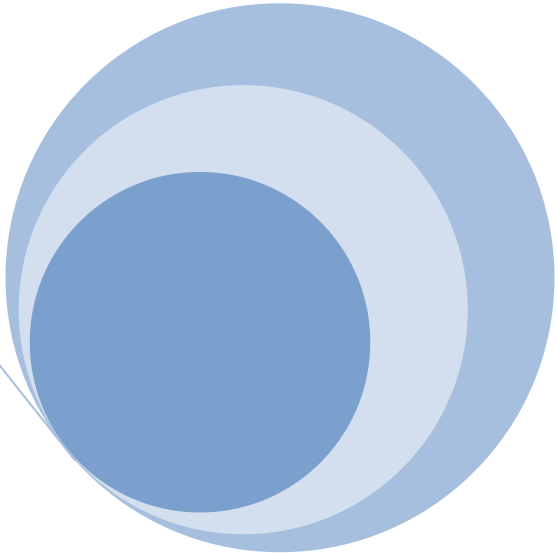
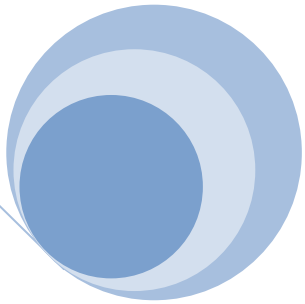


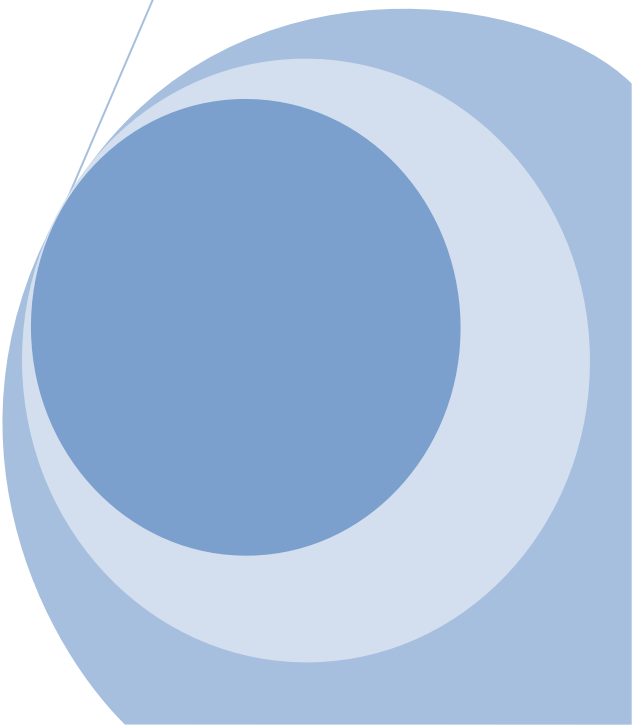


Metodología de los laboratorios de SO

Curso primavera 2011-2012

Este documento describe el funcionamiento de los laboratorios de la asignatura de Sistemas Operativos.

Profesores SO
08/02/2012



Funcionamiento de los laboratorios

Qué hay que hacer en los laboratorios

Los laboratorios de esta asignatura son semanales (2 horas) y las prácticas se realizan de forma individual. Los enunciados están todos divididos en tres partes:

- **Trabajo previo.**
  - Se pide al alumno que realice los trabajos que se especifican en el trabajo previo antes de ir al laboratorio (como su nombre indica). Estos trabajos pueden ser preparar algún comando o llamada a sistema, probar o modificar los ejemplos que se proporcionan, etc. Es importante ya que los ejercicios que debéis hacer en el laboratorio asumen que habéis hecho este trabajo.
    - **La duración aproximada del trabajo previo es de 2 horas.**
    - **Al inicio de cada laboratorio hay que entregar via Racó el trabajo previo.**
- **Ejercicios a realizar en el laboratorio.**
  - Se pide realizar una serie de ejercicios y pruebas en el mismo laboratorio. La cantidad de ejercicios está pensada para que se pueda realizar en las dos horas que dura el laboratorio. Sin embargo, lo más importante es realizar y comprender todas las sesiones aunque sea fuera de horas de clase los controles de laboratorio estarán basados en estos ejercicios.
    - **Al finalizar la sesión se entregará via Racó el trabajo realizado.**
    - **Estos ejercicios no se van a corregir explícitamente, sino que se validarán durante la sesión por parte del profesor y es responsabilidad vuestra si no habéis entendido algún ejercicio preguntar a vuestro profesor.**

Acceso a la documentación

La documentación de los laboratorios estará accesible a través de la página web de la asignatura (http://docencia.ac.upc.edu/FIB/grau/SO).

Asistencia al laboratorio

La asistencia al laboratorio es obligatoria para aquellos alumnos que quieran optar a la evaluación continua, los alumnos deberán asistir, como mínimo al 80% de las sesiones. Aunque todas las sesiones tendrán incorporadas preguntas a entregar, será solamente en las sesiones marcadas como CONTROL_LAB1 y CONTROL_LAB2 que se realizarán los exámenes de laboratorio de los cuales se extraerá la nota de laboratorio.

Arranque del sistema y consideraciones especiales

La descripción de las aulas la podéis encontrar en este enlace de la FIB http://www.fib.upc.edu/es/serveis/aulesSO.html . Los laboratorios tienen PC's que arrancan

una imagen del sistema de forma remota utilizando REMBO. Vosotros tenéis que elegir la opción “Nova Ubuntu 10.04.LTS”.

La imagen que se carga en estos PC’s se modifica cada vez que se carga una nueva imagen, por lo que normalmente al acabar la clase el trabajo realizado se pierde. Por lo tanto, es muy importante que vuestro trabajo lo almacenéis en un lugar seguro antes de marcharos. Para ello podéis utilizar el servidor Albanta que ofrece la fib. En Albanta tenéis una cuenta de usuario por alumno con el username/password que usáis en la la FIB.

Tanto para transferir archivos fácilmente como para hacer entregas en el raco de más de un fichero, es útil generar un único archivo que contenga todos los demás comprimidos. Para ello podéis utilizar el comando tar, que además de agrupar los ficheros permite comprimirlos. Por ejemplo, si queremos generar un fichero a partir de todos los ficheros que tengamos en el directorio S1, podéis hacer.

#tar zcvf S1.tar.gz S1/*

Para descomprimir el fichero anterior el comando sería:

#tar zxvf S1.tar.gz

Por ejemplo, para guardar los ejercicios de la sesión 1 , podríamos hacer:

```
#cd
#tar zcvf S1.tar.gz S1/*
#sftp tu_username@albanta.fib.upc.edu
>put S1.tar.gz
>quit
#
```

Acceso al sistema

En los laboratorios tenemos instalado Ubuntu 10.04.LTS. Para comenzar, ejecutaremos lo que llamamos una *Shell* o un intérprete de comandos. Existen varios intérpretes de comandos, en el laboratorio utilizaréis Bash (GNU-Bourne Shell), pero en general nos referiremos a él como Shell. La mayoría de las cosas que explicaremos en esta sesión podéis leerlas en el manual de Bash (#man bash). Encontrareis varios usuarios creados para que podáis hacer las pruebas que os pedimos fácilmente. Los usernames de los usuarios son:”alumne”, “so1”, “so2”, “so3”, “so4” y “so5”. El password es “sistemes” para todos ellos.

Para ello ejecutaremos el “Terminal” y se nos abrirá una ventana similar a la de la imagen. Una *Shell* es un programa que el S.O. nos ofrece para poder trabajar en un modo de texto interactivo. Este entorno puede parecer menos intuitivo que un entorno gráfico, pero es muy sencillo y potente.

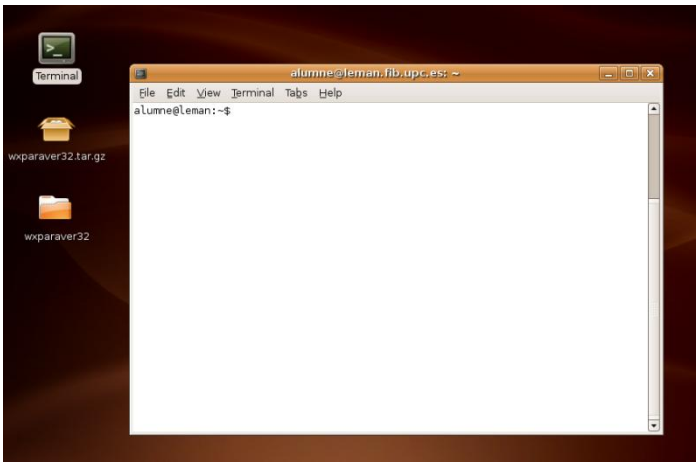


Figura 1 Ventana de la shell

El texto que aparece a la izquierda junto con el cursor que parpadea es lo que se conoce como *prompt* y es donde introduciremos nuestras órdenes o comandos.

El código de la Shell se podría resumir de la siguiente manera:

```
while(1){
      comando=leer_comando();
      ejecutar_comando(comando);
}
```

Utilización del manual

Saber utilizar el manual es básico ya que, aunque durante el curso os explicaremos explícitamente algunos comandos y opciones, el resto (incluido llamadas a sistema) deberéis buscarlo vosotros mismos en el manual. El propio man es auto contenido en este sentido, ya que podéis ejecutar:

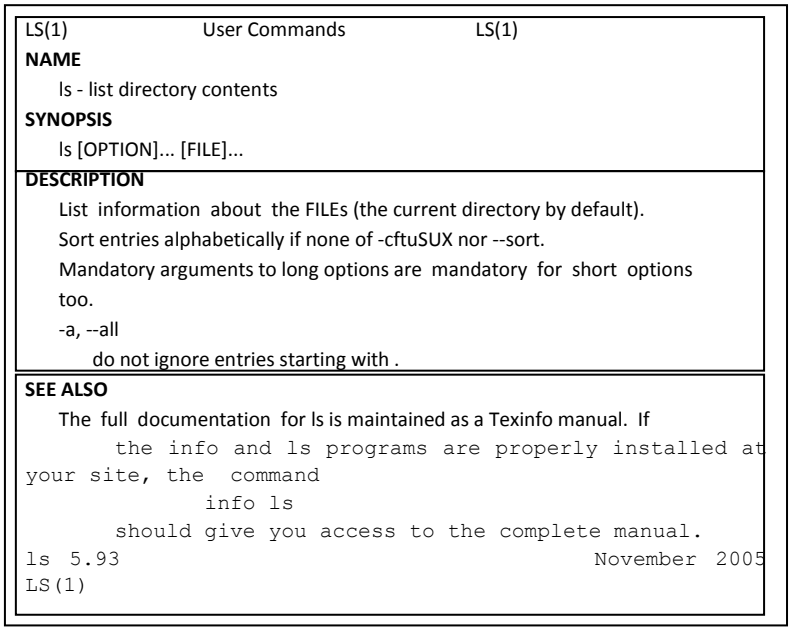
#man man

Y veréis todas sus opciones. La información del manual está organizada en secciones. La sección 2, por ejemplo, es la de llamadas a sistema. Las secciones que podemos encontrar son:

1. comandos
2. llamadas a sistema
3. llamadas a librerías de usuario o del lenguaje
4. etc.

La información proporcionada al ejecutar el man es lo que se conoce como “página de man”. Una “página” suele ser el nombre de un comando, llamada a sistema o llamada a función. Todas las páginas de man siguen un formato parecido, organizado en una serie de partes. En la Figura 2 tenéis un ejemplo de la salida del man para el comando ls (hemos borrado alguna línea para que se vean las principales partes). En la primera parte podéis encontrar tanto el nombre del comando como la descripción y un esquema (SYNOPSIS) de su utilización. En esta

parte podéis observar si el comando acepta opciones, si necesita algún parámetro fijo u opcional, etc.



LS(1) User Commands LS(1)

NAME

ls - list directory contents

SYNOPSIS

ls [OPTION]... [FILE]...

DESCRIPTION

List information about the FILEs (the current directory by default).

Sort entries alphabetically if none of -cftuSUX nor --sort.

Mandatory arguments to long options are mandatory for short options too.

-a, --all

do not ignore entries starting with .

SEE ALSO

The full documentation for ls is maintained as a Texinfo manual. If the info and ls programs are properly installed at your site, the command

info ls

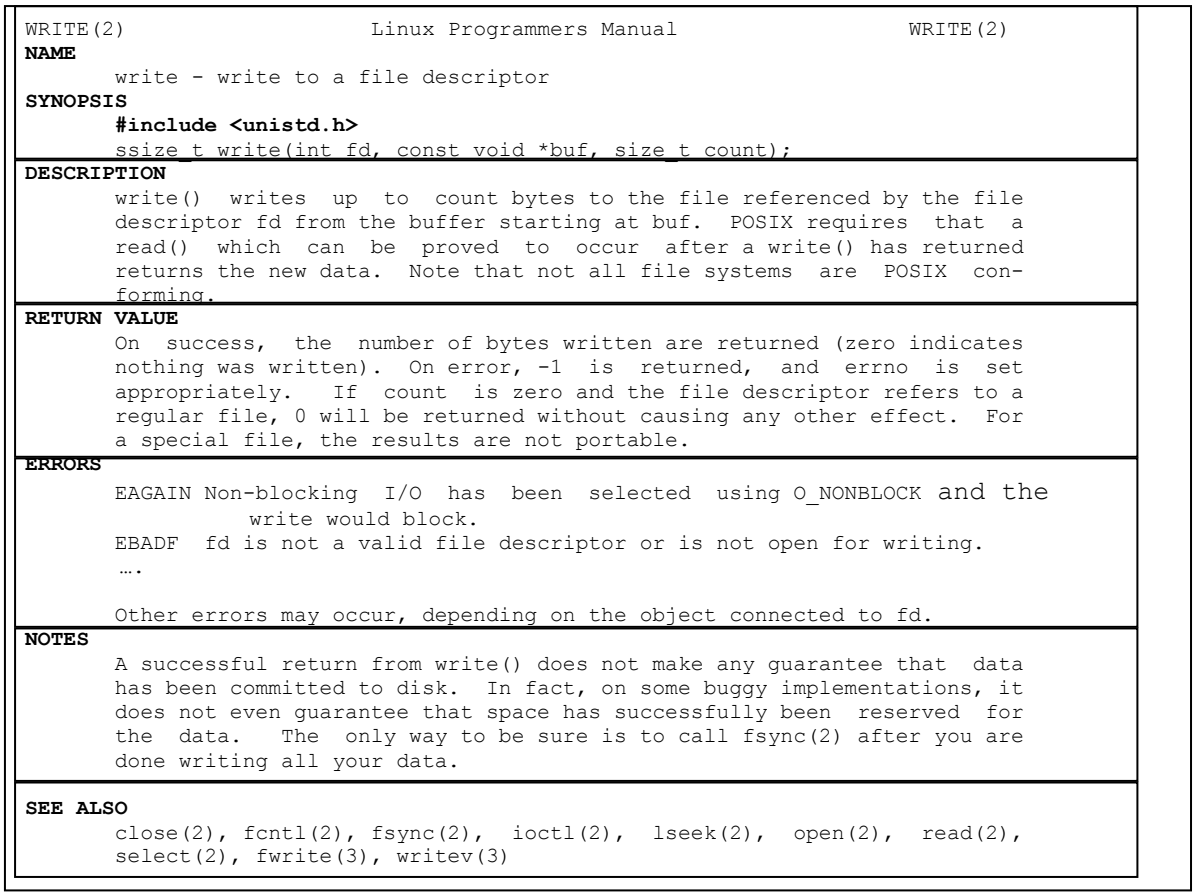
should give you access to the complete manual.

ls 5.93 November 2005 LS(1)

Figura 2 man ls (simplificado)

La siguiente parte sería la descripción (DESCRIPTION) del comando. Esta parte incluye una descripción más detallada de su utilización y la lista de opciones que soporta. Dependiendo de la instalación de las páginas de man también podéis encontrar aquí los códigos de finalización del comando (EXIT STATUS). Finalmente suele haber una serie de partes que incluyen los autores de la ayuda, la forma de reportar errores, ejemplos y comandos relacionados (SEE ALSO).

En la Figura 3 tenéis el resultado de ejecutar “man 2 write”, que corresponde con la llamada a sistema write. El número que ponemos antes de la página es la sección en la que queremos buscar y que incluimos en esta caso ya que existe más de una página con el nombre write. En este caso la SYNOPSIS incluye los ficheros que han de ser incluidos en el programa C para poder utilizar la llamada a sistema en concreto (en este caso unistd.h). Si fuera necesario “*linkar*” vuestro programa con alguna librería concreta, que no fueran las que utiliza el compilador de C por defecto, lo normal es que aparezca también en esta sección. Además de la DESCRIPTION, en las llamadas a función en general (sea llamada a sistema o a librería del lenguaje) podemos encontrar la sección RETURN VALUE (con los valores que devuelve la función) y una sección especial, ERRORS, con la lista de errores. Finalmente también encontramos varias secciones donde aquí destacamos también la sección de NOTES (aclaraciones) y SEE ALSO (llamadas relacionadas).



```
WRITE(2)                  Linux Programmers Manual                  WRITE(2)
NAME
       write - write to a file descriptor
SYNOPSIS
       #include <unistd.h>
       ssize_t write(int fd, const void *buf, size_t count);
DESCRIPTION
       write()  writes  up  to  count bytes to the file referenced by the file
       descriptor fd from the buffer starting at buf.  POSIX requires  that  a
       read()  which  can  be  proved  to  occur  after a write() has returned
       returns the new data.  Note that not all file systems  are  POSIX  con-
       forming.
RETURN VALUE
       On  success,  the  number of bytes written are returned (zero indicates
       nothing was written).  On error, -1  is  returned,  and  errno  is  set
       appropriately.   If  count  is zero and the file descriptor refers to a
       regular file, 0 will be returned without causing any other effect.  For
       a special file, the results are not portable.
ERRORS
       EAGAIN Non-blocking  I/O  has  been  selected  using O_NONBLOCK and the
                 write would block.
       EBADF  fd is not a valid file descriptor or is not open for writing.
       ….

       Other errors may occur, depending on the object connected to fd.
NOTES
       A successful return from write() does not make any guarantee that  data
       has been committed to disk.  In fact, on some buggy implementations, it
       does not even guarantee that space has successfully been  reserved  for
       the  data.   The  only way to be sure is to call fsync(2) after you are
       done writing all your data.

SEE ALSO
       close(2), fcntl(2), fsync(2),  ioctl(2),  lseek(2),  open(2),  read(2),
       select(2), fwrite(3), writev(3)
```

Figura 3 man 2 write (simplificado)

El man es simplemente una herramienta del sistema que interpreta unas marcas en ficheros de texto y las muestra por pantalla siguiendo las instrucciones de esas marcas. Las tres cosas básicas que tenéis que saber son:

- Normalmente una página de man ocupa varias pantallas, para ir avanzando simplemente hay que apretar la barra espaciadora.
- Para ir una pantalla hacia atrás podéis apretar la letra **b** (back).
- Para buscar un texto e ir directamente podéis usar el carácter "/" seguido del texto. Por ejemplo "/SEE ALSO" os llevaría directo a la primera aparición del texto "SEE ALSO". Para ir a la siguiente aparición del mismo texto simplemente utilizad el carácter **n** (next).