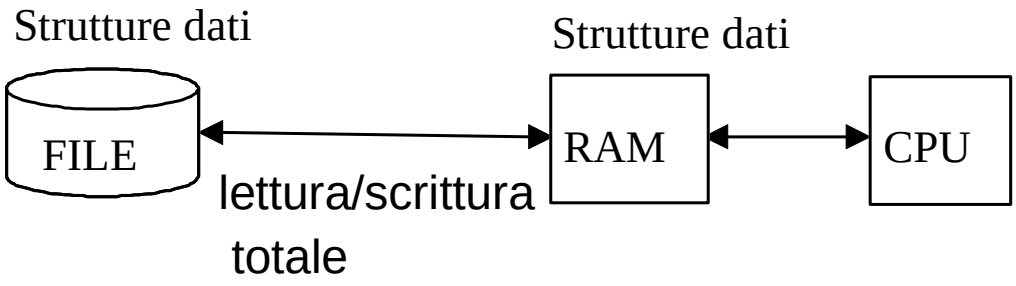
FILE E APPLICAZIONI

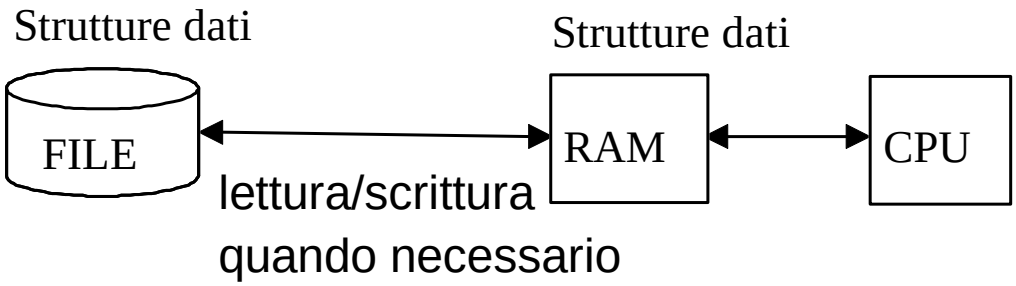
- definire strutture di memoria centrale di supporto
- definire la strutturazione del file

Struttura RAM di supporto

- file passivo

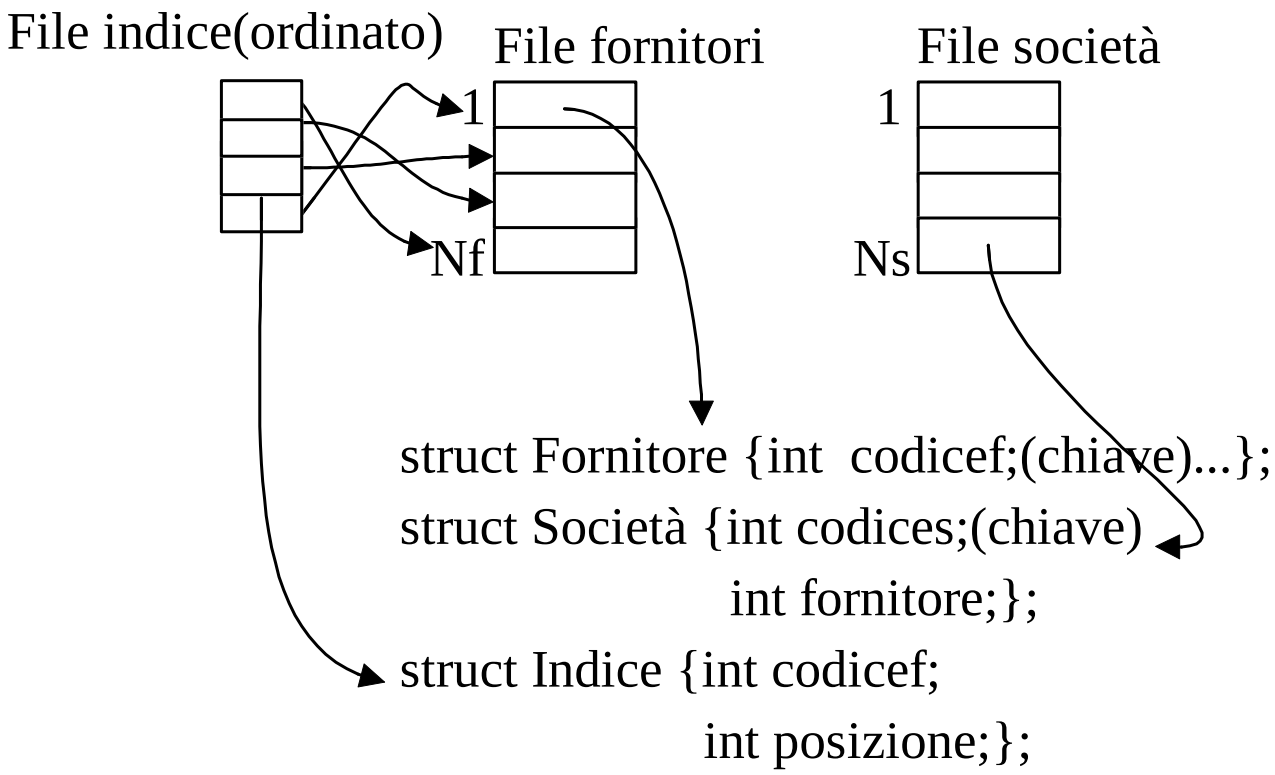


- file attivo



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:

```
LeggiRecSoc(filesocieta, var);
while (!feof(filesocieta))
  {if (var.codices == xx) return (var);
    LeggiRecSoc(filesocieta, var);
   }
```

- costo lettura file: (Nf / 2)*sizeof(varf); (equiprobabilità)
- Supporto RAM: memoria per 1 record

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

Scansione sequenziale file società:

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

- costo lettura file: Nf * sizeof(varf)
- Supporto RAM: memoria per 1 record

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

Scansione sequenziale via indice (filefornitore,fileindice):

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

- costo lettura file: (Nf / 2)*sizeof(varix) + 1* sizeof(varf)
- Supporto RAM: memoria per 1 record indice e fornitore

4. Stampare i dati dei fornitori e delle società fornite

(algoritmo “nested loop” – supporto RAM: 1 record per fornitore e società)

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

- Costo lettura file: Nf * sizeof(varf) + Nf * Ns * sizeof(vars)

(algoritmo “nested block” –supporto RAM: 1 record per società e B record per fornitore)

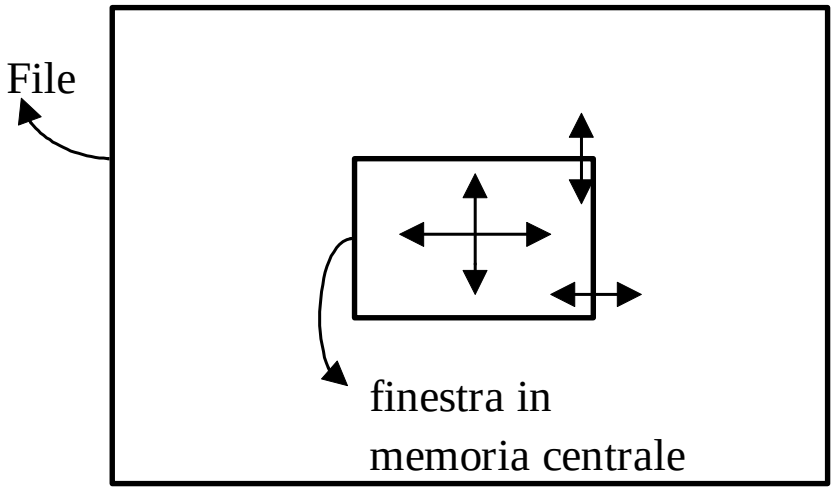
Ipotesi:

- Nf <= Ns
- Nf =K * B):

```
LeggiBForn(filefornitore, vettore[B]);
while (!feof(filefornitore))
{rewind(filesocieta)
  LeggiSoc(filesocieta, vars);
  while (!eof(filesocietà))
 {for (i=0; i < B; i++)
    if (vettore[i].codicef==vars.fornitore) print(..);
    LeggiSoc(filesocieta, vars);
 }
}
```

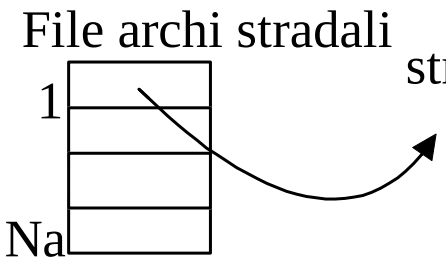
- costo: Nf * sizeof(varf) + (Nf /B) * Ns * sizeof(vars);

Esempio 2. Gioco del labirinto.

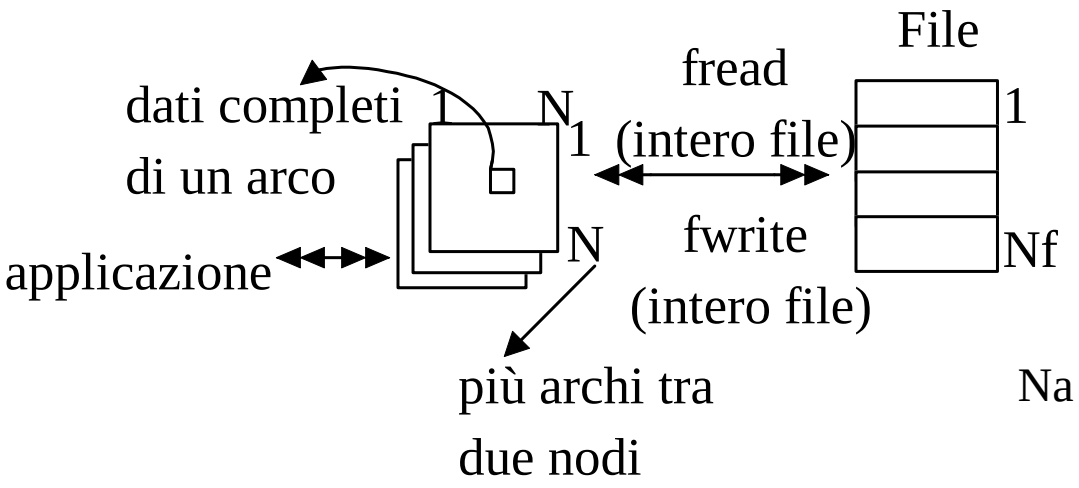


Caricamento casella prossima mossa	caricamento sottomatrice	caricamento matrice
⇑	⇑	⇑
file attivo	file attivo	file passivo

Esempio 3 . Gestione di una rete stradale.




```
struct Arco {int codice;
             char *città1;
             char *città2;
             ...  senso;
             int  lunghezza;
             bool interruzione;
            }
```



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.