COMANDOS

Sessió 7: Gestió d'Entrada / Sortida

mknod: Comando que crea un fichero especial

```
mknod [OPTION]... NAME TYPE [MAJOR MINOR]
```

> mknod - make block or character special files
>
> Create the special file NAME of the given TYPE. - c: create a character (unbuffered) special file - p: create a FIFO

insmod: Comando que inserta un módulo en el kernel

```
  insmod [filename] [module options ...]
```

> insmod — simple program to insert a module into the Linux Kernel
>
> insmod is a trivial program to insert a module into the kernel: if the filename is a hyphen, the module is taken from standard input. Only the most general of error messages are reported.

rmmod: Comando que descarga un módulo del kernel

```
rmmod [-f] [-w] [-s] [-v] [modulename]
```

> rmmod — simple program to remove a module from the Linux Kernel
>
> rmmod is a trivial program to remove a module (when module unloading support is provided) from the kernel.

lsmod: Comando que muestra el estado de los módulos cargados en el kernel

```
lsmod
```

> lsmod — program to show the status of modules in the Linux Kernel

> lsmod is a trivial program which nicely formats the contents of the */proc/modules*, showing what kernel modules are currently loaded.

sudo: Comando que permite ejecutar un comando como root

```
sudo [-D level] -h | -K | -k | -V

sudo -v [-AknS] [-D level] [-g group name|#gid] [-p prompt] [-u user name|#uid]

sudo -l[l] [-AknS] [-D level] [-g group name|#gid] [-p prompt] [-U user name] [-u
user name|#uid] [command]

sudo [-AbEHnPS] [-C fd] [-D level] [-g group name|#gid] [-p prompt] [-u user
name|#uid] [VAR=value] [-i | -s] [command]

sudoedit [-AnS] [-C fd] [-D level] [-g group name|#gid] [-p prompt] [-u user
name|#uid] file ...
```

> sudo, sudoedit - execute a command as another user
>
> sudo allows a permitted user to execute a command as the superuser or another user, as specified by the security policy. The real and effective uid and gid are set to match those of the target user, as specified in the password database, and the group vector is initialized based on the group database (unless the -P option was specified).

open: Abre un fichero o dispositivo

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
int creat(const char *pathname, mode_t mode);
```

> open, creat - open and possibly create a file or device
>
> Given a pathname for a file, open() returns a file descriptor, a small, nonnegative integer for use in subsequent system calls (read(2), write(2), lseek(2), fcntl(2), etc.). The file descriptor returned by a successful call will be the lowest-numbered file descriptor notcurrently open for the process.

write: Llamada a sistema para escribir en un dispositivo virtual

```
#include <unistd.h>
```

```
ssize_t write(int fd, const void *buf, size_t count);
```

> write - write to a file descriptor
>
> write() writes up to count bytes from the buffer pointed buf to the file referred to by the file descriptor fd. On success, the number of bytes written is returned (zero indicates nothing was written). On error, -1 is returned, and errno is set appropriately.

read: Llamada a sistema para leer de un dispositivo virtual

```
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t count);
```

> read - read from a file descriptor read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf.
>
> On success, the number of bytes read is returned (zero indicates end of file), and the file position is advanced by this number. It is not an error if this number is smaller than the number of bytes requested; this may happen for example because fewer bytes are actually available right now (maybe because we were close to end-of-file, or because we are reading from a pipe, or from a terminal), or because read() was interrupted by a signal. On error, -1 is returned, and errno is set appropriately. In this case it is left unspecified whether the file position (if any) changes.

grep: Comando que busca patrones en un fichero o en -c su entrada estándar si no se le pasa fichero como parámetro

```
grep [-E| -F][-c| -l| -q][-insvx] -e pattern_list...[-f pattern_file]...[file...]
grep [-E| -F][-c| -l| -q][-insvx][-e pattern_list]...-f pattern_file...[file...]
grep [-E| -F][-c| -l| -q][-insvx] pattern_list[file...]
```

> grep - search a file for a pattern
>
> The grep utility shall search the input files, selecting lines matching one or more patterns; the types of patterns are controlled by the options specified. - -c: Write only a count of selected lines to standard output.

ps: Comando que muestra información sobre los procesos en ejecución

```
ps [options]
```

> ps - report a snapshot of the current processes.
>
> ps displays information about a selection of the active processes.
>
> - -e: Select all processes. Identical to -A.
> - -o: format User-defined format. format is a single argument in the form of a blank-separated or comma-separated list, which offers a way to specify individual output columns.

strace: Lista las llamadas a sistema ejecutadas por un proceso

```
strace [-CdffhiqrtttTvxx] [-acolumn] [-eexpr] ... [-ofile] [-ppid] ... [-sstrsize]
[-uusername] [-Evar=val] ... [-Evar] ... [command [arg...]]
strace -c [-eexpr] ... [-Ooverhead] [-Ssortby] [command [arg...]]
```

> strace - trace system calls and signals
>
> In the simplest case strace runs the specified command until it exits. It intercepts and records the system calls which are called by a process and the signals which are received by a process.
>
> - -e: expr A qualifying expression which modifies which events to trace or how to trace them. The format of the expression is: `[qualifier=][!]value1[,value2]...` where qualifier is one of trace, abbrev, verbose, raw, signal, read, or write and value is a qualifier-dependent symbol or number. The default qualifier is trace. Using an exclamation mark negates the set of values. For example, -e open means literally -e trace=open which in turn means trace only the open system call. By contrast, -e trace=!open means to trace every system call except open. In addition, the special values all and none have the obvious meanings.
> - -c: Count time, calls, and errors for each system call nd report a summary on program exit. On Linux, this attempts to show system time (CPU time spent running in the kernel) independent of wall clock time. If -c is used with -f or -F (below), only aggregate totals for all traced processes are kept.

Sessió 8: Gestió d'Entrada/Sortida 2

mknod: [comando ---> mirar sessió 7]

mknod (llamada al sistema): Llamada al sistema que crea un fichero especial

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
int mknod(const char *pathname, mode_t mode, dev_t dev);
```

> mknod, mknodat - create a special or ordinary file
>
> The system call mknod() creates a filesystem node (file, device special file, or named pipe) named pathname, with attributes specified by mode and dev. The mode argument specifies both the file mode to use and the type of node to be created. It should be a combination (using bitwise OR) of one of the file types listed below and zero or more of the file mode bits listed in stat(2).

pipe: Llamada a sistema para crear una pipe sin nombre

```
#include <unistd.h>
int pipe(int pipefd[2]);
```

> pipe, pipe2 - create pipe
>
> pipe() creates a pipe, a unidirectional data channel that can be used for interprocess communication. The array pipefd is used to return two file descriptors referring to the ends of the pipe.

open: Abre un fichero o dispositivo

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

> open, openat, creat - open and possibly create a file
>
> - **O_NONBLOCK:** When possible, the file is opened in nonblocking mode. Neither the open() nor any subsequent operations on the file descriptor which is returned will cause the calling process to wait. Note that this flag has no effect for regular files and block devices; that is, I/O operations will (briefly) block when device activity is required, regardless of whether O_NONBLOCK is set. Since O_NONBLOCK semantics might eventually be imple mented, applications should not depend upon blocking behavior

when specifying this flag for regular files and block devices. For the handling of FIFOs (named pipes), see also fifo(7). For a discussion of the effect of O_NONBLOCK in conjunction with mandatory file locks and with file leases, see fcntl(2).
- **ENXIO:** O_NONBLOCK | O_WRONLY is set, the named file is a FIFO, and no process has the FIFO open for reading. Or, the file is a device special file and no corresponding device exists.

close: Cierra un descriptor de fichero

```
#include <unistd.h>
int close(int fd);
```

> close - close a file descriptor
>
> RETURN VALUE: close() returns zero on success. On error, -1 is returned, and errno is set appropriately.

dup/dup2: Duplica un descriptor de fichero

```
#include <unistd.h>
int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

> dup, dup2, dup3 - duplicate a file descriptor
>
> The dup() system call creates a copy of the file descriptor oldfd, using the lowest-numbered unused file descriptor for the new descriptor. - Hace la copia en la primera entrada libre de la Tabla de Canales. Se comparten los punteros de lectura y escritura. Es decir, si yo escribo algo en fd, si luego sigo escribiendo en newfd continuará por donde lo he dejado con el fd antiguo (no se ha sobreescrito). - -1 ---> error - 0 ---> canal nuevo
>
> The dup2() system call performs the same task as dup(), but instead of using the lowest-numbered unused file descriptor, it uses the file descriptor number specified in newfd. If the file descriptor newfd was previously open, it is silently closed before being reused. - Def. Cierra el canal newfd si estaba abierto y se duplica el canal fd en newfd. Lo hace de forma atómica (en una sola instrucción).

socket: (llamada al sistema) Crea un socket

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```

socket - create an endpoint for communication

socket() creates an endpoint for communication and returns a descriptor. - AF_UNIX: Local communication. The AF_UNIX (also known as AF_LOCAL) socket family is used to communicate between processes on the same machine efficiently.

`unix_socket = socket(AF_UNIX, type, 0);`

`error = socketpair(AF_UNIX, type, 0, int *sv);` - SOCK_STREAM: Provides sequenced, reliable, two-way, connection-based byte streams. An out-of-band data transmission mechanism may be supported.

bind: Asigna un nombre o dirección a un socket

```
#include <sys/types.h>
#include <sys/socket.h>
int bind(int s, const struct sockaddr *addr, socklen_t addrlen);
```

bind — assign a local protocol address to a socket

The bind() system call assigns the local protocol address to a socket. When a socket is created with socket(2) it exists in an address family space but has no protocol address assigned. The bind() system call requests that addr be assigned to the socket.

RETURN: 0 if successful; otherwise the value -1 is returned and the global variable errno is set to indicate the error.

listen: Espera conexiones a un socket

```
#include <sys/types.h>
#include <sys/socket.h>
int listen(int s, int backlog);
```

listen — listen for connections on a socket

To accept connections, a socket is first created with socket(2), a willingness to accept incoming connections and a queue limit for incoming connections are specified with

> listen(), and then the connections are accepted with accept(2). The listen() system call applies only to sockets of type SOCK_STREAM or SOCK_SEQPACKET.

accept: Acepta una conexión en un socket

```
#include <sys/types.h>
#include <sys/socket.h>
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

> accept - accept a connection on a socket
>
> The accept() system call is used with connection-based socket types (SOCK_STREAM, SOCK_SEQPACKET). It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket. The newly created socket is not in the listening state. The original socket sockfd is unaffected by this call.

connect: Inicia una conexión a un socket

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

> connect - initiate a connection on a socket
>
> The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by addr. The addrlen argument specifies the size of addr. The format of the address in addr is determined by the address space of the socket sockfd; see socket(2) for further details.
>
> RETURN: If the connection or binding succeeds, 0 is returned. On error, -1 is returned, and errno is set appropriately.

Sessió 9: Sistema de fitxers

open/creat: Abre/crea un fichero o dispositivo

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

```
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
int creat(const char *pathname, mode_t mode);
```

open, creat - open and possibly create a file or device

FLAGS: - O_CREAT: If the file does not exist it will be created. - O_TRUNC: If the file already exists and is a regular file and the open mode allows writing (i.e., is O_RDWR or O_WRONLY) it will be truncated to length 0.

df: Devuelve información sobre el sistema de ficheros

```
df [-k][-P|-t][file...]
```

df - report free disk space

df displays the amount of disk space available on the file system containing each file name argument.
- -T: --print-type. print file system type - –h: -human-readable. print sizes in human readable format - –l: --local. limit listing to local file systems - -i: --inodes list inode information instead of block usage

ln: Crea enlaces (links) a ficheros

```
ln [-fs] source_file target_file
ln [-fs] source_file ... target_dir
```

ln - link files

In the first synopsis form, the ln utility shall create a new directory entry (link) at the destination path specified by the target_file operand. If the -s option is specified, a symbolic link shall be created for the file specified by the source_file operand. - -s: Create symbolic links instead of hard links.

namei: Procesa una ruta de un fichero hasta encontrar el punto final

```
namei [options] pathname...
```

namei - follow a pathname until a terminal point is found

> namei uses its arguments as pathnames to any type of Unix file (symlinks, files, directories, and so forth). namei then follows each pathname until an endpoint is found (a file, a directory, a device node, etc). If it finds a symbolic link, it shows the link, and starts following it, indenting the output to show the context.

readlink: Lee el contenido de un link simbólico

```
readlink [OPTION]... FILE
```

> readlink - print value of a symbolic link or canonical file name
>
> Print value of a symbolic link or canonical file name

stat: Muestra información de control de un fichero

```
 stat [OPTION]... FILE...
```

> stat - display file or file system status
>
> - -Z: ???
> - -f: --file-system. display file system status instead of file status

lseek: Modifica la posición de lectura/escritura de un fichero

```
#include <unistd.h>
off_t lseek(int fildes, off_t offset, int whence);
```

> lseek — reposition read/write file offset
>
> The lseek() system call repositions the offset of the file descriptor fildes to the argument offset according to the directive whence. The argument fildes must be an open file descriptor. - SEEK_SET: If whence is SEEK_SET, the offset is set to offset bytes. - SEEK_CUR: If whence is SEEK_CUR, the offset is set to its current location plus offset bytes. - SEEK_END: If whence is SEEK_END, the offset is set to the size of the file plus offset bytes.