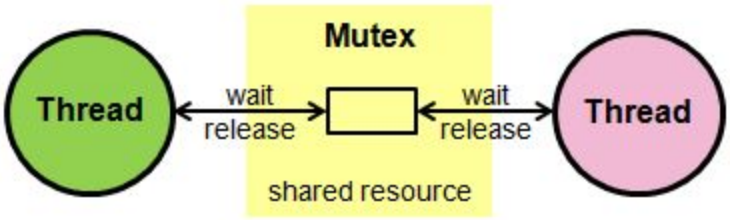
Documentation: Multiple Reader Writer with Synchronization

Computer Architecture and Operating System

Code Description

The code is a multiple reader-writer implementation that has synchronization to overcome race condition. There is a common queue(circular queue) of numbers provided to simply implement the reader-writer operations. While a writer “writes” a certain element to the queue, no “reader” can read that element and also, other writers cannot “write” at the same time. The readers may read other elements but not the one current written to. Similarly, only one “writer” can update the queue at a time. Multiple “readers” may read elements of the queue but during dequeue from the queue, other readers cannot dequeue at the same time.



The Writer Function

In the very starting, the writer function locks the control and denies other writer blocks. Inside the writer function, adding to the queue and quit option is provided. When there’s only one element in the queue, then the function also locks the reader blocks.

```
void *writer_func(void *arg)
```

The Reader Function

The writer function on execution start blocks access to both any other writer and reader thread. The reader function provides a medium to dequeue an element from the queue and to display the entire queue.

```
void *reader_func(void *arg)
```

Navya Aggarwal | 2018349 | navya18349@iiitd.ac.in

Important Functions and Structs used:

1. sem_init() - **sem_init**() initializes the unnamed semaphore at the address pointed to by *sem*. The *value* argument specifies the initial value for the semaphore.

```
sem_init(&mutex,0,1);
sem_init(&wrt,0,1);
```

2. pthread_create() - **pthread_create**() function starts a new thread in the calling process.

```
for(i=0; i<red_count; i++)
    pthread_create(&rtid[i],NULL,reader_func,(void *)i);
for(i=0; i<wrt_count; i++)
    pthread_create(&wtid[i],NULL,writer_func,(void *)i);
```

3. pthread_join() - The **pthread_join**() function waits for the thread specified by *thread* to terminate.

```
for(i=0; i<wrt_count; i++)
    pthread_join(wtid[i],NULL);
for(i=0; i<red_count; i++)
    pthread_join(rtid[i],NULL);
```

4. sem_destroy() - **sem_destroy**() destroys the unnamed semaphore at the address pointed to by *sem*.

```
sem_destroy(&mutex);
sem_destroy(&wrt);
```

5. sleep() - **sleep**() causes the calling thread to sleep either until the number of real-time seconds specified in *seconds* has elapsed or until a signal arrives which is not ignored.
6. sem_wait() - sem_wait() decrements (locks) the semaphore pointed to by sem.

```
sem_wait(&wrt);
sem_wait(&mutex);
```

7. sem_post() - **sem_post**() increments (unlocks) the semaphore pointed to by *sem*.

```
sem_post(&wrt);
sem_post(&mutex);
```

Compilation

The file is compiled using a Makefile and the executable formed can be executed using the command mentioned below.

c: > Users > ADMIN > Desktop > Semester 3 > a > Makefile

```
1  #makefile to compile semaphore code
2  CC=gcc                          #compiler
3  TARGET=semaphore_executable     #semaphore file name
4
5  all:    #target name
6          $(CC) -o $(TARGET) semaphore.c -lpthread
7  clean:  #clean the history file
8          rm $(TARGET)
```

```
navya@hp:/mnt/c/Users/ADMIN/Desktop/Semester 3/a$ make
```

```
navya@hp:/mnt/c/Users/ADMIN/Desktop/Semester 3/a$ ./semaphore_executable
```

Errors Handled

1. Proper locking of Writer and Reader threads is done to ensure that the program doesn't run into any race conditions.
2. Whenever input from the user is taken, checks are made to ensure that no false input is passed.
3. Whenever the program runs into an error situation by user end input, the code exits immediately.

Reference and Online Resources Used

1. https://images.app.goo.gl/UxbwT7MTdRL9h6Es6
2. https://gist.github.com/rajabishek/6209a575f00b122fe490
3. https://github.com/snehasi/Multiple-Reader-writer/blob/master/rw.c
4. http://man7.org/linux/man-pages/man3/sem_init.3.html
5. http://man7.org/linux/man-pages/man3/pthread_create.3.html
6. http://man7.org/linux/man-pages/man3/pthread_join.3.html
7. http://man7.org/linux/man-pages/man3/sem_destroy.3.html
8. http://man7.org/linux/man-pages/man3/sleep.3.html
9. http://man7.org/linux/man-pages/man3/sem_timedwait.3.html
10. http://man7.org/linux/man-pages/man3/sem_post.3.html