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.
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

Si distingue tra:

- 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

Metodi per la comunicazione diretta:
Memoria condivisa:
E' possibile condividere un'area di memoria tra i processi comunicanti (variabili globali o buffer per comunicazioni).
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.
Scambio di messaggi:
L'informazione viaggia all'interno di messaggi scambiati dal SO. Sono di lunghezza fissa o variabile e contengono l'informazione da inviare, l'id di mittente e 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. Vi è quindi un rendezvous tra i processi comunicanti. Si parla di comunicazione sincrona

Le funzioni fornite dal SO sono:

- Send, che invia un messaggio al processo specificato, depositandolo in un buffer libero. Se non ve ne sono, P si blocca finché non può inviare

- Receive, che 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.
La coda dei messaggi in attesa per un processo può essere schedulata.
Metodi per la comunicazione indiretta:
Mailbox:
Una mailbox è una struttura dati del SO in cui i processi prelevano/depositano messaggi, identificata da un id. La capacità della mailbox può essere illimitata, limitata o nulla. I messaggi sono di dimensione fissa o variabile e contengono l'id di mittente e mailbox destinataria, l'informazione da inviare ed eventuali informazioni di gestione.
Il SO fornisce le seguenti funzioni:

- Create: crea una mailbox
- Send e send condizionale: deposita un messaggio nella mailbox specificata
- Receive e receive condizionale: riceve un messaggio dalla mailbox specificata
- Delete: cancella una mailbox

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 deciderne i permessi. Quando termina, la mailbox può venire dellocata con esso, oppure può tornare proprietà del SO.
I messaggi nella mailbox possono essere schedulati.
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

Ogni comunicazione comunque, coinvolge solo 2 processi: P e Q.
Comunicazione tramite file e pipe:
I file sono utili per scambiare grandi quantità di dati: i messaggi sono depositati e prelevati da un file su memoria di massa. Essendo su memoria di massa, la comunicazione è piuttosto lenta. Gli accessi sono sincronizzati dal file system. 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.
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, ma possono essere usati anche per l'IPC, anche tra processi su macchine separate: 1 o più mittenti si collegano al ricevente specificandone l'indirizzo e la porta su cui è in ascolto. Una volta connessi, avviene la comunicazione.
Un socket fornisce ai processi un canale bidirezionale che può essere visto come 2 code con i versi opposti spezzate al centro, con gli estremi sulle 2 macchine comunicanti.
I messaggi sono di dimensione fissa o variabile e contengono solo l'informazione da scambiare. Sono in ordine FIFO.