1 - Processi

Processi

Definizione

Un processo è un programma in esecuzione. In un sistema multiprogrammato, la CPU alterna rapidamente tra vari processi, eseguendoli per brevi periodi (decine o centinaia di millisecondi), creando l'illusione del parallelismo, noto come pseudoparallelismo.

Il modello del processo

Un processo include i valori attuali del contatore di programma, dei registri e delle variabili. Sebbene la CPU esegua un solo processo alla volta, la rapida alternanza crea l'illusione dell'esecuzione parallela, detta multiprogrammazione.

Come funziona il processo

- Esecuzione sequenziale: Ogni processo ha un unico program counter e la CPU passa da un processo all'altro in modo sequenziale.
- Esecuzione parallela: Ogni processo ha il proprio flusso di controllo (puntatori e area di memoria). Lo scheduler del sistema operativo decide quale processo assegnare alla CPU in base a priorità e politiche di scheduling, ottimizzando l'uso delle risorse.

Informazioni associate a un processo

- PID (Process ID), UID (User ID), GID (Group ID)
- Spazio degli indirizzi di memoria
- Registri hardware (Program Counter)
- File aperti
- Segnali
- Interrupt

Gerarchia di Processi

I processi sono organizzati in una gerarchia Parent-Child. In UNIX, un processo speciale chiamato init (PID 1) avvia altri processi necessari al sistema. I moderni sistemi init avviano kthreadd (PID 2) per la gestione dei thread.

Creazione del Processo

La creazione di un processo avviene tramite:

1. Inizializzazione del sistema: processi eseguiti all'avvio del sistema, in modo attivo o in background (deamon)
2. Chiamata di sistema (fork()).
3. Richiesta dell'utente (es. tramite bash).
4. Avvio di un lavoro in modalità batch (tramite script sh).

Termine di un Processo

Un processo può terminare per:

- Uscita normale (volontaria).
- Uscita a causa di un errore (volontaria).
- Errore fatale (involontario).
- Uccisione da parte di un altro processo (involontario).

Comandi per la gestione di un processo:

- fork: crea un processo figlio, che condivide le risorse del genitore (PC e registri).
- exec: esegue un nuovo processo.
- exit: termina volontariamente il processo.
- kill: invia un segnale a un processo per terminarlo.

Gli Stati di un Processo

- Running (In esecuzione): utilizza la CPU.
- Ready (Pronto): eseguibile, in attesa di CPU.
- Blocked (Bloccato): attesa di un evento esterno.

Le transizioni tra stati includono:

- Il processo si blocca in attesa di input. (running-blocked)
- Lo scheduler termina l'esecuzione di un processo per sceglierne un altro. (running-ready)
- Lo scheduler sceglie un processo pronto per eseguirlo. (ready-running)
- Un evento esterno risveglia un processo bloccato. (blocked-ready)

Signals vs Interrupts

Interrupts

- Origine: Dispositivi hardware. (pressione di un tasto sulla tastiera)
- Gestione: Routine di servizio di interrupt (ISR).
- Uso: Comunicazione tra hardware e software.
- Asincronia: Gestiti immediatamente.

Quando un interrupt si verifica, il program counter, lo stato del processo e dei registri vengono salvati nello stack e la ISR specifica viene eseguita. Dopo l'esecuzione dell'ISR, il processo riprende da dove era stato interrotto.
Il concetto degli interrupt è che un processo nel corso della sua esecuzione può essere interrotto più e più volte, ma è fondamentale che dopo ogni interrupt il processo torni nello stato in cui si trovava prima dell'interruzione.

Signals

- Origine: Eventi software.
- Gestione: Gestori di segnali personalizzati o comportamento predefinito.
- Uso: Gestione di condizioni eccezionali.
- Asincronia: Inviati asincronamente, gestiti sincronicamente.

I segnali sono impulsi asincroni tra processi. Tipicamente, non trasmettono dati. Ogni segnale ha un comportamento standard, ad esempio SIGINT interrompe un processo, SIGTERM termina un processo, e SIGKILL lo termina immediatamente.