Лабораторная работа №4. Синхронизация

Цель

Целью выполнения этого компьютерного практикума является изучение примитивов синхронизации ОС на основе возможностей, которые дает библиотека PTHREAD, для решения классических задач синхронизации.

В результате его выполнения будут получены базовые знания библиотеки PTHREAD, а также проихойдет овладение подходами к написанию корректных программ, которые работают в конкурентной среде, с использованием системных механизмов синхронизации.

Использование библиотеки PTHREAD

Пример создания нитей с использованием PTHREAD:

```
#include <stdio.h>
#include <pthread.h>

main() {
    pthread_t f2_thread, f1_thread;
    void *f2(), *f1();
    int i1 = 1, i2 = 2;
    pthread_create(&f1_thread, NULL, f1, &i1);
    pthread_create(&f2_thread, NULL, f2, &i2);
    pthread_join(f1_thread, NULL);
    pthread_join(f2_thread, NULL);
}

void *f1(int *x) {
    int i;
    i = *x;
    sleep(1);
    printf("f1: %d", i);
    pthread_exit(0);
}

void *f2(int *x) {
    int i;
    i = *x;
```


* pdf generated on 2014-10-20 from markdown sources with SBCL 1.1.14 on Linux 3.11.0-12-generic with 3bmd, cl-pdf & cl-typesetting

```
    sleep(1);
    printf("f2: %d", i);
    pthread_exit(0);
}
```

Решение задачи "Производитель-потребитель" с помощью PTHREAD с использованием семафоров и условных переменных:

```
#include <sys/time.h>
#include <stdio.h>
#include <pthread.h>
#include <errno.h>

#define SIZE 10

pthread_mutex_t region_mutex =
    PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t space_available =
    PTHREAD_COND_INITIALIZER;
pthread_cond_t data_available =
    PTHREAD_COND_INITIALIZER;

int b[SIZE];
int size = 0;
int front, rear = 0;

main()
{
    pthread_t producer_thread;
    pthread_t consumer_thread;

    void *producer();
    void *consumer();

    pthread_create(&consumer_thread, NULL,
                   consumer, NULL);
    pthread_create(&producer_thread, NULL,
                   producer, NULL);
    pthread_join(consumer_thread,NULL);
}

void add_buffer(int i)
{
    b[rear] = i;
```


* pdf generated on 2014-10-20 from markdown sources with SBCL 1.1.14 on Linux 3.11.0-12-generic with 3bmd, cl-pdf & cl-typesetting

```
    rear = (rear+1) % SIZE;
    size++;
}

int get_buffer()
{
    int v;
    v = b[front];
    front = (front+1) % SIZE;
    size--;
    return v;
}

void *producer()
{
    int i = 0;
    while (1)
    {
        pthread_mutex_lock(&region_mutex);
        if (size == SIZE)
            pthread_cond_wait(&space_available,
                              &region_mutex);
        add_buffer(i);
        pthread_cond_signal(&data_available);
        pthread_mutex_unlock(&region_mutex);
        i++;
    }
    pthread_exit(NULL);
}

void *consumer()
{
    int i, v;
    for (i = 0; i < 100; i++)
    {
        pthread_mutex_lock(&region_mutex);
        if (size == 0)
            pthread_cond_wait(&data_available,
                              &region_mutex);
        v = get_buffer();
        pthread_cond_signal(&space_available);
        pthread_mutex_unlock(&region_mutex);
        printf("got %d  ",v);
    }
```


* pdf generated on 2014-10-20 from markdown sources with SBCL 1.1.14 on Linux 3.11.0-12-generic with 3bmd, cl-pdf & cl-typesetting

```
        pthread_exit(NULL);
    }
```

Задание

С помощью примитивов синхронизации из библиотеки PTHREAD решить на языке C одну из задач синхронизации в соответсвии с вариантом.

Оценка за лабораторную работу: 8 баллов

Дополнительное задание: решить ту же задачу с помощью одного из следующих примитивов синхронизации в соответствующем языке:

- Передача сообщений в Erlang
- STM в Clojure или Haskell
- Каналы в Go

Оценка за дополнительное задание: 8 баллов

Литература

- Little Book of Semaphores
- http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
- http://ccfit.nsu.ru/arom/data/PP_ICaG/03_pthreads_ru.pdf

* pdf generated on 2014-10-20 from markdown sources with SBCL 1.1.14 on Linux 3.11.0-12-generic with 3bmd, cl-pdf & cl-typesetting