



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики
Кафедра системного програмування і спеціалізованих
комп'ютерних систем

Лабораторна робота №4

з дисципліни «Паралельне та розподілене обчислення»

Тема: «Засоби взаємодії паралельних потоків операційної
системи Linux»

Виконав:

студент групи

КВ84

Воєводін І.П.

Перевірив:

Київ – 2021

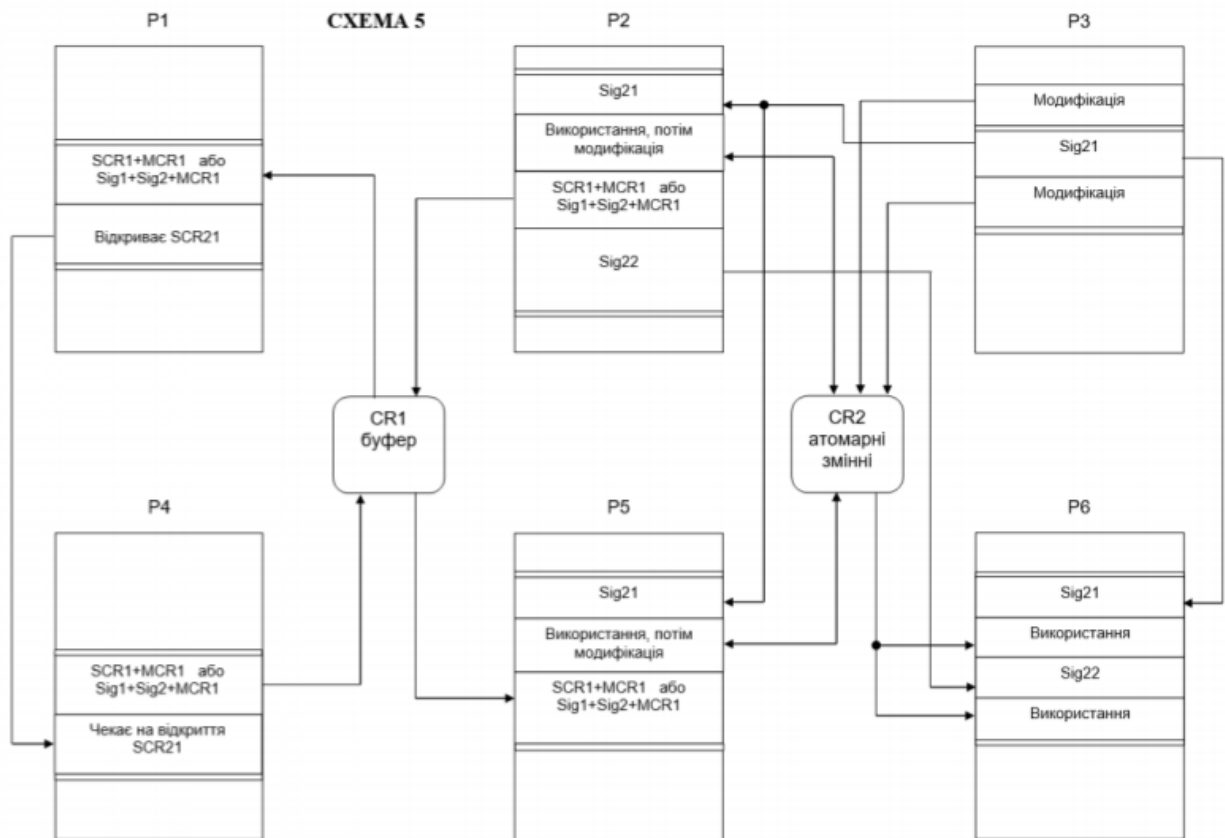
Постановка завдання та вимоги до виконання програми

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.

Варіант 29

Таблиця 1

№ варіанту	№ 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 використовується у якості спільного ресурсу 1 (CR1)	Засоби взаємного виключення при доступі до спільного ресурсу 1 SCR1 + MCR1 або Sig1+Sig2+MCR1	Атомарні дані (взяті по 2 змінних кожного 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
29	5	Стек (Вектор)	Дві сигнальні (умовні) змінні Sig1 та Sig2 і неблокуючий м'ютекс MCR1	1, 8, 4, 6, 10, 12, 13, 14	Неблокуючий	—	—	Багато-значний	Одиничний



Vector.h

```

#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define length 3

```

```

extern FILE *file;
extern int restart;
extern int arr[length];
extern short last;

```

```

short is_full();
short is_empty();
void push_back();
void pop_back();
int getelem(short p);

```

Vector.c

```

#include "vector.h"

```

```

FILE *file;

```

```
int restart = -1;
int arr[length];
short last = 0;
```

```
short is_full()
{
    if (last == length)
    {
        restart++;
        return 1;
    }
    return 0;
}
```

```
short is_empty()
{
    if (last == 0)
    {
        restart++;
        return 1;
    }
    return 0;
}
```

```
void push_back()
{
    arr[last] = ++last;
    printf("push %d\n", arr[last - 1]);
}
```

```
void pop_back()
{
    last--;
    printf("pop %d\n", arr[last]);
}
```

```
int getelem(short p) { return (p) ? arr[last - p] : arr[last]; }
```

thread.h

```
#include "vector.h"
#define repeat 3
```

```
extern pthread_t p1, p2, p3, p4, p5, p6;
```

```
extern pthread_cond_t not_empty, not_full, Sig21, Sig22;
```

```
extern pthread_mutex_t MCR1, M_Sig21, M_Sig22;
extern sem_t SCR21;
```

```
extern int int_1, int_2;
extern unsigned unsigned_1, unsigned_2;
extern long long_1, long_2;
extern unsigned long unsigned_long_1, unsigned_long_2;
```

```
void *p1_func(void *th);
void *p2_func(void *th);
void *p3_func(void *th);
void *p4_func(void *th);
void *p5_func(void *th);
void *p6_func(void *th);
```

thread.c

```
#include "thread.h"
```

```
pthread_t p1, p2, p3, p4, p5, p6;
```

```
int int_1, int_2;
unsigned unsigned_1, unsigned_2;
long long_1, long_2;
unsigned long unsigned_long_1, unsigned_long_2;
```

```
void *p1_func(void *th);
void *p2_func(void *th);
void *p3_func(void *th);
void *p4_func(void *th);
void *p5_func(void *th);
void *p6_func(void *th);
```

```
pthread_cond_t not_empty, not_full;
pthread_cond_t Sig21 = PTHREAD_COND_INITIALIZER;
pthread_cond_t Sig22 = PTHREAD_COND_INITIALIZER;
pthread_mutex_t MCR1, M_Sig21, M_Sig22;
sem_t SCR21;
```

```
void *p1_func(void *th)
{
    fprintf(file, "Thread #1 START\n");
    while (1)
    {
        if (pthread_mutex_trylock(&MCR1))
        {
```

```

        fprintf(file, "Thread #1 WAITING FOR MCR1\n");
        while (pthread_mutex_trylock(&MCR1))
        {
        }
    }
    while (is_empty())
    {
        pthread_cond_wait(&not_empty, &MCR1);
    }
    pop_back();
    fprintf(file, "Thread #1 POP ELEM %d\n", getelem(0));
    pthread_mutex_unlock(&MCR1);
    pthread_cond_broadcast(&not_full);
    fprintf(file, "Thread #1 SCR21 POST\n");
    sem_post(&SCR21);
}
fprintf(file, "Thread #1 END\n");
}

void *p2_func(void *th)
{
    fprintf(file, "Thread #2 START\n");
    while (1)
    {
        pthread_mutex_lock(&M_Sig21);
        fprintf(file, "Thread #2 Sig21 WAIT\n");
        pthread_cond_wait(&Sig21, &M_Sig21);
        fprintf(file, "Thread #2 Sig21 GOTTEN\n");
        pthread_mutex_unlock(&M_Sig21);
        fprintf(file, "Thread #2 USED & MODIFIED\n");
        __sync_and_and_fetch(&unsigned_1, 10);
        __sync_fetch_and_and(&unsigned_2, 4);
        __sync_val_compare_and_swap(&int_2, 5, 10);

        if (pthread_mutex_trylock(&MCR1))
        {
            fprintf(file, "Thread #2 WAITING FOR MCR1\n");
            while (pthread_mutex_trylock(&MCR1))
            {
            }
        }
        while (is_full())
        {
            pthread_cond_wait(&not_full, &MCR1);
        }
    }
}

```

```

        push_back();
        fprintf(file, "Thread #2 PUSH ELEM %d\n", getelem(1));
        pthread_mutex_unlock(&MCR1);
        pthread_cond_broadcast(&not_empty);
        fprintf(file, "Thread #2 Sig22 SIGNAL\n");
        pthread_cond_signal(&Sig22);
    }
    fprintf(file, "Thread #2 END\n");
}

void *p3_func(void *th)
{
    pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS, NULL);
    fprintf(file, "Thread #3 START\n");
    fprintf(file, "Thread #3 MODIFIED\n");
    __sync_bool_compare_and_swap(&int_1, 4, 2);
    while (1)
    {
        if (restart >= repeat)
        {
            break;
        }
        fprintf(file, "Thread #3 Sig21 BROADCAST\n");
        pthread_cond_broadcast(&Sig21);
        fprintf(file, "Thread #3 MODIFIED\n");
        __sync_val_compare_and_swap(&int_2, 2, 6);
    }
    fprintf(file, "Thread #3 END\n");
    pthread_cancel(p1);
    fprintf(file, "Thread #1 END\n");
    pthread_cancel(p2);
    fprintf(file, "Thread #2 END\n");
    pthread_cancel(p4);
    fprintf(file, "Thread #4 END\n");
    pthread_cancel(p5);
    fprintf(file, "Thread #5 END\n");
    pthread_cancel(p6);
    fprintf(file, "Thread #6 END\n");
}

void *p4_func(void *th)
{
    fprintf(file, "Thread #4 START\n");
    while (1)
    {

```

```

    if (pthread_mutex_trylock(&MCR1))
    {
        fprintf(file, "Thread #4 WAITING FOR MCR1\n");
        while (pthread_mutex_trylock(&MCR1))
        {
        }
    }
    while (is_full())
    {
        pthread_cond_wait(&not_full, &MCR1);
    }
    push_back();
    fprintf(file, "Thread #4 PUSH ELEM %d\n", getelem(1));
    pthread_mutex_unlock(&MCR1);
    pthread_cond_broadcast(&not_empty);
    fprintf(file, "Thread #4 SCR21 WAIT\n");
    sem_trywait(&SCR21);
    fprintf(file, "Thread #4 SCR21 GOTTEN\n");
}
fprintf(file, "Thread #4 END\n");

}

void *p5_func(void *th)
{
    fprintf(file, "Thread #5 START\n");
    while (1)
    {
        pthread_mutex_lock(&M_Sig21);
        fprintf(file, "Thread #5 Sig21 WAIT\n");
        pthread_cond_wait(&Sig21, &M_Sig21);
        fprintf(file, "Thread #5 Sig21 GOTTEN\n");
        pthread_mutex_unlock(&M_Sig21);
        fprintf(file, "Thread #5 USED & MODIFIED\n");
        __sync_fetch_and_add(&long_1, 6);
        __sync_sub_and_fetch(&long_2, 8);
        __sync_val_compare_and_swap(&int_2, 2, 6);
        if (pthread_mutex_trylock(&MCR1))
        {
            fprintf(file, "Thread #5 WAITING FOR MCR1\n");
            while (pthread_mutex_trylock(&MCR1))
            {
            }
        }
    }
    while (is_empty())

```



```

    {
        pthread_cond_wait(&not_empty, &MCR1);
    }
    pop_back();
    fprintf(file, "Thread #5 POP ELEM %d\n", getelem(0));
    pthread_mutex_unlock(&MCR1);
    pthread_cond_broadcast(&not_full);
}
fprintf(file, "Thread #5 END\n");
}

```

```

void *p6_func(void *th)
{
    fprintf(file, "Thread #6 START\n");
    while (1)
    {
        pthread_mutex_lock(&M_Sig21);
        fprintf(file, "Thread #6 Sig21 WAIT\n");
        pthread_cond_wait(&Sig21, &M_Sig21);
        fprintf(file, "Thread #6 Sig21 GOTTEN\n");
        pthread_mutex_unlock(&M_Sig21);
        fprintf(file, "Thread #6 USED\n");
        __sync_nand_and_fetch(&unsigned_long_1, 10);
        pthread_mutex_lock(&M_Sig22);
        fprintf(file, "Thread #6 Sig22 WAIT\n");
        pthread_cond_wait(&Sig22, &M_Sig22);
        fprintf(file, "Thread #6 Sig22 GOTTEN\n");
        pthread_mutex_unlock(&M_Sig22);
        fprintf(file, "Thread #6 USED\n");
        __sync_fetch_and_nand(&unsigned_long_2, 10);
    }
    fprintf(file, "Thread #6 END\n");
}

```

main.c

```
#include "thread.h"
```

```

int main()
{
    file = fopen("pro.log", "wt");
    sem_init(&SCR21, 0, 0);
    pthread_cond_init(&not_empty, NULL);
    pthread_cond_init(&not_full, NULL);
    pthread_create(&p1, NULL, p1_func, NULL);
    pthread_create(&p2, NULL, p2_func, NULL);
}

```

```
pthread_create(&p3, NULL, p3_func, NULL);
pthread_create(&p4, NULL, p4_func, NULL);
pthread_create(&p5, NULL, p5_func, NULL);
pthread_create(&p6, NULL, p6_func, NULL);
pthread_join(p3, NULL);
fprintf(file, "DONE");
fclose(file);
return 0;
}
```

makefile

.PHONY: all clean

all: pro

%.c:%.o

pro: vector.o thread.o main.o

gcc -o pro -pthread main.o vector.o thread.o

main.o: main.c

gcc -c main.c

vector.o: vector.c vector.h

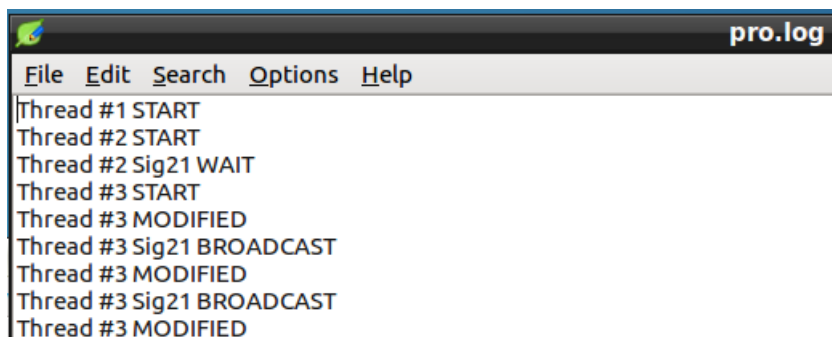
gcc -c vector.c

thread.o: thread.c thread.h

gcc -c thread.c

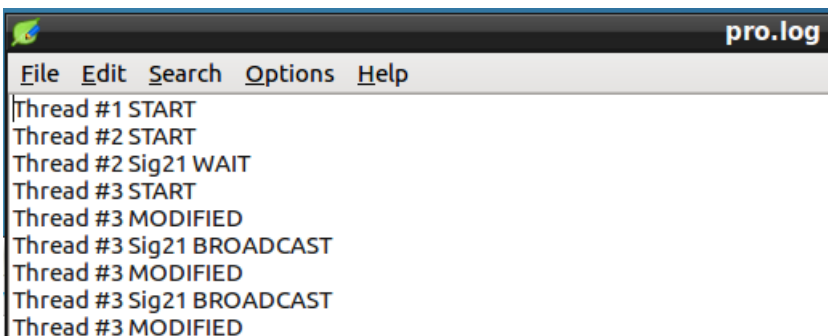
clean:

rm -r *.o pro



Thread #1 POP ELEM 3
Thread #1 SCR21 POST
Thread #1 POP ELEM 2
Thread #1 SCR21 POST
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #5 Sig21 GOTTEN
Thread #5 USED & MODIFIED
Thread #6 Sig21 GOTTEN
Thread #6 USED
Thread #6 Sig22 WAIT
Thread #2 Sig21 GOTTEN
Thread #2 USED & MODIFIED
Thread #2 PUSH ELEM 1
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #2 Sig22 SIGNAL
Thread #2 Sig21 WAIT
Thread #6 Sig22 GOTTEN
Thread #6 USED
Thread #6 Sig21 WAIT
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 3
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #3 MODIFIED
Thread #3 END
Thread #1 POP ELEM 3
Thread #1 SCR21 POST
Thread #1 POP ELEM 2
Thread #1 SCR21 POST
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 3
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #1 POP ELEM 3
Thread #1 SCR21 POST
Thread #1 POP ELEM 2

Thread #1 SCR21 POST
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 3
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #1 POP ELEM 3
Thread #1 SCR21 POST
Thread #1 POP ELEM 2
Thread #1 SCR21 POST
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 3
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #1 POP ELEM 3
Thread #1 SCR21 POST
Thread #1 POP ELEM 2
Thread #1 SCR21 POST
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 3
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #1 END
Thread #2 END
Thread #4 END
Thread #5 END
Thread #6 END
DONE



The screenshot shows a terminal window with a title bar that includes a small icon and the text 'pro.log'. Below the title bar is a menu bar with the following items: 'File', 'Edit', 'Search', 'Options', and 'Help'. The main area of the terminal displays a list of thread events, each on a new line. The events are: 'Thread #1 START', 'Thread #2 START', 'Thread #2 Sig21 WAIT', 'Thread #3 START', 'Thread #3 MODIFIED', 'Thread #3 Sig21 BROADCAST', 'Thread #3 MODIFIED', 'Thread #3 Sig21 BROADCAST', and 'Thread #3 MODIFIED'.

pro.log

File Edit Search Options Help

Thread #1 START
Thread #2 START
Thread #2 Sig21 WAIT
Thread #3 START
Thread #3 MODIFIED
Thread #3 Sig21 BROADCAST
Thread #3 MODIFIED
Thread #3 Sig21 BROADCAST
Thread #3 MODIFIED

Thread #5 Sig21 GOTTEN
Thread #5 USED & MODIFIED
Thread #5 WAITING FOR MCR1
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 WAITING FOR MCR1
Thread #5 POP ELEM 1
Thread #5 Sig21 WAIT
Thread #5 Sig21 GOTTEN
Thread #5 USED & MODIFIED
Thread #3 MODIFIED
Thread #3 END
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 3
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #1 POP ELEM 3
Thread #1 SCR21 POST
Thread #1 POP ELEM 2
Thread #1 SCR21 POST
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 3
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #1 POP ELEM 3
Thread #1 SCR21 POST
Thread #1 POP ELEM 2
Thread #1 SCR21 POST
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN

Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 3
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #1 POP ELEM 3
Thread #1 SCR21 POST
Thread #1 POP ELEM 2
Thread #1 SCR21 POST
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 3
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #5 POP ELEM 3
Thread #5 Sig21 WAIT
Thread #1 POP ELEM 2
Thread #1 SCR21 POST
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #4 PUSH ELEM 1
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 2
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #4 PUSH ELEM 3
Thread #4 SCR21 WAIT
Thread #4 SCR21 GOTTEN
Thread #1 POP ELEM 3
Thread #1 SCR21 POST
Thread #1 POP ELEM 2
Thread #1 SCR21 POST
Thread #1 POP ELEM 1
Thread #1 SCR21 POST
Thread #1 END
Thread #2 END
Thread #4 END
Thread #5 END
Thread #6 END
DONE