Системне програмне забезпечення. Лабораторна робота 3

Мета роботи: Закріпити розуміння базової реалізації примітивів синхронізації

Частина 1: Запуск помилкової реалізації багатопотокового лічильника

Скомпілювати та запустити наступний код, переконатися що лічильники відпрацьовують з помилкою.

Компілювати командою:
```
# gcc -Wall -pthread -o lab3 lab3.c
```

Код прикладу:
```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

static const unsigned long nthreads = 4;
static const unsigned long ncounts = 10000000;

// Trivial counter implementation
static unsigned long counter = 0;

void counter_inc ()
{
     counter++;
}

unsigned long read_counter ()
{
     return counter;
}

// Thread routine
void * threadfn (void *arg)
{
     int i;
     for (i = 0; i < ncounts; i++)
          counter_inc ();
     return 0;
}

int main ()
{
```

```
    int i;
    pthread_t pids[nthreads];

    for (i = 0; i < nthreads; i++) {
         int r = pthread_create (&pids[i], NULL,
                                 threadfn, NULL);
         if (r != 0)    {
                   perror ("pthread_create");
                   exit (EXIT_FAILURE);
         }
    }
    for (i = 0; i < nthreads; i++) {
         pthread_join (pids[i], NULL);
    }
    printf ("Expected counter value is: %lu\n",
                                 nthreads * ncounts);
    printf ("Real counter value is: %lu\n", read_counter());
    return 0;
}
```

Частина 2: Виправлення помилкової реалізації багатопотокового лічильника

1. Виправити реалізацію з Частини 1 додавши в функцію counter_inc() синхронізацію з допомогою POSIX мютексів.
2. Виправити реалізацію з Частини 1 зробивши операцію інкремента атомарною. Використовуйте __sync_fetch_and_add().
3. Реалізуйте власні процедури lock() та unlock(). Використовуйте функцію type __sync_fetch_and_or (type *ptr, type value)
   Використайте те що:
   __sync_fetch_and_or(p, 1) , при умові що в пам’яті 0 або 1, веде себе як виклик функції xchng(p, 1)