Network Programming

BITS Pilani
Pilani Campus

K Hari Babu
Department of Computer Science & Information Systems

Outline

- POSIX Message Queues
- POSIX Semaphores
- POSIX Shared Memory

innovate achieve lead

BITS Pilani

Pilani Campus

Overview of POSIX IPC (R1: Ch51)

POSIX IPC

innovate achieve lead

- Aim was to devise a set of IPC mechanisms that did not suffer the deficiencies of System V IPC.
  - Message Queues
  - Semaphores
  - Shared memory

API Overview

innovate achieve lead

- All three mechanisms have similar API.

Table 51-1: Summary of programming interfaces for POSIX IPC objects

Interface	Message queues	Semaphores	Shared memory
Header file	`<mqueue.h>`	`<semaphore.h>`	`<sys/mman.h>`
Object handle	*mqd_t*	*sem_t **	*int* (file descriptor)
Create/open	*mq_open()*	*sem_open()*	*shm_open()* + *mmap()*
Close	*mq_close()*	*sem_close()*	*munmap()*
Unlink	*mq_unlink()*	*sem_unlink()*	*shm_unlink()*
Perform IPC	*mq_send()*, *mq_receive()*	*sem_post()*, *sem_wait()*, *sem_getvalue()*	operate on locations in shared region
Miscellaneous operations	*mq_setattr()*—set attributes *mq_getattr()*—get attributes *mq_notify()*—request notification	*sem_init()*—initialize unnamed semaphore *sem_destroy()*—destroy unnamed semaphore	(none)

POSIX IPC API

innovate achieve lead

- Three stages
  - Create/open IPC object
  - Perform operations
  - Close/remove the object
- Create/open IPC object
  - Each IPC mechanism has an associated open call (mq_open(), sem_open(), or shm_open()),
    - which is analogous to the traditional UNIX open() system call used for files.

```
2    fd = shm_open("/mymem", O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
```

IPC Object Names

- Identifying a POSIX IPC object is via a name consisting of an initial slash, followed by one of more nonslash characters;
  - For example, /myobject.

Closing an IPC object

- *Close()* call that indicates that the calling process has finished using the object.
- IPC objects are automatically closed if the process terminates or performs an exec().
- As with open files, POSIX IPC objects are reference counted—the kernel maintains a count of the number of open references to the object.
- Each IPC object has a corresponding unlink call whose operation is analogous to the traditional *unlink()* system call for files.
  - The unlink call immediately removes the object's name, and then destroys the object once all processes cease using it

Listing All IPC Objects

- Listed under the root directory (/), and the standard ls and rm commands can be used to list and remove IPC objects.
- Compiling programs that use POSIX IPC on Linux
  - On Linux, programs employing the POSIX IPC mechanisms must be linked with the realtime library, librt, by specifying the –lrt option to the cc command.

Comparison of System V IPC and POSIX IPC

- The POSIX IPC interface is simpler than the System V IPC interface.
- The POSIX IPC model—the use of names instead of keys, and the open, close, and unlink functions—is more consistent with the traditional UNIX file model.
- POSIX IPC objects are reference counted. This simplifies object deletion, because we can unlink a POSIX IPC object, knowing that it will be destroyed only when all processes have closed it.
- There is one notable advantage in favor of System V IPC: portability.

Message Queues

- POSIX message queues are reference counted.
- Each System V message has an integer type, and messages can be selected in a variety of ways using msgrcv(). By contrast, POSIX messages have an associated priority, and messages are always strictly queued (and thus received) in priority order.
- POSIX message queues provide a feature that allows a process to be asynchronously notified when a message is available on a queue.
- On Linux, POSIX message queues can be monitored using poll(), select(), and epoll. System V message queues don’t provide this feature.
- POSIX message queues are less portable.
- The facility to select System V messages by type provides slightly greater flexibility than the strict priority ordering of POSIX messages.

Creating POSIX Message Queue

- The mq_open() function creates a new message queue or opens an existing queue.

```
2  #include <fcntl.h> /* Defines O_* constants */
3  #include <sys/stat.h> /* Defines mode constants */
4  #include <mqueue.h>
5  mqd_t mq_open(const char *name, int oflag, ...
6  /* mode_t mode, struct mq_attr *attr */);
7  //Returns a message queue descriptor on success, or (mqd_t) -1 on error
```

Table 52-1: Bit values for the *mq_open() oflag* argument

Flag	Description
O_CREAT	Create queue if it doesn't already exist
O_EXCL	With O_CREAT, create queue exclusively
O_RDONLY	Open for reading only
O_WRONLY	Open for writing only
O_RDWR	Open for reading and writing
O_NONBLOCK	Open in nonblocking mode

Closing a message queue

- The mq_close() function closes the message queue descriptor mqdes.

```
2  #include <mqueue.h>
3  int mq_close(mqd_t mqdes);
4  //Returns 0 on success, or -1 on error
```

- Removing a message queue

```
1  #include <mqueue.h>
2  int mq_unlink(const char *name);
3  //Returns 0 on success, or -1 on error
```

Differences with System V IPC

- Message Queues
- Semaphores
- Shared Memory

Q&A

Next Time

innovate achieve lead

- Please read through R1: chapters 47-48

innovate achieve lead

BITS Pilani
Pilani Campus

Thank You