Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин

Лабораторная работа №4
по курсу
«Операционные системы и системное программирование»
на тему
«Задача производители-потребители для процессов»

Выполнил: студент группы 350501
Слепица О.Н.
Проверил: старший преподаватель каф. ЭВМ
Поденок Л.П.

Минск 2025

1 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Основной процесс создает очередь сообщений, после чего ожидает и обрабатывает нажатия клавиш, порождая и завершая процессы двух типов — производители и потребители.

Очередь сообщений представляет собой классическую структуру — кольцевой буфер, содержащий указатели на сообщения, и пара указателей на голову и хвост. Помимо этого очередь содержит счетчик добавленных сообщений, счетчик извлеченных и количество свободного места в очереди.

Производители формируют сообщения и, если в очереди есть место, перемещают их туда. Потребители, если в очереди есть сообщения, извлекают их оттуда, обрабатывают и освобождают память с ними связанную.

Для работы используются два семафора для заполнения и извлечения, а также мьютекс или одноместный семафор для монопольного доступа к очереди.

Производители генерируют сообщения, используя системный генератор случайных чисел `rand(3)` или `rand_r(3)` для `size` и `data`. В качестве результата для `size` используется остаток от деления на 256. Реальный размер сообщения на единицу больше и лежит в интервале (1, 256).

Поле `data` имеет длину, кратную 4-м байтам. При формировании сообщения контрольные данные формируются только из байт сообщения длиной `size` + 1. Значение поля `hash` при вычислении контрольных данных принимается равным нулю.

Для расчета контрольных данных можно использовать любой подходящий алгоритм на выбор студента.

После помещения значения в очередь перед освобождением мьютекса очереди производитель инкрементирует счетчик добавленных сообщений. Затем после освобождения мьютекса выводит строку на `stdout`, содержащую помимо всего новое значение этого счетчика.

Потребитель, получив доступ к очереди, извлекает сообщение и удаляет его из очереди.

Перед освобождением мьютекса очереди инкрементирует счетчик извлеченных сообщений. Затем после освобождения мьютекса проверяет контрольные данные и выводит строку на `stdout`, содержащую помимо всего новое значение счетчика извлеченных сообщений.

При получении сигнала о завершении процесс должен завершить свой цикл и только после этого завершиться, не входя в новый.

Программы компилируются с ключами

```
-W -Wall -Wextra -std=c11 -pedantic
```

Допускается использование ключей

```
-Wno-unused-parameter -Wno-unused-variable.
```

Для компиляции, сборки и очистки используется `make`.

2 ОПИСАНИЕ АЛГОРИТМОВ И РЕШЕНИЙ

Программа реализует классическую модель взаимодействия производителей и потребителей с использованием разделяемой очереди сообщений на основе кольцевого буфера. Архитектура системы включает главный управляющий процесс, процессы-производители и процессы-потребители, синхронизированные через механизмы межпроцессного взаимодействия.

Основной процесс выполняет инициализацию системы: создает очередь сообщений, инициализирует структуры данных для кольцевого буфера и настраивает необходимые объекты синхронизации. Очередь сообщений организована как кольцевой буфер фиксированного размера, содержащий указатели на сообщения, индексы головы и хвоста, а также счетчики добавленных и извлеченных сообщений. Для управления доступом к разделяемым ресурсам используются два счетных семафора `POSIX` (для контроля заполненности и свободного места) и один семафор функционирующий как мьютекс.

Процессы-производители генерируют сообщения случайного размера, используя системный вызов `rand()`. Каждое сообщение содержит поле типа, контрольные данные, размер полезной нагрузки и сами данные. Контрольные данные вычисляются с использованием алгоритма `CRC16` на основе содержимого сообщения. Перед добавлением в очередь производитель проверяет наличие свободного места через соответствующий семафор, после чего получает эксклюзивный доступ к очереди через мьютекс, добавляет сообщение, обновляет счетчики и индексы, а затем освобождает ресурсы.

Процессы-потребители работают аналогично: проверяют наличие сообщений через семафор заполненности, получают доступ к очереди, извлекают сообщение, обновляют состояние буфера и счетчики. После извлечения потребитель проверяет целостность сообщения путем повторного вычисления контрольных данных и сравнивает полученное значение с хранящимся в сообщении. В случае несоответствия выводится предупреждение. Память, выделенная под сообщение, освобождается после обработки.

3 ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ПРОЕКТА

Проект реализует модель взаимодействия производителей (producers) и потребителей (consumers) через общую очередь сообщений с использованием механизмов межпроцессного взаимодействия (IPC) и синхронизации. Система состоит из нескольких ключевых модулей, каждый из которых выполняет определенную функцию.

Главный управляющий модуль (main) выполняет следующие функции:

1) initialize_queue() – создает и настраивает разделяемую память, очередь сообщений и семафоры;
2) initialize() – устанавливает обработчик сигналов SIGUSR1 для корректного завершения процессов;
3) create_prod() – создает процесс-производитель;
4) del_prod() – завершает процесс-производителя;
5) create_con() – создает процесс-потребитель;
6) del_con() – завершает процесс-потребителя;
7) display_menu() – выводит меню управления;
8) show_processes() – отображает список активных процессов;
9) get_option() – считывает выбор пользователя;
10) cleanup_resources() – освобождает ресурсы.

Модуль работы с очередью реализован в файле header.h и выполняет следующие функции:

1) calculate_hash() – вычисляет контрольную сумму сообщения;
2) verify_hash() – проверяет целостность сообщения;
3) Объявление структуры данных queue.

Для работы прозводителей создан модуль, реализованый в файле producer.c. Данный модуль выполняет следующие функции:

1) create_message() – заполняет сообщение данными и вычисляет хеш;
2) Ожидает свободного места (sem_wait(free_space));
3) Захватывает мьютекс (sem_wait(mutex));
4) Добавляет сообщение в очередь (q->buffer[q->tail]);
5) Освобождает мьютекс и увеличивает счетчик items.

Для работы потребителей создан модуль, реализованый в файле consumer.c. Данный модуль выполняет следующие функции:

1) Ожидает появления сообщения (sem_wait(items));
2) Захватывает мьютекс (sem_wait(mutex));
3) Извлекает сообщение из очереди (q->buffer[q->head]);
4) Освобождает мьютекс и увеличивает счетчик free_space;
5) verify_hash() – сравнивает вычисленный хеш с сохраненным.

4 ПОРЯДОК СБОРКИ И ЗАПУСКА ПРОЕКТА

Порядок сборки и запуска состоит в следующем:

1) Разархивировать каталог с проектом;

2) Перейти в каталог с проектом:

```
$cd "Слепица О.Н./lab04";
```

3) Собрать проект используя `make`;

4) После сборки проекта можно использовать, прописав

```
$./bin/main.
```

5 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

1)

```
helga@fedora:~/tar_working_dir/Слепица О.Н./lab04$./bin/main
```

```
╔═══════════════════════════════════╗
║         МЕНЮ УПРАВЛЕНИЯ           ║
╠═══════════════════════════════════╣
║ 1. Создать производителя          ║
║ 2. Удалить производителя          ║
║ 3. Создать потребителя            ║
║ 4. Удалить потребителя            ║
║                                   ║
║ m. Показать меню                  ║
║ l. Список процессов               ║
║ q. Выход                          ║
╚═══════════════════════════════════╝
```

```
Выберите действие: 1
```

2)

```
PRODUCER CREATED. PID: 14645
Producer started (PID: 14645)
PRODUCER 14645: TYPE=41 HASH=4869 SIZE=44 ADDED=1
PRODUCER 14645: TYPE=179 HASH=E526 SIZE=40 ADDED=2
PRODUCER 14645: TYPE=66 HASH=0651 SIZE=106 ADDED=3
3
```

3)

```
CONSUMER CREATED. PID: 14646
Consumer started (PID: 14646)
CONSUMER 14646: TYPE=41 HASH=4869 SIZE=44 EXTRACTED=1
PRODUCER 14645: TYPE=105 HASH=4067 SIZE=213 ADDED=4
CONSUMER 14646: TYPE=179 HASH=E526 SIZE=40 EXTRACTED=2
l
```

4)

```
┌──────────────────────────────────┐
│      АКТИВНЫЕ ПРОЦЕССЫ           │
├──────────────────────────────────┤
│ Основной процесс: 14644          │
├──────────────────────────────────┤
│ Производители:                   │
│     1. PID: 14645                │
├──────────────────────────────────┤
│ Потребители:                     │
│     1. PID: 14646                │
└──────────────────────────────────┘
```