Panoramica Introduttiva su Inter-Process Communication (IPC)

Inter-Process Communication: panoramica (1)

Dei processi si dicono **cooperanti** se si influenzano l'un con l'altro. Per potersi influenzare dei processi necessitano di scambiarsi dati o sincronizzarsi. Per potersi scambiare queste informazioni, i processi necessitano di **meccanismi di comunicazione** detti **IPC** (**Inter-Process Communication**).

Esistono due modelli principali di IPC, ed altri modelli intermedi tra i due principali.

Le **IPC a memoria condivisa** (shared memory) forniscono ai processi cooperanti una porzione di memoria in comune in cui possono scrivere e leggere in modo da passarsi dei dati. Un esempio è l'astrazione ***System V shared memory segment***, aderente allo standard POSIX. Occorrerà poi proteggere l'area di memoria condivisa da accessi contemporanei ad eventi di scrittura sulla memoria.

Le **IPC a scambio di messaggi** (message passing) forniscono ai processi la possibilità di inviare e ricevere messaggi con cui scambiarsi i dati di interesse. Esisteranno due funzionalità principali, send e receive, rispettivamente per inviare e ricevere i dati. Tali operazioni possono essere sincrone (bloccanti) o asincrone (non bloccanti). Un esempio è lo standard MPI (message passing interface) che fornisce MPI_Send e MPI_Recv.

- Processi su computer diversi tipicamente fanno uso di IPC a scambio di messaggi. Se però i sistemi operativi dei due computer possiedono un sotto-sistema che implementa una astrazione di memoria condivisa distribuita tra diversi computer, allora su questa memoria condivisa distribuita è possibile realizzare delle IPC a memoria condivisa.

- Le IPC a scambio di messaggio sono efficientemente utilizzate anche tra processi che risiedono su uno stesso computer, in particolare se il processore è dotato di più core.

Inter-Process Communication: panoramica (2)

Dai due modelli principali di IPC discendono alcune categorie di IPC che ereditano aspetti di entrambi i modelli principali.

Alcune IPC (**file**, **pipe**, **named pipe, memory-mapped file**) derivano dal modello a memoria condivisa ma sfruttano, come area di scambio, dei file su disco oppure dei file mappati in memoria.

Altre IPC (**mutex**, **signal** [asynchronous system trap], **semaphore**, **monitor**) si appoggiano alla memoria condivisa per fornire metodi di sincronizzazione e garantire accesso in mutua esclusione alla memoria condivisa.

Altre IPC derivano strettamente dal modello a scambio di messaggi ed implementano stream (flussi di dati) che conservano i confini dei blocchi di dati trasmessi (**message queue**) oppure appaiono come un flusso continuo i cui confini devono essere incapsulati nei dati trasmessi formattandoli opportunamente (**stream-oriented socket**).

Un caso particolare di derivato del modello a scambio di messaggi è **D-Bus**, che implementa un modello publish-subscribe, in cui ogni applicazione si registra ad un servizio centralizzato per chiedere di essere notificato per una determinata categoria di eventi.

Inter-Process Communication: panoramica (3)

Riassumiamo in questa tabella alcune tipologie di Inter-Process Communication, mettendo in evidenza 1) l'aderenza agli standard POSIX per valutarne la portabilità e 2) i nomi di alcune funzioni come spunto per chi volesse approfondire autonomamente.

	POSIX	info	functions/commands
System V Shared Memory Segment	si	shm.h	shmget() mmap()
mutex	si	pthread.h	pthread_mutex_lock() pthread_cond_wait()
semaphore	si	semaphore.h	sem_init() sem_wait() ...
monitor			wait() notify() (java)
signal	si	signal.h	signal() kill() tkill()
file	si	stdio.h	open() read() write()
pipe	si		è la | in bash per collegare stdin con stdout di due processi
named pipe	si	bash	mkfifo() mkfifo
memory-mapped file	si	shm.h	mmap() , FileChannel() (java)
MPI (message passing interface)	si	mpi.h	MPI_Send() MPI_IRecv() MPI_Probe()
message queue	si	sys/msg.h	msgget() msgctl() msgsnd() msgrcv()
stream-oriented socket	si	sys/socket.h	AF_INET socket() connect() send() recv()
stream-oriented Local socket	si		AF_UNIX socket() connect() send() recv()
D-Bus			dbus_bus_get() dbus_bus_request_name() dbus_connection_send() dbus_connection_pop_message()

Inter-Process Communication: panoramica (4)

Nel seguito delle lezioni vedremo dettagli su alcune IPC.

In particolare vedremo le IPC per **condividere segmenti di memoria** tra processi diversi.

Successivamente vedremo che, usando questi segmenti di memoria condivisa, è possibile **estendere ai processi** i meccanismi di sincronizzazione utilizzati per i Posix Thread, in particolare i meccanismi che fanno uso di mutex e condition variable. In tal modo potremo utilizzare tra i processi le stesse tecniche di sincronizzazione imparate per i thread. Stesso discorso vale per i semafori.

Per quanto riguarda le IPC a scambio di messaggi, in particolare quelle basate sui Socket, le vedrete nei più maniacali dettagli all'interno del corso di Programmazione di Reti.

.