Operating systems

Programming Assignments 2

For each assignment a student is supposed to deliver a test, that will test the correctness of implementation. Try to invent such a test that is likely to trigger incorrect behaviour.

Full documentation for synchronization and communication mechanism is available after `manpages-posix-dev` package is installed. The list of useful manual pages is provided below:

- `unix(7)`
- `mq_overview(7)`, `sem_overview(7)`, `shm_overview(7)`
- `pthread_mutex_lock(P)`
- `pthread_cond_wait(P)`, `pthread_cond_signal(P)`
- `pthread_barrier_wait(P)`

Other useful links:

1. Vim editor cheat sheet
2. Quick introduction to GNU Make
3. Quick reference for GDB

Assignment 1

Write a simple instant messaging application for two or more people. Your program should only work locally, thus you're supposed to employ named Unix domain sockets for inter-process communication. Create the socket in such a way that other users are allowed to open it. Send and receive messages after `ENTER` key is pressed.

Assignment 2

Readers-writers problem is one of the classic synchronization problems. Solve it using `POSIX` binary semaphores (`pthread_mutex`) and `POSIX` threads. Implement both versions, where readers are favoured and where writers are favoured.

Assignment 2+

Provide solution for readers-writers problem, such that no starvation can occur.

Assignment 3

Use `POSIX` semaphores to implement reusable barrier with three operations `init`, `wait` and `destroy`. Barrier's reusability means that after first batch of processes passed the barrier, it must behave as if it was initialized by `init` function (i.e. chain of `wait` operations should behave correctly without calling `init` and `destroy`).

Assignment 4+

Both message passing and semaphores plus shared memory, deliver similar expression power. In order to show that, implement simple interface for:

- mail boxes using semaphores (Hint: use shared memory to realize a buffer that stores N messages of arbitrarily chosen maximum length)
- counting semaphores using mail boxes (Hint: use separate process as a synchronization guarantee)

Assignment 5

Use `POSIX` mutexes (`pthread_mutex`) and conditional variables (`pthread_cond`) to implement consumer producer problem for a buffer of limited size of `N` entries. Is it possible to implement aforementioned monitor in both ways (i.e. Hoare and Mesa) using `POSIX` threads library?

Assignment 6+

Three kinds of threads share access to a singly-linked list data structure:

- Searchers: they only browse the list without modifying it, hence multiple searches are allowed to perform concurrent look-up operations,
- Inserters: they append a new item at the end of list, thus they have to avoid a situation where they modify last element of the list concurrently; however one append operation can coexist with many concurrent look-up operations,
- Deleters: they are allowed to modify arbitrary element of the list, hence for the time of delete operation, all other operations must not be allowed.

Impement a concurrent container backed by singly-linked lists, with three operations described as above.