CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

POSIX Shared Memory Example

Solution to producer-consumer problem using POSIX shared memory

Full code is in textbook

Need these routines to create and manage shared memory

- `shm_open`
- `ftruncate`
- `mmap`
- `shm_unlink`

As always, `man` pages are a good resource

```
shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);
```

Create a "shared memory object" (like a file that is not written to disk)

Can access it by name in `/dev/shm/`

Allows different flags and access permissions

Returns a file descriptor just like regular open

```
ftruncate(shm_fd, SIZE);
```

Assign a size to your shared memory object

```
ptr = mmap(0, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0
```

`mmap` works with regular files as well as shared memory objects

Maps file (or shared memory object) into memory so it can be read and written via pointers, just like any other memory

Without this, `shm_fd` would be treated like a file, which would require system calls to read and write

Note the `MAP_SHARED` argument

```
shm_unlink(name);
```

Remove shared memory segment and delete filename from `/dev/shm/`

Why `unlink`? You are really just removing the name. Only once last reference is gone is resource actually reclaimed.

`rm` uses `unlink` system call underneath (which you can verify with `strace`)

Producer

```
/* create the shared memory segment */
shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);

/* configure the size of the shared memory segment */
ftruncate(shm_fd,SIZE);

/* now map the shared memory segment in the address space of the process */
ptr = mmap(0,SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
if (ptr == MAP_FAILED) {
 printf("Map failed\n");
 return -1;
}

/**
 * Now write to the shared memory region.
  *
 * Note we must increment the value of ptr after each write.
 */
sprintf(ptr,"%s",message0);
ptr += strlen(message0);
sprintf(ptr,"%s",message1);
ptr += strlen(message1);
sprintf(ptr,"%s",message2);
ptr += strlen(message2);

return 0;
```

Consumer

```
/* open the shared memory segment */
shm_fd = shm_open(name, O_RDONLY, 0666);
if (shm_fd == -1) {
 printf("shared memory failed\n");
 exit(-1);
}

/* now map the shared memory segment in the address space of the process */
ptr = mmap(0,SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
if (ptr == MAP_FAILED) {
 printf("Map failed\n");
 exit(-1);
}

/* now read from the shared memory region */
printf("%s",ptr);

/* remove the shared memory segment */
if (shm_unlink(name) == -1) {
 printf("Error removing %s\n",name);
 exit(-1);
}
```