TANKERINO.com

Lezione 140

https://www.tankerino.com/it/corsi/4/tepi-3/lezioni/140/le-system-call

Lezione: System Call nei Sistemi Operativi

Le System Call sono interfacce fornite dal sistema operativo che consentono ai programmi utente di comunicare con le risorse di basso livello del sistema, come dispositivi hardware e funzionalità di sistema. In sostanza, rappresentano il principale punto di contatto tra il software utente e il kernel del sistema operativo.

Per comprendere meglio il concetto di System Call, è importante analizzare alcuni esempi specifici di operazioni che richiedono l'uso di queste chiamate di sistema.

Principali Tipologie di System Call

Esistono diverse categorie di System Call, ognuna delle quali offre funzionalità specifiche per interagire con il sistema operativo. Le principali tipologie di System Call includono:

- Gestione dei File: Queste System Call consentono ai programmi di creare, aprire, leggere, scrivere e chiudere file sul sistema di archiviazione.
- Gestione dei Processi: Queste System Call permettono di creare, terminare, sospendere, riprendere e gestire i processi in esecuzione sul sistema.
- Gestione della Memoria: Queste System Call forniscono funzionalità per l'allocazione e la gestione della memoria sia per i processi utente che per il kernel.
- Comunicazione e Sincronizzazione: Queste System Call consentono ai processi di comunicare tra loro e sincronizzare le loro attività, utilizzando meccanismi come le code di messaggi, i semafori e i mutex.
- Gestione dei Dispositivi: Queste System Call permettono l'accesso e il controllo dei dispositivi hardware collegati al sistema, come unità di archiviazione, periferiche di rete e dispositivi di input/output.

Esempi di Utilizzo delle System Call

Per comprendere meglio come le System Call vengono utilizzate nei programmi, consideriamo alcuni esempi pratici:

- Apertura di un File: Un programma che desidera leggere o scrivere su un file utilizzerà la System Call *open()* per aprire il file specificato.
- Lettura da un File: Dopo l'apertura di un file, il programma può utilizzare la System Call *read()* per leggere i dati dal file aperto e caricarli in memoria.
- Scrittura su un File: Per scrivere dati su un file, il programma utilizzerà la System Call *write()* per scrivere i dati dalla memoria sul file.
- Creazione di un Processo: Per avviare un nuovo processo, un programma utilizza la System Call *fork()* per duplicare il processo corrente e *exec()* per eseguire un nuovo programma nel processo figlio.
- Gestione della Memoria: Le System Call come *malloc()* e *free()* sono utilizzate per allocare e liberare la memoria dinamica per le variabili e le strutture dati.

Esempi di utilizzo della system call "stampare su schermo"

In questo esempio, utilizzeremo un approccio pseudo-assembly per illustrare il processo di esecuzione di una System Call per la scrittura di dati sullo schermo.

Di seguito è riportato il codice pseudo-assembly che simula l'esecuzione di una System Call:

```
MOV REGISTRO_OPERAZIONE, CODICE_SYS_WRITE

MOV REGISTRO_FILE_DESCRIPTOR,STDOUT_FILENO

MOV REGISTRO_DATO,INDIRIZZO_MESSAGGIO

MOV REGISTRO_LUNGHEZZA, LUNGHEZZA_MESSAGGIO

INT 0x80 ; Interruzione software per eseguire la System Call
```

Spiegazione

1. MOV: Questa istruzione viene utilizzata per spostare i valori nei registri specificati.
2. REGISTRO_OPERAZIONE, REGISTRO_FILE_DESCRIPTOR, REGISTRO_DATO e REGISTRO_LUNGHEZZA: Sono registri immaginari utilizzati per memorizzare rispettivamente il codice dell'operazione, il file descriptor, l'indirizzo del dato e la lunghezza del dato.
3. CODICE_SYS_WRITE: Rappresenta il codice univoco associato alla System Call `write()`.
4. STDOUT_FILENO: È il file descriptor per lo standard output, che indica lo schermo.
5. INDIRIZZO_MESSAGGIO: È l'indirizzo in memoria del messaggio che si desidera stampare.

6. LUNGHEZZA_MESSAGGIO: Rappresenta la lunghezza del messaggio.
7. INT 0x80: Questa istruzione provoca un'interruzione software che indica al kernel del sistema operativo di eseguire la System Call corrispondente. Il kernel interpreta i valori nei registri e esegue l'operazione richiesta, che nel nostro caso è la scrittura del messaggio sullo schermo.

Abbiamo utilizzato un'esposizione pseudo-assembly per mostrare il processo di esecuzione di una System Call per la scrittura di dati sullo schermo. Anche se questo è solo un esempio semplificato, illustra il concetto fondamentale di come i programmi comunicano con il kernel del sistema operativo per eseguire operazioni di basso livello.

Conclusioni

Le System Call rappresentano un'interfaccia vitale tra il software utente e il kernel del sistema operativo, consentendo l'accesso e il controllo delle risorse di sistema in modo sicuro e controllato. Comprendere il funzionamento delle System Call è fondamentale per gli sviluppatori di software e gli ingegneri di sistema, poiché influisce direttamente sulla progettazione e sull'implementazione dei programmi e dei servizi.

Ricorda che le System Call forniscono un'astrazione delle funzionalità di basso livello del sistema operativo, consentendo ai programmi utente di essere portabili tra diverse piattaforme senza dover conoscere i dettagli specifici dell'hardware sottostante.

Questa lezione e' stata realizzata grazie al contributo di:

Baobab.school
Risorse per la scuola
https://www.baobab.school

FrancescoBelloni.it software development
Siti web a Varese
https://www.francescobelloni.it