvcreate	create a logical volume
umount -f -v	**umount a filesystem**

Misc Commands

pwd -P	**print current working directory**
bc	high precision calculator
expr	**evaluate expression**
cal	print calender
export	**assign or remove environment variable**
` [command]	backquote, execute command
date -d	**print formatted date**
$[variable]	if set, access the variable

Read the Blog Post »
bit.ly/Linux-Commands

Compiled by Alvin Khoo