CAPITOLO 5

#Sistemi_Operativi

Principi dell'Hardware di I/O

Classificazione dei dispositivi I/O:

I dispositivi I/O possono essere suddivisi in tre tipologie principali:

- Dispositivi a blocchi : ad es. HD e SSD; archiviano i dati in blocchi di dimensioni fisse, ciascuno con il proprio indirizzo. Ogni blocco è trattato come un'unità indipendente, per questo i trasferimenti avvengono in unità di uno o più blocchi interi. Supportano inoltre operazioni di ricerca diretta sui dati, rendendoli adatti per l'archiviazione di file e la gestione di database.
- Dispositivi a caratteri : ad es. mouse, tastiere e stampanti; operano a flussi di caratteri senza una struttura a blocchi definita. A differenza dei dispositivi a blocchi questi dispositivi non sono indirizzabili a livello di blocco e non supportano operazioni di ricerca sui dati. Invece si occupano di gestire input e output come flusso continuo di dati, il che li rende ideali come dispositivi che trasmettono informazioni in modo sequenziale.
- Altre tipologie : in questa categoria si includono altri dispositivi specializzati per scopi diversi e che non rientrano nelle altre due, tra questi i dispositivi di rete, i clock e altri ancora.

Controller dei dispositivi:

Ogni dispositivi I/O è costituito da un controller , la parte elettronica responsabile dell'interfacciamento tra dispositivo fisico e sistema digitale. Le interfacce tra controller e dispositivo possono essere standardizzate, come SATA, SCS, USB.

Porte di comunicazione

Le comunicazioni tra dispositivi I/O e computer avvengono tramite diverse porte di comunicazione, ad esempio:

- Porta Parallela : trasmette dati multi-bit simultaneamente su più canali utilizzando 25 o 36 pin. Era tipicamente utilizzata per connessioni ad alta velocità e per dispositivi come stampanti o scanner. Tuttavia tendenzialmente non sono più comunemente utilizzate nei calcolatori moderni, essendoci alternative più veloci e versatili come USB.

- Porta USB : Universal Serial Bus, è un'interfaccia standardizzata per la comunicazione tra dispositivi I/O e computer. USB non solo trasmette dati ma può anche fornire alimentazione elettrica ai dispositivi collegati, il che la rende molto versatile. USB utilizza un numero minore di pin rispetto alle porte parallele, ma ogni pin è specializzato in una funzione specifica come trasmettere e ricevere dati, alimentare, gestire segnali di controllo.

Metodi di accesso ai dispositivi I/O

Due principali:

- Port-Mapped I/O : in questo metodo, i dispositivi sono accessibili tramite porte di I/O con numeri di porta specifici. In particolare ogni registro di controllo del dispositivo (che si trova all'interno del dispositivo I/O) è mappato a un numero di porta di I/O (che rappresenta un indirizzo utilizzato dalla CPU per comunicare con il dispositivo). Per leggere o scrivere dati ai dispositivi, vengono utilizzati dal processore istruzioni speciali come IN e OUT , specifiche per l'accesso alle porte I/O. Ciò offre il vantaggio di riservare uno spazio separato per l'I/O, evitando potenziali conflitti per l'indirizzamento della memoria, ma lo svantaggio che avendo istruzioni specifiche complica la programmazione dei driver.
- Memory-Mapped I/O : i registri di controllo dei dispositivi I/O sono mappati direttamente nello spazio di indirizzamento della memoria. Ciò implica che i dispositivi I/O sono trattati come fossero normali locazioni di memoria, e che quindi sono accessibili usando le normali istruzioni di lettura e scrittura della memoria LOAD e STORE Al contrario del Port-Mapped I/O ciò semplifica la programmazione dei driver con codici più uniformi e spesso efficienti, ma poiché i registri di controllo I/O occupano parte dello spazio di indirizzamento della memoria lo spazio in RAM è ridotto e possono avvenire dei conflitti tra memoria e I/O.

Direct Memory Access DMA

Il DMA è un dispositivo hardware che consente ai dispositivi I/O di trasferire dati direttamente nella memoria principale senza coinvolgere la CPU.
Questo processo riduce il carico sulla CPU e aumenta l'efficienza del sistema, poiché la CPU non è impegnata nel trasferimento dati. Infatti:

- Senza DMA il controller del disco legge i dati, li memorizza nel suo buffer e genera interrupt. Il SO deve quindi copiare manualmente i dati dalla memoria del controller del disco (componente hardware che gestisce l'interfaccia tra disco (HD o SSD) e il resto del computer) alla memoria principale impegnando la CPU e rallentando il processo complessivo.
- Con DMA invece la CPU inizializza il controller DMA e invia un comando al controller del disco per avviare la lettura/scrittura dei dati. Il controller DMA coordina direttamente il trasferimento dati tra dispositivi I/O e memoria principale "collaborando" con il controller

del disco. Al termine del trasferimento, il DMA notifica la CPU tramite un interrupt, permettendo così alla CPU di occuparsi di altre operazioni durante il trasferimento.

Le modalità di interazione tra DMA e BUS includono:

- Cycle Stealing : il DMA prende temporaneamente il controllo del bus per eseguire il trasferimento dei dati, sottraendo alcuni cicli di bus alla CPU. Questo permetta di minimizzare l'impatto del trasferimento DMA sul funzionamento generale della CPU.
- Modalità Burst : Il DMA prende il controllo completo del bus per eseguire il trasferimento di un blocco di dati in un'unica operazione. Questa modalità è efficiente per il trasferimento di grandi quantità di dati in modo rapido.
- Modalità Fly-By-Mode : standardizzata con il nome Fly-By-DMA, è il tipo di trasferimento di dati che consente di spostare dati direttamente tra il dispositivo di I/O e la memoria senza richiedere un'ulteriore elaborazione da parte della CPU durante il trasferimento stesso.

Interrupt

Gli interrupt sono segnali inviati dai dispositivi di I/O (o da altre componenti del sistema) per richiedere l'attenzione immediata della CPU su eventi specifici.
La CPU può dover interrompere il suo lavoro attuale per gestire l'evento segnalato.
Gli interrupt possono essere causati da:

- Trap : è un tipo di interrupt causato da un'azione intenzionale all'interno di un programma. Le trap sono solitamente utilizzate per passare dal codice utente al codice del kernel (ad esempio, quando si esegue una system call).
- Fault o Eccezione : questi si verificano a causa di eventi inattesi o errori, come errori di segmentazione (segmentation fault) o divisione per zero. Sono eventi che richiedono l'intervento immediato del sistema operativo per gestire l'errore.
- Interrupt Hardware : questi sono segnali inviati dai dispositivi hardware (come stampanti, tastiere, o schede di rete) alla CPU per segnalare che un'operazione è completata o che è richiesta attenzione per un evento (ad esempio, un dato è pronto per essere letto).

Quando un dispositivo I/O invia un segnale di interrupt alla CPU, il segnale viene gestito da un componente hardware chiamato Interrupt Controller (che può essere parte del chipset della scheda madre o un componente separato).
L'interrupt Controller determina la priorità dell'interrupt e decide se deve essere gestito immediatamente o messo in coda (se ci sono altri interrupt in corso). Se l'interrupt ha una priorità superiore, può causare la sospensione dell'operazione corrente della CPU per gestire l'interrupt.

Gestione degli interrupt

I passaggi principali sono:

- Segnalazione : quando un dispositivo I/O richiede l'attenzione della CPU, invia un segnale interrupt al controller degli interrupt. Quest'ultimo allora assegna un numero di linea di interrupt al dispositivo (detto numero di interrupt) e poi invia il segnale di interrupt alla CPU.
- Interruzione : quando la CPU riceve l'interrupt, interrompe l'attività in corso. A questo punto la CPU disabilita ulteriori interrupt (o solo quelli di priorità inferiore) per evitare di essere ulteriormente interrotta nell'arco della gestione dell'interrupt.
- Vettore degli interrupt : la CPU utilizza il numero di interrupt ricevuto per accedere alla Tabella del Vettore degli Interrupt. Tale tabelle contiene gli indirizzi delle procedure ISR per ciascun tipo di interrupt, e grazie all'associazione numero di interrupt - ISR la CPU ottiene l'ISR corretta.
- Gestione : L'ISR viene eseguita per gestire l'interrupt. Questa procedura esegue le operazioni necessarie per rispondere all'evento che ha causato l'interrupt (ad esempio, leggere un dato da un dispositivo di I/O). Al termine dell'ISR, il completamento viene segnalato al controller degli interrupt, che libera la linea di interrupt.
- Salvataggio dello stato : Durante il processo di gestione dell'interrupt, la CPU deve salvare lo stato del processo interrotto (almeno il Program Counter e altri registri cruciali) per poter riprendere correttamente l'esecuzione dopo aver gestito l'interrupt. Questo salvataggio dello stato avviene nello stack , che può essere lo stack del processo o lo stack del kernel, a seconda dell'implementazione del sistema operativo. La scelta tra l'utilizzo dello stack del processo o dello stack del kernel può influenzare la gestione della memoria e le prestazioni complessive del sistema.

Tipologie di interrupt

Gli interrupt possono presentarsi in due circostanze diverse:

- Interrupt Precisi : un interrupt è considerato "preciso" quando il sistema può determinare esattamente quali istruzioni sono state completate e quali no. In altre parole, tutte le istruzioni eseguite prima del Program Counter (PC) sono completate, nessuna istruzione dopo il PC è stata eseguita, e lo stato dell'istruzione puntata dal PC è noto. In questo caso la CPU riesce a gestire l'interrupt garantendo compatibilità e prevedibilità.
- Interrupt Imprecisi : un interrupt è considerato "impreciso" quando, al momento dell'interrupt, diverse istruzioni vicino al Program Counter si trovano in vari stati di completamento. Questo rende incerto lo stato esatto del programma. Poiché non è chiaro quali istruzioni siano state completate e quali no, la CPU deve salvare una quantità significativa di stato interno sullo stack per poter riprendere correttamente l'esecuzione del programma.

Principi del Software I/O

Si delineano obiettivi, tipologie di software e la struttura del sistema di gestione degli I/O.

Obiettivi

1. Indipendenza dal dispositivo : il software di I/O dovrebbe consentire l'accesso a diversi dispositivi senza richiedere specifiche anticipate sul tipo di dispositivo.
2. Denominazione Uniforme: I nomi dei file o dei dispositivi dovrebbero essere rappresentati da stringhe o numeri indipendenti dal dispositivo specifico.
3. Gestione degli errori : gli errori dovrebbero essere gestiti il più vicino possibile all'Hardware, preferibilmente dal controller o dal driver del dispositivo.
4. Trasferimenti Sincroni vs Asincroni : sebbene la maggior parte dell'I/O fisico sia asincrono, molti programmi trattano l'I/O come se fosse sincrono per semplicità. Il sistema operativo offre comunque l'accesso all'I/O asincrono per applicazioni ad alte prestazioni.
5. Buffering : spesso i dati dai dispositivi non vanno direttamente alla destinazione finale, ma passano attraverso un buffer temporaneo. L'uso del buffering può influenzare le prestazioni, specialmente per dispositivi con requisiti di tempo reali.
6. Dispositivi Condivisibili vs Dedicati : Dispositivi come dischi e SSD possono essere condivisi da più utenti contemporaneamente, mentre altri come stampanti e scanner sono tipicamente dedicati a un singolo utilizzatore.

Tipologie di Software per I/O

- I/O Programmato : la CPU gestisce direttamente il trasferimento dei dati, impegnando continuamente la CPU nel polling per controllare lo stato del dispositivo.
- I/O Guidato dagli interrupt : utilizza gli interrupt per segnalare alla CPU quando un dispositivo è pronto per l'elaborazione successiva, riducendo il polling e consentendo alla CPU di eseguire altre operazioni.
- I/O con DMA : il Direct Memory Access (DMA) permette al controller DMA di trasferire dati tra il dispositivo e la memoria principale senza coinvolgere direttamente la CPU, migliorando l'efficienza complessiva del sistema.

Struttura del Software di I/O

Il software di I/O è organizzato di 4 livelli principali:

- Gestore degli interrupt : gestisce gli interrupt hardware e passa il controllo alla procedura di servizio degli interrupt appropriata.
- Driver dei dispositivi : gestisce le operazioni specifiche del dispositivo attraverso registri di dispositivo dedicati, facilitando l'interazione tra il sistema operativo e l'hardware del

dispositivo.

- Software del SO indipendente dal dispositivo : fornisce un'interfaccia uniforme per i driver dei dispositivi, gestendo il buffering dei dati, la gestione degli errori e l'allocazione dei dispositivi dedicati.
- Software per I/O a Livello Utente : include librerie di I/O che semplificano le chiamate di sistema per operazioni di I/O come lettura e scrittura, migliorando l'interfaccia tra le applicazioni utente e il sistema operativo.

In conclusione, il software di I/O è progettato per gestire in modo efficiente l'interazione tra le applicazioni utente e l'hardware del sistema, fornendo un'interfaccia standardizzata e gestendo le operazioni di I/O in modo da massimizzare le prestazioni complessive del sistema informatico.