SISTEMAS OPERATIVOS

Comunicação entre processos

António Godinho

COMUNICAÇÃO ENTRE PROCESSOS

Os processos em execução concorrente num SO podem ser independentes ou cooperantes:

- Processos independentes: não afecta ou não é afectado por qualquer outro processo em execução no SO; não partilha dados com qualquer outro processo.
- Processos cooperantes: pode afectar ou ser afectado pela execução de outros processos; partilham dados entre si.

Vantagens na cooperação entre processos:

- Partilha de informação;
- Aumento do desempenho;
- Modularidade;
- Conveniência;

Os processos cooperantes requerem um mecanismo de comunicação entre processos (interprocess communication - IPC).

IPC - INTER-PROCESS COMMUNICATION

Os processos precisam comunicar entre si em muitas situações. A comunicação entre processos, ou IPC, é um mecanismo que permite que os processos se comuniquem:

- IPC ajuda os processos a sincronizar as suas atividades, partilhar informações e evitar conflitos ao aceder a recursos partilhados.
- Existem dois métodos de IPC: memória partilhada e passagem de mensagens. Um sistema operativo pode implementar ambos os métodos de comunicação.

MODELOS FUNDAMENTAIS DOS MECANISMOS IPC

Os processos precisam comunicar entre si em muitas situações. A comunicação entre processos, ou IPC, é um mecanismo que permite que os processos se comuniquem:

- IPC ajuda os processos a sincronizar as suas atividades, partilhar informações e evitar conflitos ao aceder a recursos partilhados.

Modelos fundamentais dos mecanismos IPC:

- Memória partilhada;
- Passagem de mensagens.

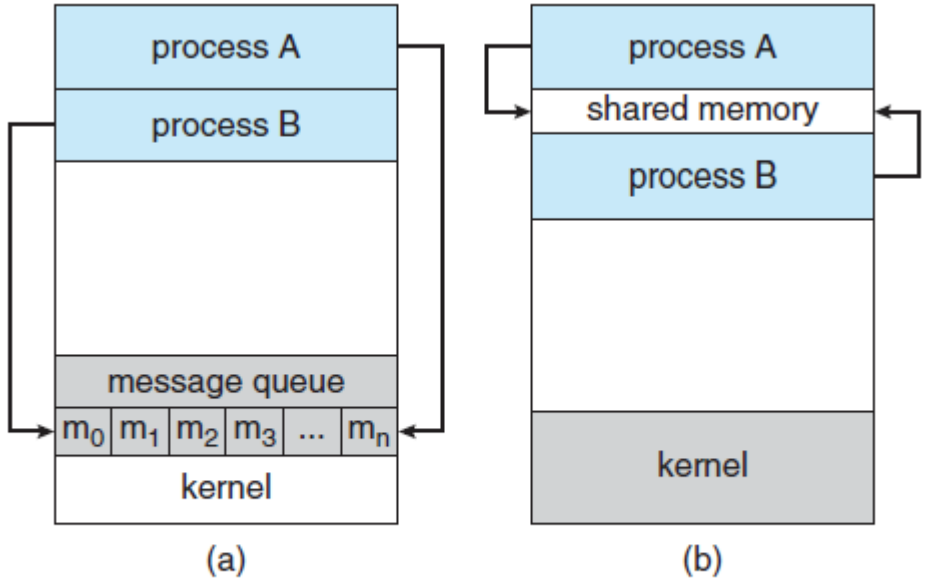


Figure 3.12 Communications models. (a) Message passing. (b) Shared memory.

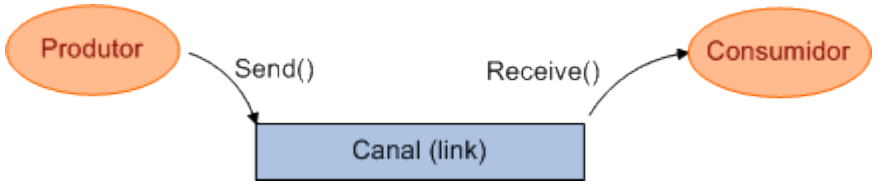
Fonte da imagem: [Silberschatz2014]

MEMÓRIA PARTILHADA

- Permite que dois ou mais processos tenham acesso ao mesmo segmento de memória. As modificações efetuadas por um dos processos são visíveis pelos restantes.
- Forma de IPC mais rápida.
- O acesso à memória partilhada (leitura ou escrita) não requer qualquer chamada ao SO.
- Não requer qualquer duplicação de dados em memória (comparativamente à passagem de mensagens).
- O SO não disponibiliza qualquer mecanismo de sincronização entre processos. Por forma a evitar as condições de corrida (acesso simultâneo à memória partilhada), o programador deverá utilizar mecanismos de sincronização, por exemplo, recorrendo a semáforos.
- Mecanismo de comunicação bidirecional entre quaisquer número de processos.

PASSAGEM DE MENSAGENS

- Permite implementar simultaneamente a sincronização e comunicação entre processos.
- A comunicação entre processos pode ser sempre reduzida à interacção entre um produtor e um consumidor de informação. No caso de a comunicação entre processos ser feita através de mensagens, o produtor é o remetente ou emissor da mensagem e o consumidor é o destinatário ou receptor da mensagem.
- Um canal de comunicação (link) é estabelecido entre o emissor e o receptor e acedido por duas operações elementares: send() e receive().



- Implementação lógica da ligação e das operações send()/receive():
  - Comunicação direta ou indireta;
  - Comunicação síncrona ou assíncrona;
  - Capacidade da ligação

PASSAGEM DE MENSAGENS: COMUNICAÇÃO DIRECTA

- Comunicação directa: cada processo deve explicitamente indicar o emissor ou o receptor da mensagem.
- Primitivas de comunicação por endereçamento simétrico:
  - send (P, mensagem) - envio de mensagem para o processo P
  - receive (Q, mensagem) - recepção de mensagem do processo Q
- Primitivas de comunicação por endereçamento assimétrico:
  - send (P, mensagem) - envio de mensagem para o processo P
  - receive (id, mensagem) - recepção de mensagem de qualquer processo (id é actualizado com o identificador do processo que enviou a mensagem)
- Propriedades:
  - Uma ligação é automaticamente associada entre qualquer par de processos que pretendem comunicar.
  - Uma ligação é associada entre apenas dois processos.
  - Entre cada par de processos, existe apenas uma ligação.

PASSAGEM DE MENSAGENS: COMUNICAÇÃO INDIRECTA

- Comunicação indireta: as mensagens são enviadas e recebidas através de caixas de correio (mailboxes) residentes no núcleo do SO (implementadas através de filas de mensagens).
- Primitivas de comunicação indireta:
  - send (A, mensagem) - envio de mensagem para a mailbox A
  - receive (A, mensagem) - receção de mensagem da mailbox A
- Dentro da mailbox, as mensagens são acedidas por ordem de chegada (FIFO) ou de acordo com um tipo ou uma prioridade associada à mensagem.
- Propriedades:
  - Uma ligação é estabelecida entre um par de processos apenas se partilharem uma mailbox.
  - Uma ligação pode ser associada entre mais do que dois processos.
  - Entre cada par de processos comunicantes, podem existir diferentes ligações, em que cada uma delas corresponde a uma mailbox.

PASSAGEM DE MENSAGENS: SINCRONISMO

- Para a implementação lógica das primitivas send()/receive() é possível considerar:
  - envio bloqueante: o processo emissor fica bloqueado até que a mensagem seja recebida pelo processo receptor ou pela mailbox.
  - envio não bloqueante: o processo emissor envia a mensagem e não fica bloqueado, continuando a sua execução.
  - recepção bloqueante: o processo receptor fica bloqueado até que a mensagem esteja disponível.
  - recepção não bloqueante: aquando da invocação da primitiva receive(), o processo receptor recebe uma mensagem válida ou a indicação de que não existe mensagem disponível.

- Modelos de comunicação:
  - Comunicação síncrona: envio e recepção bloqueante (rendez-vous). O emissor fica bloqueado até que a mensagem seja recebida.
  - Comunicação assíncrona: envio não bloqueante. O emissor envia a mensagem e continua a execução.
  - Cliente-Servidor: o emissor fica bloqueado até o receptor lhe responder (ex. modelo pedido-resposta).

PASSAGEM DE MENSAGENS: CAPACIDADE DA LIGAÇÃO

- Uma mailbox tem uma capacidade de armazenamento que determina o número de mensagens que pode comportar temporariamente:
  - Capacidade zero: não pode haver mensagens armazenadas. O emissor tem de esperar pela disponibilidade do receptor. Os processos têm de sincronizar em qualquer troca de mensagens (rendez-vous).
  - Capacidade limitada: se a mailbox estiver cheia, o emissor pode ficar bloqueado. O sucesso do envio não implica o sucesso da recepção.
  - Capacidade ilimitada: O emissor nunca é bloqueado. O sucesso do envio não implica o sucesso da recepção.

MECANISMOS IPC EM LINUX

- Memória partilhada: mecanismo IPC mais rápido; envio e recepção de informação realizado por operações de escrita e leitura em memória; não oferece mecanismos de sincronização entre processos.
- Semáforos: sincronização de processos (exclusão mútua, cooperação entre processos, barreiras de sincronização, competição por recursos); comunicação indirecta.
  - O Apache utiliza semáforos para lidar com processos de trabalho e gerir a memória partilhada.
  - Ao usar o Prefork MPM (Multi-Processing Module), os semáforos impedem que vários processos modifiquem dados partilhados simultaneamente.
- Sinais: envio e recepção de sinais (sem qualquer dados associados); usados na sincronização de processos; comunicação directa por endereçamento assimétrico.
  - Systemd (Gestor de Serviços) utiliza os seguintes sinais utilizados: SIGTERM, SIGKILL, SIGHUPO
  - systemd envia SIGTERM para parar serviços graciosamente.
  - Se um processo não termina, ele envia SIGKILL para forçar.

MECANISMOS IPC EM LINUX

- Pipes unidireccionais: mecanismo de comunicação unidireccional entre processos hierarquicamente relacionados; usado na comunicação entre processo pai e processos filhos ou tarefas do mesmo processo; ligação com capacidade limitada; envio e recepção bloqueantes; sincronização automática entre processos comunicantes (produtor ➔ consumidor).
- Pipes nomeados (FIFO): idêntico a pipes unidireccionais mas utiliza um ficheiro como ligação; permite a comunicação entre processos não necessariamente relacionados. Ex: logs em Linux
- Filas de mensagens: implementa um mecanismo de comunicação baseado em caixas de correio (mailboxes); mensagens de tamanho fixo e com estrura definida; primitivas de envio e recepção de mensagens em modo bloqueante ou não bloqueante, isto é, comunicação síncrona ou assíncrona.
  - Ao contrário da memória partilhada, em que os processos manipulam diretamente a memória, o SysV MQ permite que os processos enviem e recebam mensagens estruturadas.
  - Em Linux: System V Message Queues (msgget, msgsnd, msgrcv)

MECANISMOS IPC EM LINUX

- Sockets: implementa um mecanismo bidireccional de comunicação entre processos residentes no mesmo equipamento ou residentes em equipamentos interligados por uma rede de comunicação de dados; constituem o componente básico da comunicação entre

```
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
unix  2      [ ACC ]     STREAM     LISTENING     1984605  327530/systemd       /run/user/0/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     1984613  327530/systemd       /run/user/0/bus
unix  2      [ ACC ]     STREAM     LISTENING     1984615  327530/systemd       /run/user/0/gnupg/S.dirmngr
unix  2      [ ACC ]     STREAM     LISTENING     1984617  327530/systemd       /run/user/0/gnupg/S.gpg-agent.browser
unix  2      [ ACC ]     STREAM     LISTENING     1984619  327530/systemd       /run/user/0/gnupg/S.gpg-agent.extra
unix  2      [ ACC ]     STREAM     LISTENING     1984621  327530/systemd       /run/user/0/gnupg/S.gpg-agent.ssh
unix  2      [ ACC ]     STREAM     LISTENING     1984623  327530/systemd       /run/user/0/gnupg/S.gpg-agent
unix  2      [ ACC ]     STREAM     LISTENING     1984625  327530/systemd       /run/user/0/pk-debconf-socket
unix  2      [ ACC ]     STREAM     LISTENING     11162    674/tmux             /tmp/tmux-1001/default
unix  2      [ ACC ]     STREAM     LISTENING     17478    1/init               /run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     17480    1/init               /run/pcscd/pcscd.comm
unix  2      [ ACC ]     STREAM     LISTENING     13770    1/init               /run/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     22914    631/php-fpm: master  /run/php/php8.2-fpm.sock
unix  2      [ ACC ]     STREAM     LISTENING     13772    1/init               /run/systemd/userdb/io.systemd.DynamicUser
unix  2      [ ACC ]     STREAM     LISTENING     13773    1/init               /run/systemd/io.system.ManagedOOM
unix  2      [ ACC ]     STREAM     LISTENING     22920    756/mariadbd         /run/mysqld/mysqld.sock
unix  2      [ ACC ]     STREAM     LISTENING     13786    1/init               /run/systemd/fsck.progress
unix  2      [ ACC ]     STREAM     LISTENING     13794    1/init               /run/systemd/journal/stdout
unix  2      [ ACC ]     SEQPACKET  LISTENING     13796    1/init               /run/udev/control
unix  2      [ ACC ]     STREAM     LISTENING     11432    360/systemd-journal  /run/systemd/journal/io.systemd.journal
```