Sistemas Operativos

Comunicación y Sincronización - II

Sistemas Operativos

- ☑ ***Versión: Abril 2017***
- ☑ ***Palabras Claves: Proceso, Comunicación, Mensajes, mailbox, port, send, receive, IPC, Productor, Consumidor***

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) , el de Silberschatz (Operating Systems Concepts)

Comunicación entre procesos: IPC

- ***Es un mecanismo para comunicar y sincronizar procesos.***
- ***Consta de:***
  - Sistema de mensajes
  - Memoria Compartida (*shared memory)*
  - Semaforos

IPC - Mensajes

- ***Provee dos operaciones***
  - *send* y *receive*.
- ***Se establece un link de comunicación entre los procesos que se quieren comunicar.***
- ***Ese link puede ser unidireccional o bidireccional, simétrico o asimétrico.***
- ***La operación send puede ser por copia o por referencia; los mensajes de medida fija o variable.***

IPC - Mensajes(cont.)

- ***Comunicación directa: cada proceso que quiere comunicarse con otro, explícitamente nombre quien recibe o manda la comunicación***

Send (P, mensaje) ***Envía un mensaje al proceso P***

Receive (Q, mensaje) ***Recibe un mensaje desde el proceso Q***

IPC - Mensajes(cont.)

- ***Comunicación Indirecta: usa un mailbox o port.***
- ***Un mailbox puede verse como un objeto donde se ponen y sacan mensajes.***
- ***Cada mailbox tiene una identificación única***

Send (A, mensaje) ***Envía un mensaje al mailbox A***

Receive (A, mensaje) ***Recibe un mensaje desde el mailbox A***

IPC - Mensajes(cont.)

- ***En un esquema de comunicación indirecta el sistema operativo debe proveer un mecanismo para que un proceso pueda:***
- ***Crear un nuevo mailbox***
- ***Compartir un mailbox***
- ***Enviar y recibir mensajes a través del mailbox***
- ***Destruir un mailbox***

IPC - Mensajes(cont.)

Capacidad del Link: ¿Cuántos mensajes puede mantener el link?

- ***Cero: no puede haber mensajes esperando. Es lo que se llama Rendezvous: el emisor debe esperar que el receptor reciba el mensaje para poder mandar otro. Hay sincronismo.***
- ***Capacidad limitada: la cola tiene una longitud finita***
- ***Capacidad ilimitada: tiene una longitud "infinita". El emisor nunca espera.***

Ejemplo: Productor consumidor con mensajes

Productor

```
repeat
...
  produce un ítem en
  nextp
  ...

  send(consumidor,ne
  xtp);
until false;
```

Consumidor

```
repeat

  receive(productor,n
  extc);
    ...
  consume el ítem en
  nextc
  ...
  until false;
```

Naming asimétrico

- ***Send (P,message)*** ***Envía un mensaje a P***

- ***Receive (id, message)*** ***Recibe un mensaje desde cualquier proceso. Id identifica el nombre del proceso con el que se ha establecido la comunicación.***

Naming indirecto: propiedad del mailbox

- ***El mailbox puede ser propiedad del proceso o del sistema.***
- ***Propiedad del proceso: esta definido como parte de el o asignado directamente a el.***
- ***Dueño del mailbox es el que recibe mensajes a traves de él***
- ***Usuario es quien envía los mensajes a ese mailbox.***

Caracteristicas de IPC

- ***Emisor y receptor pueden ser bloqueantes o no bloqueantes.***
- ***Caso receptor:***

1. ***Si el mensaje ya se mandó, lo recibe.***
2. ***Si no hay mensajes: o se bloquea o continua sin recepción***

Combinaciones típicas

- ***Envio bloqueante, recepcion bloqueante (rendezvous)***
- ***Envío no bloqueante, recepción bloqueante (la más utilizada)***
- ***Envío no bloqueante, recepción no bloqueante***

IPC – System V

- ***Cada objeto IPC tiene un ID único***
  - Objeto: Un mensaje, Un Semáforo, Un segmento de memoria compartido
- ***Para trabajar con IPC siempre debemos obtener el ID único***
  - Todos los procesos que se quieren comunicar, lo compartirán.
- ***Se utiliza la función ftok() para obtener un ID***

IPC – System V (cont.)

ftok

```
key_t ftok ( char *nombre, char proj );
```

Retorna:

- nueva clave IPC si tuvo éxito.
- -1 si no tuvo éxito, dejando errno con el valor de la llamada stat()

```
key_t miclave;
miclave = ftok("/tmp/miaplic", 'a');
```

IPC – Mensajes - SysCalls

- ***msgget***

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgget (key_t clave, int msgflag);
```

- ***Permite obtener/crear una cola de mensajes***
- ***Flags:***
  - IPC_CREATE: Crearlo si no existe
  - IPC_EXCL: Falla si al crear uno, ya existe
  - MODOS: RWX (Owner, Group, Others)

IPC – Mensajes - Estructura

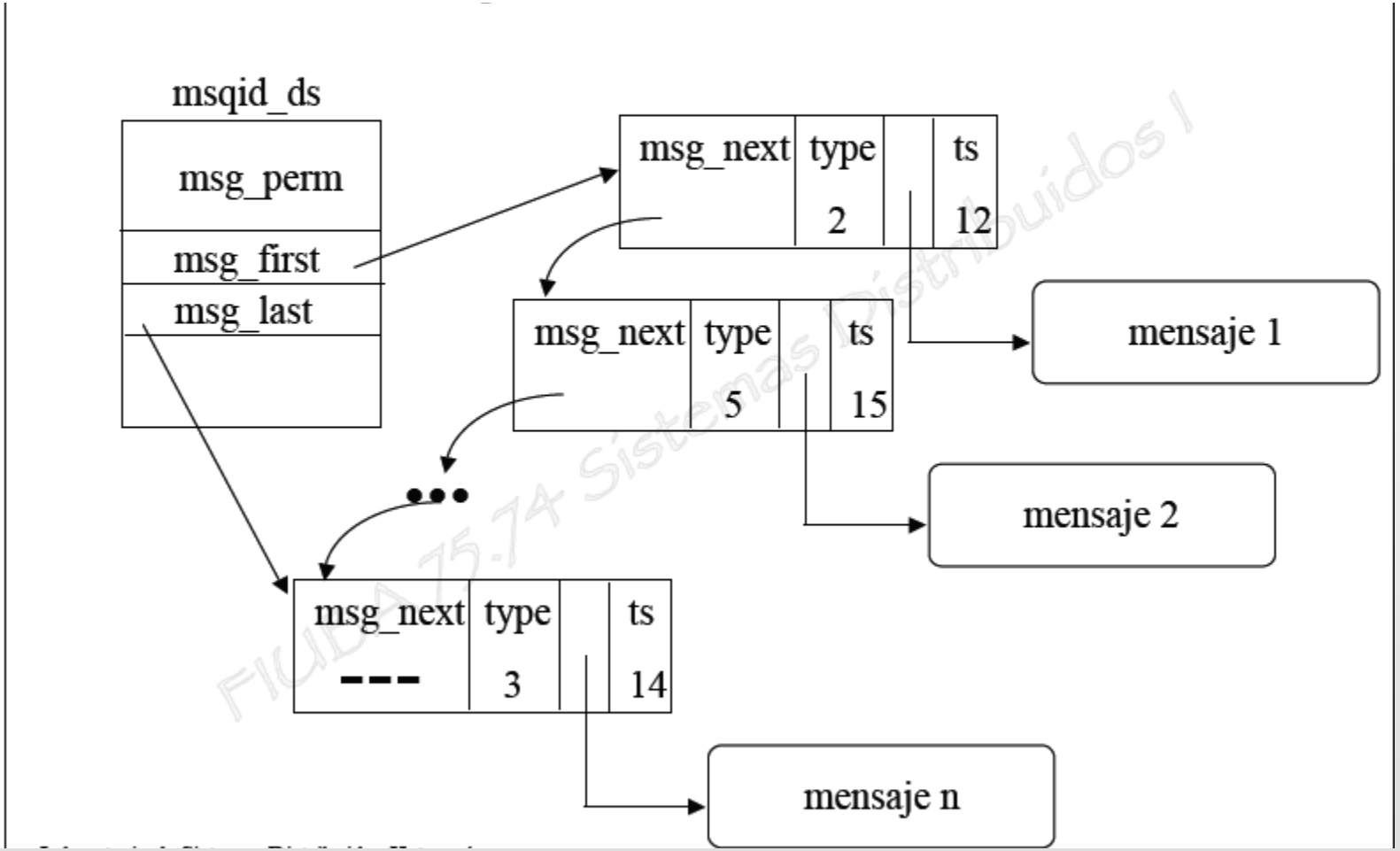
```
struct msqid_ds (definida en <sys/msg.h>
{
    struct ipc_perm msg_perm; /* permisos para operar */
    struct msg *msg_first; /* ptr. al 1° mensaje en cola */
    struct msg *msg_last; /* ptr. al ultimo mensaje en cola */
    time_t msg_stime; /* hora del último msgsnd */
    time_t msg_rtime; /* hora del último msgrcv */
    time_t msg_ctime; /* hora del último cambio */
    struct wait_queue *wwait;
    struct wait_queue *rwait;
    ushort msg_cbytes; /* #bytes actual en cola */
    ushort msg_qnum; /* # de mensajes en cola */
    ushort msg_lspid; /* pid del último msgsnd */
    ushort msg_lrpid; /* pid del último msgrcv */
}
```

IPC – Mensajes – Estructura (cont.)

```
struct wait_queue
{
    struct task_struct * task;
    struct wait_queue * next;
}

struct msg /* una por cada mensaje */
{ /* en msg.h */
    struct msg *msg_next; /* prox. mensaje en la cola */
    long msg_type;
    char *msg_spot; /* dirección texto del mensaje */
    short msg_ts; /* tamaño del texto */
}
```

IPC – Mensajes – Estructura (cont.)



IPC – Mensajes - SysCalls

- ***msgctl***

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgctl (int cmsgid, int cmd, [struct msqid_ds *buf]);
```

- ***Permite modificar una cola de mensajes***
- ***cmd:***
  - IPC_STATE: Obtener información
  - IPC_SET: Modificar
  - IPC_RMID: Eliminar

IPC – Mensajes - SysCalls

msgsnd

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgsnd (int cmsgid, const void *msg, size_t tam, int msgflag);
```

Permite enviar un mensaje

msgflag:

- 0: Si la cola esta llena, el proceso se bloquea hasta que haya espacio.
- IPC_NOWAIT: devuelve el control retornando error -1 y errno=EAGAIN

IPC – Mensajes - SysCalls

msgrcv

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgrcv (int cmsgid, void *msg, size_t tam,
            long msgtype, int msgflag);
```

- **Permite recibir un mensaje**
- **msgflag:**
  - 0: Se bloquea hasta que haya un mensaje en la cola
  - IPC_NOWAIT: devuelve el control retornando error -1 y errno=ENOMSG
  - MSG_NOERROR: Truncar mensajes a tam

IPC – Mensajes - SysCalls

- ***Permisos requeridos en las colas:***
  - WRITE
    - Para msgsnd
    - Para msgctl (modificación)
  - READ
    - Para msgrcv
    - Para msgctl (Consulta)