Sistemi operativi

6. I/O

Mauro Fiorentini

6. I/O

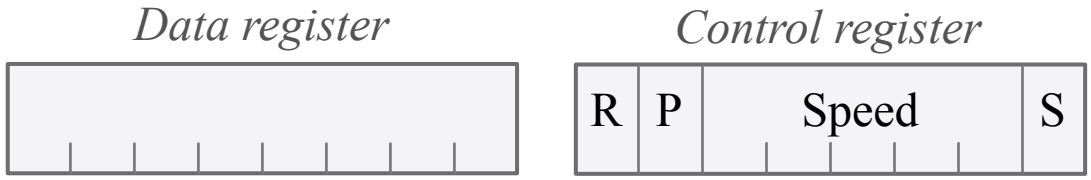
Tipi di dispositivi

- A carattere: le operazioni possono avvenire un carattere per volta.
  - Tastiere, schermi a caratteri, linee seriali;
- A blocchi: le operazioni possono avvenire solo a blocchi di dimensione prefissata
  - Dispositivi magnetici (nastri e dischi), CD, DVD, memoria grafica.

Operazioni di I/O

- Per scrivere su di una periferica il sistema deve:
  - leggere i registri di stato e/o scrivere i registri comando;
  - leggere o scrivere i dati.
- Ogni operazione può iniziare solo dopo il termine della precedente.
  - Lo stato dell'operazione in corso e la sua terminazione sono riportati in alcuni bit del registro di stato.

Esempio di periferica seriale semplificata



R	1 se periferica pronta.
P	1 se controllo parità abilitato.
Speed	codifica la velocità.
S	1 se due bit di stop, 0 se uno solo.

Polling

- Il metodo più semplice per attendere che una periferica sia pronta è eseguire un ciclo, che esamina continuamente i registri di controllo.

Esempio di polling

```
typedef         unsigned char   Byte;

void            WriteByte(Byte Value,
                    volatile Byte *DataPointer,
                    volatile Byte *ControlPointer)
        {
        while ((*ControlPointer & (0x80)) != OK);
        *DataPointer = Value;
        }
```

Svantaggi del polling

- L'attesa utilizza inutilmente la CPU.
  - Per scrivere 1 carattere su di una seriale a 9600 baud serve circa 1 ms; nello stesso tempo un processore può eseguire da migliaia a milioni di istruzioni.
  - In caso di guasto il sistema resta **intrappolato in un ciclo infinito**.
- Accettabile solo in pochi casi:
  - attese brevissime;
  - terminazione dell'attesa garantita, a meno di un guasto catastrofico.

Interrupt

- L'alternativa al polling è l'**interrupt**, un segnale **asincrono**, generato da una periferica, che interrompe il programma in esecuzione.
  - Richiede supporto hardware.
  - Evita perdite di tempo dovute all'I/O.
  - Soluzione di uso universale.
- In generale serve per avvertire il sistema di un evento esterno.

Generazione di un interrupt

- L'interrupt viene generato da una periferica, al verificarsi di condizioni predeterminate.
- Di solito esiste un flag di abilitazione specifico per ogni periferica.
  - Spesso più flag di abilitazione per ogni periferica, uno per ogni evento.
- Il segnale viene inviato alla CPU mediante il control bus.

Ricezione di un interrupt

- La CPU decide se servire o meno gli interrupt in arrivo, alla fine dell'istruzione in esecuzione.
  - La ricezione può essere inibita da un flag nel registro di stato.
  - L'esecuzione di eventuali istruzioni già iniziate viene abortita.

Gestione hardware dell'interrupt (1)

- Quando la CPU accetta l'interrupt:
  - invia un segnale alla periferica;
    - il protocollo di comunicazione con la periferica può variare da una macchina all'altra;
  - salva sullo stack alcuni registri;
    - come minimo, Program Counter e registro di stato;
  - disabilita gli interrupt;
  - passa in modo kernel.

Gestione hardware dell'interrupt (2)

- La CPU carica il Program Counter:
  - con una costante, nelle macchine più semplici;
  - da una locazione fissa;
  - con un valore contenuto in un registro;
  - da un vettore, il cui indirizzo di inizio è fisso o contenuto in un registro e il cui indice è fornito dalla periferica.
- Il controllo passa quindi al sistema.

Gestione software dell'interrupt (1)

- La routine di interrupt del sistema:
  - salva tutti i registri visibili al programmatore, che la routine stessa userà;
  - determina la periferica che ha generato l'interrupt, esaminando i registri di stato delle periferiche;
    - il numero di interrupt può bastare a identificarla;
  - serve l'interrupt.

Gestione software dell'interrupt (2)

- Alla fine la routine di interrupt:
  - ritorna sullo stack utente;
  - ricarica i registri salvati;
  - ricarica Program Counter e registro di stato, con un'istruzione apposita (Return from Interrupt), riabilitando gli interrupt.
- Il processo interrotto riprende la sua esecuzione.

Stack di sistema (1)

- Solo nei sistemi più semplici si utilizza lo stack del processo interrotto per servire l'interrupt:
  - Potrebbe essere troppo piccolo in caso di più interrupt serviti insieme.
  - Il processo potrebbe essere ucciso durante la gestione dell'interrupt stesso, dopo un context switch, deallocando e distruggendo lo stack.

Stack di sistema (2)

- Di solito si utilizza uno stack separato.
  - In molte CPU vi sono due registri Stack Pointer e la CPU usa quello corrispondente al modo di funzionamento corrente, rendendo automatico il cambio di stack.
  - In altri casi il sistema deve provvedere via software.

Gestione degli interrupt da parte del sistema (1)

- Un interrupt può rendere ready un processo e richiedere una schedulazione.
- In questo caso la routine di interrupt cede il controllo ad altre componenti del sistema.
- Prima del context switch il sistema deve completare le operazioni necessarie alla periferica, ma può cambiare il processo running senza completare la routine di gestione dell'interrupt.

Gestione degli interrupt da parte del sistema (2)

- Se il sistema deve eseguire operazioni lunghe, può riabilitare gli interrupt nel frattempo.
- In questo caso il sistema dev'essere scritto con molta attenzione:
  - un nuovo interrupt potrebbe reinnescare lo stesso procedimento, rendendo possibile l'esecuzione parallela di parti diverse del sistema;
  - all'interno del sistema vanno individuate e protette le regioni critiche.

Latenza degli interrupt

- E’ il tempo massimo per il quale un interrupt può essere tenuto in sospeso, considerando:
  - il tempo di esecuzione dell’istruzione più lunga;
  - il tempo di esecuzione della più lunga parte di codice eseguita a interrupt disabilitati.
- Di solito dell’ordine dei microsecondi o nanosecondi nei processori più veloci.

Ridurre la latenza (1)

- E' importante soprattutto nei sistemi real time.
  - Alcuni eventi devono essere gestiti entro tempi molto brevi.
- Se il tempo di latenza è troppo lungo:
  - si possono perdere interrupt consecutivi generati dalla stessa periferica. P. es.:
    - nel caso del clock, l'orologio resta indietro, rendendo inaffidabile la misura del tempo;
    - nel caso di una seriale asincrona, si possono perdere caratteri in arrivo.

Ridurre la latenza (2)

- Renderla molto breve implica:
  - accettare interrupt anche durante l'esecuzione del sistema;
  - frammentare il sistema creando molte piccole regioni critiche, separate da codice che può essere eseguito a interrupt abilitati;
  - in definitiva peggiorare le prestazioni, aumentando la frazione di tempo che la CPU dedica al sistema.

Interrupt con priorità (1)

- In molte macchine è disponibile un piccolo numero di livelli di priorità (da 2 a 8).
  - Il livello di priorità corrente è contenuto nel registro di stato.
- A ogni periferica viene attribuito un livello di priorità
  - Può essere contenuto in un registro, da trasmettere insieme alla richiesta di interrupt, oppure in unit' di controllo interrupt della CPU.
  - I processi girano alla minima priorità.

Interrupt con priorità (2)

- Un interrupt viene accettato solo se di livello superiore al livello corrente.
  - All'accettazione di un interrupt, la CPU alza il livello corrente a quello dell'interrupt.
  - In questo modo si può accettare un interrupt mentre se ne sta gestendo un altro.
  - Il numero di livelli usati limita gli interrupt contemporaneamente in corso di gestione.

Interrupt non mascherabile

- Quasi tutti i processori dispongono di un NMI (Not Maskable Interrupt): un interrupt che non può essere disabilitato in alcun modo.
- Viene utilizzato di solito per segnalare eventi catastrofici, che necessitano di una gestione immediata.

Esempio di NMI

- In caso di mancanza di alimentazione, la CPU può essere avvertita da un apposito circuito, mentre le capacità dell'alimentatore (condensatori) le assicurano ancora centesimi o decimi di secondo di autonomia.
  - La CPU tenta di porre tutta la periferia in uno stato tale da evitare danni all'esterno, salvare alcuni dati in memoria permanente o ad alimentazione indipendente ecc..

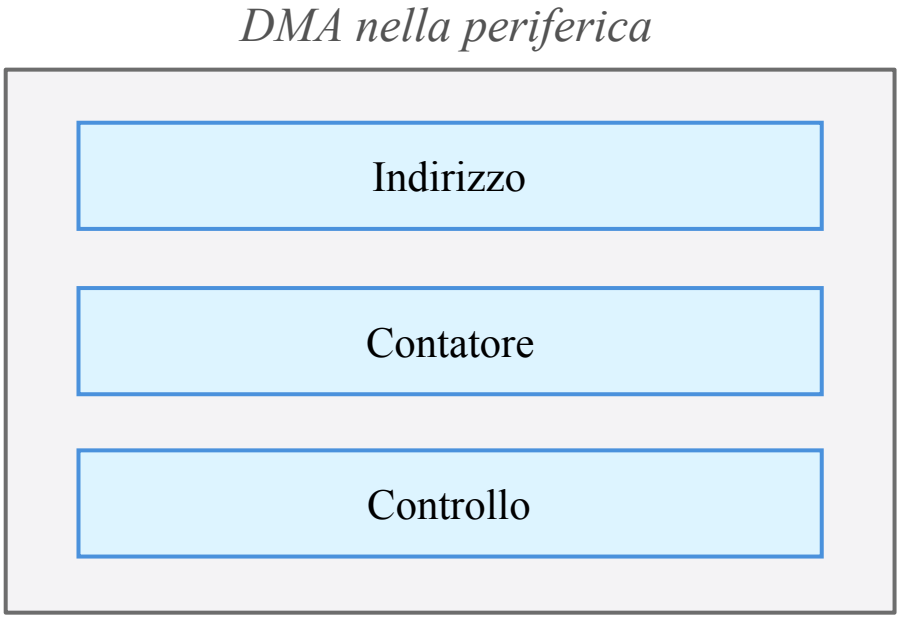
DMA

- Molte periferiche leggono o scrivono dati in blocchi di grandi dimensioni (es.: dischi).
- Per ridurre il tempo speso dalla CPU, conviene evitare di generare un interrupt per ogni byte o parola, utilizzando una tecnica detta DMA (Direct Memory Access).
  - Implementata con un circuito che consente a una periferica di diventare controllore del bus e trasferire autonomamente blocchi di dati.

DMA e bus

- All'inizio il DMA sfruttava i cicli liberi del bus.
- All'aumentare della velocità delle CPU, restano sempre meno cicli disponibili.
- Un circuito di arbitraggio decide chi abbia la priorità negli accessi al bus:
  - priorità ciclica tra DMA e CPU;
  - priorità al DMA;
  - mai priorità alla CPU, che potrebbe bloccare il DMA per un tempo illimitato.

Registri di un'unità DMA semplice



I/O con DMA (1)

- Il sistema carica nei registri:
  - l'indirizzo di un buffer;
  - il numero di byte da leggere o scrivere;
  - i comandi necessari al trasferimento.
- Il sistema innesca quindi l'operazione.

I/O con DMA (2)

- Man mano che i byte vengono letti (scritti), il DMA:
  - li trasferisce dalla periferica alla memoria o viceversa,
  - avanza il puntatore,
  - decrementa il contatore.
- Quando il contatore diventa zero, il DMA genera un interrupt.
- In questo modo si genera **un solo interrupt** ogni qualche migliaio di byte.

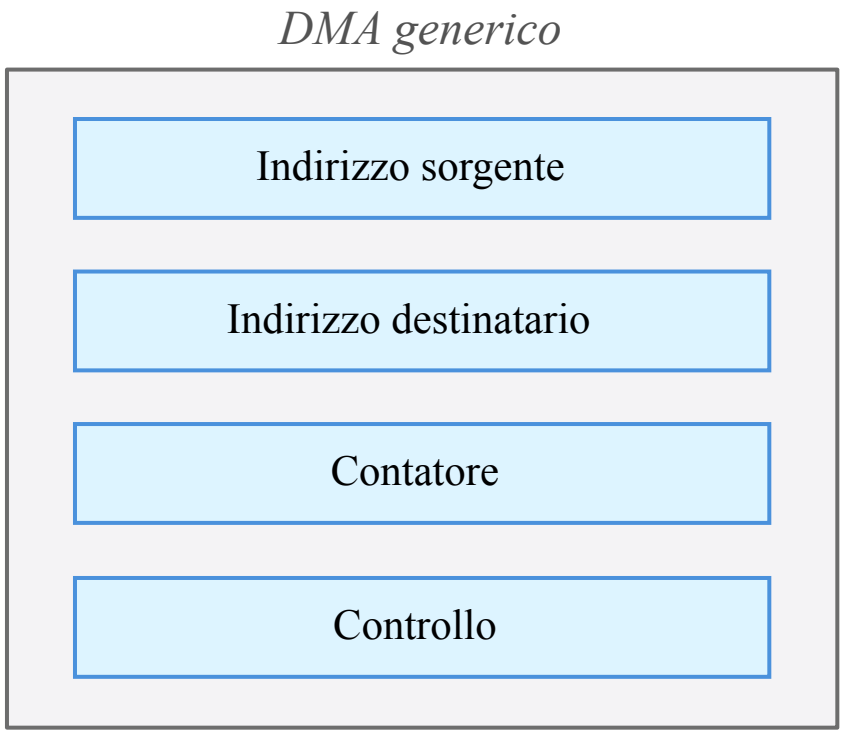
DMA e indirizzi

- Gli indirizzi dei registri del DMA sono reali, non virtuali.
  - Non viene applicata alcuna traduzione.
  - Non viene effettuato nessun controllo.
  - Il sistema controlla e converte in reali gli indirizzi prima di caricarli nei registri del DMA.
  - Le pagine di memoria lette o scritte non devono essere mosse fino alla fine dell'operazione.
  - Il caching di tali pagine deve essere disabilitato.

Il DMA generico

- Circuito simile a un DMA di una periferica, ma con due registri indirizzo.
- Può trasferire dati tra memoria e periferiche memory mapped in qualsiasi combinazione.
  - Gli indirizzi delle periferiche non vengono incrementati.
- In un calcolatore vi possono essere più DMA (tipicamente da 2 a qualche decina).

Registri di un'unità DMA più complessa



Dischi

- Sono dispositivi rotanti, nei quali lettura e scrittura sono effettuate da testine (una per faccia), con metodi ottici o magnetici.
- Le testine sono sorrette da bracci, capaci di muoversi in senso radiale.
- La combinazione di rotazione del disco e movimento dei bracci permette alle testine di raggiungere tutti i punti della superficie.

Dischi magnetici (1)

- La superficie è divisa in blocchi, tutti della stessa dimensione.
  - Tipicamente da 64 byte a 8Kbyte.
  - Costituiscono la minima unità di accesso.
- Più blocchi ad uguale distanza dal centro costituiscono una traccia.
  - Tipicamente da 4 a 64 blocchi per traccia.
- Più tracce su superfici sovrapposte costituiscono un cilindro.
  - Fino a 20 dischi sullo stesso perno, solidali tra loro.

Dischi magnetici (2)

- Spesso le testine sono supportate da un unico braccio, che le sposta tutte insieme.
  - In pratica, ne viene utilizzata una sola per volta.
- Talvolta si usano bracci indipendenti per le varie testine.
  - Unità più costose, ma molto più veloci, perché le testine possono operare indipendentemente in parallelo.

Indirizzo di disco

- Costituito da:
  - t numero di traccia, da 0 a $T - 1$ (T = numero tracce);
  - f numero di faccia, da 0 a $F - 1$ (F = numero facce);
  - s numero di settore, da 0 a $S - 1$ (S = numero settori).
- Generalmente di 4 byte.
- Può essere trasformato in un indirizzo lineare.
  - $I_{\text{lineare}} = s + S * (f + t * F)$
  - Il file system di solito utilizza indirizzi lineari.

Dischi magneto-ottici

- Simili ai dischi magnetici.
- Scrittura magnetica, lettura ottica, con **testine separate**.
  - In lettura, un laser invia un fascio, che viene riflesso in modo diverso, a seconda dello stato di magnetizzazione della superficie.
  - La testina di lettura resta molto più lontana dal disco, senza rischi di un catastrofico contatto.

Dispositivi ottici

- Famiglia di dispositivi che comprende CD e DVD, anche scrivibili.
- Letti mediante, un laser, che viene riflesso dalla superficie intatta, non dalle deformazioni che costituiscono la “scrittura”.
- Le informazioni sono scritte lungo **una spirale**, dalla periferia al centro.

Differenze tra dischi magnetici e ottici

- I dischi magnetici:
  - girano a **velocità costante**;
  - non hanno ridondanza dei dati;
  - richiedono tracce in posizione fissata con estrema precisione.
- I dischi ottici:
  - girano a **velocità variabile**;
  - hanno molta ridondanza dei dati;
  - tollerano una minor precisione della posizione delle tracce.

Velocità di rotazione

- Costante per i dischi magnetici.
  - Inevitabile, a causa delle velocità elevate e delle masse maggiori.
- Variabile per i dischi ottici.
  - Miglior sfruttamento della superficie.
- Il tempo di lettura/scrittura è sempre identico per ogni bit.
  - I bit sono più vicini e **più piccoli** al centro che alla periferia nei dischi magnetici, hanno densità e **dimensioni costanti** in quelli ottici.

Sicurezza e ridondanza

- Nei dischi magnetici un CRC per blocco permette di determinare se lo stesso è stato letto correttamente.
  - La perdita di un blocco è irreparabile.
- Nei dischi ottici un sofisticato sistema di ridondanza permette di ricostruire un intero blocco, con dati contenuti nei due contigui.
  - Sicurezza molto maggiore, minor sfruttamento della superficie.

Posizione delle tracce

- Nei dischi magnetici la distanza delle tracce dall'asse di rotazione è fissata con estrema precisione.
- Nei dischi ottici la precisione è inferiore, perché il supporto deve essere rimovibile, e la testina deve "cercare" la traccia.
  - La testina si mantiene centrata rispetto alla traccia correggendo continuamente la sua posizione radiale.

Caratteristiche dei compact disk

- Scrittura per stampaggio, lettura ottica.
  - Stampati con una matrice, che deforma un sottile foglio metallico.
  - In lettura, un laser invia un fascio, che viene riflesso dalla superficie intatta, non dalle deformazioni incise dalla matrice.
  - La superficie riflettente è protetta da uno strato di plastica trasparente.
  - La ridondanza permette di ignorare piccoli graffi e abrasioni.

Compact disk

- Molto resistenti e affidabili.
  - Insensibili alla luce.
- Bassissimi costi di produzione in volumi medio-grandi.
  - Il costo della matrice li rende convenienti a partire da qualche migliaio di copie.

DVD

- Simili ai CD, ma con capacità maggiore di un ordine di grandezza, a parità di ingombro.
- Maggior densità dei bit.
  - Due strati per superficie (quello superiore è parzialmente trasparente).
    - Una lente doppia focalizza il fascio di lettura su due piani diversi contemporaneamente.

CD/DVD scrivibili (WORM)

- Analoghi ai CD/DVD.
- Una pellicola fotosensibile viene modificata da impulsi molto più intensi di quelli di lettura.
  - La scrittura è molto più lenta della lettura.
  - **Vantaggi**: dispositivi di grandi capacità, a basso costo.
  - **Svantaggi**: più delicati dei CD/DVD e meno affidabili nel lungo periodo (soffrono la luce).
- Possono essere scritti una sola volta.

CD/DVD riscrivibili.

- Analoghi ai CD/DVD scrivibili.
  - La pellicola fotosensibile può essere riportata nello stato iniziale da un fascio laser.
  - Si possono cancellare un numero elevato di volte, ma non all'infinito.
- La scrittura è molto più lenta della lettura
- Vantaggi: dispositivi di grandi capacità, a basso costo.
- Svantaggi: più delicati dei CD/DVD e meno affidabili nel lungo periodo (soffrono la luce).

RAID

- Da Redundant Array of Inexpensive Disks.
- Nella sua forma più comune consiste nello scrivere *n* dati su *n* + 1 dischi.
  - Un disco contiene lo XOR dei dati scritti sugli altri.
  - Le operazioni sono sempre effettuate in parallelo a indirizzi di disco uguali.
  - Il file system vede il RAID come un unico disco di capacità maggiore.
    - Il parallelismo viene gestito localmente, dal controller del disco.

Caratteristiche del RAID

- Aumenta l'affidabilità
  - In caso di completa distruzione di un disco, gli altri consentono di ricostruire i dati.
- Le operazioni di I/O, fatte in parallelo su più dischi, sono più veloci.
  - Solo per quanto riguarda i tempi di lettura/scrittura.
- Lo spreco di spazio disco è contenuto.
  - Generalmente si usano 5 o 9 dischi, con spreco del 20% o dell'11%.

Tempo di accesso in un disco magnetico o ottico

- E' uguale alla somma dei seguenti tempi:
  - **tempo di seek**, per spostare la testina sulla traccia voluta;
    - nel caso del disco ottico, bisogna anche attendere che la velocità di rotazione venga cambiata.
  - **tempo di latenza rotazionale**, per attendere che i dati desiderati passino sotto la testina;
  - **tempo di lettura/scrittura** dei dati;
  - **tempo di trasferimento dei dati** alla memoria centrale.

Tempi di accesso (1)

- Tempo di seek:
  - è dell'ordine dei millisecondi;
  - varia linearmente con la distanza tra la posizione iniziale della testina e quella voluta, più un piccolo intervallo necessario a stabilizzare la testina.
- Tempo di latenza rotazionale:
  - da qualche decimo a pochi millesimi di secondo;
  - dipende dalla velocità di rotazione del disco:
    - dai 300 giri/minuto di un floppy, ai 7200 o 10000 delle più veloci unità a disco.

Tempi di accesso (2)

- Tempo di lettura:
  - è dell'ordine dei millisecondi;
  - varia linearmente con la lunghezza del blocco.
- Tempo di trasferimento:
  - di solito inferiore al millisecondo per blocco;
  - generalmente le unità disco operano con una memoria locale.
    - Possono trasferire dati da e per la memoria centrale mentre eseguono un'altra operazione.

Riduzione dei tempi

- I tempi di seek e di latenza rotazionale possono essere ridotti con accorgimenti software.
- Allocazione dei file.
  - Allocando ogni file su un unico cilindro o su cilindri adiacenti, si riduce il tempo di seek.
  - Allocando i file in ordine lungo le tracce, si riduce la latenza rotazionale per gli accessi sequenziali.
- Seek e latenza possono essere ridotti ordinando le richieste di accesso, prima di passarle alla periferica.

Algoritmi di ordinamento semplici (1)

- FIFO (First In First Out).
  - Semplice.
  - Equo, garantisce che tutte le richieste siano esaudite a breve termine;
  - Non provoca miglioramenti prestazionali:
    - tempo medio di attesa elevato;
    - banda di disco sprecata.

Algoritmi di ordinamento semplici (2)

- SSTF (Shortest Seek Time First): si esegue per prima l'operazione che provoca il minimo movimento della testina:
  - mediamente efficiente;
  - non garantisce l'esaudimento di tutte le richieste in un tempo finito.

Algoritmi di ordinamento a scansione

- Si passano i cilindri in ordine, eseguendo su ognuno tutte le operazioni in coda.
  - Algoritmi efficienti, che garantiscono che tutte le richieste siano esaudite a breve termine.
- Varianti:
  - SCAN (algoritmo dell'ascensore): arrivati all'ultimo cilindro, si effettua una scansione in senso opposto.
  - C-SCAN: come SCAN, ma ripartendo sempre dal primo cilindro.

Tempi di I/O

- Le operazioni di I/O sono molto più lente degli accessi in memoria.
- Per migliorare le prestazioni sono possibili varie soluzioni.
  - Non far attendere il processo (I/O asincrono).
  - Limitare il numero di accessi fisici (buffer management).
  - Evitare operazioni ridondanti (caching).
- Si possono sfruttare i tempi di attesa di un processo per far lavorare gli altri.

I/O asincrono

- Quando il processo richiede un'operazione di I/O, questa viene lanciata, mentre il processo resta libero di proseguire il suo lavoro.
  - Relativamente facile da implementare per le scritture, estremamente complesso per le letture.
- Deve poter essere disabilitato, per permettere operazioni interattive con l'utente.

Problemi dell'I/O asincrono

- Non riduce il carico complessivo del sistema.
  - Lo aumenta leggermente, perché rende il sistema più complesso.
- Resta il problema di notificare al processo eventuali errori.
  - Talvolta non si riesce a sapere con certezza l'esito dell'operazione.
- Può rendere più complesso il codice dei processi.

Buffer management

- Le operazioni sui dispositivi a blocchi vengono eseguite solo a blocchi relativamente grandi (parecchi Kbyte).
  - Quando un processo legge, viene letto un blocco, indipendentemente dalla quantità di byte richiesti; il resto del blocco viene conservato in un buffer dal sistema, in modo che le successive letture sequenziali non abbiano bisogno di accessi fisici.
  - Quando un processo scrive, i dati sono scritti in un buffer, che viene scaricato solo quando pieno o quando viene fatto un accesso non sequenziale.

Caratteristiche del buffer management

- I blocchi sono condivisi tra tutti i processi.
- Implica I/O asincrono, con i problemi relativi.
  - Generalmente si può disabilitare in modo selettivo.
- Il sistema può effettuare pre-fetching, se scopre accessi sequenziali.
  - Aumenta il carico del sistema, ma possono migliorare le prestazioni dei processi.

Caching

- Gli ultimi blocchi di dati letti o scritti dai processi vengono conservati in memoria per un certo tempo, supponendo che verranno riutilizzati in tempi brevi.

Svantaggi di buffer management e caching (1)

- Riduzione dell'affidabilità.
  - Un'eventuale caduta del sistema provoca la perdita dei dati non ancora scritti sul dispositivo esterno.
  - I dati non vengono effettivamente scritti nell'ordine richiesto: in caso di caduta del sistema possono verificarsi incoerenze tra i dati.
  - Se i problemi riguardano i dati di sistema (directory, struttura del file system), possono verificarsi danni per tutti gli utenti.

Svantaggi di buffer management e caching (2)

- Possibile riduzione delle prestazioni.
  - Il sistema diventa più complesso.
  - Serve un trasferimento di dati tra memoria del processo e buffer di sistema.