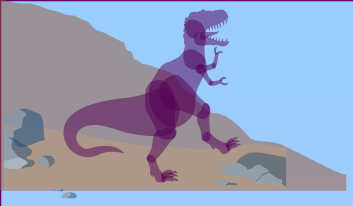
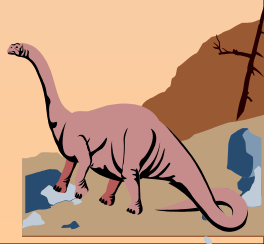


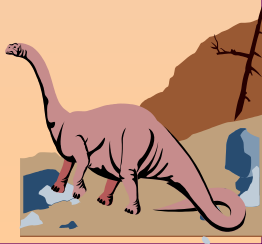
Capitolo 13: Sistemi di I/O

- Introduzione
- Architetture e dispositivi di I/O
- Interfaccia di I/O per le applicazioni
- Sottosistema per l'I/O del kernel
- Trasformazione delle richieste di I/O in operazioni dei dispositivi
- STREAMS
- Prestazioni



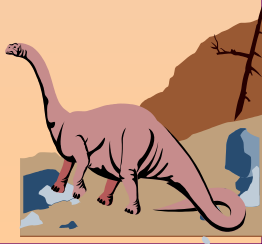
Introduzione

- I due compiti di un computer sono l'I/O e l'elaborazione.
- Il ruolo di un sistema operativo nell'I/O di un computer è quello di gestire e controllare le operazioni e I dispositivi di I/O.
- I dispositivi di I/O sono diversi per funzioni e velocità, quindi abbiamo diversi metodi di controllo.
- Questi metodi rappresentano il **sottosistema di I/O** del kernel.
- Questo sottosistema separa il resto del nucleo dalla complessità di gestione dei dispositivi di I/O.
- Esiste una grande varietà di dispositivi di I/O.
- I **driver dei dispositivi** offrono al sottosistema di I/O un'interfaccia uniforme per l'accesso ai dispositivi,
  - così come le system call forniscono un'interfaccia uniforme tra le applicazioni ed il sistema operativo.



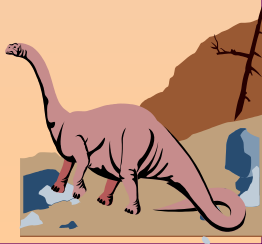
Architetture e dispositivi di I/O

- Esistono tre categorie generali di dispositivi di I/O:
  - memoria secondaria e memoria terziaria:
    - dischi, nastri, etc.,
  - dispositivi di trasmissione:
    - modem, schede di rete),
  - interfaccia uomo macchina :
    - tastiera, schermo,videocamere digitali, etc.
- Ogni dispositivo comunica con il sistema di calcolo inviando segnali attraverso un cavo o attraverso l'etere.
  - Comunica con il calcolatore tramite un punto di connessione (porta).
- I dispositivi di I/O comunicano tra loro e con il nucleo del sistema mediante un bus:
  - un insieme di fili ed un protocollo ben definito che specifica l'insieme dei messaggi che si possono inviare attraverso i fili.

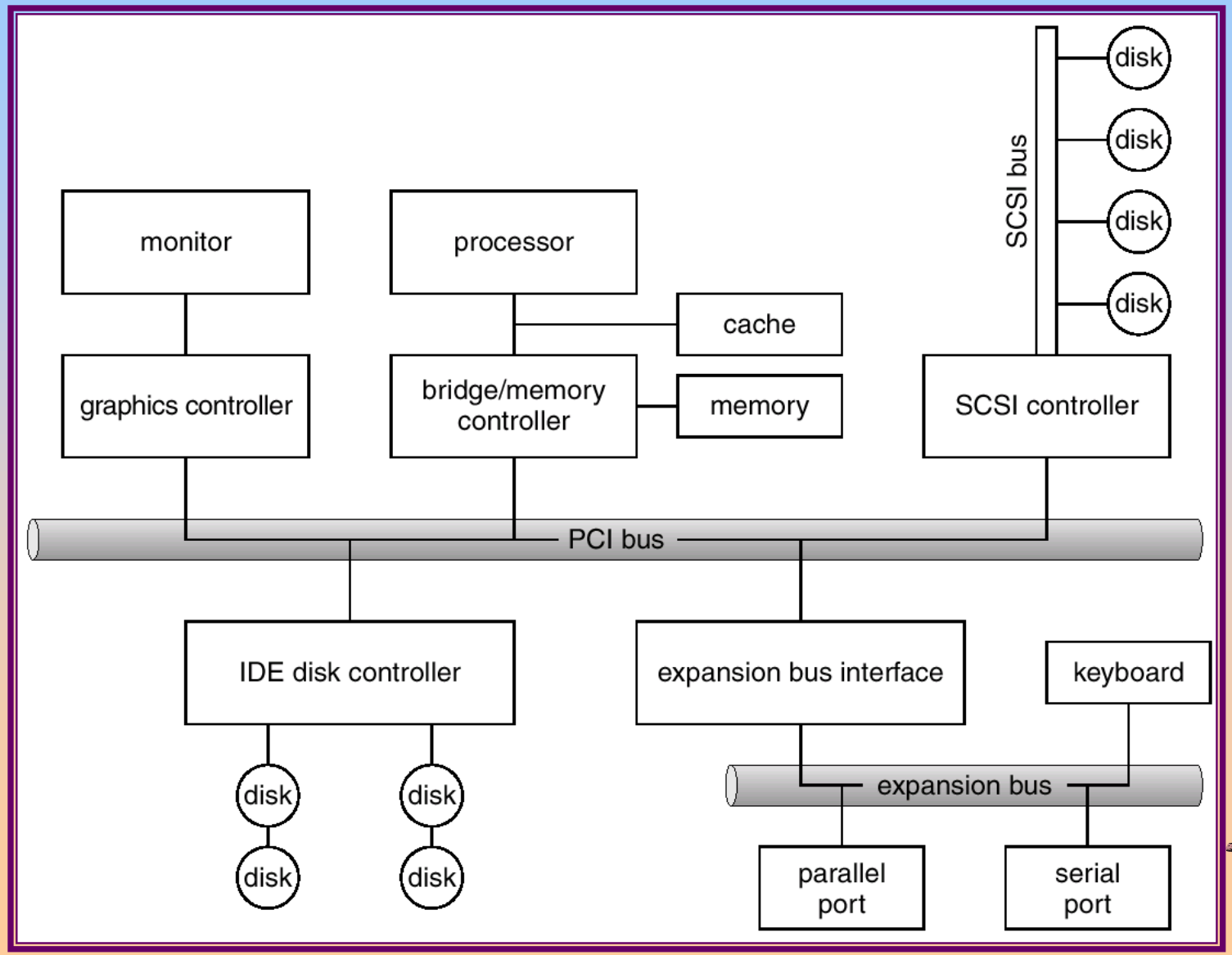


Architetture e dispositivi di I/O (II)

- I messaggi sono inviati tramite configurazioni di livelli di tensione elettrica applicati ai fili con una scansione temporale ben definita.
  - Quando un dispositivo A ha un cavo che si connette ad un dispositivo B ed il dispositivo B ha un cavo che si connette ad un dispositivo C che a sua volta è collegato ad una porta di un calcolatore si ottiene il cosiddetto collegamento a margherita (daisy chain) che di solito funziona come un bus.
- Un classico esempio di bus, che è divenuto uno standard come bus di sistema nei PC è il bus PCI:
  - Il comune bus di sistema dei PC che connette il sottosistema processore-memoria ai dispositivi veloci.
- Un bus di espansione viene utilizzato per connettere al sistema i dispositivi relativamente lenti
  - tastiera, porte seriali o parallele, etc..



Tipica struttura del bus di un PC

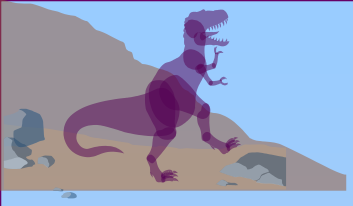


Hardware di I/O: controllori

- La parte hardware che svolge il ruolo di interfaccia tra il bus e i componenti di I/O è chiamata **controllore**.
- Un controllore è un insieme di componenti elettronici che può far funzionare una porta, un bus o un dispositivo..
- La complessità hardware di un controllore dipende dall'attività che esso deve svolgere sul dispositivo.
- Esistono controllori semplici, come ad es. il controllore delle porte seriali:
  - un singolo circuito integrato che controlla i segnali presenti nei fili della porta seriale.
- Altri, invece, possono disporre di un proprio microprocessore, di una propria memoria, dei propri registri e addirittura di piccoli sistemi operativi microcodificati per svolgere le operazioni dedicate esclusivamente al controllore.
  - Ad es. il controllore del bus SCSI è spesso realizzato con una scheda circuitale separata **(adattatore**) che contiene un'unità di elaborazione, microcodice e memoria locale che gli consentono di elaborare i messaggi del protocollo SCSI.

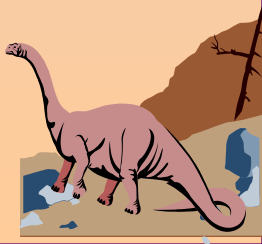
Hardware di I/O: scambio di dati

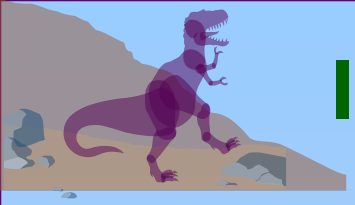
- Il processore dà comandi e fornisce dati al controller per portare al termine trasferimenti di I/O tramite i suoi registri di dati e di controllo.
- Quindi la comunicazione tra processore e controller avviene mediante la scrittura e la lettura di configurazioni di bit in tali registri da parte del processore.
- La comunicazione può avvenire in due modi:
  - **Utilizzo di speciali istruzioni di I/O**: L'istruzione di I/O attiva le linee di bus che selezionano il giusto dispositivo e trasferiscono bit nei (o dai) registri del dispositivo stesso.
  - **I/O associato alla memoria**: Ai registri del controller viene assegnato una sequenza di indirizzi in memoria centrale, in modo che ogni operazione di lettura o scrittura a questi indirizzi comporta un trasferimento diretto di dati con i registri del dispositivo.
    - **vantaggi**: aumento dell'efficienza del sistema; il S.O. non deve preoccuparsi di cercare spazio disponibile in memoria dove inserire le informazioni.
    - **svantaggi**: rischio di avere errori software: necessita' di tecniche di protezione della memoria.



Hardware di I/O: porte di I/O

- Una **porta di I/O** permette la trasmissione di informazioni tra il sistema centrale e un dispositivo.
- Tipicamente consiste di 4 registri:
  - **status**: contiene dei bit che possono essere letti dall'host e indicano lo stato della porta,
    - Ad es. indicano se è stata portata a termine l'esecuzione del comando corrente, se un byte è disponibile per essere letto, se si è verificato un errore nel dispositivo.
  - **control**: viene scritto dall'host per attivare un comando o per cambiare il modo operativo del dispositivo.
  - **data-in**: mantiene i dati che vengono letti dall' host.
  - **data-out**: è il registro in cui l'host scrive i dati in output.
- La tipica dimensione dei registri varia tra 1 e 4 byte.



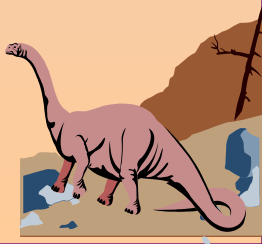


Indirizzi delle porte dei dispositivi di I/O nei PC (elenco parziale)

I/O address range (hexadecimal)	device
000-00F	DMA controller
020-021	interrupt controller
040-043	timer
200-20F	game controller
2F8-2FF	serial port (secondary)
320-32F	hard-disk controller
378-37F	parallel port
3D0-3DF	graphics controller
3F0-3F7	diskette-drive controller
3F8-3FF	serial port (primary)

Interrogazione ciclica

- Con il crescere del numero dei dispositivi cresce l'esigenza di avere un controllo continuo e asincrono sull'I/O.
- Il modo in questo avviene è attraverso la negoziazione (handshaking).
- Ad esempio, si assuma l'uso di due bit per controllare la relazione di tipo produttore/consumatore fra il controllore e la CPU.
- Il controllore specifica il suo stato per mezzo del bit *busy* del registro *status*.
- Pone a 1 il bit busy quando è impegnato in una operazione, e lo pone a 0 quando è pronto per il comando successivo.
- La CPU comunica le sue richieste tramite il bit *command-ready* del registro *command*.
- Pone questo bit a 1 quando il controllore deve eseguire un comando.

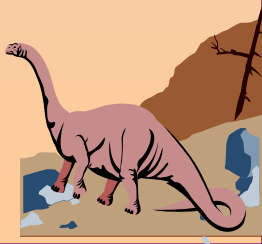


Interrogazione ciclica (II)

- La CPU scrive in una porta coordinandosi con un controllore tramite negoziazione in questo modo:
  1. La CPU legge ripetutamente il bit *busy* fino a che esso non vale 0.
  2. La CPU pone a 1 il bit *write* nel registro *command* e scrive un byte nel registro *data-out*.
  3. La CPU pone a 1 il bit *command-ready*.
  4. Quando il controller si accorge che il bit *command-ready* è posto a 1, pone a 1 il bit *busy*.
  5. Il controller legge il registro *command* e riceve il comando *write*, legge il registro *data-out* per ottenere il byte da scrivere, ed effettua l'operazione di scrittura nel dispositivo.
  6. Il controller pone a 0 il bit *command-ready*. Se l'operazione di I/O è stata eseguita correttamente pone a 0 il bit *error* nel registro status, e pone a 0 il bit *busy* per indicare che l'operazione è terminata.
- Questa sequenza viene ripetuta per ogni byte.
- Durante l'esecuzione del passo 1 la CPU è in **attesa attiva** o in **interrogazione ciclica**:
  - ☞ itera la lettura del registro *status* fino a che il bit *busy* assume il valore 0.

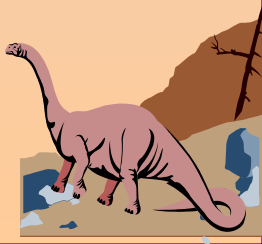
Interruzioni

- La CPU ha un contatto detto **linea di richiesta dell'interruzione** del quale controlla lo stato dopo l'esecuzione di ogni istruzione.
- Quando rileva il segnale di un controllore, la CPU memorizza lo stato del processo corrente e salta alla **procedura di gestione delle interruzioni** che si trova ad un indirizzo prefissato,
  - ☞ tramite questa procedura gestirà l'interruzione.
- Quando l'I/O è terminato, un interrupt segnala che i dati sono pronti e il processo può essere ripreso.
- Nel frattempo, la CPU può mandare in esecuzione altri processi o altri thread dello stesso processo.

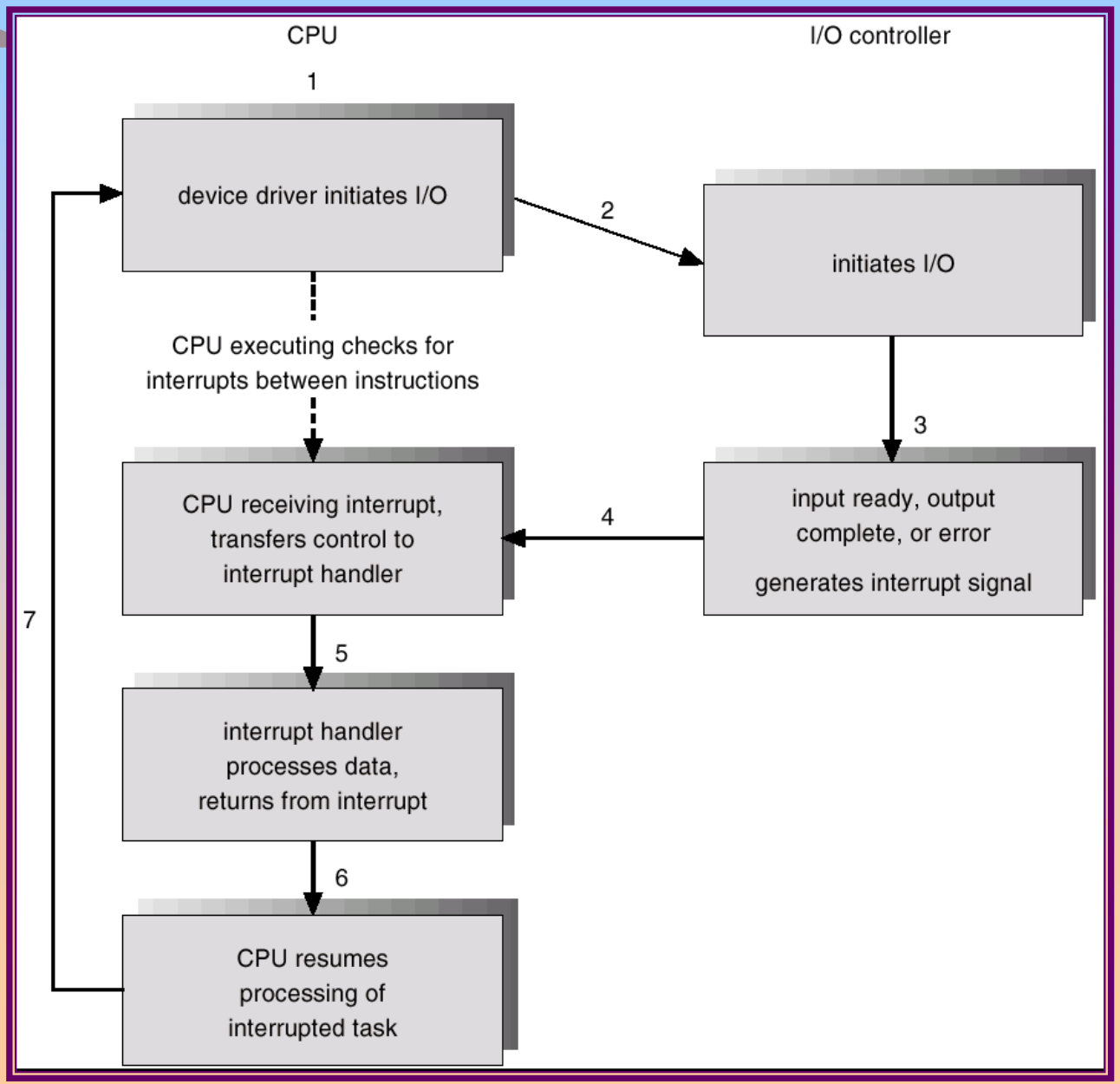


Interruzioni (II)

- **Vettore delle interruzioni**:
  - tabella che associa ad ogni interrupt l'indirizzo di una corrispondente routine di gestione.
- In genere esistono due linee di richiesta delle interruzioni:
  - **Interruzioni non mascherabili**
    - ad es. errori di memoria irrecuperabili.
  - **Interruzioni mascherabili**
    - Interruzioni che possono essere disattivate dalla CPU prima di una sequenza critica di istruzioni che non deve essere interrotta.
- Un sistema di **livelli di priorità delle interruzioni** permette alla CPU di differire la gestione delle interruzioni di bassa priorità e permette ad una interruzione di priorità alta di sospendere l'esecuzione di quella di una bassa.
- Gli interrupt vengono usati anche per indicare eccezioni



Ciclo di I/O basato sulle interruzioni



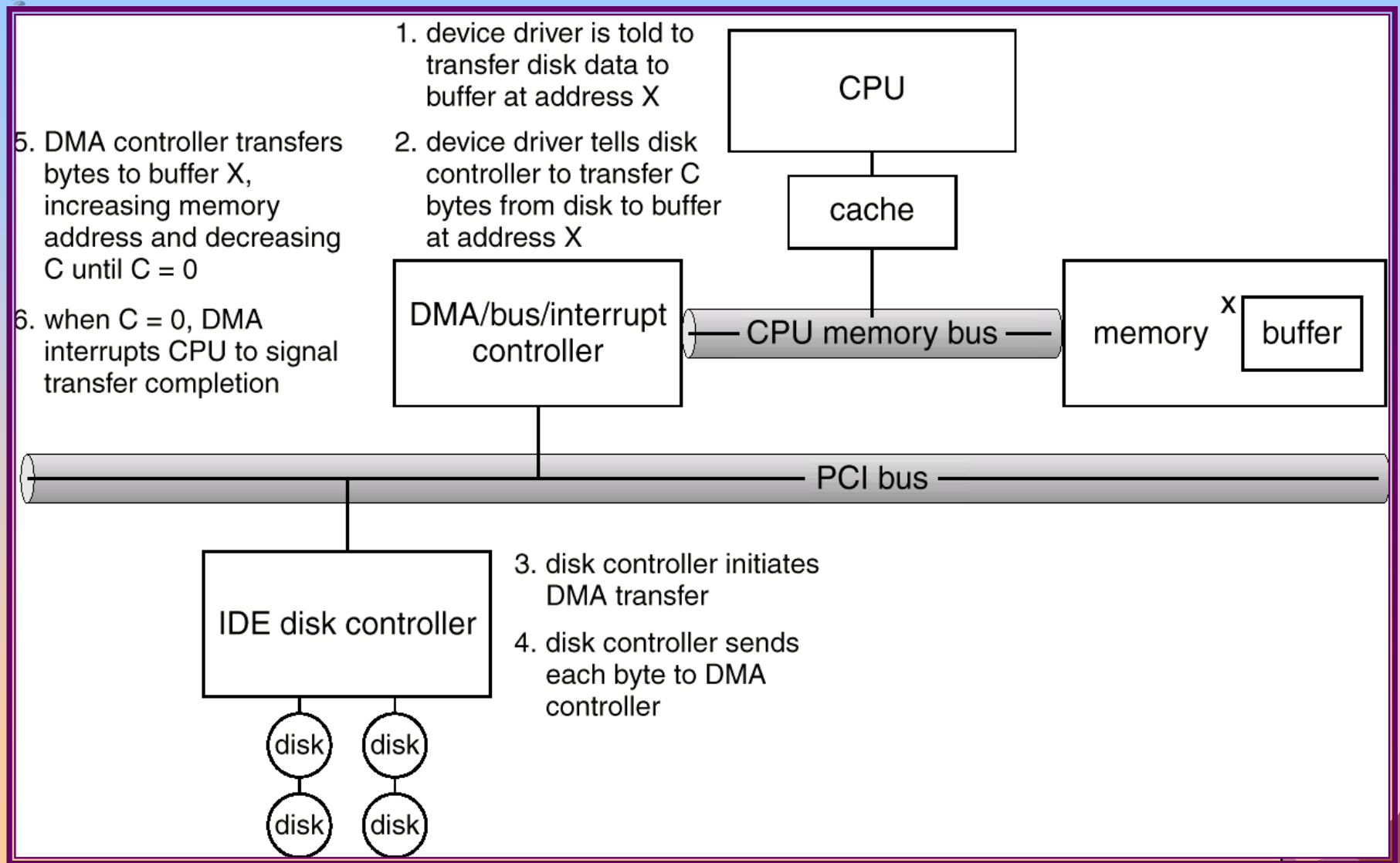
Vettore delle interruzioni della CPU Intel Pentium

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19Ð31	(Intel reserved, do not use)
32Ð255	maskable interrupts

Accesso diretto alla memoria (DMA)

- In molti sistemi di elaborazione si utilizza una unità di elaborazione specializzata, detta controllore dell'accesso diretto alla memoria (DMA).
- Il controllore di DMA è dedicato al trasferimento diretto delle informazioni dalle periferiche alla memoria centrale.
- Il DMA viene utilizzato nelle operazioni di trasferimento di grosse quantità di dati.
  - Per dare avvio ad un trasferimento DMA, la CPU scrive in memoria un comando che contiene un puntatore alla locazione dei dati ed un altro alla destinazione, ed il numero di byte da trasferire.
  - Poi passa l'indirizzo del comando al controllore del DMA e prosegue con l'esecuzione di altro codice.
- Il controllore DMA agisce direttamente sul bus della memoria, presentando al bus gli indirizzi di memoria necessari al trasferimento.
- Con l'accesso diretto alla memoria virtuale (DVMA) l'accesso diretto avviene allo spazio indirizzi virtuale del processo, e non a quello fisico (per l'I/O associato alla memoria).
  - In questo caso si usano indirizzi virtuali che poi si traducono in indirizzi fisici.

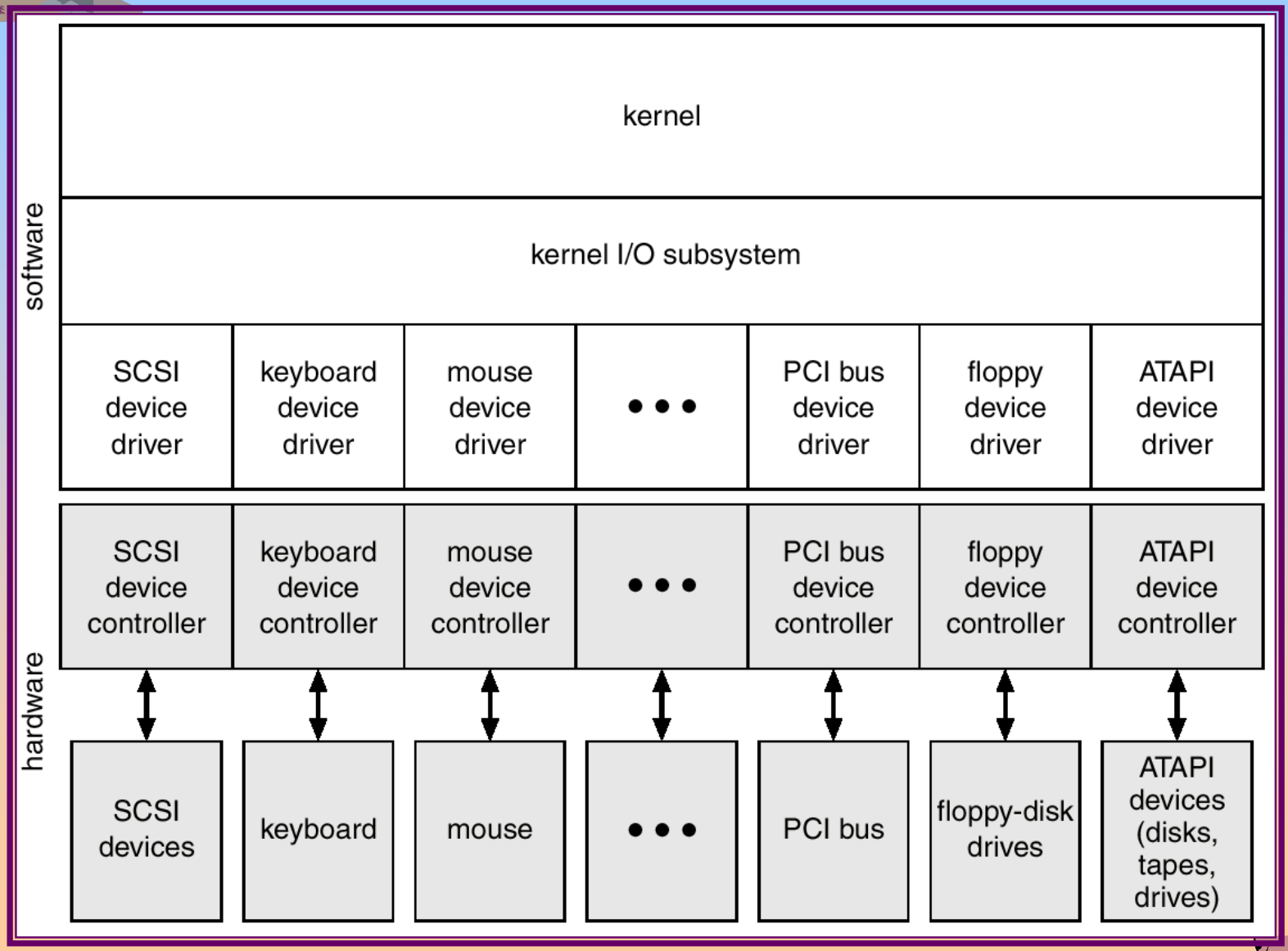
Passi di un trasferimento DMA



Interfaccia di I/O per le applicazioni

- E' necessario avere un trattamento uniforme dei dispositivi di I/O,
  - quindi le chiamate di sistema di I/O incapsulano il comportamento dei dispositivi in alcuni tipi generali.
- Le effettive differenze tra I dispositivi sono contenute nei driver,
  - moduli del kernel dedicati a controllare ogni diverso dispositivo.
  - Sfortunatamente ogni tipo di S.O. ha le sue convenzioni riguardanti l'interfaccia dei driver dei dispositivi.
  - Così un dato dispositivo sarà venduto con driver diversi, ad es. per MS-DOS, Windows, Unix, etc..
- Per l'accesso delle applicazioni ai dispositivi, le chiamate di sistema raggruppano tutti i dispositivi in poche classi generali, uniformando I modi di accesso.
  - Solitamente queste classi sono:
    - I/O a blocchi o a flusso di caratteri,
    - accesso mappato in memoria,
    - socket di rete.
- Spesso è disponibile una system call scappatoia (*back-door*) dove si fa rientrare ciò che non rientra nei casi precedenti (ad es. *ioctl* in Unix).

Struttura relativa all'I/O di un nucleo

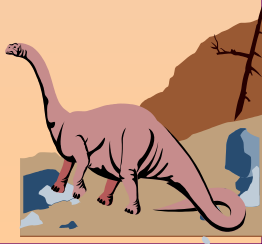


Caratteristiche dei dispositivi di I/O

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only readÐwrite	CD-ROM graphics controller disk

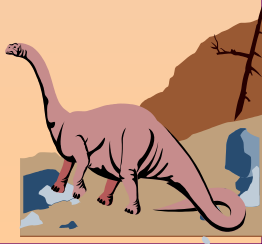
Dispositivi a blocchi e a caratteri

- L'interfaccia per i **dispositivi a blocchi** sintetizza gli aspetti necessari per accedere alle unità a disco e ad altri dispositivi basati sul trasferimento di blocchi di dati.
  - Comandi tipici: *read, write, seek***.**
- Di solito le applicazioni comunicano con i dispositivi tramite un file system che funge da interfaccia.
- Altre volte i dispositivi sono trattati come una semplice sequenza lineare di blocchi (*I/O a basso livello*).
- I file possono essere *associati alla memoria*:
  - si fa coincidere una parte dello spazio indirizzi virtuale di un processo con il contenuto di un file.
- I **dispositivi a carattere** sono dispositivi che generano o accettano uno stream di dati,
  - ad es.: tastiera, mouse, porte seriali.
  - Comandi tipici *get*, *put* di singoli caratteri o parole.
- Non è possibile la seek.



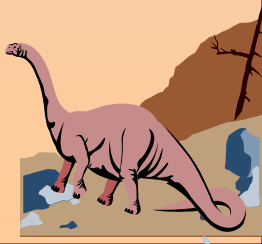
Dispositivi di rete

- I modi di indirizzamento e le prestazioni tipiche dell'I/O di rete sono differenti da quelli dei dispositivi a blocchi o caratteri.
- La maggior parte dei S.O. fornisce un'interfaccia per l'I/O di rete diversa da quella per i dischi.
- L'interfaccia per l'I/O di rete fornita su molti S.O. è l'interfaccia di rete *socket*.
- Le system call fornite da questa interfaccia permettono di
  - creare un socket,
  - collegare un socket locale all'indirizzo di un altro punto della rete,
  - controllare se un'applicazione si sia inserita nella socket locale, e
  - inviare o ricevere pacchetti di dati lungo la connessione.
- Una funzione *select* gestisce un insieme di socket, restituendo informazioni
  - sui socket per i quali sono presenti pacchetti che attendono di essere ricevuti e
  - su quelli che hanno spazio per accettare un pacchetto da inviare.
- *select* elimina la necessità di interrogazione ciclica.



Orologi e temporizzatori

- La maggior parte dei computer ha temporizzatori e orologi hardware che offrono tre funzioni essenziali:
  - segnare l'ora corrente.
  - segnalare il lasso di tempo trascorso.
  - regolare un temporizzatore 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 è detto *temporizzatore programmabile.*
- Si può regolare in modo da attendere un certo tempo e poi generare un segnale di interruzione.
  - Viene utilizzato anche per segnalare lo scadere della *slice* di tempo di un processo, per riversare periodicamente nei dischi il contenuto della *buffer cache*, etc..
- Per l'ora corrente vi è un contatore di interrupt ad alta frequenza chiamato *orologio hardware.*

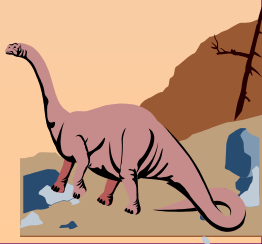


I/O bloccante e non bloccante

- Un altro aspetto delle chiamate del sistema è la scelta tra I/O bloccante e non bloccante.
- Una system call **bloccante** sospende l'esecuzione dell'applicazione che l'ha invocata.
  - Questa passa dalla coda dei processi pronti alla coda d'attesa del sistema.
  - Quando la chiamata a sistema termina l'applicazione torna nella coda dei processi pronti.
- Alcuni processi necessitano di una forma **non bloccante** di I/O.
- Una system call di questo tipo restituisce rapidamente il controllo all' applicazione che la ha invocata, fornendo un parametro che indica quanti byte di dati sono stati trasferiti.
- Un' alterrnativa alle system call *non bloccanti* è costituita dalle chiamate del sistema **asincrone**.
  - Con esse, l' applicazione continua ad essere eseguita e il completamento dell' I/O viene successivamente comunicato all' applicazione per mezzo di una variabile o di un interrupt software o tramite una routine di ritorno.

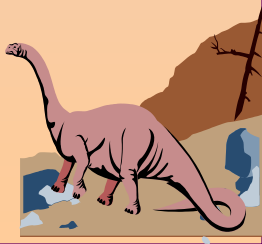
Sottosistema per l'I/O del nucleo: scheduling

- Il nucleo fornisce i servizi riguardanti l'I/O.
  - Molti sono offerti dal sottosistema per l'I/O del nucleo e sono realizzati a partire dai dispositivi e dai relativi driver.
- Fare lo scheduling di un insieme di richieste di I/O significa stabilirne un ordine di esecuzione efficace.
- Lo scheduling è necessario per:
  - concorrere a migliorare le prestazioni globali del sistema
  - distribuire equamente gli accessi ai dispositivi
  - ridurre il tempo di attesa medio per il completamento di un' operazione di I/O associata ad un dispositivo
  - Ad es. riordinamento delle sequenze di servizio di richieste relative ad operazioni di lettura su disco.



Sottosistema per l'I/O del nucleo: scheduling (II)

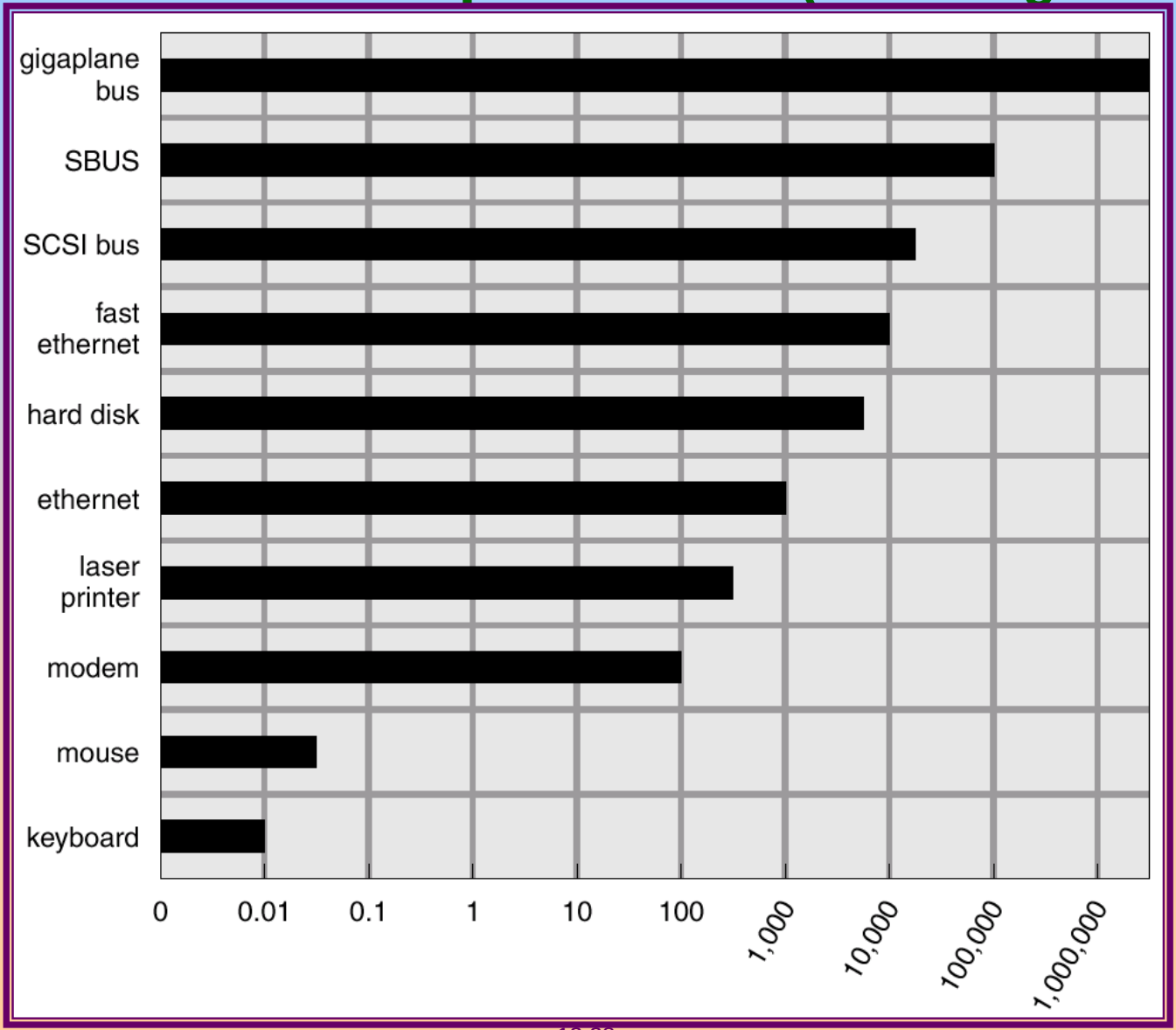
- I progettisti di S.O. realizzano lo scheduling mantenendo una coda di richieste di I/O associata ad ogni dispositivo.
  - Quando una applicazione richiede l'esecuzione di una chiamata a sistema di I/O bloccante, si aggiunge alla coda relativa al dispositivo la richiesta.
  - Lo scheduler dell'I/O riorganizza l'ordine della coda per migliorare l'efficienza globale del sistema ed il tempo medio di attesa cui sono sottoposte le applicazioni.
- Lo scheduling dell'I/O è uno dei modi in cui il sottositema per l'I/O migliora l'efficienza di un calcolatore.
- Un altro è l'uso di spazio di memorizzazione nella memoria centrale o nei dischi, per tecniche di memorizzazione transitoria, uso di cache e di code per la gestione asincrona dell'I/O.



Memorizzazione transitoria

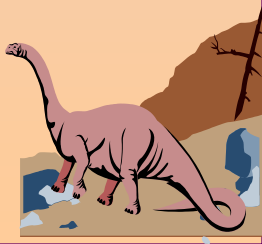
- Una **memoria di transito** (buffer) è un'area di memoria che contiene dati mentre essi sono trasferiti tra due dispositivi o tra una applicazione e un dispositivo.
- La memorizzazione transitoria viene utilizzata per tre ragioni principali:
  - Sincronizzare dispositivi caratterizzati da diverse velocità di trasferimento (ad es. modem disco: doppia memorizzazione transitoria).
  - Gestione dei dispositivi che trasferiscono blocchi di dati in blocchi di dimensione diverse: (ad es. scambi di messaggi in rete).
  - Realizzare la *semantica delle copie:* meccanismo per la coerenza delle informazioni (ad es. applicazione disco).
    - Si supponga che una applicazione disponga di una area di memoria contenente dati da trasferire su disco e che richieda al sistema di effettuare una write su disco.
    - Cosa succede se dopo che la chiamata a sistema restituisce il controllo all'applicazione questa modifica il contenuto della memoria?
    - La semantica delle copie garantisce che la versione dei dati scritta su disco sia conforme a quella al momento della chiamata a sistema.

Velocità di trasferimento dei dispositivi di un SUN Enterprise 6000 (scala logaritmica)



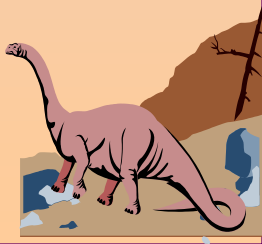
Cache

- Una *cache* è una memoria ad alta velocità utilizzata dai dispositivi di I/O per rendere più efficace il trasferimento di dati.
- Essa viene utilizzata per mantenere copie di dati contenuti in un dispositivo.
- L'uso della cache e l'uso delle memorie di transito sono due funzioni distinte.
- Caching vs buffering (memorie di transito):
  - a volte una stessa regione di memoria può essere usata per entrambi gli scopi (es. I/O da disco),
  - differiscono poichè un buffer contiene dati di cui non esiste un'altra copia, mentre la cache contiene, per definizione, copie di informazioni già memorizzate.



Code ed uso esclusivo dei dispositivi

- Una coda di file da stampare (spool) è una memoria di transito contenente dati per un dispositivo che non può accettare flussi di dati intercalati (ad es. una stampante).
- La tecnica dell'accodamento (spooling) viene realizzata utilizzando un buffer particolare: lo *spool.*
- La tecnica dello *spooling* viene utilizzata per ottenere un uso esclusivo dei dispositivi
  - Per esempio, quando più applicazioni richiedono simultaneamente la stampa del proprio output su di una stampante, è necessario l' utilizzo di questa tecnica per gestire le stampe in modo esclusivo, evitando così stampe di documenti con dati interfogliati.
- Quindi consente al S.O. di coordinare un output simultaneo.
- In alcuni S.O. lo spooling è gestito da un processo di sistema specializzato (*daemon di spool*), in altri da un thread del kernel.

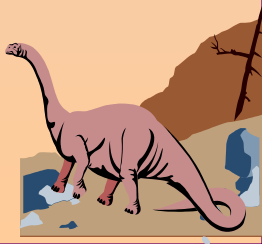


Gestione degli errori

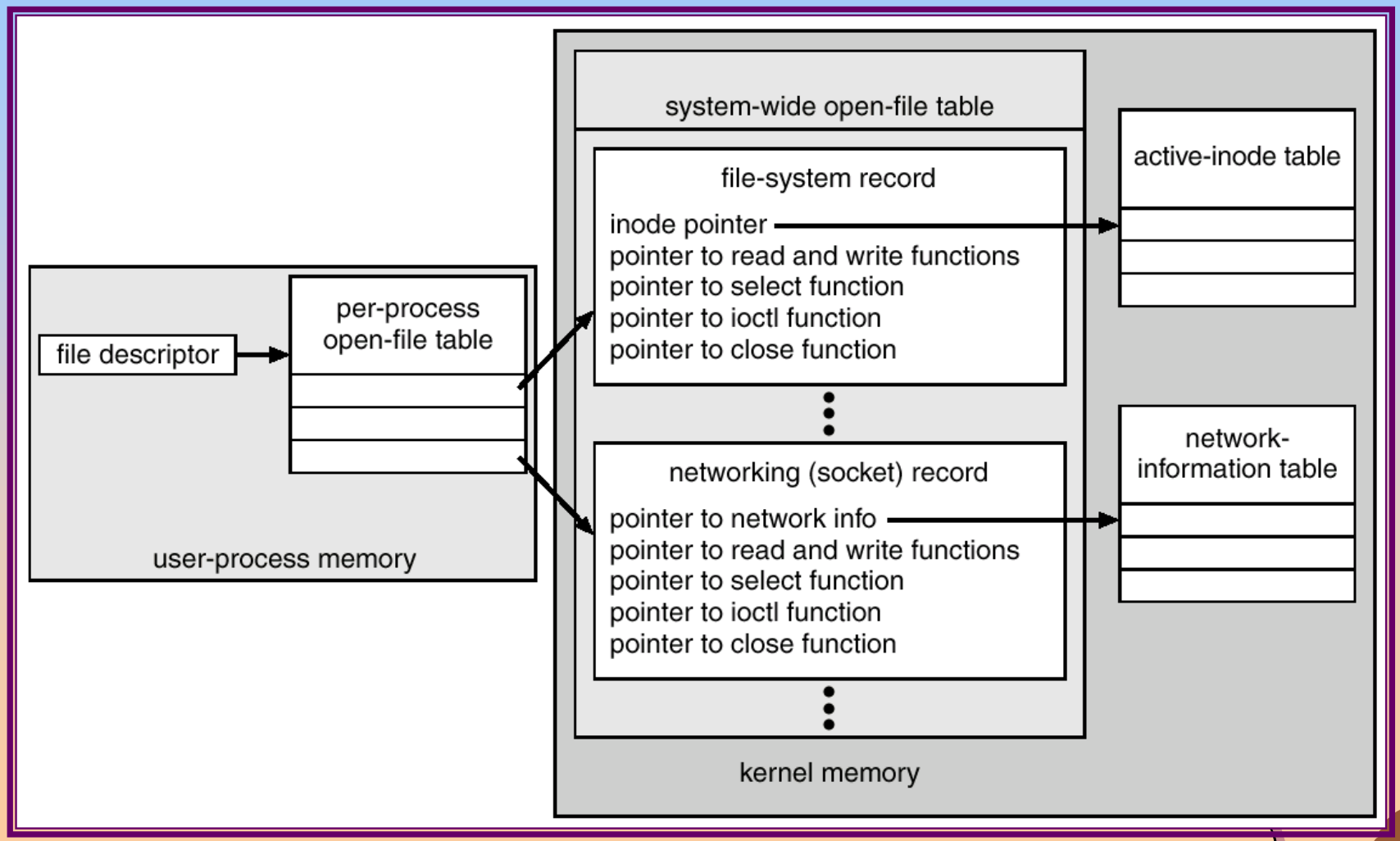
- Il S.O. deve proteggersi dal malfunzionamento dei dispositivi.
- Gli errori possono essere:
  - transitori
    - ad es. rete sovraccarica
  - permanenti
    - ad es. disco rotto.
- Nel caso di situazioni transitorie, solitamente il S.O. può tentare di recuperare la situazione
  - ad es. richiedere che si effettui di nuovo l'operazione di I/O.
- Le chiamate di sistema, quando non vanno a buon fine, segnalano una situazione di errore.
- In situazioni con errori permanenti, la gestione degli errori è più difficile.
- Alcuni S.O., come UNIX, forniscono una mappatura degli errori.
- Alcuni tipi di hardware, poichè gli errori sono in relazione al tipo di dispositivo, forniscono un protocollo per la rilevazione dell'errore.
  - Ad es. Standard SCSI.

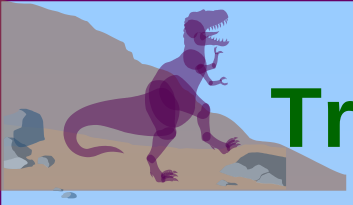
Strutture di dati del nucleo

- Il S.O. utilizza particolari strutture dati interne al kernel per mantenere informazioni riguardo allo stato dei componenti coinvolti nelle operazioni di I/O.
  - Ad esempio la tabella dei file aperti.
- Sia UNIX che Window NT risolvono il problema dell'ottenere una struttura dati che consenta di effettuare migliori operazioni, seguendo quello che è l' approccio della *programmazione orientata agli oggetti*;
  - cioè aggregano nella struttura dati sia dati che metodi (funzioni).



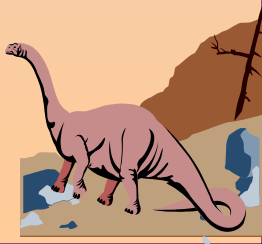
Strutture di dati del nucleo per la gestione dell' I/O in UNIX

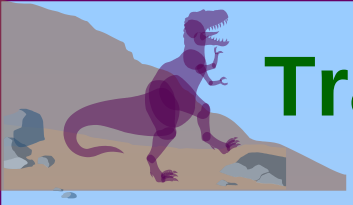




Trasformazione delle richieste di I/O in operazioni dei dispositivi

- Come si può giungere da un nome di file ad un controller di un dispositivo?
- In MS-DOS la prima parte del nome di un file identifica in modo univoco una periferica:
  - ad es. **C:** è la parte iniziale di ogni nome di file residente nell'unità dischi principale,
  - per convenzione del S.O. a **C:** viene associato uno specifico indirizzo di porta per mezzo di una tabella dei dispositivi.

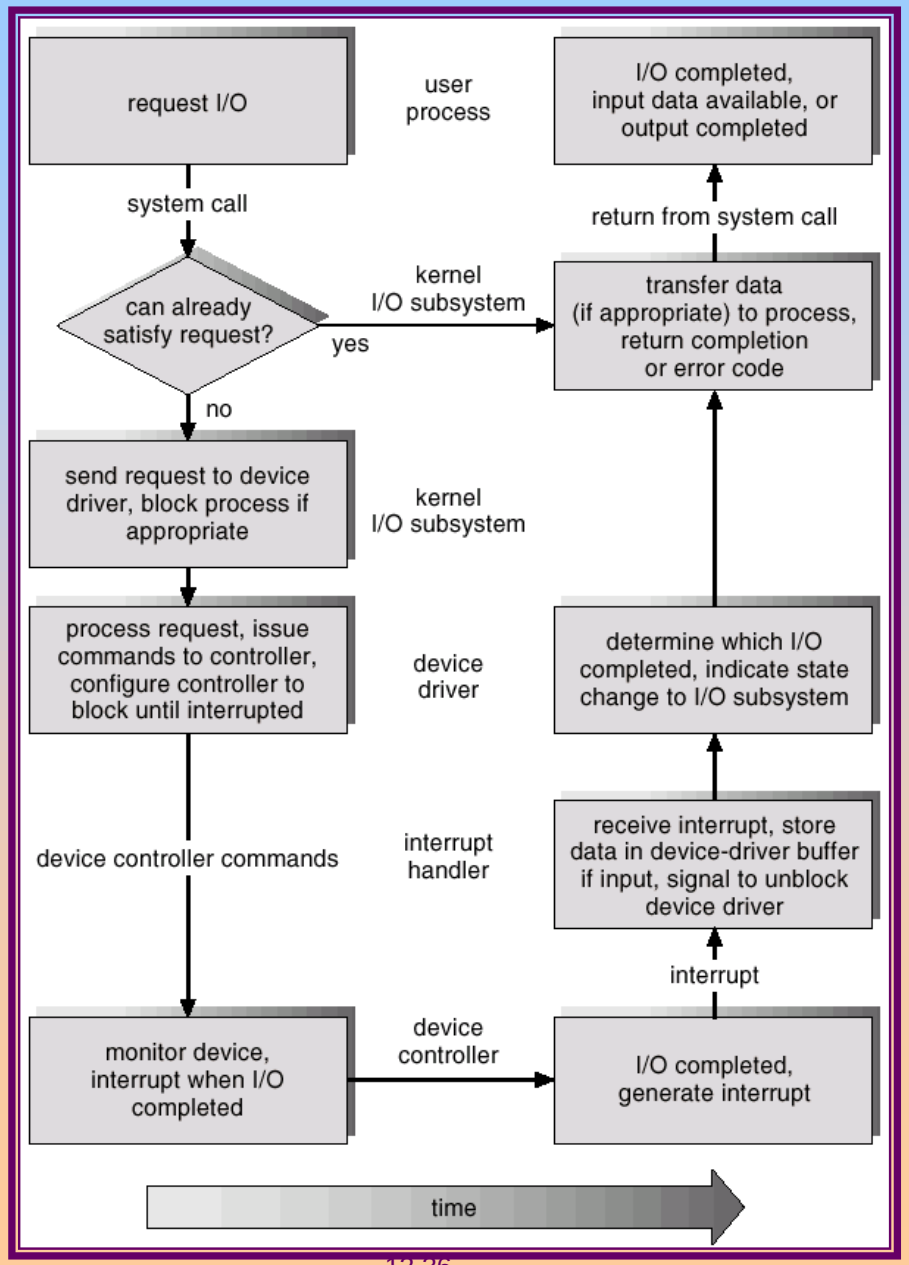




Trasformazione delle richieste di I/O in operazioni dei dispositivi (II)

- In UNIX non c'è nel pathname una vera separazione fra il dispositivo interessato ed il nome del file.
- Viene impiegata una **tabella di montaggio**, per associare un prefisso di pathname ad uno specifico nome di dispositivo.
- Quando deve risolvere il nome di percorso, il sistema esamina la tabella per trovare il più lungo prefisso corrispondente:
  - questo elemento indica il nome del dispositivo voluto.
- Il nome del dispositivo viene rappresentato come un oggetto del file system,
  - non come un i-node ma come una coppia di numeri **<principale,secondario>**.
  - **principale** individua il driver del dispositivo che deve essere usato per gestire l' I/O.
  - **secondario** individua l'indirizzo della porta o l'indirizzo mappato in memoria del controller del dispositivo.

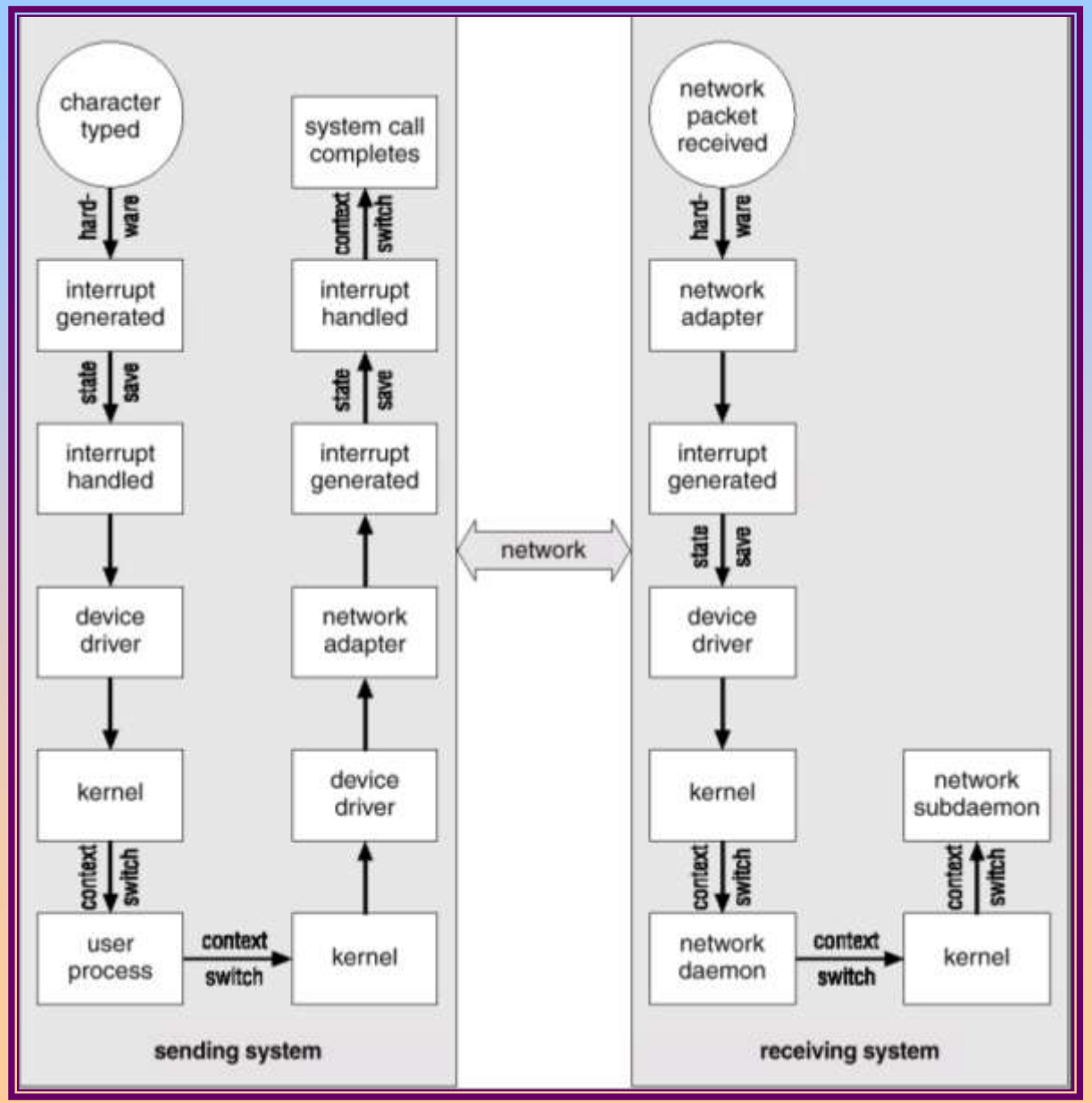
Esecuzione di una richiesta di I/O



Prestazioni

- L'I/O è un fattore predominante nelle performance di un sistema:.
  - Consuma tempo di CPU per eseguire i driver e il codice kernel di I/O
  - Continui cambi di contesto all'avvio dell' I/O e alla gestione degli interrupt.
  - Trasferimenti dati da/per I buffer consumano cicli di clock e spazio in memoria.
  - Il traffico di rete è particolarmente pesante (es. telnet).
- Per migliorare l'efficienza dell'I/O si possono applicare diversi principi:
  - Ridurre il numero dei cambi di contesto.
  - Ridurre il numero di copiature dei dati nella memoria durante i trasferimenti tra dispositivi e applicazioni.
  - Ridurre la frequenza degli interrupt preferendo trasferimenti di grandi quantità di dati: usare controller intelligenti e l'interrogazione ciclica.
  - Aumentare il tasso di concorrenza usando controllori DMA o bus dedicati, per sollevare la CPU dalle semplici copiature di dati
  - Implementare le primitive in hardware, dove possibile, per aumentare il parallelismo
  - Equilibrare le prestazioni di CPU, del sottosistema per la gestione della memoria, del bus e dell'I/O: Il sovraccarico di un elemento comporta l'inutilizzo degli altri.

Comunicazione tra computer e suoi costi



Successione delle realizzazioni dei servizi di I/O