# Archivos Stream

- Definición
- Utilización en C

# Archivos Stream - Características

- Archivo sin estructura de registros.
- Puede ser visto como una tira de bytes.
- Permite la implementación de registros de longitud variable.
- Son manejados por lenguajes como Pascal, Ada y C.
- Acceso secuencial o random
- Entrada/Salida por bloques.

# Archivos Stream - Manejo de registros de longitud variable

Formas de implementación :

• Byte separador, con valor especial. (dicho valor no debe ser un valor posible de los datos almacenados)

• Almacenar la longitud del registro al comienzo del mismo.

• Almacenar la longitud del registro en otro archivo.

# Archivos Stream - C

En C hay 2 formas de manejar un archivo :

- Utilizando Rutinas de Bibliotecas

  Utilizamos rutinas preescritas, que ocultan algunos parámetros, pero en definitiva utilizan system calls.
  (las rutinas se encuentran en el archivo stdio.h)
  La entrada/salida se realiza en registros

- Utilizando System Calls

  Utilizamos directamente servicios del Unix.

  La entrada/salida se realiza en bytes

# Archivos Stream - Operaciones en C (con rutinas)

## Declaración

- FILE * puntero_al_archivo;

## Apertura

- FILE * fopen(char * nombre_externo,char * modo);

Los modos pueden ser :

| | |
|---|---|
| r : lectura solamente | w : escritura solamente |
| a : escritura al final | + : se agrega a r y w |
| b : modo binario | t : modo texto |

# Archivos Stream - Operaciones en C (con rutinas)

## Lectura

• unsigned fread(void * bufferp, unsigned tamaño, unsigned cantidad, FILE * puntero_al_archivo);

## Escritura

• unsigned fwrite(void * bufferp, unsigned tamaño, unsigned cantidad, FILE * puntero_al_archivo);

En ambos casos la cantidad pedida y cumplida es en registros

# Archivos Stream - Operaciones en C (con rutinas)

## Posicionamiento

• int fseek(FILE * puntero_al_archivo, long offset, int lugar_desde);

Los valores de lugar_desde pueden ser :

SEEK_SET : desde el principio
SEEK_END : desde el final

SEEK_CUR : desde la posición actual

## Cierre

• int fclose(FILE * puntero_al_archivo);

# Archivos Stream - Operaciones en C (con system calls)

## Apertura

• int open(char * nombre_externo, int flag_apertura [,int permisos]

Los valores del flag_apertura pueden ser :

O_READ: lectura solamente  O_WRITE: escritura solamente

O_APPEND: escritura al final  O_RDWRITE: lectura/escritura

O_CREATE : crea el archivo  O_EXCL : crea si no existe (sino da error)

O_TRUNC : si el archivo existe, lo trunca.

Los 3 últimos pueden conbinarse con los 4 primeros con el uso del pipe (|)

# Archivos Stream - Operaciones en C (con system calls)

## Apertura (Continuación)

**Permisos :** (se utilizan solo en la creación)

• Se representan con un número de 3 dígitos octales.

• El primer dígito contiene los permisos del usuario (dueño del archivo), el segundo los del grupo al que pertenece y el tercero al resto de los usuarios.

• Cada dígito se obtiene del valor en binario de cada uno de los 3 bits que lo componen.

• Si el bit está en 1 indica que el permiso está otorgado, si está en 0 es denegado.

• Los permisos que se otorgan son para lectura (R), escritura(W) y ejecución (X), siempre en ese orden.

# Archivos Stream - Operaciones en C (con system calls)

## Lectura

• int read(int filedescriptor, void * buffer, unsigned cantidad);

## Escritura

• int write(int filedescriptor, void * buffer, unsigned cantidad);

En ambos casos la cantidad pedida y cumplida es en bytes

# Archivos Stream - Operaciones en C (con system calls)

## Posicionamiento

• int lseek(int filedescriptor, long offset, int lugar_desde);

Los valores de lugar_desde pueden ser :

0 : desde el principio  2 : desde el final

1 : desde la posición actual

## Cierre

• int close(int filedescriptor);

# Archivos Stream - Operaciones en C (con system calls)

## Control de errores

- int errno (contiene el resultado de la última operación de entrada/salida)

- void perror (char * mensaje); (emite el string seguido por una coma y el código de error)