Sistemas Operativos En Tiempo Real
Maza Casas Lamberto
3MV7

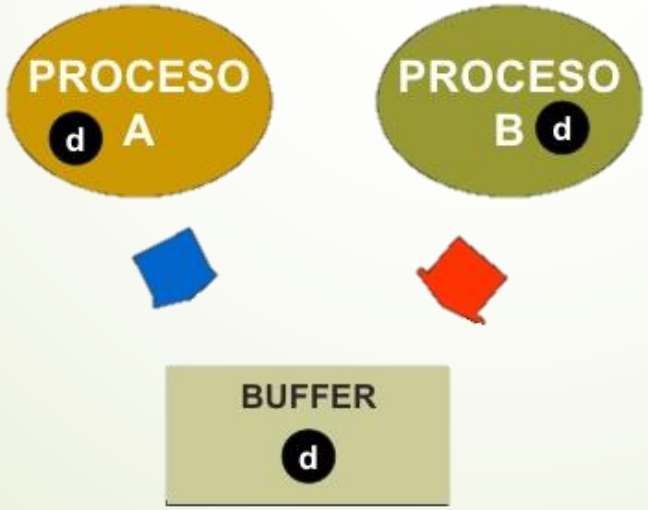
Mensajes

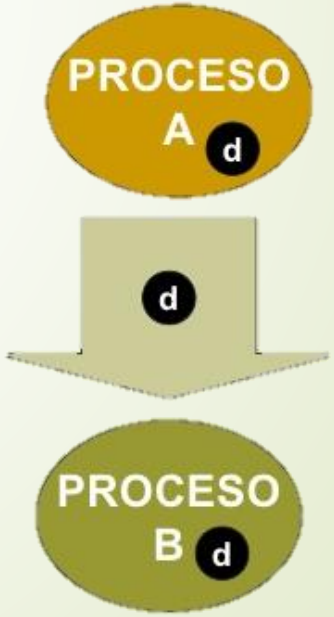
Comunicación de procesos con mensajes

Ángeles Pacheco Jorge Armando
Calva Sánchez Fernando Alonso
Eduardo Sánchez Axl

Comunicación entre procesos (IPC)

- La comunicación entre procesos es una función básica de los sistemas operativos que provee un mecanismo que permite a los procesos comunicarse y sincronizarse entre sí, normalmente a través de un sistema de bajo nivel de paso de mensajes que ofrece la red subyacente. Las técnicas de IPC están divididas dentro de métodos para: paso de mensajes, sincronización, memoria compartida y llamadas de procedimientos remotos





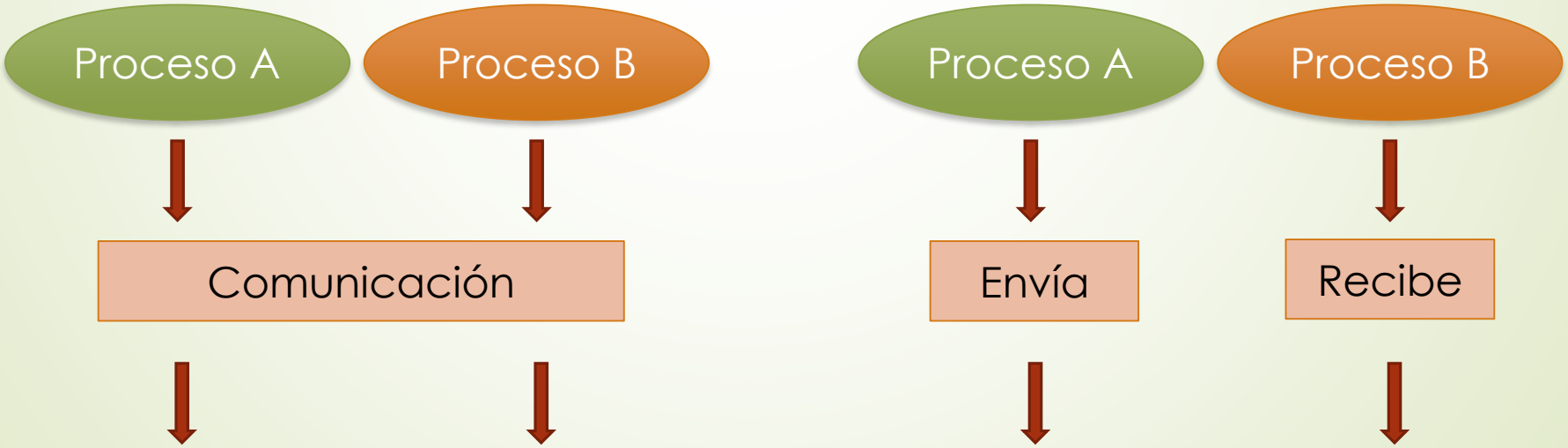
Paso de mensajes

- De forma abstracta, se llama *mensaje* a una porción de información que un proceso emisor envía a un destinatario (El cual puede ser otro proceso, un actor o un objeto). El modelo de paso de mensajes es el que define los métodos y funciones para poder llevar a cabo el envío de un mensaje de un proceso emisor a un destinatario. Supone un enfoque opuesto al paradigma tradicional en el cual los procesos, funciones y subrutinas sólo podían ser llamados directamente a través de su nombre.

Tipos de comunicación por mensaje

Existen dos tipos de comunicación por mensaje

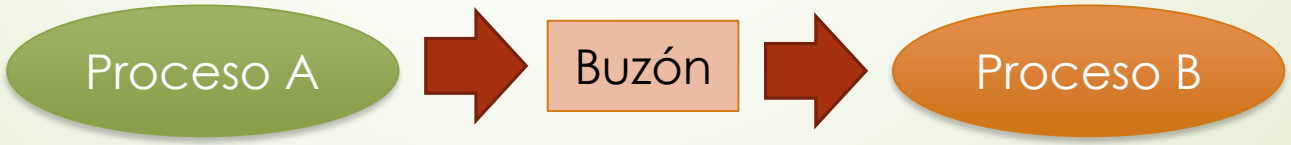
- Síncrona -> Ambos procesos esperan a que la comunicación se realice para continuar su ejecución.
- Asíncrona -> El proceso emisor puede continuar con su ejecución, no en cambio el receptor, quien debe esperar a que le sea enviada la información.



Principio de funcionamiento

- Directa: se establece entre ambos procesos una conexión denominada link. Facilita el envío de información haciendo la comunicación bidireccional, sin embargo limita a que la información sea accesible solo por dos procesos.

- Indirecta: En este tipo de link, la información puede ser recibida por múltiples procesos, sin embargo requiere del uso de "buzones".



Colas de mensaje

Es la capacidad de mensajes que puede almacenar el link.

- Capacidad cero: El link no almacena mensajes (comunicación directa).
- Capacidad limitada: Almacena “n” mensajes en la forma FIFO, es decir, el primer mensaje que entra es el primer mensaje que sale.
- Capacidad infinita: La cola tiene potencialmente longitud infinita.

Funciones principales

- **msgget(key_t key, int msgflg);**

Se encarga de crear la pila de mensajes, regresa -1 si hay un error, o el valor del identificador (msqid) de la pila de mensajes.

- **msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg);**

Envía mensajes a la pila creada anteriormente por msgget

- **msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg);**

Recibe la información enviada por el otro proceso

Ejemplo de aplicación (Proceso A)

```
#include <sys/ipc.h> /*ipc*/
#include <sys/msg.h>        /*msgget*/
#include <sys/types.h>      /*msgget*/
#include <stdlib.h>    /*exit*/
#include <stdio.h>     /*printf*/
#include <string.h>    /*strcpy*/
#define TAM 128
typedef struct msgbuf {
        long  tipo;
        char texto[TAM];
} mensaje;
main(){
        int msgflg = IPC_CREAT | 0666; key_t key; size_t buf_length; int msqid; mensaje sbuf;
        key = 5678;
        printf("Creando cola de mensajes: msgget(%i,%o)\n",key, msgflg);
        if((msqid = msgget(key, msgflg)) < 0){
                perror("msgget");
                exit(1);
        }
        else
                printf("Cola creada con exito: msqid = %d\n",msqid);
        sbuf.tipo = 1 ;
        strcpy(sbuf.texto, "Fernando Idex");//Mensaje a enviar
        buf_length = strlen(sbuf.texto) + 1;
        if (msgsnd(msqid, &sbuf, buf_length, msgflg) < 0){
                printf("%d, %ld, %s, %ld\n", msqid, sbuf.tipo, sbuf.texto, buf_length);
                perror("msgsnd");
                exit(1);
        }
        else
                printf("Mensaje: ' %s ' Enviado\n", sbuf.texto);
        exit(0);
}
```

Ejemplo de aplicación (Proceso B)

```
#include <sys/ipc.h>  /*IPC*/
#include <sys/msg.h> /*msgget*/
#include <sys/types.h>      /*msgget*/
#include <stdlib.h>     /*exit*/
#include <stdio.h>      /*printf*/
#include <string.h>     /*strcpy*/
#define TAM128

typedef struct msgbuf {
        long       tipo;
        char       texto[TAM];
} mensaje;

main(){
        int msqid; key_t key; mensaje rbuf;
        key = 5678;
        if((msqid = msgget(key, 0666)) < 0){
                perror("msgget");
                exit(1);
        }
        if(msgrcv(0, &rbuf, TAM, 1, 0) < 0){
                perror("msgrcv");
                exit(1);
        }
        printf("%s\n", rbuf.texto);
        exit(0);
}
```

¡Gracias por su atención!

Bibliografía:

- http://teoriapa1112.blogspot.mx/2011/10/comunicacion-entre-procesos-ipc.html
- http://www.somoslibres.org/modules.php?name=News&file=article&sid=552
- https://es.wikipedia.org/wiki/Comunicaci%C3%B3n_entre_procesos#Comunicaci.C3.B3n_orientada_a_mensajes
- http://slideplayer.es/slide/5390114/
- http://man7.org/linux/man-pages/man2/msgget.2.html

Repositorios

- Ángeles Pacheco: https://github.com/ArmandoAngeles/SistemasOperativosEnTmpR
- Calva Sánchez: https://github.com/idexc/CalvaSOTR
- Eduardo Sánchez: https://github.com/Slae2924/SOTR