НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський Політехнічний Інститут
імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра спеціалізованих комп’ютерних систем

ЛАБОРАТОРНА РОБОТА No4
з дисципліни
“Паралельні та розподілені обчислення”

ТЕМА: «Засоби взаємодії паралельних потоків операційної системи Linux»

Виконав:студент групи КВ-62,
Марченко І.О.
Перевірив:

Київ
2018

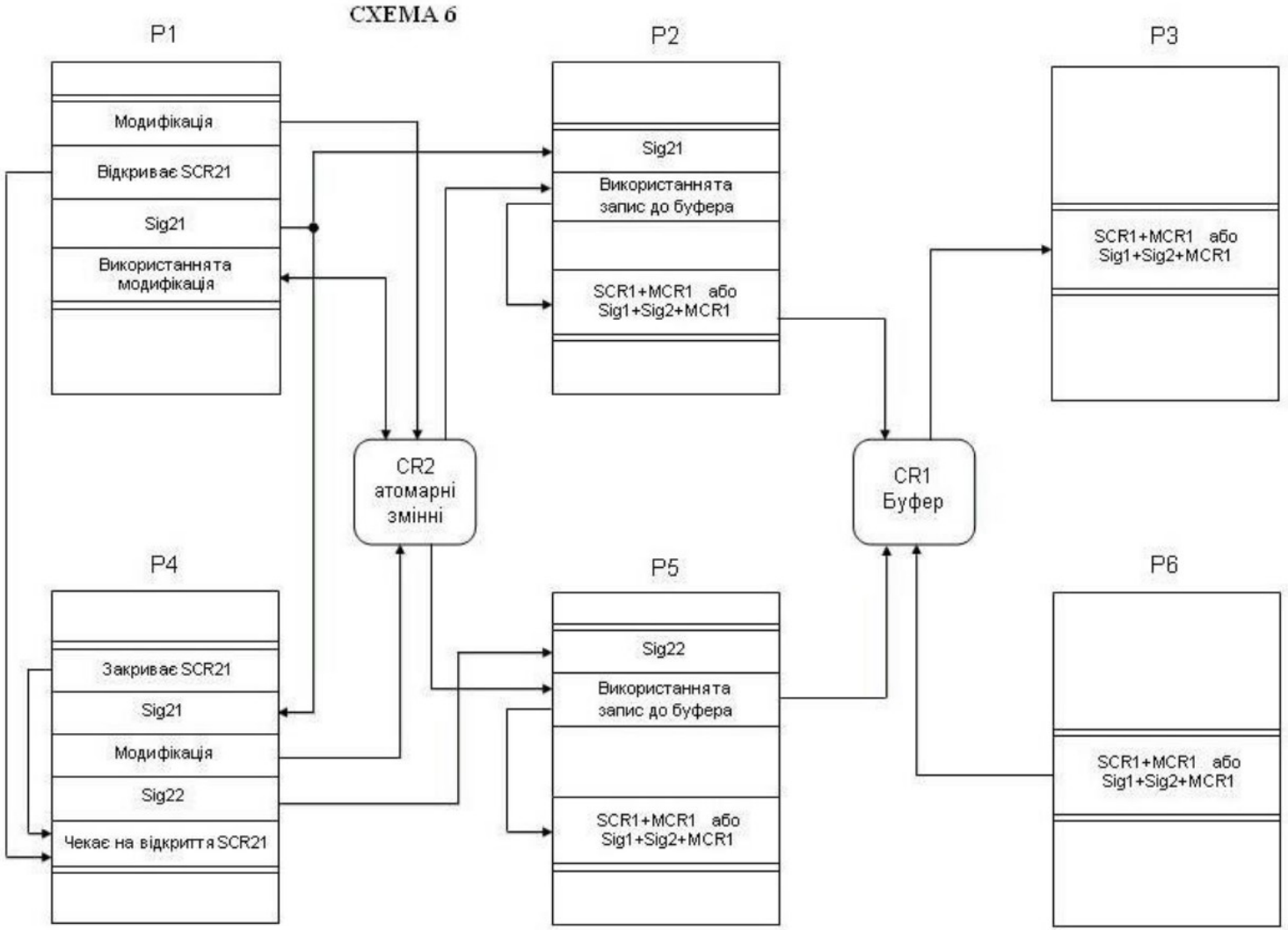
Постановка завдання та вимоги до виконання програми

1. Написати програму, яка реалізує роботу паралельних потоків згідно заданої за варіантом
схеми. Особливості реалізації синхронізації паралельних потоків та взаємного виключення
потоків при доступі до спільних ресурсів задані за варіантами у таблицях 1 та 2.
2. При написанні програми виконати повне трасування роботи програми за допомогою
операторів друку, тобто розставити в програмі оператори друку таким чином, щоб можна
було прослідкувати всі варіанти виконання паралельних потоків і впевнитись у коректності
роботи програми. Протокол трасування рекомендується записувати у файл (log-файл).
3. Запуск усіх потоків повинен бути виконаний у головній програмі.
4. Кожен потік повинен бути організованим у вигляді нескінченного циклу.
5. Всі дії задані за варіантами, що вказані у таблиці, повинні бути виконані всередині цього
нескінченного циклу.
6. Взаємне розташування операторів синхронізації та доступу до спільного ресурсу, якщо вони
знаходяться у одному потоці, є довільним.
7. Оскільки синхронізація за допомогою семафорів SCR21, SCR22 згідно завдання розташована
всередині нескінченних циклів, то відразу після виконання синхронізації ці семафори
повинні бути знову встановлені у початковий закритий стан.
8. Закінчення програми можна виконати двома способами:
• примусовим перериванням за допомогою натиснення комбінації клавіш Ctrl+C;
• оператором break при виконанні умови, яка стає істинною, коли буфер спільного
ресурсу повністю заповнюється і повністю звільняється мінімум по два рази.
9. Якщо при реалізації паралельних потоків була використана функція usleep(), то передбачити
режим запуску програми з «відключеними» функціями usleep().
10. Виконати налагодження написаної програми.

Завдання (варіант №16)

Номер атомарної операції	Ідентифікатор вбудованої функції, що відповідає атомарній операції
1.	type __sync_fetch_and_add (type *ptr, type value)
4.	type __sync_fetch_and_and (type *ptr, type value)
5.	type __sync_fetch_and_xor (type *ptr, type value)
8.	type __sync_sub_and_fetch (type *ptr, type value)
9.	type __sync_or_and_fetch (type *ptr, type value)
10.	type __sync_and_and_fetch (type *ptr, type value)
13.	bool __sync_bool_compare_and_swap(type *ptr, type oldval, type newval)
14.	type __sync_val_compare_and_swap (type *ptr, type oldval, type newval)

СХЕМА 6


P1
Модифікація
Відкриває SCR21
Sig21
Використання модифікація
P2
Sig21
Використання запис до буфера
SCR1+MCR1 або Sig1+Sig2+MCR1
P3
SCR1+MCR1 або Sig1+Sig2+MCR1
CR2 атомарні змінні
CR1 Буфер
P4
Закриває SCR21
Sig21
Модифікація
Sig22
Чекає на відкриття SCR21
P5
Sig22
Використання запис до буфера
SCR1+MCR1 або Sig1+Sig2+MCR1
P6
SCR1+MCR1 або Sig1+Sig2+MCR1

№ варі-анту	№ схеми	Спільний ресурс 1 CR1 (буфер обміну даними)		Спільний ресурс 2 CR2	Засоби синхронізації паралельних потоків				
					Семафори та бар'єр для повної або неповної синхронізації потоків			Сигнальні (умовні) змінні синхронізації потоків	
		Структура даних, що використо-вується у якості спільного ресурсу 1 (CR1)	Засоби взаємного виключення при доступі до спільного ресурсу 1 SCR1 + MCR1 або Sig1+Sig2+MCR1	Атомарні дані (взяти по 2 змінних кож-ного з типів: *int, unsigned, long, long unsigned*) та операції (таблиця 2)	Вид двійкового семафору SCR21	Вид двійкового семафору SCR22	Бар'єр BCR2	Вид сигналу сигнальної (умовної) змінної Sig21	Вид сигналу сигнальної (умовної) змінної Sig22
16	6	Черга (ДСД)	Дві сигнальні (умовні) змінні Sig1 та Sig2 і блокуючий м'ютекс MCR1	1, 8, 4, 5, 9, 12, 13, 14	Блокуючий	–	–	Багато-значний	Одиничний

Текст програми:

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <stdbool.h>
#include <semaphore.h>
#include <unistd.h>
#include <pthread.h>

int sum = 0;

int a = 9, b = 3;
unsigned c = 15, d = 16;
long e = 20, f = 1;
unsigned long g = 5, h = 6;
bool fl = false;

FILE *file;

/* Оголошення та ініціалізація сигнальних (умовних) змінних. */
pthread_cond_t sig1 = PTHREAD_COND_INITIALIZER;
pthread_cond_t sig2 = PTHREAD_COND_INITIALIZER;
pthread_cond_t sig21 = PTHREAD_COND_INITIALIZER;
pthread_cond_t sig22 = PTHREAD_COND_INITIALIZER;


/* Оголошення та ініціалізація м'ютекса */
pthread_mutex_t mcr1 = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mut1 = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mut2 = PTHREAD_MUTEX_INITIALIZER;


/* прапорець для запам'ятовування факту того, що буфер вже був звільнений
чи заповнений*/
int empty = 0, full = 0;
```

```
int sig_21 = 0, sig_22 = 0;

/* Оголошення семафору. */
sem_t  scr21;

/*Оголошення змінних для потоків*/
pthread_t thread1;
pthread_t thread2;
pthread_t thread3;
pthread_t thread4;
pthread_t thread5;
pthread_t thread6;

int max_queue_length = 10;
int curr_queue_length = 0;

/*************************************************************************/
/* Опис структур даних реалізації спільного буфера (черги)
*************************************************************************/

typedef struct Elem{
    int value;
    struct Elem *next;
} el;

el *beg_q = NULL;
el *end_q = NULL;

bool is_empty(){
    return (beg_q == NULL)?true:false;
}

bool is_full(){
    return (curr_queue_length == 10)?true:false;
}

void add_elem(){
    if(beg_q == NULL){
        beg_q = (el*) malloc(sizeof(el));
        beg_q->value = curr_queue_length + 1;
        beg_q->next = NULL;
        end_q = beg_q;
    }
    else{
        end_q->next = (el*) malloc(sizeof(el));;
        end_q = end_q->next;
        end_q->value = curr_queue_length + 1;
        end_q->next = NULL;
    }
}

el *get_elem(){
    if(beg_q != NULL){
        el *p = beg_q;
        beg_q = beg_q->next;
        return p;
    }
    else{
```

```
        end_q = beg_q;
        return NULL;
    }
}

void summ(int value){
    sum+= value;
}

void *thread_1(void *arg){
    while(1){
        fprintf(file, "Function \"__sync_fetch_and_add\" was called by
the thread No 1. The return value is %d;\n", __sync_fetch_and_add(&a,
5));
        fprintf(file, "Thread 1: atomic_int variable a  after
function \"__sync_fetch_and_add\" is %d\n", a);
        fprintf(file, "Function \"__sync_or_and_fetch\" was called by the
thread No 1. The return value is %ld:\n", __sync_or_and_fetch(&e, 10));
        fprintf(file, "Thread 1: atomic_long variable e after
function \"__sync_or_and_fetch\" is %ld\n", e);
        fprintf(file, "Function \"__sync_val_compare_and_swap\" was
called by the thread No 1. The result value is %u;\n",
__sync_val_compare_and_swap(&c, c, 2));
        fprintf(file, "Thread 1: atomic_uint variable c after
function \"__sync_val_compare_and_swap\" is %u\n", c);
        //fprintf(file, "Function \"__sync_fetch_and_xor\" was called by
the thread No 1. The result value is %lu;\n", __sync_fetch_and_xor(&g,
17));
       // fprintf(file, "Thread 1: atomic_ulong variable g  after
function \"__sync_fetch_and_xor\" is %lu\n", g);
       // fprintf(file, "Function \"__sync_fetch_and_and\" was called by
the thread No 1. The result value is %d;\n", __sync_fetch_and_and(&b,
4));
       // fprintf(file, "Thread 1: atomic_int variable b after function
is \"__sync_fetch_and_and\" %d\n", b);
       // fprintf(file, "Function \"__sync_sub_and_fetch\" was called by
the thread No 1. The result value is %ld;\n", __sync_sub_and_fetch(&f,
1));
        //fprintf(file, "Thread 1: atomic_long variable f after
function \"__sync_sub_and_fetch\" is %ld\n", f);
        //fprintf(file, "Function \"__sync_nand_and_fetch\" was called by
the thread No 1. The result value is %u;\n", __sync_nand_and_fetch(&d,
77));
        //fprintf(file, "Thread 1: atomic_uint variable d after
function \"__sync_nand_and_fetch\" is %u\n", d);
       // bool tmp = false;
       // fprintf(file, "Function \"__sync_bool_compare_and_swap\" was
called by the thread No 1. The result value is %s;\n", ((tmp =
__sync_bool_compare_and_swap(&fl, fl, true))?"true":"false"));
       // fprintf(file, "Thread 1: atomic_bool variable fl after function
\"__sync_bool_compare_and_swap\" is %s", (fl?"true\n":"false\n"));
        sem_post(&scr21);
        fprintf(file, "The semaphore scr21 is opened by the thread 1\n");
     //   printf("The semaphore scr21 is opened by the thread 1\n");
        pthread_mutex_lock(&mut1);
        fprintf(file, "The thread 1 has locked the mutex mut1\n");
    //    printf("The thread 1 has locked the mutex mut1\n");
        pthread_cond_broadcast(&sig21);
```

```
        fprintf(file, "The signal sig21 was repeatedly send from the
thread 1\n");
     //   printf("The signal sig21 was repeatedly send from the thread
1\n");
        sig_21 = 2;
        fprintf(file, "The thread 1 has unlocked the mutex mut1\n");
       // printf("The thread 1 has unlocked the mutex mut1\n");
        pthread_mutex_unlock(&mut1);
        int s1 = 0;
        fprintf(file, "Thread 1: The sum of a and b is %d\n", s1 = a +
b);
        unsigned s2 = 0;
        fprintf(file, "Thread 1: The sum of c and d is %u\n", s2 = c +
d);
        long s3 = 0;
        fprintf(file, "Thread 1: The sum of e and f is %ld\n", s3 = e +
f);
        unsigned long s4 = 0;
        fprintf(file, "Thread 1: The sum of g and h is %lu\n", s4 = g +
h);
        bool bl = false;
        fprintf(file, "Thread 1: The value of flag is %s", ((bl =
fl)?"true\n":"false\n"));
        //fprintf(file, "Function \"__sync_fetch_and_add\" was called by
the thread No 1. The return value is %d;\n", __sync_fetch_and_add(&a,
5));
        //fprintf(file, "Thread 1: atomic_int variable a  after
function \"__sync_fetch_and_add\" is %d\n", a);
        //fprintf(file, "Function \"__sync_or_and_fetch\" was called by
the thread No 1. The return value is %ld:\n", __sync_or_and_fetch(&e,
10));
       // fprintf(file, "Thread 1: atomic_long variable e after
function \"__sync_or_and_fetch\" is %ld\n", e);
       // fprintf(file, "Function \"__sync_val_compare_and_swap\" was
called by the thread No 1. The result value is %u;\n",
__sync_val_compare_and_swap(&c, c, 2));
        //fprintf(file, "Thread 1: atomic_uint variable c after
function \"__sync_val_compare_and_swap\" is %u\n", c);
        fprintf(file, "Function \"__sync_fetch_and_xor\" was called by
the thread No 1. The result value is %lu;\n", __sync_fetch_and_xor(&g,
17));
        fprintf(file, "Thread 1: atomic_ulong variable g  after
function \"__sync_fetch_and_xor\" is %lu\n", g);
        fprintf(file, "Function \"__sync_fetch_and_and\" was called by
the thread No 1. The result value is %d;\n", __sync_fetch_and_and(&b,
4));
        fprintf(file, "Thread 1: atomic_int variable b after function  is
\"__sync_fetch_and_and\" %d\n", b);
        fprintf(file, "Function \"__sync_sub_and_fetch\" was called by
the thread No 1. The result value is %ld;\n", __sync_sub_and_fetch(&f,
1));
        fprintf(file, "Thread 1: atomic_long variable f after
function \"__sync_sub_and_fetch\" is %ld\n", f);
        //fprintf(file, "Function \"__sync_nand_and_fetch\" was called by
the thread No 1. The result value is %u;\n", __sync_nand_and_fetch(&d,
77));
        //fprintf(file, "Thread 1: atomic_uint variable d after
function \"__sync_nand_and_fetch\" is %u\n", d);
```

```
        //tmp = false;
       // fprintf(file, "Function \"__sync_bool_compare_and_swap\" was
called by the thread No 1. The result value is %s;\n", ((tmp =
__sync_bool_compare_and_swap(&fl, fl, true))?"true":"false"));
       //fprintf(file, "Thread 1: atomic_bool variable fl after
function \"__sync_bool_compare_and_swap\" is %s",
(fl?"true\n":"false\n"));
        if(*(int*)arg == 1)
            usleep(600);
    }
    return NULL;
}

void *thread_2(void *arg){
    while(1){
        pthread_mutex_lock(&mut1);
        fprintf(file, "The thread 2 has locked the mutex mut1\n");
    //    printf("The thread 2 has locked the mutex mut1\n");
        if(sig_21 == 0){
            fprintf(file, "The thread 2 is waiting for a signal sig
21\n");
      //      printf("The thread 2 is waiting for a signal sig 21\n");
            while(sig_21 == 0)
                pthread_cond_wait(&sig21, &mut1);
        }
        sig_21--;
        fprintf(file, "The thread 2 has unlocked the mutex mut1\n");
     //   printf("The thread 2 has unlocked the mutex mut1\n");
        pthread_mutex_unlock(&mut1);
        int s1 = 0;
        fprintf(file, "Thread 2: The sum of a and b is %d\n", s1 = a +
b);
        unsigned s2 = 0;
        fprintf(file, "Thread 2: The sum of c and d is %u\n", s2 = c +
d);
        long s3 = 0;
        fprintf(file, "Thread 2: The sum of e and f is %ld\n", s3 = e +
f);
        unsigned long s4 = 0;
        fprintf(file, "Thread 2: The sum of g and h is %lu\n", s4 = g +
h);
        bool bl = false;
        fprintf(file, "Thread 2: The value of flag is %s", ((bl =
fl)?"true\n":"false\n"));
        pthread_mutex_lock(&mcr1);
        fprintf(file, "The thread 2 has locked the mutex mcr1\n");
     //   printf("The thread 2 has locked the mutex mcr1\n");
        while(is_full() == true){
            fprintf(file, "The thread 2 is waiting for a signal sig2\n");
         //   printf("The thread 2 is waiting for a signal sig2\n");
            pthread_cond_wait(&sig2, &mcr1);
        }
        add_elem();
        fprintf(file, "Thread 2 has added the element to the queue with
the number %d\n", curr_queue_length);
        printf("Thread 2 has added the element to the queue with the
number %d\n", curr_queue_length);
        curr_queue_length++;
```

```
        fprintf(file, "The thread 2 has unlocked the mutex mcr1\n");
        printf("The thread 2 has unlocked the mutex mcr1\n");
        pthread_mutex_unlock(&mcr1);

/* Повідомляємо про те, що в черзі з'явилося нове завдання.
 * Якщо потоки-споживачі заблоковані в очікуванні цього сигналу,
 * то один з них буде розблоковано для обробки нового завдання. */
        pthread_cond_signal(&sig1);
        fprintf(file, "The signal sig1 was send from the thread 2\n");
      //  printf("The signal sig1 was send from the thread 2\n");
        if(*(int*)arg == 1)
            usleep(96);
    }

    return NULL;
}

void *thread_3(void *arg){
    while(1){
        pthread_mutex_lock(&mcr1);
        fprintf(file, "The thread 3 has locked the mutex mcr1\n");
        if(is_empty() == true){
            empty = empty + 1;
            if(empty >= 2){
                fprintf(file, "The CR1 was EMPTY AT LEAST TWICE\n");
               printf("The CR1 was EMPTY AT LEAST TWICE\n");
            }
            else{
               fprintf(file, "The CR1 was EMPTY ONCE\n");
               printf("The CR1 was EMPTY ONCE\n");
            }
        }
        if(is_full() == true){
                full = full + 1;
            if(full >=2){
                fprintf(file, "The CR1 was FULL AT LEAST TWICE\n");
                printf("The CR1 was FULL AT LEAST TWICE\n");
            }
            else{
                fprintf(file, "The CR1 was FULL ONCE\n");
                printf("The CR1 was FULL ONCE\n");
            }
        }
        printf("The thread 3 has locked the mutex mcr1\n");
        if((empty >= 2) && (full >= 2 )){
            fprintf(file, "The thread 3 has unlocked the mutex mcr1\n");
            printf("The thread 3 has unlocked the mutex mcr1\n");
            pthread_mutex_unlock(&mcr1);
            break;
        }
        while(is_empty() == true){
            fprintf(file, "The thread 3 is waiting for a signal sig1\n");
            printf("The thread 3 is waiting for a signal sig1\n");
            pthread_cond_wait(&sig1, &mcr1);
        }
        if(is_full() == true){
            full = full + 1;
            if(full >=2){
```

```
                fprintf(file, "The CR1 was FULL AT LEAST TWICE\n");
                printf("The CR1 was FULL AT LEAST TWICE\n");
            }
            else{
                fprintf(file, "The CR1 was FULL ONCE\n");
                printf("The CR1 was FULL ONCE\n");
            }
        }
        el *tmp = get_elem();
        curr_queue_length--;
        fprintf(file, "Thread 3 has taken the element from the queue with
the number %d\n", curr_queue_length);
        printf("Thread 3 has taken the element from the queue with the
number %d\n", curr_queue_length);
        if(is_empty() == true){
            empty = empty + 1;
            if(empty >= 2){
                fprintf(file, "The CR1 was EMPTY AT LEAST TWICE\n");
               printf("The CR1 was EMPTY AT LEAST TWICE\n");
            }
            else{
               fprintf(file, "The CR1 was EMPTY ONCE\n");
               printf("The CR1 was EMPTY ONCE\n");
            }
        }
        summ(tmp->value);
        free(tmp);
        fprintf(file, "The thread 3 has unlocked the mutex mcr1\n");
        printf("The thread 3 has unlocked the mutex mcr1\n");
        pthread_mutex_unlock(&mcr1);

/* Повідомляємо про те, що до черги можна додати хоча б одне нове
завдання.
 * Якщо потоки-постачальники заблоковані в очікуванні сигналу,
 * то один з них буде розблоковано для додавання нового завдання. */
      pthread_cond_broadcast (&sig2);
        fprintf(file, "The signal sig2 was repeatedly send from the
thread 3\n");
        printf("The signal sig2 was repeatedly send from the thread
3\n");

        if(*(int*)arg == 1)
            usleep(25);
    }

    /* Відміна всіх інщих потоків*/
    pthread_cancel(thread1);
    fprintf(file, "Thread 1 is cancelled\n");
    pthread_cancel(thread2);
    fprintf(file, "Thread 2 is cancelled\n");
    pthread_cancel(thread4);
    fprintf(file, "Thread 4 is cancelled\n");
    pthread_cancel(thread5);
    fprintf(file, "Thread 5 is cancelled\n");
    pthread_cancel(thread6);
    fprintf(file, "Thread 6 is cancelled\n");
```

```
    fprintf(file, "Consumer thread 3 of the data, stored in the CR1 is
stopped !!!\n");

    return NULL;
}

void *thread_4(void *arg){
    while(1){
        sem_close(&scr21);
        fprintf(file, "The semaphore scr21 is closed by the thread 4\n");
        printf("The semaphore scr21 is closed by the thread 4\n");
        pthread_mutex_lock(&mut1);
        fprintf(file, "The thread 4 has locked the mutex mut1\n");
        printf("The thread 4 has locked the mutex mut1\n");
        if(sig_21 == 0){
            fprintf(file, "The thread 4 is waiting for a signal sig
21\n");
            printf("The thread 4 is waiting for a signal sig 21\n");
            while(sig_21 == 0)
                pthread_cond_wait(&sig21, &mut1);
        }
        sig_21--;
        fprintf(file, "The thread 4 has unlocked the mutex mut1\n");
        printf("The thread 4 has unlocked the mutex mut1\n");
        pthread_mutex_unlock(&mut1);
       // fprintf(file, "Function \"__sync_fetch_and_add\" was called by
the thread No 4. The return value is %d;\n", __sync_fetch_and_add(&a,
5));
        //fprintf(file, "Thread 4: atomic_int variable a  after
function \"__sync_fetch_and_add\" is %d\n", a);
       // fprintf(file, "Function \"__sync_or_and_fetch\" was called by
the thread No 4. The return value is %ld:\n", __sync_or_and_fetch(&e,
10));
        //fprintf(file, "Thread 4: atomic_long variable e after
function \"__sync_or_and_fetch\" is %ld\n", e);
        //fprintf(file, "Function \"__sync_val_compare_and_swap\" was
called by the thread No 4. The result value is %u;\n",
__sync_val_compare_and_swap(&c, c, 2));
        //fprintf(file, "Thread 4: atomic_uint variable c after
function \"__sync_val_compare_and_swap\" is %u\n", c);
        //fprintf(file, "Function \"__sync_fetch_and_xor\" was called by
the thread No 4. The result value is %lu;\n", __sync_fetch_and_xor(&g,
17));
        //fprintf(file, "Thread 4: atomic_ulong variable g  after
function \"__sync_fetch_and_xor\" is %lu\n", g);
        //fprintf(file, "Function \"__sync_fetch_and_and\" was called by
the thread No 4. The result value is %d;\n", __sync_fetch_and_and(&b,
4));
        //fprintf(file, "Thread 4: atomic_int variable b after function
is \"__sync_fetch_and_and\" %d\n", b);
        //fprintf(file, "Function \"__sync_sub_and_fetch\" was called by
the thread No 4. The result value is %ld;\n", __sync_sub_and_fetch(&f,
1));
        //fprintf(file, "Thread 4: atomic_long variable f after
function \"__sync_sub_and_fetch\" is %ld\n", f);
        fprintf(file, "Function \"__sync_nand_and_fetch\" was called by
the thread No 4. The result value is %u;\n", __sync_nand_and_fetch(&d,
77));
```

```
        fprintf(file, "Thread 4: atomic_uint variable d after
function \"__sync_nand_and_fetch\" is %u\n", d);
        bool tmp = false;
        fprintf(file, "Function \"__sync_bool_compare_and_swap\" was
called by the thread No 4. The result value is %s;\n", ((tmp =
__sync_bool_compare_and_swap(&fl, fl, true))?"true":"false"));
        fprintf(file, "Thread 4: atomic_bool variable fl after
function \"__sync_bool_compare_and_swap\" is %s",
(fl?"true\n":"false\n"));
        pthread_mutex_lock(&mut2);
        fprintf(file, "The thread 4 has locked the mutex mut2\n");
     //   printf("The thread 4 has locked the mutex mut2\n");
        pthread_cond_signal(&sig22);
        sig_22 = 1;
        fprintf(file, "The thread 4 sends the signal sig22 to the thread
5\n");
    //    printf("The thread 4 sends the signal sig22 to the thread
5\n");
        fprintf(file, "The thread 4 has unlocked the mutex mut2\n");
    //    printf("The thread 4 has unlocked the mutex mut2\n");
        pthread_mutex_unlock(&mut2);
        sem_wait(&scr21);
        fprintf(file, "The thread 4 is waiting for an open semaphore
scr21\n");
      //  printf("The thread 4 is waiting for an open semaphore
scr21\n");
        if(*(int*)arg == 1)
            usleep(400);
    }
    return NULL;
}

void *thread_5(void *arg){
   while(1){
        pthread_mutex_lock(&mut2);
        fprintf(file, "The thread 5 has locked the mutex mut2\n");
        printf("The thread 5 has locked the mutex mut2\n");
        if(sig_22 != 1){
            fprintf(file, "The thread 5 is waiting for a signal sig
22\n");
            printf("The thread 5 is waiting for a signal sig 22\n");
            while(sig_22 != 1)
                pthread_cond_wait(&sig22, &mut2);
        }
        sig_22 = 0;
        fprintf(file, "The thread 5 has unlocked the mutex mut2\n");
        printf("The thread 5 has unlocked the mutex mut2\n");
        pthread_mutex_unlock(&mut2);
        int s1 = 0;
        fprintf(file, "Thread 5: The sum of a and b is %d\n", s1 = a +
b);
        unsigned s2 = 0;
        fprintf(file, "Thread 5: The sum of c and d is %u\n", s2 = c +
d);
        long s3 = 0;
        fprintf(file, "Thread 5: The sum of e and f is %ld\n", s3 = e +
f);
        unsigned long s4 = 0;
```

```
        fprintf(file, "Thread 5: The sum of g and h is %lu\n", s4 = g +
h);
        bool bl = false;
        fprintf(file, "Thread 5: The value of flag is %s", ((bl =
fl)?"true\n":"false\n"));
        pthread_mutex_lock(&mcr1);
        fprintf(file, "The thread 5 has locked the mutex mcr1\n");
        printf("The thread 5 has locked the mutex mcr1\n");
        while(is_full() == true){
            fprintf(file, "The thread 5 is waiting for a signal sig2\n");
            printf("The thread 5 is waiting for a signal sig2\n");
            pthread_cond_wait(&sig2, &mcr1);
        }
        add_elem();
        fprintf(file, "Thread 5 has added the element to the queue with
the number %d\n", curr_queue_length);
        printf("Thread 5 has added the element to the queue with the
number %d\n", curr_queue_length);
        curr_queue_length++;
        fprintf(file, "The thread 5 has unlocked the mutex mcr1\n");
        printf("The thread 5 has unlocked the mutex mcr1\n");
        pthread_mutex_unlock(&mcr1);

/* Повідомляємо про те, що в черзі з'явилося нове завдання.
 * Якщо потоки-споживачі заблоковані в очікуванні цього сигналу,
 * то один з них буде розблоковано для обробки нового завдання. */
        pthread_cond_signal(&sig1);
        fprintf(file, "The signal sig1 was send from the thread 5\n");
        printf("The signal sig1 was send from the thread 5\n");
        if(*(int*)arg == 1)
            usleep(95);
    }

    return NULL;
}

void *thread_6(void *arg){
    while(1){
        pthread_mutex_lock(&mcr1);
        fprintf(file, "The thread 6 has locked the mutex mcr1\n");
        printf("The thread 6 has locked the mutex mcr1\n");
        while(is_full() == true){
            fprintf(file, "The thread 6 is waiting for a signal sig2\n");
            printf("The thread 6 is waiting for a signal sig2\n");
            pthread_cond_wait(&sig2, &mcr1);
        }
        add_elem();
        fprintf(file, "Thread 6 has added the element to the queue with
the number %d\n", curr_queue_length);
        printf("Thread 6 has added the element to the queue with the
number %d\n", curr_queue_length);
        curr_queue_length++;
        fprintf(file, "The thread 6 has unlocked the mutex mcr1\n");
        printf("The thread 6 has unlocked the mutex mcr1\n");
        pthread_mutex_unlock(&mcr1);

/* Повідомляємо про те, що в черзі з'явилося нове завдання.
 * Якщо потоки-споживачі заблоковані в очікуванні цього сигналу,
```

```
 * то один з них буде розблоковано для обробки нового завдання. */
        pthread_cond_signal(&sig1);
        fprintf(file, "The signal sig1 was send from the thread 6\n");
        printf("The signal sig1 was send from the thread 6\n");
        if(*(int*)arg == 1)
            usleep(97);
    }
    return NULL;
}

int main(){
    int flag = 0;
    while(1){
        printf("Press 1 for threads with usleep, 2 - without usleep\n");
        scanf("%d", &flag);
        if((flag != 1) && (flag != 2))
            printf("Error input. Try again\n");
        else
            break;
    }
    file = fopen("Log.txt", "wt");
    if(file == NULL){
        perror("Log.txt");
        exit(EXIT_FAILURE);
    }

    int i = 0;

/* Створення початкової черги із заданою кількостю елементів
length_at_start
перед запуском потоків*/
    for(i=0;i<5;i++) {
      add_elem();
        fprintf(file, "Main Thread has added the element to the queue
with the number %d\n", curr_queue_length);
        curr_queue_length++;
   }
    fprintf(file, "Queue with elements from 0-th to 4-th has been created
!!!\n");
    fprintf(file, "The current value of the sum is %d\n", sum);

/*Ініціалізується семафор. */
    sem_init(&scr21,0,0);

/* Кожному потоку передається вказівник на його номер, приведений до типу
void*  */
    pthread_create (&thread1,NULL,&thread_1,(void*)&flag);
    pthread_create (&thread2,NULL,&thread_2,(void*)&flag);
    pthread_create (&thread3,NULL,&thread_3,(void*)&flag);
    pthread_create (&thread4,NULL,&thread_4,(void*)&flag);
    pthread_create (&thread5,NULL,&thread_5,(void*)&flag);
    pthread_create (&thread6,NULL,&thread_6,(void*)&flag);

/* Очікуємо завершення всіх потоків */
    pthread_join(thread3,NULL);

    fprintf(file, "All threads stopped !!!\n");
    fprintf(file, "The sum is %d\n", sum);
```

```
    printf("\nAll threads has stopped\n");

    if(beg_q != NULL){
        while(beg_q != NULL)
            get_elem();
    }

    return 0;
}
```

Log-file з результатом виконання програми

```
Main Thread has added the element to the queue with the number 0
Main Thread has added the element to the queue with the number 1
Main Thread has added the element to the queue with the number 2
Main Thread has added the element to the queue with the number 3
Main Thread has added the element to the queue with the number 4
Queue with elements from 0-th to 4-th has been created !!!
The current value of the sum is 0
The thread 2 has locked the mutex mut1
The thread 2 is waiting for a signal sig 21
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 9;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
14
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 15;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
The thread 3 has locked the mutex mcr1
Thread 1: The sum of a and b is 17
Thread 1: The sum of c and d is 18
Thread 1: The sum of e and f is 31
Thread 1: The sum of g and h is 11
Thread 1: The value of flag is false
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 3;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
0
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 14;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
19
```

```
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 3 has taken the element from the queue with the number 4
The thread 2 has unlocked the mutex mut1
Thread 2: The sum of a and b is 19
Thread 2: The sum of c and d is 18
Thread 2: The sum of e and f is 30
Thread 2: The sum of g and h is 26
Thread 2: The value of flag is false
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 19
Thread 1: The sum of c and d is 18
The thread 3 has unlocked the mutex mcr1
Thread 1: The sum of e and f is 30
Thread 1: The sum of g and h is 26
Thread 1: The value of flag is false
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 20;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 5
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -1;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-1
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 19;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
24
The signal sig2 was repeatedly send from the thread 3
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 24
Thread 1: The sum of c and d is 18
Thread 1: The sum of e and f is 29
Thread 1: The sum of g and h is 11
```

```
Thread 1: The value of flag is false
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
The thread 3 has locked the mutex mcr1
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -2;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-2
Thread 3 has taken the element from the queue with the number 3
The semaphore scr21 is closed by the thread 4
The thread 5 has locked the mutex mut2
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 24;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
29
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
The thread 3 has unlocked the mutex mcr1
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
The thread 5 is waiting for a signal sig 22
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 29
Thread 1: The sum of c and d is 18
Thread 1: The sum of e and f is 28
Thread 1: The sum of g and h is 26
Thread 1: The value of flag is false
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 20;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 5
The thread 4 has locked the mutex mut1
The signal sig2 was repeatedly send from the thread 3
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -3;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-3
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 29;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
34
```

```
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The thread 4 has unlocked the mutex mut1
The semaphore scr21 is opened by the thread 1
Function "__sync_nand_and_fetch" was called by the thread No 4. The
result value is 4294967295;
Thread 4: atomic_uint variable d after function "__sync_nand_and_fetch"
is 4294967295
Function "__sync_bool_compare_and_swap" was called by the thread No 4.
The result value is true;
The thread 1 has locked the mutex mut1
The thread 3 has locked the mutex mcr1
Thread 4: atomic_bool variable fl after function
"__sync_bool_compare_and_swap" is true
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 34
Thread 3 has taken the element from the queue with the number 2
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 27
Thread 1: The sum of g and h is 11
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -4;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-4
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 34;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
39
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 39
Thread 1: The sum of c and d is 1
```

```
Thread 1: The sum of e and f is 26
Thread 1: The sum of g and h is 26
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 20;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 5
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -5;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-5
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 39;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
44
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The thread 3 has unlocked the mutex mcr1
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 44
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 25
Thread 1: The sum of g and h is 11
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -6;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-6
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 44;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
49
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
```

```
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 49
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 24
Thread 1: The sum of g and h is 26
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 20;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 5
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -7;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-7
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 49;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
54
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 54
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 23
Thread 1: The sum of g and h is 11
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
The signal sig2 was repeatedly send from the thread 3
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -8;
```

```
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-8
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 54;
The thread 6 has locked the mutex mcr1
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
59
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 59
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 22
Thread 1: The sum of g and h is 26
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 20;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 5
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -9;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-9
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 59;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
64
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
The thread 4 has locked the mutex mut2
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 64
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 21
Thread 1: The sum of g and h is 11
Thread 1: The value of flag is true
```

```
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -10;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-10
Thread 6 has added the element to the queue with the number 2
The thread 6 has unlocked the mutex mcr1
The thread 4 sends the signal sig22 to the thread 5
The signal sig1 was send from the thread 6
The thread 4 has unlocked the mutex mut2
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 64;
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 3
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
69
The thread 4 is waiting for an open semaphore scr21
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
The semaphore scr21 is closed by the thread 4
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The thread 6 has unlocked the mutex mcr1
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 69
The thread 2 has locked the mutex mcr1
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 20
Thread 1: The sum of g and h is 26
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 20;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 5
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -11;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-11
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 69;
```

```
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
74
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 74
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 19
Thread 1: The sum of g and h is 11
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -12;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-12
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 74;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
79
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 5 has unlocked the mutex mut2
Thread 5: The sum of a and b is 79
Thread 5: The sum of c and d is 1
Thread 5: The sum of e and f is 18
Thread 5: The sum of g and h is 26
Thread 5: The value of flag is true
Thread 2 has added the element to the queue with the number 4
The thread 4 has locked the mutex mut1
The thread 2 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The signal sig1 was send from the thread 2
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 5
The thread 6 has unlocked the mutex mcr1
```

```
The signal sig1 was send from the thread 6
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 6
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 7
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 8
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 9
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The thread 6 has locked the mutex mcr1
The thread 6 is waiting for a signal sig2
The thread 4 has unlocked the mutex mut1
Function "__sync_nand_and_fetch" was called by the thread No 4. The
result value is 4294967218;
Thread 4: atomic_uint variable d after function "__sync_nand_and_fetch"
is 4294967218
Function "__sync_bool_compare_and_swap" was called by the thread No 4.
The result value is true;
Thread 4: atomic_bool variable fl after function
"__sync_bool_compare_and_swap" is true
The thread 4 has locked the mutex mut2
The thread 4 sends the signal sig22 to the thread 5
The thread 4 has unlocked the mutex mut2
The thread 4 is waiting for an open semaphore scr21
The semaphore scr21 is closed by the thread 4
The thread 4 has locked the mutex mut1
The thread 4 has unlocked the mutex mut1
Function "__sync_nand_and_fetch" was called by the thread No 4. The
result value is 4294967295;
Thread 4: atomic_uint variable d after function "__sync_nand_and_fetch"
is 4294967295
Function "__sync_bool_compare_and_swap" was called by the thread No 4.
The result value is true;
Thread 4: atomic_bool variable fl after function
"__sync_bool_compare_and_swap" is true
The thread 4 has locked the mutex mut2
The thread 4 sends the signal sig22 to the thread 5
The thread 4 has unlocked the mutex mut2
The thread 4 is waiting for an open semaphore scr21
The semaphore scr21 is closed by the thread 4
The thread 4 has locked the mutex mut1
The thread 4 is waiting for a signal sig 21
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
The thread 3 has locked the mutex mcr1
The CR1 was FULL ONCE
Thread 1: The sum of a and b is 79
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 18
```

```
Thread 1: The sum of g and h is 26
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 20;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 5
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -13;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-13
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 79;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
84
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 2 has locked the mutex mut1
The thread 2 has unlocked the mutex mut1
Thread 2: The sum of a and b is 84
Thread 2: The sum of c and d is 1
Thread 2: The sum of e and f is 17
Thread 2: The sum of g and h is 11
The thread 1 has locked the mutex mut1
Thread 2: The value of flag is true
The CR1 was FULL AT LEAST TWICE
Thread 3 has taken the element from the queue with the number 9
The thread 3 has unlocked the mutex mcr1
The signal sig2 was repeatedly send from the thread 3
The thread 3 has locked the mutex mcr1
Thread 3 has taken the element from the queue with the number 8
The thread 3 has unlocked the mutex mcr1
The signal sig2 was repeatedly send from the thread 3
The thread 3 has locked the mutex mcr1
Thread 3 has taken the element from the queue with the number 7
The thread 3 has unlocked the mutex mcr1
The signal sig2 was repeatedly send from the thread 3
The thread 3 has locked the mutex mcr1
Thread 3 has taken the element from the queue with the number 6
The thread 3 has unlocked the mutex mcr1
The signal sig2 was repeatedly send from the thread 3
The thread 3 has locked the mutex mcr1
Thread 3 has taken the element from the queue with the number 5
The thread 3 has unlocked the mutex mcr1
The signal sig2 was repeatedly send from the thread 3
The thread 3 has locked the mutex mcr1
Thread 3 has taken the element from the queue with the number 4
The thread 3 has unlocked the mutex mcr1
```

```
The signal sig2 was repeatedly send from the thread 3
The thread 3 has locked the mutex mcr1
Thread 3 has taken the element from the queue with the number 3
The thread 3 has unlocked the mutex mcr1
The signal sig2 was repeatedly send from the thread 3
The thread 3 has locked the mutex mcr1
Thread 3 has taken the element from the queue with the number 2
The thread 3 has unlocked the mutex mcr1
The signal sig2 was repeatedly send from the thread 3
The thread 3 has locked the mutex mcr1
Thread 3 has taken the element from the queue with the number 1
The thread 3 has unlocked the mutex mcr1
The signal sig2 was repeatedly send from the thread 3
The thread 3 has locked the mutex mcr1
Thread 3 has taken the element from the queue with the number 0
The CR1 was EMPTY ONCE
The thread 3 has unlocked the mutex mcr1
The signal sig2 was repeatedly send from the thread 3
The thread 3 has locked the mutex mcr1
The CR1 was EMPTY AT LEAST TWICE
The thread 3 has unlocked the mutex mcr1
Thread 6 has added the element to the queue with the number 0
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The thread 2 has locked the mutex mcr1
Thread 2 has added the element to the queue with the number 1
The thread 2 has unlocked the mutex mcr1
The signal sig1 was send from the thread 2
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 2
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 3
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 4
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 5
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 6
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The thread 6 has locked the mutex mcr1
Thread 6 has added the element to the queue with the number 7
The thread 6 has unlocked the mutex mcr1
The signal sig1 was send from the thread 6
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 84
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 17
Thread 1: The sum of g and h is 11
```

```
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
The thread 5 has locked the mutex mcr1
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -14;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-14
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 84;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
89
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 89
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 16
Thread 1: The sum of g and h is 26
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 20;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 5
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -15;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-15
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 89;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
94
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
```

```
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 94
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 15
Thread 1: The sum of g and h is 11
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -16;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-16
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 94;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
99
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the m the thread 6
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 84
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 17
Thread 1: The sum of g and h is 11
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
The thread 5 has locked the mutex mcr1
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -14;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-14
```

```
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 84;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
89
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 89
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 16
Thread 1: The sum of g and h is 26
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 20;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 5
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -15;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-15
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 89;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
94
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mutex mut1
The signal sig21 was repeatedly send from the thread 1
The thread 1 has unlocked the mutex mut1
Thread 1: The sum of a and b is 94
Thread 1: The sum of c and d is 1
Thread 1: The sum of e and f is 15
Thread 1: The sum of g and h is 11
Thread 1: The value of flag is true
Function "__sync_fetch_and_xor" was called by the thread No 1. The result
value is 5;
Thread 1: atomic_ulong variable g  after function "__sync_fetch_and_xor"
is 20
```

```
Function "__sync_fetch_and_and" was called by the thread No 1. The result
value is 0;
Thread 1: atomic_int variable b after function  is "__sync_fetch_and_and"
0
Function "__sync_sub_and_fetch" was called by the thread No 1. The result
value is -16;
Thread 1: atomic_long variable f after function "__sync_sub_and_fetch" is
-16
Function "__sync_fetch_and_add" was called by the thread No 1. The return
value is 94;
Thread 1: atomic_int variable a  after function "__sync_fetch_and_add" is
99
Function "__sync_or_and_fetch" was called by the thread No 1. The return
value is 30:
Thread 1: atomic_long variable e after function "__sync_or_and_fetch" is
30
Function "__sync_val_compare_and_swap" was called by the thread No 1. The
result value is 2;
Thread 1: atomic_uint variable c after function
"__sync_val_compare_and_swap" is 2
The semaphore scr21 is opened by the thread 1
The thread 1 has locked the mThread 5 has added the element to the queue
with the number 8
Thread 1 is cancelled
Thread 2 is cancelled
Thread 4 is cancelled
Thread 5 is cancelled
Thread 6 is cancelled
Consumer thread 3 of the data, stored in the CR1 is stopped !!!
All threads stopped !!!
The sum is 67
```