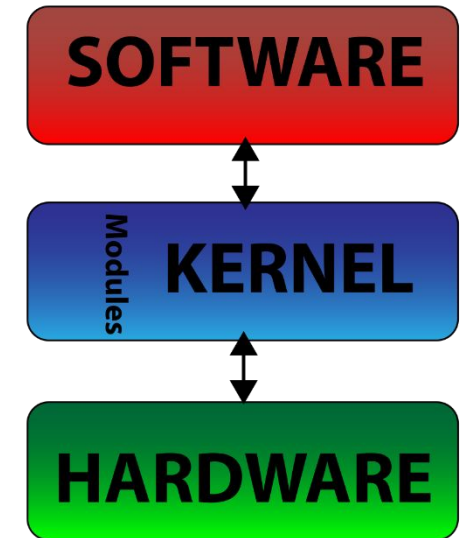




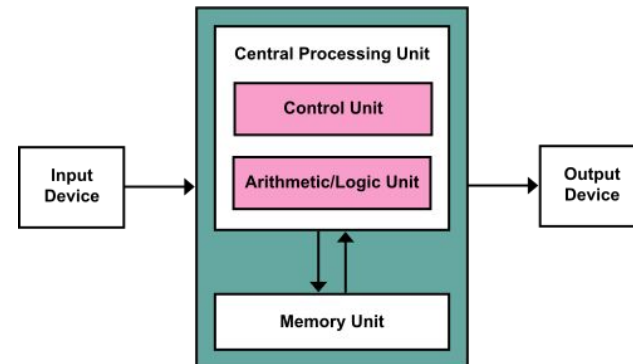
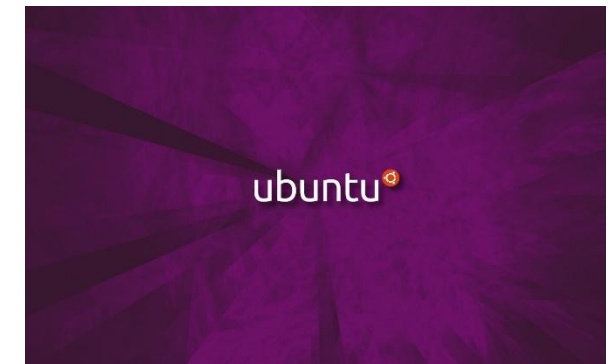
**UNIVERSITÀ DEGLI STUDI  
DELLA BASILICATA**

## *Corso di Sistemi Operativi*

# Esercitazione File System

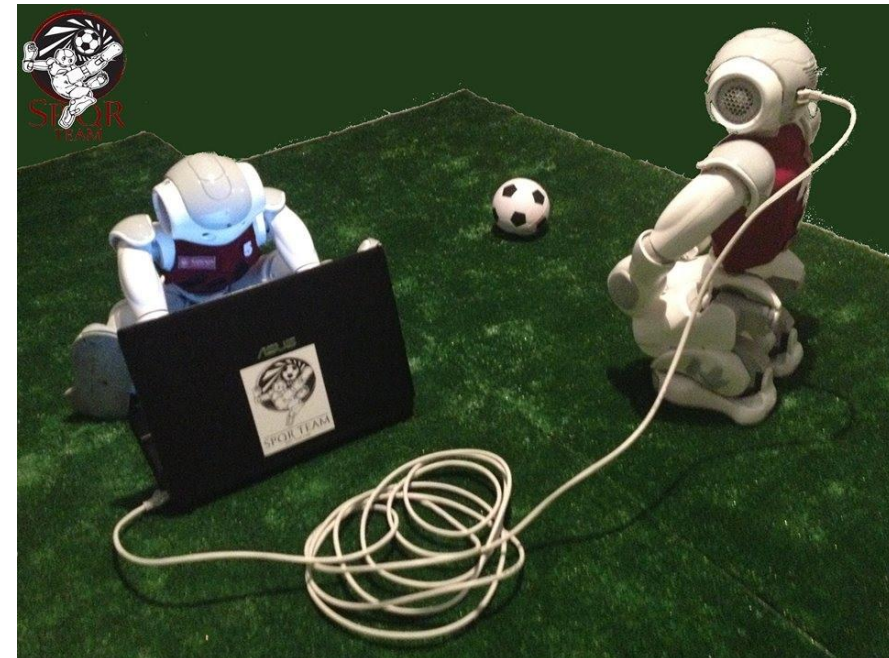
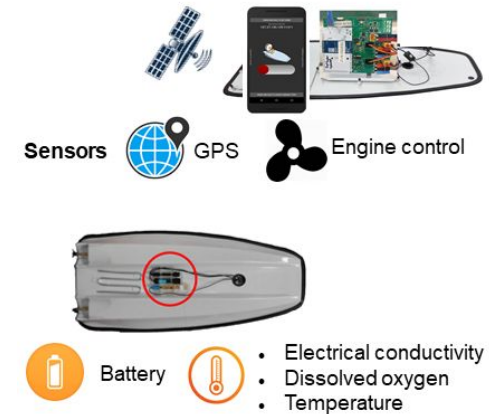


Docente:  
Domenico Daniele  
Bloisi



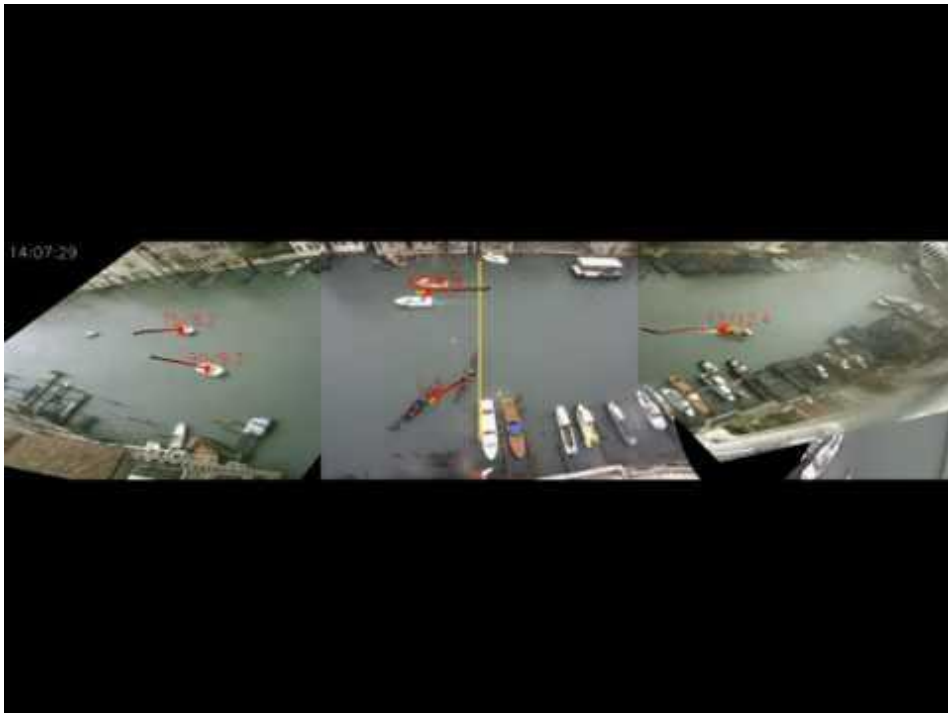
# Domenico Daniele Bloisi

- Professore Associato  
Dipartimento di Matematica, Informatica  
ed Economia  
Università degli studi della Basilicata  
<http://web.unibas.it/bloisi>
- SPQR Robot Soccer Team  
Dipartimento di Informatica, Automatica  
e Gestionale Università degli studi di  
Roma “La Sapienza”  
<http://spqr.diag.uniroma1.it>



# Interessi di ricerca

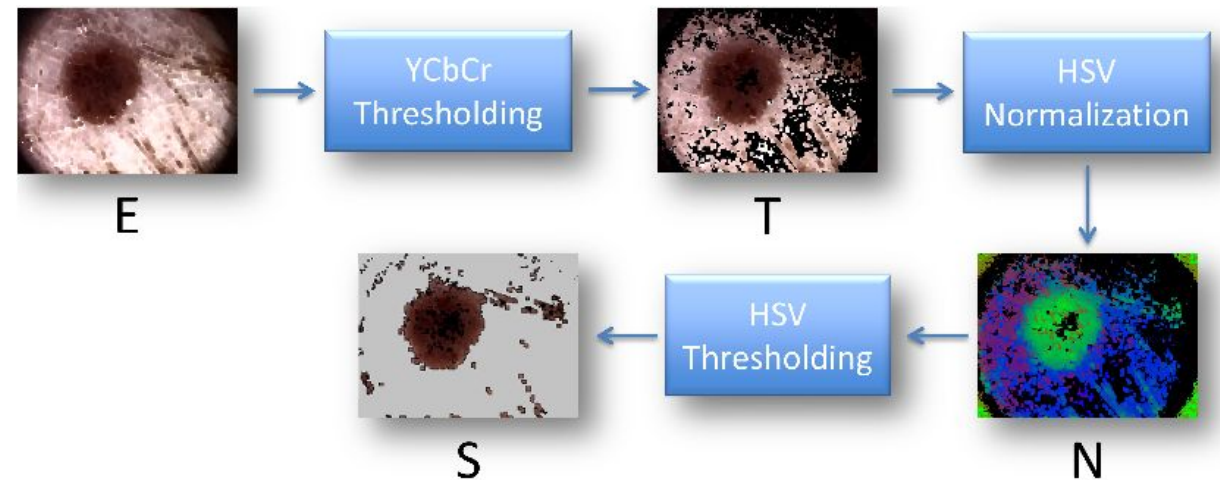
- Intelligent surveillance
- Robot vision
- Medical image analysis



[https://youtu.be/9a70Ucgbi\\_U](https://youtu.be/9a70Ucgbi_U)



<https://youtu.be/2KHNZX7UIWQ>





# UNIBAS Wolves <https://sites.google.com/unibas.it/wolves>



- UNIBAS WOLVES is the robot soccer team of the University of Basilicata. Established in 2019, it is focussed on developing software for NAO soccer robots participating in RoboCup competitions.

- UNIBAS WOLVES team is twinned with SPQR Team at Sapienza University of Rome



<https://youtu.be/ji0OmkaWh20>

# Informazioni sul corso

---

- Home page del corso:  
<http://web.unibas.it/bloisi/corsi/sistemi-operativi.html>
- Docente: Domenico Daniele Bloisi
- Periodo: I semestre ottobre 2022 – gennaio 2023
  - Lunedì dalle 15:00 alle 17:00 (Aula Leonardo)
  - Martedì dalle 08:30 alle 10:30 (Aula 1)

# Ricevimento

---

- In presenza, durante il periodo delle lezioni:  
Lunedì dalle 17:00 alle 18:00  
presso Edificio 3D, Il piano, stanza 15  
**Si invitano gli studenti a controllare regolarmente la bacheca degli avvisi per eventuali variazioni**
- Tramite google Meet e al di fuori del periodo delle lezioni:  
da concordare con il docente tramite email

Per prenotare un appuntamento inviare  
una email a  
[domenico.bloisi@unibas.it](mailto:domenico.bloisi@unibas.it)



# Domanda 1

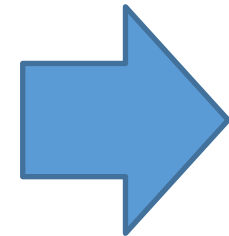
---

- Cosa è il File Control Block (FCB)?
- Quali sono le informazioni contenute al suo interno?
- Integrare la spiegazione con opportuni schemi grafici.

# Risposta Domanda 1

---

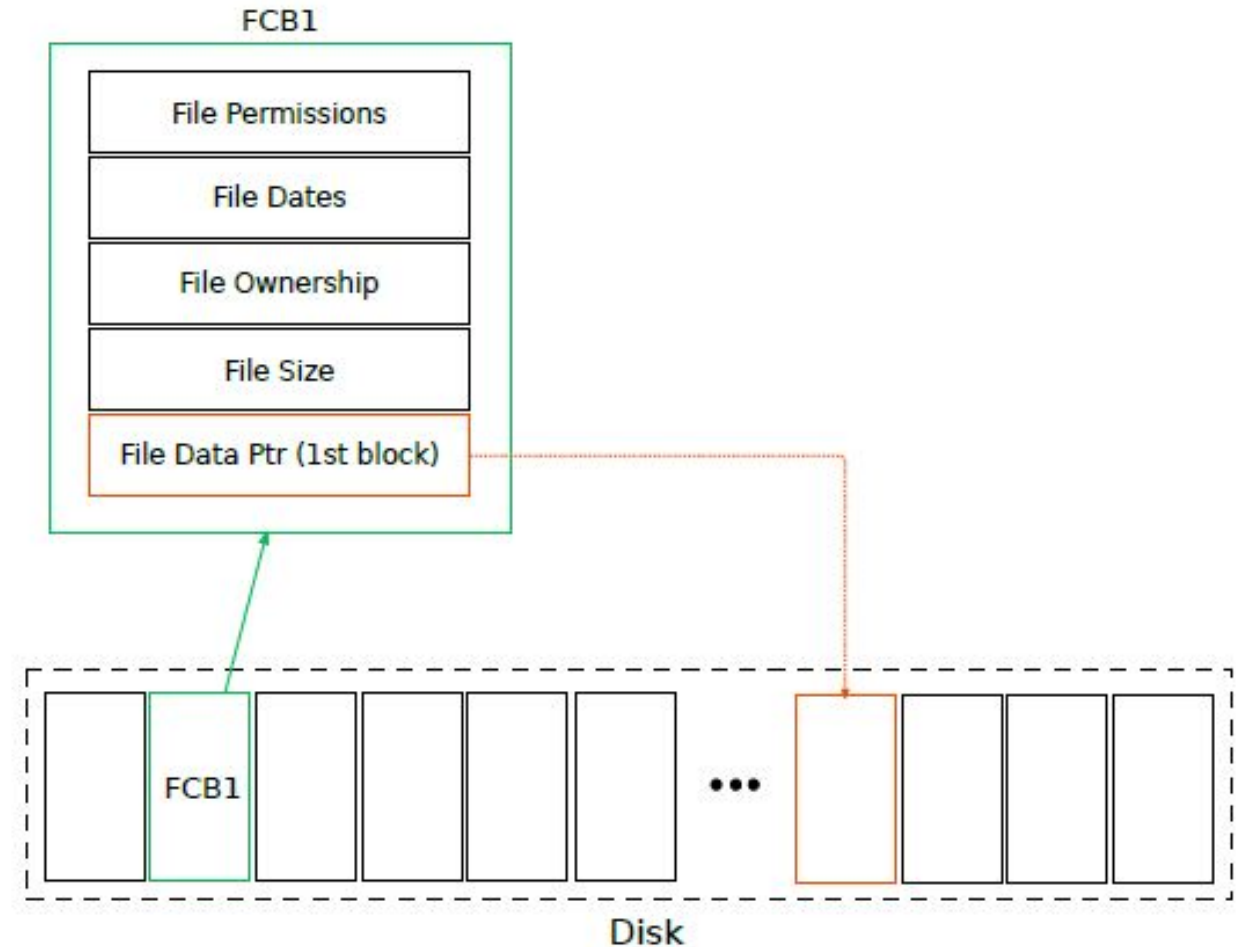
- Il FCB è una struttura dati che contiene tutte le informazioni relative al file a cui essa è associata.
- Esempi di informazioni possono essere: permessi, dimensione, data di creazione, ecc.
- Inoltre, il FCB contiene informazioni sulla locazione sul disco dei dati del file (ad esempio, in un File System con allocazione concatenata, il puntatore al primo blocco del file).





# Risposta Domanda 1

- Schema generale per il **FCB**.
- La sua struttura contiene tutti gli attributi del file, compresa la locazione dei dati. Quest'ultima è rappresentata nello schema dal puntatore al primo blocco della lista contenente i dati (supponendo che il file system considerato adotti una allocazione concatenata).



# Domanda 2

---

Che cos'è un inode?

# Risposta Domanda 2

- Un inode è una struttura dati per memorizzare i metadati del file system usata in UNIX.
- Il nome del file è associato ad un numero di inode. L'inode corrispondente contiene le informazioni necessarie per individuare lo spazio allocato.
- La struttura di un inode è mostrata nella figura di fianco.

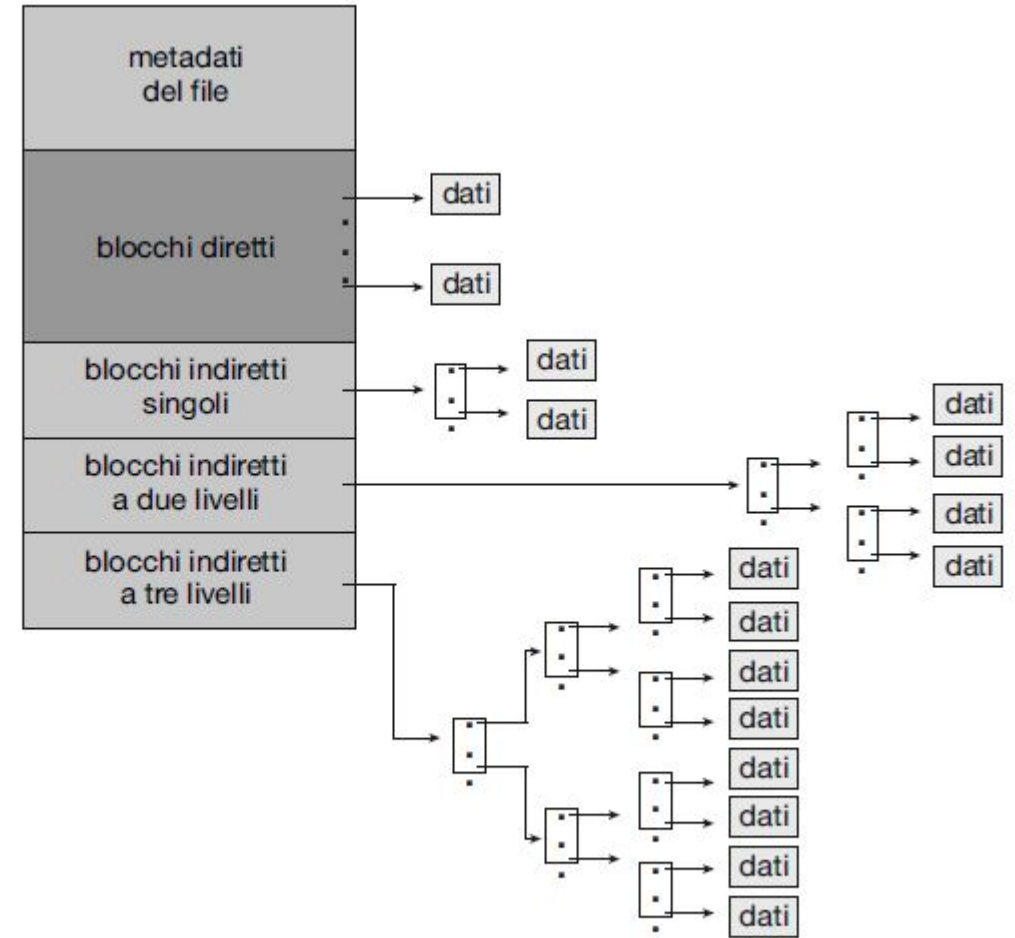


Figura 14.8 Inode di UNIX.

# Domanda 3

---

Spiegare brevemente la differenza tra `fopen()` e `open()`

# Risposta Domanda 3

---

- `fopen()` è una **funzione di alto livello** che restituisce uno STREAM

```
FILE* fopen(const char* pathname, const char* mode);
```

- `open()` è una **system call di basso livello** che restituisce un file descriptor (intero)

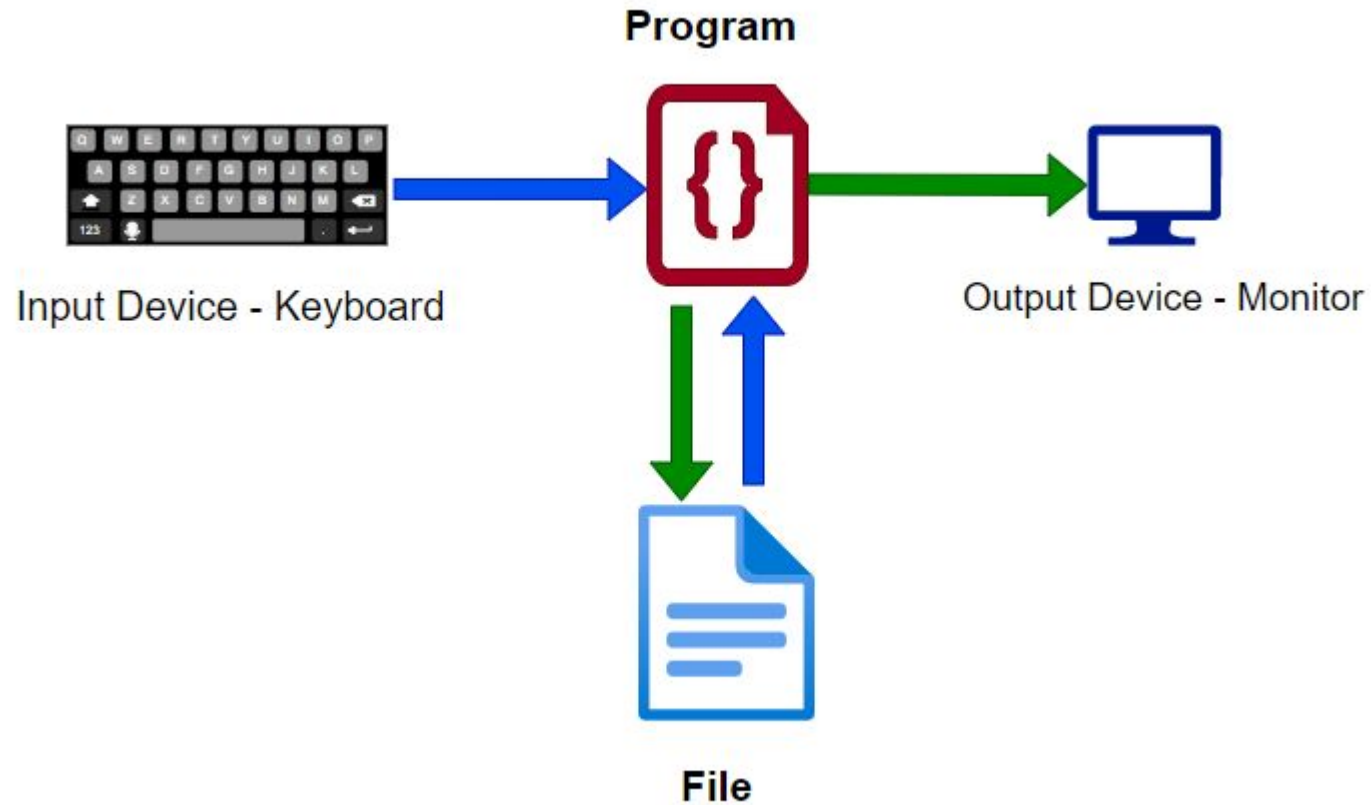
```
int open(const char* pathname, int flags);
```

- Si noti che `fopen()` contiene nella sua implementazione una chiamata alla system call `open()`



# Risposta Domanda 3

---



# Risposta

---

## Domanda 3

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <errno.h>
#include "local.h"

FILE *
fopen(file, mode)
    const char *file;
    const char *mode;
{
    register FILE *fp;
    register int f;
    int flags, oflags;

    if ((flags = __sflags(mode, &oflags)) == 0)
        return (NULL);
    if ((fp = __sfp()) == NULL)
        return (NULL);
    if ((f = open(file, oflags, DEFFILEMODE)) < 0) {
        fp->_flags = 0;           /* release */
        return (NULL);
    }
    fp->_file = f;
    fp->_flags = flags;
    fp->_cookie = fp;
    fp->_read = __sread;    ■ ■ ■
```

# Domanda 4

---

Fornire esempi di applicazioni che accedono ai file utilizzando i seguenti metodi

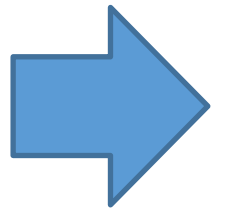
- Accesso sequenziale
- Accesso casuale

# Risposta Domanda 4

---

Esempi di applicazioni che accedono ai file in modalità sequenziale sono:

- Word processor
- Video player
- Audio player
- Web server



# Risposta Domanda 4

---

Esempi di applicazioni che accedono ai file in modalità diretta sono:

- Video editor
- Audio editor
- Database



# Domanda 5

---

Se il sistema operativo sapesse che un applicativo aprirà un file in modalità sequenziale, come potrebbe sfruttare questa informazione per migliorare le performance?

# Risposta Domanda 5

---

Nel momento in cui un blocco viene utilizzato dal processo utente, il sistema operativo può pre-caricare (prefetch) i blocchi successivi a quello in uso. Questo permette, nel caso in cui effettivamente i blocchi pre-caricati saranno effettivamente richiesti nel futuro, di ridurre il tempo di attesa del processo utente.

# Domanda 6

---

Quali sono i vantaggi e quali gli svantaggi della modalità di allocazione contigua dei file in memoria secondaria?

# Risposta Domanda 6

## Vantaggi

- Maggiore velocità possibile nella lettura dei contenuti del file
- Accesso sequenziale e diretto molto efficiente

## Svantaggi

- Frammentazione esterna
- Problemi nella gestione della crescita di dimensione dei file

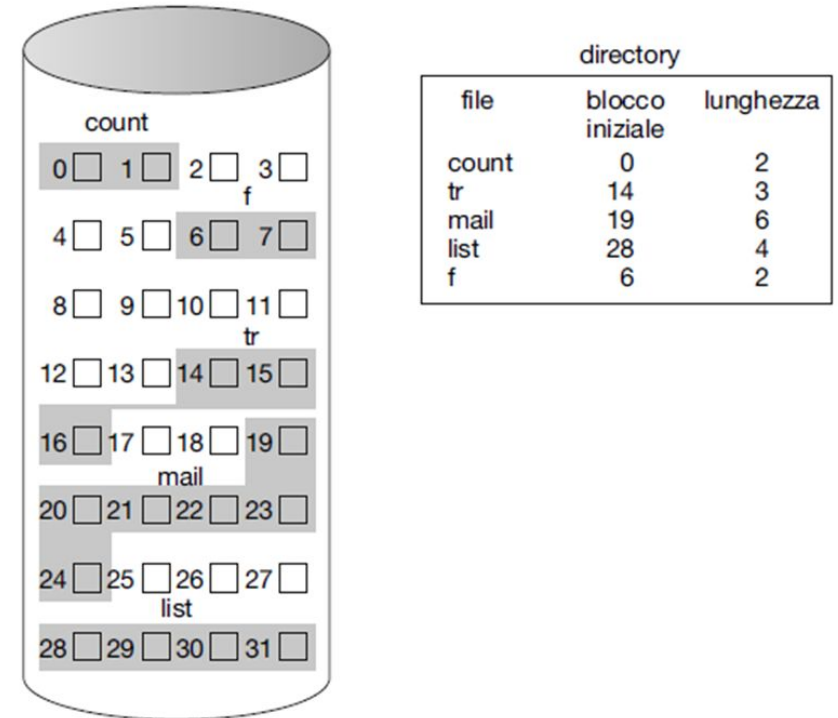


Figura 14.4 Allocazione contigua dello spazio dei dischi.

# Domanda 7

---

Quali sono i vantaggi e quali gli svantaggi della modalità di allocazione concatenata dei file in memoria secondaria?



# Risposta Domanda 7

## Vantaggi

- Facile creare, ridurre e far crescere la dimensione dei file
- Assenza di frammentazione esterna

## Svantaggi

- Impossibilità di avere un accesso diretto ai dati
- Affidabilità ridotta (cosa succede se si perde una connessione?)

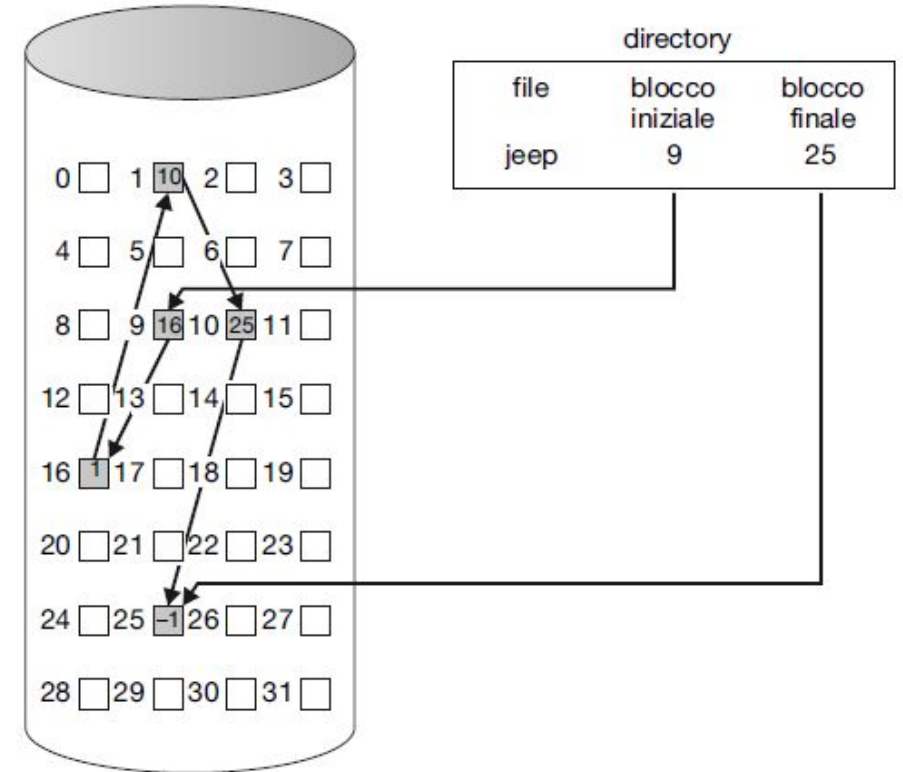


Figura 14.5 Allocazione concatenata dello spazio dei dischi.

# Domanda 8

---

Quali sono i vantaggi e quali gli svantaggi della modalità di allocazione indicizzata dei file in memoria secondaria?

# Risposta Domanda 8

## Vantaggi

- Facile creare, ridurre e far crescere la dimensione dei file
- Frammentazione esterna contenuta
- Possibilità di supporto per accesso diretto

## Svantaggi

- Overhead per file di ridotte dimensioni
- Difficoltà nella gestione di file di grossa dimensione

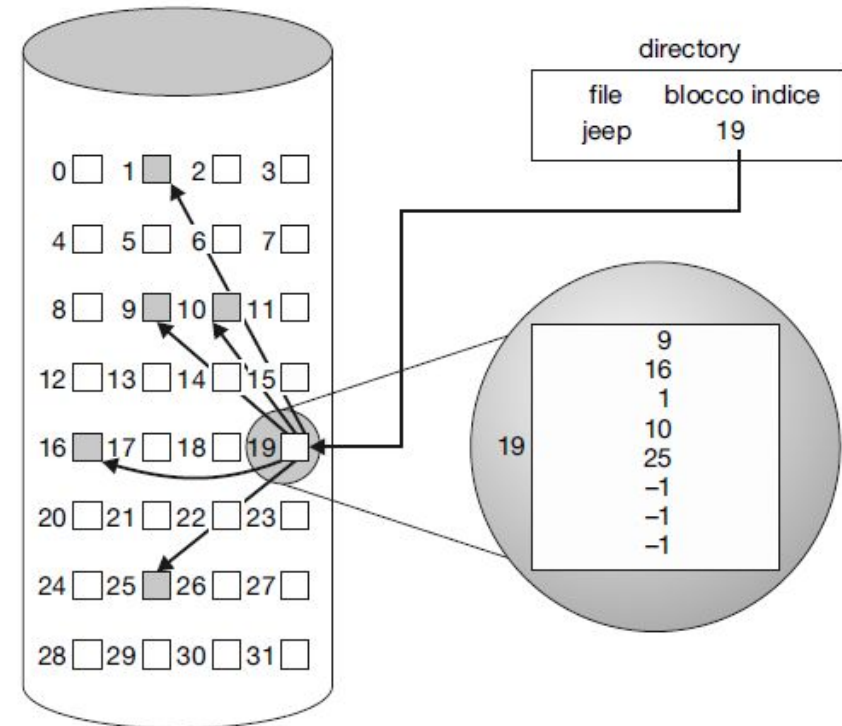


Figura 14.7 Allocazione indicizzata dello spazio dei dischi.

# Esercizio 1

---

- Si consideri un file di grandezza pari a 100 blocchi avente il suo File Control Block (FCB) già caricato in memoria.
- Siano dati due file system, gestiti rispettivamente tramite allocazione concatenata e allocazione contigua.
- Si assuma che nel caso di allocazione concatenata, eventuale spazio per estendere il file sia disponibile solo alla fine dello stesso - non all'inizio.

Calcolare il numero di operazioni di I/O su disco (IO-ops) necessarie per eseguire le seguenti azioni in entrambi i file-system:

1. Rimozione di un blocco all'inizio del file
2. Rimozione di un blocco a metà del file (51° blocco)
3. Rimozione di un blocco alla fine del file

# Soluzione Esercizio 1

1. Rimozione di un blocco all'inizio del file con **allocazione concatenata**

Totale numero di operazioni di I/O su disco:  
1 IO-ops

Con una allocazione concatenata andremo ad accedere al primo blocco (1 IO-ops) per recuperare la posizione del secondo blocco. Tale posizione verrà scritta nel FCB (nessun IO-ops, il FCB è già in memoria).

In totale avremo 1 IO-ops.

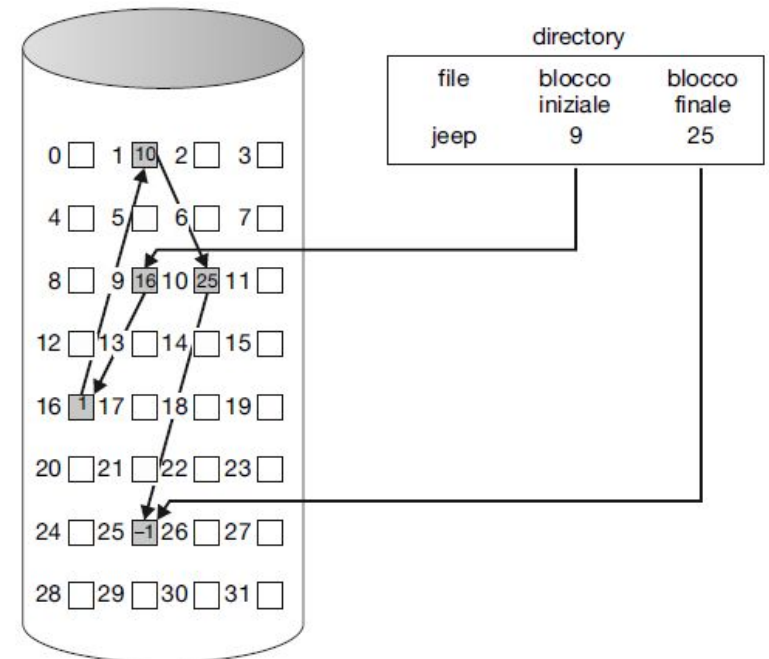
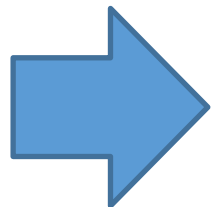


Figura 14.5 Allocazione concatenata dello spazio dei dischi.





# Soluzione Esercizio 1

Nell'implementazione con allocazione contigua, i blocchi sono allocati in maniera sequenziale sul disco, quindi per eliminare un blocco in posizione  $n$  bisognerà copiare tutti i blocchi posteriori a tale blocco, in modo da "compattare" la memoria.

Per spostare un blocco è necessario prima leggerlo (1 IO-ops) e poi scriverlo nella posizione giusta (1 IO-ops).

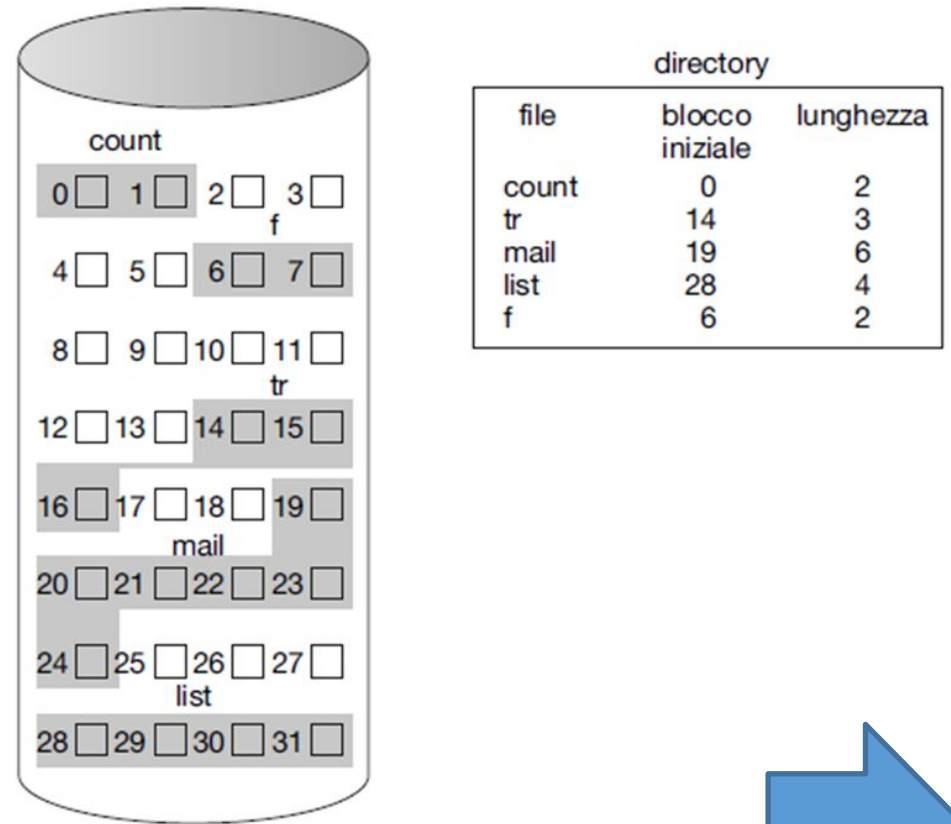


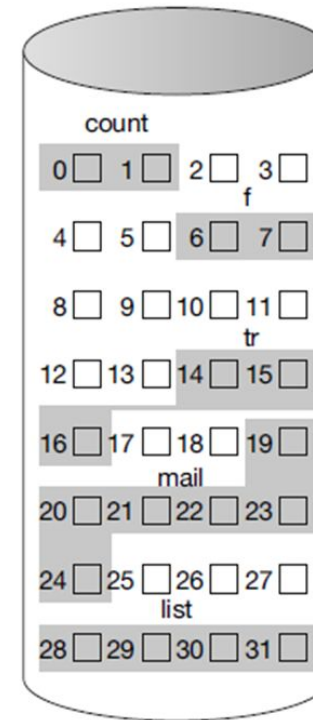
Figura 14.4 Allocazione contigua dello spazio dei dischi.

# Soluzione Esercizio 1

1. Rimozione di un blocco all'inizio del file con **allocazione contigua**

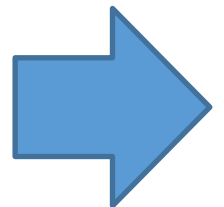
Totale numero di operazioni di I/O su disco:  
198 IO-ops

Con una allocazione contigua andremo a modificare la posizione del primo blocco scritto nel FCB e la lunghezza del file (nessun IO-ops, il FCB è già in memoria), poi sposteremo in sequenza 99 blocchi dalla posizione corrente al blocco precedente (2 IO-ops per ogni blocco) per un totale di  $99 \times 2 = 198$  IO-ops.



directory		
file	blocco iniziale	lunghezza
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Figura 14.4 Allocazione contigua dello spazio dei dischi.



# Soluzione Esercizio 1

2. Rimozione di un blocco a metà del file (51° blocco) con **allocazione concatenata**

Totale numero di operazioni di I/O su disco:  
52 IO-ops

Con una allocazione concatenata andremo ad accedere al blocco che si trova al 51° blocco per recuperare la posizione del 52° blocco (51 IO-ops). Poi scriveremo quest'ultima posizione nel 50° blocco (1 IO-ops).

In totale avremo  $51 + 1 = 52$  IO-ops.

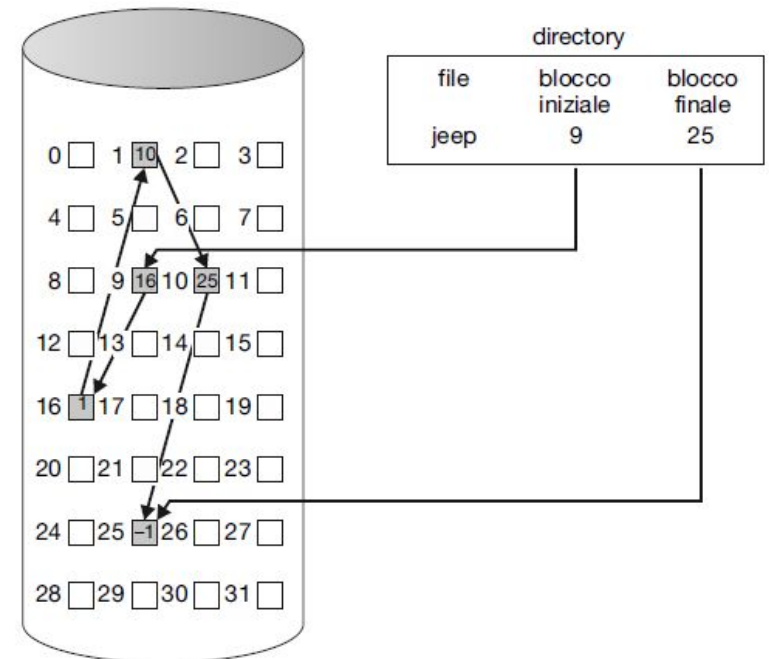
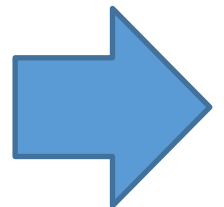


Figura 14.5 Allocazione concatenata dello spazio dei dischi.

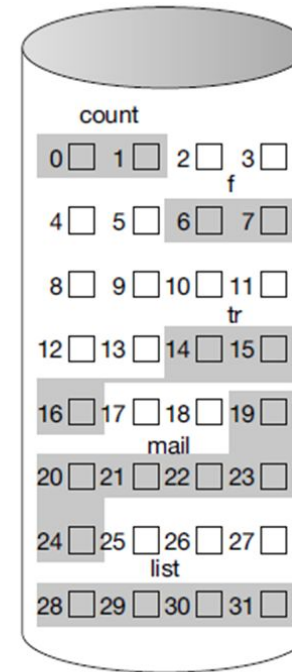


# Soluzione Esercizio 1

2. Rimozione di un blocco a metà del file (51° blocco) con **allocazione contigua**

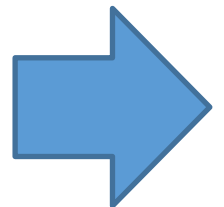
Totale numero di operazioni di I/O su disco:  
98 IO-ops

Con una allocazione contigua andremo a modificare il valore della lunghezza del file (nessun IO-ops, il FCB è già in memoria), sposteremo in sequenza 49 blocchi dalla posizione corrente al blocco precedente (2 IO-ops per ogni blocco) per un totale di  $49 \times 2 = 98$  IO-ops.



directory		
file	blocco iniziale	lunghezza
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Figura 14.4 Allocazione contigua dello spazio dei dischi.



# Soluzione Esercizio 1

3. Rimozione di un blocco alla fine del file con **allocazione concatenata**

Totale numero di operazioni di I/O su disco:  
100 IO-ops

Con una allocazione concatenata andremo ad accedere al blocco che si trova alla fine del file (99 IO-ops) e andremo a scrivere in esso il valore -1 (1 IO-ops). Inoltre, scriveremo la posizione di tale blocco nel FCB (nessun IO-ops, il FCB è già in memoria). In totale avremo  $99 + 1 = 100$  IO-ops.

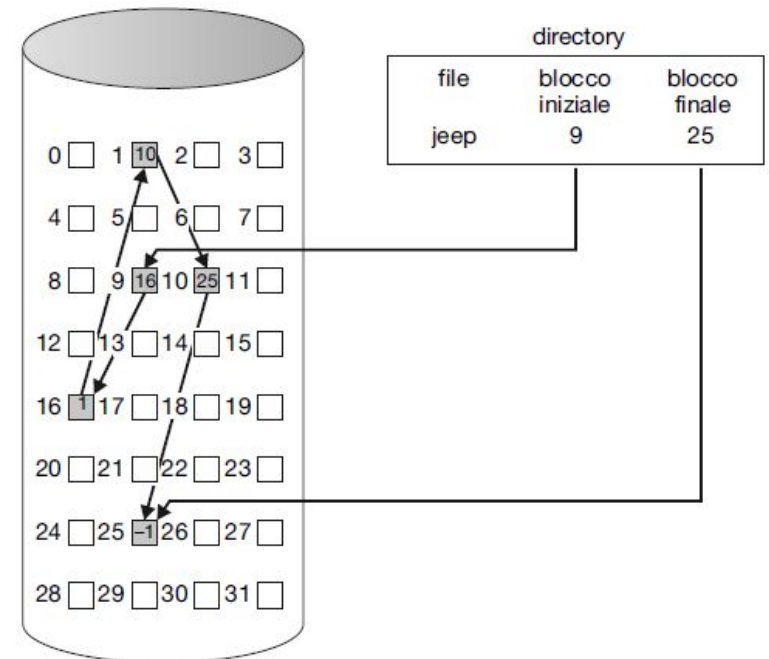
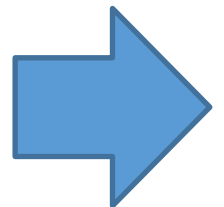


Figura 14.5 Allocazione concatenata dello spazio dei dischi.



# Soluzione Esercizio 1

3. Rimozione di un blocco alla fine del file con **allocazione contigua**

Totale numero di operazioni di I/O su disco:  
0 IO-ops

Con una allocazione contigua andremo a modificare il valore della lunghezza del file (nessun IO-ops, il FCB è già in memoria), per un totale di 0 IO-ops.

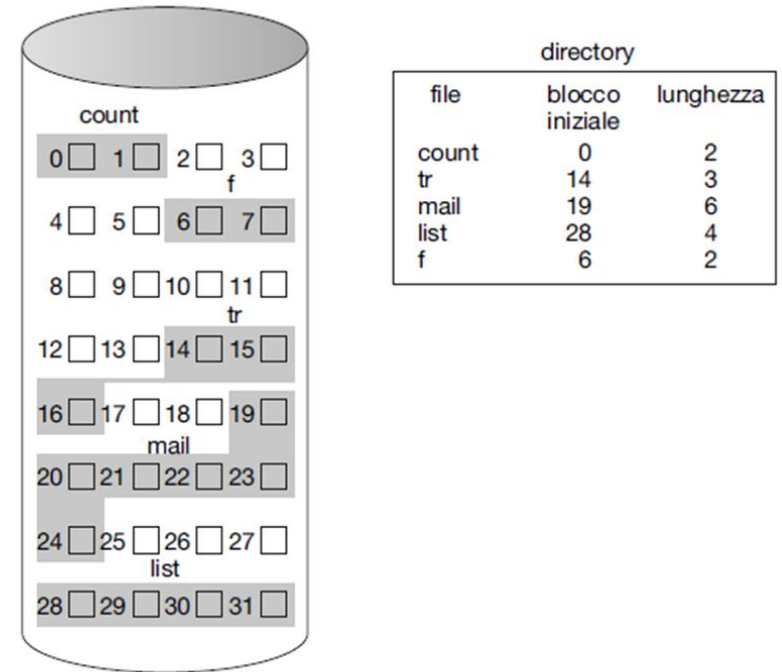


Figura 14.4 Allocazione contigua dello spazio dei dischi.

# Esercizio 2

---

Si consideri un file di grandezza pari a 60 blocchi e che il suo File Control Block (FCB) sia già in memoria.

Siano dati due file-system, gestiti rispettivamente tramite allocazione concatenata e allocazione contigua. Si assuma che nel caso di allocazione concatenata, eventuale spazio per estendere il file sia disponibile solo alla fine dello stesso - non all'inizio.

Si calcoli il numero di operazioni di I/O su disco (IO-ops) necessarie per eseguire le seguenti azioni in entrambi i file-system:

1. Rimozione di un blocco all'inizio del file
2. Rimozione di un blocco a un terzo del file (21° blocco)
3. Rimozione di un blocco alla fine del file

# Soluzione Esercizio 2

---

1. Rimozione di un blocco all'inizio del file
  - allocazione concatenata: 1 IO-ops
  - allocazione contigua: 118 IO-ops
2. Rimozione di un blocco a un terzo del file (21° blocco)
  - allocazione concatenata: 22 IO-ops
  - allocazione contigua: 78 IO-ops
3. Rimozione di un blocco alla fine del file
  - allocazione concatenata: 60 IO-ops
  - allocazione contigua: 0 IO-ops



# Esercizio 3

---

Si consideri l'implementazione di un file system con allocazione concatenata e un file system che invece utilizzi una allocazione indicizzata.

Illustrare brevemente i vantaggi dell'uno e dell'altro nell'eseguire le seguenti operazioni:

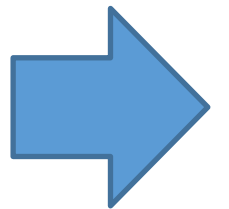
- A. accesso sequenziale
- B. accesso diretto
- C. operazioni su file di testo

# Soluzione Esercizio 3

---

## A. accesso sequenziale

In questo caso, il file system che usa l'allocazione concatenata sarà favorito, garantendo una maggiore velocità dell'operazione. Ciò si verifica poiché non è necessario effettuare alcuna ricerca per trovare il blocco successivo: esso sarà semplicemente il blocco successivo (next) nella lista.

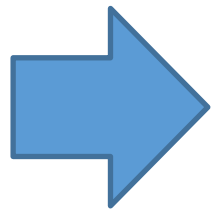


# Soluzione Esercizio 3

---

## B. accesso diretto

Questa operazione - contrariamente all'accesso sequenziale - risulta essere molto onerosa per il file system che usa l'allocazione concatenata. Infatti, per ogni accesso, bisognerà scorrere tutta la lista finché non verrà trovato il blocco desiderato. La ricerca tramite allocazione indicizzata risulterà molto più efficiente.



# Soluzione Esercizio 3

---

## C. accesso su file di testo

Per la natura del tipo di file testo, l'allocazione concatenata risulterà più efficiente rispetto all'allocazione indicizzata.

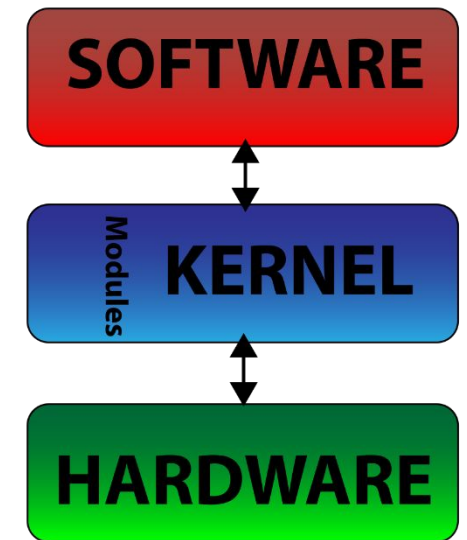
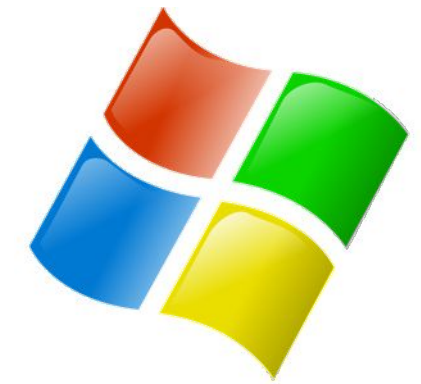
Questo accade poiché i file di testo sono memorizzati in maniera sequenziale sul disco, riportandoci quindi al caso A (accesso sequenziale).



**UNIVERSITÀ DEGLI STUDI  
DELLA BASILICATA**

## *Corso di Sistemi Operativi*

# Esercitazione File System



Docente:  
Domenico Daniele  
Bloisi

