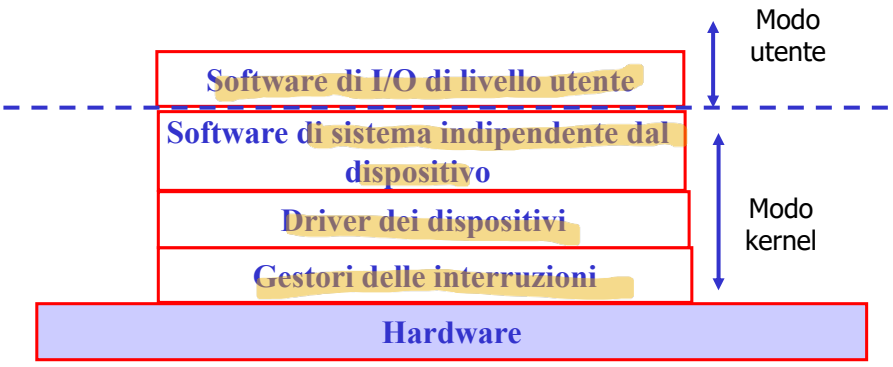
Input/Output

1

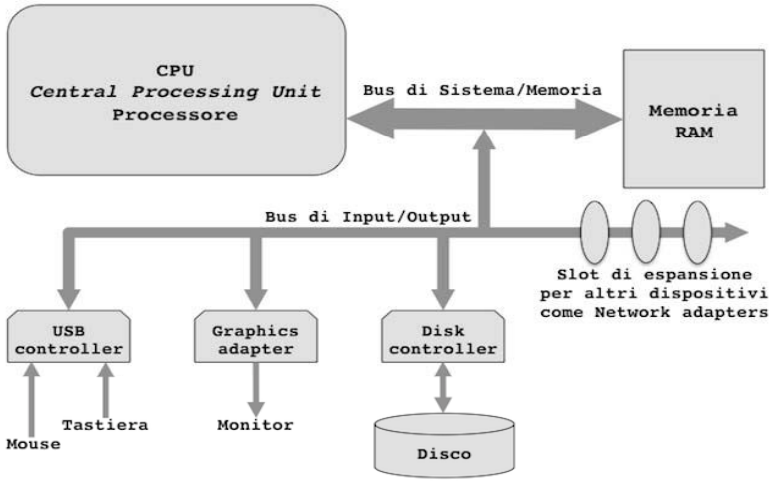
Livelli del sottosistema di I/O



2

Chi offre periferiche offre pezzo di SW necessario a gestire la periferica.
Questo deve interfacciarsi con il sistema operativo.
Quello VOS, perché contiene interfaccia con driver.

I controllori dei dispositivi



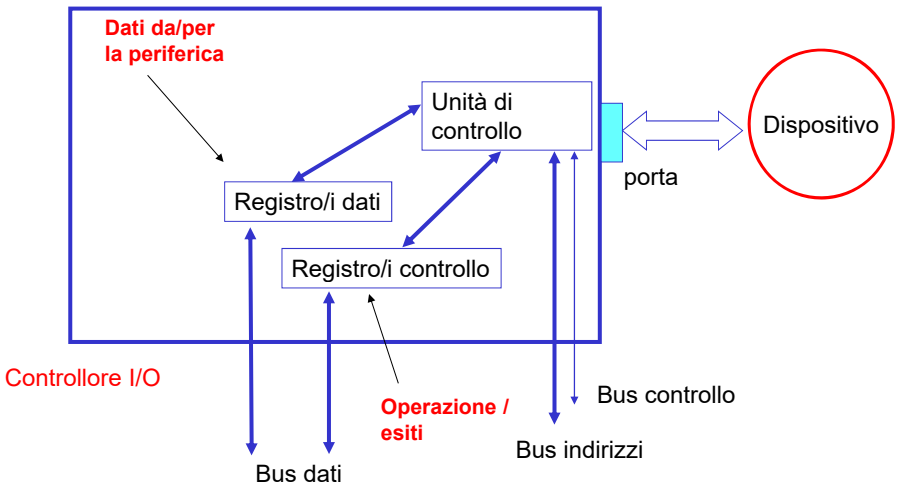
La parte meccanica viene controllata da un controllore del dispositivo (device controller) o adapter

3

Punto del bus: filo di interconnessione, CPU, memoria e dispositivi periferici. Tra bus e dispositivo vero e proprio c'è un controllore, che fa da interfaccia.

I controllori dei dispositivi (2)

- Esempio di controllore di una porta.



4

Come è vista da elaboratore? Scheda elettronica offre qualcosa di programmabile. Cosa è associato a schede? Una serie di registri. La logica che interfacciando è quella classica che ha CPU quando legge e scrive da memoria centrale: leggere o scrivere dei registri (non di memoria ma del controllore).
Registro dati: sorta di buffer che fa trasferire dati da memoria a periferica di output. Per periferiche di input questo diventa il posto da cui dati input sono letti.

Registri di controllo: collegati a unità di controllo. Inviamo ordini a dispositivo fisico. Servono quindi = interfaccia comando e periferica.
Registri di stato: utile che driver può leggere per capire cosa sta succedendo nel controller.
Driver interagisce con registri.
NOTA: controllore è molto personalizz. Devo scrivere bene come lo programmo.

I controllori dei dispositivi (3)

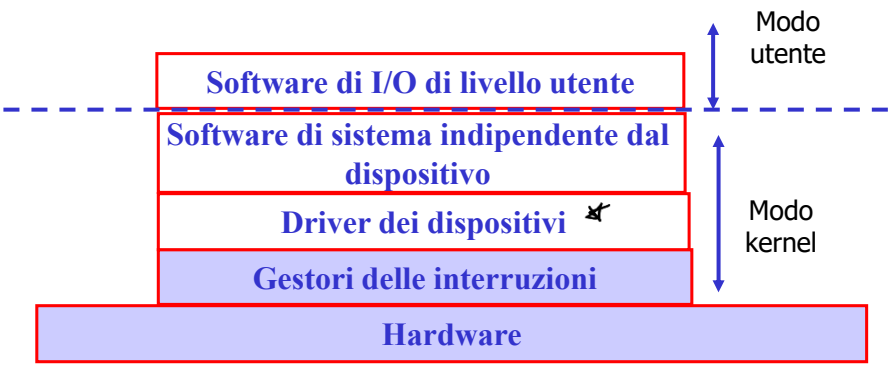
- Problema 1 : come si accede ai registri dei controllori ?
  - Si utilizzano istruzioni assembler 'speciali' per l'I/O (es. `IN`, `OUT`)
    - ogni registro dei controllori è contraddistinto da un numero di porta di I/O es. `IN R0, 4`
  - Si 'mappano' i registri del controllore su particolari indirizzi di memoria, e si utilizzano le normali `LOAD/STORE` (*memory mapped I/O*)

I/O MAPPED
Devo associare a registri degli indirizzi.
MEMORY MAPPED
Oppure:
Alcuni indirizzi non sono indirizzati di memoria => ho dei buchi.

5

Creo buchi virtuali ma non uso istruz. di input e output
Posso usare anche altre operazioni: bit-test e vedo se manca la carta

Livelli del sottosistema di I/O



6

* Per fare op. anche semplici di I/O richiede fare operazioni anche con organi meccanici. Può avere tempi anche molto lunghi. Driver tipicamente tende a lavorare sotto interruzione: non si mette in attesa a fare bit-test continui.
Driver fatto in 2 pezzi: uno che avvia operazione, uno che riceve segnale da operazione. Driver deve essere schedulato, con interrupt ecc. Si collega ISR associata a operazione.

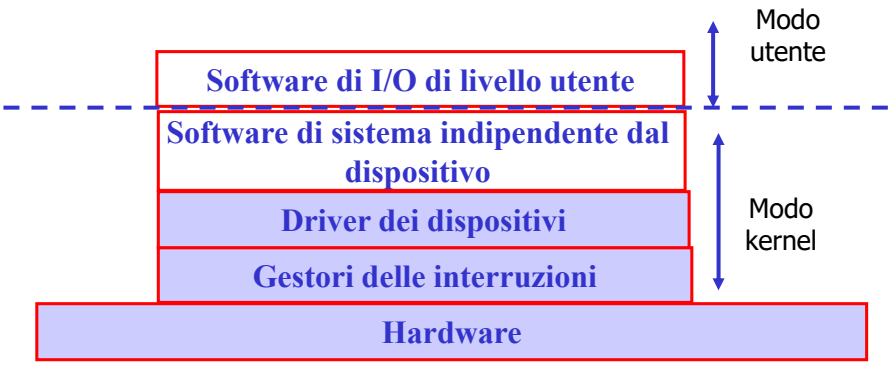
Gestori delle Interruzioni (*Interrupt Handlers*)

- Tipicamente i driver sono bloccati in attesa che arrivi una interruzione dalla periferica che stanno controllando
  - es: il driver si blocca dopo aver iniziato una operazione di I/O
- Quando arriva una interruzione
  - viene mandato in esecuzione il gestore delle interruzioni (GI) di quel tipo (selezionato in base al vettore di interruzione)
  - GI sblocca il driver utilizzando un opportuno meccanismo di IPC

↳ inter process communication: si chiama pezzo giusto del driver che trova info in qualche locazione di memoria.

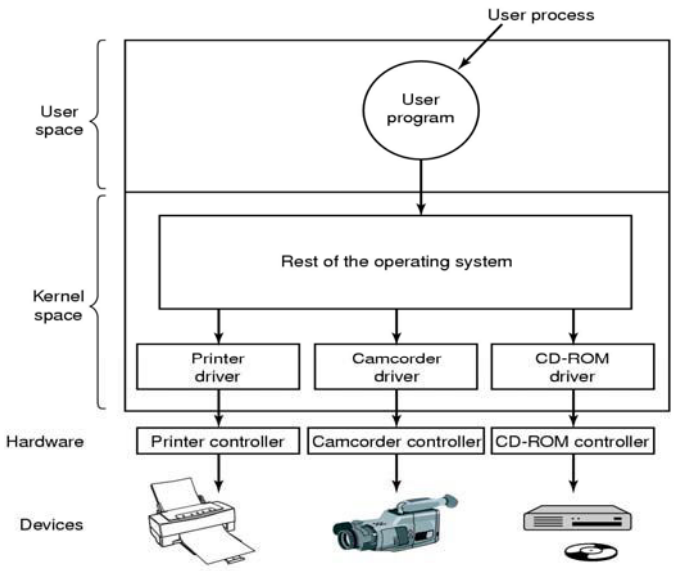
7

Livelli del sottosistema di I/O



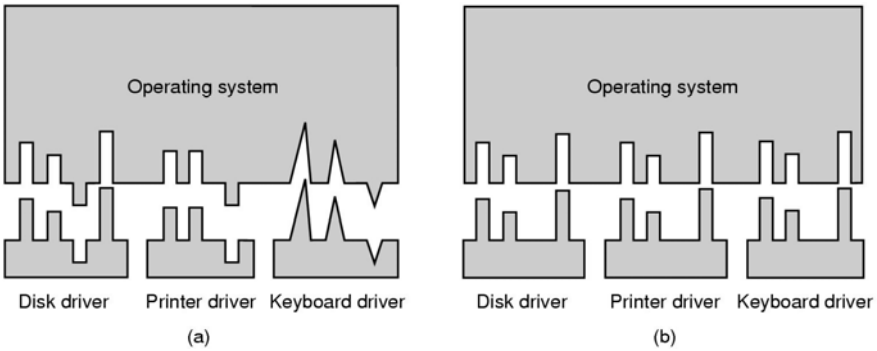
8

Driver dei Dispositivi (1)



9

Device-Independent I/O Software



(a) Without a standard driver interface

(b) With a standard driver interface

10

Soluzione: ho interfaccia standard: ogni sistema op. chiede a chi sviluppa driver che offrono determinate funzioni o interfaccia.

Es: periferica di input a caratteri? Voglio che mi offra la funzione read.

Driver dei Dispositivi (2)

- Il driver di un dispositivo è la parte del sistema operativo che interagisce con il dispositivo
  - legge/scrive i registri di controllo
  - tratta le caratteristiche a basso livello
  - fornisce una interfaccia astratta del dispositivo indipendente dai dettagli hw al resto del sistema operativo
  - tipicamente è sviluppato dal costruttore del dispositivo (…)

11

→ deve poter usufruire di servizi di OS, quindi anche OS fornisce interfaccia. Se deve p.es. allocare spazio di memoria lo posso fare.

Driver dei Dispositivi (3)

Tipico funzionamento di un driver :

1. Inizializza il dispositivo
2. Accetta richieste di operazioni e ne controlla la correttezza
3. Gestisce le code delle richieste che non possono essere subito servite
4. Sceglie la prossima richiesta da servire e la traduce in una sequenza *S* di comandi a basso livello da inviare al controllore
5. Trasmette i comandi in *S* al controllore eventualmente bloccandosi in attesa del completamento dell'esecuzione di un comando
6. Controlla l'esito di ciascun comando gestendo eventuali errori
7. Invia l'esito dell'operazione ed eventuali dati al richiedente

12

Driver dei Dispositivi (4)

- Tipicamente le interfacce astratte fornite dai driver vengono classificate in due categorie principali :
  - *interfacce a blocchi (block-oriented)* :
    - la lettura/scrittura sul dispositivo fisico avviene un blocco alla volta,
    - tipicamente i dati scritti vengono bufferizzati nel SO finchè non si raggiunge l'ampiezza di un blocco
    - es : dischi, nastri …

13

→ quello che il SO si aspetta per poter leggere o scrivere su una periferica. (UNIX usa astrazione dei file per le periferiche)

Driver dei Dispositivi (5)

- Tipicamente le interfacce astratte fornite dai driver vengono classificate in due categorie principali (cont.):
  - *interfacce a caratteri (character-oriented)* :
    - la lettura/scrittura sul dispositivo fisico avviene un carattere alla volta,
    - non c'è bufferizzazione,
    - es : tipicamente tastiera, mouse,
    - es : si possono avere interfacce a caratteri anche per dischi, nastri

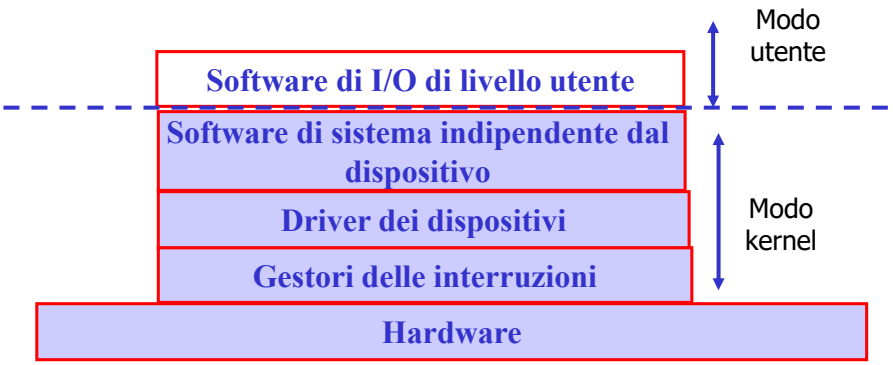
14

Driver dei Dispositivi (6)

- Alcuni driver possono fornire sia interfaccia a caratteri che a blocchi
  - es. driver del disco
- Le interfacce fissano un insieme di funzioni standard che tutti i driver devono implementare
  - es. tutti i driver che forniscono una interfaccia a blocchi forniscono una implementazione di una funzione read() per scrivere/leggere blocchi con formato fissato

15

Livelli del sottosistema di I/O



16

Software di sistema indipendente dal dispositivo (1)

- Accetta le richieste dal livello utente e invoca il driver opportuno utilizzando le funzioni di interfaccia
  - Meccanismi per risalire dal nome del dispositivo al driver
- Fornisce un insieme uniforme di SC invocabili da chi scrive il driver
  - allocazione di aree di memoria fisica contigua per i buffer
  - interazione con il controllore DMA, la MMU

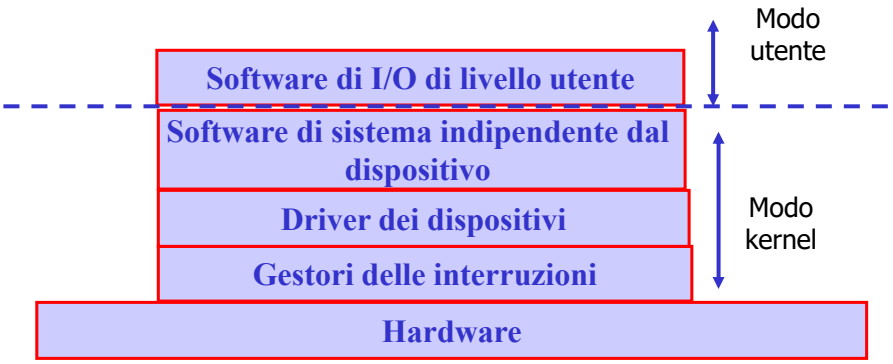
17

Software di sistema indipendente dal dispositivo (2)

- Funzioni del software di I/O indipendente dal dispositivo
  - bufferizzare le informazioni
  - segnalare, gestire errori
  - allocare e rilasciare dispositivi dedicati
  - fornire una dimensione del blocco indipendente dal dispositivo
  - fornire funzionalità di sistema ai driver attraverso una interfaccia uniforme

18

Livelli del sottosistema di I/O



19

Software di I/O in spazio utente

- Funzionalità del software di I/O che gira in spazio utente :
  - librerie linkabili da programmi utente (es. stdio, unistd ...)
    - passano i parametri alle SC nel modo giusto
    - gestiscono la formattazione (es. printf()..)
  - spooling
    - processo utente (demone di stampa)
    - directory di spool (in cui l'utente copia il file da stampare)

20