UD 4 - COMUNICAZIONE TRA PROCESSI

Quando più processi condividono tra loro informazioni e le loro computazioni concorrono a uno scopo applicativo comune, si dicono cooperanti. Affinché ciò sia possibile, il SO deve fornire politiche e meccanismi di comunicazione (IPC) e sincronizzazione. Le entità coinvolte nella comunicazione sono il processo mittente (da ora P), il ricevente (da ora Q) e il canale di comunicazione.

Ogni metodo di IPC ha delle caratteristiche proprie. La scelta di un metodo da utilizzare in un'applicazione va fatta in base a:

- Quantità di informazioni da trasmettere
- Velocità di esecuzione
- Scalabilità, ossia se il metodo scelto permette anche a un numero variabile di processi di comunicare
- Semplicità d'uso nelle applicazioni
- Omogeneità delle comunicazioni: è preferibile evitare di utilizzare più metodi di IPC nella stessa applicazione, così da evitare errori comuni
- Integrazione nel linguaggio di programmazione, che garantisce la portabilità
- Affidabilità
- Sicurezza
- Protezione

Ci sono dei vantaggi nello scrivere applicazioni fatte da processi cooperanti:

- modularità
- parallelizzazione
- scalabilità
- specializzazione
- qualità del progetto e della realizzazione

Le due modalità di realizzazione sono:

- Comunicazione diretta, in cui i processi comunicanti conoscono l'ID l'uno dell'altro. Richiede che entrambi i processi siano attivi affinché avvenga la comunicazione
- Comunicazione indiretta, in cui i processi sanno solo dove prelevare/depositare le informazioni

Le varie implementazioni dei meccanismi di comunicazione possono essere:

Memoria condivisa

Variabili globali

E' una modalità di comunicazione diretta e può essere realizzata attraverso l'uso di variabili globali o buffer. Nella condivisione di variabili globali ho due processi con una porzione del loro spazio di indirizzamento (le variabili globali, appunto) che si sovrappone. Si può realizzare in 2 modi:

- Il SO copia la zona di memoria condivisa tra i processi comunicanti. In questo modo i processi comunicano pur rimanendo fisicamente separati. Comporta uno spreco di tempo e memoria per la copia.
- Il SO mappa parte dello spazio di indirizzamento logico dei processi sulla stessa area di memoria fisica, che quindi è fisicamente condivisa. Questa tecnica è molto più rapida della precedente e non spreca memoria. I processi rimangono comunque logicamente separati e protetti dal SO. Pur essendo un sistema semplice e veloce, si richiede al programmatore di utilizzare i meccanismi di sincronizzazione poiché i processi possono accedere contemporaneamente agli stessi dati, potenzialmente in conflitto.

Condivisione di buffer

Nella condivisione di buffer ciò che cambia rispetto a prima è l'utilizzo di un buffer. La comunicazione rimane diretta, ma il processo mittente scriverà stavolta le sue informazioni all'interno del buffer dal quale poi il processo ricevente andrà a recuperarle. I buffer sono significativamente più piccoli delle porzioni di memoria centrale condivisa, quindi le operazioni di copiatura sono molto più veloci.

Scambio di messaggi

La comunicazione con scambio di messaggi è diretta e prevede che l'informazione viaggia all'interno di messaggi scambiati dal SO.

Sono di lunghezza fissa o variabile e contengono l'informazione da inviare, l'id del mittente e del destinatario ed eventuali informazioni di gestione. I messaggi sono memorizzati in buffer forniti dal SO ad ogni coppia di processi comunicanti, oppure di uso generale tra tutti i processi. È il SO a gestirli, non il programmatore, e non c'è memoria condivisa.

La quantità di buffer può essere:

- **Illimitata**, P può depositare, teoricamente, infiniti messaggi. In seguito Q potrà riceverli in un qualsiasi momento. Si parla di **comunicazione asincrona**.
- **Limitata**, P può depositare messaggi finché ci sono buffer liberi, poi si blocca finché Q non ne libera alcuni. Si parla di **comunicazione bufferizzata**.
- **Nulla**, P non può depositare nessun messaggio, e rimane bloccato finché Q non è pronto a ricevere. Si parla di **comunicazione sincrona.**

Le funzioni messe a disposizione per la gestione dei messaggi sono :

- **Send**(). Invia un messaggio al processo specificato, depositandolo in un buffer libero. Se non ve ne sono, P si blocca finchè non può inviare
- **Receive**(). Riceve un messaggio da un processo specificato (caso simmetrico), oppure da un qualsiasi processo che abbia inviato un messaggio a Q (caso asimmetrico). Se non vi sono messaggi ricevibili, Q si blocca finché non ve ne sono.

Esistono inoltre le versioni condizionali, che invece di bloccare il processo in attesa, ritornano un codice di errore.

- **Cond_Send**(). Se vi sono buffer liberi nel processo destinatario allora il processo P invia i dati, se invece non vi sono buffer liberi in cui scrivere viene ritornata una condizione di errore non bloccando il mittente. La coda dei messaggi in attesa per un processo può essere schedulata.
- **Cond_Receive**(). Se vi sono messaggi ricevibili li riceve altrimenti segnala una condizione di errore senza bloccare il destinatario.

Mailbox

La comunicazione con mailbox è un sistema indiretto, in cui non c'è conoscenza esplicita tra i processi che comunicano. Una mailbox è una struttura dati del SO in cui i processi prelevano/depositano messaggi, identificata da un id univoco. I messaggi nella mailbox possono essere schedulati.

La capacità della mailbox può essere illimitata, limitata o nulla. I messaggi sono di dimensione fissa o variabile e contengono l'id del mittente e mailbox destinataria, l'informazione da inviare ed eventuali informazioni di gestione.

In generale la mailbox non è proprietà del processo, ma del sistema operativo . Il SO fornisce le seguenti funzioni:

- **Create()**, crea una mailbox
- **Send()** e **cond_send()**: deposita un messaggio nella mailbox specificata
- **Receive()** e **cond_receive()**: riceve un messaggio dalla mailbox specificata
- **Delete()** cancella una mailbox

Politiche di sincronizzazione delle mailbox dipendono dalla loro capacità:

- **Illimitata**. Ho una comunicazione asincrona, ovvero il mittente depone il suo messaggio indipendentemente dallo stato di computazione del processo ricevente
- **Nulla**. Ho una comunicazione sincrona. Ho infatti un processo mittente che non trova spazio per deporre il suo messaggio, ed è dunque obbligato a mettersi in attesa
- **Limitata**. La comunicazione verrà bufferizzata, ovvero passerà un po' di tempo prima che il messaggio inviato venga ricevuto.

La mailbox si trova nella memoria del SO, che ne gestisce gli accessi sincronizzandoli. Tipicamente è di sua proprietà e permette a tutti i processi di utilizzarla. È però possibile assegnare una mailbox a un processo, il quale potrà decidere i permessi. Quando esso termina, la mailbox può venire deallocata con esso oppure può tornare proprietà del SO.

Le mailbox sono particolarmente utili per:

- Comunicazioni molti a 1: un processo di servizio riceve ed esegue le richieste dei client.
- Comunicazioni 1 a molti: un processo client invia richieste ad un qualsiasi processo di servizio disponibile. Il primo a ricevere una richiesta la esegue
- Comunicazioni molti a molti: più processi client inviano richieste a un qualsiasi processo di servizio disponibile. Il primo a ricevere una richiesta la esegue.

Comunque, ogni comunicazione coinvolge solo 2 processi: P e Q.

Comunicazione tramite file e pipe

La comunicazione mediante file condivisi rappresenta una diretta estensione della comunicazione con le mailbox ed è utile per scambiare grandi quantità di dati: i file sono memorizzati in memoria di massa. La sua gestione sarà al solito demandata al sistema operativo, che garantirà che le operazioni di accesso siano fatte in modo corretto e solo dai processi autorizzati. L'ordinamento dei messaggi dipende dal processo scrivente.

Una pipe può essere vista come un file memorizzato in memoria del SO. Le funzioni per utilizzarle sono le stesse che si usano per i file, e la sincronizzazione avviene allo stesso modo. L'ordinamento dei messaggi è FIFO. E' una struttura dati di tipo FIFO residente in memoria centrale e che condivide coi file molte delle loro funzioni.

Sia nei file che nelle pipe i messaggi sono di lunghezza fissa o variabile e contengono l'id del mittente, l'informazione da trasmettere ed eventuali informazioni di gestione.

Comunicazione tramite socket

I socket sono utilizzati per le comunicazioni in rete (una sorta di generalizzazione in rete delle pipe), ma possono essere usati anche per l'IPC, anche tra processi su macchine separate: 1 o più mittenti si collegano al ricevente specificando l'indirizzo e la porta su cui è in ascolto. Una volta connessi, avviene la comunicazione.

In sostanza ho il sistema operativo che virtualizza la comunicazione tra macchine diverse utilizzando una pipe spezzata in due porzioni, ognuna residente nella memoria centrale delle macchine coinvolte. L'architettura che sta dietro a tale modalità di comunicazione è di tipo client-server. I socket si possono creare o distruggere, e si può leggerci o scriverci sopra tramite due canali mono-direzionali.

I messaggi sono di dimensione fissa o variabile e contengono solo l'informazione da scambiare. Sono in ordine FIFO.