Shared Memory Details

Matt Ruffalo

March 19, 2013

Case Western Reserve University Est. 1826

Recap: System V API

- Creating/Obtaining: `shmget(...)`
- Control (modify, remove): `shmctl(...)`
- Accessing (attach/detach): `smhat(...)`, `shmdt(...)`

Creating Shared Memory

shmget call

```
#include <sys/ipc.h>
#include <sys/shm.h>

int shmget(key_t key, int size, int shmflag);
```

Creating Shared Memory

Parameters

```
int shmget(key_t key, int size, int shmflag);
```

- `key`: Can be whatever you want, or `IPC_PRIVATE`
- `size`: The shared memory's size, in bytes
- `shmflg`: Creation flags: use `IPC_CREAT` to make a new shared memory segment
  - Lower 9 bits are permissions; you should probably use 0666

```
int shm_id = shmget(IPC_PRIVATE, 256,
    IPC_CREAT | 0666);
```

Obtaining Shared Memory

Parameters

```
int shmget(key_t key, int size, int shmflag);
```

- `key`: Use the same key!
- `size`: Doesn't matter; should probably use 0
- `shmflg`: Don't specify `IPC_CREAT`, but probably should use same permissions

```
int shm_id = shmget(key, 0, 0666);
```

Attaching Shared Memory

Parameters

```
void* shmat(int shmid, const void *shmaddr,
    int shmflag);
```

- `shmid`: Return value of `shmget`
- `shmaddr`: Address to map; can specify 0 for "don't care"
- `shmflg`: Probably not necessary

```
*void mem_ptr = shmat(shm_id, (void *) 0, 0);
```

Detaching Shared Memory

Parameters

```
int shmdt(const void *shmaddr);
```

- `shmaddr`: Address that was returned by `shmat`

```
int success = shmdt(mem_ptr);
```

Removing Shared Memory

Parameters

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

- `shmid`: Return value of `shmget`
- `cmd`: use `IPC_RMID` to remove
- `buf`: Can pass 0 for this

```
int success = shmdt(mem_ptr);
```

Using Shared Memory

Parameters

```
*void mem_ptr = shmat(shm_id, (void *) 0, 0);
```

- Attaches a block of memory that's shared between processes
- Gives you a "raw" pointer: no type
- Can cast it to whatever type you want, like a `struct`