Sistemi Operativi

Ionut Zbirciog

19 December 2023

Principi del Software di I/O

Obiettivi

- **Indipendenza dal Dispositivo:** Il software di I/O dovrebbe permettere l'accesso a diversi dispositivi senza specificare il tipo di dispositivo in anticipo.
- **Denominazione Uniforme:** I nomi di file o dispositivi dovrebbero essere stringhe o numeri indipendenti dal dispositivo.
- **Gestione degli Errori:** Gli errori vanno gestiti il più vicino possibile all'hardware, idealmente dal controller stesso o dal driver del dispositivo. Errori transitori, come lettura da disco, spesso scompaiono ripetendo l'operazione.
- **Trasferimenti Sincroni vs Asincroni:** La maggior parte dell'I/O fisico è asincrono, ma per semplicità, molti programmi utente trattano l'I/O come se fosse sincrono. Il sistema operativo rende operazioni asincrone sembranti bloccanti, ma fornisce l'accesso all'I/O asincrono per applicazioni di alte prestazioni.
- **Buffering:** Spesso i dati da un dispositivo non vanno direttamente alla destinazione finale, richiedendo un buffer temporaneo. L'uso di buffer può influenzare le prestazioni, soprattutto per dispositivi con vincoli real-time.
- **Dispositivi Condivisibili vs Dedicati:** Dispositivi come dischi e SSD possono essere condivisi da più utenti, mentre altri come stampanti e scanner sono tipicamente dedicati.

Tipologie di Software per I/O

I/O Programmato

La CPU gestisce direttamente tutto il processo di trasferimento di dati. Un esempio pratico:

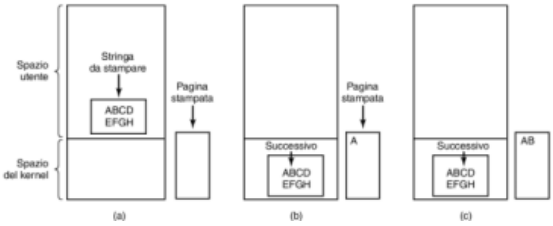

Figure 1: Processo di stampa con I/O programmato

Un processo utente prepara un stringa in un buffer dello spazio utente. Il processo effettua una chiamata di sistema per stampare la stringa, dopo aver ottenuto l'acceso alla stampante (a). Il sistema operativo copia il buffer in uno spazio kernel. Invia i caratteri alal stampante uno alla volta, aspettando che questa sia pronta per ogni carattere (b) e (c). Il sistema operativo entra in un ciclo di polling (busy waiting), controllando il registro di stato della stampante e inviando un carattere alla volta.
Occupa la CPU a tempo pieno durante il processo di I/O, facendo continuamente polling sullo stato della stampante. L'I/O programmato è efficace quando il tempo di elaborazione di un carattere è breve.
Il codice di esempio:

```
copy_from_user(buffer, p, count);
for(int i = 0; i < count; i++){
    while (*printer_status_reg != READY);
    *printer_data_register = p[i];
}
return_to_user();
```

I/O Guidato dagli Interrupt

Andiamo a considerare una stampante che non ha buffer, ma stampa un carateere dopo l'altro appena arriva, con un ritardo di 10 ms. In questo modo la CPU ha un tempo di 10 ms in cui potrebbe fare altro. Per permettere ciò, Dopo la chiamata di sistema per stampare la stringa, il buffer viene copiato nello spazio del kernel, come abbiamo mostrato prima, e il primo carattere è copiato nella stampante appena è in grado di accettarlo. A questo punto la CPU chiama lo scheduler e viene eseguito un altro processo. Il processo che ha richiesto la stampa della stringa è bloccato finché non è stampata l'intera stringa Quando la stampante ha stampato il carattere ed è pronta ad accettare il successivo, genera un interrupt che ferma il processo attuale e ne salva lo stato. Poi è eseguita la procedura di servizio di interrupt della stampante;

I/O con DMA

Uno svantaggio ovvio dell'I/O guidato dagli interrupt è che avviene un interrupt a ogni carattere. Gli interrupt richiedono tempo, pertanto questo schema spreca una certa quantità di tempo della CPU. Una soluzione è l'uso del DMA. In questo caso l'idea è di lasciare che il controller DMA invii i caratteri alla stampante uno alla volta, senza disturbare la CPU. In sostanza, il DMA è I/O programmato, solo che il lavoro è delegato al controller DMA anziché alla CPU principale. Questa strategia richiede un hardware speciale (il controller DMA), ma lascia che la CPU faccia altre cose durante l'I/O. Il grosso vantaggio del DMA consiste nella riduzione del numero degli interrupt da uno per carattere a uno per buffer stampato.

Struttura del Software di I/O

Il software di I/O è generalmente organizzato in quattro livelli, come mostrato nella figura, ciascuno dei quali ha una funzione ben definita da eseguire e un'interfaccia ben definita verso i livelli adiacenti.

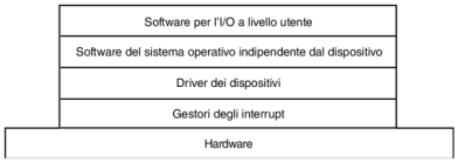

Figure 2: Struttura del software di I/O

Gestore degli Interrupt

Passaggi nel software dopo il completamento dell'interrupt hardware:

1. Salvataggio di tutti i registri (incluso il PSW) non ancora salvati dall'interrupt hardware.
2. Impostazione di un contesto per la procedura di servizio dell'interrupt. Potrebbe implicare l'impostazione di TLB, MMU e una tabella delle pagine.
3. Impostazione di uno stack per la procedura di servizio dell'interrupt.
4. Conferma al controller degli interrupt. Qualora manchi il controller centralizzato degli interrupt, riabilitazione degli interrupt.
5. Copia dei registri da dove erano stati salvati (magari su qualche stack) alla tabella dei processi.
6. Esecuzione della procedura di servizio dell'interrupt. Tipicamente estrarrà informazioni dai registri del controller del dispositivo che ha generato l'interrupt.
7. Scelta di quale processo eseguire come successivo. Se l'interrupt ha fatto sì che qualche processo a priorità alta che era bloccato sia ora pronto, potrebbe essere scelto adesso per l'esecuzione.

8. Impostazione del contesto della MMU per il processo successivo da eseguire. Potrebbe essere anche necessario impostare il TLB.
9. Caricamento dei nuovi registri del processo, incluso il suo PSW.
10. Avvio dell'esecuzione del nuovo processo.

Driver dei Dispositivi

Il ruolo del driver è di gestire i dispositivi di I/O attraverso registri di dispositivi specifici. I driver oltre a gestire dispositivi specifici, possono anche gestire classi di dispositivi. Ogni dispositivo necessita di un codice specifico, noto come driver del dispositivo solitamente fornito dal produttore. Tecnologie come USC utilizzano una pila di driver per gestire una vasta gamma di dispositivi. I driver di solito fanno parte del kernel del sistema operativo per poter accedere ai registri del controller del dispositivo. Se sono usati nello spazio utente, sono più facili da installare, mettono meno a rischio il SO ma sono più lenti. Il sistema operativo deve permettere l'installazione di codice scritto da terze parti. I driver si posizionano sotto il resto del sistema operativo. In alcuni sistemi operativi, i driver sono inclusi nel programma binario del sistema operativo, nei sistemi moderni, i driver vengono caricati dinamicamente. I sistemi operativi classificano i driver in due categorie. A blocchi, come i dischi, a caratteri come le stampanti e le tastiere.
Il driver gestisce le scriture e le letture, inizializza il dispositivo, verifica la validità dei parametri di input, traduce i parametri in comandi specifici per il dispositivo.

Software del SO Indipendente dal Dispositivo

Il software di I/O indipendente dal dispositivo funge da intermediario tra i driver specifici dei dispositivi e le applicazioni utente. Mira a semplificare l'intereazione con i dispositivi hardware offrendo un interfaccia uniforme e gestendo operazioni comuni.

- Interfaccia Uniforme dei Driver dei Dispositivi: Fornisce un'interfaccia standard per diversi tipo di driver di dispositivo. Evita la necessita di modificare il sistema operativo ogni volta che viene introdotto un nuovo dispositivo. Standardizza un modello uniforme dove tutti i driver condividono la stessa interfaccia verso il SO. Ogni classe di dispositivi ha un insieme definito di funzioni che i driver devono superare.
- Buffering: Gestisce i buffer per l'efficienza del trasferimento dei dati tra i dispositivi e il sistema.
- Gestione degli Errori: Identifica e comunica gli errori provenienti dai dispositivi all'utente o ad altri sistemi.
- Gestione dei Dispositivi Dedicati: Gestisce l'assegnazione e la liberazione di dispositivi dedicati a specifici compiti o utenti.
- Uniformità della Dimensione dei Blocchi: Assicura che la dimensione dei blocchi di dati sia gestita in modo uniforme, indipendentemente dalla specifica del dispositivo.

Software per I/O a Livello Utente

Il ruolo delle librerie di I/O è di facilitare le chiamate di sistema per l'I/O, per esempio la funzione write() o read() in C. Funzioni come printf() e scanf() trasformano e gestiscono i dati prima di invocare funzioni di sistema, facilitando operazioni di input e output. Queste librerie semplificano la programmazione di I/O, permettendo ai programmatori di concentrarsi sulla logia dell'applicazione. Lo spooling è un modo per interagire con i dispositivi dedicati in un sistema multiprogrammato, evitando il blocco prolungato da parte di un processo. E' implementato utilizzando un processo daemon e una directory di spooling, per ordinare e gestire i lavori di stampa. Lo spooling incrementa l'efficienza nell'uso dei dispostivi dedicati e migliora la gestione delle risorse, permettendo a più processi di accedere ai dispositivi in modo sequenziale. Spooling: mettere in coda per la stampa.