GESTIONE DELLE PERIFERICHE D'INGRESSO/USCITA

ARGOMENTI

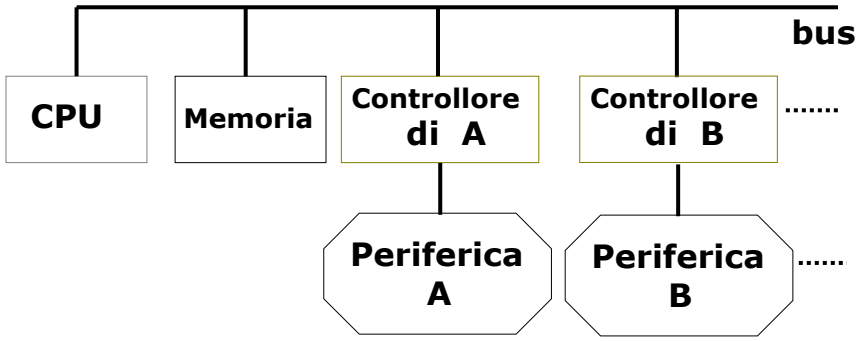
- Compiti del sottosistema di I/O
- Architettura del sottosistema di I/O
- Gestore di un dispositivo di I/O
- Gestione e organizzazione dei dischi

COMPITI DEL SOTTOSISTEMA DI I/O

1. Nascondere al programmatore i dettagli delle interfacce hardware dei dispositivi;
2. Omogeneizzare la gestione di dispositivi diversi;
3. Gestire i malfunzionamenti che si possono verificare durante un trasferimento di dati;
4. Definire lo spazio dei nomi (*naming*) con cui vengono identificati i dispositivi;
5. Garantire la corretta sincronizzazione tra un processo applicativo che ha attivato un trasferimento dati e l'attività del dispositivo.

COMPITI DEL SOTTOSISTEMA DI I/O

1. Nascondere al programmatore i dettagli delle interfacce hardware dei dispositivi



COMPITI DEL SOTTOSISTEMA DI I/O

2) Omogeneizzare la gestione di dispositivi diversi

dispositivo	velocità di trasferimento
tastiera	10 bytes/sec
mouse	100 bytes/sec
modem	10 Kbytes/sec
linea ISDN	16 Kbytes/sec
stampante laser	100 Kbytes/sec
scanner	400 Kbytes/sec
porta USB	1.5 Mbytes/sec
disco IDE	5 Mbytes/sec
CD-ROM	6 Mbytes/sec
Fast Etherneet	12.5 Mbytes/sec
FireWire (IEEE 1394)	50 Mbytes/sec
monitor XGA	60 Mbytes/sec
Ethernet gigabit	125 Mbytes/sec

COMPITI DEL SOTTOSISTEMA DI I/O

2) Omogeneizzare la gestione di dispositivi diversi

TIPOLOGIE DI DISPOSITIVI

- **Dispositivi a carattere (es. tastiera, stampante, mouse,...)**
- **Dispositivi a blocchi (es. dischi, nastri, ..)**
- **Dispositivi speciali (es. timer)**

COMPITI DEL SOTTOSISTEMA DI I/O

3. Gestire i malfunzionamenti che si possono verificare durante un trasferimento di dati

TIPOLOGIE DI GUASTI

- **Eventi eccezionali (es. mancanza di carta sulla stampante, end-of-file);**
- **Guasti transitori (es. disturbi elettromagnetici durante un trasferimento dati);**
- **Guasti permanenti (es. rottura di una testina di lettura/scrittura di un disco).**

COMPITI DEL SOTTOSISTEMA DI I/O

4. **Definire lo spazio dei nomi (*naming*) con cui vengono identificati i dispositivi**

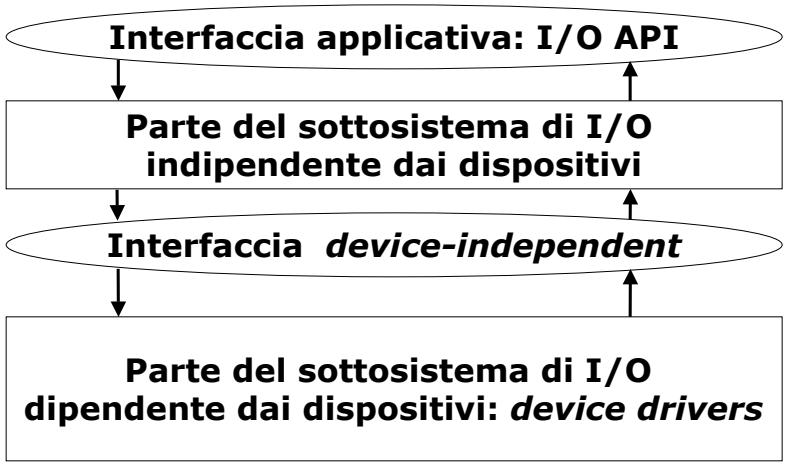
- **Uso di nomi unici (valori numerici) all'interno del sistema per identificare in modo univoco i dispositivi;**
- **Uso di nomi simbolici da parte dell'utente (*I/O API Input/Output Application Programming Interface*);**
- **Uniformità col meccanismo di *naming* del file-system.**

COMPITI DEL SOTTOSISTEMA DI I/O

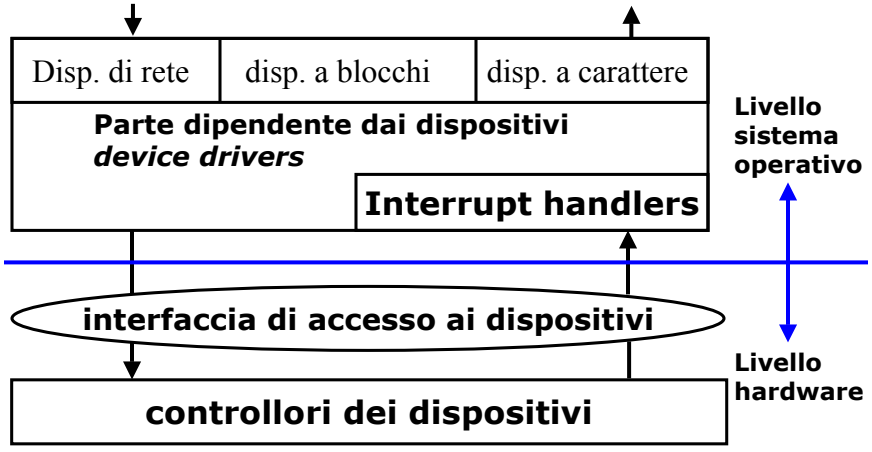
5. **Garantire la corretta sincronizzazione tra un processo applicativo che ha attivato un trasferimento dati e l'attività del dispositivo.**

- **Gestione sincrona dei trasferimenti: un processo applicativo attiva un dispositivo e si blocca fino al termine del trasferimento;**
- **Gestione asincrona dei trasferimenti: un processo applicativo attiva un dispositivo e prosegue senza bloccarsi;**
- **Necessita di gestire la "*bufferizzazione*" dei dati.**

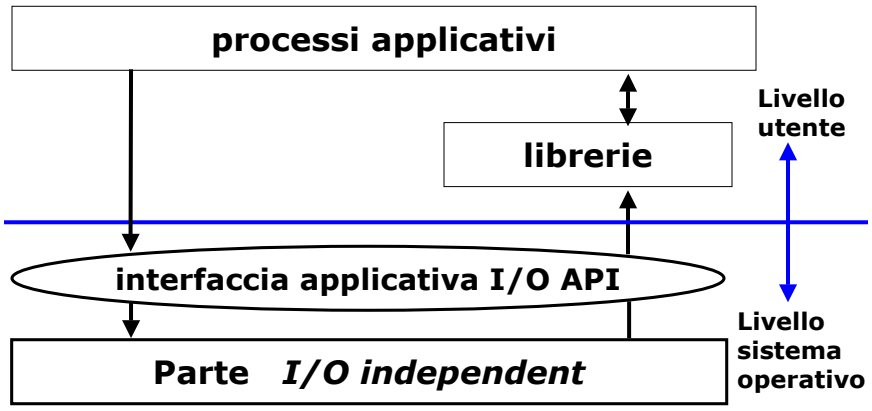
ARCHITETTURA DEL SOTTOSISTEMA DI I/O



ARCHITETTURA DEL SOTTOSISTEMA DI I/O



ARCHITETTURA DEL SOTTOSISTEMA DI I/O



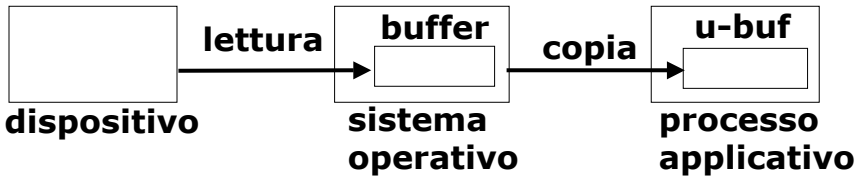
LIVELLO INDIPENDENTE DAI DISPOSITIVI

FUNZIONI

- **Naming**
- **Buffering**
- **Gestione malfunzionamenti**
- **Allocazione dei dispositivi ai processi applicativi**

BUFFERING

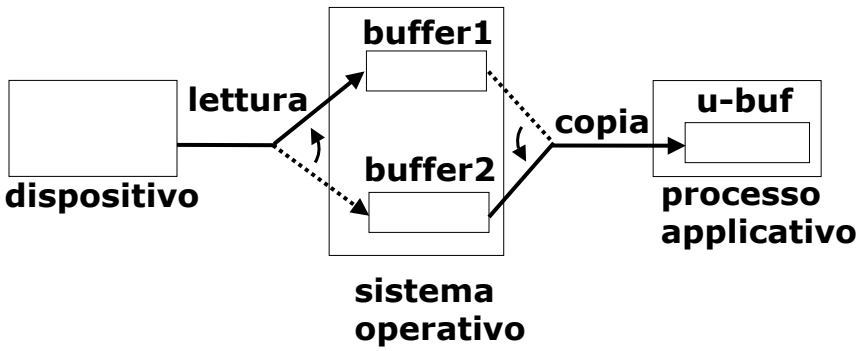
ES. operazione di lettura con singolo buffer



- **Buffer: area tampone nella memoria del sistema operativo**
- **u-buf: area tampone nella memoria vertuale del processo applicativo**

BUFFERING

ES. operazione di lettura con doppio buffer



GESTIONE MALFUNZIONAMENTI

- **Tipi di gestione degli eventi anomali:**
  - **Risoluzione del problema (mascheramento dell'evento anomalo);**
  - **Gestione parziale e propagazione a livello applicativo;**
- **Tipi di eventi anomali:**
  - **Eventi propagati dal livello inferiore (es. guasto HW permanente;**
  - **Eventi generati a questo livello (es. tentativo di accesso a un dispositivo inesistente).**

ALLOCAZIONE DEI DISPOSITIVI

- **Dispositivi condivisi da utilizzare in mutua esclusione;**
- **Dispositivi dedicati ad un solo processo (*server*) a cui i processi *client* possono inviare messaggi di richiesta di servizio;**
- **Tecniche di *spooling* (dispositivi virtuali).**

LIVELLO DIPENDENTE DAI DISPOSITIVI

Funzioni:

- **fornire i gestori dei dispositivi (*device drivers*)**
- **offrire al livello superiore l'insieme delle funzioni di accesso ai dispositivi (interfaccia "*device-independent*"), es:**

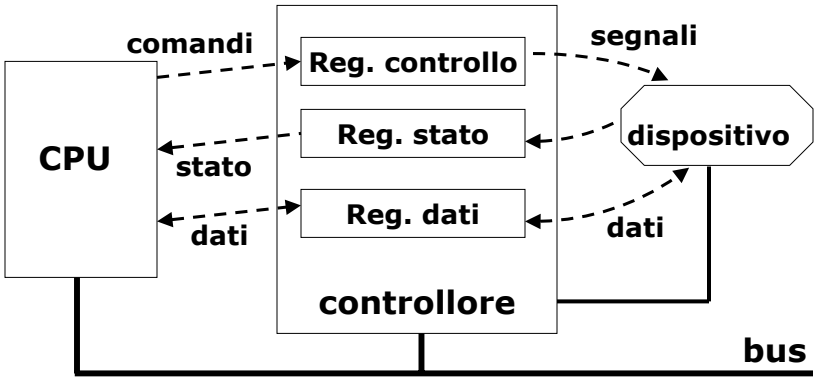
```
N= read (disp, buffer, nbytes)
```

nome unico del dispositivo

Buffer di sistema

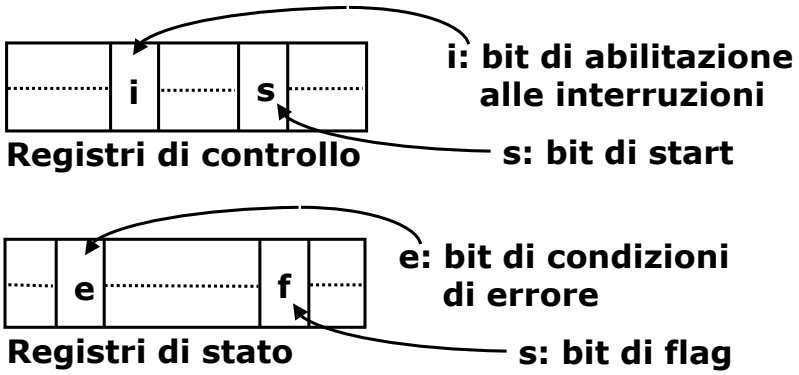
GESTORE DI UN DISPOSITIVO

Schema semplificato di un controllore

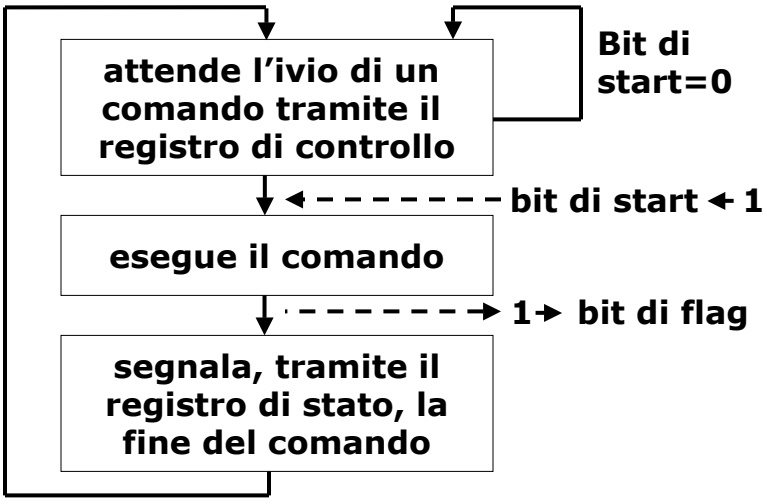


GESTORE DI UN DISPOSITIVO

Registri di stato e controllo



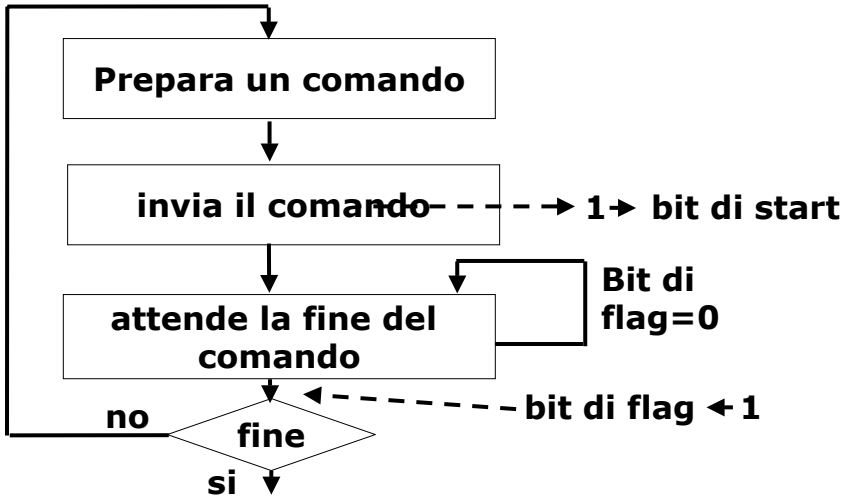
PROCESSO ESTERNO



PROCESSO ESTERNO

```
processo esterno
{
      while (true)
      {
            do{;} while (start ==0)//stand-by
            <esegue il comando>;
            <registra l'esito del comando
              ponendo flag = 1>;
      }
}
```

PROCESSO APPLICATIVO



PROCESSO APPLICATIVO

```
processo applicativo
{
      ……….
      for (int i=0; i++; i<n)
      {     <prepara il comando>;
            <invia il comando>;
            do{;} while (flag ==0)
             //ciclo di attesa attiva
            <verifica l'esito>;
      }
       ……….
}
```

GESTIONE A INTERRUZIONE

- **Lo schema precedente viene detto anche "*a controllo di programma*".**
- **Non adatto per sistemi multiprogrammati a causa dei clicli di attesa attiva.**
- **Per evitare l'attesa attiva:**
  - **Riservare, per ogni dispositivo un semaforo: `dato_disponibile`**
  - **`dato_disponibile` = 0;**
- **Attivare un dispositivo abilitandolo a interrompere (ponendo nel registro di controllo il bit di abilitazione a 1.**

GESTIONE A INTERRUZIONE

```
processo applicativo
{
     ……….
     for (int i=0; i++; i<n)
     {    <prepara il comando>;
          <invia il comando>;
          wait (dato_disponibile );
           <verifica l'esito>;
     }
      ……….
}
```

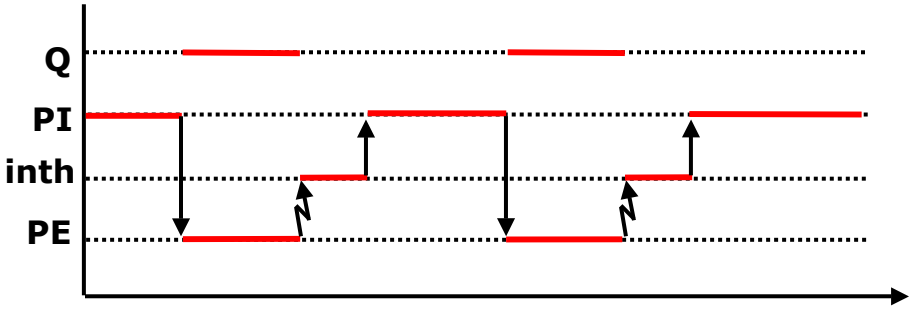
commutazione di contesto

FUNZIONE DI RISPOSTA ALLE INTERRUZIONI

```
Interrupt_handler
{
     ……….
     signal (dato_disponibile );
     ……….
}
```

riattiva il processo applicativo

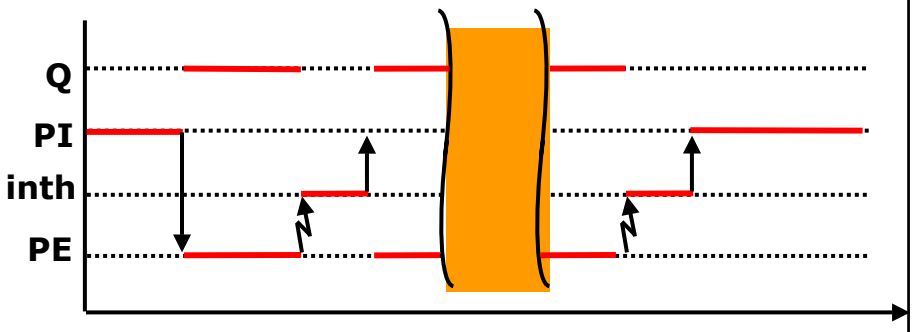
DIAGRAMMA TEMPORALE



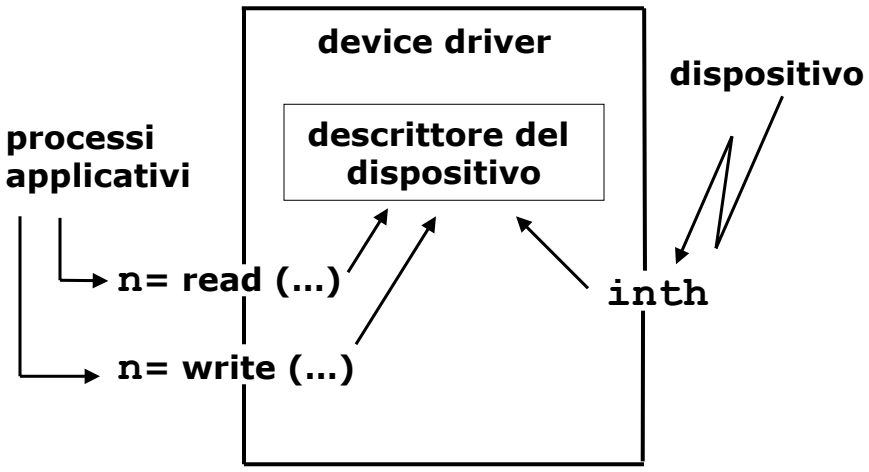
PI: processo applicativo che attiva il dispositivo
PE: processo esterno
Inth: routine di gestione interruzioni
Q: altro processo applicativo

DIAGRAMMA TEMPORALE

E` preferibile uno schema in cui il processo applicativo che ha attivato un dispositivo per trasferire n dati viene risvegliato solo alla fine dell'intero trasferimento:



ASTRAZIONE DI UN DISPOSITIVO



DESCRITTORE DI UN DISPOSITIVO

indirizzo registro di controllo
indirizzo registro di stato
indirizzo registro dati
semaforo Dato_disponibile
contatore dati da trasferire
puntatore al buffer in memoria
esito del trasferimento

DRIVER DI UN DISPOSITIVO

ESEMPIO:

```
int read(int disp,char *buf,int cont)
```

CON:

- **la funzione che restitiuisce -1 in caso di errore o il numero di caratteri letti se tutto va bene,**
- **`disp` è il nome unico del dispositivo,**
- **`buf` è l'indirizzo del buffer in memoria,**
- **`cont` il numero di dati da leggere**

DRIVER DI UN DISPOSITIVO

```
int read(int disp,char *buf,int cont)
{ descrittore[disp].contatore=cont;
  descrittore[disp].puntatore=buf;
  <attivazione dispositivo> ;
  wait(descrittore[disp].
              dato_disponibile);
  if (descrittore[disp].esito
              == <codice di errore>)
     return (-1);
  return (cont-descrittore[disp].
              contatore);
}
```

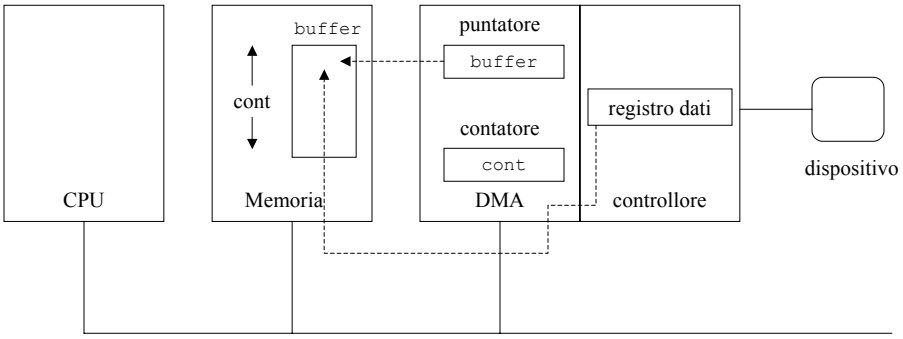
DRIVER DI UN DISPOSITIVO

```
void inth() //interrupt handler
 { char b;
   <legge il valore del registro di stato>;
   if (<bit di errore> == 0)
     {<ramo normale della funzione> }
   else
   {<ramo eccezionale della funzione> }
   return //ritorno da interruzione
 }
```

RAMO NORMALE DELLA FUNZIONE

```
{  < b = registro di stato >;
   *(descrittore[disp].puntatore)= b;
   descrittore[disp].puntatore ++;
   descrittore[disp].contatore --;
   if (descrittore[disp].contatore!=0)
      <riattivazione dispositivo>;
   else
      {descrittore[disp].esito =
         <codice di terminazione corretta>;
       <disattivazione dispositivo>;
       signal (descrittore[disp].
                        dato_disponibile);
      }
}
```

Gestione di un dispositivo in DMA



RAMO ECCEZIONALE DELLA FUNZIONE

```
{   < routine di gestione errore >;
    if (<errore non recuperabile>)
      {descrittore[disp].esito =
          <codice di terminazione anomala>;
      signal (descrittore[disp].
                          dato_disponibile);
       }
}
```