Sistemi Operativi 1

AA 2021/2022

Inter Process Communications

RELAZIONE TRA PROCESSI

UNIVERSITY
OF TRENTO

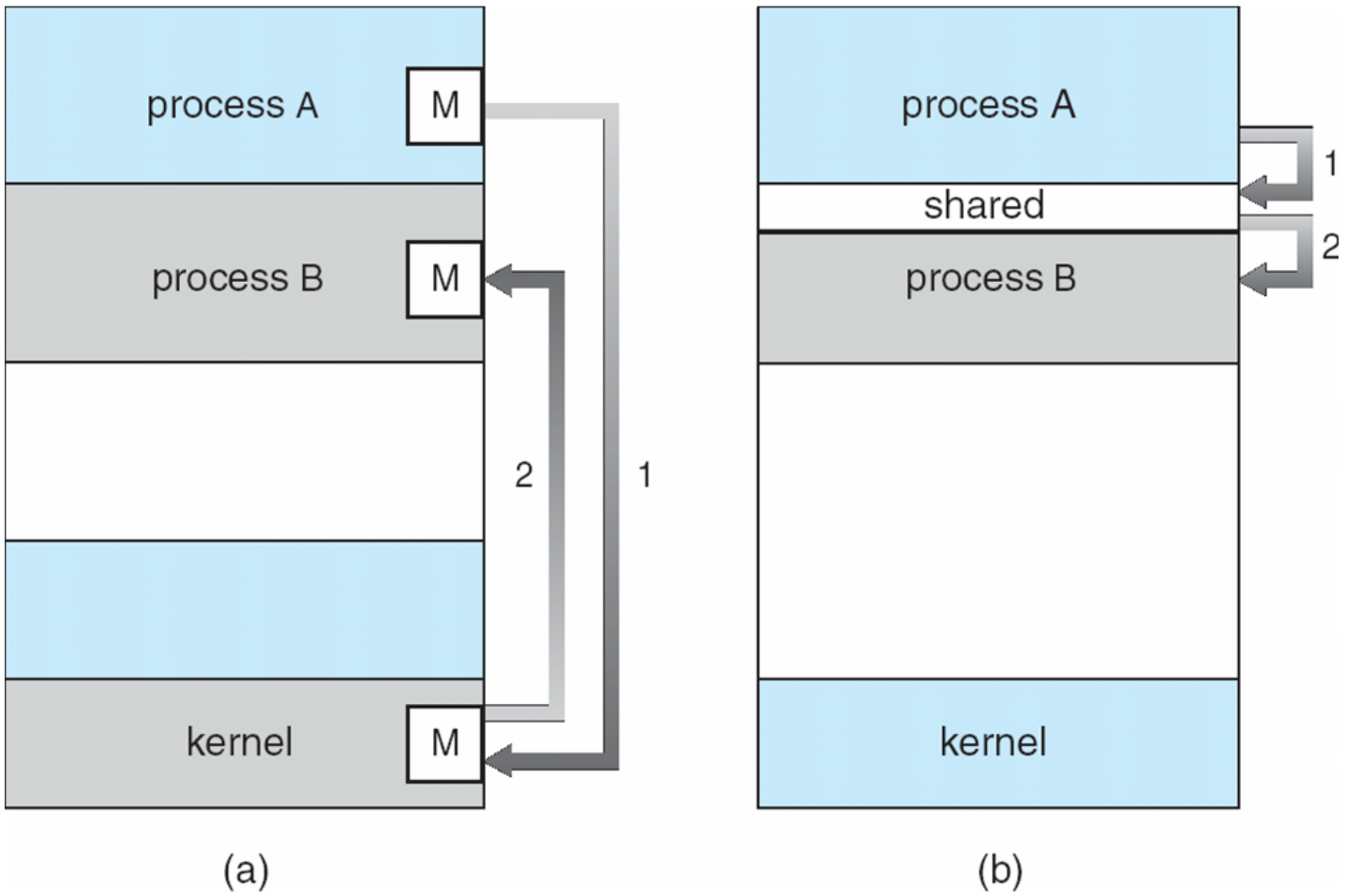
Relazione tra processi

- Processi indipendenti
  - Esecuzione deterministica (dipende solo dal proprio input) e riproducibile
  - Non influenza, né viene influenzato da altri processi
  - Nessuna condivisione dei dati con altri processi
- Processi cooperanti
  - Influenza e può essere influenzato da altri processi
  - Esecuzione non deterministica e non riproducibile

Processi cooperanti

- Motivi
  - Condivisione informazioni
  - Accelerazione del calcolo
    - Esecuzione parallela di “subtask” su multiprocessore
  - Modularità
    - Funzioni distinte su vari processi
  - Convenienza

Modelli di comunicazione



Modelli di comunicazione tra processi basati su (a) scambio di messaggi, e (b) condivisione della memoria.

IPC – MESSAGE PASSING

UNIVERSITY
OF TRENTO

IPC – message passing

- Meccanismi utilizzati dai processi per comunicare e sincronizzare le loro azioni
- Scambio di Messaggi– i processi comunicano tra loro senza condividere variabili
- Le IPC forniscono due operazioni:
  - **send**(*message*) – la lunghezza del messaggio può essere fissa o variabile
  - **receive**(*message*)
- Se *P* e *Q* desiderano comunicare, devono:
  - Stabilire un *canale di comunicazione*
  - Scambiarsi messaggi via send/receive
- Implementazione del canale di comunicazione
  - fisico (e.g., shared memory, hardware bus)
  - logico (e.g., proprieta logiche)

Decisioni implementative

- Come vengono stabiliti i canali?
- Può un canale essere associato a piú processi?
- Quanti canali ci possono essere per ogni coppia di processi comunicanti?
- Qual e' la capacità di un canale?
- La lunghezza dei messaggi che viaggiano nel canale hanno lunghezza fissa o variabile?
- Il canale e uni-direzionale o bi-direzionale?

Nominazione

- Varianti
  - Comunicazione DIRETTA
  - Comunicazione INDIRETTA

Comunicazioni Dirette

- I processi devono nominarsi esplicitamente
- Simmetrica
  - send (P1, message)
  - receive (P2, message)
- Asimmetrica
  - send (P1, message)
  - receive (id, message)
- Svantaggio
  - Se un processo cambia nome… devo ri-codificare gli altri

Invia il mgs a P1

Riceve in *message* un messaggio da P2

Riceve messaggi da tutti e in id si trova il nome del processo che ha eseguito send

"All problems in computer science can be solved by another level of indirection" (David Wheeler…one of the inventor of EDSAC)

Comunicazioni indirette

- I messaggi sono spediti e ricevuti da mailboxes (anche riferiti come *porte*)
  - Ogni mailbox ha un unico id
  - I processi possono comunicare solo se condividono una mailbox

Comunicazioni indirette

- Operazioni
  - creare una mailbox nuova
  - send and receive messaggi tramite mailbox
  - eliminare una mailbox
- Primitive sono definite come:
  - **send**(*A, message*) – spedire un messaggio alla mailbox A
  - **receive**(*A, message*) – ricevere un messaggio dalla mailbox A

Comunicazioni indirette

- Proprietá di un canale di comunicazione
  - Canale stabilito solo se i processi condividono una mailbox comune
  - Un canale può essere associato con molti processi
  - Ogni coppia di processi puo' condividere molti canali di comunicazione
  - I canali possono essere unidirezionali o bi-direzionali

Comunicazioni indirette

- Condivisione di mailbox
  - P_1, P_2, e P_3 condividono la mailbox A
  - P_1, spedisce; P_2 e P_3 ricevono
  - Chi ottiene il messaggio?
- Soluzioni
  - Permettere ad un canale che sia associato con al più due processi
  - Permettere a solo un processo alla volta di eseguire l'operazione *receive*
  - Permettere al sistema di selezionare in modo arbitrario il ricevente. Il mittente è notificato di chi ha ricevuto il messaggio

Sincronizzazione

- Lo scambio di messaggi può essere bloccante o non-bloccante
- Bloccante (sincrono)
  - **Send bloccante** il mittente si blocca finche i messaggio è ricevuto
  - **Receive bloccante** il ricevente è bloccato finchè il messaggio è disponibile

- Non-bloccante (asincrono)
  - **Send non-bloccante** il mittente spedisce il messaggio e continua
  - **Receive non-bloccante** il ricevente riceve un messaggio valido o nulla

IPC – MEMORIA CONDIVISA

UNIVERSITY
OF TRENTO

IPC – memoria condivisa

- Esempio POSIX
  - Processo prima crea il segmento di memoria condivisa

```
segment id = shmget(IPC_PRIVATE, size, S_IRUSR | S_IWUSR);
```

  - Il processo che vuole accedere alla memoria condivisa deve attacarsi a questa

```
shared memory = (char *) shmat(id, NULL, 0);
```

IPC – memoria condivisa

- Ora il processo puo scrivere nel segmento condiviso

```
sprintf(shared memory, "Writing to shared memory");
```

- Quando finito il processo *stacca* il segmento di memoria dal proprio spazio di indirizzi

```
shmdt(shared memory);
```

- To remove the shared memory segment

```
shmctl(shm_id, IPC_RMID, NULL);
```

Pipe

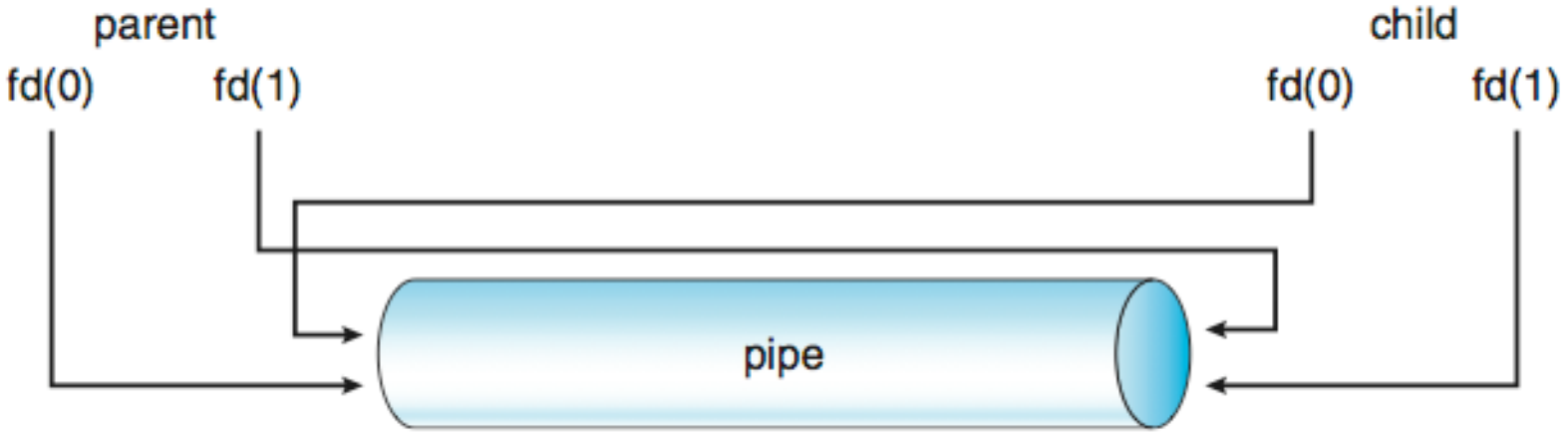
- Agiscono come *condotte* che permettono a 2 processi di comunicare
- Problemi
  - La comunicazione è uni o bidirezionale?
  - Nel caso di comunicazioni a 2 vie, si ha half o full duplex?
  - Deve esistere una relazione tra i processi che comunicano (i.e. parent-child)?
  - Possono essere usate attraverso una rete?

Pipe Ordinarie

- **Pipe Ordinarie** permettono la comunicazione in un stile standard produttore-consumatore
  - Il produttore scrive ad un estremitá (la *write-end* della pipe)
  - Il consumatore legge all'altra estremitá (la *read-end* della pipe)

- Le pipe ordinarie sono quindi unidirezionali
- Richiedono una relazione parent-child tra i processi comunicanti

Pipe Ordinarie



Pipe con Nome

- Le *pipe con nome* sono piú potenti delle pipe ordinarie
- Le comunicazioni sono bidirezionali
- Non è richiesta la relazione parent-child tra i processi comunicanti
- Piú processi possono usare la stessa pipe per comunicare
- Disponibili sia in sistemi UNIX sia in sistemi MS Windows