**Name**: Akshat Rajesh Andhale **Class**: TEIT

**Roll** **No**.: 03

**Assignment no. 04(B)**: Reader-Writer problem with reader priority

**CODE**:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

// no. of readers

int rcnt=0;

// no. of writers

int wcnt=0;

// no. of readers waiting

int waitr=0;

// no. of writers waiting

int waitw=0;

// condition variable to check whether reader can read

pthread\_cond\_t canread;

// condition variable to check whether writer can write

pthread\_cond\_t canwrite;

// mutex for synchronization

pthread\_mutex\_t condlock;

pthread\_t r[5], w[5];

int id[5];

// mutex provide synchronization so that no other thread

// can change the value of data

void beginread(int i)

{

pthread\_mutex\_lock(&condlock);

// if there are active or waiting writers

if (wcnt == 1 || waitw > 0)

{

// incrementing waiting readers

waitr++;

// reader suspended

pthread\_cond\_wait(&canread, &condlock);

waitr--;

}

// else reader reads the resource

rcnt++;

printf("reader %d is reading\n", i);

pthread\_mutex\_unlock(&condlock);

pthread\_cond\_broadcast(&canread);

}

void endread(int i)

{

// if there are no readers left then writer enters monitor

pthread\_mutex\_lock(&condlock);

if (--rcnt == 0)

pthread\_cond\_signal(&canwrite);

pthread\_mutex\_unlock(&condlock);

}

void beginwrite(int i)

{

pthread\_mutex\_lock(&condlock);

// a writer can enter when there are no active

// or waiting readers or other writer

if (wcnt == 1 || rcnt > 0)

{

++waitw;

pthread\_cond\_wait(&canwrite, &condlock);

--waitw;

}

wcnt = 1;

printf("writer %d is writing\n", i);

pthread\_mutex\_unlock(&condlock);

}

void endwrite(int i)

{

pthread\_mutex\_lock(&condlock);

wcnt = 0;

// if any readers are waiting, threads are unblocked

if (waitr > 0)

pthread\_cond\_signal(&canread);

else

pthread\_cond\_signal(&canwrite);

pthread\_mutex\_unlock(&condlock);

}

void \*reader(void\* id)

{

int c = 0;

int i = \*(int\*)id;

// each reader attempts to read 5 times

while (c < 5)

{

usleep(1);

beginread(i);

endread(i);

c++;

}

}

void \*writer(void\* id)

{

int c = 0;

int i = \*(int\*)id;

// each writer attempts to write 5 times

while (c < 5)

{

usleep(1);

beginwrite(i);

endwrite(i);

c++;

}

}

int main()

{

pthread\_cond\_init(&canread, NULL);

pthread\_cond\_init(&canwrite, NULL);

pthread\_mutex\_init(&condlock, NULL);

for (int i = 0; i < 5; i++)

{

id[i] = i;

// creating threads which execute reader function

pthread\_create(&r[i], NULL, &reader, &id[i]);

// creating threads which execute writer function

pthread\_create(&w[i], NULL, &writer, &id[i]);

}

for (int i = 0; i < 5; i++)

{

pthread\_join(r[i], NULL);

}

for (int i = 0; i < 5; i++)

{

pthread\_join(w[i], NULL);

}

}

**OUTPUT**:

writer 0 is writing

reader 1 is reading

reader 2 is reading

writer 2 is writing

writer 3 is writing

writer 1 is writing

reader 4 is reading

reader 3 is reading

reader 4 is reading

reader 0 is reading

writer 3 is writing

writer 2 is writing

writer 3 is writing

reader 2 is reading

reader 1 is reading

reader 0 is reading

reader 4 is reading

writer 1 is writing

writer 0 is writing

reader 2 is reading

writer 4 is writing

reader 0 is reading

writer 1 is writing

reader 1 is reading

reader 4 is reading

writer 3 is writing

writer 2 is writing

writer 0 is writing

writer 1 is writing

reader 3 is reading

writer 4 is writing

reader 1 is reading

reader 2 is reading

reader 0 is reading

writer 4 is writing

reader 0 is reading

reader 2 is reading

reader 4 is reading

reader 1 is reading

reader 3 is reading

writer 2 is writing

writer 0 is writing

writer 3 is writing

writer 2 is writing

writer 0 is writing

writer 1 is writing

reader 3 is reading

writer 4 is writing

writer 4 is writing

reader 3 is reading

...Program finished with exit code 0

Press ENTER to exit console.