Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра ЭВМ

Дисциплина: Операционные системы и системное программирование

ОТЧЁТ
к лабораторной работе №5
на тему
Потоки исполнения, взаимодействие и синхронизация.

Выполнил студент гр.230501 Лазовский И.А.

Проверил старший преподаватель кафедры ЭВМ
Поденок Л.П.

Минск 2024

1 УСЛОВИЕ ЛАБОРАТОРНОЙ РАБОТЫ.

Задание

Аналогична лабораторной No 4, но только с потоками, семафорами и мьютексом в рамках одного процесса.

Дополнительно обрабатывается еще две клавиши – увеличение и уменьшение размера очереди.

2 ОПИСАНИЕ АЛГОРИТМОВ И РЕШЕНИЙ.

`message`: Эта структура представляет собой сообщение. Она содержит следующие поля:

`type`: Тип сообщения (представленный в виде `uint8_t`).

`hash`: Хэш сообщения (представленный в виде `uint16_t`).

`size`: Размер сообщения (представленный в виде `uint8_t`).

`data`: Указатель на данные сообщения (представленные в виде строки `char*`).

`queue`: Эта структура представляет собой очередь сообщений. Она включает следующие поля:

`head`: Указатель на начало очереди.

`h`: Индекс начала очереди (представленный в виде `int`).

`tail`: Указатель на конец очереди.

`t`: Индекс конца очереди (представленный в виде `int`).

`buff`: Буфер для хранения сообщений (массив структур `message`).

`count_added`: Количество добавленных сообщений (представленное в виде `int`).

`count_extracted`: Количество извлеченных сообщений (представленное в виде `int`).

Функции:

`getSize()`: Генерирует случайный размер сообщения (от 1 до 256 байт).

`getType()`: Определяет тип сообщения (0 или 1) на основе его размера.

`getData()`: Генерирует случайные данные для сообщения (строку из случайных букв).

`FNV1_HASH()`: Вычисляет хэш сообщения с использованием алгоритма `FNV-1`.

`createMessage()`: Создает новое сообщение с случайными данными (размер, тип, данные и хэш).

`start()`: Инициализирует разделяемую память и семафоры.

deleteConsumers(), deleteProducers(): Удаляют потребителей и производителей.
fromProgExit(): Очищает ресурсы при завершении программы.
viewStatus(): Выводит информацию о текущем состоянии очереди.
addMessage(), extractedMessage(): Добавляют и извлекают сообщения из очереди.
addConsumer(), addProducer(): Создают потребителей и производителей.
menu(): Выводит меню с опциями.
viewProcesses(): Выводит информацию о запущенных процессах.
void dec_queue_func(): Функция декремента очереди.
void inc_queue_func(): Функция инкремента очереди.

3. ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ПРОЕКТА.

Проект собирается с помощью makefile. Пример запуска:

```
ilua@fedora:~/Рабочий стол/labFive$ build/debug/main
```

Для запуска проекта нам требуется в терминале запустить программу main. Где программа сразу переходит в цикл обработки символов

В проекте имеется каталог для сборки debug и release. Каталог git для системы контроля версий моего проекта. Директория src с исходным кодом. И makefile для компиляции и сборки проекта.

4. ПОРЯДОК СБОРКИ И ИСПОЛЬЗОВАНИЯ.

Для компиляции и сборки проекта используется makefile.

Порядок сборки:

4CC: Это неявная переменная, которая указывает на компилятор. В данном случае используется gcc.

CFLAGS_DEBUG: Флаги компиляции для режима отладки.

CFLAGS_RELEASE: Флаги компиляции для релизного режима.

DEBUG: Путь к директории с отладочными файлами.

RELEASE: Путь к директории с релизными файлами.

OUT_DIR: Исходно установлен на $(DEBUG), но может изменяться на $(RELEASE) в зависимости от режима сборки.

all: Цель, которая собирает исполняемый файл main.

`$(prog)`: Это правило для сборки исполняемого файла `main`. Оно зависит от объектных файлов `($(objects))`.

`$(OUT_DIR)/%.o`: Это правило для сборки объектных файлов из исходных файлов `.c`.

`ifeq ($(MODE), release)`: Если переменная `MODE` установлена в `release`, то используются флаги для релизного режима.

`.PHONY: clean`: Это объявление говорит `make`, что `clean` - это фиктивная цель (не связанная с файлами).

`clean`: Удаляет все файлы в директориях `$(DEBUG)` и `$(RELEASE)`.

Порядок использования.

1. Компиляция:

Для отладочной сборки: `make` или `make MODE=debug`

Для релизной сборки: `make MODE=release`

2. Очистка:

`make clean` - удаляет все объектные файлы и исполняемый файл.

3. Запуск:

После успешной компиляции запустите исполняемый файл, например: ./build/debug/main.

5. МЕТОД ТЕСТИРОВАНИЯ И РЕЗУЛЬТАТ ТЕСТИРОВАНИЯ.

```
ilua@fedora:~/Рабочий стол/labs/project/labFive#
build/debug/main
Write 'm' to display menu.
1
producer01 producer message: HASH=A618, counter_added=1
1
producer02 producer message: HASH=7409, counter_added=2
1
producer03 producer message: HASH=61A8, counter_added=3
1
producer04 producer message: HASH=7706, counter_added=4
1
producer05 producer message: HASH=61A8, counter_added=5
1
producer01 producer message: HASH=7817, counter_added=6
producer02 producer message: HASH=9D1E, counter_added=7
producer03 producer message: HASH=8B0B, counter_added=8
producer04 producer message: HASH=DCB5, counter_added=9
producer05 producer message: HASH=741D, counter_added=10
-
producer01 producer message: HASH=AC11, counter_added=11
producer02 producer message: HASH=A10B, counter_added=12
```

```
producer03 producer message: HASH=54DA, counter_added=13
producer04 producer message: HASH=CF1A, counter_added=14
s
Queue max size:14
Current size:14
Added:14
Extracted:0
Consumers:0
Producers:5
2
Was delete producer with name:producer04
2
Was delete producer with name:producer03
2
Was delete producer with name:producer02
2
Was delete producer with name:producer01
2
Was delete producer with name:producer00
3
consumer_01 consumer message: HASH=0000, counter_extracted=1

3
consumer_02 consumer message: HASH=7409, counter_extracted=2

3
consumer_03 consumer message: HASH=61A8, counter_extracted=3

-
-
-
sconsumer_01 consumer message: HASH=7706,
counter_extracted=4

Queue max size:11
Current size:10
Added:14
Extracted:4
Consumers:3
Producers:0
consumer_02 consumer message: HASH=61A8, counter_extracted=5

consumer_03 consumer message: HASH=7817, counter_extracted=6

4
Was delete consumer with name:consumer_02
4
```

```
Was delete consumer with name:consumer_01
4
Was delete consumer with name:consumer_00
4
No consumers.1
producer01 producer message: HASH=DB1A, counter_added=15
1
producer02 producer message: HASH=1B05, counter_added=16
1
producer03 producer message: HASH=6394, counter_added=17
1
1
1
s
Queue max size:11
Current size:11
Added:17
Extracted:6
Consumers:0
Producers:5
```