Федеральное агентство связи

Сибирский государственный университет телекоммуникаций и информатики

Лабораторная работа №11

Выполнил: студент группы ИП-211

Оганесян Альберт

Лацук Андрей

Проверил:

Профессор кафедры ПМиК

Малков Е. А.

Новосибирск 2024

**Цель:** получение навыков синхронизации с использованием мьютексов и семафоров

**Задание:** реализовать алгоритм “производитель-потребитель” для конечного буфера

**Ход работы:**

1. Конечный буфер реализуем массивом фиксированного размера *(BUFFER\_SIZE)*. Для чтения и записи используем индексы *in* и *out* соответственно.

| *#define BUFFER\_SIZE 5 int buffer[BUFFER\_SIZE]; int in = 0, out = 0;* |
| --- |

1. Определим мьютекс и семафоры. Мьютекс гарантирует, что только один поток (производитель или потребитель) имеет доступ к критической секции в любой момент времени. Семафоры отслеживают количество свободных и заполненных мест.

| pthread\_mutex\_t mutex;  sem\_t empty;  sem\_t full; |
| --- |

1. Реализуем функцию производителя. Производитель генерирует случайное число и помещает его в буфер. После он ждёт, пока в буфере освободится место (если буфер полон).

| void \*producer(void \*arg) {  int item;  while (!stop\_flag)  {  item = rand() % 100;   sem\_wait(&empty);   pthread\_mutex\_lock(&mutex);   if (stop\_flag)  {  pthread\_mutex\_unlock(&mutex);  sem\_post(&empty);  break;  }   buffer[in] = item;  printf("Производитель произвел: %d\n", item);  in = (in + 1) % BUFFER\_SIZE;  produced\_count++;   pthread\_mutex\_unlock(&mutex);  sem\_post(&full);    if (produced\_count >= MAX\_ITEMS)  {  stop\_flag = 1;  break;  }  sleep(rand() % 2);  }  return NULL; } |
| --- |

1. Реализуем функцию потребителя. Потребитель извлекает число из буфера. После он ждёт, пока в буфере появится хотя бы один элемент (если буфер пуст).

| void \*consumer(void \*arg) {  int item;  while (!stop\_flag || consumed\_count < produced\_count)  {  sem\_wait(&full);   pthread\_mutex\_lock(&mutex);   if (stop\_flag && consumed\_count >= produced\_count)  {  pthread\_mutex\_unlock(&mutex);  sem\_post(&full);  break;  }   item = buffer[out];  printf("Потребитель потребил: %d\n", item);  out = (out + 1) % BUFFER\_SIZE;  consumed\_count++;   pthread\_mutex\_unlock(&mutex);  sem\_post(&empty);    sleep(rand() % 3);  }  return NULL; } |
| --- |

1. Добавим флаг и сигнал завершения и лимит обработки. Если глобальная переменная set\_flag установлена в 1, то производитель и потребитель завершают свои циклы. Обработчик сигналов SIGINT (Ctrl+C) позволяет корректно завершить программу по запросу пользователя. После обработки MAX\_ITEMS элементов программа завершает работу.

| #define MAX\_ITEMS 8  int produced\_count = 0;  int consumed\_count = 0;  int stop\_flag = 0;  void handle\_signal(int sig) {  stop\_flag = 1;  printf("\nПолучен сигнал завершения. Завершаем программу...\n"); } |
| --- |

1. Соберем все в функции main

| int main() {  pthread\_t prod, cons;   // Устанавливаем обработчик сигнала завершения  signal(SIGINT, handle\_signal);   // Инициализация мьютекса и семафоров  pthread\_mutex\_init(&mutex, NULL);  sem\_init(&empty, 0, BUFFER\_SIZE);  sem\_init(&full, 0, 0);   // Создание потоков производителя и потребителя  pthread\_create(&prod, NULL, producer, NULL);  pthread\_create(&cons, NULL, consumer, NULL);   // Ожидание завершения потоков  pthread\_join(prod, NULL);  pthread\_join(cons, NULL);   // Очистка ресурсов  pthread\_mutex\_destroy(&mutex);  sem\_destroy(&empty);  sem\_destroy(&full);   printf("Программа завершена. Произведено: %d, Потреблено: %d\n", produced\_count, consumed\_count);  return 0; } |
| --- |

1. Пример вывода программы с автоматическим завершением

| Производитель произвел: 83 Потребитель потребил: 83 Производитель произвел: 15 Потребитель потребил: 15 Производитель произвел: 86 Потребитель потребил: 86 Производитель произвел: 21 Потребитель потребил: 21 Производитель произвел: 90 Производитель произвел: 63 Потребитель потребил: 90 Потребитель потребил: 63 Производитель произвел: 26 Производитель произвел: 11 Потребитель потребил: 26 Потребитель потребил: 11 Программа завершена. Произведено: 8, Потреблено: 8 |
| --- |

1. Пример вывода программы с ручным завершением

| Производитель произвел: 83 Потребитель потребил: 83  Производитель произвел: 15 Потребитель потребил: 15  ^C Получен сигнал завершения. Завершаем программу... Программа завершена. Произведено: 2, Потреблено: 2 |
| --- |

**Вывод:** Алгоритм "производитель-потребитель" успешно реализован с использованием синхронизации потоков средствами мьютексов и семафоров. Программа обеспечивает корректный обмен данными между производителем и потребителем без гонок данных и взаимоблокировок.