LABORATORIO SISTEMI OPERATIVI: SYSTEM FILES

- **STEAM**:
  - Le **modalità d'accesso al file system** sono caratterizzate da un **flusso continuo** di accesso alle informazioni (mette a disposizione delle formattazioni e dei buffer).
  - L'**accesso** avviene **mediante puntatori di tipo FILE**.
  - I dati sono letti e scritti in modi differenti e possono essere manipolati al volo.

```c
#include <stdio.h>


int main(){
  FILE *fp;
  int c;
  printf("Esempio stream\n");
  fp = fopen("stream.c", "r"); //syscall che apre un file in modalita' read (r)

  while(1){
    c = fgetc(fp); //funzione che dato un riferimento ad un file, leggere un carattere (e continua)

    //funzione che verifica se il corsore del file (fp) non ha raggiunto la fine del file
    if(feof(fp)) break; //se raggiungo la fine, esco dal while

    printf("%c", c);
  };

  fclose(fp);
  printf("Fine.\n");

  return 0;
}
```

- **DESCRIPTORS**:
  - Sono **un'altra modalità d'accesso al file system** (accesso a più **basso livello**, attraverso poche funzioni specifiche).
  - I **dati** vengono **letti e scritti un buffer alla volta** (dimensione prefissata).
  - I **file sono riferiti mediante un indice numerico** (il riferimento è salvato in una variabile di tipo INT), questo indice servirà per accedere al file stesso:
    - È un **riferimento ad una entry di una tabella** particolare che viene usata per accedere al file e per tutte le operazioni su di esso.

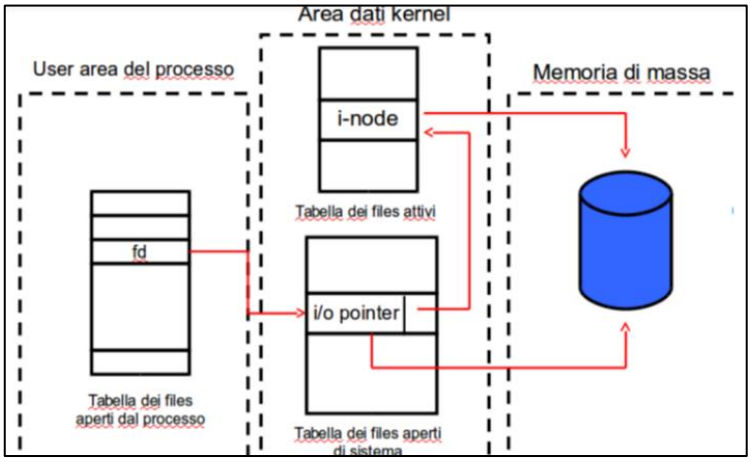
  - **Ogni processo in esecuzione ha la sua tabella** dove **salva la posizione di tutti i file aperti** (ogni riga equivale ad un file).
  - Di default ci sono **3 descrittori aperti ed attivi** (i primi 3 record della tabella):
    - **0 stdin**: standard input (collegato alla tastiera).
    - **1 sdout**: standard output (collegato a video).
    - **2 stderr**: standard error (collegato a video).

  - **Funzioni**:
    - **open(): Crea un canale di comunicazione**, inizializza la tabella di riferimento e restituisce un indice che utilizzeremo per riferirci al file.
    - **read():** Permette la **lettura** dei dati.
    - **write():** Permette di **scrivere** dati nel file.
    - **lseek():** Permette di **manipolare l'indice di lettura/scrittura** (posso muovere l'indice).
    - **close():** Funzione che **libera le risorse** e **chiude** il **canale** di comunicazione con il file.

  - La **syscall open()** quando viene invocata **apre il canale verso il file** cercando l'i-node:
    - Cerca la **posizione effettiva** (fisica) del file stesso **nella memoria**.
    - Questa informazione viene **memorizzata in una tabella** che contiene un elenco di entry che tengono traccia della posizione fisica sul disco di tutti i file aperti.
    - **L'entry della tabella viene restituita** ed è l'indice che verrà usato per riferirsi al file.

- **TABLES**:
  - Nel User Space, **ogni processo ha la sua tabella**.
  - Si hanno altre **due tabelle principali** (generali) salvate nel **Kernel Space**:
    - **FILE ATTIVI**:
      - Contiene un **record per ogni file attivo** (utilizzato da un processo).
      - Questo record contiene una serie di informazioni riguardo all' i-node.
      - Se un processo accede ad un nuovo file, questa tabella (oltre a quella del processo) viene aggiornata con una nuova entry, in modo da salvare la sua posizione fisica sul disco.

    - **OPERAZIONI**:
      - Tabella delle operazioni, si ha una **entry per ogni singola operazione effettuata** su di un file da un particolare processo (operazioni di lettura/scrittura).
      - Contiene:
        - **I/O pointer**: Indica la **posizione** dove è avvenuta/avviene/avverrà la lettura/scrittura.
        - **Riferimento alla tabella dei file attivi.**



ESERCIZIO CALCOLA DIMENSIONE FILE

```c
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>

int main(){
  int fd; //indice file
  //funzione per riservare una zona di memoria (inizializzata a 0)
  //viene allocata una zona di memoria formata da 100 unita', ognuna che occupa 1B (dimensione di char)
  char *txt = (char *) calloc(100, sizeof(char)); //riserva 100 Byte
  int rd; //read

  fd = open("fd.c", O_RDONLY); //file aperto in lettura
  if(fd<0) exit(1);

  printf("FD=%d\n", fd);

  //qua bisogna stare attenti... nel nostro caso vengono stampati 10 caratteri, perche'
  //abbiamo inizializzato la zona di memoria txt con calloc() che mette tutto a 0
  //quindi quando stampo, alla 11 cella viene trovato uno 0 che fa da terminatore di stringa
  //printf("txt=%s\n", txt);


  //la funzione read() restituisce il numero di caratteri letti (byte letti)
  while(rd=read(fd, txt, 10)){
    if(rd>0){
       //SOLUZIONE: porre alla fine il terminatore di stringa
       //txt[10]=0; per forzare ma non funziona sempre bene
       txt[rd]=0; //migliore, perche' se leggo meno caratteri non ho problemi (MOLTO IMPORTANTE)
       printf("%s", txt);
       //printf("RD=%d\n", rd);
    }else{
      break;
    };
  };
  printf("\n");

  close(fd);
  return 0;
}
```