Arquitectura de sistemas

Abelardo Pardo

University of Sydney
School of Electrical and Information Engineering
NSW, 2006, Australia
Autor principal del curso de 2009 a 2012

Iria Estévez Ayres

Damaris Fuentes Lorenzo

Pablo Basanta Val

Pedro J. Muñoz Merino

Hugo A. Parada

Derick Leony

Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática
Avenida Universidad 30, E28911 Leganés (Madrid), España

Capítulo 11. El entorno de trabajo en Linux

Tabla de contenidos

Para la lectura de este documento se asume que tienes la máquina virtual ejecutando. Se recomienda probar las aplicaciones y comandos que se describen, pues es la mejor forma de familiarizarse con ellos. Una lectura sin la máquina ejecutando será mucho menos efectiva.

11.1. El escritorio

La pantalla inicial de trabajo es el "escritorio" porque se parece a una mesa de trabajo en la que nos acabamos de sentar. El área en la parte inferior se conoce como "el panel" y alberga una serie de iconos que sirven para mostrarnos información y para arrancar algunas aplicaciones.

La apariencia del escritorio es configurable en la mayor parte de sus aspectos. El panel se puede poner en cualquier borde de la pantalla, se puede esconder cuando no se usa, se puede cambiar el fondo de escritorio, añadir aplicaciones fijas en el escritorio que se llaman "widgets", etc. En Linux existen varios entornos gráficos. El que utilizamos es KDE, uno de los más populares junto con GNOME. La siguiente figura muestra el escritorio en la máquina virtual.

Es habitual que el texto de cada ventana se pueda seleccionar con el ratón o el teclado. Dado un texto seleccionado de una ventana, suele ser posible (dependiendo de la aplicación):

- Copiarlo
- Cortarlo (copiarlo + borrarlo)
- Pegarlo (en la misma ventana o en otra)

Cada programa tiene diferentes combinaciones de teclas para hacer estas tres operaciones. Copiar y pegar es un ejercicio que hace que nuestras sesiones de trabajo sean mucho más eficientes, pues se ahorra tiempo y se evitan los errores típicos al teclear. Lo probaremos más adelante con el editor de texto.

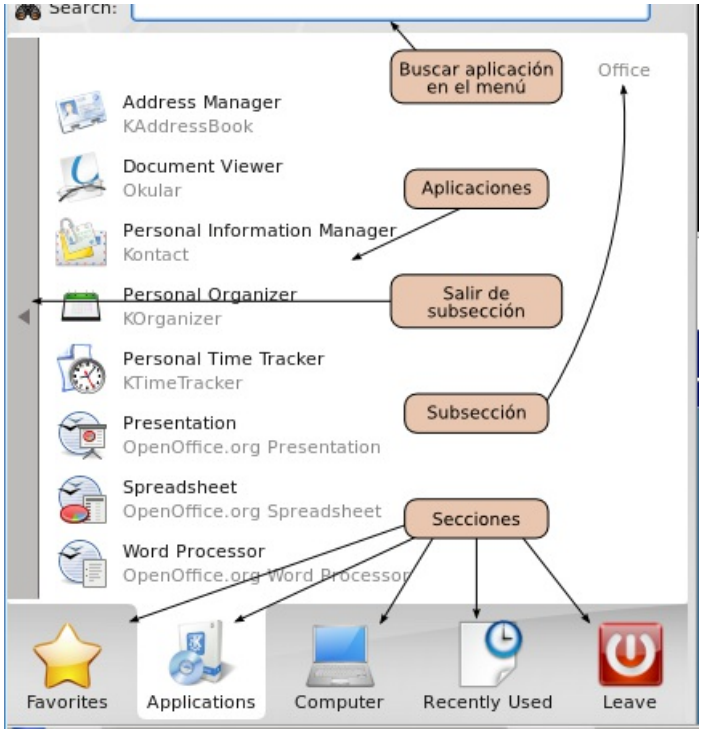
11.2. El panel

El panel es la barra en la parte inferior de la pantalla que contiene un conjunto de iconos que permiten acceso a las funcionalidades más comunes. En la siguiente figura se muestra el significado de estos iconos.

De estos iconos, los más relevantes son los siguiente:

- Terminal (programa **konsole**). Quizás el que más vamos a utilizar a lo largo del curso. Abre una ventana en la que se ejecuta un intérprete de comandos. Se describe en detalle en la siguiente sección.
- Editor (programa **kate**). Abre una ventana con el editor avanzado de texto **kate**. Utilizaremos esta aplicación al comienzo del curso para crear los programas en C.
- Escritorios virtuales. El entorno gráfico permite manipular varios escritorios de forma simultanea pero mostrando sólo uno en pantalla. Las aplicaciones se pueden abrir en cualquiera de estos escritorios y mover entre ellos. Puedes cambiar de escritorio pulsando en el icono del panel, o mediante la combinación de teclas **Ctrl-F1** y **Ctrl-F2** (para ir al escritorio 1 y 2 respectivamente). Tanto el número de escritorios disponibles como la combinación de teclas para acceder son configurables.
- Cambiar de resolución. Este icono ejecuta una aplicación para cambiar la resolución de la pantalla. Los valores que ofrece están restringidos por la capacidad gráfica de la tarjeta.
- Notificador general. Este icono muestra los mensajes del sistema sobre diferentes situaciones. Por ejemplo: la copia de un fichero ha terminado, un nuevo mensaje en el chat, la impresora ha terminado, etc. Los mensajes que se muestran son configurables.
- Notificador de dispositivos. Cuando se conecta un dispositivo USB, este icono nos muestra su nombre y la posibilidad de abrirlo con el gestor de ficheros o extraerlo de forma segura.

Los dos iconos en la parte izquierda del panel ofrecen acceso rápido a la carpeta accedida más frecuentemente, y al menú general de aplicaciones. Este menú organiza las aplicaciones en secciones y subsecciones. En la siguiente figura se muestra la subsección de aplicaciones de oficina que contiene aplicaciones para procesado Word, hojas de cálculo, presentaciones, etc.

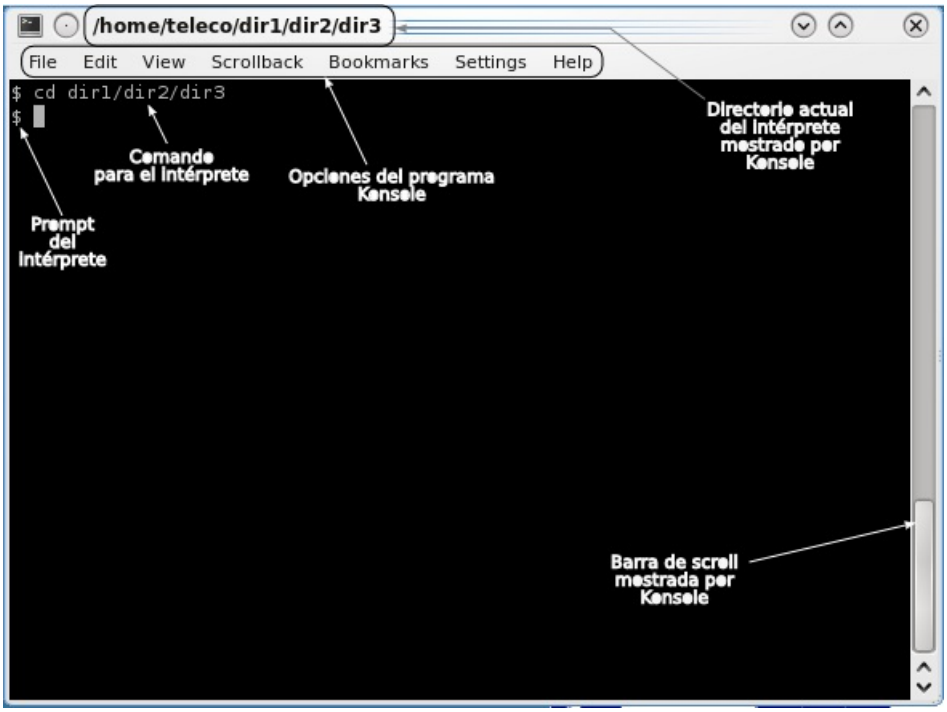


Es típico colocar las aplicaciones más utilizadas directamente como iconos en el panel (tal y como está el Gestor de Ficheros, Navegador, Editor y Terminal). Para añadir una aplicación hay que seleccionarla en el menú principal y a continuación seleccionar la opción de añadir al panel con el botón derecho del ratón.

11.3. Terminal de comandos

El terminal de comandos es una de las aplicación que vamos a utilizar cuando trabajemos sobre Linux. La siguiente figura muestra la ventana y sus partes más importantes.

Figura 11.1. Terminal de comandos

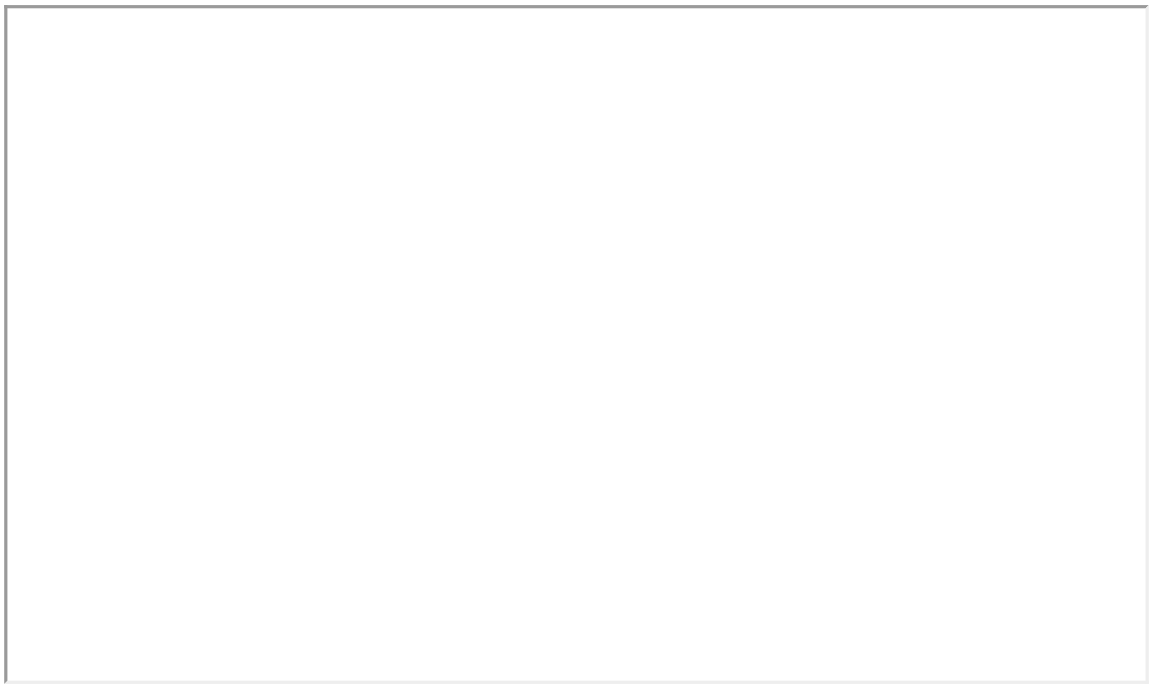


Esta figura muestra en realidad la ejecución de dos programas. El primero es **konsole** y se encarga de abrir una ventana en el escritorio como la que muestra la figura. Dentro de esa ventana se muestra el contenido de la ejecución de ***un segundo programa***, el intérprete de comandos (o "*shell*") **bash**. Los dos programas coexisten, pero cada elemento que se muestra en la figura corresponde a uno de los

programas. El intérprete de comandos **bash** es un programa que espera a que introduzcamos un comando y cuando lo terminamos con un salto de línea, lo ejecuta. Al terminar vuelve a quedarse a la espera del siguiente comando. Al texto "$" que muestra cuando está esperando por el siguiente comando se le llama "prompt". Los mensajes producidos por los comandos ejecutados se muestran en la pantalla. La barra de scroll de la izquierda la gestiona el programa **konsole** y permite acceder a mensajes de comandos anteriores que ya no estén visibles en la pantalla. **konsole**, por tanto, se encarga de almacenar estos mensajes hasta cierto número de líneas. Las opciones que aparecen en la parte superior de la ventana las ofrece **konsole** para configurar su comportamiento.

En la parte interna de la ventana se ve un comando BASH. El primer símbolo es el prompt seguido de un comando. En este caso el comando no ha producido ningún mensaje, y cuando ha terminado se muestra de nuevo el prompt. Este intérprete lo vamos a utilizar para casi todos los ejercicios de la asignatura. Como consecuencia, en la siguiente sección se explican los principales comandos de este intérprete.

En el siguiente video puedes ver una breve demostración de cómo manejar el intérprete de comandos.



11.3.1. Preguntas de autoevaluación

Responde a las siguientes preguntas

1. Qué mensajes muestra el notificador general?
   - Los mensajes del sistema sobre diferentes situaciones
   - El número de escritorios disponibles
2. Qué es el interprete de comandos bash?
   - Es una ventana en el escritorio
   - Un programa que espera a que introduzcamos un comando y cuando lo terminamos con un salto de línea, lo ejecuta

11.4. El intérprete de comandos bash

Al comienzo de su ejecución, el intérprete de programa nos muestra el prompt, y está listo para

ejecutar cualquier ***comando*** o ***programa*** en el ***directorio actual***. Esto nos lleva a responder las tres primeras preguntas sobre este programa.

1. ¿Cuántos comandos puede ejecutar BASH? Esto se puede responder utilizando el comando **man** del propio BASH. Escribe el comando **man bash** y se muestra una descripción muy detallada (¡y extensa!) de todo lo que es capaz de hacer BASH. Al ser tan grande el texto, se muestra "paginado". Puedes pasar a la siguiente página pulsando la barra espaciadora, y la letra "q" para abandonar (*quit*). El comando **man** ofrece información para prácticamente todos los comandos y programas disponibles en Linux.

   Pero volviendo a la pregunta inicial, el número de comandos ofrecidos por BASH es simplemente demasiado grande para memorizar. Por tanto, la estrategia para aprender a trabajar con BASH es buscar un comando cuando se necesite, y si se utiliza a menudo, se acabará recordando. En la red puedes encontrar numerosos documentos que listan los comandos más utilizados. Como muestra se ofrece la Guía de Referencia de Comandos Unix/Linux publicada por FOSSwire. Los comandos en BASH típicamente tienen nombres muy cortos, pues hay que escribirlos en el terminal, y a menudo son abreviaturas de otros nombres (algunos igualmente crípticos). Por ejemplo, el comando **man** es la abreviatura de "manual".

2. ¿Cuántos programas puede ejecutar BASH? En realidad internamente no se distinguen entre comandos y programas. Por tanto, cualquier programa que se cree nuevo se puede ejecutar en BASH de forma idéntica a cualquier comando. Esta es la razón por la que la lista de comandos en BASH no ha parado de crecer en los últimos años.

   Toda aplicación tiene un nombre, y para ejecutarla sólo hay que escribirlo en el intérprete de comandos. Prueba a ejecutar el gestor de ficheros con nombre **dolphin**, el editor de texto (**kate**), el navegador (**firefox**) o la propia terminal de comandos (**konsole**). Fíjate que hasta que no terminas la ejecución del programa no se vuelve a mostrar el prompt en el intérprete.

3. ¿Cómo sé cuál es el "directorio actual"? Pues como no puede ser de otra forma, BASH nos ofrece un comando para esto. El comando **pwd** muestra el directorio actual del intérprete por pantalla. El nombre es la abreviatura de "*print working directory*". Y para saber exactamente cómo funciona este comando no tenemos más que combinar los dos comandos que hemos visto hasta ahora en uno sólo: **man pwd**. Recuerda pulsar la barra espaciadora para pasar de página, "b" para ir hacia atrás, y "q" para abandonar el manual.

   El contenido (ficheros y subcarpetas) del directorio actual se puede mostrar por pantalla con el comando **ls** (abreviatura de "list"). Prueba a ejecutar este comando en el intérprete.

Siguiendo con la filosofía que hemos descrito anteriormente, el propio manual, como todo programa, ofrece una página en la que describe su comportamiento. ¿Qué comando crees que hay que introducir para verla?

11.4.1. Los argumentos de un comando

Aunque cada comando ejecuta una tarea muy concreta (como el comando **man**, que muestra el manual), hay múltiples aspectos de esa tarea que pueden ser modificados. Por ejemplo, el comando **man** puede mostrar aquellos comandos que tienen en su definición una palabra, o mostrar las páginas con diferentes codificaciones, o cambiar el tamaño de página que utiliza para mostrar el documento, etc. Estas variaciones sobre la funcionalidad de un comando se especifican mediante cadenas de texto a continuación del comando en lo que se denominan las "opciones", "parámetros" o "argumentos". La descripción de qué opciones ofrece cada comando y lo que hacen está incluida en su página de manual.

Los argumentos de un comando se dividen en dos tipos: aquellos objetos sobre los que hay que operar, y los argumentos para modificar su comportamiento. Los primeros se suelen incluir al final del comando separados por espacios (si hay más de uno). Los argumentos para modificar el comportamiento de un comando se suelen especificar precedidos del símbolo "-". Por ejemplo, el

comando **man** en lugar de buscar el nombre de un comando, puede buscar aquellos comandos en cuya descripción breve se encuentra una palabra dada. Este comportamiento se invoca con la opción **-k** seguido de la palabra clave, esto es:

```
$ man -k directorio
```

Prueba a ejecutar este comando. Verás que por pantalla aparece una línea por cada comando. Comprueba si aparece alguno de los que se han visto hasta ahora. En la página de manual del comando **ls** busca la opción **-a**. Ejecuta el comando **ls** con esa opción y comprueba que hace lo esperado.

11.4.2. Nombres de ficheros y rutas

Una vez sabemos el directorio actual en el que ejecutan los comandos, ¿cómo lo cambiamos? O más en general, ¿cómo se pueden referenciar ficheros y carpetas en el intérprete?

Linux organiza ficheros y carpetas en un estructura de árbol. Un fichero está siempre en una carpeta, y una carpeta puede contener ficheros y subcarpetas. La carpeta de más alto nivel en esta jerarquía en Linux siempre tiene el nombre "/". Cuidado con la inclinación de esta barra, pues "\" tiene otro significado diferente.

Todo fichero o carpeta tiene una ***ruta absoluta*** que es la secuencia de carpetas que se atraviesan desde la raíz separadas por el símbolo "/" y que termina con su propio nombre. Por ejemplo, `/dirA/dirB/dirC/fichero.txt` es la ruta absoluta del fichero `fichero.txt` almacenado en la carpeta `dirC` que a su vez está contenida en la carpeta `dirB`, a su vez contenida en la carpeta `dirA` almacenada en la raíz del sistema de ficheros. El programa **konsole** que abre la ventana con el intérprete está configurado para mostrar la ruta absoluta del directorio actual en la barra de título de la ventana tal y como se muestra en la figura 11.1.

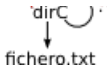
A todo usuario en Linux se le asigna una carpeta a partir de la cual puede almacenar sus ficheros. El intérprete permite abreviar su ruta con el símbolo "~". El comando **ls** sin ningún dato adicional muestra los ficheros en el directorio actual, pero si tras el nombre del comando se escribe una ruta, muestra los ficheros en esa ruta. Por tanto, el comando **ls ~** muestra los ficheros en la carpeta del usuario.

Además de las carpetas y ficheros creados por el usuario o los programas, toda carpeta Linux tiene dos subcarpetas definidas por defecto:

- "..": Es la carpeta del nivel superior en la jerarquía de ficheros que contiene a esta. Esta subcarpeta también esta presente en la raíz del sistema de ficheros (la carpeta con ruta absoluta "/"), pero apunta a si misma.
- ".": Es la propia carpeta que aparece como si fuese también una subcarpeta. Se puede considerar como una "autoreferencia" o apuntador a si misma.

Estas carpetas son a todos los efectos carpetas normales en el sistema; es decir, se pueden utilizar en las rutas. Por ejemplo, la ruta `/dirA/dirB/dirC/../fichero2.txt` se refiere al fichero que está almacenado en la carpeta `dirB`, pues desde `dirC`, la carpeta `..` apunta a su predecesor, `dirB`. La siguiente figura muestra la interpretación de esta ruta así como el equivalente de las carpetas ".." y ".".

Figura 11.2. Ruta absoluta que contiene la carpeta ".."



Como consecuencia de la existencia de las carpetas "." y "..", un fichero puede tener múltiples rutas absolutas equivalentes. Por ejemplo, las siguientes rutas se refieren todas al `fichero.txt` de la figura 11.2: `/dirA/dirB/dirC/../dirC/fichero.txt`, `/dirA/dirB/dirC/./././fichero.txt`, `/dirA/dirB/dirC/../../../dirA/dirB/dirC/./fichero.txt`, etc. Utiliza el comando **ls** para mostrar por pantalla el contenido de las carpetas de los niveles superiores del directorio actual hasta la carpeta raíz.

La rutas absolutas identifican de manera única cualquier fichero o carpeta en el sistema, pero pueden llegar a ser muy largas y por tanto costosas para escribir en un comando en el intérprete. La alternativa es utilizar ***rutas relativas***.

Una "ruta relativa" es una secuencia de nombres de carpetas separadas por "/" que no comienza por el símbolo "/" (si lo hiciera sería una ruta absoluta) y que se interpretan considerando como prefijo el directorio actual. Asumiendo que el directorio actual del intérprete es `dirC` la siguiente figura muestra la interpretación de diferentes rutas relativas. Nótese que ninguna de ellas comienza por el "/".

11.4.3. Comandos para gestionar ficheros

El intérprete ofrece los comandos para realizar las operaciones básicas sobre ficheros y el directorio actual. La funcionalidad detallada de cada uno de ellos la puedes acceder mediante el comando **man**. Los más comunes son:

- **ls** (list directory). Muestra los ficheros y directorios contenidos en el directorio actual, o el de la ruta que se escriba tras el nombre del comando. Su sintaxis general es **ls [opciones] [ruta]** siendo las "opciones" y la "ruta" parámetros opcionales. Algunos de las opciones más relevantes son "-l" para dar un listado con información detallada de los ficheros y directorios (modo, tamaño, número de enlaces, propietario, etc.), "-a" para incluir también ficheros y directorios ocultos en el listado, "-t" para dar el listado en un orden teniendo en cuenta la última modificación del fichero, o "-r" para revertir el orden del listado. Para practicar con este comando, desde la ruta inicial `/home/teleco`, liste en primer lugar todos los ficheros y directorios que hay allí, sin ninguna información adicional con **ls** . A continuación, liste todos los ficheros y directorios que hay en otra ruta diferente a la actual, poniendo la ruta absoluta, en concreto todo lo que hay debajo del directorio `/bin` , ejecutando **ls /bin** . Para conseguir no sólo un listado de dichos ficheros y directorios sino información precisa sobre ellos, ejecute **ls -l** y **ls -l /bin** respectivamente. Finalmente, ejecute **ls -lat** , **ls -lart** , **ls -lat /bin** y **ls -lart /bin** y observe las diferencias de ordenación de los ficheros.

- **cd** (change directory). Cambia el directorio actual por el que se escribe a continuación del comando. Si no se escribe nada, se cambia al directorio del usuario (el que se abrevia como "~"). Por ejemplo, ejecute **cd /bin** para cambiar a un nuevo directorio dado por una ruta absoluta, puede ver en qué directorio se encuentra tras el cambio ejecutando **pwd**. A continuación, vuelva al directorio del usuario ejecutando **cd** sin ninguna ruta, que llevará al directorio origen del usuario. Puede corroborarlo, ejecutando **pwd**

- **mkdir** (make a directory). Crea una carpeta en la ruta dada como argumento. Sólo se puede crear la última carpeta de la ruta a no ser que se utilice otra opción para permitir la creación de las carpetas de nivel superior. Estando desde `/home/teleco` , cree una nueva carpeta que se llame proyecto con **mkdir proyecto** de esta forma se ha creado pasando una ruta relativa, la carpeta proyecto desde el directorio donde estábamos. A continuación crearemos una nueva carpeta pero pasando una ruta absoluta, de la siguiente forma **mkdir /home/teleco/proyecto/grupoA** y también una nueva carpeta pasando una ruta relativa de la siguiente forma **mkdir /home/teleco/proyecto/grupoB**. Observe que ahora debajo del directorio de trabajo, se ha creado la carpeta proyecto y dentro de esta dos subcarpetas denominadas grupoA y grupoB

- **rmdir** (remove directory). Borra la carpeta dada como argumento. Si la carpeta no está vacía no se permite su borrado. Desde el directorio `/home/teleco` borre el subdirectorio grupoB pasando la ruta absoluta del siguiente modo **rmdir /home/teleco/proyecto/grupoB** ¿Cómo lo habría realizado para hacer el mismo borrado pero utilizando una ruta relativa?

- **touch** (touch file). Modifica el tiempo de acceso y modificación de un fichero si el parámetro pasado como argumento de fichero existe, o crea un fichero vacio si no existe. Ejecute **touch /home/teleco/proyecto/grupoA/test.txt** para crear un fichero vacio. Seguidamente, ejecute **ls -lart /home/teleco/proyecto/grupoB/test.txt**, seguidamente haga de nuevo **touch /home/teleco/proyecto/grupoA/test.txt** y al realizar otra vez seguidamente **ls -lart /home/teleco/proyecto/grupoB/test.txt** observará como la información de tiempos del fichero ha cambiado. En cualquier caso, el contenido del fichero permanece vacío.

- **mv** (move file). "Mueve" o renombra un fichero del nombre dado como primer argumento al segundo. Si sólo se especifican nombres de ficheros, el comando únicamente cambia el nombre de fichero. Pero si el segundo argumento es una ruta, lo cambia de lugar en el árbol de ficheros. Por ejemplo, si desde `/home/teleco` ejecutamos **mv proyecto/grupoA/test.txt proyecto/grupoB/test.txt** estaremos moviendo el fichero desde el directorio grupoA al grupoB

- **cp** (copy files). Copia un fichero fuente a uno destino o un conjunto de ficheros a una carpeta destino. Por ejemplo, si ejecutamos **cp /home/teleco/proyecto/grupoA/test.txt /home/teleco/proyecto/grupoB/test.txt** estaremos copiando el fichero del directorio grupoA al grupoB.

- **rm** (remove file). Borra el fichero en la ruta dada como argumento. El comportamiento de este comando tiene múltiples variaciones que se invocan inluyendo diferentes opciones (ver sección 11.4.1). Borre el fichero creado anteriormente ejecutando **rm /home/teleco/proyecto/grupoA/test.txt** , luego haga el listado de dicho directorio con **ls** y compruebe que dicho fichero ha desaparecido. Especialmente útil es borrar no sólo un fichero sino todos los directorios y ficheros que estén debajo de un determinado directorio. Esto se puede obtener utilizando algunas opciones concretas. Así por ejemplo, ejecutando **rm -rf /home/teleco/proyecto** se borrará todos los ficheros y directorios que haya debajo de la ruta especificada de un modo recursivo.

A continuación se muestra una sesión con el intérprete. El comando **exit** termina la ejecución del intérprete y por tanto se cierra la ventana. Te recomendamos que repliques esta sesión en tu ordenador para verificar que los comandos funcionan como esperas.

```
$ pwd
/home/teleco
$ ls
Descargas Documentos Escritorio
$ mkdir nombre
$ ls
Descargas Documentos Escritorio nombre
$ cd nombre
```

```
$ pwd
/home/teleco/nombre
$ ls
$ cd ..
$ pwd
/home/teleco
$ rmdir nombre
$ ls
Descargas Documentos Escritorio
$ exit
```

El intérprete arranca con la carpeta `/home/teleco` como directorio actual. A continuación se crea una subcarpeta **nombre** con el comando **mkdir**, se cambia el directorio actual a ella, se muestra su contenido con el comando **ls** (está vacía pues se acaba de crear), se vuelve a la carpeta superior con el comando **cd ..**, se utiliza el comando **rmdir** para borrar la carpeta (sólo la borra si está vacía) y finalmente muestra de nuevo el directorio actual que no ha cambiado.

11.4.4. Otros comandos útiles

El intérprete ofrece muchos comandos útiles. En esta sección vamos a repasar algunos de ellos. La funcionalidad detallada de cada uno de ellos la puedes acceder mediante el comando **man**. Esta es una selección de otros comandos:

- **date** (mostrar la fecha actual). Con este comando se puede mostrar la fecha actual en el sistema que se está ejecutando. Se puede obtener según diferentes formatos, dependiendo de las opciones que se utilicen en su invocación. Ejecute el comando **date** y observe el resultado.

- **echo** (imprime en la salida estándar). Imprime un cierto texto pasado como parámetro en la salida estándar. Por ejemplo, **echo "Esto es una prueba"** mostrará el texto correspondiente por pantalla.

- **Redirecciones de salida** Todo comando que se ejecuta en un intérprete de comandos, por defecto muestra su resultado por pantalla. Pero si queremos que su resultado no se muestre por pantalla y en su lugar se guarde en un fichero, debemos utilizar lo que se denomina como redirección de salida. Cuando se realiza una redirección de salida, se debe indicar un fichero destino donde se va a almacenar la salida. Si el fichero destino indicado no existe, el efecto será que se creará uno nuevo donde se almacenará la salida. Si el fichero destino ya existe, se almacenará en dicho fichero. Existen dos maneras de redirección de salida: **comando > fichero_destino** en el que si el fichero ya existe se sobreescribe el contenido que anteriormente hubiera, y **comando >> fichero_destino** en el que si el fichero ya existe no se sobreescribe el contenido que anteriormente hubiera sino que se añade al final del fichero a lo que ya hubiera. Ejecute **date > prueba.txt** y observará que el resultado ya no sale por pantalla sino que se crea un nuevo fichero de nombre prueba.txt donde se almacenará la fecha actual del sistema. A continuación ejecute **echo "Texto a poner tras la fecha" >> prueba.txt** y observe que el nuevo texto resultado de la ejecución del comando, aparece en el fichero tras la fecha en lugar de por pantalla. Si finalmente ejecutamos **ls -lart > prueba.txt** observaremos que se sobreescribe lo que anteriormente hubiera en el fichero por el listado completo del directorio actual.

- **cat** (print the content of files). Este comando imprime el contenido de los diferentes ficheros que son pasados como argumentos. Por ejemplo con **cat prueba.txt** puede ver el contenido del fichero prueba.txt

- **grep** (search a string in a files). Busca un string en uno o varios ficheros. Aunque este comando tiene muchas opciones vamos a ver aquí dos casos típicos de uso. En primer lugar para buscar un string en un determinado fichero se utiliza **grep string fichero**. Por ejemplo, para buscar el string "la" en el fichero "prueba.txt", ejecutariamos **grep "la" prueba.txt** . Por otro lado, si queremos buscar un string en todos los ficheros por debajo de un determinado directorio

queremos buscar un string en todos los ficheros por debajo de un determinado directorio, haremos lo siguiente, utilizando la opción -r **grep -r string ***

- **ps** (print the current processes). Permite listar los procesos que están en ese momento en el sistema operativo. Por ejemplo, cuando ejecutamos un programa, en muchas ocasiones un sólo proceso es creado asociado a ese programa, aunque también se pueden crear varios procesos para un sólo programa. Probaremos a ejecutar **ps** que nos mostrará los procesos que hay en el sistema operativo. Si queremos obtener más información sobre los procesos, podemos añadir ciertos parámetros, por ejemplo ejecutando **ps auxw** obtendremos información más detallada sobre los procesos. Finalmente, vamos a crear un proceso que estará indefinidamente en el sistema. Para ello, pondremos una línea de código que represente un bucle infinito, esto es por ejemplo "while(1);" como esta condición se cumplirá siempre, entonces el proceso estará indefinidamente ejecutándose en el sistema. Esta condición la pondremos en el main de un fichero .c y compilaremos y crearemos el ejecutable. Seguidamente lanzamos el proceso en segundo plano (para que nos devuelva el control del intérprete de comandos) con **./programa &**, si seguidamente ejecutamos **ps auxw** veremos el nuevo proceso que se ha creado. Si queremos eliminar el proceso, podemos enviarle una señal para terminar el proceso utilizando el comando **kill**. Para ello, deberemos ver el identificador del proceso PID tras la ejecución de **ps auxw** y seguidamente utilizar **kill -9 pid**

Preguntas de autoevaluación

Responde a las siguientes preguntas

1. Seleccione la frase correcta
   - Los argumentos de un comando se dividen en: el nombre del comando y el caracter "-"
   - Los argumentos de un comando se dividen en: aquellos objetos sobre los que hay que operar, y los argumentos para modificar su comportamiento
2. Seleccione la ruta equivalente a la ruta absoluta /dirA/dirB/fichero.txt
   - /dirA/dirB/./../../file.txt
   - /dirA/dirB/../dirB/file.txt
3. Que hace el comando `$mv Notes.txt notes.old`?
   - Cambia el nombre del fichero Notes.txt a notes.old
   - Mueve el fichero Notes.txt a la carpeta notes.old
4. Selecciona la frase correcta referente al comando `rmdir`:
   - Cambia el nombre de un fichero
   - Mueve un fichero
   - Borra un fichero
   - Borra un directorio
5. Seleccione el comando correcto para buscar la cadena ***"wordl"*** en el fichero ***main.c***
   - `$grep 'wordl' main.c`

- $grep main.c 'wordl'
- $grep main.c |'wordl'

11.4.5. Edición de comandos

El intérprete nos facilita la introducción de comandos mediante el uso de las teclas del cursor. Las de "subir" y "bajar" se recorre el histórico de comandos ya escritos en la sesión. Esto es muy útil para repetir la ejecución de un comando. Además, con las flechas horizontales se puede recorrer un comando para modificarlo. También se pueden usar combinaciones **CTRL-A** para mover el cursor al principio de la línea o **CTRL-E** para moverlo al final.

La tecla **Tabulador** se puede utilizar cuando se escriben nombres de ficheros para que el intérprete los termine automáticamente. Por ejemplo, si en una carpeta hay un fichero con nombre `ResultadosDeLaEjecucion`, se teclea solo `Re` y no hay ningún otro fichero que comience por esas dos letras en la carpeta (es decir, este prefijo es inequívoco), al pulsar el tabulador, el intérprete termina de escribir el fichero. Si hubiese más de un fichero con este mismo prefijo, el intérprete intentaría completar el nombre hasta el mayor prefijo común entre ellos. Este mecanismo es muy útil para escribir comandos de forma muy eficiente.

Te recomendamos que pruebes esta funcionalidad en tu equipo creando los ficheros `ResultadosDeLaEjecución1.txt` y `ResultadosDeLaEjecución2.txt`, y utilizándolos como argumentos para diferentes comandos. Comprueba como, aunque los nombres son largos, el tabulador te permite manejarlos con tan sólo unas cuantas teclas.

11.5. Otros programas

Al igual que con los comandos, el número de programas disponibles en Linux en muy grande. Seguiremos, por tanto la misma estrategia que con los comandos, los iremos utilizando conforme los necesitemos. Los que utilizaremos con más frecuencia serán:

- **kate**: el editor de texto
- **gcc**: el compilador de C
- **gdb**: el depurador
- **valgrind**: el detector de fugas de memoria.
- **soffice**: programa OpenOffice que contiene un procesador de textos, hoja de cálculo, programa de dibujo, y diseño de presentaciones.

11.6. Permisos

Todo fichero y carpeta en Linux tiene un conjunto de permisos. Para mostrar los permisos se utiliza la opción **-l** del comando **ls**. A continuación se muestra un ejemplo de la información que muestra este comando.

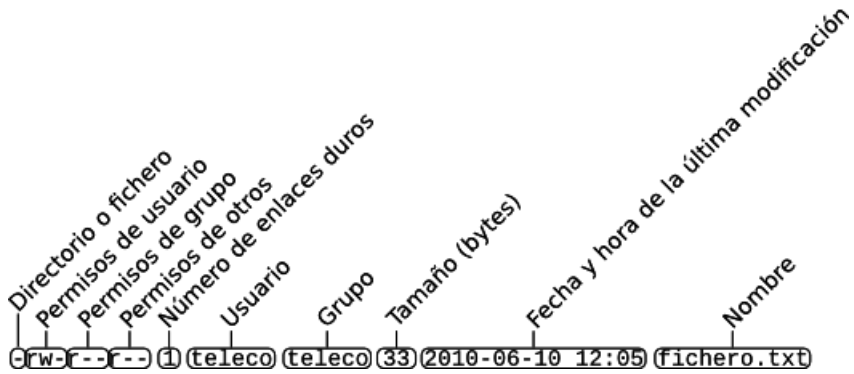
```
$ ls -l fichero.txt
-rw-r--r-- 1 teleco teleco 33 2010-06-10 12:05 fichero.txt
```

La convención utilizada por el intérprete para mostrar esta información es la siguiente (campos por orden de izquierda a derecha):

- El primer símbolo es "-" para un fichero y "d" para un directorio. Los 9 símbolos siguientes son los permisos. "r" para lectura, "w" para escritura, "x" para ejecución y "-" para la ausencia de ese permiso.

- Un número natural mayor que uno que denota el número de ***enlaces duros*** que apuntan al fichero (irrelevante por ahora).
- Nombre del usuario propietario
- Grupo del propietario
- Tamaño en bytes
- Fecha de la última modificación
- Nombre del fichero

La siguiente figura muestra el significado de cada campo para el ejemplo.



Los permisos se agrupan en tres categorías: usuario (u), grupo (g) y otros (o). Cada categoría contiene a su vez permisos para tres tipos de eventos: lectura (r), escritura (w) y ejecución (x). El permiso es un valor binario, o se dispone o se carece de él. En conclusión, cada fichero o carpeta contiene un subconjunto de nueve posibles permisos. Además de los permisos, cada fichero y carpeta tiene también dos nombre: el nombre del usuario propietario, y el nombre de un grupo de usuarios al que pertenece el propietario.

Cada usuario en Linux tiene asignado un nombre (su login), y pertenece a uno o varios grupos de usuarios. El comando **id** muestra el nombre del usuario y los grupos a los que pertenece. Tanto usuarios como grupos se definen con un nombre y un número natural.

Supongamos que un usuario con nombre "uname" quiere realizar una operación sobre un fichero. El sistema primero selecciona qué categoría de permisos debe utilizar. Si el propietario del fichero es "uname" utiliza la categoría "usuario". Si no es el caso, pero el grupo del fichero es uno de los que es miembro el usuario "uname" entonces se utiliza la segunda categoría ("grupo"). Si tampoco es el caso, entonces se utilizan los permisos de la tercera categoría. Una vez seleccionada la categoría se comprueba si se dispone del permiso para realizar la operación (lectura, escritura o ejecución) y se autoriza o rechaza la operación.

Para cambiar los permisos de un fichero o carpeta desde el intérprete de comandos se utiliza el comando **chmod** ("*change mode*"). En su versión más simple recibe dos parámetros, una cadena de texto describiendo los cambios en los permisos y una ruta a un fichero o carpeta sobre el que aplicar estos cambios. La cadena tiene tres partes: una o varias de las letras "u", "g" u "o", seguida del signo "+" o "-", seguida de una o varias de las letras "r", "w" o "x". La interpretación es que a la categoría especificada en el primer grupo de letras se le añade (si se utiliza "+") o se le retira (si se utiliza "-") el permiso indicado por el segundo grupo de letras. Por ejemplo, para añadir a un fichero el permiso de ejecución para su propietario y su grupo se ejecuta el comando:

```
$ chmod ug+x fichero.txt
```

Otra manera de cambiar los permisos de un fichero o carpeta desde el intérprete de comandos utilizando el comando **chmod** es también recibiendo dos parámetros: el primer parámetro son 3

números, cada uno del 0 al 7, que se transforma a 3 bits en binario, coincidiendo respectivamente cada bit si está activo o no si tiene permiso de lectura, escritura y ejecución. Los tres números se corresponden con usuario, grupo y otros. Por ejemplo, para dar todos los permisos del archivo al propietario y ninguno al resto se ejecuta:

```
$ chmod 700 fichero.txt
```

11.7. Resumen

Al arrancar la máquina virtual se muestra un escritorio con un panel en la parte inferior. Este panel contiene varios iconos que nos muestran información del sistema y nos permiten arrancar ciertas aplicaciones. El resto de ellas se pueden arrancar a través del menú principal.

El terminal de comandos es una ventana que ejecuta el intérprete de comandos. En la máquina virtual esta aplicación se llama **konsole**. Este intérprete espera a recibir un comando terminado por una nueva línea y lo ejecuta. En todo momento el intérprete mantiene un directorio actual. Los comandos se ejecutan escribiendo su nombre seguido de argumentos en los que se indican los objetos sobre los que operar o modificaciones en su modo de operación.

Los ficheros y carpetas en Linux se organizan en una estructura de árbol con la carpeta / en la raíz. Nos podemos referir a los ficheros o con una ruta absoluta (comenzando desde la raíz) o con una relativa al directorio actual. Toda carpeta contiene dos subcarpetas, "." que apunta a la propia carpeta y ".." que apunta a la superior en la jerarquía de ficheros.

Los comandos más comunes para gestionar ficheros son:

Comando	Función	Ejemplo
ls	listar el contenido de directorios	**ls -l ~**
pwd	Mostrar el directorio actual	**pwd**
cd	cambia el directorio actual	**cd ../.**
mkdir	Crear un directorio	**mkdir nuevodir**
rmdir	Borrar un directorio vacío	**rmdir dir**
rm	borrar ficheros y directorios	**rm fichero1 fichero2**
mv	renombrar de ficheros y cambiar de ubicación de ficheros a otros directorios	**mv viejo nuevo**, **mv ficheros directorio**
cp	copiar ficheros	**cp forigen fdestino**, **cp ficheros directorio**
chmod	Cambiar los permisos	**chmod go-rwx .**

11.8. Preguntas de autoevaluación

Comprueba con estas preguntas que has entendido nociones básicas del entorno de trabajo en Linux.

1. Si quiero obtener información sobre cómo funciona el propio manual de BASH, ¿qué comando debería introducir?:

   - `man man`
   - `man -k man`
   - `ls man`
   - `man pwd`
   - Ninguna de las anteriores.

2. ¿Qué tarea realiza el comando `ls -a`?:

   - Lista sólo las entradas de tipo fichero, excluyendo las de directorio.
   - Lista el contenido del directorio actual, mostrando un detalle largo de su información por cada entrada.
   - Lista el contenido del directorio actual, mostrando el tamaño para cada directorio o fichero.
   - Lista el contenido del directorio actual, incluyendo ficheros y directorios que empiezan por ".”
   - Ninguna de las anteriores.

3. Selecciona la frase correcta referente a rutas:

   - Una determinada ruta absoluta puede referirse a más de un fichero distinto.
   - Una ruta absoluta empieza con el símbolo ".”.
   - Si desde cualquier directorio quiero cambiar al directorio raíz, tengo que hacerlo con el comando "cd .”
   - El comando "mv” no sólo sirve para cambiar de lugar ficheros, sino también para renombrarlos.
   - Ninguna de las anteriores.

4. El fichero `Notes.txt` tiene los siguientes permisos:

   ```
   -rw-r--r-- 1 teleco teleco 7 2011-09-07
          14:18 Notes.txt
   ```

   Quiero que mi grupo pueda escribir en el fichero, pero no otros usuarios. ¿Cuál es la opción correcta?

   - chmod +gw Notes.txt
   - chmod group +w Notes.txt
   - chmod +gw-ow Notes.txt
   - chmod group +w other -w Notes.txt
   - chmod g+w Notes.txt
   - chmod g+w o+w Notes.txt
   - Ninguna de las anteriores.

5. El fichero `Notes.txt` tiene los siguientes permisos:

   ```
   -rw-r--r-- 1 teleco teleco 7 2011-09-07
          14:18 Notes.txt
   ```

¿Qué significa el primer guión de la tanda de permisos?

- Siempre hay un guión al principio de las entradas, no significa nada.
- Significa que el usuario "teleco" no tiene permisos de lectura.
- Significa que la entrada es un fichero, no un directorio.
- Ninguna de las anteriores.

6. Acabo de ejecutar un comando en el intérprete que quiero volver a ejecutar. Sin embargo, es largo y para ahorrar tiempo no quiero volver a teclearlo, así que voy a usar el histórico de comandos. ¿Como puedo acceder a ese comando a través del histórico?

- Tecla de "subir"
- CTRL-A
- Tabulador
- Cualquiera de las anteriores.
- Ninguna de las anteriores.

11.9. Bibliografía de apoyo

- Mendel Cooper "Advanced Bash-Scripting Guide", http://www.tldp.org/LDP/abs/html/index.html , Parte 4, Capítulos 15, 16, y 17
- "The GNU bash reference manual", http://www.gnu.org/software/bash/manual/bash.html#Bourne-Shell-Builtins