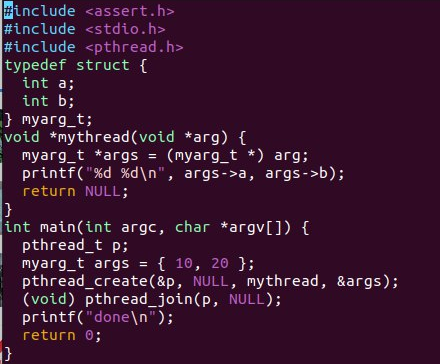
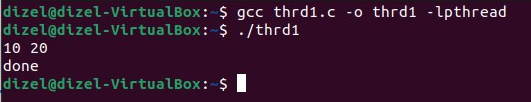
Синев Денис Евгеньевич

**Пример 1 из Презентации**

****

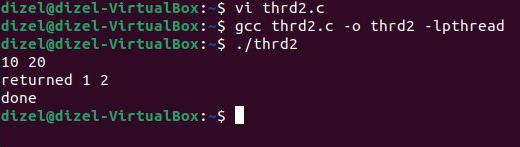
В этой программе мы создаем поток, в который передаем 2 аргумента из типа myarg\_t. После создания поток может перевести аргумент к заранее заданному типу и распаковать аргументы нужным способом.



**Пример 2 из Презентации**

****

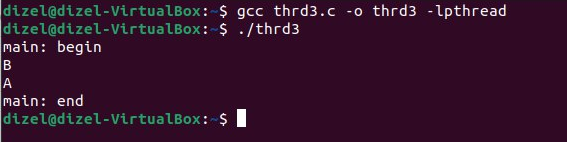
В этой программе мы создаем поток, который выполняет функцию mythread с аргументами args. В этой функции выводятся значения переданных аргументов, создается новая структура с значениями 1 и 2, которая в итоге возвращается из функции. В главной функции мы получаем результат выполнения потока, выводим на экран значения поля a и b новой структуры, освобождаем память и завершаем программу.



**Пример 3 из Презентации**

****

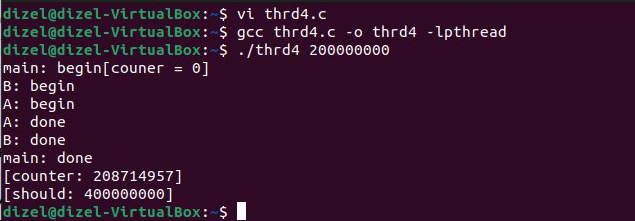
В данной программе имеем три потока: главный поток p1 и p2. Они выводят переданные им аргументы A и B соответственно.



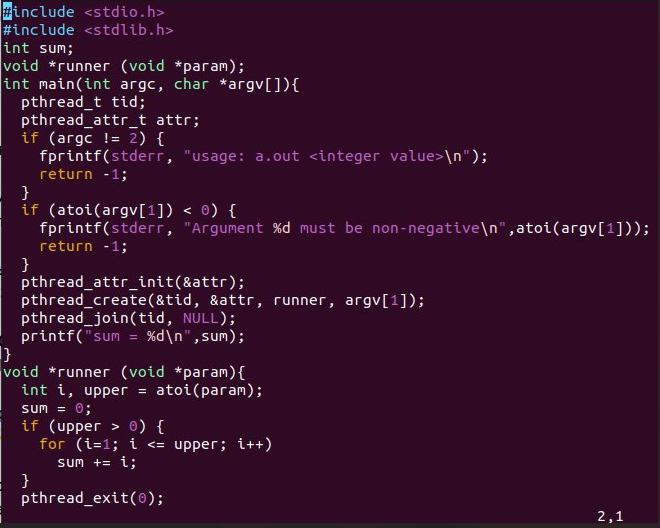
**Пример 4 из Презентации**

****

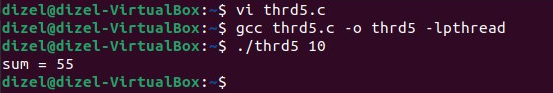
Программа создает два потока, которые выполняют функцию mythread. Каждый поток увеличивает переменную counter на значение max, которое мы задаем при запуске. После завершения работы потоков, в главной функции main выводится текущее значение переменной counter и ожидаемое значение (max\*2). Таким образом, программа считает сумму переменной counter с помощью двух потоков и выводит результат на экран.



**Пример 5 из Презентации**



В данной программе создается поток, который выполняет функцию runner. После создания потока основной поток ждет его завершения с помощью функции pthread\_join. Функция runner вычисляет сумму всех чисел от 1 до числа, переданного в качестве аргумента потоку. Полученная сумма затем выводится на экран основным потоком.



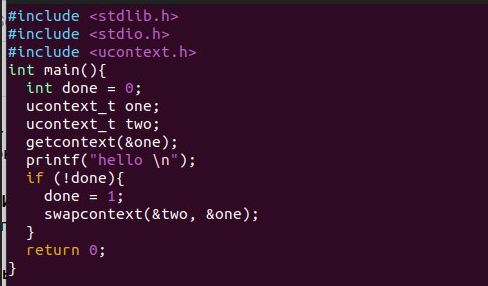
**Пример 6 из Презентации**

****

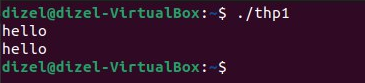
Данная программа создает два потока, которые параллельно выполняют вычисление суммы элементов массива. В функции mythread происходит вычисление суммы элементов переданного массива. В начале функция преобразует void указатель vptr в struct task\_data указатель pd. Затем происходит итерация по элементам массива, вычисление суммы и запись ее в поле s структуры task\_data. В функции main создается массив data длиной N (1024\*1024) и заполняется случайными числами. Затем создаются два объекта t1 и t2 структуры task\_data, которые содержат указатели на разные части массива data, разделенного пополам. Затем создаются два потока th1 и th2, каждому из которых передается соответствующий объект t1 и t2. После завершения работы потоков выводится на экран среднее значение сумм элементов массива (t1.s + t2.s) / N и общее среднее значение всех элементов массива s / N. Таким образом, программа использует многопоточность для параллельного вычисления суммы элементов массива, что позволяет ускорить выполнение программы.

****

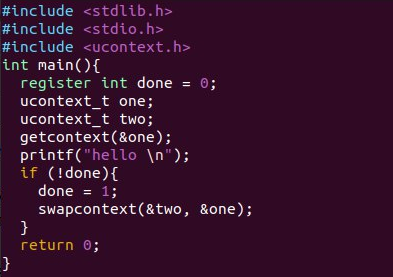
**Пример 1.1 из Пдф**

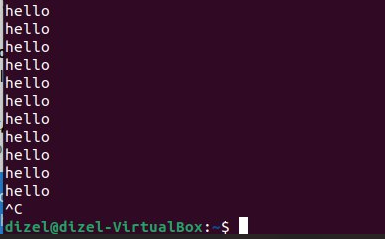
****

При запуске программы мы видим два вывода “hello”, так как done = 0. Сначала работает второй контекст, только потом, когда done становится равной единице, у нас начинает работать первый контекст и мы видим второе “hello”



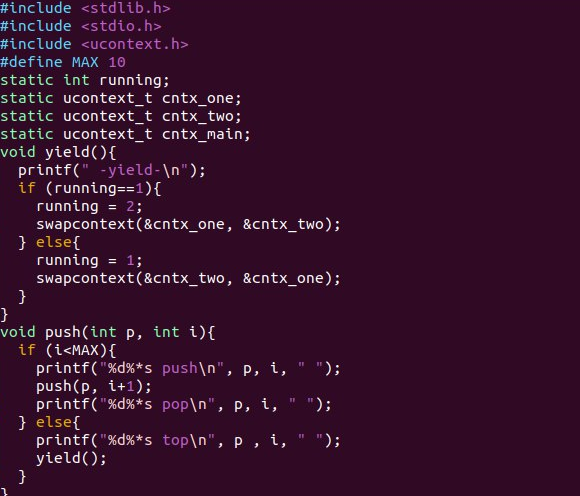
**Пример 1.2 из Пдф**

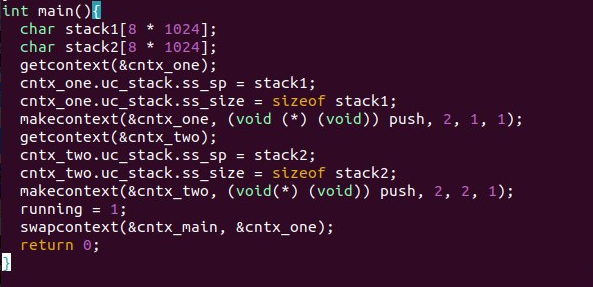
****

****

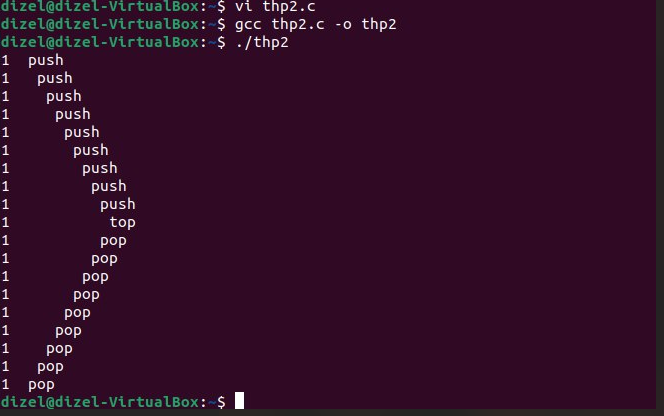
В итоге видим бесконечный вывод hello, пока не остановим работу нажатием Ctrl + C

**Пример 1.3 из Пдф**

****

****

Программа выполняет заполнение и освобождение стека.

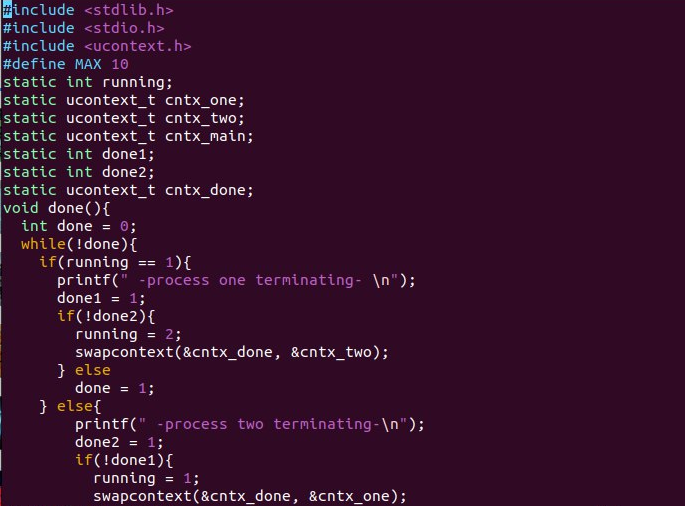
****

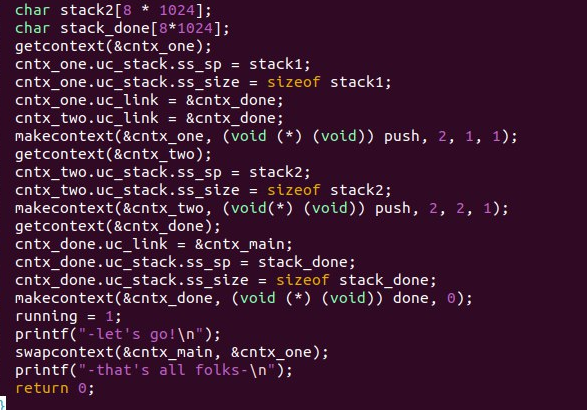
Но если мы добавим вызов yield в push(), то контексты начнут выполняться по очереди.

****

****

**Пример 1.4 из Пдф**

****

****

В этом примере мы модифицируем прошлый пример. Путём добавления ещё одного контекста done, мы будем переключать на другой, а не завершать всю программу сразу, как в прошлом примере.

