МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем

Кафедра «Прикладная математика и фундаментальная информатика»

**Домашнее задание**

|  |  |  |
| --- | --- | --- |
| по дисциплине | Операционные системы |  |

|  |  |
| --- | --- |
| Студента | Иванютин Сергей Александрович |
|  | фамилия, имя, отчество полностью |
| Курс | 2 Группа ФИТ-221 |
| Направление | 02.03.02 Фундаментальная информатика |
|  | и информационные технологии |
|  | код, наименование |
| Руководитель | ассистент |
|  | должность, ученая степень, звание |
|  | Карабцов Р.Д. |
|  | фамилия, инициалы |
| Выполнил | 04.06.2024 |
|  | дата, подпись студента |
| баллы |  |
|  | дата, подпись руководителя |

Омск-2024

**Задание**

Разработать многопоточную программу, отображающие на экране взаимодействие трех нитей «читателей» из общей области данных и трех «писателей», записывающих в этот буфер данные. Буфер предназначен для хранения 12 символов. Первая нить-писатель выводит латинскими буквами название города "Novosibirsk", вторая нить-писатель выводит латинскими буквами название города "Semipalatink", а третья — название города "Ekaretinburg". Такой вывод эти три нити осуществляют в два приема, первый из которых записывает половину своего текста без завершающего этот промежуточный текст нуля. Между такими половинами вывода нити производят задержку на случайную величину миллисекунд, но не более 1 сек. После вывода своего текста в буфер каждая нить-писатель переходит в ожидание порядка 2-3 сек до следующей попытки записи в буфер. Нити-читатели - через случайный интервал порядка 300 мсек - читают данные из буфера, если это позволяют средства синхронизации доступа между нитями, и вывод прочитанный текст на экран, каждая в свой столбец. Каждый вывод нити-читателя осуществляется в новую строку своего столбца, поэтому суммарные действия вывода в таких нитях предусмотреть только для 20 - 24 строк. Синхронизацию осуществить с помощью семафоров.

1. Чем абстрактные семафоры отличаются от известных вам программных реализаций.

2. Укажите в каких ситуациях предпочтительней использование мьютексов вместо семафоров и в каких ситуациях целесообразно обратное предпочтение.

**Решение**



Рисунок 1. Вывод программы

1.Отличие абстрактных семафоров от программных реализаций:

Программные реализации: Обычно это конкретные механизмы синхронизации, такие как мьютексы, семафоры, условные переменные и прочее, предоставляемые операционной системой или библиотекой. Эти реализации обычно имеют явный API и связаны с конкретной платформой.

Абстрактные семафоры: Это более высокоуровневые абстракции, предоставляющие удобный интерфейс для управления доступом к ресурсам или выполнением потоков. Они не обязательно привязаны к конкретным механизмам операционной системы и предоставляют общие концепции для управления параллельным выполнением.

2.Выбор между мьютексами и семафорами:

Мьютексы:

Предпочтительны, когда нужно обеспечить эксклюзивный доступ к ресурсу: Если важно, чтобы только один поток одновременно мог использовать определенный ресурс или выполнить критическую секцию кода, мьютексы являются хорошим выбором.

Простота использования: Мьютексы обычно проще в использовании, чем семафоры, и легче поддерживают основные сценарии блокировки.

Обеспечение взаимного исключения: Мьютексы часто используются для создания критических секций, где только один поток может выполнять код в определенный момент времени.

Семафоры:

Управление ресурсами с ограниченной емкостью: Если нужно управлять доступом к ресурсам, которые могут использовать несколько потоков одновременно (например, пул потоков), семафоры с ограниченной емкостью могут быть более подходящими.

Синхронизация между несколькими потоками: Семафоры могут использоваться для синхронизации между потоками, не обязательно ограничивая количество одновременно активных потоков.

Сложные сценарии блокировки: В более сложных сценариях, где нужно управлять группами потоков, а не просто предоставлять или ограничивать доступ к ресурсам, семафоры могут быть более гибкими.

**Листинг программы**

#include <semaphore.h>

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

char buffer[12] = "Started...";

int n = 0;

sem\_t s, w;

void reader(void \*args)

{

int id, column, color;

id = (int)args;

column = id\*20;

color = 33+id;

for (int line = 0; line < 20; line++)

{

sem\_wait(&s);

n += 1;

if (n == 1)

{

sem\_wait(&w);

}

sem\_post(&s);

printf("\033[%d;%dH\033[%dm%s", line+1, column+1, color, buffer);

fflush(stdout);

sem\_wait(&s);

n -= 1;

if (n == 0)

{

sem\_post(&w);

}

sem\_post(&s);

usleep(300000);

}

}

void writer(void \*args)

{

int id;

id = (int)args;

while (1)

{

sem\_wait(&w);

fflush(stdout);

switch (id)

{

case 0:

sprintf(buffer, "Novos");

usleep(1000000);

sprintf(buffer+5, "ibirsk");

break;

case 1:

sprintf(buffer, "Semipalatink");

usleep(1000000);

sprintf(buffer+5, "alatink");

break;

case 2:

sprintf(buffer, "Ekaterinburg");

usleep(1000000);

sprintf(buffer+5, "rinburg");

break;

default:

sprintf(buffer, "nothing");

break;

}

usleep(1000000);

sem\_post(&w);

usleep(2500000);

}

}

int main()

{

pthread\_t rtid[3], wtid[3];

int rc;

sem\_init(&s, 0, 1);

sem\_init(&w, 0, 1);

printf("\033[2J\n");

rc = pthread\_create(&rtid[0], NULL, (void \*)reader, (void \*)0);

rc = pthread\_create(&rtid[1], NULL, (void \*)reader, (void \*)1);

rc = pthread\_create(&rtid[2], NULL, (void \*)reader, (void \*)2);

rc = pthread\_create(&wtid[0], NULL, (void \*)writer, (void \*)0);

rc = pthread\_create(&wtid[1], NULL, (void \*)writer, (void \*)1);

rc = pthread\_create(&wtid[2], NULL, (void \*)writer, (void \*)2);

pthread\_join(rtid[0], 0);

pthread\_join(rtid[1], 0);

pthread\_join(rtid[2], 0);

sem\_destroy(&s);

sem\_destroy(&w);

putchar('\n');

}