# CAPITOLO 13: SISTEMI DI I/O

I due compiti principali di un calcolatore sono l’I/O e l’elaborazione. Spesso il compito principale è costituito dall’I/O mentre l’elaborazione è semplicemente accessoria.

Il ruolo di un sistema operativo nell’I/O di un calcolatore è quello di gestire e controllare le operazioni e i dispositivi di I/ O.

## INTRODUZIONE

Il controllo dei dispositivi connessi a un calcolatore è una delle questioni più importanti che riguardano i progettisti di sistemi operativi. Poiché i dispositivi di I/O sono così largamente diversi per funzioni e velocità ( es un mouse, un disco e un jukebox di CD-ROM) altrettanto diversi devono essere i metodi di controllo.

Tali metodi costruiscono il *sottosistema di I/O* del kernel.

D’altra parte, però, si assiste a una crescente varietà di dispositivi di I/O; alcuni di loro sono tanto diversi dai dispositivi precedenti dal rendere molto difficile il compito di integrarli nei calcolatori e nei sistemi operativi esistenti. Questo problema si affronta strutturando gli elementi di base dell’architettura di I/O in modo da potervi connettere un ampia varietà di dispositivi di I/O e strutturando il kernel del sistema operativo in moduli di driver di dispositivi allo scopo di incapsulare i dettagli e le particolarità dei diversi dispositivi.

I driver dei dispositivi offrono al sottosistema di I/O un’interfaccia uniforme per l’accesso ai dispositivi,cosi come le chiamate di sistema forniscono un’interfaccia uniforme tra le applicazioni e il SO.

## ARCHITETTURE E DISPOSITIVI DI I/O

I calcolatori fanno funzionare un gran numero di tipi di dispositivi: la maggior parte rientra nella categoria dei dispositivi di memorizzazione secondaria e terziaria , dispositivi di trasmissione. Un dispositivo comunica con un sistema di calcolo inviando segnali attraverso un cavo o attraverso l’etere e comunica con il calcolatore tramite un punto di connessione (porta), ad esempio una porta seriale. Se uno o più dispositivi usano in comune un insieme di fili. La connessione e detta bus. Un bus è un insieme di fili e un protocollo rigorosamente definito che specifica l’insieme dei messaggi che si possono inviare attraverso i fili. In termini elettronici, i messaggi si inviano tramite configurazioni di livelli di tensione elettrica applicate ai fili con una definita scansione temporale.

I bus sono ampiamente usati nell’architettura dei calcolatori e differiscono tra loro per metodo di segnale, velocità, throughput e metodi di connessione.

Un calcolatore è un insieme di componenti elettronici che può far funzionare un porta, un bus o un dispositivo. Un controllore di porta seriale è un semplice controllore di dispositivo; di tratta di un singolo circuito integrato nel calcolatore che controlla i segnali presenti nei fili della porta seriale. Alcuni dispositivi sono dotati di propri controllori incorporati.

L’unità d’elaborazione dà comandi e fornisce dati al controllore per portare a termine trasferimenti di I/O tramite uno o più registri per dati e segnali di controllo. La comunicazione con il controllore avviene attraverso la lettura e la scrittura, da parte dell’unità d’elaborazione, di configurazioni di bit in questi registri. Un modo in cui questa comunicazione può avvenire è tramite l’uso di speciali istruzioni di I/O che specificano il trasferimento di un byte o una parola a un indirizzo di porta I/O.

l’istruzione di I/O attiva le linee di bus per selezionare il giusto dispositivo e trasferire bit dentro o fuori dal registro di dispositivo.

In alternativa, il controllore di dispositivo può disporre dell’I/O associato alla memoria o mappato in memoria. In questo caso i registri di controllo del dispositivo si fanno corrispondere ad un sottoinsieme dello spazio di indirizzi della CPU, che esegue le richieste di I/O usando le ordinate istruzioni di trasferimento dati per leggere e scrivere i registri di controllo del dispositivo.

Certi sistemi usano le tecniche. I PC ad esempio usano istruzioni di I/O per controllare alcuni dispositivi e l’I/O mappato in memoria per controllare altri. il controllore della grafica ha alcune porte di I/O per le operazioni di controllo di base, ma dispone di un’ampia regione mappata in memoria, detta memoria grafica, che serve a mantenere i contenuti dello schermo. Il processo scrive sullo schermo inserendo i dati nella regione mappata in memoria; il controllore genera l’immagine dello schermo sulla base del contenuto di questa regione di memoria.

Questa tecnica è semplice da usare; inoltre la scrittura di milioni di byte nella memoria grafica è molto più veloce dell’invio di milioni di istruzioni di I/O.

La facilità di scrittura in memoria è controbilanciata da uno svantaggio: un comune errore di programmazione è la scrittura in una regione di memoria sbagliata causata da un errato puntatore.

Una porta di I/O consiste in genere in quattro registri: status, control, data-in e data-out.

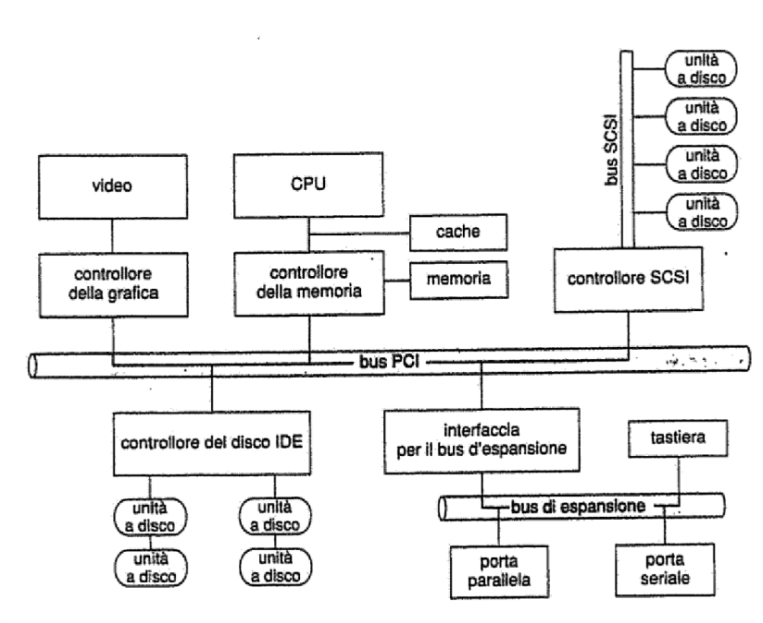
1)La CPU legge il registro data-in per ricevere dati.

2)La CPU scrive nel registro data-out per emettere dati.

3)Il registro status contiene alcuni bit che possono essere letti e indicano lo stato della porta; ad esempio indicano se è stata portata a termine l’esecuzione del comando corrente

4)Il registro control può essere scritto per attivare un comando o per cambiare il modo di funzionamento del dispositivo. Ad esempio, un certo bit nel registro control della porta seriale determina il tipo di comunicazione tra *half-duplex e full-duplex*

La tipica dimensione dei registri di dati varia tra 1 e 4 byte. Certi controllori hanno circuiti integrati FIFO che possono contenere parecchi byte per l’immissione e l’emissione dei dati, in modo da espandere la capacità del controllore oltre la dimensione del registro di dati.



## Interrogazione ciclica

Il protocollo completo per l’interazione fra la CPU e un controllore può essere intricato, ma la fondamentale nozione di negoziazione *(handshaking)* è semplice, ed è illustra con un esempio.

Il controllore specifica il suo stato per mezzo del bit busy del registro status; pone a 1 il bit busy quando è impegnato in un’operazione, e lo pone a 0 quando è pronto a eseguire il comando successo. La CPU comunica le sue richieste tramite il bit comand-ready nel registro command; pone questo bit a 1 quando il controllore deve eseguire un comando.

1. La CPU legge ripetutamente il bit busy finché non valga 0.
2. La CPU pone a 1 il bit write del registro dei comandi e scrive un byte nel registro data-out
3. La CPU pone a 1 il bit command-ready .
4. Quando il controllore si accorge che il bit command-ready è posto a 1, pone a 1 il bit busy
5. Il controllore legge il registro dei comandi e trova il comando write; legge il registro data-out per ottenere il byte da scrivere, e compie l’operazione di scrittura nel dispositivo
6. Il controllore pone a 0 il bit command-ready, pone a 0 il bit error nel registro status per indicare che l’operazione di I/O ha avuto esito positivo, e pone a 0 il bit busy per indicare che l’operazione è terminata.

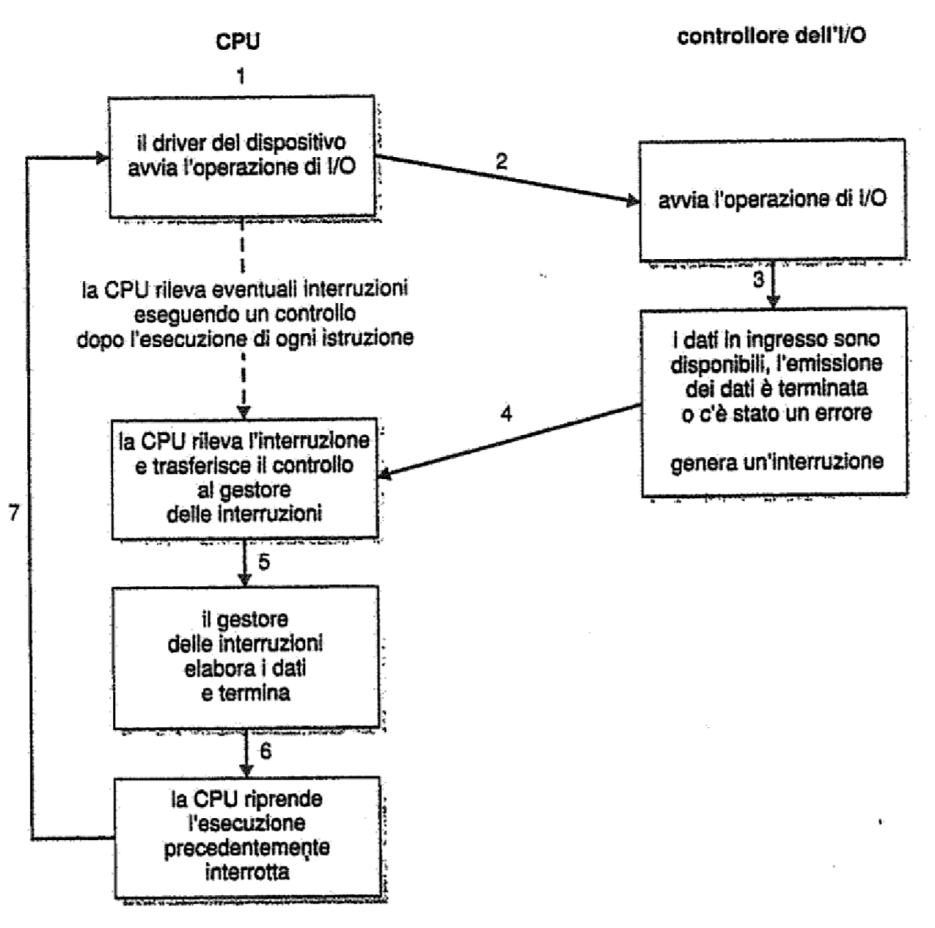
(La sequenza appena descritta si ripete per ogni byte. )

Durante l’esecuzione del passo 1, la CPU è in attesa attiva (*busy-waiting)* o in interrogazione ciclica (polling): itera la lettura del registro status finché il bit busy assume il valore 0. Se il controllore e il dispositivo sono veloci, questo metodo è ragionevole, ma se l’attesa rischia di prolungarsi, sarebbe probabilmente meglio se la CPU si dedicasse a un’altra operazione.

Quando, ad esempio, i dati affluiscono in una porta seriale o della tastiera. Il piccolo buffer del controllore diverrà presto pieno, e se la CPU attende troppo a lungo prima di riprendere la lettura dei byte, si prenderanno informazioni.

In molte architetture di calcolatori sono sufficienti tre cicli di istruzioni di CPU per in-terrogare ciclicamente un dispositivo: read, lettura di un registro del dispositivo; logical-and, configurazione logica usata per estrarre il valore di un bit di stato, e branch, salto a un altro punto del codice se l’argomento è diverso da zero.

Anziché richiede alla CPU di eseguire un’interrogazione ciclica, può essere più efficiente far si che il controllore comunichi alla CPU che il dispositivo è pronto. Il meccanismo dall’architettura che permette tale comunicazione si chiama interruzione della CPU o, più brevemente, interruzione *(interrupt).*



## Interruzioni

La CPU ha un contatto, detto linea di richiesta dell’interruzione, del quale la CPU controlla lo stato dopo l’esecuzione di ogni istruzione. Quando rileva il segnale di un controllore nella linea di richiesta dell’interruzione, la CPU salva lo stato corrente e salta alla routine di gestione dell’ interruzione *(interrupt-handler routine),* che si trova e un indirizzo prefissato di memoria. Questa procedura determina le cause dell’interruzione, porta a termine l’elaborazione necessaria ad esegue un’istruzione return from interrupt per far sì che la CPU ritorni nello stato in cui si trovava prima della sua interruzione. Il controllore del dispositivo *genera* un segnale d’interruzione della CPU lungo la linea di richiesta delle interruzioni, che la CPU *rileva* e *recapita* al gestore delle interruzioni, che a sua *evade* il compito corrispondente servendo il dispositivo. Il meccanismo di base delle interruzioni permette alla CPU di rispondere ad un evento asincrono

Nei sistemi operativi moderni necessarie capacità di gestione delle interruzioni più raffinate.

1. Si deve poter posporre la gestione dell’interruzione durante le fasi critiche dell’elaborazione.
2. Si deve disporre di un meccanismo efficiente per passare il controllo all’appropriato gestore delle interruzioni, senza dover esaminare ciclicamente tutti i dispositivi *(polling)* per determinare quale abbia generato l’interruzione.
3. Si deve disporre di più livelli d’interruzione, di modo che il sistema possa distinguere le interruzioni ad altra priorità da quelle a priorità inferiore, servendo le richieste con la celerità appropriata del caso.

In un calcolatore moderno queste tre caratteristiche sono fornite dalla CPU e dal controllore delle interruzioni .

La maggior parte delle CPU ha due linee di richiesta delle interruzioni. Una è quella delle interruzioni non mascherabili, riservata a eventi quali gli errori di memoria irrecuperabili. La secondo linea è quella delle interruzioni mascherabili: può essere disattivata dalla CPU prima dell’esecuzione di una sequenza critica di istruzioni che non deve essere interrotta. Il meccanismo delle interruzioni accetta un indirizzo, nella maggior parte delle architetture questo indirizzo è uno scostamento relativo a una tabella detta vettore delle interruzioni, contenente gli indirizzi di memoria gli specifici gestori delle interruzioni. In pratica, tuttavia, i calcolatori hanno più dispositivi (e quindi, più gestori delle interruzioni) che elementi nel servirsi di una tecnica detta concatenamento delle interruzioni (*interrupt chaining)*, in cui ogni elemento del vettore delle interruzioni punta alla testa di una lista di gestori nella lista delle interruzioni. Quando si verifica un’interruzione, si chiamano uno alla volta i gestori nella lista corrispondente finché non se ne trova uno che può soddisfare la richiesta .

E descritto il vettore delle interruzioni della CPU Intel Pentium.

Il meccanismo delle interruzioni realizza anche un sistema di livelli di proprietà delle interruzioni. Esso permette alla CPU di differire la gestione delle interruzioni di bassa priorità senza mascherare tutte le interruzioni, e permette a un’interruzione di priorità alta di sospendere l’esecuzione della procedura di servizio di un ‘interruzione di priorità bassa. Un sistema operativo moderno interagisce con il meccanismo delle interruzioni in vari modi. All’accensione della machina esamina i bus per determinare quali dispositivi siano presenti, e installa gli indirizzi dei corrispondenti gestori delle interruzioni nel vettore delle interruzioni. Durante l’I/O, i vari controllori di dispositivi generano i segnali d’interruzione della CPU quando sono pronti per un servizio.

Il meccanismo delle interruzioni si usa anche per gestire un’ampia gamma di eccezioni, come la divisione per zero, l’accesso a indirizzi di memoria protetti o inesistenti o il tentativo di eseguire un’istruzione privilegiata in modalità utente.

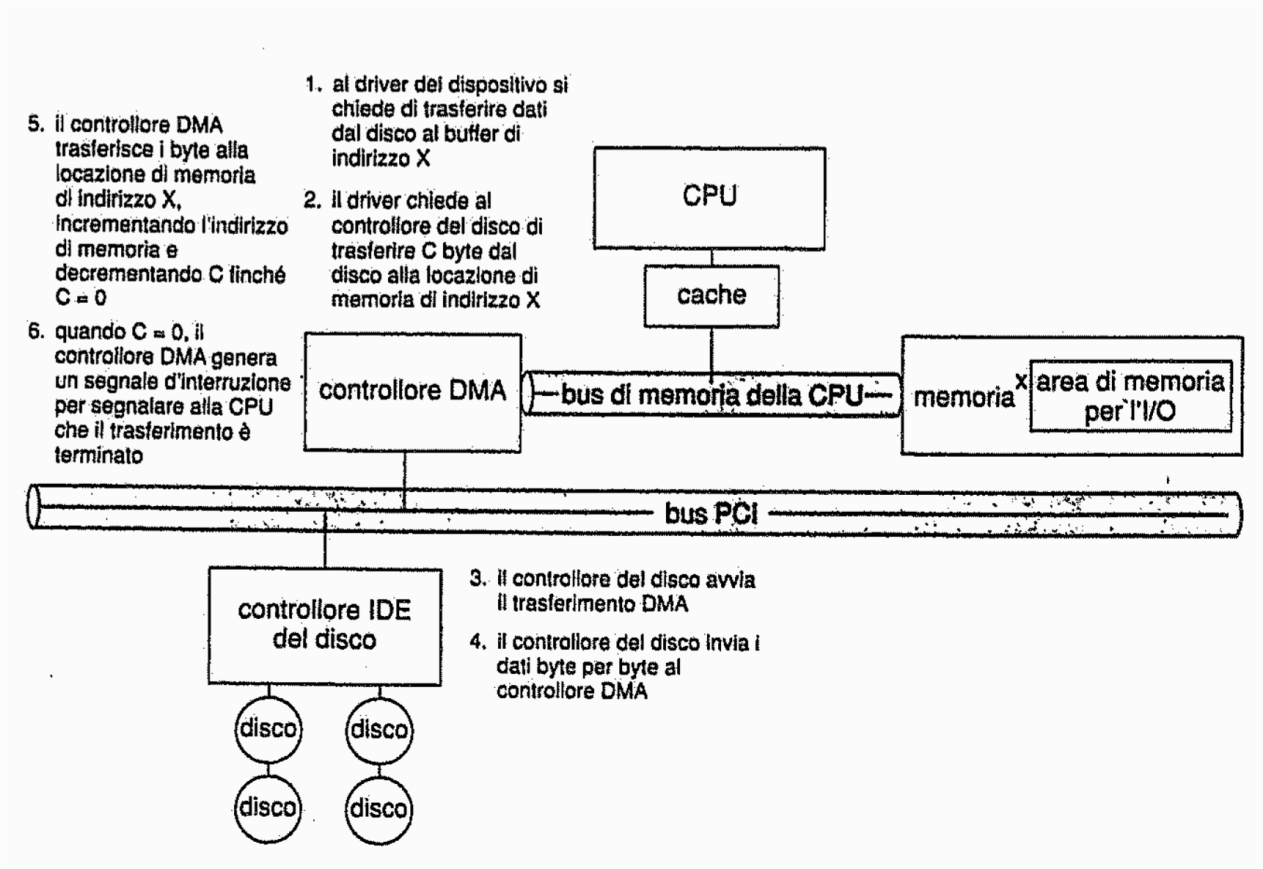
Un altro esempio è dato dall’esecuzione delle chiamate di sistema. Solitamente i programmi sfruttano routine di libreria per eseguire chiamate di sistema. La routine controlla i parametri passati dall’applicazione, li assembla in una struttura dati appropriata da passare al kernel, e infine esegue una particolare istruzione detta interruzione software o trap. Quando la chiamata di sistema esegue l’istruzione di eccezione, l’architettura delle interruzioni memorizza le informazioni riguardanti lo stato cui era giunta l‘esecuzione del codice utente, passa al modo supervisore e recapita l’interruzione alla procedura del kernel che realizza il servizio richiesto. Le interruzioni si possono inoltre usare per gestire il controllo del flusso all’interno del kernel. Si consideri ad esempio l’elaborazione richiesta per completare una lettura da un disco. Un passo necessario è quello i copiare dati dalla regione di memoria usata dal kernel al buffer dell’utente. Questa azione richiede tempo, ma non è urgente e non dovrebbe bloccare la gestione delle interruzioni con priorità più alta. Un altro passo è quello di avviare l’evasione desse successive richieste di IO relative a quell’unita a disco. Questo passo ha priorità più alta: se le unta a disco si devono usare in modo efficiente, è necessario avviare l’evasione della successiva richiesta di IO non appena la precedente sia stata soddisfatta.

Un’architettura del kernel basata su thread è adatta alla realizzazione di più livelli di priorità delle interruzioni e a dare la precedenza alla gestione delle interruzioni rispetta alle elaborazioni in sottofondo delle procedure del kernel e delle applicazioni. Riassumendo i segnali d’interruzione sono usati diffusamente dai SO moderni per gestire eventi asincroni e per eseguire in modalità supervisore le procedure del kernel. Per far si che i compiti più urgenti siano portati a termine per primi, i calcolatori moderni usano un sistema di priorità delle interruzioni. I controllori dei dispositivi, gli errore i e le chiamate di sistema generano segnali d’interruzione al fine di innescare l’esecuzione di procedure del kernel.

## Accesso diretto alla memoria DMA

Quando un dispositivo compie trasferimenti di grandi quantità di dati, come nel caso di un’unita a disco, l’uso di una costosa CPU per il controllo dei bit di stato e per la scrittura di dati nel registro del controllore byte alla volta, detto IO programmato, sembra essere uno spreco. In molti calcolatori si evita di sovraccaricare la CPU assegnando una parte di questi compiti a un’unita di elaborazione specializzata, della controllore dell’accesso diretto alla memoria DMA. Per dar avvio a un trasferimento DMA la CPU scrive in memoria un comando strutturato per il DMA, esso contiene un puntatore alla locazione dei dati da trasferire, un altro puntatore alla destinazione dei dati e il numero di byte da trasferire. La CPU scrive l’indirizzo di questo comando nel controllore del DMA, e prosegue con la sua esecuzione. Il controllore DMA agisce quindi direttamente sul bus della memoria, presentano al bus gli indirizzi di memoria necessarie per eseguire il trasferimenti senza l’aiuto della CPU. Un semplice controllore DMA è un componente o ordinario dei PC e le schede di IO dette bus mastermind di un PC includono di solito componenti DMA ad alta velocità.

La procedura di negoziazione tra il controllore del DMA e il controllore del dispositivo si svolge grazie a una coppia di fili detti DMA request e DMA acknowledge. Il controllore del dispositivo manda un segnale sulla linea DMA request quando una parola di dati è disponibile per il trasferimento. Questo segnale fa si che il controllore DMA prenda possesso del bus di memoria, presenti l'indirizzo desiderato ai fili d’intirizzimento della memoria e mandi un segnale lungo la linea DMA acknowledge. Quando il controllore del dispositivo riceve questo segnale, trasferisce in memoria la parola di dati e rimuove il segnale dalla linea DMA request. Quando l’intero trasferimento termina, il controllore del DMA interrompe la CPU. Quando il controllore del DMA prende possesso del bus di memoria, la CPU è temporaneamente impossibilitata ad accedere alla memoria centrale, sebbene abbia accesso ai dati contenuti nella sua cache primaria e secondaria. Questo fenomeno noto come sottrazione di cicli, può rallentare la computazione della CPU; ciononostante l’assegnamento del lavoro di trasferimento di dati a un controllore DMA migliora e in generale le prestazioni complessive del sistema. In alcune architetture per realizzare la tecnica DMA si usano gli indirizzi della memoria fisica, mentre in altre s’impiega l’accesso diretto alla memoria virtuale, in questo caso si usano indirizzi virtuali che poi si traducono in indirizzi fisici.



## INTERFACCIA DI I/O PER LE APPLICAZIONI

Si spiega come un’applicazione possa aprire n file residente in un disco senza sapere di che tipo di disco si tratti e come si possano aggiungere al calcolatore nuove unita a disco e latri dispositivi senza che si debba modificare il sistema operativo.

I metodi qui esposti coinvolgono l’astrazione, l’incapsulamento e la stratificazione dei programmi. In particolare si puo compiere un procedimento di astrazione rispetto ai dettagli delle differenze tra i dispositivi per l’IO identificandone alcuni tipi generali. A ognuno di questi tipi si accede per mezzo di un unico insieme di funzioni- un’interfaccia. Le differenze sono incapsulate in moduli del kernel detti driver dei dispositivi.

Lo scopo dello strato dei driver dei dispositivi è di nascondere al sottosistema di IO del kernel le differenze tra i controllori dei dispositivi in mood simile a quello con cui le chiamate di sistema di IO incapsulando il comportamento dei dispostivi in alcun classi generiche che nascondo le differenze alle applicazioni.

Sfortunatamente per i produttori di dispositivi, ogni tipo di sistema operativo ha le sue convenienze riguardanti l’interfaccia dei driver dei dispositivi. I dispositivi possono differire in molti aspetti:

* **Trasferimento a flusso di caratteri o a blocchi.** Un dispositivo del primo tipo trasferisce dati un byte alla volta mentre uno del secondo tipo ne trasferisce un blocco alla volta.
* **Accesso sequenziale o diretto.** Un dispositivo del primo tipo trasferisce dati sentendo ordine prestabilito, mentre l’utente di un dispositivo ad accesso diretto può richiedere l’accesso a una qualunque delle possibili locazioni di memorizzazione
* **Dispositivi sincroni o asincroni.** Un dispositivo sincrono trasferisce dati con un tempo di risposta prevedibile, mentre un dispositivo asincrono ha tempi di risposta irregolari
* **Condivisibili o riservati.** Un dispositivo condivisibile può essere usato in modo concorrente da diversi processi mentre ciò è impossibile se un dispositivo è riservato.
*  **Velocità di trasferimento.** Può variare da alcuni byte a alcuni gb
*  **Lettura e scritture, solo lettura o solo scrittura.**

Per ciò che riguarda l’accesso delle applicazioni ai dispositivi, molte di queste differenze sono nascoste dal sistema operativo e i dispostivi sono raggruppati in poche classi convenzionali.

## 

## Dispositivi con trasferimento a blocchi o a caratteri

L’interfaccia per i dispostivi a blocchi sintetizza tutti gli aspetti necessari per accedere alle unita a disco e ad altri dispositivi basati sul trasferimento di blocchi di dati. Il SO e certe applicazioni particolari come quelle per la gestione della basi di dati possono trovare più convenirne trattare questi dispositivi come una semplice sequenza lineare di blocchi. In questo caso si parla di IO a basso livello. La tastiera è un esempio di dispositivo al quale si accede tramite un’interfaccia a flusso di caratteri.

## 

## Dispositivi di rete

Poiché i modi di intirizzimento e le prestazioni tipiche dell’IO di rete sono notevolmente differenti da quelli dell’IO dell’unita a disco, la maggior parte dei SO fornisce un’interfaccia per l’IO di rete diversa da quelle delle normali operazioni read(),write() e seek().Un’interfaccia disponibile in molti SO è l’interfaccia di rete socie. Una volta creata una socket si possono usare le normali operazioni di IO.

## Orologi e timer

La maggior parte dei calcolatori ha timer e orologi che forniscono 3 funzioni essenziali:

* Segnare l’ora corrente;
* Segnalare il tempo trascorso;
* Regolare un timer in modo da avviare l’operazione “x” al tempo “t”;

Il dispositivo che misura la durata di un lasso di tempo e che può avviare un’operazione si chiama timer programmabile. Lo scheduler usa questo meccanismo per generare un segnale d’interruzione che sospende un processo quando il suo quanto di tempo è scaduto. Il SO può inoltre fornire un’interfaccia per permettere ai processi utenti di usare il timer.

In certi casi , simulando orologi virtuali, il SO può anche gestire un numero di richieste d’uso dei timer maggiore del numero dei timer fisici. Per far ciò il driver del timer (kernel) mantiene una lista ordinata cronologicamente delle interruzioni richieste dagli utenti e dalle proprie procedure, e imposta il timer per la prima scadenza.

Nella maggio parte parte dei calcolatori, l’orologio è costruito sulla base di un contatore ad alta frequenza.

## 

## IO bloccante e non bloccante

Un altro aspetto delle chiamate di sistema è la scelta fra IO bloccante e non bloccante. Quando un’applicazione impiega una chiamata di sistema bloccante si sospende l’esecuzione delle applicazioni, che passa dalla coda dei processi pronti per l’esecuzione alla coda d’attesa. Quando la chiamata di sistema termina l’applicazione è posta nuovamente nella coda dei processi pronti in modo che possa riprendere l’esecuzione solo allora essa ricevere i valori riportati dalla chiamata di sistema.

Alcuni processi a livello utente necessitano di una forma di IO non bloccante. Un esempio è quello di un’interfaccia utente con cui s’interagisce col mouse e la tastiera mentre elabora dati e li mostra sullo schermo.

La differenza tra chiamata non bloccante e asincrona è che una read non bloccante restituisce immediatamente il controllo fornendo i dati che è stato possibile leggere. Una chiamata read asincrona richiede un trasferimento di cui il sistema garantisce il completamento ma solo in un momento successivo e non prevedibile.