LEZIONE 13

Sistemi di I/O

Architettura e dispositivi di I/O

* Grande numero di tipi di dispositivi
* Concetti comuni
* **Porta**
* **Bus** (canali di comunicazione, possono essere collegamenti a margherita o accesso diretto condiviso)
* **Controller:** dispositivi che collegano il bus di sistema con il sottosistema specifico del sistema di I/O. Questi controller sono necessari perché realizzano una serie di funzionalità che permettono proprio l’uso della particolare periferica (particolare tipo di I/O).
* Per ogni dispositivo bisogna anche parlare di **driver** i quali sono dei programmi veri e propri che realizzano le funzionalità di accesso al dispositivo (quindi sia di lettura, scrittura su dispositivo), e sono spesso visti come **black box**, perché spesso il costruttore di un nuovo dispositivo per evitare che il SO per interfacciarsi con esso debba modificarsi, è stesso il costruttore a proporre metodi di accesso alle periferiche.

Quindi il controller fondamentalmente ha la possibilità di connettersi alle periferica grazie al driver che viene realizzato dal produttore del particolare dispositivo, e il bus di comunicazione permette un accesso tramite il controller al dispositivo (quindi da un punto di vista fisico).

* dispositivi di controllo delle istruzioni di I/O
* I dispositivi hanno indirizzi usati da:
* Speciali istruzioni di I/O
* I/O associato alla memoria

Immagine che contiene screenshot

Descrizione generata automaticamenteTipica struttura del bus in un PC

**Spiegazione**

Ci sta il bus PCI che è quello principale, poi si potrebbe avere come in questo caso, quello SCSI (small-computer-system-interface), si tratta di un insieme di dispositivi come ad esempio hard disk SCSI che sono particolarmente efficaci ed efficienti per le loro prestazioni, ed hanno bisogno proprio di un canale specializzato (bus SCSI).

Al bus PCI sono agganciati tramite dei controller, la CPU, video, hard disk. Attaccata alla CPU ci sta la cache, proprio perché come ben si sa, la CPU è il dispositivo più veloce sull’intero sistema, quindi spesso e volentieri si ha bisogno di depositare dei dati su una cache, in modo tale che anche questa cache funga da interfaccia verso la CPU, verso gli altri sottosistemi network e calcolo.

* In realtà tutti questi canali (bus PCI, bus SCSI, bus di espansione), hanno il compito sia di fare un routing dei vari protocolli che esistono su ogni canale, in quanto **ogni bus ha una vera e propria regola di comunicazione, e i controller servono per interfacciare questi dispositivi (bus) la quale hanno come detto prima diverse regole di funzionamento, ma anche per regolamentare le diverse velocità con i quali tali canali riescono a lavorare.** Spesso e volentieri queste velocità vengono ad essere integrabili, poiché sui vari controller ci sono anche dei buffer di memoria sui quali vanno a depositarsi i dati che sono in transito da e verso dispositivi, oppure da e verso altri canali di comunicazione, che vengono mantenuti in questi buffer per poi essere consumati dai dispositivi in funzione delle disponibilità.

Indirizzi delle porte dei dispositivi di I/O nei PC (elenco parziale)

Immagine che contiene edificio

Descrizione generata automaticamente**spiegazione**

Tutti i dispositivi del PC in genere colloquiano mediante dei programmi. A seconda della classe a cui appartengono questi dispositivi, hanno a disposizione indirizzi specifici di I/O di riferimento

Interrogazioni dei sistemi di I/O

Quando si parla di sistemi di I/O, si deve stabilire quali sono i metodi principali di interrogazione di essi.

Ci sono due metodi principali:

1. **Interrogazione ciclica (Polling)**

Il polling è un tipo di interrogazione mediante la quale, grazie ad una turnazione di interrogazione ai singoli dispositivi, permette di conoscere se ognuno di essi ha bisogno di compiere una determinata operazione di I/O.

* Le funzionalità principali di questo tipo di colloquio sono:
* **Command-ready**
* **Busy** (si legge bisi)
* **Error**
* L’interrogazione ciclica è in sé un’operazione efficiente; tale tecnica diviene però **inefficiente se le ripetute interrogazioni trovano raramente un dispositivo pronto** per il servizio mentre altre utili elaborazioni attendono la CPU. **Cioè questo tipo di interrogazione è buona con alto traffico.**

1. **Interruzioni (interrupt)**

* I segnali di interruzione sono usati diffusamente dai SO moderni **per gestire eventi asincroni** e per eseguire nel modo supervisore le procedure del nucleo.
* I controller dei dispositivi, gli errori e le chiamate del sistema generano segnali d’interruzione al fine d’innescare l’esecuzione di procedure del nucleo. Cioè funziona sulla richiesta espressa che un dispositivo fa per poter richiedere un operazione. In realtà l’interrupt viaggia proprio su una linea fisica (hardware) d’interruzione. Quando questa linea assume un determinato valore/tensione, allora il SO presta attenzione e va a verificare qual è il dispositivo che deve effettuare l’input/output.
* Poiché le interruzioni sono usate in modo massiccio per affrontare situazioni in cui il tempo è un fattore critico, è necessario avere un’efficiente gestione delle interruzioni per ottenere buone prestazioni del sistema. **Questo tipo di interrogazione è buono con basso traffico.**

**Immagine che contiene testo, segnale

Descrizione generata automaticamenteEsempio ciclo I/O basato sulle interruzioni**

**Spiegazione**

Tutto il processo si innesca perché per esempio un programma fa richiesta di accedere ad un I/O, quindi il driver del dispositivo avvia le operazioni di input/output (punto 1), attivando il controller (punto 2) il quale avvia le operazioni I/O. il controller a questo punto (punto 3) esegue fisicamente l’accesso al dispositivo, quindi esegue il trasferimento di questi dati e poi ritorna il controllo alla CPU (punto 4).

Arrivati al punto 4, la CPU rileva che i dati sono stati trasferiti, dopo di che tali dati vengono elaborati (punto 5) e la CPU può riprendere l’elaborazione (punto 7) tornando così al primo blocco.

**Ricorda**

Nell’immagine la CPU può effettuare 2 percorsi. Dal punto 1 passa a 2 poi a 3, 4,5,6,7; oppure dal punto 1 direttamente al punto 4 (quadrato sotto il primo); ma ciò dipende dalle regole scelte per operare sui dispositivi I/O.

Vettore di interruzioni della CPU intel pentium

Immagine che contiene screenshot

Descrizione generata automaticamente**spiegazione**

poiché le interruzioni sono gestite da un vettore, anche in questo caso ci sono dei particolari indici dedicati a particolari tipi di interruzioni. Ovviamente cambiano a seconda della CPU presa in considerazione. Ad esempio in questa tabella riguardano intel pentium.

Accesso diretto alla memoria (DMA)

* Con accesso diretto alla memoria si intende il trasferimento dei dati nella memoria centrale senza l’ausilio della CPU.
* Ciò è molto vantaggioso perché rende tutto molto più veloce, inoltre la CPU in questo modo può dedicarsi a qualcos’altro.
* Usato per evitare l’I/O programmato per **trasferimenti di grandi quantità di dati.**
* Richiede un controller DMA. Tale controller agisce direttamente sul bus della memoria, presentando al bus gli indirizzi di memoria necessari per eseguire il trasferimento **senza l’aiuto della CPU.**

Passi di un trasferimento DMA

Immagine che contiene screenshot

Descrizione generata automaticamente

**Spiegazione**

**Punto 1:** il drive del dispositivo chiede di trasferire dei dati per una certa area di memoria (guarda dove sta quadrato con scritto memoria).

**Punto 2**: il drive chiede al controller del disco di trasferire C byte che verranno trasferiti da vari bus di comunicazione, fino a passare dall’unità di disco alla RAM (la memoria assegnata parte dall’indirizzo X).

**Punto 3**: il controller del disco avvia il trasferimento DMA.

**Punto 4**: il controller del disco invia i dati byte per byte al controller DMA.

**Punto 5**: il controller DMA trasferisce i byte alla locazione di memoria di indirizzo X, incrementando l’indirizzo di memoria e decrementando C finché C=0.

**Punto 6**: quando C=0, il controller DMA genera un segnale d’interruzione per segnalare alla CPU che il trasferimento è terminato.

**Come si è visto tutto questo è avvenuto senza l’utilizzo della CPU.**

Interfaccia di I/O per le applicazioni

* Le chiamate del sistema di I/O incapsulano il comportamento dei dispositivi in alcune classi generiche. Cioè dato che esiste un gran numero di dispositivi, i metodi possono essere racchiusi in classi che hanno molto in comune, tipo lettura, scrittura o altre caratteristiche. Quindi il SO racchiude delle funzionalità di base per l’accesso a questi dispositivi, poi la particolarizzazione dell’accesso allo specifico dispositivo viene realizzata ,mediante ad esempio un driver che dettaglia nel particolare quel determinato tipo di accesso.
* **Lo scopo dello strato dei driver dei dispositivi è di nascondere al sottosistema di I/O del nucleo le differenze fra i controller dei dispositivi.**
* **I dispositivi possono differire in molti aspetti:**
* Trasferimento a flusso di caratteri o a blocchi
* Accesso sequenziale o diretto
* Sincroni o asincroni
* Condivisibili o riservati
* Velocità di funzionamento
* Lettura e scrittura oppure solo lettura o solo scrittura

Struttura relativa all’ I/O di un nucleo

Immagine che contiene screenshot

Descrizione generata automaticamente

**Spiegazione**

* In questo esempio ci sta un nucleo, con il sottosistema di I/O del nucleo, poi ci sono i vari driver. La prima parte (dove sta scritto programmi) è una parte software, cioè riguarda i veri e propri programmi che devono effettuare la gestione dell’I/O. Questa poi si interfaccia all’hardware vero e proprio, ovvero i vari controller e dispositivi fisici.
* Quindi se il nucleo deve colloquiare con uno dei dispositivi, lo deve fare con l’interfaccia (sottosistema I/O) la quale provvede a contattare il driver del dispositivo interessato che verrà poi relazionato al corrispondente controller che mediante il canale specifico comunicherà col dispositivo.

Caratteristiche per dispositivi per l’I/O

Immagine che contiene screenshot

Descrizione generata automaticamente

**Spiegazione**

Per necessità è possibile realizzare una sorta di tabella che categorizzi in base all’aspetto, le variazioni che si possono avere su una determinata caratteristica. Ad esempio il metodo di accesso, può essere sequenziale o casuale.

Dispositivi con trasferimento a blocchi o a caratteri

I dispositivi con trasferimenti sono caratterizzati dal fatto che possono essere a blocchi o a caratteri.

* I dispositivi con trasferimento a blocchi includono le unità a disco.
* Le istruzioni comprendono read, write e seek
* I/O a basso livello (raw I/O) o accesso tramite file-system
* Possibile accesso al file associato alla memoria
* **I dispositivi con trasferimento a caratteri includono tastiere, mouse, porte seriali. Si può ipotizzare di impacchettare questi caratteri per simulare un accesso a blocchi. Per esempio è possibile costruire servizi aggiuntivi quali l’accesso riga per riga; con l’accesso riga per riga si sta dicendo espressamente che si sta leggendo blocchi di caratteri.**
* I comandi principali per il trasferimento a caratteri comprendono le funzioni **get , put.**

Dispositivi di rete

Quando si parla di I/O, bisogna qualificare se questi accessi sono locali o remoti, quindi si parla di dispositivi di rete. In realtà conviene fare questo distinguo perché i dispositivi di rete necessitano di regole completamente diverse da quelle che sono i dispositivi di I/O locali.

* Poiché **i modi di indirizzamento e le prestazioni tipiche dell’I/O di rete sono notevolmente differenti da quelli dell’I/O delle unità a disco,** la maggior parte dei sistemi operativi fornisce un’interfaccia per l’I/O di rete diversa da quella caratterizzata dalle operazioni read, write e seek usata per i dischi.
* **Un’interfaccia disponibile in molti sistemi operativi, tra i quali Unix e Windows NT è l’interfaccia di rete socket** (letteralmente “presa di corrente”)
* Separa il protocollo di rete dalle operazioni di rete
* Include la funzione **select**
* **Gli approcci variano ampiamente ( pipe half-duplex, code FIFO full-duplex, STREAMS, code di messaggi e socket)**

Orologi e temporizzatori

* **Segnano l’ora corrente, segnalano il tempo trascorso, regolano un temporizzatore**
* Il dispositivo che misura la durata di un lasso di tempo e che può avviare un’operazione si chiama temporizzatore programmabile.
* ioctl (UNIX), input-output-controller tratta gli aspetti dell’I/O quali orologi e temporizzatori

I/O bloccante e non bloccante

* **Bloccante:** si sospende l’esecuzione dell’applicazione
* Codice più facilmente comprensibile
* Insufficiente per alcune necessità
* **Non bloccante:** sovrappone elaborazione e I/O; quindi si ha una sorta di parallelismo, ovvero si ha sia I/O in progress e sia la CPU che continua l’elaborazione.
* Interfaccia d’utente, applicazione per il video digitale
* Si realizza attraverso il multithreading
* Restituisce rapidamente il controllo dell’applicazione fornendo un parametro che indica quanti byte di dati sono stati trasferiti
* **Chiamate del sistema asincrone ( fa parte dei non bloccanti): restituiscono immediatamente il controllo al chiamante senza attendere che l’I/O sia stato completato**

Sottosistema per l’I/O del nucleo

* **Scheduling**
* Fare lo scheduling di un insieme di richieste di I/O significa stabilirne un ordine d’esecuzione efficace; l’ordine in cui si verificano le chiamate del sistema delle applicazioni è raramente la scelta migliore.
* Alcuni sistemi operativi tentano di essere equi
* **Memorizzazione transitoria:** un buffer (memoria di transito) è un’area di memoria che contiene dati mentre vengono trasferiti tra due dispositivi o fra un’applicazione e un dispositivo
* **Necessità di gestire la differenza di velocità fra il produttore e il consumatore di un flusso di dati**
* Gestione dei dispositivi che trasferiscono dati in blocchi di dimensioni diverse
* Realizzazione della “semantica delle copie”
* Cache: **regione di memoria veloce per copie di dati**
* Sempre solo copia di informazioni già memorizzate
* Migliora l’efficienza
* **Code (spooling): memoria di transito contenente dati per un dispositivo che non può accettare flussi di dati intercalati**
* Quando il dispositivo può gestire solo una richiesta alla volta
* Es. una stampante
* Uso esclusivo dei dispositivi: fornisce accesso esclusivo a un dispositivo
* Un processo può accedere a un dispositivo che non sia già attivo riservandosene l’uso e restituendolo al sistema quando non ne ha più bisogno
* Le applicazioni hanno la responsabilità di evitare situazioni di stallo
* **altra problematica dei sistemi I/O sono legate alle cache, le quali è vero che migliorano l’efficienza in quanto accelerano l’accesso ai dispositivi, ma complicano un aspetto, ovvero, quello delle copie, questo perché una copia si troverà nella cache, un’altra in RAM o nell’hard disk; quindi il problema della cache viene a tradursi con quello di sincronizzazione della cache. Di conseguenza ad esse saranno associate strategie dio sincronizzazione.**

Gestione degli errori

* Un sistema operativo che usi la protezione della memoria può proteggersi da molti tipi di **errori dovuti ai dispositivi o alle applicazione**
* **Di norma, una chiamata del sistema per l’I/O riporta un bit d’informazione sullo stato d’esecuzione della chiamata**
* Molti dispositivi SCSI mantengono alcune pagine di informazioni sugli errori avvenuti; queste pagine possono essere richieste dalla macchina, ma ciò accade raramente.

Strutture di dati del nucleo

* **Il nucleo ha bisogno di mantenere informazioni sullo stato dei componenti coinvolti nelle operazioni di I/O**
* Il SO UNIX, per mezzo del file system, permette l’accesso a diversi oggetti: i file degli utenti, **i dispositivi,** lo spazio d’indirizzi dei processi, e altri ancora
* **Alcuni sistemi operativi applicano metodi orientati agli oggetti e un sistema basato sullo scambio di messaggi.**

**Struttura di dati del nucleo per la gestione dell’I/O nello UNIX**

**Immagine che contiene screenshot

Descrizione generata automaticamente**

**Spiegazione**

Da una parte la memoria del processo utente dall’altra la memoria del nucleo. In questo caso si vuole fare l’accesso a un file, quindi si ha bisogno di un descrittore di un file che accede ad una tabella che a sua volta accede ad una serie di puntatori i quali sono registrati nel file system, oppure se si deve accedere in remoto si dovrebbe accedere sempre a strutture dati le quali sono puntatori che riguardano invece la rete. In ogni caso, ovvero, sia nella registrazione dei record file system, che dei record di rete, si dovrebbe mantenere una tabella su quali sono le risorse mantenute attive.

Trasformazione delle richieste di I/O in operazioni dei dispositivi

Si consideri una semplice operazione come una lettura di un file sul disco, fare tale operazione è molto complesso in quanto bisogna tenere in considerazione:

* Si determina il dispositivo che detiene il file
* Si traduce il nome nella rappresentazione del dispositivo
* Si trasferiscono i dati dal disco al buffer
* Si rendono disponibili i dati al processo
* Si restituisce il controllo al processo

Struttura di STREAMS

Immagine che contiene testo, screenshot

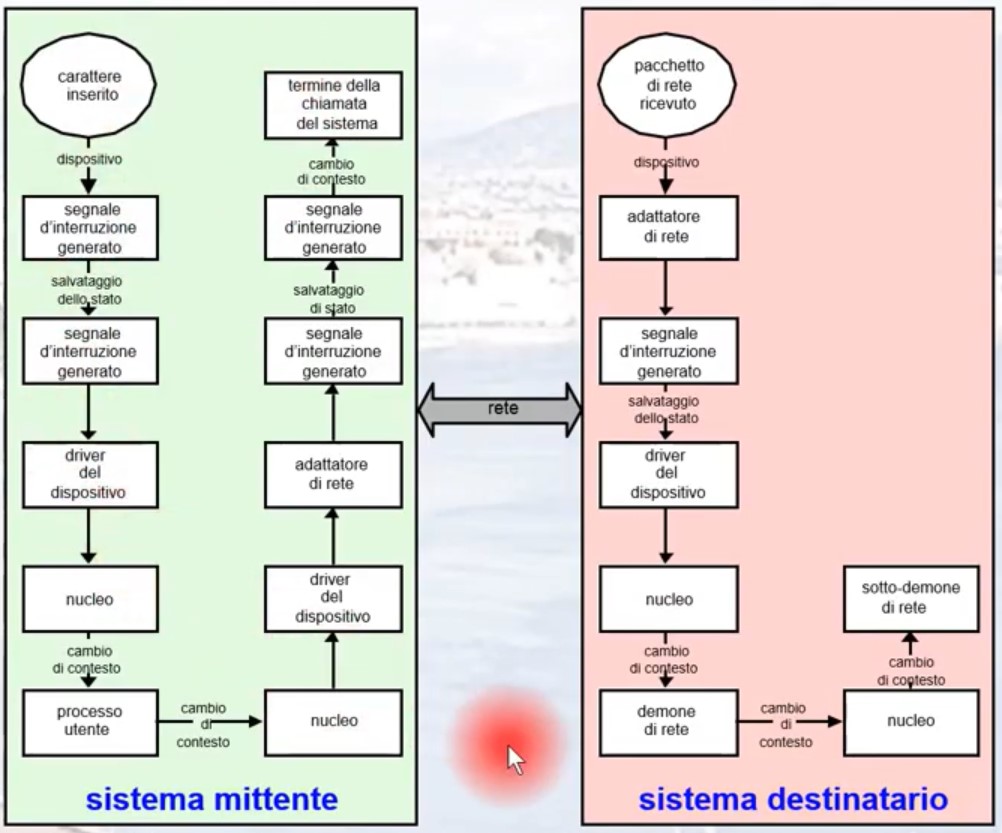
Descrizione generata automaticamente**spiegazione**

Ci sta un processo utente che ha una fase iniziale ed una fase terminale, ed è interposta da una serie di strutture e sottostrutture (moduli) che determinano delle code di lettura e code di scrittura a seconda del tipo di processo col la quale si sta lavorando.

Prestazioni

L’I/O è uno tra i principali fattori che influiscono sulle prestazioni di un sistema:

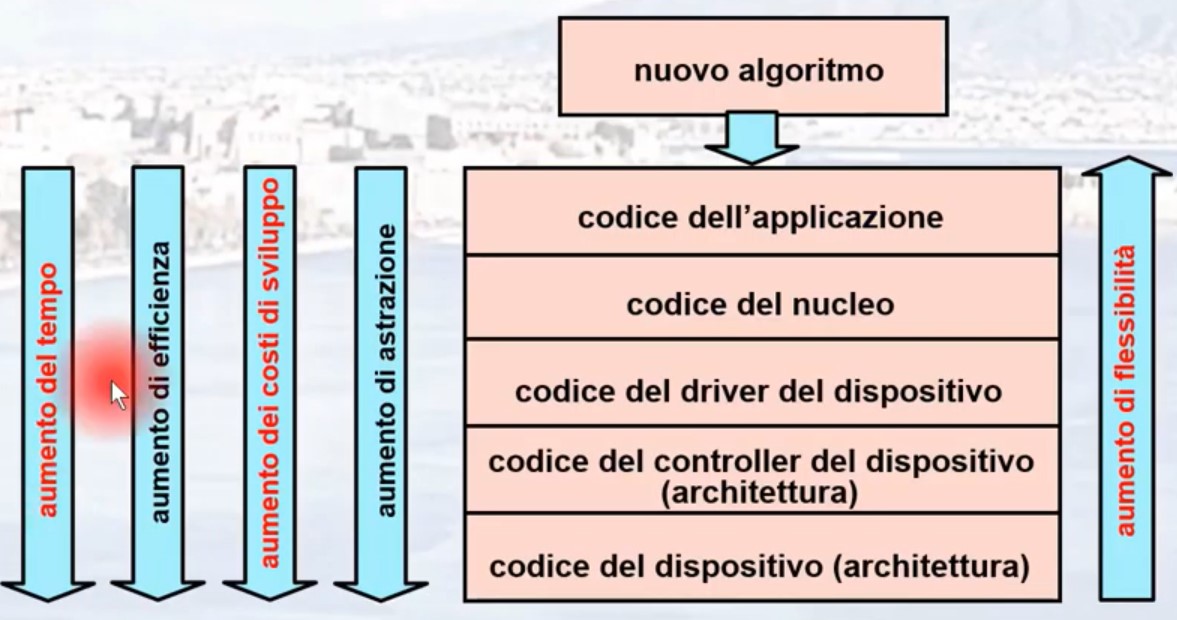
* Richiede un notevole impegno di CPU per l’esecuzione del codice del driver e per uno scheduling equo ed efficiente.
* I risultanti cambi di contesto sfruttano fino in fondo la CPU e le sue memorie cache
* Copia dei dati
* Traffico di rete



Migliorare le prestazioni

* Per migliorare l’efficienza dell’I/O si possono applicare diversi principi:
* Ridurre il numero dei cambi di contesto
* **Ridurre il numero di copiature dei dati**
* **Ridurre la frequenza delle interruzioni tramite il trasferimento di grandi quantità di dati in un’unica soluzione,** l’uso di **controller intelligenti** e mediante l’interrogazione ciclica
* Uso di controllori DMA intelligenti
* Equilibrare le prestazioni della CPU, del sottosistema per la gestione della memoria, del bus e dell’I/O

Successione delle funzionalità dei servizi di I/O

 **Spiegazioni**

Gli aspetti da tener presente nel momento in cui si realizza un nuovo algoritmo, soprattutto in fase di prototipizzazione sono diversi.

Spesso in fase di prototipizzazione si ha la necessità di coinvolgere non solo uno strato generale che coinvolge il codice dell’applicazione, ma anche livelli più bassi come ad esempio codici di driver del dispositivo, codici del controller del dispositivo oppure codice del dispositivo stesso.

Ovviamente questa fase iniziale (dove sta scritto nuovo algoritmo) avrà un costo maggiore, infatti c’è un aumento del tempo (prima freccia) da tenere in considerazione, cosi come l’aumento dei costi di sviluppo.

Si può concludere quindi che tale fase iniziale è molto importante perché coinvolge aspetti che poi nella fase di prototipizzazione avranno un aumento della flessibilità, in quanto l’algoritmo che si sta creando, dopo la fase di prototipizzazione, escluderà dal suo campo visivo tutti questi aspetti (vari rettangoli elencati in figura) **i quali sono legati a livelli inferiori e vengono normalmente gestiti dal SO.**