P-Threads

VARIABLES DE CONDICIÓN & Mutexes

CON

Usando LINKED LISTS

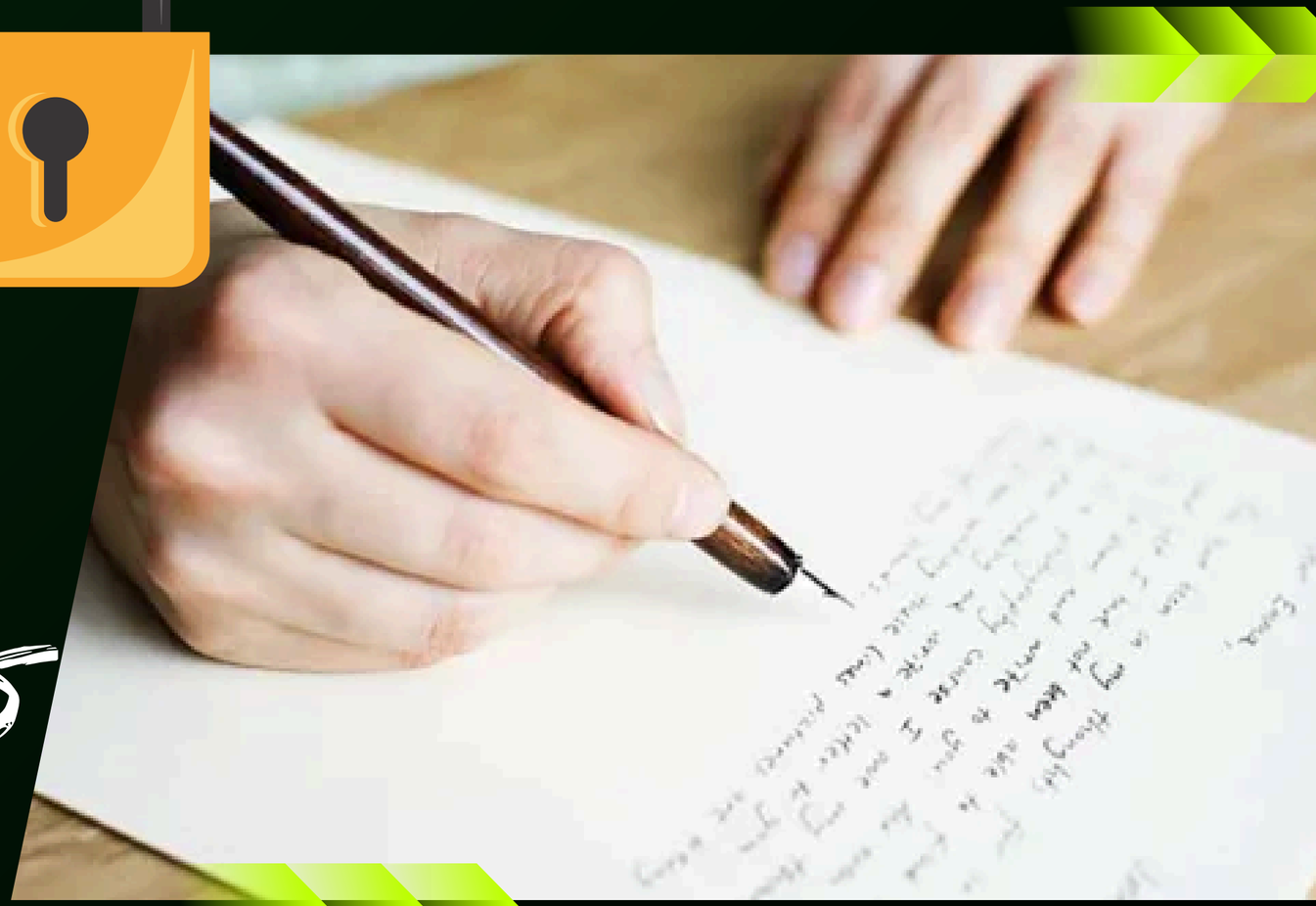
JORGE LÓPEZ - 12141356
GERARDO DÍAZ - 12141095
VÍCTOR ROMERO - 12211079

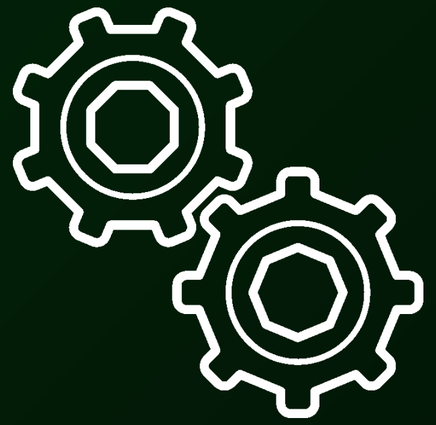
Home Game About Us Contact

P-Threads, CONDITION VARS & Mutexes

Home Game About Us Contact

Read WRITE & Locks





Program 4.6

Write a Pthreads program that uses two condition variables and a mutex to implement a read-write lock. Download the online linked list program that uses Pthreads read-write locks, and modify it to use your read-write locks. Now compare the performance of the program when readers are given pref- erence with the program when writers are given preference.

Can you make any generalizations?

Lista.h

Define una lista enlazada **(list_t)** que utiliza un bloqueo de lectura-escritura **(rwlock_t)** para proteger el acceso concurrente, junto con funciones para inicializar, insertar, eliminar y mostrar elementos de la lista.

```c
#ifndef LIST_H
#define LIST_H

#include "rwlock.h"

typedef struct node {
    int data;
    struct node *next;
} node_t;

typedef struct {
    node_t *head;
    rwlock_t lock;
} list_t;

void list_init(list_t *list);
void list_insert(list_t *list, int data);
void list_delete(list_t *list, int data);
void list_print(list_t *list);

#endif
```

RWlock.h

```c
#ifndef RWLOCK_H
#define RWLOCK_H

#include <pthread.h>

// Read-Write Lock structure
typedef struct {
    pthread_mutex_t mutex;
    pthread_cond_t read_cond, write_cond;
    int readers, writers;
} rwlock_t;

void rwlock_init(rwlock_t *lock);
void rwlock_rlock(rwlock_t *lock);
void rwlock_runlock(rwlock_t *lock);
void rwlock_wlock(rwlock_t *lock);
void rwlock_wunlock(rwlock_t *lock);

#endif // RWLOCK_H
```

Define una estructura de bloqueo de lectura-escritura **(rwlock_t)** con mutex y variables de condición, y funciones para inicializar el bloqueo, bloquear para lectura/escritura y desbloquear.

Main.c

Este código crea una **lista enlazada** protegida con un bloqueo de lectura-escritura, donde **dos hilos** (*Uno para insertar y otro para eliminar*) acceden concurrentemente a la lista. El primer hilo inserta los valores del 0 al 9, y el segundo hilo elimina esos valores, cada uno durmiendo entre inserciones o eliminaciones.

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "list.h"

// Insert function
void *insert_func(void *arg) {
    list_t *list = (list_t *)arg;
    for (int i = 0; i < 10; i++) {
        list_insert(list, i);
        sleep(1);
    }
    return NULL;
}

// Delete function
void *delete_func(void *arg) {
    sleep(2);
    list_t *list = (list_t *)arg;
    for (int i = 0; i < 10; i++) {
        list_delete(list, i);
        sleep(1);
    }
    return NULL;
}


int main() {
    list_t list;
    list_init(&list);

    pthread_t thread1, thread2;
    pthread_create(&thread1, NULL, insert_func, &list);
    pthread_create(&thread2, NULL, delete_func, &list);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    return 0;
```

Conclusiones

El code nos muestra una implementación de una lista enlazada con operaciones concurrentes, las cuales están protegidas por un lock de read-write, permitiendo la inserción y eliminación seguras por múltiples hilos.

Gracias por

su Atención