# Отчет по 5 домашней работе курса ABC

Шипиль Анна (БПИ203)

Вариант 30

## Условие:

*30. Первая задача об Острове Сокровищ.*

Шайка пиратов под предводительством Джона Сильвера высадилась на берег Острова Сокровищ. Не смотря на добытую карту старого Флинта, местоположение сокровищ попрежнему остается загадкой, поэтому искать клад приходится практически на ощупь. Так как Сильвер ходит на деревянной ноге, то самому бродить по джунглям ему не с руки. Джон Сильвер поделил остров на участки, а пиратов на небольшие группы. Каждой группе поручается искать клад на одном из участков, а сам Сильвер ждет на берегу. Пираты, обшарив свою часть острова, возвращаются к Сильверу и докладывают о результатах. Требуется создать многопоточное приложение с управляющим потоком, моделирующее действия Сильвера и пиратов.

1. Поставленную задачу я решила реализовать с использованием библиотеки POSIX Threads языка программирования C.

2. **Модель**: в качестве модели моего многопоточного приложения я выбрала модель “Управляющий и рабочие” (она же модель делегирования), тк всем процессом руководит Сильвер (руководитель пиратов), а собственно работу выполняют пираты. Таким образом, Сильвер это управляющий поток, пираты - рабочие, остальные потоки.
   “*Центральный поток («управляющий») создает потоки («рабочие»), назначая каждому из них задачу. Управляющий поток может ожидать до тех пор, пока все потоки не завершат выполнение своих задач*” (источник)
   В моей реализации Сильвер делит пиратов на группы по одному и отправляет их по участкам, после изучения предложенного участка пират возвращается к Сильверу за дальнейшими распоряжениями - либо его отправляют на следующий участок, либо отправляют отдыхать (если сокровище найдено или на всех оставшихся участках уже есть другие пираты).
   *“В качестве альтернативного варианта управляющий поток может создать пул потоков, которым будут переназначаться новые запросы.”* (источник) - именно так я делаю в данной задаче, главный поток дает задачи рабочим, задачи он придумывает сам (посмотреть какой-то участок острова).

3. В своем решении я не создаю отдельный pthread_t для действий Сильвера, потому что сама программа (функция SilverLeads(), описывающая действия Сильвера) уже является потоком (для большей наглядности можно было бы и создать отдельный pthread_t, но я решила, что это неоптимально и выбрала считать функцию SilverLeads() главным потоком). Создаю массив потоков остальных пиратов, Сильвер выдает им новые участки на проверку, пока они не найдут сокровища. Вывод программы описывает происходящее. Т.к. значения глобальных переменных меняет только Сильвер, не возникнет проблем, что несколько потоков что-то делают с одними и теми же данными. Пираты могут только читать данные - проверять участки, меняет же данные только управляющий поток. Таким образом, ни мьютексы, ни критические секции мне не пригодились.

4. **Ввод-вывод данных:** ввод происходит с использованием файла и командной строки. Пользователь вводит имя входного файла в качестве аргумента командной строки при запуске программы. Если пользователь хочет продублировать вывод программы в файл (по умолчанию он выводится в консоль), то вторым аргументом должен быть указан выходной файл. Во входном файле 3 натуральных числа: размеры острова (два числа) и количество пиратов. При некорректном вводе программа сообщает об этом пользователю.
   После чтения данных из файла программа просит ввести через консоль координаты сокровища (или выбрать функцию генерации их рандомно).
   Все события и результаты работы программы выводятся в консоль, а также дублируются в файл, если пользователь его указал. Общее время работы программы (с момента выбора координат клада до момента завершения программы) также выводится (в секундах).

5. **Тестирование программы**: в репозитории представлена папка с различными входными данными. Так же я записала файлы с результатами работы. В предложенных вариантах входных данных рассмотрены разные крайние случаи, для каждого варианта я пробовала два сценария: случайная генерация координат клада и ввод с консоли.

   Я использовала следующие команды для сборки файлов и для запуска программы в командной строке Linux:

   **gcc -pthread main.c**

   **./a.out tests/test1.txt tests/test1_output_random.txt**

   (во второй команде вместо tests/test1.txt - любой входной файл, вместо tests/test1_output_random.txt - любой выходной файл)

6. Размеры исходных файлов: main.c - 4КБ, PirateCrew.c - 3 КБ, PirateCrew.h - 1 КБ. Размер исполняемого файла: a.out - 18 КБ

   Время работы на разных входных данных:

| | |
|---|---|
| Карта 50x50, 100 пиратов, координаты клада (27, 45) | 80.377907 секунд |
| Карта 50x50, 100 пиратов, координаты клада (48, 47) | 118.190998 секунд |
| Карта 5x5, 1 пиратов, коорд. клада (4, 4) | 25.007420 секунд |
| Карта 5x5, 1 пиратов, коорд. клада (3, 4) | 20.006177 секунд |
| Карта 25x1, 7 пиратов, коорд. клада (12, 0) | 2.007336 секунд |
| Карта 1x1, 8 пиратов, коорд. клада (0, 0) | 1.000 секунд |
| Карта 100x100, 9 пиратов, коорд. клада (18, 96) | 240.031947 секунд |
| Карта 100x100, 9 пиратов, коорд. клада (20, 30) | 255.745693 секунд |

   В более наглядном виде:

| | |
|---|---|
| Проверка 1400 ячеек, 100 пиратов | 80.377907 секунд |
| Проверка 2500 ячеек, 100 пиратов | 118.190998 секунд |
| Проверка 25 ячеек, 1 пиратов | 25.007420 секунд |
| Проверка 20 ячеек, 1 пиратов | 20.006177 секунд |
| Проверка 14 ячеек, 7 пиратов | 2.007336 секунд |
| Проверка 1 ячеек, 8 пиратов | 1.000 секунд |
| Проверка 1899 ячеек , 9 пиратов | 240.031947 секунд |
| Проверка 2034 ячеек, 9 пиратов | 255.745693 секунд |

7. **Выводы**: если пиратов мало, время на поиск сокровища по конкретной координате (в моем решении это 1 секунда) ощутимо, если же пиратов больше, они ищут одновременно и поэтому быстрее. Таким образом, использование нескольких потоков в несколько раз ускоряет процесс, по сравнению с одним потоком.