Assignment 10 | Operating System CE-092

Assignment submission for Operating System subject week 10.

nevilparmar24@gmail.com

Task 1:

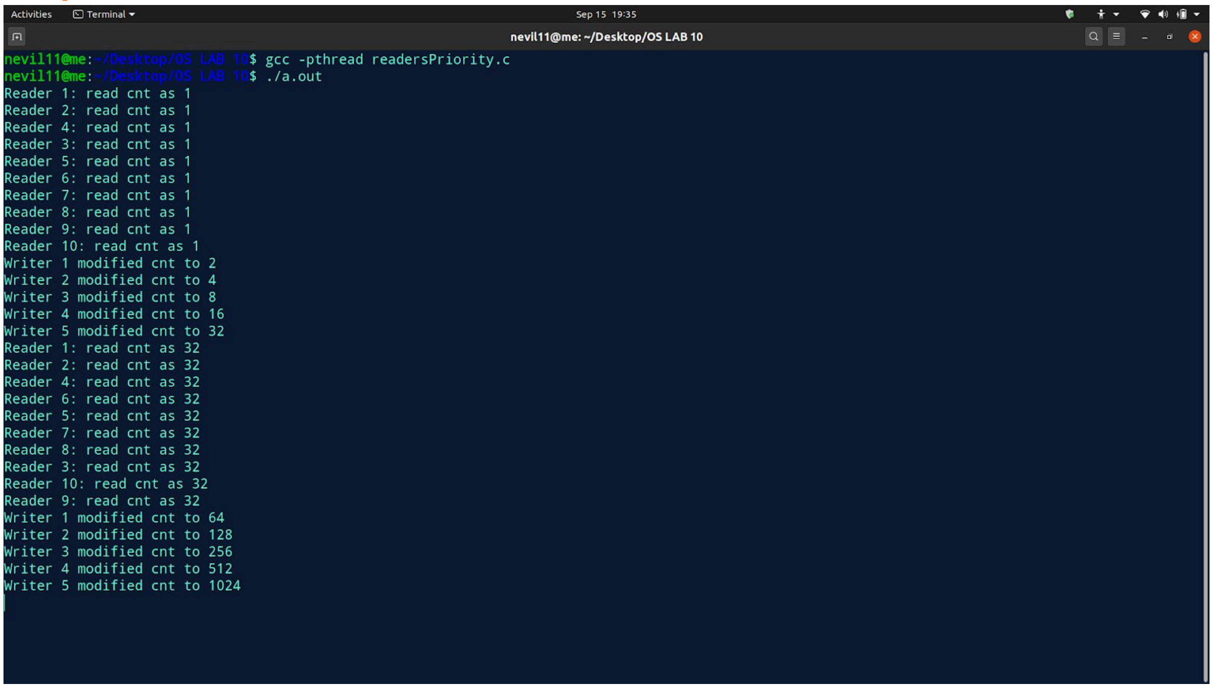
Reader Writers problem (Readers have priority).

Code:

```
/*
* @Author: nevil11
* @Date:   2020-09-17 10:34:17
* @Last Modified by:   nevil11
* @Last Modified time: 2020-09-17 10:38:31
*/
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<limits.h>

sem_t wrt;
pthread_mutex_t mutex;
int cnt = 1;
int readCount = 0;


/* Function Signatures */
```

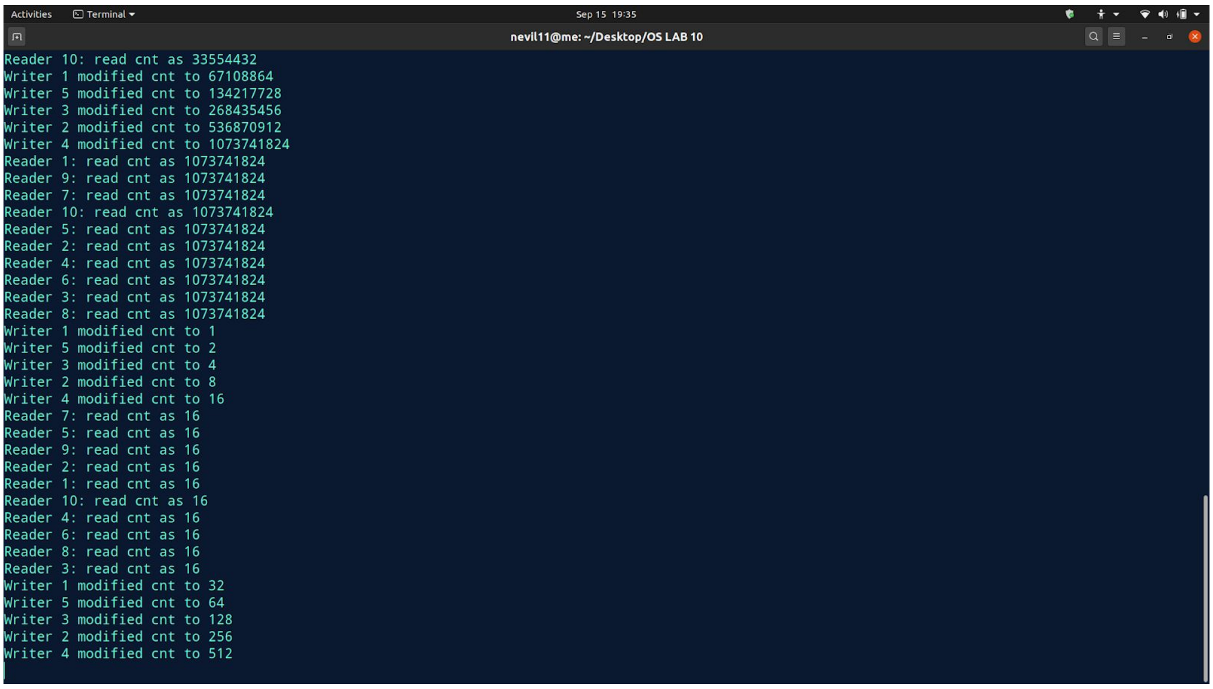
```c
void doReading(void *rno);
void doWriting(void *wno);
void reader(void *rno);
void writer(void *wno);

// In real application we can perform some DB
operations
void doReading(void *rno) {
    // Reading Section
    printf("Reader %d: read cnt as %d\n", *((int
*)rno), cnt);
}

void doWriting(void *wno) {
    // since we are running it inside the infinite loop
, the value of cnt may go beyond the limit of 32 bit
int
    if ( ((long long int)cnt * 2) > INT_MAX)
        cnt = 1;
    else
        cnt = cnt * 2;
    printf("Writer %d: modified cnt to %d\n", (*((int
*)wno)), cnt);
}

void writer(void *wno)
{
    while (1) {

        // writer's job is to simply write and exit, so
it interacts with wrt semaphor only
        sem_wait(&wrt);

        doWriting(wno);
        sem_post(&wrt);

        // sleep is used just to slow down the output
on the terminal
        sleep(2);
    }
}

void reader(void *rno)
{
    while (1) {

        // Reader acquire the lock before modifying
readCount
        pthread_mutex_lock(&mutex);
        readCount++;
        if (readCount == 1) {
            sem_wait(&wrt); // If this id the first
reader, then it will block the writer
        }
        pthread_mutex_unlock(&mutex);

        doReading(rno);

        // Reader acquire the lock before modifying
readCount
        pthread_mutex_lock(&mutex);
        readCount--;
        if (readCount == 0) {
            sem_post(&wrt); // If this is the last
reader, it will wake up the writer.
        }
        pthread_mutex_unlock(&mutex);
        // sleep is used just to slow down the output
on the terminal
        sleep(2);

    }
}

int main()
{

    pthread_t read[10], write[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&wrt, 0, 1);

    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; //Just
used for numbering the producer and consumer

    for (int i = 0; i < 10; i++) {
        pthread_create(&read[i], NULL, (void *)reader,
(void *)&a[i]);
    }
    for (int i = 0; i < 5; i++) {
        pthread_create(&write[i], NULL, (void *)writer,
(void *)&a[i]);
    }

    for (int i = 0; i < 10; i++) {
        pthread_join(read[i], NULL);
    }
    for (int i = 0; i < 5; i++) {
        pthread_join(write[i], NULL);
    }
    pthread_mutex_destroy(&mutex);
    sem_destroy(&wrt);

    return EXIT_SUCCESS;

}
```

Output:



```
nevil11@me:~/Desktop/OS LAB 10$ gcc -pthread readersPriority.c
nevil11@me:~/Desktop/OS LAB 10$ ./a.out
Reader 1: read cnt as 1
Reader 2: read cnt as 1
Reader 4: read cnt as 1
Reader 3: read cnt as 1
Reader 5: read cnt as 1
Reader 6: read cnt as 1
Reader 7: read cnt as 1
Reader 8: read cnt as 1
Reader 9: read cnt as 1
Reader 10: read cnt as 1
Writer 1 modified cnt to 2
Writer 2 modified cnt to 4
Writer 3 modified cnt to 8
Writer 4 modified cnt to 16
Writer 5 modified cnt to 32
Reader 1: read cnt as 32
Reader 2: read cnt as 32
Reader 4: read cnt as 32
Reader 6: read cnt as 32
Reader 5: read cnt as 32
Reader 7: read cnt as 32
Reader 8: read cnt as 32
Reader 3: read cnt as 32
Reader 10: read cnt as 32
Reader 9: read cnt as 32
Writer 1 modified cnt to 64
Writer 2 modified cnt to 128
Writer 3 modified cnt to 256
Writer 4 modified cnt to 512
Writer 5 modified cnt to 1024
```

Handle Overflow :



```
Reader 10: read cnt as 33554432
Writer 1 modified cnt to 67108864
Writer 5 modified cnt to 134217728
Writer 3 modified cnt to 268435456
Writer 2 modified cnt to 536870912
Writer 4 modified cnt to 1073741824
Reader 1: read cnt as 1073741824
Reader 9: read cnt as 1073741824
Reader 7: read cnt as 1073741824
Reader 10: read cnt as 1073741824
Reader 5: read cnt as 1073741824
Reader 2: read cnt as 1073741824
Reader 4: read cnt as 1073741824
Reader 6: read cnt as 1073741824
Reader 3: read cnt as 1073741824
Reader 8: read cnt as 1073741824
Writer 1 modified cnt to 1
Writer 5 modified cnt to 2
Writer 3 modified cnt to 4
Writer 2 modified cnt to 8
Writer 4 modified cnt to 16
Reader 7: read cnt as 16
Reader 5: read cnt as 16
Reader 9: read cnt as 16
Reader 2: read cnt as 16
Reader 1: read cnt as 16
Reader 10: read cnt as 16
Reader 4: read cnt as 16
Reader 6: read cnt as 16
Reader 8: read cnt as 16
Reader 3: read cnt as 16
Writer 1 modified cnt to 32
Writer 5 modified cnt to 64
Writer 3 modified cnt to 128
Writer 2 modified cnt to 256
Writer 4 modified cnt to 512
```

Task 2:

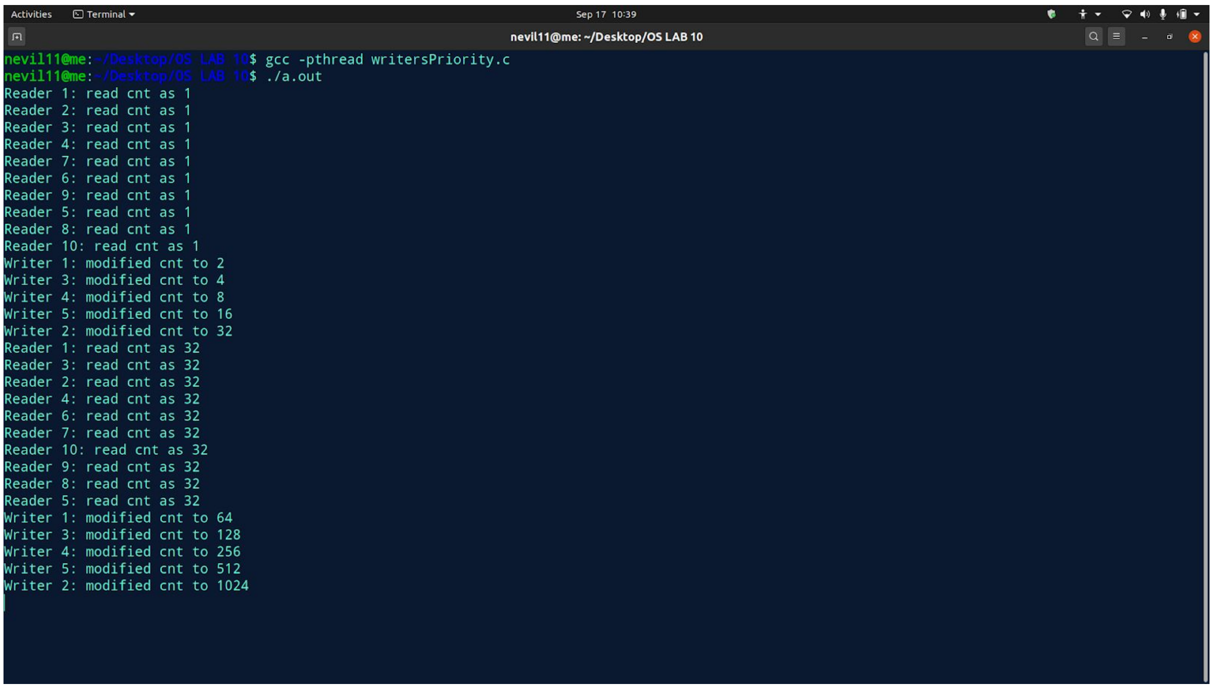
Reader Writers problem (Writers have priority).

Code:

```
/*
* @Author: nevil11
* @Date:   2020-09-17 09:55:41
* @Last Modified by:   nevil11
* @Last Modified time: 2020-09-17 10:35:59
*/
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<limits.h>

int readcount = 0, writecount = 0;
sem_t rsem, wsem ;
pthread_mutex_t x, y, z ;
int cnt = 1;


/* Function Signatures */
void doReading(void *rno);
void doWriting(void *wno);
void reader(void *rno);
void writer(void *wno);

// In real application we can perform some DB
operations
void doReading(void *rno) {
    // Reading Section
    printf("Reader %d: read cnt as %d\n", *((int
*)rno), cnt);
}

void doWriting(void *wno) {
    // since we are running it inside the infinite loop
, the value of cnt may go beyond the limit of 32 bit
int
    if ( ((long long int)cnt * 2) > INT_MAX)
        cnt = 1;
    else
        cnt = cnt * 2;
    printf("Writer %d: modified cnt to %d\n", (*((int
*)wno)), cnt);
}

void reader(void *rno) {
    while (1) {

        pthread_mutex_lock(&z);
        sem_wait(&rsem);
        pthread_mutex_lock(&x);
        readcount++;
        if (readcount == 1)
            sem_wait(&wsem);
        pthread_mutex_unlock(&x);
        sem_post(&rsem);
        pthread_mutex_unlock(&z);

        doReading(rno);

        pthread_mutex_lock(&x);
        readcount--;
        if (readcount == 0)
            sem_post(&wsem);
        pthread_mutex_unlock(&x);

        // sleep is used just to slow down the output
on the terminal
        sleep(2);

    }
}

void writer(void *wno) {
    while (1) {

        pthread_mutex_lock(&y);
        writecount++;
        if (writecount == 1)
            sem_wait(&rsem);
        pthread_mutex_unlock(&y);

        sem_wait(&wsem);
        doWriting(wno);
        sem_post(&wsem);

        pthread_mutex_lock(&y);
        writecount--;
        if (writecount == 0)
            sem_post(&rsem);
        pthread_mutex_unlock(&y);

        // sleep is used just to slow down the output
on the terminal
        sleep(2);

    }
}
```

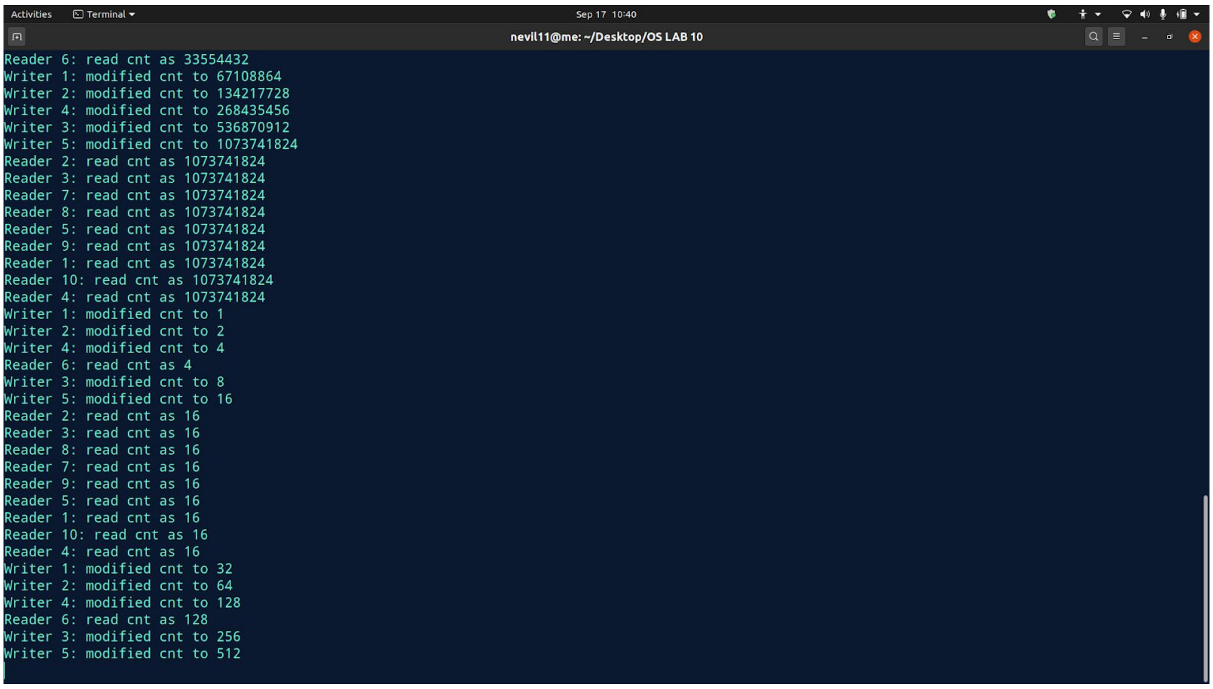
```
int main()
{

    pthread_t read[10], write[5];
    sem_init(&wsem, 0, 1);
    pthread_mutex_init(&z, NULL);
    pthread_mutex_init(&y, NULL);
    pthread_mutex_init(&x, NULL);
    sem_init(&rsem, 0, 1);

    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; //Just
used for numbering the producer and consumer

    for (int i = 0; i < 10; i++) {
        pthread_create(&read[i], NULL, (void *)reader,
(void *)&a[i]);
    }
    for (int i = 0; i < 5; i++) {
        pthread_create(&write[i], NULL, (void *)writer,
(void *)&a[i]);
    }

    for (int i = 0; i < 10; i++) {
        pthread_join(read[i], NULL);
    }
    for (int i = 0; i < 5; i++) {
        pthread_join(write[i], NULL);
    }

    sem_destroy(&wsem);
    pthread_mutex_destroy(&z);
    sem_destroy(&rsem);
    pthread_mutex_destroy(&y);
    pthread_mutex_destroy(&x);

    return EXIT_SUCCESS;

}
```

Output:



```
nevil11@me:~/Desktop/OS LAB 10$ gcc -pthread writersPriority.c
nevil11@me:~/Desktop/OS LAB 10$ ./a.out
Reader 1: read cnt as 1
Reader 2: read cnt as 1
Reader 3: read cnt as 1
Reader 4: read cnt as 1
Reader 7: read cnt as 1
Reader 6: read cnt as 1
Reader 9: read cnt as 1
Reader 5: read cnt as 1
Reader 8: read cnt as 1
Reader 10: read cnt as 1
Writer 1: modified cnt to 2
Writer 3: modified cnt to 4
Writer 4: modified cnt to 8
Writer 5: modified cnt to 16
Writer 2: modified cnt to 32
Reader 1: read cnt as 32
Reader 3: read cnt as 32
Reader 2: read cnt as 32
Reader 4: read cnt as 32
Reader 6: read cnt as 32
Reader 7: read cnt as 32
Reader 10: read cnt as 32
Reader 9: read cnt as 32
Reader 8: read cnt as 32
Reader 5: read cnt as 32
Writer 1: modified cnt to 64
Writer 3: modified cnt to 128
Writer 4: modified cnt to 256
Writer 5: modified cnt to 512
Writer 2: modified cnt to 1024
```

Handle Overflow :



```
Reader 6: read cnt as 33554432
Writer 1: modified cnt to 67108864
Writer 2: modified cnt to 134217728
Writer 4: modified cnt to 268435456
Writer 3: modified cnt to 536870912
Writer 5: modified cnt to 1073741824
Reader 2: read cnt as 1073741824
Reader 3: read cnt as 1073741824
Reader 7: read cnt as 1073741824
Reader 8: read cnt as 1073741824
Reader 5: read cnt as 1073741824
Reader 9: read cnt as 1073741824
Reader 1: read cnt as 1073741824
Reader 10: read cnt as 1073741824
Reader 4: read cnt as 1073741824
Writer 1: modified cnt to 1
Writer 2: modified cnt to 2
Writer 4: modified cnt to 4
Reader 6: read cnt as 4
Writer 3: modified cnt to 8
Writer 5: modified cnt to 16
Reader 2: read cnt as 16
Reader 3: read cnt as 16
Reader 8: read cnt as 16
Reader 7: read cnt as 16
Reader 9: read cnt as 16
Reader 5: read cnt as 16
Reader 1: read cnt as 16
Reader 10: read cnt as 16
Reader 4: read cnt as 16
Writer 1: modified cnt to 32
Writer 2: modified cnt to 64
Writer 4: modified cnt to 128
Reader 6: read cnt as 128
Writer 3: modified cnt to 256
Writer 5: modified cnt to 512
```

Nevil Parmar
CE-092
https://nevilparmar.me