Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №7
по дисциплине ОСиСП

Тема: «Блокировки чтения/записи и условные переменные»

Выполнил:
студент группы 150501
Ковальчук Д.И.

Проверил:
старший преподаватель
_____ Поденок Л. П.

Минск
2023

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Задача «производители-потребители» для потоков с использованием условных переменных.

Задача производители-потребители для потоков. Основной поток создает очередь сообщений, после чего ожидает и обрабатывает нажатия клавиш, порождая и завершая потоки 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 и счетчик извлеченных.

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 две условные переменные для заполнения и извлечения, а также мьютекс для монопольного доступа к очереди.

Сообщения имеют следующий формат (размер и смещение в байтах):

Имя	Размер	Смещение	Описание
`type`	1	0	тип сообщения
`hash`	2	1	контрольные данные
`size`	1	3	длина данных в байтах (от 0 до 256)
`data`	`((size + 3)/4)*4`	4	данные сообщения

Производители генерируют сообщения, используя системный генератор `rand(3)` для `size` и `data`. В качестве результата для `size` используется остаток от деления на 257.

Если остаток от деления равен нулю, `rand(3)` вызывается повторно. Если остаток от деления равен 256, значение `size` устанавливается равным 0, реальная длина сообщения при этом составляет 256 байт.

При формировании сообщения контрольные данные формируются из всех байт сообщения. 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 поднятия семафора 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 поднятия семафора проверяет контрольные данные и выводит строку на `stdout`, содержащую помимо всего новое значение счетчика извлеченных сообщений.

Следует предусмотреть задержки, чтобы вывод можно было успеть прочитать в процессе работы программы. Следует предусмотреть защиту от тупиковых ситуаций из-за отсутствия производителей или потребителей.

Также дополнительно обрабатывается еще две клавиши – увеличение и уменьшение размера очереди.

1.2 Конкурентный доступ к совместно используемому файлу, используя блокировку чтения-записи.

Программа в режиме конкурентного доступа читает из и пишет в файл, содержащий записи фиксированного формата. Формат записей произвольный. Примерный формат записи:

```
struct record_s {
     char      name[80];     // Ф.И.О. студента
     char      address[80];  // адрес проживания
     uint8_t   semester;     // семестр
}
```

Файл должен содержать не менее 10 записей. Создается и наполняется с помощью любых средств. Программа должна выполнять следующие операции:

1) `LST` – Отображение содержимого файла с последовательной нумерацией записей;

2) `GET(Rec_No)` – получение записи с порядковым номером `Rec_No`;

3) Модификацию полей записи;

4) `PUT()` сохранение последней прочитанной и модифицированной записи по месту.

Интерфейс с пользователем на «вкус» студента.

Алгоритм конкурентного доступ к записи

```
     REC <-- get(Rec_No)
Again:
     REC_SAV <-- REC // сохраним копию
     if (REC модифицирована) {
          // блокируем запись для модификации в файле
          lock(Rec_No)
          REC_NEW <-- get(Rec_No) // и перечитываем
          // кто-то изменил запись после получения ее нами
          if (REC_NEW != REC_SAV) {
               unlock(Rec_No) // освобождаем запись и
               // повторим все с ее новым содержимым
               REC <-- REC_NEW
               goto Again
          }
          put(REC, Rec_No) // сохраняем новое содержимое
          unlock(Rec_No)   // освобождаем запись
     }
```

Для отладки и тестирования используется не менее двух экземпляров программы.

2 АЛГОРИТМ

2.1 Задача «производители-потребители» для потоков с использованием условных переменных.

В программе `main` нужно объявить все необходимые переменные (мьютекс для монопольного доступа к очереди, две условные переменные для количества сообщений в очереди и свободных мест в очереди, собственно саму очередь), проинициализировать очередь, проинициализировать условные переменные, проверяя при этом возвращаемые значения функций, и создать мьютекс. Далее нужно войти в бесконечный цикл считывания опций, при вводе опции вызвать соответствующую функцию.

При вводе опции „`q`“ нужно уничтожить мьютекс и условные переменные, а также удалить все созданные потоки.

При вводе опции „`i`“ необходимо увеличить размер очереди на 1, при вводе опции „`d`“ нужно уменьшить размер очереди на 1.

При вводе опции „`1`“ необходимо создать новый поток для производителя и запустить функцию, внутри которой производитель будет генерировать сообщения и класть их в

очередь, при вводе опции „2“ нужно удалить последнего производителя и вернуть занятые им ресурсы.

При вводе опции „3“ необходимо создать новый поток для потребителя и запустить функцию, внутри которой производитель будет проверять контрольную сумму сообщений и извлекать их из очереди, при вводе опции „4“ нужно удалить последнего потребителя и вернуть занятые им ресурсы.

При вводе опции „m“ нужно выводить меню со всеми возможными опциями.

При вводе опции „l“ необходимо вывести id главного потока, а также id потоков потребителей и производителей.

При вводе опции „s“ необходимо вывести текущий размер очереди.

2.2 Конкурентный доступ к совместно используемому файлу, используя блокировку чтения-записи.

Для начала нужно сгенерировать файл с записями, после этого в программе main нужно объявить все необходимые переменные, открыть файл с записями при этом возвращаемые значения функций. Далее нужно войти в бесконечный цикл считывания опций, при вводе опции вызвать соответствующую функцию.

При вводе опции „MOD“ необходимо считать индекс изменяемой записи и ввести новое значение записи, сохранив ее при этом в последнюю прочитанную (модифицированную) запись.

При вводе опции „PUT“ необходимо считать индекс записи, которую хотим проверить, получить запись по индексу и вызвать функцию, реализующую алгоритм конкурентного доступа к записи из постановки задачи.

При вводе опции „LST“ необходимо пройтись в цикле по всем записям, получить их по индексам и вывести.

При вводе опции „GET“ нужно взять запись по индексу и вывести ее, сохранив при этом в последнюю прочитанную (модифицированную) запись.

3 ТЕСТ

3.1 Задача «производители-потребители» для потоков с использованием условных переменных.

```
darya@darya-VivoBook-ASUSLaptop-X521EQ-S533EQ:~/osisp/lab07/
part1$ ./main
```

```
              MENU
         1  CREATE PRODUCER
         2  DELETE PRODUCER
         3  CREATE CONSUMER
         4  DELETE CONSUMER
         i  INCREASE SIZE OF QUEUE
         d  DECREASE SIZE OF QUEUE
         m  SHOW MENU
         l  SHOW THREADS INFO
         s  SHOW SIZE OF QUEUE
         q  EXIT PROGRAMM

1

PRODUCER CREATED. PID: 139646166103744

1

PRODUCER CREATED. PID: 139646157711040

1

PRODUCER CREATED. PID: 139646149318336

3

CONSUMER CREATED. PID: 139646140925632

3

CONSUMER CREATED. PID: 139646132532928

3

CONSUMER CREATED. PID: 139646124140224

PROCESS (TID): 139646166103744 CREATED MESSAGE:
TYPE: 0
HASH: 637F
SIZE: 66
AMOUNT OF PUSHED MESSAGES: 1

PROCESS (TID): 139646157711040 CREATED MESSAGE:
TYPE: 0
HASH: 0A3C
SIZE: 188
AMOUNT OF PUSHED MESSAGES: 2

PROCESS (TID): 139646149318336 CREATED MESSAGE:
TYPE: 0
```

```
HASH: 33E3
SIZE: 170
AMOUNT OF PUSHED MESSAGES: 3

PROCESS (TID): 139646140925632 CONSUMED MESSAGE:
TYPE: 0
HASH: 637F
SIZE: 66
AMOUNT OF EXTRACTED MESSAGES: 1

PROCESS (TID): 139646132532928 CONSUMED MESSAGE:
TYPE: 0
HASH: 0A3C
SIZE: 188
AMOUNT OF EXTRACTED MESSAGES: 2

PROCESS (TID): 139646124140224 CONSUMED MESSAGE:
TYPE: 0
HASH: 33E3
SIZE: 170
AMOUNT OF EXTRACTED MESSAGES: 3

PROCESS (TID): 139646166103744 CREATED MESSAGE:
TYPE: 0
HASH: 2B6F
SIZE: 83
AMOUNT OF PUSHED MESSAGES: 4

PROCESS (TID): 139646157711040 CREATED MESSAGE:
TYPE: 0
HASH: FD87
SIZE: 58
AMOUNT OF PUSHED MESSAGES: 5

PROCESS (TID): 139646149318336 CREATED MESSAGE:
TYPE: 0
HASH: A702
SIZE: 132
AMOUNT OF PUSHED MESSAGES: 6

i

INCREASED SIZE OF QUEUE. NOW IT IS 11

PROCESS (TID): 139646140925632 CONSUMED MESSAGE:
TYPE: 0
HASH: 2B6F
SIZE: 83
AMOUNT OF EXTRACTED MESSAGES: 4

dPROCESS (TID): 139646132532928 CONSUMED MESSAGE:
TYPE: 0
```

```
HASH: FD87
SIZE: 58
AMOUNT OF EXTRACTED MESSAGES: 5



DECREASED SIZE OF QUEUE. NOW IT IS 10

PROCESS (TID): 139646124140224 CONSUMED MESSAGE:
TYPE: 0
HASH: A702
SIZE: 132
AMOUNT OF EXTRACTED MESSAGES: 6

dPROCESS (TID): 139646166103744 CREATED MESSAGE:
TYPE: 0
HASH: A2BD
SIZE: 92
AMOUNT OF PUSHED MESSAGES: 7



DECREASED SIZE OF QUEUE. NOW IT IS 9

dPROCESS (TID): 139646157711040 CREATED MESSAGE:
TYPE: 0
HASH: 1F17
SIZE: 222
AMOUNT OF PUSHED MESSAGES: 8

PROCESS (TID): 139646149318336 CREATED MESSAGE:
TYPE: 0
HASH: E6ED
SIZE: 123
AMOUNT OF PUSHED MESSAGES: 9

l

DECREASED SIZE OF QUEUE. NOW IT IS 8

       PROCESSES(PID)
             MAIN
        139646169048896
           PRODUCERS
        139646166103744
        139646157711040
        139646149318336
          CONSUMERS
        139646140925632
        139646132532928
        139646124140224
```

```
PROCESS (TID): 139646140925632 CONSUMED MESSAGE:
TYPE: 0
HASH: A2BD
SIZE: 92
AMOUNT OF EXTRACTED MESSAGES: 7

PROCESS (TID): 139646132532928 CONSUMED MESSAGE:
TYPE: 0
HASH: 1F17
SIZE: 222
AMOUNT OF EXTRACTED MESSAGES: 8

PROCESS (TID): 139646124140224 CONSUMED MESSAGE:
TYPE: 0
HASH: 637F
SIZE: 66
AMOUNT OF EXTRACTED MESSAGES: 9

PROCESS (TID): 139646166103744 CREATED MESSAGE:
TYPE: 0
HASH: 73DC
SIZE: 178
AMOUNT OF PUSHED MESSAGES: 10

PROCESS (TID): 139646157711040 CREATED MESSAGE:
TYPE: 0
HASH: 8ED1
SIZE: 234
AMOUNT OF PUSHED MESSAGES: 11

qPROCESS (TID): 139646149318336 CREATED MESSAGE:
TYPE: 0
HASH: 44FC
SIZE: 116
AMOUNT OF PUSHED MESSAGES: 12



CONSUMER DELETED. PID: 139646124140224

CONSUMER DELETED. PID: 139646132532928

CONSUMER DELETED. PID: 139646140925632

PRODUCER DELETED. PID: 139646149318336

PRODUCER DELETED. PID: 139646157711040

PRODUCER DELETED. PID: 139646166103744

PROGRAMM FINISHED
```

3.2 Конкурентный доступ к совместно используемому файлу, используя блокировку чтения-записи.

```
darya@:~/osisp/lab07/part2$ ./main

MENU:
MENU - SHOW MENU
LST - AHOW ALL RECORDS
GET - GET RECORD
MOD - EDIT RECORD
PUT - PUT LAST USED RECORD TO FILE
QUIT - FINISH PROGRAMM

LST
RECORDS:
NUMBER        NAME              ADDRESS         SEMESTER
-----------------------------------------------------------
   1      MARGARITA              SMORGON           4
   2           VLAD               BRAGIN           4
   3          DARYA               GRODNO           4
   4        TATIANA                 LIDA           4
   5         ALEXEI               BREST            2
   6         MIKOLA               GOMEL            4
   7         NIKITA         KALINKOVICHI           4
   8      ALEXANDRA           MOLODECHNO           4
   9       VIOLETTA              BEREZINO           4
  10          ALINA              ZHODINO           4

CURRENT RECORD:
NUMBER        NAME              ADDRESS         SEMESTER
-----------------------------------------------------------
  10          ALINA              ZHODINO           4


5
GET
WHAT RECORD WOULD YOU LIKE (INPUT):
5
   5         ALEXEI               BREST            2
CURRENT RECORD:
NUMBER        NAME              ADDRESS         SEMESTER
-----------------------------------------------------------
   5         ALEXEI               BREST            2


MOD
CURRENT RECORD:
NUMBER: 5
NAME: ALEXEI
ADDRESS: BREST
SEMESTER: 2
```

```
INPUT WHAT TO EDIT:
1 - NAME
2 - ADDRESS
3 - SEMESTER
0 - EXIT

2
INPUT ADDRESS:
CHERNICHKI
CURRENT RECORD:
NUMBER: 5
NAME: ALEXEI
ADDRESS: CHERNICHKI
SEMESTER: 2

INPUT WHAT TO EDIT:
1 - NAME
2 - ADDRESS
3 - SEMESTER
0 - EXIT

0
CURRENT RECORD:
NUMBER        NAME              ADDRESS         SEMESTER
-------------------------------------------------------------
   5         ALEXEI           CHERNICHKI           2


PUT
INPUT ENTER TO SAVE

RECORD SAVED!
CURRENT RECORD:
NUMBER        NAME              ADDRESS         SEMESTER
-------------------------------------------------------------
   5         ALEXEI           CHERNICHKI           2


LST
RECORDS:
NUMBER        NAME              ADDRESS         SEMESTER
-------------------------------------------------------------
   1      MARGARITA              SMORGON           4
   2           VLAD               BRAGIN           4
   3          DARYA               GRODNO           4
   4         TATIANA                 LIDA           4
   5         ALEXEI           CHERNICHKI           2
   6         MIKOLA                GOMEL           4
   7         NIKITA         KALINKOVICHI           4
   8      ALEXANDRA           MOLODECHNO           4
   9       VIOLETTA             BEREZINO           4
```

```
  10          ALINA               ZHODINO           4

CURRENT RECORD:
NUMBER        NAME               ADDRESS         SEMESTER
-------------------------------------------------------
  10          ALINA               ZHODINO           4


QUIT
PROGRAMM FINISHED
```

4 ВЫВОД

В ходе лабораторной работы была решена задача производители-потребители для потоков, были изучены механизмы работы общей памяти, семафоров, были предусмотрены тупиковые ситуации, если отсутвуют потребители или производители. Также были использованы такие функции как `pthread_cond_init()`, `pthread_cond_destroy()`, `pthread_cond_wait()`, `pthread_cond_signal()`, `pthread_mutex_lock()`, `pthread_join()`, `pthread_mutex_unlock()`, `pthread_create()`. `pthread_cancel()`.

Также в ходе лабораторной работы был изучен конкурентный доступ к совместно используемому файлу, используя блокировку чтения-записи, при этом использовалось 2 экземпляра программы. Во время выполнения были изучены следующие системные вызовы: `fcntl(F_GETLK, F_SETLK, F_SETLKW, F_UNLK)`