

ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

ФАКУЛЬТЕТ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Кафедра вычислительных систем

ОТЧЁТ

по лабораторной работе №2
по дисциплине «Параллельные вычислительные технологии»

«Разработка параллельных программ
с использованием библиотеки POSIX Threads»

Вариант №2

Выполнил:
студент группы ИУ-423
Горлов Р.В.

Проверил:
к.т.н. Пазников А.А.

Введение

POSIX Threads — стандарт реализации потоков (нитей) выполнения. Стандарт *POSIX.1c, Threads extensions (IEEE Std 1003.1c-1995)* определяет API для управления потоками, их синхронизации и планирования.

Реализации данного API существуют для большого числа UNIX-подобных ОС (GNU/Linux, Solaris, FreeBSD, OpenBSD, NetBSD, OS X), а также для Microsoft Windows и других ОС.

Библиотеки, реализующие этот стандарт (и функции этого стандарта), обычно называются **std::thread**, **std::mutex**, **boost::barrier**.

В данном варианте лабораторной работы при помощи библиотек POSIX std::threads и std::mutex на языке C++ была реализована потокобезопасная версия структуры данных типа двусвязная очередь.

Потокобезопасность структуры данных проверяется за счёт использования свободно распространяемого инструмента Valgrind. Который позволяет проверять наличие гонок данных и других ошибок в многопоточных приложениях.

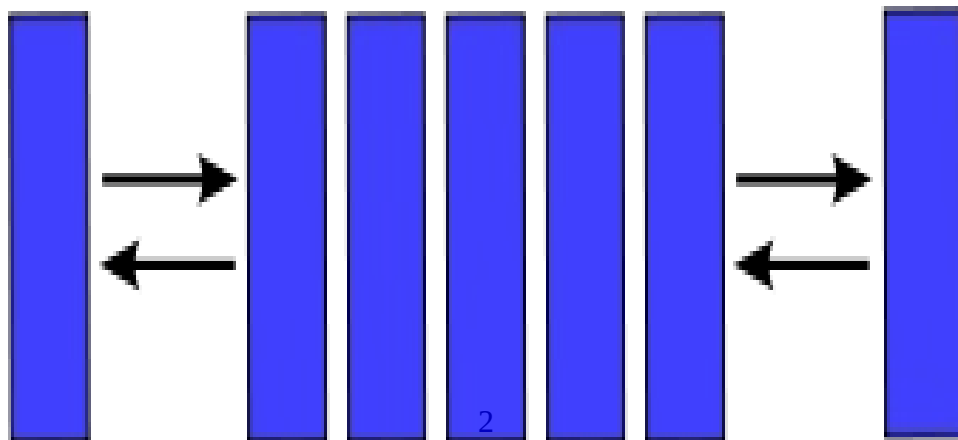
Описание структуры данных

Двусвязная очередь - структура данных, в которой элементы можно добавлять и удалять как в начало, так и в конец, то есть дисциплинами обслуживания являются одновременно FIFO и LIFO.

Типовые операции:

1. PushBack — добавление в конец очереди.
2. PushFront — добавление в начало очереди.
3. PopBack — выборка с конца очереди.
4. PopFront — выборка с начала очереди.
5. Проверка наличия элементов.
6. Очистка.

Ниже приведена логическая схема доступа к данным очереди.



Реализация алгоритма

В данном случае была использована классическая реализация двусвязной очереди на языке C++. Гарантия отсутствия гонок данных обеспечивается за счет использования мьютексов.

Конструкция `unique_lock<mutex> lock (changeMutex);` позволяет избежать не только гонок данных в потоках, но и безопасность всей программы в случае возникновения исключений.

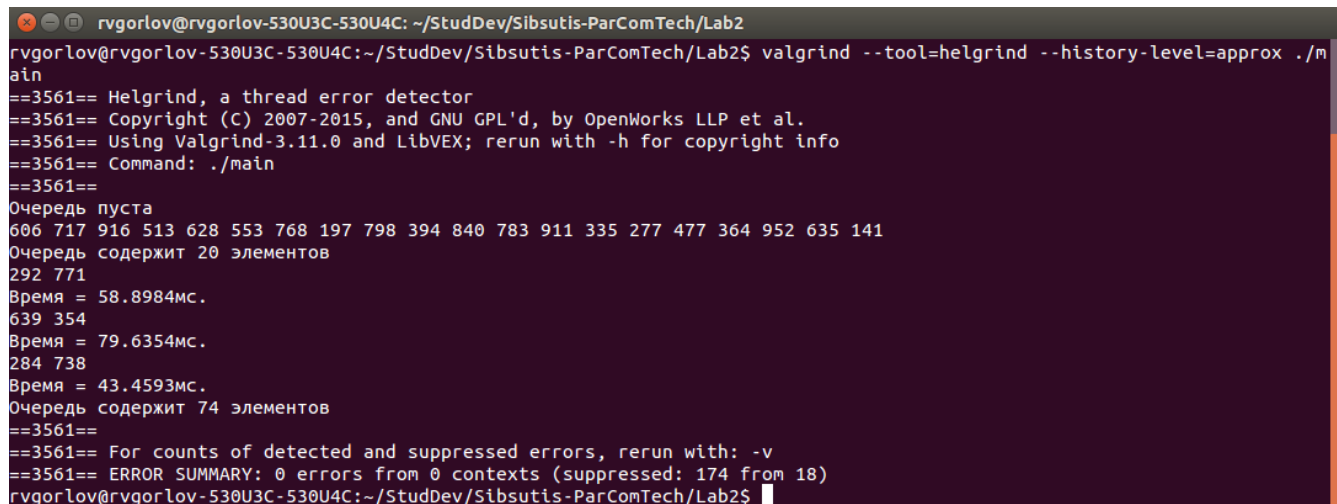
В данной реализации создается максимально доступное количество потоков для тестируемой системы. Все потоки синхронизированы в начале с помощью `boost::barrier` и выполняют N операций над структурой данных одновременно.

Проверка алгоритма

Данная реализация «двусвязной очереди» была протестирована на наличие гонок данных или других ошибок в многопоточной части алгоритма с помощью распространяемого по лицензии GNU General Public License, version 2 программного пакета Valgrind. В частности с помощью пакета Helgrind. Тестирование производилось в среде Ubuntu Linux с помощью команды:

```
valgrind --tool=helgrind --history-level=approx ./main
```

Результат выполнения программы предоставлен ниже.



```
rvgorlov@rvgorlov-530U3C-530U4C: ~/StudDev/Sibsutis-ParComTech/Lab2
rvgorlov@rvgorlov-530U3C-530U4C:~/StudDev/Sibsutis-ParComTech/Lab2$ valgrind --tool=helgrind --history-level=approx ./main
==3561== Helgrind, a thread error detector
==3561== Copyright (C) 2007-2015, and GNU GPL'd, by OpenWorks LLP et al.
==3561== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==3561== Command: ./main
==3561==
Очередь пуста
606 717 916 513 628 553 768 197 798 394 840 783 911 335 277 477 364 952 635 141
Очередь содержит 20 элементов
292 771
Время = 58.8984мс.
639 354
Время = 79.6354мс.
284 738
Время = 43.4593мс.
Очередь содержит 74 элементов
==3561==
==3561== For counts of detected and suppressed errors, rerun with: -v
==3561== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 174 from 18)
rvgorlov@rvgorlov-530U3C-530U4C:~/StudDev/Sibsutis-ParComTech/Lab2$
```

Вывод

В ходе выполнения данной лабораторной работы была реализована и протестирована структура данных «двусвязная очередь» с использованием библиотеки POSIX Threads на языке C++. В ходе реализации были изучены следующие аспекты: создание потоков, передача аргументов в поток, завершение потоков. Тесты показали отсутствие ошибок в во время выполнения многопоточной части программы.

Также можно отметить, что данная реализация является простым примерном использования библиотеки c++ thread.