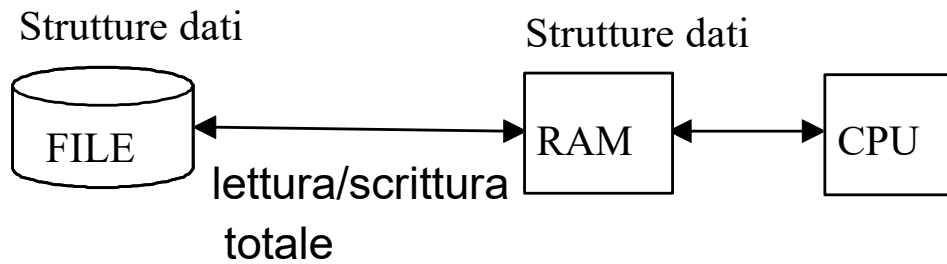


# FILE E APPLICAZIONI

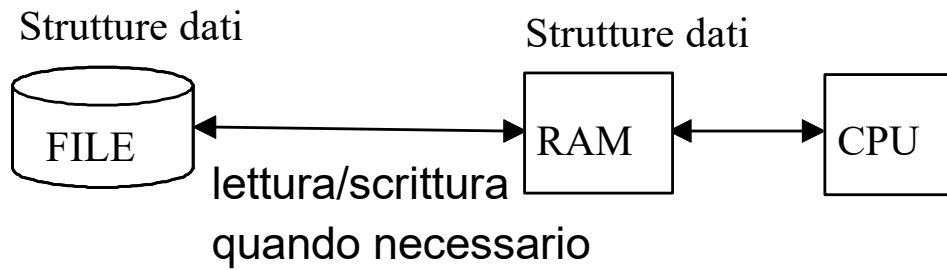
- definire strutture di memoria centrale di supporto
- definire la strutturazione del file

## Struttura RAM di supporto

- file passivo

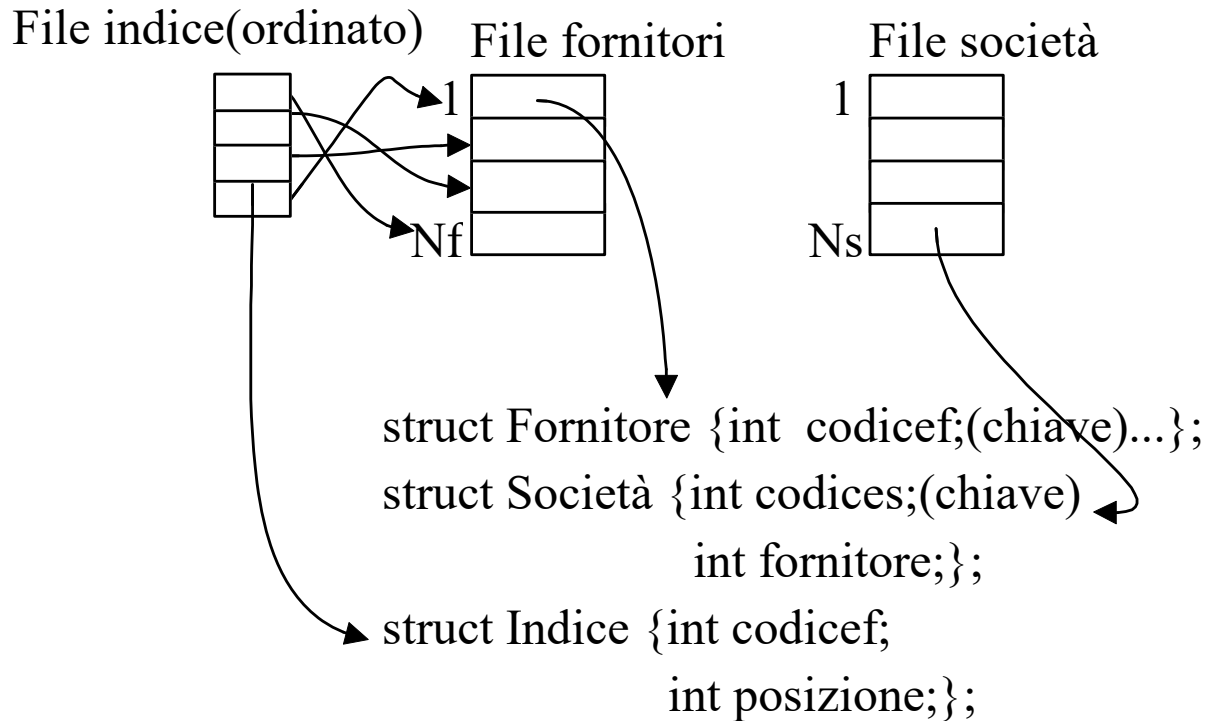


- file attivo



**Esempio 1:** Applicazione per la gestione di due archivi: fornitori e società; una società può avere un solo fornitore, mentre un fornitore può fornire più società.

Strutturazione file:



### 1. Trovare la società con codices = xx; (codices è chiave)

Scansione sequenziale file società

```

int trovato=0;
LeggiSoc(filesocieta, rec);
while (!feof(filesocieta) && (trovato==0))
  
```

```

  {if (var.codices == xx) trovato=1;
    LeggiSoc(filesocieta, rec);
  }
  
```

- costo lettura file:  $(N_f / 2) * \text{sizeof}(\text{rec})$ ; (equiprobabilità)
- Supporto RAM: memoria per 1 record

## 2. Trovare le società con fornitore = xx; (fornitore non chiave)


Scansione sequenziale file società:

```
LeggiSoc(filesocieta, rec);
while (!feof(filesocieta))
{if (var.fornitore == xx) print (rec);
  LeggiSoc(filesocieta, rec);
}
```

- costo lettura file:  $N_f * \text{sizeof}(\text{rec})$
- Supporto RAM: memoria per 1 record

## 3. Trovare fornitore con codicef = xx; (codicef è chiave)

Scansione sequenziale via indice (filefornitore, fileindice):

```
LeggiIndex(fileindice, ix);
while (!feof(fileindice))
{if (ix.codicef == xx)
 {lseek(filefornitore, ix.posizione);
  LeggiForn(filefornitore, forn);
  return (forn);
}
  LeggiIndex(fileindice, ix);
}
```

- costo lettura file: (in RAM:  $(N_f / 2) * \text{sizeof}(\text{ix}) + 1 * \text{sizeof}(\text{forn})$ )
- Supporto RAM: memoria per 1 record indice e fornitore

#### 4. Stampare i dati dei fornitori e delle società fornite (algoritmo “nested loop”)

```
LeggiForn(filefornitore, recf);
while (!feof(filefornitore))
{
    rewind(filesocieta);
    LeggiSoc(filesocieta, recs);
    while (!eof(filesocieta))
        {if (recf.codicef==recs.fornitore) print(...);
        LeggiSoc(filesocieta, recs);
        }
    LeggiForn(filefornitore, varf);
}
```

- Costo lettura file:  $N_f * \text{sizeof}(\text{recf}) + N_f * N_s * \text{sizeof}(\text{recs})$
- supporto RAM: 1 record per fornitore e società

(algoritmo “nested block”)

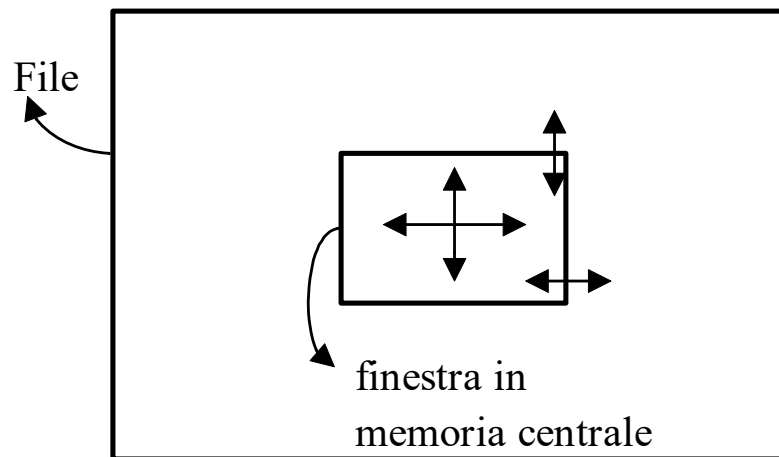
Ipotesi:

- $N_f \leq N_s$
- $N_f = K * B$ :

```
LeggiBForn(filefornitore, vet[B]);
while (!feof(filefornitore))
{
    rewind(filesocieta);
    LeggiSoc(filesocieta, recs);
    while (!eof(filesocieta))
        {itera sui B recf di vet(B)
        if (vet[i].codicef==recs.fornitore) print(..);
        LeggiSoc(filesocieta, vars);
        }
}
```

- costo:  $N_f * \text{sizeof}(\text{recf}) + (N_f / B) * N_s * \text{sizeof}(\text{recs})$ ;
- supporto RAM: 1 record per società e B record per fornitore

## Esempio 2. Gioco del labirinto.



Caricamento  
casella prossima  
mossa

↑↑  
file attivo

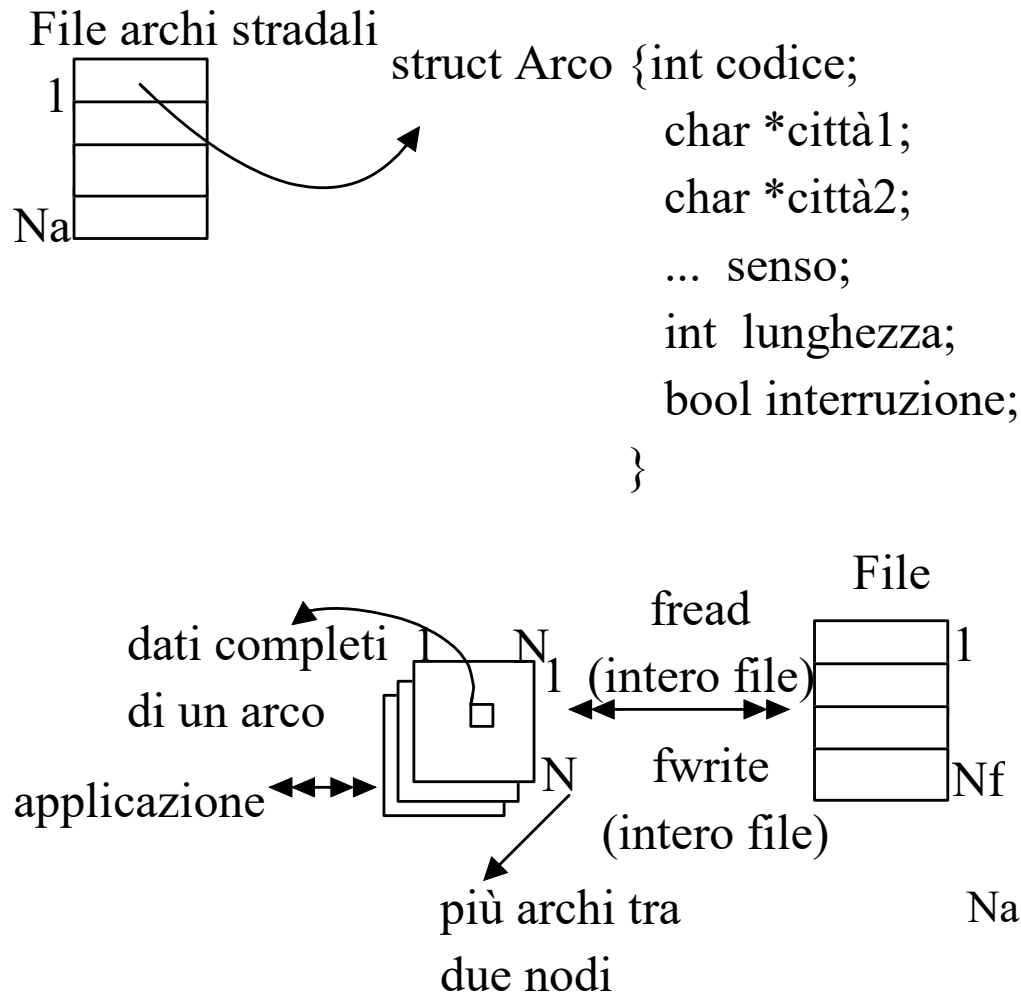
caricamento  
sottomatrice

↑↑  
file attivo

caricamento  
intera matrice

↑↑  
file passivo

### Esempio 3 . Gestione di una rete stradale.



#### 1. Connessione del grafo stradale (matrice di adiacenza)

BOOLEAN adiacenza [N,N]=FALSE;

nodo i connesso a nodo j

implica che  $adiacenza[i,j] = adiacenza[j,i] = TRUE$

#### 2. Percorso minimo tra due nodi:

int matrice [N,N]= ∞;

matrice[i,j] = lunghezza minima degli archi orientati dal nodo i verso il nodo j (se esiste)

#### 3. Aggiornamenti dei dati della rete

L'aggiornamento è tipicamente un'operazione che coinvolge un arco alla volta.