Sistemi di I/O

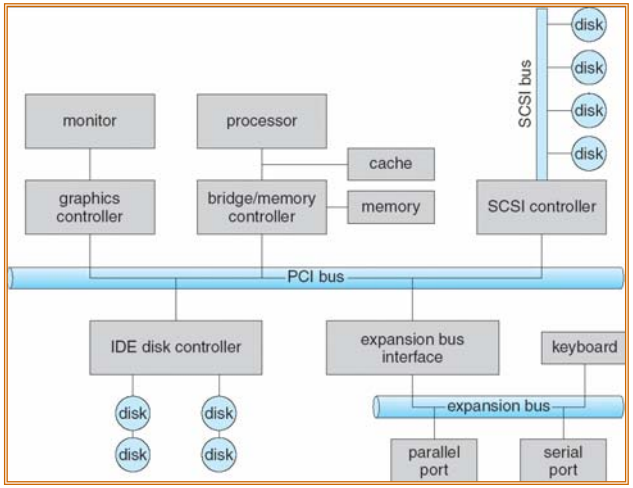
Contenuti

- L'hardware di I/O.
- Le interfacce I/O per le applicazioni.
- Il sottosistema di I/O del kernel.
- Trasformazione di richieste di I/O in operazioni hardware.
- Streams.
- Prestazioni.

L'hardware di I/O

- Molteplici dispositivi di I/O.
- Concetti comuni:
  - porte (punti di connessione – accessibili tramite indirizzi)
  - bus (*shared direct access* o *daisy chain*),
  - controller (circuiteria +/- semplice).
- I controller dei dispositivi hanno registri per ricevere dati e comandi.
  - Istruzioni di I/O.
- I registri possono anche essere aree di memoria RAM:
  - I/O mappato in memoria.

Bus in un PC



Indirizzi di alcune porte di I/O in un PC

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)

Struttura di una porta di I/O

- registro **data-in**, letto dalla CPU per ricevere input
- registro **data-out**, scritto dalla CPU per inviare output
- registro **status**, contiene bit letti dalla CPU (e.g., comando completato, byte disponibile, errore…)
- registro **control**, contiene bit scritti dalla CPU (e.g., avvia un comando, cambia modo del device …)

I registri di dati memorizzano tipicamente da 1 a 4 byte

I/O Programmato. Polling

- La CPU determina lo stato del diapositivo:
  - Bit busy nel registro status
- Comunica richieste al dispositivo pronto
  - Bit command-ready nel registro control
- Il computer è in attesa attiva (*busy-waiting*): continua a leggere il registro status finchè il bit busy non risulta 0.

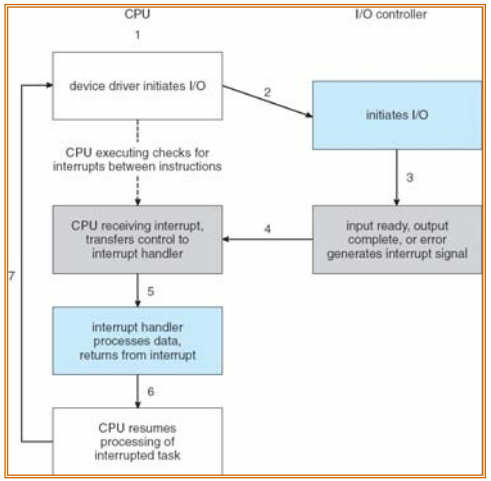
Handshaking

1. L'host legge il bit busy fino a quando non è zero
2. Setta il bit write nel registro control e scrive un byte in data-out
3. L'host setta il bit command-ready
4. Quando il controller nota command-ready, setta il bit busy
5. Il controller legge il registro control e trova il comando *write*. Legge da data-out ed effettua *write* sul device
6. Il controller azzera il bit command-ready, azzera il bit error e azzera il bit busy

Interrupt

- Il dispositivo di I/O attiva la linea di richiesta di interrupt della CPU (*interrupt-request line*).
- Il gestore degli interrupt (*interrupt-handler*) riceve il segnale di interrupt e lo evada attivando l'opportuna routine.

Ciclo di I/O guidato da interrupt



Meccanismo degli interrupt

- Proprietà che devono essere soddisfatte
  - Poter posporre interrupt in momenti critici
  - Metodo efficiente per passar il controllo all'appropriato gestore dell'interrupt
  - Diversi livelli di interrupt (+ importanti / -importanti)

Meccanismo degli interrupt

- Molte CPU hanno due linee di interrupt: mascherabili e non mascherabili.
- Il vettore di interrupt permette di inviare l'interrupt al gestore opportuno
  - richiesta = indirizzo (i.e., scostamento in una tabella)
  - tabella troppo grande (interrupt chaining)
  - gestione degli interrupt basato su priorità.

Vettore degli eventi in un processore 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
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Sistema operativo e interruzioni

- Il sistema operativo interagisce con il meccanismo degli interrupt in vari modi eccezioni
  - Avvio del sistema
  - I/O
  - Eccezioni
  - Memoria virtuale
  - System call
  - Controllo del flusso nel kernel (e.g., lettura disco con coppia di gestori)

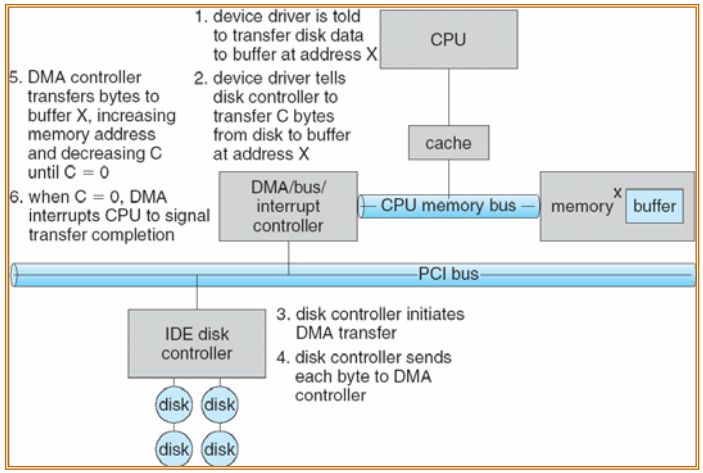
Accesso diretto alla memoria

- Usato per evitare l'I/O programmato per grossi trasferimenti di dati.
- Richiede un controller di DMA.
- Non richiede l'intervento della CPU nei trasferimenti dati dispositivo-memoria
- CPU scrive in memoria un comando strutturato per il DMA e comunica al controller DMA l'indirizzo di memoria in cui si trova il comando

Controller DMA / Controller Dispositivo

1. Controller dispositivo
   - Manda un segnale sulla linea DMA-request quando ci sono dati disponibili per trasferimento
2. Controller DMA
   - Prende possesso del bus di memoria, presenta l'indirizzo desiderato ai fili d'indirizzamento della memoria e manda un segnale sulla linea DMA-acknowledge
3. Controller dispositivo
   - Trasferisce dati in memoria e rimuove il segnale dalla linea DMA-request

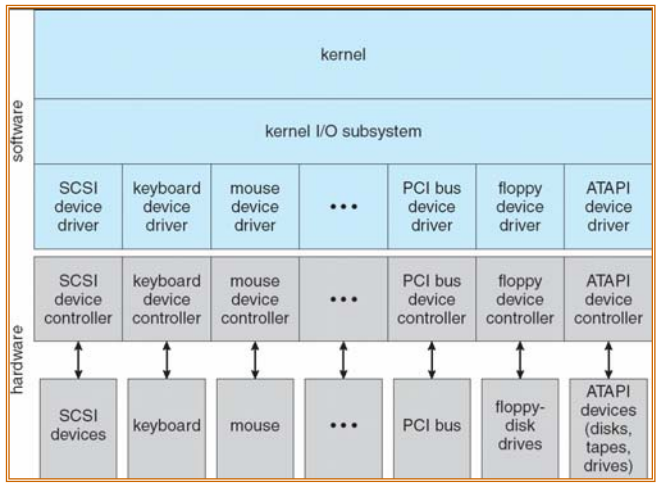
Trasferimento dati tramite DMA



Interfaccia I/O per le applicazioni

- Tipi di dispositivi. Accesso attraverso un unico insieme di funzioni (i.e., interfaccia)
- Lo strato dei **device driver** nasconde al sottosistema di I/O le differenze tra i controller delle periferiche in modo simile a come le chiamate di sistema di I/O incapsulano il comportamento delle periferiche in poche classi generiche che nascondono le differenze alle applicazioni

Struttura del sottosistema di I/O del kernel



Caratteristiche dei dispositivi

- Le periferiche possono differire in molti aspetti:
  - trasferimento dati a carattere o a blocchi;
  - accesso sequenziale o diretto;
  - periferica condivisibile o dedicata;
  - velocità di elaborazione;
  - lettura e scrittura, sola lettura o sola scrittura.

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 read–write	CD-ROM graphics controller disk

Dispositivi a blocchi e a caratteri

- I dispositivi a blocchi includono i drive di disco.
  - I comandi includono *read, write, seek*
  - I/O grezzo (raw I/O) o accesso al file system.
  - Accesso ai file mappati in memoria.
- I dispositivi a caratteri includono tastiere, mouse e porte seriali.
  - I comandi includono *get, put*.
  - Librerie al livello superiore permettono l'accesso ad intere sequenze di caratteri.

Periferiche di rete

- Differiscono in maniera significativa dai dispositivi a blocchi e a caratteri per avere una propria interfaccia.
- Unix e Windows NT/9*x*/2000: *socket di rete*.
- Un'applicazione crea un socket per:
  - legare il socket locale ad un indirizzo remoto
  - ascoltare richieste di connessione da applicazioni remote
  - inviare e ricevere pacchetti
- Fornisce la funzione *select* per implementare server
- Altre astrazioni (pipe, FIFO, stream, code di messaggi).

Orologi e temporizzatori

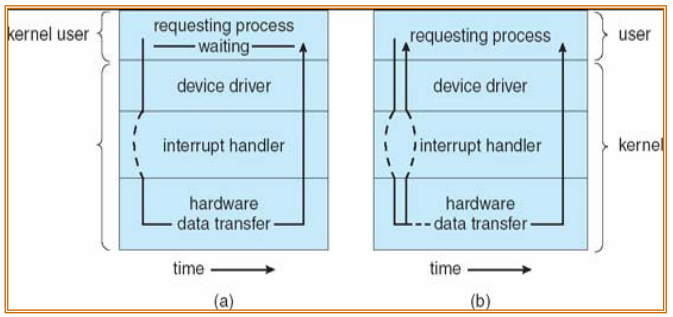
- Forniscono l'ora corrente, il tempo trascorso, e timer.
- Un *temporizzatore programmabile* genera interrupt allo scadere di un intervallo di tempo. Possono essere periodici.

Alcuni sistemi operativi, oltre all'interfaccia di I/O, forniscono una "backdoor" per comunicare con i device driver. La chiamata `ioctl` (su UNIX) permette di gestire alcuni aspetti dell'I/O direttamente, passando comandi direttamente al device driver.

I/O bloccante e non bloccante

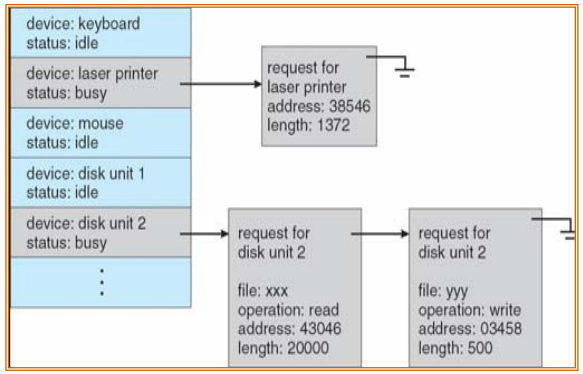
- Bloccante – l'esecuzione viene sospesa finchè I/O non è completato.
  - Facile da usare e capire.
  - Insufficiente per alcune necessità.
- Non bloccante – la chiamata di I/O ritorna immediatamente se la periferica non è disponibile.
  - Ritorna rapidamente con il conteggio dei byte letti o scritti.
  - Multi-threading.
- Asincrono – il processo è in esecuzione concorrentemente all'I/O.
  - Più difficile da usare.
  - Il completamento di un I/O viene comunicato all'applicazione tramite l'invio di un segnale o un interrupt software.

Sincrono ed asincrono



Il sottosistema di I/O del kernel

- **Schedulazione**:
  - ordinamento di richieste di I/O - una coda di dispositivo;
  - alcuni SO garantiscono equità.

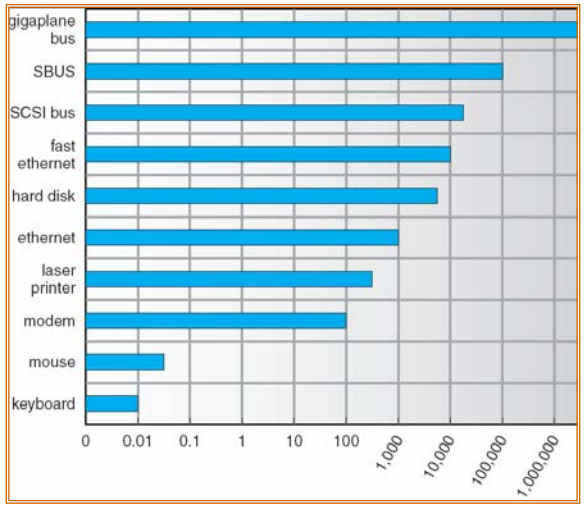


Il sottosistema di I/O del kernel

Buffering - immagazzina i dati in memoria mentre vengono trasferiti fra due periferiche.

- Per far fronte a differenti velocità fra dispositivi.
- Per far sì che il buffer funga da adattatore fra periferiche aventi differenti dimensioni dei blocchi di dati (i.e., impacchettamento).
- Per supportare la "sematica della copia", i.e. garantire che il blocco che si copia su disco è lo stesso in memoria al momento della system call.

Velocità di trasferimento delle periferiche per Sun Enterprise 6000



Velocità delle porte di I/O di un PC

Porta	Velocità massima
Porta Seriale RS232	115 Kbps
Porta Parallela standard	150 Kbps
Porta Parallela ECP	2 Mbps
USB 1.11	2 Mbps
USB 2.0	480 Mbps
FireWire 400	400 Mbps
FireWire 800	800 Mbps
Bluetooth 1.1	723 Kbps
Bluetooth 2.02.	1 Mbps

Il sottosistema di I/O del kernel

- **Caching** – memoria veloce che conserva una copia dei dati.
  - Sempre una sola copia.
  - Fondamentale per le prestazioni.
- **Spooling** – conserva l'output per una periferica.
  - Se la periferica può servire solo una richiesta alla volta, e.g., una stampante.
- **Prenotazione dei dispositivi** – fornisce accesso esclusivo ad una periferica.
  - Chiamate di sistema per allocare e deallocare.
  - Attenzione ai deadlock.

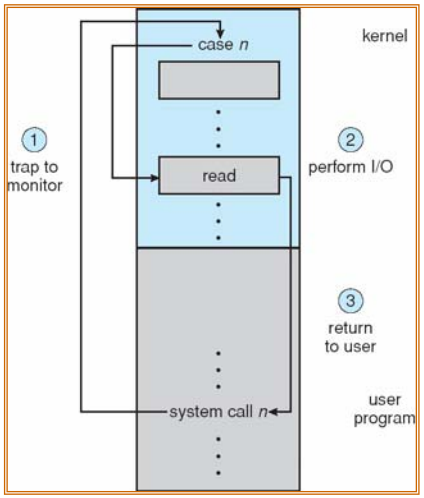
Gestione degli errori

- Il sistema operativo deve recuperare da fallimenti temporanei, e.g., lettura e scrittura su disco, indisponibilità di dispositivo ...
- La maggior parte riportano un numero di errore o un codice quando la richiesta di I/O fallisce.
- Alcuni controller hardware (e.g., SCSI) forniscono informazioni dettagliate sugli errori verificatisi

Protezione dell'I/O

- Un processo utente può accidentalmente o volontariamente tentare di danneggiare il corretto funzionamento tramite istruzioni di I/O illegali
  - Tutte le istruzioni di I/O sono definite privilegiate
  - L' I/O deve essere effettuato attraverso system call
    - Le locazioni usate per l'I/O mappato in memoria e per le porte di I/O devono essere protette

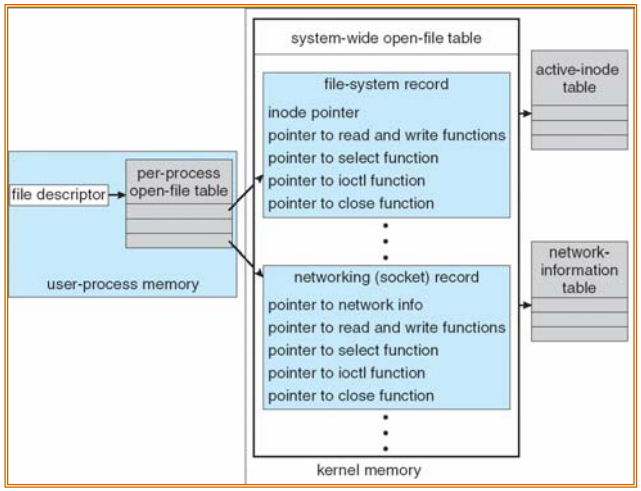
Uso di system call per I/O



Strutture dati del kernel

- Il kernel mantiene informazioni di stato per le componenti di I/O, inclusi le tabelle dei file aperti, le connessioni di rete, stato dei dispositivi a carattere.
- Il kernel utilizza strutture dati per tenere traccia dei buffer, dell'allocazione di memoria e dei blocchi "sporchi".
- Alcuni usano metodi orientati agli oggetti o basati sullo scambio di messaggi per implementare l'I/O.

Struttura dell'I/O nel kernel di UNIX



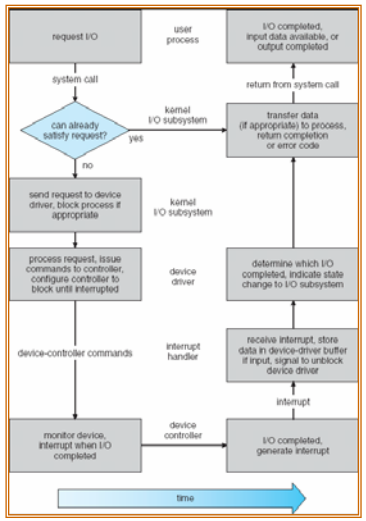
Richieste di I/O e operazioni hardware

- Consideriamo la lettura di un file da disco per un processo:
  - Determinare il dispositivo che memorizza il file.
  - Tradurre il nome nella corrispondente rappresentazione nel dispositivo.
  - Leggere fisicamente i dati dal disco nel buffer.
  - Rendere i dati disponibili al processo richiedente.
  - Restituire il controllo al processo.

Dal nome del file al controller del dispositivo

- MS-DOS – la prima parte del nome identifica il dispositivo
  C:\path
  All'interno del kernel C: - attraverso una tabella di dispositivi - viene mappato su una specifica porta
- UNIX – nessuna parte della path è il nome del dispositivo
- Una **mount table** associa il prefisso di una path ad un nome di dispositivo. Nella struttura del file system questo nome non corrisponde ad un numero di inode ma un numero di device <major, minor>
- major – device driver, minor – indice tabella device

Ciclo di vita di una richiesta di I/O



STREAM – UNIX SYSTEM V

- Una connessione full-duplex (bi-direzionale) tra un processo utente e il driver di un dispositivo.
- Uno STREAM consiste di:
  - *Head:* si interfaccia con il processo utente.
  - *Driver end* : si interfaccia con il dispositivo.
  - Zero o più *moduli* tra di esse.
- Ogni modulo contiene una **coda di lettura** e una **coda di scrittura.**
- Lo scambio di messaggi è utilizzato per comunicare tra le code.

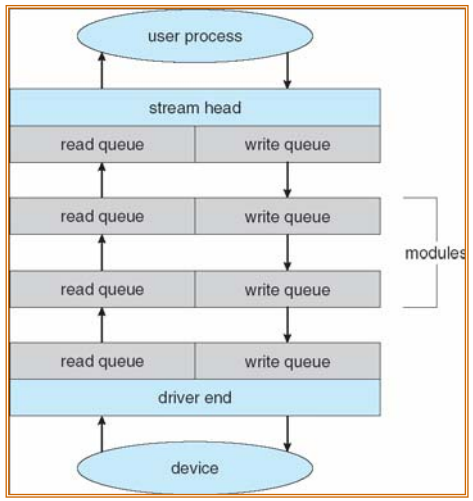
Struttura di uno STREAM

I moduli sono inseriti tramite la sys call **ioctl()**

Le code supportano il controllo di flusso

Un processo scrive con **write()** o **putmsg()** e legge con **read()** o **getmsg()**

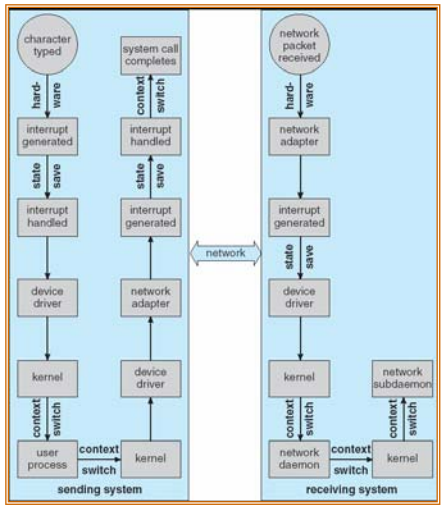
In Unix System V e Solaris i socket sono implementati usando STREAM



Prestazioni

- L'I/O è un fattore di notevole importanza per le prestazioni di un sistema.
  - Chiede alla CPU di eseguire il codice dei driver delle periferiche e il codice del sottosistema di I/O del kernel.
  - I cambi di contesto dovuti agli interrupt sono onerosi.
  - Copia dei dati sovraccarica il bus.
  - Anche un traffico di rete alto incrementa i cambi di contesto.

Comunicazione tra computer



Migliorare le prestazioni

- Ridurre il numero di cambi di contesto.
- Ridurre la copia dei dati.
- Ridurre la frequenza degli interrupt usando grandi trasferimenti, controller intelligenti e la scansione periodica (*polling*).
- Usare il DMA (e front-end processor o I/O channel)
- Bilanciare l'uso della CPU, della memoria, del bus e dei dispositivi di I/O per elevare il throughput.

Sviluppo di una funzionalità per una periferica