SISTEMI OPERATIVI

09.a

Gestione dell'Input/Output

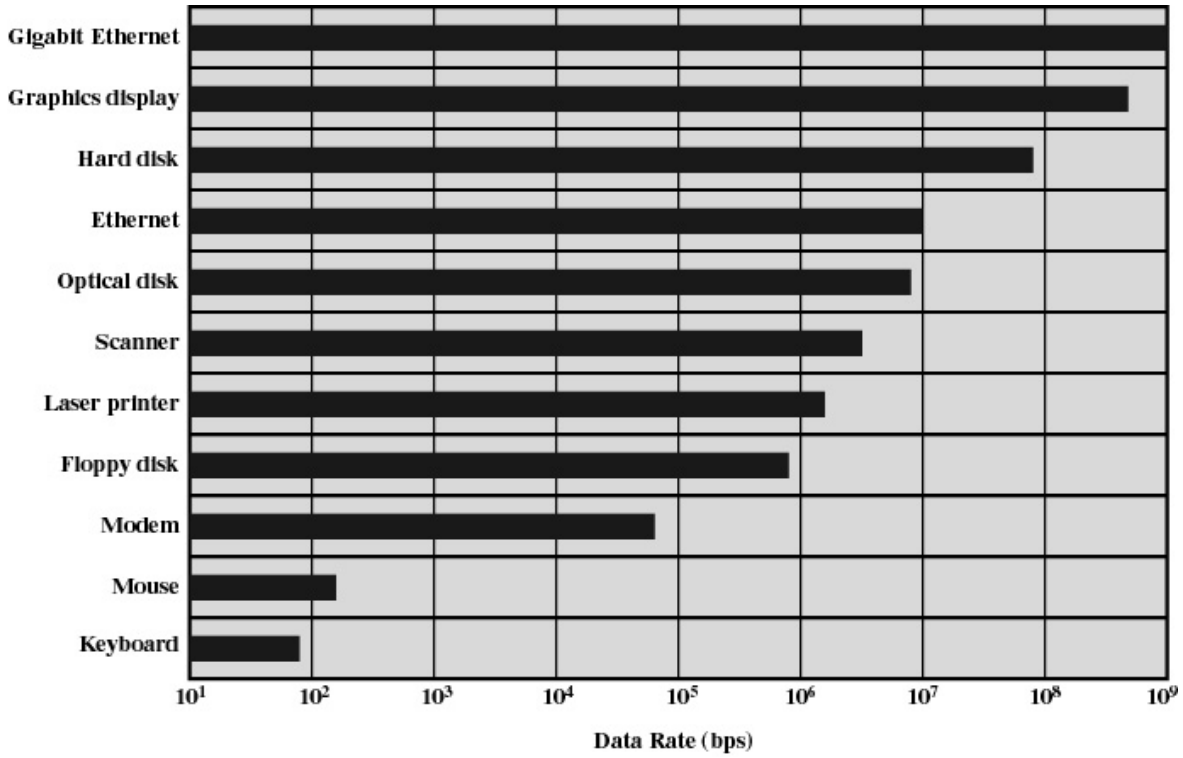
Gestione dell'I/O

- Obiettivi
- Procedure
- Device handler
- Buffering
- Spooling

Dispositivi di I/O

- Un sistema di elaborazione ha periferiche per diversi impieghi:
  - per comunicare con l'uomo (tastiera, schermo, stampante, ...)
  - per memoria di supporto (dischi, nastri, ...)
  - per comunicare con altri calcolatori (modem, reti, ...)
- Le velocità di trasferimento possono essere diversissime:
  - tastiera: 80 b/s
  - CD-ROM 6 Mb/s
  - video display 1Gb/s
- Altre diversità:
  - complessità del controllo (tastiera più semplice del disco)
  - tipo di dati trasferiti (caratteri, blocchi, ...)
  - rappresentazione dei dati (codifica, parità, ...)
  - necessità di file system
  - ...

Diverse velocità



Tecniche di I/O

- I/O programmato:
  - il processore attende in busy waiting che il dato sia stato trasferito
- I/O gestito ad interrupt:
  - il processore comanda una operazione di I/O e poi prosegue ad eseguire un altro processo o thread
  - il dispositivo interrompe il processore quando il dato è stato trasferito
- Direct Memory Access (DMA):
  - il processore comanda il trasferimento di un blocco di dati
  - il dispositivo trasferisce i dati (cycle stealing)
  - il dispositivo interrompe il processore quando tutti i dati sono stati trasferiti

Obiettivi di gestione

- Nel gestire le operazioni di I/O il SO intende garantire:
  - efficienza:
    - le operazioni di I/O sono lente (perché lo sono i dispositivi)
    - il processore non deve perdere tempo per esse
  - indipendenza dal particolare dispositivo:
    - per semplificare la vita al programmatore
    - per poter usare lo stesso programma con (modelli di) dispositivi diversi

Efficienza nella gestione

- I dispositivi di I/O sono molto più lenti della CPU
- La gestione centralizzata di tutto l'I/O da parte del SO (ad interrupt o via DMA) è il presupposto per il multitasking (che consente di usare in modo efficiente la CPU, mandando in esecuzione un altro processo quando un altro richiede una operazione di I/O)
- L'I/O può diventare un collo di bottiglia (ready list vuota)
- Per aumentare il numero di processi nella ready list, spesso vengono effettuati degli swap (in e out), che sono anch'essi operazioni di I/O
- L'efficienza negli accessi al disco è di grande importanza e ad essa viene dedicata particolare attenzione.

Indipendenza dal dispositivo

- Si vuole un trattamento uniforme dei dispositivi in riferimento al
  - modo in cui i processi vedono i dispositivi
  - modo in cui i dispositivi sono gestiti dal SO
- Non è facile ottenere questa uniformità, a causa della diversa natura e delle diverse modalità di funzionamento dei dispositivi
- Si cerca di usare un approccio modulare in modo da:
  - nascondere nelle routine di basso livello gran parte dei dettagli dei dispositivi
  - distinguere i dispositivi fisici, cui sono associate le routine di basso livello, dai dispositivi logici (canali), cui sono associate funzionalità generali di alto livello (open, read, write, close)
  - associare un dispositivo fisico a uno logico
  - consentire ai processi di interagire con i dispositivi logici

Driver di I/O

- Il software del SO che realizza ciò si chiama driver. Un driver controlla un tipo di dispositivo, ne nasconde i dettagli e consente le operazioni di I/O ai processi tramite comandi di alto livello
- Un driver è solitamente costituito da due parti, che operano in modo asincrono, sincronizzate tramite semafori:
  - una parte, eseguita dal processo utente, nei fatti una subroutine di interfaccia (API), che il processo invoca per richiedere una operazione di I/O (OPEN, CLOSE, READ, ...);
    - le subroutine di interfaccia attivano una routine (DOIO) del SO specificandone i parametri (eventualmente tramite SVC)
  - una seconda parte, eseguita dal SO con l'ausilio di:
    - una routine di servizio delle interruzioni (ISR),
    - un processo di sistema (Device Handler) che, per rendere veloce l'esecuzione della ISR, la affianca e svolge le operazioni non strettamente urgenti.

Richieste di I/O dei processi

- Il processo che intende usare un dispositivo (disp) deve:
  - aprire un canale (*ch*) associato al dispositivo, chiamando una subroutine di interfaccia, ad es. OPEN (*disp*, *ch*, ...)
    - questa subroutine porta il SO ad associare al dispositivo fisico *disp* il dispositivo logico *ch* ed a consentirne l'uso al processo
  - richiedere le operazioni di I/O, chiamando una subroutine di interfaccia, ad es. WRITE (*ch*, *dati*, ...)
    - questa subroutine porta il SO ad eseguire o avviare la corrispondente operazione;
  - chiudere il canale *ch*, al termine dell'uso del dispositivo, tramite una subroutine di interfaccia ad es. CLOSE(*ch*, ...)
    - il SO elimina le associazioni fatte dalla OPEN.
- Le subroutine di interfaccia contengono eventualmente una SVC: questa attiva una routine del SO (DOIO) che esegue quanto richiesto e può sospendere il processo richiedente.

Attivazione della DOIO

- La DOIO, attivata direttamente (o indirettamente tramite SVC) dalla subroutine di interfaccia, ha il compito di predisporre le strutture di dati che servono per eseguire l'operazione richiesta, sulla base dei parametri ricevuti;

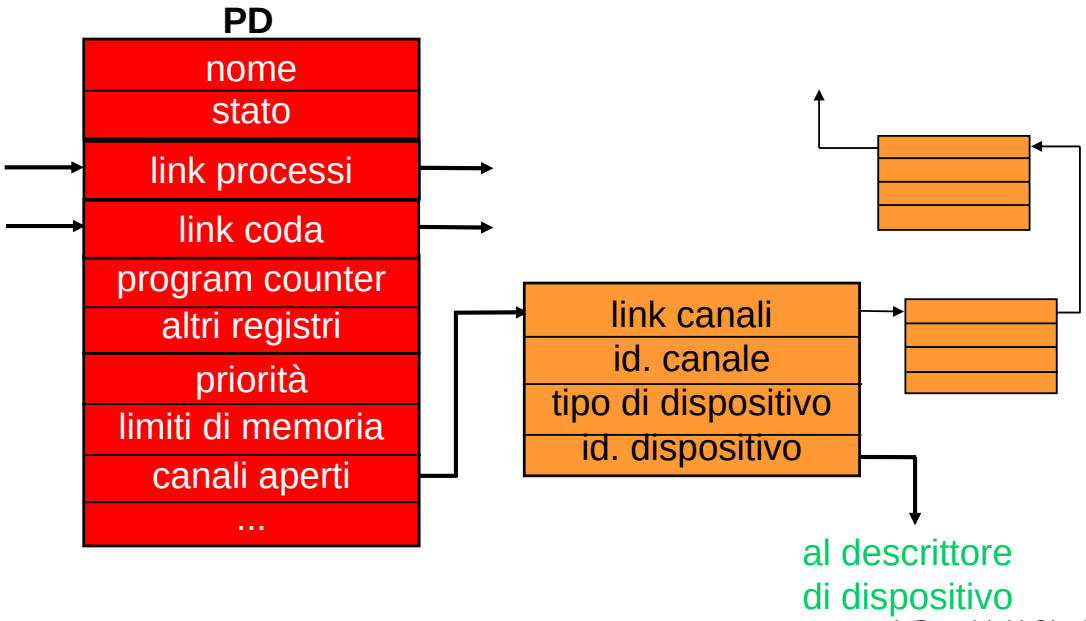
- Parametri della DOIO:
  - DOIO (*ch*, *mode*, *count*, *pointer*, *sem*);
    - *ch* = canale (identificatore di dispositivo logico)
    - *mode* = tipo di operazione (open, read, write, …)
    - *count* = numero di dati da trasferire
    - *pointer* = indirizzo di memoria di origine o di destinazione del trasferimento
    - *sem* = semaforo di supporto (RS - Richiesta Servita) per segnalare quando l'operazione di I/O sarà completata

Strutture dati usate dal driver

- Per eseguire le operazioni di I/O richieste da un processo, il driver deve disporre di alcune di informazioni, tra le quali:
  1. il processo richiedente
  2. il dispositivo logico (canale)
  3. il dispositivo fisico
  4. l'operazione da eseguire
  5. i semafori per le sincronizzazioni
- Le strutture di dati che contengono queste informazioni sono:
  - il descrittore di processo (PD)
  - il descrittore di dispositivo (DD)
  - il descrittore della richiesta di I/O (IORB)
- Segue una possibile organizzazione di queste strutture di dati e le informazioni in esse contenute (condivise dalle due attività asincrone del driver).

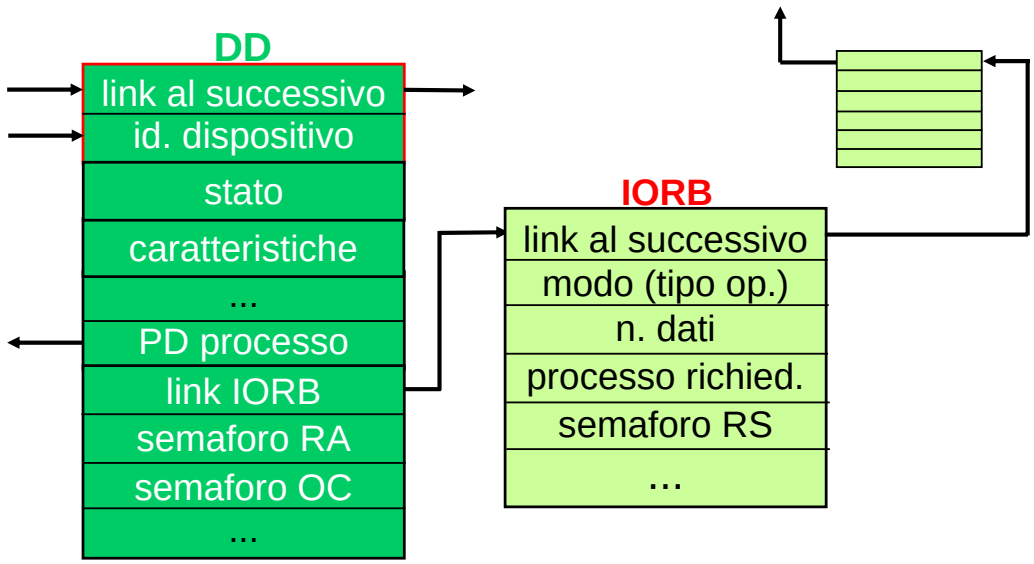
Descrittore di processo - PD

- OPEN (*disp*, *ch*, ...): il SO crea l'associazione tra canale e dispositivo e la lega al descrittore del processo, il processo opera utilizzando *ch*:



Descrittore di dispositivo e IORB

- Il descrittore di dispositivo (DD device descriptor) contiene informazioni che definiscono le caratteristiche specifiche del dispositivo
- DOIO (*ch*, *mode*, ...): completa un IORB e lo aggiunge alla lista:



Funzioni della DOIO

- DOIO (*ch*, *mode*, *count*, *pointer*, *sem*);
  - la routine accede al descrittore PD del processo in esecuzione e, tramite il link canali aperti, trova il dispositivo associato a *ch* e il relativo descrittore DD
  - verifica quindi la validità dei parametri passati, confrontandoli con le caratteristiche contenute nel DD
    - se i parametri non sono validi, termina con un'eccezione d'errore (che può provocare la terminazione del processo)
    - altrimenti, usa i parametri per costruire l'IORB (I/O Request Block) che descrive la richiesta del processo e lo inserisce nella lista degli IORB (contenuto nel DD)
  - esegue un signal sul semaforo RA (*Richiesta Attiva*, contenuto nel DD) per indicare che c'è un nuovo IORB nella lista del dispositivo: questo signal sincronizza il Device Handler del dispositivo (2a parte del driver)
  - esegue un wait sul semaforo RS (*Richiesta Servita*, contenuto nell'IORB), che provoca la sospensione del processo fino a che quella richiesta non è stata servita

Struttura della DOIO

DOIO (*ch, mode, count, pointer, RS*) {

. . . /* verifica la compatibilità dei parametri con le caratteristiche del dispositivo (DCB); */

. . . /* se i parametri non sono validi, abortisce l'operazione segnalando l'errore; */

. . . /* costruisci un IORB con i parametri ricevuti; inserisci l'IORB nella coda di richieste al dispositivo (link IORB nel DCB); */

signal (RA); // segnala il nuovo IORB al Device Handler

wait (RS); // il processo richiedente si sospende …;

return // l'operazione è terminata

Struttura del Device Handler


```
while (true) {
    wait (RA)     // attende un signal dalla DOIO (IORB presente)
    . . .         /* preleva un IORB dalla lista associata al
                     dispositivo (DD) e avvia l'operazione richiesta */
    wait (OC)     // attende, sul semaforo OC Op Completa (nel DD),
                  // che l'operazione sia completata (signal dalla ISR)
    . . .         /* verifica eventuali errori,
                     esegue le altre operazioni non eseguite dalla ISR */
    signal (RS)   // il processo sospeso ritorna ready
    . . .         /* restituisce l'IORB alla lista degli IORB liberi */
}
```


Struttura della ISR

- La struttura della routine di servizio delle interruzioni del dispositivo (supposto di input) è molto semplice:

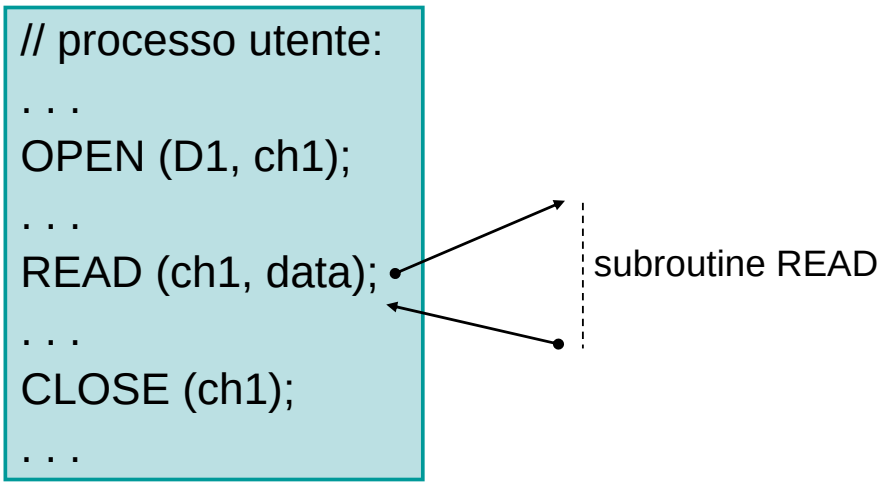
```
ISR:
    . . .          // gestione del dispositivo – elimina l'interruzione
inp (pointer)      // effettua l'input, copia il dato nella posizione voluta
pointer ++         // incrementa il puntatore
count  --          // decrementa il contatore
If count =0 then signal (OC)          // simula un DMA
                   // avvisa il DH che l'operazione di I/O è completata
terminate          // restituisce il controllo allo scheduler
                   /* (che deciderà se riprendere il processo interrotto
                      o mandare in esecuzione il Device Handler) */
```

Struttura del driver

- La parte del driver eseguita dal processo utente (sincrona con il processo utente) è costituita da:
  - le subroutine di interfaccia
  - la routine DOIO attivata dalle subroutine di interfaccia
- La parte del driver eseguita dal processo di I/O del SO (sincrona con le operazione del dispositivo) è costituita da:
  - il Device Handler (processo di sistema che gestisce l'I/O del particolare dispositivo),
  - la routine di servizio delle interruzioni (ISR) del dispositivo.
- Le due parti del driver (tra loro asincrone) si sincronizzano tramite i semafori RA ed RS e comunicano tramite le strutture di dati condivise (in particolare la lista concatenata di IORB).

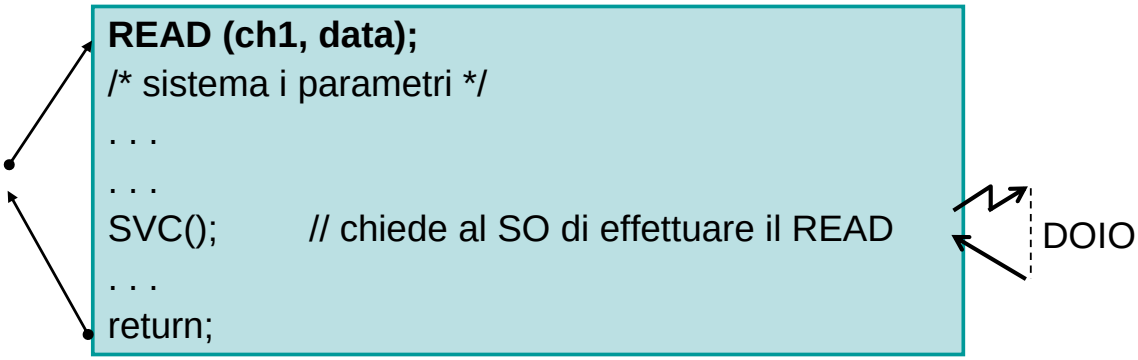
Processo utente

- OPEN, READ, CLOSE sono le subroutine di interfaccia (API) del driver del dispositivo D1



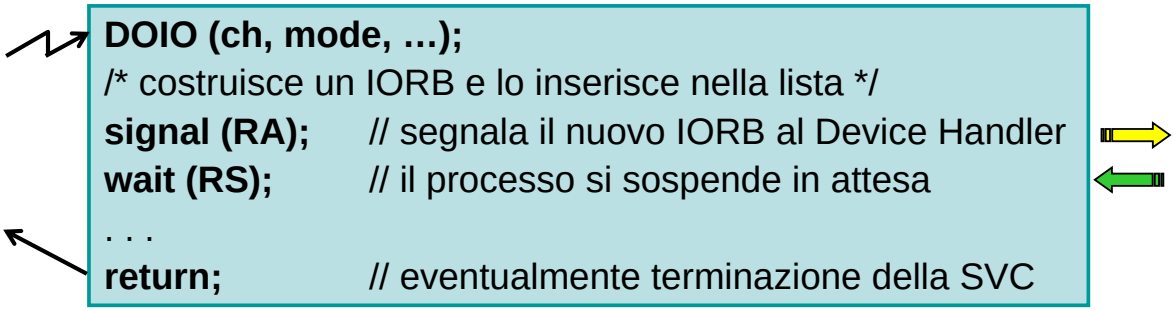
Subroutine di interfaccia

- SVC() è una interruzione software di chiamata a sistema e provoca:
  - l'attivazione della routine di servizio DOIO



Routine DOIO

- La routine DOIO si sincronizza con il Device Handler (processo di sistema) tramite i semafori:
  - RA (richiesta attiva)
  - RS (richiesta servita)



Device Handler

- Il Device Handler (processo di sistema) effettua le operazioni più complesse (transcodifica, controlli etc.) specifici per il dispositivo, costituisce la parte più importante del driver :

```
while (true) {
        . . .
        wait (RA);      // attende un signal dalla DOIO
        . . .
        wait (OC);      // attende un signal dalla ISR
        . . .
        signal (RS);    // segnala il completamento alla DOIO
}
```

- Il DH e la ISR si sincronizzano tramite il semaforo *OC*

Gestione dell'interruzione - ISR

- La sincronizzazione della routine di servizio delle interruzioni del dispositivo è riportata in figura:

```
ISR:
    . . .

If count=0 then signal (OC);
                    // avvisa il DH che l'operazione è
  completa
terminate           /* restituisce il controllo allo scheduler */
```

Driver più snello

- Per i dispositivi più semplici, per i quali le operazioni da eseguire in occasione di una interruzione sono leggere, si può rendere più snella la struttura del driver eliminando il Device Handler e facendo svolgere le sue funzioni in parte alla DOIO, in parte alla ISR.
- In questo caso la DOIO si sincronizza direttamente con la ISR (basta il solo semaforo RS):
  - la DOIO, costruisce l'IORB, lo mette a disposizione della ISR, avvia anche l'operazione; poi attende: wait(RS)
  - la ISR verifica eventuali errori, esegue tutte le operazioni richieste e poi segnala: signal(RS)

Bufferizzazione

- Per aumentare l'efficienza del sistema di I/O il SO prevede alcune aree di memoria (buffer) in cui trasferire i dati di input provenienti dai dispositivi (prima di trasferirli ai processi utente) e i dati di output (prima di trasferirli ai dispositivi):
  - si riduce l'overhead dovuto alle sincronizzazioni
  - disponendo di dispositivi DMA le operazioni di I/O sono più veloci
  - con buffer multipli i dispositivi operano con maggior efficienza, il dispositivo può trasferire un blocco di dati mentre il processo ne può elaborare un altro in parallelo
  - il processo può subire uno swap out, perché il trasferimento coinvolge un frame del sistema: lo swap out di un processo mentre è in corso un trasferimento in una sua pagina creerebbe problemi seri,
  - il SO deve tener traccia dei buffer assegnati ai processi

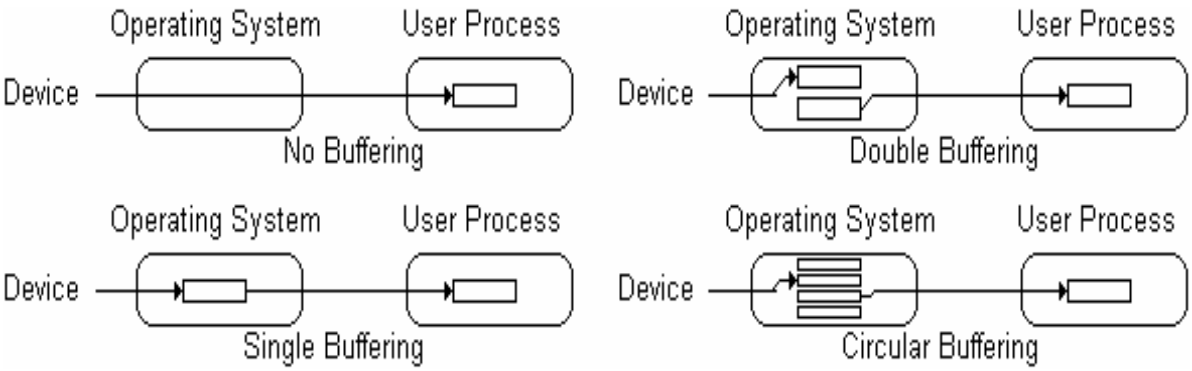
Tipi di bufferizzazione

- A seconda del dispositivo, la bufferizzazione può essere:
  - di tipo a blocchi
    - i buffer sono di dimensione fissa, ad es. di un frame
    - i trasferimenti avvengono per blocchi
    - usato per dischi, nastri, …
  - di tipo stream
    - i trasferimenti avvengono per flussi di caratteri
    - usato per terminali, stampanti, porte seriali, mouse
    - …

Numero di buffer

- Per la gestione bufferizzata dell'I/O si può pensare di usare:
  - un buffer singolo
    - il processo può elaborare un blocco di dati mentre l'altro viene trasferito
  - un doppio buffer
    - utilizza due buffer di sistema, invece di uno solo
    - il processo può trasferire dati da un blocco mentre il SO sta svuotando o riempiendo l'altro
    - diminuiscono i tempi di attesa del processo
  - una bufferizzazione circolare o un buffer multiplo (pool)
    - utilizza più di due buffer di sistema
    - ciascun buffer è un elemento di una coda circolare o di una lista
    - consente di far fronte a picchi (temporanei) di operazioni di I/O effettuate da un processo

Schema delle bufferizzazioni



Bufferizzazione per input a caratteri

- Talvolta la ISR gestisce un dispositivo di input a caratteri (byte o interi) il cui rate di produzione è irregolare (ad esempio tastiera):
  - logicamente di tipo stream
  - il buffer può essere organizzato come una coda circolare di caratteri
  - se la ISR riempie il buffer, vengono persi caratteri (gli ultimi oppure i più vecchi)
    - la ISR NON può sospendersi, quindi non può attendere lo svuotamento del buffer da parte del processo utente o del DH

Spooling

- Un processo che richieda l'uso di un dispositivo non condivisibile deve attendere che si liberi, nel caso esso sia già in uso
- Per i dispositivi di I/O (principalmente di output), per evitare queste attese, il SO utilizza la tecnica di spooling (Simultaneous Peripheral Operation On Line), che consiste nell'assegnare, ad ogni processo che lo richieda, un dispositivo virtuale, realizzato da un file su disco:
  - ciascun processo ottiene, senza attesa, l'accesso al file che rappresenta il dispositivo (non condivisibile)
  - le richieste dei processi di trasferire dati al dispositivo vengono trasformate in trasferimenti sul file assegnato
  - essendo i dischi veloci, l'I/O è spesso più rapido
  - quando un processo chiude il proprio dispositivo virtuale, il suo file viene inserito nella coda dei file che saranno trasferiti al dispositivo fisico, uno alla volta, da un processo di sistema (Spooler)
  - la coda dei file si mantiene da un'attivazione all'altra del sistema

Fine

Gestione dell'Input/Output