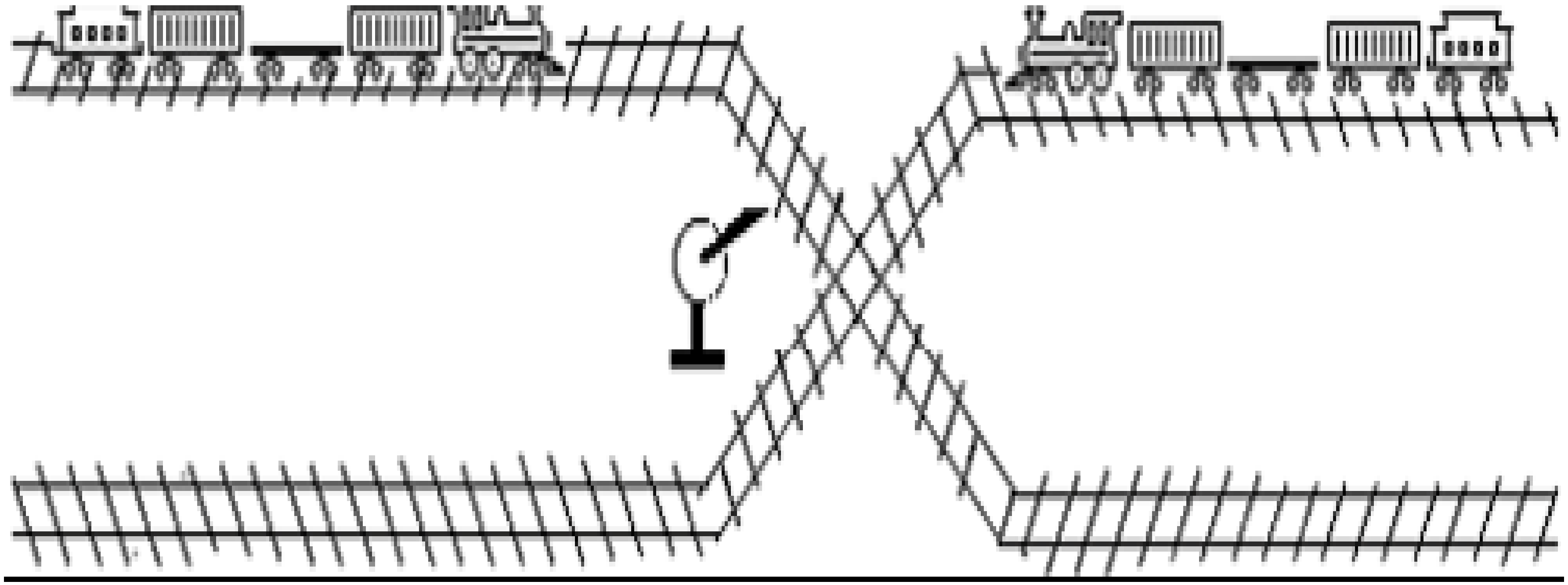
Semaphores

Chapter 7 from Inter-process Communications in Linux:
The Nooks & Crannies
by John Shapley Gray
Publisher: Prentice Hall
Pub Date: January 13, 2003

Topics

- Semaphore Definition
- Creating and Accessing Semaphore Sets
- Semaphore Control
- Semaphore Control Details
- Semaphore Operations
- Semaphore Operation Details
- Deleting Semaphores

Semaphores



Semaphore Definition

- A semaphore is a data structure that is shared by several processes. Semaphores are most often used to synchronize operations (to avoid race conditions) when multiple processes access a common, non-shareable resource.
- By using semaphores, we attempt to avoid other multi-programming problems such as:
    - Starvation
        - » Occurs when a process is habitually denied access to a resource it needs.
    - Deadlock
        - » Occurs when two or more processes each hold a resource that the other needs while waiting for the other process to release its resource.

Semaphore Definition

- To indicate a process has gained access to the resource, the process decrements the semaphore.
- For events to progress correctly, the test and decrement operation on the semaphore must be
- atomic (i.e., noninterruptible/indivisible).
- There are two kinds of Semaphores:
  - Binary semaphores
    - » Control access to a single resource, taking the value of 0 (resource is in use) or 1 (resource is available).
  - Counting semaphores
    - » Control access to multiple resources, thus assuming a range of nonnegative values.

Semaphore Definition

- Semaphore is a nonnegative integer that is stored in the kernel.
- Access to the semaphore is provided by a series of semaphore system calls.

Creating and Accessing Semaphore Sets

- Before a semaphore set can be used, it must be created.
- The creation of the semaphore set generates a unique data structure that the system uses to identify and manipulate the semaphores.
- A conceptual arrangement of a system semaphore structure for a newly allocated set of three semaphores is shown in Figure 7.1.

Creating and Accessing Semaphore Sets

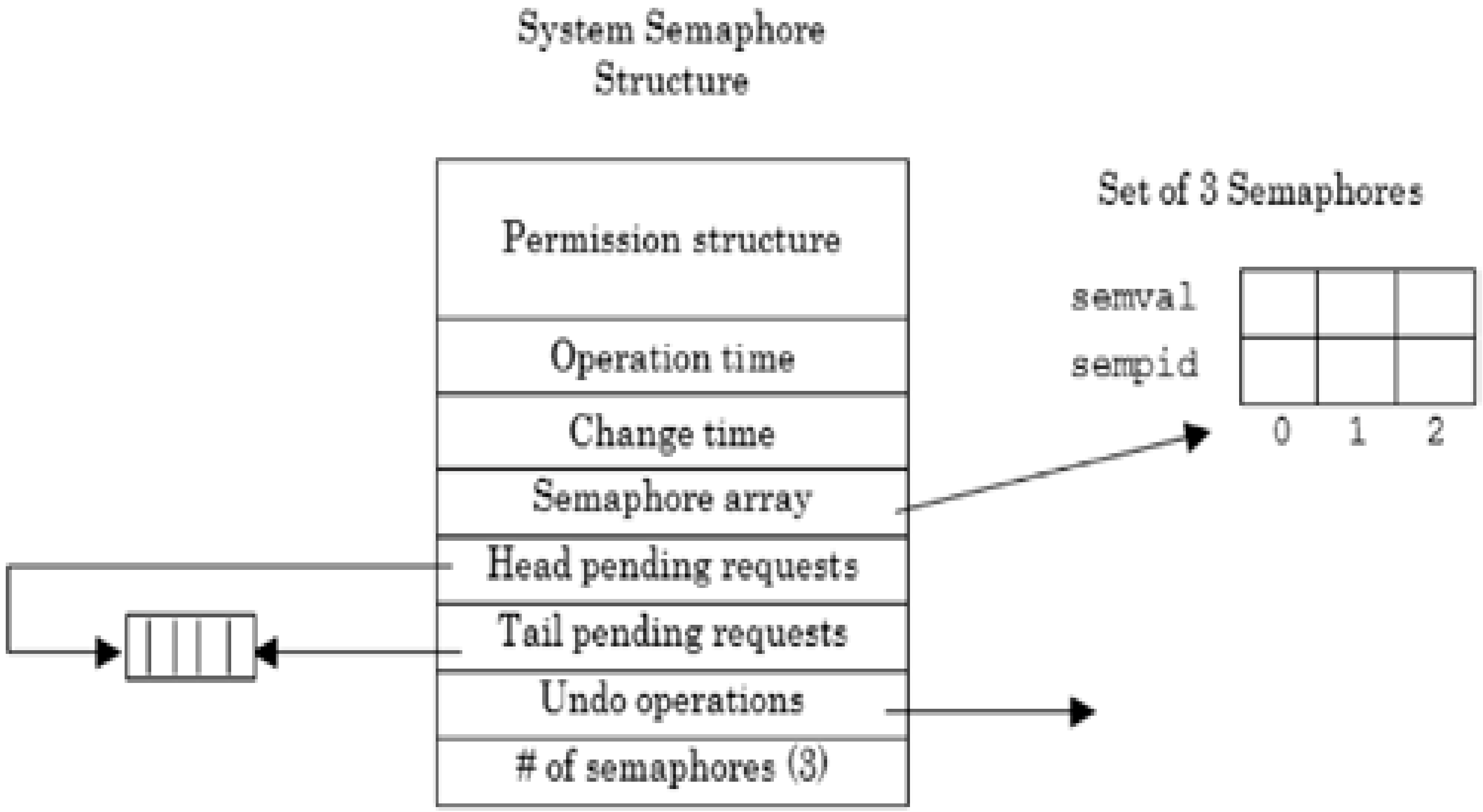


Figure 7.1. Data structures for a set of three semaphores.

Creating and Accessing Semaphore Sets

- To create a semaphore or gain access to one that exists, the semget system call is used.

  (Table 7.1)
- Exp 7.1 : A program to create semaphores

Creating and Accessing Semaphore Sets

Include File(s)	**<sys/types.h> <sys/ipc.h> <sys/sem.h>**	**Manual Section**	**2**
Summary	**int semget (key_t key,intnsems,int semflg);**		
Return	**Success**	**Failure**	**Sets errno**
	The semaphore identifier	**-1**	**Yes**

Table 7.1. Summary of the semget System Call

Creating and Accessing Semaphore Sets

- The semget system call takes three arguments:
  - The first argument, *key*, is used by the system to generate a unique semaphore identifier.
  - The second argument, *nsems*, is the number of semaphores in the set.
  - The third argument, *semflg*, is used to specify access permission and/or special creation conditions.

Creating and Accessing Semaphore Sets

- If the semget system call fails, it returns a −1 and sets the value stored in errno.

(Table 7.2.)

Creating and Accessing Semaphore Sets

#	Constant	perror Message	Explanation
2	EOENT	No such file or directory	Semaphore identifier does not exist for this key, and IPC_CREAT was not set.
12	ENOMEM	Cannot allocate memory	Insufficient system memory to allocate the semaphore set.
13	EACCES	Permission denied	Semaphore identifier exists for this key, but requested operation is not allowed by current access permissions.
17	EEXIST	File exists	Semaphore identifier exists for this key, but the flags IPC_CREAT and IPC_EXCL are both set.
28	ENOSPC	No space left on device	System-imposed limit (SEMMNI) for the number of semaphore sets or systemwide maximum number of semaphores (SEMMNS) has been reached.
43	EIDRM	Identifier removed	Specified semaphore set is marked for removal.

Table 7.2. semget Error Messages.

Semaphore Control

- The semctl system call allows the user to perform a variety of generalized control operations on the system semaphore structure, on the semaphores as a set, and on individual semaphores.

  (Table 7.3)

Semaphore Control

Include File(s)	<sys/types.h> <sys/ipc.h> <sys/sem.h>	Manual Section	2
Summary	int semctl(int semid, int semnum, int cmd, union semun arg);		
Return	Success	Failure	Sets errno
	0 or the value requested	-1	Yes

Table 7.3. Summary of the semctl System Call

Semaphore Control

- The semctl system call takes four arguments:
  - The first argument, semid, is a valid semaphore identifier that was returned by a previous semget system call.
  - The second argument, semnum, is the number of semaphores in the semaphore set (array), 0 means this number (index) is not relevant.
  - The third argument to semctl, cmd, is an integer command value. the cmd value directs semctl to take one of several control actions. Each action requires specific access permissions to the semaphore control structure.

Semaphore Control

- The fourth argument to semctl, arg, is a union of type semun. Given the action specified by the preceding cmd argument, the data in arg can be one of any of the following four values:
  - » An integer already was set in the val member of sem_union that used with SETVAL to indicate a change of specific value for a particular semaphore within the semaphore set.
  - » A reference to a semid_ds structure where information is returned when IPC_STAT or IPC_SET is specified.
  - » A reference to an array of type unsigned short integers; the array is used either to initialize the semaphore set or as a return location when specifying GETALL.
  - » A reference to a seminfo structure when IPC_INFO is requested.

Semaphore Control

- If semctl fails, it returns a value of −1 and sets errno to indicate the specific error.

  (Table 7.4.)

Semaphore Control

#	Constant	perror Message	Explanation
1	EPERM	Operation not permitted	Value for cmd is IPC_RMID or IPC_SET and the calling process in not the owner or superuser.
13	EACCES	Permission denied	The requested operation is not allowed by the current access permissions for this process.
14	EFAULT	Bad address	The fourth argument to semctl contains a reference to an illegal address (the union semun may not have been declared).
22	EINVAL	Invalid argument	• The semaphore identifier is invalid. • The number of semaphores specified is less than 0 or greater than the number in the semaphore set. • The value for cmd is invalid. • The value for cmd is IPC_SET, but the value for sem_perm.uid or sem_perm.gid is invalid.

Table 7.4. semctl Error Messages.

Semaphore Control

#	Constant	perror Message	Explanation
34	ERANGE	Numerical result out of range	The value for cmd is SETVAL or SETALL, and the value to be assigned is greater than the system maximum or less than 0.
43	EIDRM	Identifier removed	Specified semaphore set is marked for removal.

Table 7.4. semctl Error Messages.

Semaphore Control Details

- The following cmd values cause semctl to act upon the system semaphore structure
    - IPC_STAT
        - Return the current values of the semid_ds structure for the indicated semaphore identifier. The returned information is stored in a user-generated structure referenced by the fourth argument to semctl. To specify IPC_STAT, the process must have read permission for the semaphore set associated with the semaphore identifier.

Semaphore Control Details

- IPC_SET
  - Modify a restricted number of members in the semid_ds structure. The members sem_perm.uid, sem_perm.gid and sem_perm.mode can be changed if the effective ID of the accessing process is that of the superuser or is the same as the ID value stored in sem_perm.cuid or sem_perm.uid. To make these changes, a structure of the type semid_ds must be allocated. The appropriate members' values are then assigned, and a reference to the modified structure is passed as the fourth argument to the semctl system call.
- IPC_RMID
  - Remove the semaphore set associated with the semaphore identifier.

Semaphore Control Details

- The following cmd values cause semctl to act upon the entire set of semaphores:
  - GETALL
    - Return the current values of the semaphore set. The values are returned via the array reference passed as the fourth argument to semctl. The user is responsible for allocating the array of the proper size and type prior to passing its address to semctl. Read permission for the semaphore set is required to specify GETALL. When specifying GETALL, the argument semnum is ignored.

Semaphore Control Details

- SETALL
  - Initialize all semaphores in a set to the values stored in the array referenced by the fourth argument to semctl. Again, the user must allocate the initializing array and assign values prior to passing the address of the array to semctl. The process must have alter access for the semaphore set to use SETALL. When specifying SETALL, the sem_ctime member of the system semaphore data structure is updated.

Semaphore Control Details

- The last set of semctl cmd values acts upon individual semaphores or upon specific members in the semid_ds structure. All of these commands require read permission except for SETVAL, which requires alter permission:
  - GETVAL
    - » Return the current value of the individual semaphore referenced by the value of the semnum argument.
  - SETVAL
    - » Set the value of the individual semaphore referenced by the semnum argument to the value specified by the fourth argument to semctl.

Semaphore Control Details

- GETPID
  - » Return the PID from the sem_perm structure within the semid_ds structure.
- GETNCNT
  - » Return the number of processes waiting for the semaphore referenced by the semnum argument to increase in value.
- GETZCNT
  - » Return the number of processes waiting for the semaphore referenced by the semnum argument to become 0.

Semaphore Operations

- Additional operations on individual semaphores are accomplished by using the semop system call.

(Table 7.5.)

Semaphore Operations

Include File(s)	**<sys/types.h> <sys/ipc.h> <sys/sem.h>**	**Manual Section**	**2**	
Summary	**int semop(int semid, struct sembuf *sops, unsigned nsops);**			
Return	**Success**		**Failure**	**Sets errno**
	0		**-1**	**Yes**

Table 7.5. Summary of the semop System Call

Semaphore Operations

- The semop system call takes three arguments:
  - The first argument, semid, is a valid semaphore identifier that was returned by a previous successful semget system call.
  - The second argument, sops, is a reference to the base address of an array of semaphore operations that will be performed on the semaphore set associated with by the semid value.
  - The third argument, nsops, is the number of elements in the array of semaphore operations.

Semaphore Operations

- If semop fails, it returns a value of −1 and sets errno to indicate the specific error.

  (Table 7.9.)

Semaphore Operations

#	Constant	perror Message	Explanation
4	EINTR	Interrupted system call	While in a wait queue for the semaphore, a signal was received by the calling process.
7	E2BIG	Argument list too long	The value for nsops is greater than the system limit.
11	EAGAIN	Resource temporarily unavailable	The requested operation would cause the calling process to block, but IPC_NOWAIT was specified.
12	ENOMEM	Cannot allocate memory	The limit for number of processes requesting SEM_UNDO has been exceeded.
13	EACCES	Permission denied	The requested operation is forbidden by the current access permissions.
14	EFAULT	Bad address	The value for sops references an illegal address.
22	EINVAL	Invalid argument	• The semaphore identifier is invalid. • The number of semaphores requesting SEM_UNDO is greater than the system limit.

Table 7.9. semop Error Messages.

Semaphore Operations

#	Constant	perror Message	Explanation
27	EFBIG	File too large	The value for sem_num is < 0 or >= to the number of semaphores in the set.
34	ERANGE	Numerical result out of range	The requested operation would cause the system semaphore adjustment value to exceed its limit.
43	EIDRM	Identifier removed	The semaphore set associated with semid value has been removed.

Table 7.9. semop Error Messages.

Semaphore Operation Details

- When the sem_op value is negative, the process specifying the operation is attempting to decrement the semaphore.
- The decrement of the semaphore is used to record the acquisition of the resource affiliated with the semaphore.
- When a semaphore value is to be modified, the accessing process must have alter permission for the semaphore set.

  (Table 7.6.)

Semaphore Operation Details

Condition	Flag Set	Action Taken by semop
semval >= abs(semop)		Subtract abs(sem_op) from semval.
semval >= abs(semop)	SEM_UNDO	Subtract abs(sem_op) from semval and update the undo counter for the semaphore.
semval < abs(semop)		Increment semncnt for the semaphore and wait (block) until •semval >= abs(semop), then adjust semncnt and subtract as noted in the previous two rows of table. •semid is removed, then return −1 and set errno to EIDRM. •A signal is caught, then adjust semncnt and set errno to EINTR.
semval < abs(semop)	IPC_NOWAIT	Return −1 immediately and set errno to EAGAIN.

Table 7.6. Actions Taken by semop when the Value for sem_op is Negative.

Semaphore Operation Details

- When the sem_op value is positive, the process is adding to the semaphore value. The addition is used to record the return (release) of the resource affiliated with the semaphore.
- Again, when a semaphore value is to be modified, the accessing process must have alter permission for the semaphore set.

  (Table 7.7.)

Semaphore Operation Details

Condition	Flag Set	Action Taken by semop
		Add sem_op to semval.
	SEM_UNDO	Add sem_op to semval and update the undo counter for the semaphore.

Table 7.7. Actions Taken by semop when the Value for sem_op is Positive.

Semaphore Operation Details

- When the sem_op value is zero, the process is testing the semaphore to determine if it is at 0.
- When a semaphore is at 0, the testing process can assume that all the resources affiliated with the semaphore are currently allocated (in use).
- For a semaphore value to be tested, the accessing process must have read permission for the semaphore set.
- (Table 7.8.)

Semaphore Operation Details

Condition	Flag Set	Action Taken by semop
semval == 0		Return immediately.
semval != 0	IPC_NOWAIT	Return −1 immediately and set errno to EAGAIN.
semval != 0		Increment semzcnt for the semaphore and wait (block) until • semval == 0, then adjust semzcnt and return. • semid is removed, then return −1 and set errno to EIDRM. • A signal is caught, then adjust semzcnt and set errno to EINTR.

Table 7.8. Actions Taken by semop when the Value for sem_op is Zero.

Deleting Semaphores

- The semctl command with IPC_RMID removes the semaphore in the program
- The command "ipcs -s" will list all semaphores on a system.
- The command "ipcrm -s {semid}" will delete a semaphore on system promt.
- To delete all semaphores you have authority over, you can use this on system;

for semid in `ipcs -s | cut -d\  -f2`; do ipcrm -s $semid; done

- Exp 7.2 , Exp 7.3: See these header files which contain all the semaphore system calls discussed so far.