Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 6

з дисципліни «Операційні системи»

Тема: «Потоки. Управління»

Виконав:

студент 3 курсу

групи КС-32

Безрук Юрій Русланович

Перевірив: Споров О.Є.

Харків – 2020

Целью данной работы является продолжение изучения потоков исполнения и знакомство с дополнительными возможностями управления, такими как атрибуты потоков, отмена потоков, потоковые данные и обработчики очистки.

# ХОД РАБОТЫ

Во всех заданиях по умолчанию отключена буферизация стандартного потока вывода в консоль.

**Задание 1**

В задании 1 требуется написать программу с использованием отсоединенного потока (потока, который не сохраняет информацию о себе после завершения, т.е. его не нужно ожидать). Программа создаёт указанное количество расчетных потоков, которые при создании засыпают на число секунд, соответствующее их номеру, генерируют случайное число в диапазоне [1, 10], помещают его в глобальный массив и засыпают уже на это сгенерированное число секунд. В тоже время отсоединённый поток раз в секунду выводит состояние массива, пока он не окажется заполнен.

Несколько слов о доп. определениях: структура thread\_arg содержит в себе переменную, которая будет представлять собой номер потока и адрес переменной, в которой хранится текущее количество значимых чисел в глобальном массиве (чисел, помещённых потоками). Функция seed() как и раньше, задаёт зерно для псевдослучайных чисел, а get\_random() – возвращает случайное число в указанном диапазоне.

При запуске программы число расчётных потоков получается из аргументов командной строки (по умолчанию - 1). После происходит выделение памяти под динамический глобальный массив. Далее создаётся массив идентификаторов потоков и вместе с количеством передаётся в функцию create\_threads(), в которой производится вся работа с потоками.

В ней для начала указывается начальная заполненность массива и создаётся и инициализируется всё, что нужно для работы отсоединенного потока – структура атрибутов с значением PTHREAD\_DETACHED и аргументы, передаваемые так же при помощи структуры thread\_args, но вместо индекса отсоединённому потоку передается значение общего количества отсоединенных потоков (размер массива).

Далее создается массив аргументов для расчётных потоков и они циклически создаются, после чего выполнятся ожидание завершения каждого из них.

Потоки выполняют свою работу внутри функции thread\_func, а отсоединенный поток – внутри detached\_func.

Отсоединённый поток завершается, как только значение заполненности становится равным размеру массива. Первый поток его не ожидает. Пример работы программы:

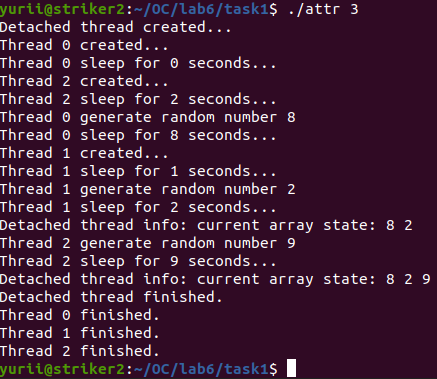


Рисунок 1 - пример работы программы с отсоединенным потоком

Задания 2-4 демонстрируют различные режимы и типы «отменяемости» потока.

**Задание 2**

В задании 2 рассматривается процесс создания асинхронно отменяемого потока (т.е. потока, который можно отменить в любой момент).

С консоли программа получает время задержки в секундах (по умолчанию – 1). Далее основной поток создаёт дочерний, засыпает на переданное количество секунд, просит дочерний поток прекратить свою работу и ожидает его завершения. После этого выводится сообщение о том, каким образом был завершен дочерний поток.

В дочернем потоке устанавливается тип отменяемости – асинхронный и бесконечно выводятся информационные сообщения о текущей итерации потока, которые производятся раз в секунду. Таким образом, на самом деле, поток может завершить вою работу только если будет прерван.

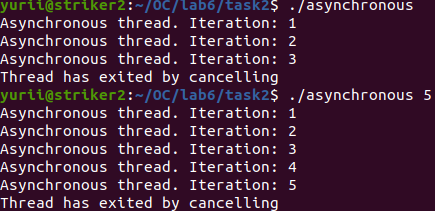


Рисунок 2 - пример асинхронно отменяемого потока

**Задание 3**

Задание 3 является точной копией задания 2, за исключением того факта, что теперь дочерний поток делается неотменяемым вовсе, и, для того, что бы он не зациклился на бесконечность, итерации он выводит строго определенное количество раз, передаваемое потоковой функции через аргументы, и равное удвоенному времени задержки.



Рисунок 3 - пример неотменяемого потока

**Задание 4**

Задание 4 так же является модификацией задания 2 с тем, что бы применить в нём синхронно отменяемый поток (т.е. поток, который можно отменить только в конкретных местах, точках выхода). Главный поток работает также, как и в предыдущих программах.

Дочерний поток устанавливает тип «отменяемости» - синхронный, и вычисляет приближение числа Пи по формуле Лейбница. На каждой итерации выводится текущее приближение. Поток производит 1 000 000 итераций, после чего завершается (число итераций было увеличено для наглядности, т.к. при числе итераций 100 000, поток успевал произвести их менее, чем за секунду). В случае если поток был отменен, выводится соответствующее сообщение, а если досчитал до конца – возвращенный результат.

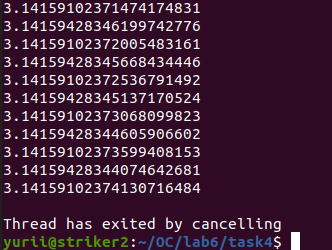


Рисунок 4 - пример синхронно отменяемого потока. Число итераций - 1 000 000, ожидание - 5 секунд. (поток не успел завершить вычисления)

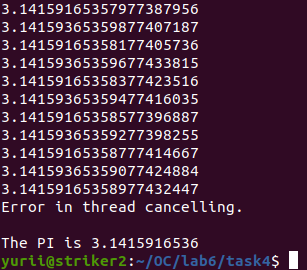


Рисунок 5 - пример синхронно отменяемого потока. Число итераций - 1 000 000, ожидание - 10 секунд. (поток успел завершить вычисления)

**Задание 5**

Задание 5 рассчитано на работу с потоковыми данными. В программе создаётся указанное количество потоков и ожидается завершение их работы. При этом каждый поток выводит в стандартный поток вывода случайное количество строк вида «thread\_id thread is running. Iteration: x/y. Random: z», где thread\_id – идентификатор потока, х – номер текущей итерации, у – общее количество итераций для данного потока (генерируется случайно на заданном диапазоне), z – случайное число, генерирующее на всё том же диапазоне. При этом неизменяемая в цикле часть строки хранится в потоковых данных.

Структура range используется для передачи потоку необходимого диапазона псевдослучайных чисел. Функция digits() возвращает количество знаков числа. Потоковые данные подключаются при помощи ключа str\_key, который инициализируется в потоковой функции при помощи метода однократной инициализации pthread\_once (инициализация происходит только в первом потоке, вызвавшем эту функцию). С ним так же ассоциирована функция очистки, освобождающая память, которая была выделена под строку.

Пример работы программы (создаётся 3 потока, диапазон случайных чисел – от 1 до 5):

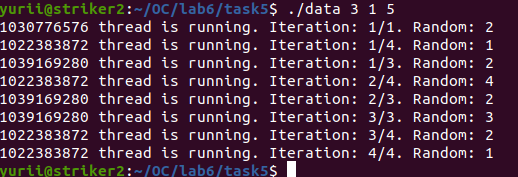


Рисунок 6 - пример работы программы с использованием потоковых данных

**Задание 6**

Задание 6 заключается в перестройке задания 2 с использованием обработчиков очистки. Иными словами, с добавлением функции, которая будет вызываться автоматически по завершению потока.

К программе добавляется функция-обработчик завершения потока, которая показывает, на какой итерации поток завершил свою работу, когда произошла отмена потока.

В потоковой функции регистрируются обработчик очистки макросом pthread\_cleanup\_push(). Из-за особенностей реализации добавляем так же макрос pthread\_cleanup\_pop().

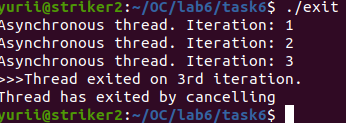


Рисунок 7 - пример работы функции-обработчика завершения потока

# ВЫВОДЫ

Таким образом, в ходе выполнения данной работы были рассмотрены возможности управления потоками, такие как атрибуты потоков, потоковые данные, обработчики очистки, а также различные способы отмены потока. Соответствующие исходные файлы, исполняемые файлы и скрипты сборки помещены в архив.