Comunicación entre procesos y sincronización

IPC(Inter Process Communication)—mecanismos estandarizados para que los procesos puedan intercambiar info.

El SSOO debe resolver el problema de sincronización.

Señales(Signals)mecanismo de comunicación asíncrona del Kernel con sus procesos.Parecido a una interrupción de sw.

Memoria compartida—Es un servicio que proporciona el Kernel a los procesos. Uno de ellos debe pedirla de forma explicita.El sistema reserva un rango en RAM que mapea en el mapa de mem virtual. El acceso necesita una clave que el ‘peticionario puede compartir. Si otro proceso quiere acceder el sistema mapea los marcos en la pagina de mem compartida del proceso con el que se comparte. Pueden haber mas de dos procesos compartiendo memoria.

Este proceso no tiene sincronización. El SSOO tiene q llevar cuenta de cuantos procesos están activos compartiendo memoria para no liberar espacio hasta q acaben.

**Pipes:** Los pipes o tuberías son un medio de comunicación secuencial no estructurado, el lado receptor está a la escucha de la salida del emisor. Unidireccional y no se puede parar.

Existen los named pipes o FIFO. Se utilizan parecido a un fichero con path, nombre y comandos de apertura,escritura ,lectura y cierre. Hay q borrarlas de forma explícita. Diferencias con ficheros—no se puede abrir como escritura y lectura a la vez.

**Colas de mensajería** Mecanismo de comunicación asíncrono con envío y recepción no bloqueantes y funcionan con Publisher/subscriber. (System V y POSIX). Se crean y manejan como ficheros especiales con nombre o no. Se instancia como estructura de datos en el Kernel que se encarga de sincronizar los procesos. Se pueden creas solo de lectura, solo de escritura o bidimensionales.

**Socket** Punto lógico de un canal de comunicación entre procesos soportado por una red IP.

Dos tipos—Datagram(UDP) no orientados a sesión----Stream(TCP)orientados a conexión.Bind procedimiento para que un soquet se enganche.

**Concurrencia**—Se pide un recurso limitado por varios procesos.Recursos posibles:Físicos(registro de un controlador) o lógicos(campo de una BBDD)—Reutilizables(la operación no los destruye--ficheros) o consumibles—Exclusivos(temporizador) o compartidos(un fichero)—Expropiables(página de RAM)o no expropiables(mutex).

Proceso cooperante es cuando esta diseñado para colaborar con otros o independiente cuando no. Cooperación implícita(el programador no se entera) o explicita. Operación atómica es una operación que ocurre en un instante.

Sección crítica—cuando se llega a conflicto el primero en entrar termina sus cosas y desbloquea esa parte para que el ultimo termine. Condición de carrera—varios procesos acceden al mismo recurso y el orden de llegada es aleatorio.

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza mediaInterbloqueo—un proceso retiene un recurso y no lo suelta hasta que otro proceso suelte un recurso que necesita pero el proceso 2 necesita el que esta holdeando el proceso 1. Deadlock es cuando esta todo tan jodido q ni patras ni palante.

**Sincronización** Para usar un recurso existe una llave que se deben dar los procesos que terminan de usarlo a los que están primeros en la cola(mecanismo de exclusión mutua).

Problema es que se interrumpa y mate al proceso con la llave y este no la devuelva

**Mutex** solución bloquea el thread para que acabe la sección crítica sin que le puedan interrumpir y a los que loo intente los deja bloqueados por el Kernel. Problema de estampida, cuando se liberan a todos los threrads hay uno que s epone primero en la cola y los demás van detrás retrasando el scheduler anterior.Inversión de prioridad es otro problema.

**SpinLock**--(candado)inhibe el scheduler y parte las interrupciones. Mucho riesgo. Los candados son recomendables para secciones críticas muy breves.

**Semáforos**—Mecanismo para arbitrar el acceso de M peticionarios a N recursos de forma simultanea.

**Candado sobre ficheros**—mecanismo de protección de ficheros