Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Операционные системы и системное программирование

Лабораторная работа №5
«Потоки исполнения, взаимодействие и синхронизация»

Выполнил:
Студент группы 350501
Рутковский В.К.

Проверил:
Поденок Л.П.

Минск 2025

1 ПОСТАНОВКА ЗАДАЧ

В данной лабораторной работе две задачи. Обе «производители-потребители» для потоков.

Первая задача пятой лабораторной работы аналогична четвертой лабораторной работе, но только с потоками, POSIX-семафорами и мьютексом в рамках одного процесса. Дополнительно обрабатывается еще две клавиши – увеличение и уменьшение размера очереди. Следует предусмотреть обработку запроса на уменьшение очереди таким образом, чтобы при появлении пустого места уменьшался размер очереди, а не очередной производитель размещал там свое сообщение.

Вторая задача пятой лабораторной работы аналогична предыдущей задаче, но с использованием условных переменных.

2 ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

2.1 Описание работы программы

Программы реализует классическую модель взаимодействия производителей и потребителей с использованием общей очереди сообщений фиксированного размера. Целью является демонстрация синхронизированного доступа нескольких потоков к общему ресурсу с учётом конкурентных условий.

В рамках программы производители создают сообщения и помещают их в очередь, а потребители извлекают эти сообщения и проверяют их целостность. Очередь имеет фиксированный размер — 10 элементов, что ограничивает максимальное количество сообщений, которые могут одновременно находиться в буфере. Все действия пользователя выполняются через интерактивное меню в консоли.

При вводе команды (`'p'`) создаётся производитель. Он запускается в отдельном потоке, генерирует сообщения со случайными параметрами (тип, размер, хэш) и помещает их в очередь. Если очередь заполнена, производитель приостанавливает работу до тех пор, пока не освободится место. Завершение производителя осуществляется командой (`'P'`).

Команда `c` запускает потребителя. Он также работает в отдельном потоке, извлекает сообщения из очереди и проверяет их хэш-сумму, подтверждая целостность данных. Если очередь пуста, потребитель ожидает появления новых сообщений. Завершение потребителя осуществляется командой (`'C'`).

Команда (`'s'`) позволяет просмотреть текущее состояние очереди. Отображаются: общий размер очереди, количество занятых и свободных ячеек, число добавленных и удалённых сообщений, а также количество активных производителей и потребителей. Команда (`'q'`) завершает работу программы.

Каждое сообщение, создаваемое производителем, содержит три поля: тип размер и хэш. Потребитель сравнивает вычисленную хэш-сумму с переданной, чтобы убедиться в корректности полученных данных.

2.2 Описание основных функций

В программах для обработки сообщений используются несколько ключевых функций, каждая из которых выполняет важную роль в взаимодействии между производителями, потребителями и очередью сообщений.

Краткое описание основных функций для первой части задания, обеспечивающих корректную работу системы:

Функция `main` (от `lab05.1`)

Основная функция программы, создает очередь с начальной емкостью 10, инициализирует массивы для хранения идентификаторов потоков производителей и потребителей, а также настраивает терминал в неблокирующий режим без эха ввода. Выводит меню для управления программой (создание/удаление производителей и потребителей, изменение размера очереди, отображение статуса, выход). В бесконечном цикле обрабатывает ввод пользователя: создает потоки для производителей ('`p`') и потребителей ('`c`'), отменяет их ('`P`', '`C`'), показывает статистику очереди ('`s`'), увеличивает ('`+`') или уменьшает ('`-`') размер очереди, либо завершает программу ('`q`'). При выходе отменяет все потоки, освобождает ресурсы и восстанавливает настройки терминала.

Функция `compute_hash` (от `lab05.1`)

Функция вычисляет хэш для сообщения типа `Message`. Суммирует значения поля `type`, поля `size` и всех байтов массива `data` (длина определяется полем `size`). Возвращает 16-битное значение хэша, которое используется для проверки целостности сообщения.

Функция `init_queue` (от `lab05.1`)

Инициализирует очередь (`Queue`) заданного размера `size`. Выделяет память для массива сообщений, устанавливает начальные значения: емкость (`capacity`), индексы головы и хвоста (`head`, `tail`), счетчики добавленных и удаленных сообщений (`added_count`, `removed_count`) и свободное место (`free_space`). Инициализирует мьютекс для синхронизации и два семафора: `empty` (для свободных слотов) и `full` (для занятых слотов).

Функция `resize_queue` (от `lab05.1`)

Изменяет размер очереди на `new_size`. Захватывает мьютекс, проверяет, что новый размер не меньше числа занятых слотов, и перераспределяет память для массива сообщений. Если размер уменьшается, и голова очереди находится за хвостом, перемещает сообщения для сохранения их порядка. Обновляет индексы, емкость, свободное место, пересоздает семафоры с учетом нового размера и выводит сообщение об успешном изменении размера очереди. Если перераспределение памяти не удалось, выводит ошибку.

Функция `producer` (от `lab05.1`)

Функция реализует производителя в модели производитель-потребитель. Принимает указатель на очередь (`Queue`) и бесконечно создает сообщения. Генерирует случайные значения для полей `type`, `size` и массива `data` с использованием `seed`, основанного на времени и идентификаторе потока. Вычисляет хэш сообщения, ожидает свободное место через семафор `empty`, захватывает мьютекс, добавляет сообщение в хвост очереди, обновляет индекс хвоста, счетчик добавленных сообщений и свободное место. Снимает мьютекс, сигнализирует семафору `full` и выводит информацию о созданном сообщении (номер, тип, размер, хэш). Делает паузу в 2 секунды перед созданием следующего

Функция `consumer` (от `lab05.1`)

Функция реализует потребителя в модели производитель-потребитель. Принимает указатель на очередь (`Queue`) и бесконечно ожидает сообщения, используя семафор `full` для проверки наличия данных. Захватывает мьютекс для безопасного доступа к очереди, извлекает сообщение с головы очереди, обновляет индекс головы, увеличивает счетчик удаленных сообщений и свободных слотов. После снятия мьютекса сигнализирует семафору `empty` о наличии свободного места. Вычисляет хэш сообщения, выводит информацию о нем (номер, тип, размер, хэш, результат проверки) и делает паузу в 2 секунды перед обработкой следующего сообщения.

Далее представлено краткое описание основных функций для второй части задания:

Функция `main` (от `lab05.2`)

Основная функция программы, создает очередь с начальной емкостью 10, инициализирует массивы для хранения идентификаторов потоков производителей и потребителей, настраивает терминал в неблокирующий

режим без эха ввода. Выводит меню для управления: создание/удаление производителей ('p'/'P') и потребителей ('c'/'C'), отображение статуса очереди ('s'), изменение размера ('+'/'-'), выход ('q'). В цикле обрабатывает ввод, создает или отменяет потоки, вызывает изменение размера очереди или выводит статистику (емкость, занятое/свободное место, счетчики сообщений, число потоков). При выходе отменяет все потоки, освобождает ресурсы (мьютекс, условные переменные, память) и восстанавливает настройки терминала.

Функция `compute_hash` (от `lab05.2`)

Вычисляет хэш сообщения типа Message. Суммирует значения полей `type`, `size` и всех байтов массива `data` (длина определяется полем `size`). Возвращает 16-битное значение хэша для проверки целостности сообщения.

Функция `cleanup_producer` (от `lab05.2`)

Функция-обработчик очистки для потока производителя. Принимает указатель на очередь (`Queue`), снимает блокировку с мьютекса очереди (`mutex`), чтобы избежать взаимоблокировок в случае отмены потока, обеспечивая корректное завершение операций с очередью.

Функция `producer` (от `lab05.2`)

Реализует производителя в модели производитель-потребитель. Принимает указатель на очередь (`Queue`) и бесконечно создает сообщения. Генерирует случайные значения для полей `type`, `size` и массива `data` сообщения с использованием `seed`, основанного на времени и идентификаторе потока. Вычисляет хэш сообщения, захватывает мьютекс очереди, ожидает свободное место с помощью условной переменной `not_full`, добавляет сообщение в хвост очереди, обновляет индексы и счетчики, сигнализирует о наличии данных через `not_empty`, выводит информацию о сообщении (номер, тип, размер, хэш) и делает паузу в 3 секунды.

Функция `cleanup_consumer` (от `lab05.2`)

Функция-обработчик очистки для потока потребителя. Принимает указатель на очередь (`Queue`), снимает блокировку с мьютекса очереди (`mutex`) и сигнализирует условной переменной `not_full` о наличии свободного места, предотвращая взаимоблокировки при отмене потока.

Функция `consumer` (от `lab05.2`)

Реализует потребителя в модели производитель-потребитель. Принимает указатель на очередь (`Queue`) и бесконечно обрабатывает сообщения. Захватывает мьютекс, ожидает данные с помощью условной переменной `not_empty`, извлекает сообщение с головы очереди, обновляет индексы и счетчики, сигнализирует о свободном месте через `not_full`, вычисляет хэш сообщения, выводит информацию о нем (номер, тип, размер, хэш, результат проверки) и делает паузу в 3 секунды.

Функция `resize_queue` (от `lab05.2`)

Изменяет размер очереди на `new_size`. Захватывает мьютекс, проверяет, что новый размер не меньше числа занятых слотов, перераспределяет память для массива сообщений. При уменьшении размера перемещает сообщения, если голова находится за хвостом, обновляет индексы, емкость и свободное место. Сигнализирует через `not_full` и `not_empty` для уведомления потоков, выводит сообщение об успешном изменении размера или ошибку при неудачном перераспределении памяти.

Функция `init_queue` (от `lab05.2`)

Инициализирует очередь (`Queue`) заданного размера `size`. Выделяет память для массива сообщений, устанавливает начальные значения: емкость (`capacity`), индексы головы и хвоста (`head`, `tail`), счетчики добавленных и удаленных сообщений (`added_count`, `removed_count`) и свободное место (`free_space`). Инициализирует мьютекс и условные переменные `not_full` и `not_empty` для синхронизации.

2.3 Описание запуска и сборки

Для сборки проекта используется утилита `cmake` и файл `CMakeLists.txt`, которая автоматически создает `makefile`, с помощью которого производится компиляция исходного кода и создание исполняемых файлов из `child.c` и `parent.c`. Запускать `cmake` следует в директории `build` для структуризации проекта (если находитесь в папке `build`: `$cmake ../CmakeLists.txt`; далее: `$make`).

Для удаления скомпилированного файла и очистки директории `build` предусмотрена команда `make clean`. Это позволяет поддерживать рабочую среду в чистоте и гарантировать корректность последующих сборок.

3 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

```
vadim@ubuntu:~/Projects/Lab05.1/build$ ./lab05.1
Commands:
 p - create producer
 c - create consumer
 + - increase queue
 - - decrease queue
 s - show queue status
 q - quit
Produced message 1: type=1, size=148, hash=19711
Produced message 2: type=1, size=90, hash=12408
c
Consumed message 1: type=1, size=148, hash=19711 — OK
Produced message 3: type=1, size=187, hash=24474
Consumed message 2: type=1, size=90, hash=12408 — OK
Produced message 4: type=1, size=24, hash=3343
+
Queue resized to 11
Consumed message 3: type=1, size=187, hash=24474 — OK
Produced message 5: type=1, size=64, hash=7693
-
Shrink to 10 requested
vadim@ubuntu:~/Projects/Lab05.1/build$
```

Рисунок 3.1 – Результаты работы программы первой части задания

```
vadim@ubuntu:~/Projects/Lab05.2/build$ ./lab05.2
Commands:
 p - create producer
 c - create consumer
 + - increase queue
 - - decrease queue
 s - show queue status
 q - quit
Produced message 1: type=1, size=148, hash=19711
Produced message 2: type=1, size=21, hash=12408
Consumed message 1: type=1, size=148, hash=19711 — OK
Produced message 3: type=1, size=192, hash=23324
Consumed message 2: type=1, size=21, hash=12408 — OK
Produced message 4: type=1, size=44, hash=3343
Queue resized to 11
```

```
Consumed message 3: type=1, size=192, hash=23324 — OK
Produced message 5: type=1, size=84, hash=12003
Shrink to 10 requested
vadim@ubuntu:~/Projects/Lab05.2/build$
```

Рисунок 3.2 – Результаты работы программы второй части задания

4 ВЫВОД

В ходе первого задания пятой лабораторной работы реализована модель производитель-потребитель с использованием потоков, POSIX-семафоров и мьютекса. Программа обеспечивает синхронизированный доступ к очереди, поддерживает изменение ее размера и проверку целостности сообщений через хэш-суммы. Семафоры `empty` и `full` гарантируют корректную обработку состояний очереди, а функция `resize_queue` безопасно управляет размером, предотвращая потерю данных. Результаты подтверждают стабильную работу и эффективную синхронизацию.

В ходе второго задания пятой лабораторной работы реализована та же модель, но с условными переменными вместо семафоров. Условные переменные `not_full` и `not_empty` обеспечивают гибкую синхронизацию, а функции очистки предотвращают взаимоблокировки при отмене потоков. Изменение размера очереди и проверка хэш-сумм работают корректно. Программа демонстрирует стабильность, но требует более сложной реализации по сравнению с семафорами.