Comunicación entre procesos en Linux (IPC)

En esta clase:

- Mecanismos de Comunicación entre procesos (IPC) en Linux
- Memoria Compartida

Mecanismos IPC

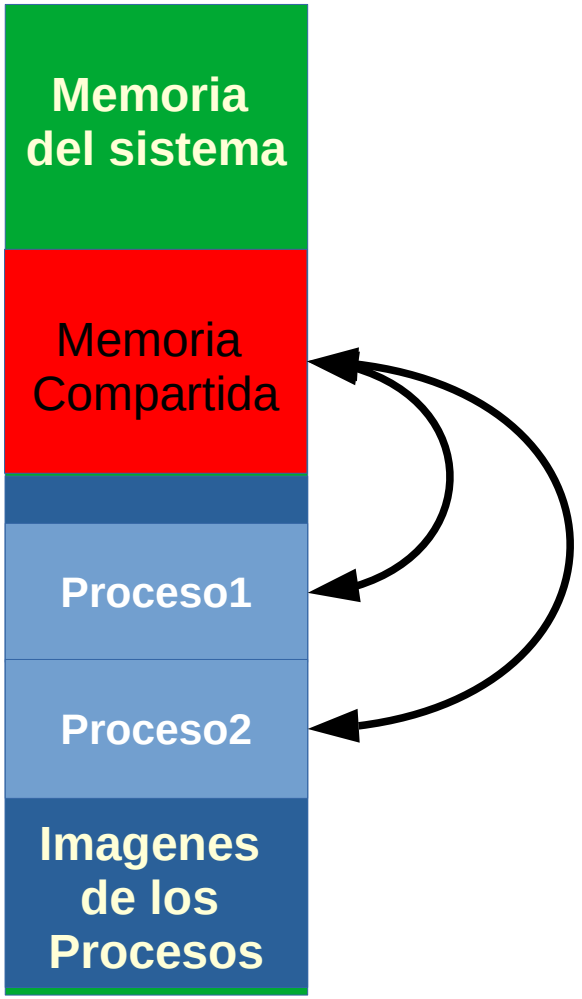
- Dentro de los mecanismos de comunicación entre procesos podemos mencionar:
  - Memoria compartida
  - Semáforos
  - Colas de Mensajes

Memoria Compartida

- La memoria compartida permite que dos procesos intercambien información.
- El Sistema Operativo provee un espacio de memoria destinado a ser compartido entre los procesos.
- Para acceder a este espacio compartido de memoria, los procesos deben realizar un procedimiento basado en llamadas al sistema.

Memoria Compartida

- Ambos procesos deben acordar el espacio de memoria que van a compartir.
- El Sistema Operativo debe verificar que es posible asignar ese espacio de memoria



Memoria Compartida. Obtencion de clave única

Para acordar el recurso compartido se genera una clave (key) que lo identifica.

Se usa la llamada al sistema `ftok()`

```
key_t  ftok (char * nom_arch, int Numero )
```

`nom_arch:` Nombre de archivo, asegurarse que exista, por ejemplo **`"/bin/ls"`**

`Numero:` cualquier entero

Todos los programas que usen el mismo nombre de archivo y el mismo número obtendrán la misma clave

Memoria Compartida. Obtencion de clave única

Ejemplo:

```
#include <sys/shm.h>
#include <stdio.h>
#include <unistd.h>
#define NUM 33
int main(void)
{
key_t  miclave;
miclave=ftok (“/bin/ls”, NUM);
}
```

Memoria Compartida. Creación del espacio de memoria

Usando esta llamada al sistema conseguimos que se asigne un espacio de memoria del tamaño requerido y se asocie con la clave obtenida anteriormente. De esta manera puede ser utilizado por uno o más procesos.

```
int shmget (key_t clave, int tamanio, int flags)
   clave: la obtenida con ftok
   tamanio: en bytes
   flags:
   permisos de escritura/lectura (como en los archivos)
   flags de control (IPC_CREAT)
```

Ejemplo:

```
   int mem_id;
   mem_id=shmget (miclave, 1024, 0777|IPC_CREAT);
```

Memoria Compartida. Asociación de un puntero dentro del proceso

Usando esta llamada al sistema conseguimos que se asocie el espacio de memoria ya obtenido con un puntero que me permita acceder desde el programa.

Pueden usarse conversiones de tipo para facilitar esta asociación.

```
char * shmat (int mem_id, char * addr, int flags)
```

mem_id es el identificador obtenido antes con shmget()

addr permite indicar a que direccion se debe asociar, es mejor pasar un NULL y dejar que el SO asigne solo

flags permite elegir entre R/W o Read Only entre otras cosas, conviene pasar NULL, (R/W)

Ejemplo:

```
int *arreglo
arreglo = (int *) shmat(mem_id, NULL,NULL);
```

Memoria Compartida. Liberación de recursos

Cuando un proceso termina de usar la memoria compartida debe liberarla para que el SO pueda disponer de ella.

Este es un proceso en dos pasos, primero deben desasociarse los punteros que referencian la ubicación de memoria en cada uno de los procesos que la utilizan

```
int shmdt (char * puntero)
```

puntero es el puntero que queremos desasociar en el programa, esta llamada debe hacerse en cada programa que utilice la memoria compartida.

Ejemplo:

```
shmdt((char *)arreglo);
```

Memoria Compartida. Liberación de recursos

En el segundo paso luego que todos los procesos desasociaron los punteros que referencian la ubicación de memoria compartida, debe liberarse, es decir devolverle al sistema operativo el espacio de memoria solicitado.

```
shmctl (int mem_id, int flags, struct shmid_ds * ds)
```

mem_id: es el identificador obtenido antes con shmget()

flags: IPC_RMID para borrarla

ds: puede ser NULL

Ejemplo:

```
shmctl(mem_id,IPC_RMID,NULL)
```

Memoria Compartida. Liberación de recursos

En el proceso de liberación de memoria se utiliza una estructura que nos permite obtener información acerca del estado de la memoria compartida: **`shmid_ds`**

> **`shmid_ds:`** Es una estructura asociada al segmento de memoria compartida que contiene información sobre la misma.

Algunos de sus campos son:

`shm_atime` toma el valor de la hora actual.

`shm_lpid` toma el valor del PID del proceso llamador**.**

`shm_nattch` se incrementa o decrementa en uno cada vez que un proceso asocia o desasocia la memoria respectivamente

struct shmid_ds

```
struct shmid_ds {
  struct ipc_perm shm_perm;     Ownership and permissions
  size_t          shm_segsz;    Size of segment (bytes)
  time_t          shm_atime;    Last attach time
  time_t          shm_dtime;    Last detach time
  time_t          shm_ctime;    Last change time
  pid_t           shm_cpid;     PID of creator
  pid_t           shm_lpid;     PID of last shmat(2)
                                o shmdt(2)
  shmatt_t        shm_nattch;   No. of current attaches
};
```

Comandos para IPC

ipcs:

Muestra los recursos IPC utilizados por el usuario

ipcrm:

Me permite eliminar un recurso IPC.