

Gestione dell'I/O

- Introduzione
- Scopi del software di I/O
- Organizzazione del software di I/O
 - Gestori delle interruzioni
 - Driver dei dispositivi
 - SW di I/O indipendente dai dispositivi
 - SW di I/O di livello utente

Gestione dell'I/O

1

Varietà dei dispositivi di I/O

- Modalità di trasferimento
 - A caratteri
 - A blocchi di caratteri
- Comunicazione
 - Seriale
 - parallela
- Temporizzazione
 - Sincrona
 - Asincrona

Gestione dell'I/O

2

Varietà dei dispositivi di I/O (cont.)

- Condivisione
 - Accesso dedicati
 - Accesso condiviso
- Operazioni
 - Lettura
 - Scrittura
 - Lettura/scrittura
- Velocità
 - Variabile da dispositivo a dispositivo

Gestione dell'I/O

3

Scopi del software di I/O

- **Indipendenza dal dispositivo**
 - Un programma deve funzionare indipendentemente dal particolare dispositivo su cui i dati sono memorizzati
- **Denominazione comune**
 - Il nome del dispositivo non deve dipendere dal dispositivo stesso
- **Trattamento degli errori**
 - Gli errori dovrebbero essere trattati il più vicino possibile all'hardware
 - Se il controllore non è in grado di correggere l'errore allora dovrebbe intervenire il driver del dispositivo (es. rileggendo il dato)
 - L'errore deve essere propagato ai livelli alti solo quando i livelli più bassi non sono in grado di gestirlo.

Gestione dell'I/O 4

Scopi del software di I/O (cont.)

- **Sincronizzazione**
 - La maggior parte delle operazioni di I/O avviene in modo asincrono (basato su interruzione)
 - I programmi utenti sono più facili da scrivere assumendo operazioni di I/O sincrone
 - Il sistema operativo fa apparire come sincrone operazioni di I/O che nella realtà avvengono in modo asincrono.
- **Bufferizzazione**
 - I dati che escono da un dispositivi non sempre possono essere portati immediatamente nella destinazione finale
 - Es. un pacchetto ricevuto dalla rete deve essere esaminato prima di essere consegnato al processo destinatario
 - La bufferizzazione richiede molte operazioni di copiatura

Gestione dell'I/O 5

Scopi del software di I/O (cont.)

- **Condivisione**
 - Il sistema operativo deve garantire l'accesso corretto ai vari tipi di dispositivo
 - L'accesso a dispositivi dedicati può dare luogo a problemi quali stallo

Gestione dell'I/O 6

Organizzazione del sistema di I/O



Gestione dell'I/O

7

Sw di livello utente

- Procedure di libreria
- Sistema di spool
 - Stampante
 - Demone di stampa
 - Directory di spool

Gestione dell'I/O

8

Sw indipendente dai dispositivi

- Naming
 - Associa fra nome simbolico del dispositivo e driver relativo
- Bufferizzazione
 - Scollega la velocità del dispositivo da quella della CPU
- Gestione errori
 - Gestisce errori gestibili a questo livello
 - Propaga al processo applicativo quelli non gestibili
- Allocazione e rilascio dispositivi dedicati
 - Fornisce gestori dei dispositivi
- Dimensione dei blocchi indipendente dal dispositivo

Gestione dell'I/O

9

Sw indipendente dai dispositivi

```
n=read(IN, ubuf, ubufsize) // chiamata del processo
```



```
int read(device dp, char* punt, int cont) { // sw ind. disp.  
    int n, D;  
    char buffer[N]; // buffer di sistema  
    <controllo accessi>;  
    n=_read(D, buffer, cont); // funzione del driver  
    <trasferimento dati da buffer[] a ubuf[]>;  
    return n;  
}
```

Gestione dell'IO

10

Diversi dispositivi

- Ogni controllore di dispositivo contiene registri di comando e di stato
- Il numero di registri e i comandi che si possono dare al controllore variano notevolmente a seconda del dispositivo
 - Mouse
 - Disco
 - ...
- Il sistema operativo prevede due categorie di dispositivi
 - A caratteri (tastiere, stampanti, ...)
 - A blocchi di caratteri (dischi, ...)

Gestione dell'IO

11

Interfaccia device-independent

- Il SO definisce due interfacce standard a cui tutti i dispositivi si devono attenere
 - Lettura di un carattere (blocco di caratteri)
 - Scrittura di un carattere (blocco di caratteri)
- Driver di dispositivo
 - Realizza interfaccia device-independent
 - Adatta le caratteristiche del dispositivo all'interfaccia standard
 - Contiene codice specifico per il dispositivo
 - Scritto dal produttore del dispositivo e consegnato insieme
 - Specifico per un certo sistema operativo

Gestione dell'IO

12

Driver di dispositivo: funzioni

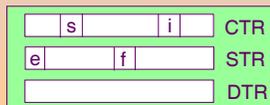
- Accetta richieste di lettura (scrittura) dal sw indipendente dal dispositivo
- Traduce le richieste in sequenza di comandi comprensibili per il dispositivo (controllore del dispositivo)
- Scrive tali comandi nei registri del controllore
- Si blocca in attesa che il controllore faccia il lavoro
-
- Viene sbloccato da una interruzione
- Esegue un controllo per rilevare eventuali errori
- Se tutto è andato bene passa i dati letti al sw indipendente dal dispositivo

Gestione dell'I/O

13

Modello di controllore

- Registro di controllo (CTR)
 - Abilitazione dispositivo (S)
 - Abilitazione a generare interruzioni (I)
- Registro di stato (STR)
 - Flag di errore (e)
 - Flag di fine operazione
- Registro dati (DTR)



Gestione dell'I/O

14

Modello di dispositivo

- Attende la ricezione di comando dalla CPU
 - Tramite registro di controllo (s=1)
- Esegue comando
- Segnala la fine operazione
 - Tramite registro di stato (f=1)
 - E contestualmente l'esito dell'operazione

Gestione dell'I/O

15

Processo esterno

- Il comportamento del controllore può essere immaginato come un **processo esterno**
- Il processo esterno è un processo hw

```
Processo esterno: {  
  while(1) {  
    <attende comando da CPU>  
    <esegue comando>  
    <imposta registro stato> // inclusi flag e, f  
  }  
}
```

Gestione dell'I/O

16

Processo interno

- Esempio: Lettura a interruzione di programma

```
semaforo dato_pronto=0;  
...  
for(int i=0; i<n; i++) {  
  <prepara comando in registro R>  
  <invia comando> // s=1  
  dato_pronto.wait(); // attesa dato disponibile  
  <verifica esito operazione>  
}  
....
```

Gestione dell'I/O

17

Chi sveglia il processo interno?

- Il processo deve sbloccarsi quando il dato diventa pronto
- Il processo esterno non ha accesso al semaforo
- Il processo viene sbloccato dalla funzione di gestione dell'interruzione
- Questa costituisce il prolungamento sw del processo esterno (hw)

Gestione dell'I/O

18

Descrittore di dispositivo

- Rappresenta una astrazione del controllore
- Racchiude al suo interno i dettagli del controllore
- Permette la comunicazione fra processo applicativo (interno) e controllore (processo esterno)
 - Numero di dati da trasferire
 - Indirizzo del buffer dove metterli
 - Informazioni sull'esito dell'operazione
 - Contiene semaforo di sincronizzazione

Descrittore + Funzioni di accesso = Driver di dispositivo

Gestione dell'I/O

19

Descrittore di dispositivo

- Indirizzo registro di controllo
- Indirizzo registro di stato
- Indirizzo registro dati
- Semaforo di sincronizzazione (dato_pronto)
- Numero di dati da trasferire (contatore)
- Indirizzo de buffer (puntatore)
- Risultato dell'operazione (esito)
- ...

Gestione dell'I/O

20

Driver di dispositivo

- Esempio: lettura (si leggono un certo numero di bytes)

```
int _read (int disp, char *pbuf, int cont)
```

disp: dispositivo su cui operare

pbuf: puntatore al buffer di sistema dove mettere i dati

cont: numero di bytes da leggere

Gestione dell'I/O

21

Driver di dispositivo

```
int _read (int disp, char *pbuf, int cont) {
    descrittore[disp].contatore=cont;
    descrittore[disp].puntatore=pbuf;
    <attivazione del controllore> // bit di start s=1
    descrittore[disp].dato_pronto.wait(); //sospensione
    if(descrittore[disp].esito==<codice errore>)
        return(-1);
    return(cont-descrittore[disp].contatore);
}
```

Gestione dell'IO

22

Gestione dell'interruzione

- Appena un dato diventa pronto → interruzione
- Viene eseguita una funzione di gestione inth()
 - Legge registro di stato per verificare la causa dell'interruzione
 - Se errore cerca di gestirlo
 - Legge il dato dal controllore
 - Mette il dato letto nel buffer di sistema (puntatore)
 - Incrementa puntatore e decrementa contatore
 - Se trasferimento non completato riattiva il controllore
 - Altrimenti riattiva processo (interno) sospeso

Gestione dell'IO

23

Gestione dell'interruzione

```
void inth() {
    char b;
    <legge registro di stato STR>;
    if(<presenza di errori>) {
        <funzione di gestione dell'errore>;
        if(<errore non recuperabile>) {
            descrittore[disp].esito= <codice errore>;
            descrittore[disp].dato_pronto.signal(); // riattivazione proc.
        }
    }
    else { // assenza di errori
```

Gestione dell'IO

24

Gestione dell'interruzione

```
else {                                     // assenza di errori
    <b ← registro dati DTR>
    *descrittore[disp].puntatore=b;
    descrittore[disp].puntatore++;
    descrittore[disp].contatore--;
    if(descrittore[disp].contatore!=0)
        <riattivazione controllore>;
    else {
        descrittore[disp].esito=<fine corretta>;
        <disattivazione controllore>;
        descrittore[disp].dato_pronto.signal();
    }
}
```

Gestione dell'I/O

25
