PRÁCTICA 0:
Introducción

PRELIMINARES 1: PASO DE PARÁMETROS DESDE LA LÍNEA DE COMANDOS A UN PROGRAMA EN C

```
main (int argc, char *argv[]) {
    int i;

    printf("Numero de argumentos: %d\n",argc);
    for (i=0;i<argc;i++) printf("Argumento %d: %s\n",i,argv[i]);
}
```

- La variable argc contiene el número total de argumentos pasados al programa, incluyendo el nombre del programa principal.
- El vector de cadenas argv contiene, en cada una de sus posiciones, los valores de dichos argumentos considerados como cadenas.

PRELIMINARES 2: COMPILACIÓN Y EJECUCIÓN DE UN PROGRAMA EN C BAJO UNIX

Compilación: cc fuente.c –o ejecutable

Ejecución: para ejecutar directamente el programa compilado sin errores, su directorio debe estar en el PATH del sistema operativo:

- PATH=$PATH:$HOME si nuestro directorio por defecto es el de arranque, si estamos trabajando en el directorio actual PATH=$PATH:.
- Ejecución directa: ./ejecutable argumentos

LLAMADAS AL SISTEMA

- Proporcionan la interfaz entre el Sistema Operativo y un programa en ejecución.
- UNIX permite realizar llamadas al Sistema Operativo desde un lenguaje de programación de alto nivel, en concreto, el lenguaje C.

El formato general de una llamada al sistema es:

status=llamada_sistema(argumentos)

Si la llamada al sistema resulta en un error:

- En **status** se almacena el valor –1.
- En la variable global **errno** se almacena un número de error.
- El fichero /usr/src/linux/include/asm/errno.h contiene los códigos.

Se puede obtener información sobre la sintaxis de cualquier llamada al sistema mediante el comando **man**.

CREACIÓN DE UN ARCHIVO

fd = **creat** (nombre_fichero, derechos)

- Crea el archivo nombre_fichero con los derechos especificados (número octal).
- Devuelve el descriptor del archivo fd: número entero que servirá para identificar al archivo a partir de ahora.

APERTURA DE UN ARCHIVO

fd = **open** (nombre_fichero, modo)

- Abre el archivo existente especificado, devolviendo su descriptor de archivo.
- Modo: 0 lectura, 1 escritura, 2 lectura/escritura.

LECTURA DE DATOS DE UN ARCHIVO

valor = **read** (fd, direccion_memoria, numbytes)

- Lee del archivo especificado mediante su descriptor tantos bytes como se especifica en el tercer parámetro y los coloca en una dirección de memoria.
- Si la lectura fue correcta: valor número de bytes realmente leídos.
- Si se llegó al final del fichero: valor 0.

ESCRITURA DE DATOS EN UN ARCHIVO

valor = **write** (fd, direccion_memoria, numbytes)

- Escribe en el archivo especificado mediante su descriptor tantos bytes como se especifica en el tercer parámetro, tomándolos de una dirección de memoria.
- Si la escritura fue correcta: valor número de bytes realmente escritos.

LAS OPERACIONES DE LECTURA Y ESCRITURA ACTUALIZAN LA POSICIÓN ACTUAL EN EL FICHERO

CIERRE DE UN ARCHIVO

valor = **close** (fd)

- Cierra el archivo especificado mediante su descriptor, con lo que el descriptor quedará libre pudiendo ser asignado a otro archivo.

EJEMPLO: Realizar un programa en C mediante llamadas al sistema que admita como parámetros un fichero origen (suponemos que ya existe) y un fichero destino (suponemos que no existe). El programa realizará la copia del fichero origen en el fichero destino.

```
char buffer[512];

main (int argc, char *argv[]) {
    int fd1, fd2, leidos;


    if (argc!=3) {
    printf("Sintaxis: %s <origen> <destino>\n",argv[0]);
    exit(-1);
    }

    fd1=open(argv[1],0);
    if (fd1==-1) {
    perror("Fichero origen");
    exit(-1);
    }

    fd2=creat(argv[2],0777);
    if (fd2==-1) {
    perror("Fichero destino");
    exit(-1);
    }

    while(1) {
    leidos=read(fd1,buffer,sizeof(buffer));
    if (leidos==0) break;
    write(fd2,buffer,leidos);
```

```
    }

    close(fd1);
    close(fd2);
}
```