System V Shared Memory

UNX511 Week 13 Class 1

John Sellens

August 5, 2025

Seneca Polytechnic

Outline

System V Shared Memory

System V Shared Memory

- System V Shared Memory allows unrelated processes to share memory
- Allows fast, unrestricted data sharing
- No inherent locking or synchronization mechanisms
  - Use semaphores or some other method to coordinate
- There is also POSIX shared memory

Shared Memory Operations

- `shmget(2)` – create a memory segment that can be used
  - Global namespace, has owner, permissions
- `shmat(2)` – attach – return a pointer to the memory for use by a process
- `shmdt(2)` – detach – disconnect memory from process
- `shmctl(2)` – control – e.g. delete the memory segment

Shared Memory Code Samples

- Let's have a look in unx511_samples
  - `https://github.com/jsellens/unx511_samples`
- `week13_1/1_transfer` – simple shared memory example
- `week13_1/2_mempool` – invalid attempt to share memory pointers
- `week13_1/3_dispatch` – message sending via shared memory
- `week13_1/4_radioChannels` – four communication "channels" in a shared memory segment

References

- On Ubuntu, for the posix man pages:
  `sudo apt install manpages-posix-dev`
- `https://github-pages.senecapolytechnic.ca/unx511/Week12/Week12.html`
- The Linux Programming Interface book, chapter 48 “System V Shared Memory”

Summary

- A useful, but simple, way to share data across unrelated processes
  - If there was a common thread parent, you could just use global variables
- You can get similar functionality using a memory-mapped file, using `mmap(2)`