Memoria Compartida

Por:
Orlando Alemán Ortiz
Samuel Díaz Cabrera

¿Qué veremos?

Introducción

- Comunicación entre procesos
- Comunicación en Unix (IPC)
- Memoria Compartida

Utilización de M.C.

- Llamadas al sistema: shmget, shmctl, shmat y shmdt

Estructuras principales:

- Memoria compartida shmid_ds, shmid_kernel, shm_info, shminfo
- Relacionadas: kern_ipc_perm, ipc_ids

Código

Comunicación entre procesos

Una necesidad que surgió con la aparición de los primeros sistemas multitarea fue que los procesos pudieran comunicarse entre si para:

- Sincronizarse
- Compartir información

Unix, y todas sus variantes, implementan un mecanismo para llevar a cabo esta comunicación denominado IPC.

Comunicación en Unix (IPC)

- ***Inter**Process **C**ommunication* (IPC)
- Conjunto de mecanismos clave en los sistemas Unix para llevar a cabo la compartición de datos y sincronización entre distintos procesos.
- Actualmente en Linux están disponibles 3 mecanismos:
  - Colas de mensajes.
  - Memoria compartida.
  - Semáforos.

Comunicación en Unix (IPC)

- Cola de mensajes.- Similar a un buzón, un proceso deposita un mensaje y otros los leen.
- Semáforo.- Herramienta puramente de sincronización. Controlan el acceso de varios procesos a recursos comunes.
- Memoria Compartida.- Permite establecer una zona común de memoria entre varias aplicaciones.

Comunicación en Unix (IPC)

- Funcionamiento basado en un sistema de claves.
- La creación o utilización de un IPC requiere un identificador denominado *clave*.
- Identifica al IPC de manera única en el sistema.
- Se obtiene mediante la función de biblioteca *ftok*

  *key_t ftok (char *pathname, char proj);*

- Parámetros:
  - *pathname*.- Nombre de un fichero accesible al proceso.
  - *Proj.-* Identificador.

Memoria Compartida

- La memoria que habitualmente puede direccionar un proceso a través de su espacio de direcciones virtuales es local a ese proceso.
- Cualquier intento de direccionar esa memoria desde otro proceso provoca una violación de segmento.
- La memoria compartida permite a dos o más procesos compartir datos a través de un segmento de memoria.

Memoria Compartida

- Para disfrutar de las ventajas que aporta el uso de memoria compartida se dispone de 4 llamadas al sistema:
  - *shmget().- Obtención de la zona de memoria*
  - *shmctl().- Proporciona operaciones para el control de la memoria compartida*
  - *shmat().-Se adjunta la zona de memoria compartida al espacio de direcciones del proceso.*
  - *shmdt().- Desvincula una zona de memoria compartida del proceso.*

Llamadas al Sistema.- Shmget()

- Reserva el espacio necesario para alojar un segmento de memoria compartida y si existe permite acceder a ella.

  int shmget(key_t key, size_t size, int shmflg);

- Parámetros:
  - key.- Una clave.
  - size.- El tamaño del segmento a crear.
  - shmflg.- El flag inicial de operación
- Salida:
  - Devuelve un entero, denominado *shmid* que identifica al Segmento.

Llamadas al Sistema.- Shmget()

- Valores de *shmflg:*
  - IPC_CREAT: Para crear un nuevo segmento. Si este flag no se precisa, *shmget()* buscará el segmento asociado con la clave y comprobará si el usuario tiene permisos para acceder a él.
  - IPC_EXCL: Se utiliza junto a IPC_CREAT para asegurar el fallo si el segmento existe.
  - mode_flags: Los 9 bits menos significativos del número están reservados para el establecimiento de permisos de acceso. Actualmente los permisos de ejecución no son utilizados por el sistema.

Owner			Group			World		
R	W	X	R	W	X	R	W	X

Llamadas al Sistema.- Shmctl()

- Conjunto de operaciones para el control de la memoria compartida.

  int shmctl (int shmid, int cmd, struct shmid_ds *buf);

- Parámetros:
  - shmid.- Identificador de la zona de memoria compartida.
  - cmd.- Operación a realizar.
  - shmid_ds.- Estructura utilizada como buffer de almacenamiento o carga.
- Salida:
  - Devuelve 0 si se lleva a cabo correctamente
  - Devuelve -1 en caso contrario.

Llamadas al Sistema.- Shmctl()

- Operaciones disponibles (valores para cmd):
  - IPC_STAT: Lee el estado de la estructura de control y lo devuelve por *buf*.
  - IPC_SET: Modifica el valor de los campos *shm_perm.uid shm_perm.gid shm_perm.mode* de la estructura de datos asociada a shmid con el valor almacenado en *buf*.
  - IPC_RMID: Marca una región de memoria compartida como destruida. Si el segmento aún estaba asignado a otro proceso, la clave será puesta a IPC_PRIVATE. El borrado se hará efectivo cuando el útimo proceso que la adjuntaba deje de hacerlo.
  - SHM_LOCK: Bloquea la zona de memoria compartida especificada por *shmid*. Su uso está restringido solamente para procesos que tenga un ID de superusuario. El segmento es grabado en memoria, no siendo posible borrarlo de ella a partir de esta operación. Se garantiza la permanencia de los datos en la memoria.
  - SHM_UNLOCK: Desbloquea la región de memoria compartida especificada por *shmid*. Sólo la podrán ejecutar los procesos cuyo usuario tenga Id de Superusuario

Llamadas al Sistema.- Shmat()

- Adjunta una zona de memoria compartida dentro de su espacio de direcciones.

  void* shmat (int shmid, const void *shmaddr, int option);

- Parámetros:
  - shmid.- Identificador del segmento.
  - shmaddr.- Dirección de memoria donde adjuntar el segmento. Si se especifica NULL el sistema operativo buscará una posición de memoria libre.
  - option.- Opciones.
- Salida: Devuelve la dirección de comienzo del segmento de memoria asignado y se actualizan campos de la estructura *shmid_kernel:*
  - *shm_atim.-* fecha actual;
  - *shm_lprid.-* Último proceso que la llamo
  - *shm_nattch*.-Se incrementa el número de procesos que han adjuntado la zona de memoria.

Llamadas al Sistema.- Shmat()

- Opciones:
  - SHM_RND: El sistema intentará vincular la zona de memoria a una dirección múltiplo de SHMLBA (shmparam.h) lo más próxima posible a la especificada.
  - SHM_RDONLY: Acceso al segmento de memoria en modo lectura. Si no se precisa, el segmento se vinculará en modo lectura/escritura.

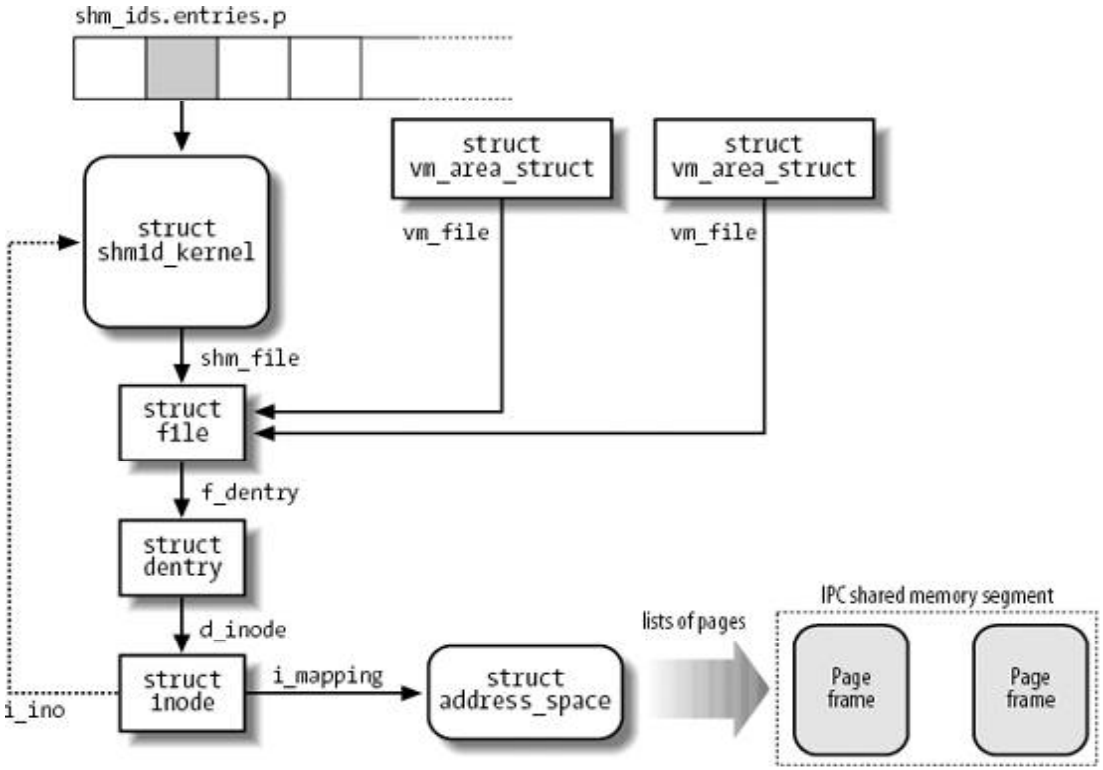
Llamadas al Sistema.- Shmdt()

- Desvincula una zona de memoria compartida de un proceso.

  int shmdt(void *shmaddr);

- Parámetros:
  - shmaddr.- Dirección de memoria compartida a desvincular.
- Salida: Actualiza los campos de la estructura *shmid_kernel:*
  - *shm_dtim.- Fecha actual de desvinculación.*
  - *shm_lprid.- Proceso que se ha desvinculado.*
  - *shm_nattch.- Se decrementa el número de procesos vinculados a la memoria compartida.*

Visión general



Estructuras de datos principales

- Las principales estructuras de datos usadas en memoria compartida son las siguientes:
  - shmid_ds
  - shmid_kernel
  - shm_info
  - shminfo
- Estructuras relacionadas:
  - *kern_ipc_perm*
  - *ipc_ids*

Estructura de datos.- shmid_ds

- Obsoleta, se mantiene sólo por compatibilidad hacia detrás, básicamente con “libc”.
- La llamada al sistema “shmctl” espera como tercer parámetro un puntero a una estructura de este tipo. En la versión 2.6 de Linux ha sido reemplazada en las gestiones del núcleo por *shmid_kernel.*
- Almacena información sobre un segmento de memoria: permisos, tamaño, etc.

Estructura de datos.- shmid_ds

- Su estructura es la siguiente:

Dirección en fuentes: /include/linux/shm.h

```
22 struct shmid_ds {
23          struct ipc_perm         shm_perm;
24          int                     shm_segsz;
25          __kernel_time_t         shm_atime;
26          __kernel_time_t         shm_dtime;
27          __kernel_time_t         shm_ctime;
28          __kernel_ipc_pid_t      shm_cpid;
29          __kernel_ipc_pid_t      shm_lpid;
30          unsigned short          shm_nattch;
31          unsigned short          shm_unused;
32          void                    *shm_unused2;
33          void                    *shm_unused3;
34 };
```

Estructura de datos.- shmid_ds

- Los campos significativos son:

Campo	Descripción
Shm_perm	Permisos de la memoria compartida
Shm_segsz	Tamaño del segmento (bytes)
Shm_atime	Fecha de la última asociación
Shm_dtime	Fecha de la última desasociación
Shm_ctime	Fecha de la última vez que se produjo el cambio.
Shm_cpid	pid del proceso creador
Shm_lpid	pid del último proceso que modifico.
Shm_nattch	número de asociaciones realizadas sobre la memoria compartida.

Estructura de datos.- shmid_kernel

- Contiene información importante sobre el segmento de memoria.
- Su estructura es la siguiente:

Dirección en fuentes: /include/linux/shm.h

```
76 struct shmid_kernel /* private to the kernel */
77 {
78          struct kern_ipc_perm    shm_perm;
79          struct file *           shm_file;
80          int                     id;
81          unsigned long           shm_nattch;
82          unsigned long           shm_segsz;
83          time_t                  shm_atim;
84          time_t                  shm_dtim;
85          time_t                  shm_ctim;
86          pid_t                   shm_cprid;
87          pid_t                   shm_lprid;
88          struct user_struct      *mlock_user;
89 };
```

Estructura de datos.- shmid_kernel

- Los campos significativos son:

Campo	Descripción
id	Índice del slot del segmento de memoria
Shm_file	Almacena la dirección de un objeto fichero.
Shm_perm	Estructura donde se almacenan los permisos
Shm_segsz	Tamaño del segmento (bytes)
Shm_atim	Fecha de la última vinculación
Shm_dtim	Fecha de la última desvinculación
Shm_ctim	Fecha de la última modificación
Shm_cprid	PID del creador
Shm_lprid	PID del último proceso que accedió
mlock_user	Puntero al descriptor *user_struct* del usuario usuario que bloqueó en RAM el recurso de memroia compartida.
Shm_nattch	Número de procesos que actualmente se encuentran vinculados

Estructura de datos.- shm_info

- Esta estructura se utiliza para obtener información sobre los recursos del sistema consumidos por el uso de memoria compartida. Esto es posible invocando a *shmctl* con cmd igual a SHM_INFO.
- Su estructura es la siguiente:

Dirección en fuentes: /include/linux/shm.h

```
66 struct shm_info {
67         int used_ids;
68         unsigned long shm_tot;
69         unsigned long shm_rss;
70         unsigned long shm_swp;
71         unsigned long swap_attempts;
72         unsigned long swap_successes;
73 };
```

Estructura de datos.- shm_info

- Los campos son los siguientes:

Campo	Descripción
used_ids	Segmentos que se están utilizando actualmente
shm_tot	Número total de páginas compartidas
shm_rss	Número total de páginas residentes
shm_swp	Número total de páginas intercambiadas (swapping)
swap_attempts	Número de intentos de swapping sobre una página de la región
swap_successes	Número de intentos de swapping que resultaron exitosos

Estructura de datos.- shminfo

- Obsoleta. Se mantiene por compatibilidad
- Se utiliza únicamente en la llamada a shmctl con IPC_INFO como argumento.

Dirección en fuentes: /include/linux/shm.h

```
57 /* Obsolete, used only for backwards compatibility */
58 struct shminfo {
59 int shmmax;
60 int shmmin;
61 int shmmni;
62 int shmseg;
63 int shmall;
64 };
```

Estructura de datos.-shminfo

- Los campos son los siguientes:

Campo	Descripción
shmmax	Tamaño máximo del segmento(bytes)
shmmin	Tamaño mínimo del segmento(bytes)
shmmni	Número máximo de segmentos
shmseg	Número máximo de segmentos por proceso
shmall	Número máximo de segmentos en número de páginas de memoria.

Estructura relacionadas

- *kern_ipc_perm.-* Cada descriptor IPC tiene un objeto de datos de este tipo como primer elemento. Esto hace posible acceder a cualquier descriptor desde cualquier función genérica IPC usando un puntero de este tipo de datos.

Estructura relacionadas

Dirección en fuentes: /include/linux/ipc.h

```
57 struct kern_ipc_perm
58 {
59         spinlock_t      lock;
60         int             deleted;
61         key_t           key;      // La clave que proporcionó shmget()
62         uid_t           uid;      // Usuario propietario
63         gid_t           gid;      // Grupo del propietario
64         uid_t           cuid;     // Usuario Creador
65         gid_t           cgid;     // Grupo del creador
66         mode_t          mode;     // Modo de acceso (9 bits)
67         unsigned long   seq;      // Número de secuencia
68         void            *security;
69 };
```

Los campos referentes al creador son fijos y no es posible modificarlos. Los campos relacionados con el propietario indican que usuarios pueden realizar operaciones de control sobre el segmento, y sólo pueden modificarlos los usuarios creador o propietario. Los permisos de operación de un proceso sobre un segmento de memoria compartida afectan a la lectura y escritura sobre el mismo. El *lock* protege el acceso a esta estructura.

Estructura relacionadas

- La estructura *ipc ids* describe los datos comunes para los semáforos, colas de mensajes, y memoria compartida.
- Tres instancias globales:
  - *semid_ids.- semáforos.*
  - *msgid_ids*.- mensajes.
  - *shmid_ids.-* memoria compartida.

Estructura relacionadas

Dirección en fuentes: /include/linux/util.h

```
23 struct ipc_ids {
24          int in_use;
25          int max_id;
26          unsigned short seq;
27          unsigned short seq_max;
28          struct semaphore sem;
29          struct ipc_id_ary nullentry;
30          struct ipc_id_ary* entries;
31 };
32
```

- El semáforo *sem* es usado para proteger el acceso a la estructura. El campo *entries* apunta a una matriz de descriptores de IPC donde se encuentran todos los recursos IPC creados. El campo *seq* es una secuencia de números global la cual será incrementada cuando un nuevo recurso IPC sea creado.

Código

- Las funciones que más interés nos causan son:
  - sys_shmget
  - sys_shmctl
  - sys_shmdt
  - do_shmat

Referencias bibliográficas

[***Oreilly***] Daniel P. Bovet, Marco Cesati, *"Understanding The Linux Kernel"*. 3rd Edition. O'Reilly Media, Inc., 2005. ISBN: 0-596-00565-2

[***INSIDE***] Tigran Aivazian, *"Dentro del n´ucleo Linux 2.4"*. 3rd Edition. Proyeto Lucas, 2001. URL:http://es.tldp.org/Manuales-LuCAS/DENTRO-NUCLEO-LINUX

[***Oreilly***] Rémy Card, Éric Dumas. *"Programación Linux 2.0. API y funcionamiento del cúcleo".* Ediciones Gestión, 2000.

Gracias

Felices Vacaciones a todos

Esperemos no vernos en Septiembre