Отчет по лабораторной работе №8

«Анализ производительности алгоритмов с использованием утилиты GPROF»

Выполнил: студент группы РИС-22-1б

Поважный Виталий Евгеньевич

Задание:

1. Написать Си программу, вычисляющую числа Фибоначчи двумя способами: итеративным и рекурсивным.
2. Скомпилировать программу с флагом профилирования (-pg).
3. Запустить программу с n = 40.
4. Используя gprof определить, сколько времени функции тратят на вычисления. Сравнить результаты для обеих программ.
5. На основе полученных данных сделать вывод об эффективности/неэффективности каждого из методов.

На листинге 1 представлен исходный код программы с двумя функциями: FibRecursive – рекурсивный, FibIterative – итеративный

|  |
| --- |
| #include <stdio.h>  #include <time.h>  #include <stdlib.h>  long long FibRecursive(int n)  {  if (n <= 1)  return n;    return FibRecursive(n - 1) + FibRecursive(n - 2);  }  long long FibIterative(int n)  {  if (n <= 1)  return n;    long long prevFirst = 0;  long long prevSecond = 1;  long long result = 0;    for (int i = 2; i <= n; i++)  {  result = prevFirst + prevSecond;  prevFirst = prevSecond;  prevSecond = result;  }  return result;  }  int main(int\* argc, char \*argv[])  {  int n = atoi(argv[1]);    clock\_t start = clock();  long long resRecursive = FibRecursive(n);  clock\_t end = clock();  double timeRecursive = (double)(end-start)/CLOCKS\_PER\_SEC;  printf("Recursive: %d = %11d, Time: %f sec\n", n, resRecursive, timeRecursive);    start = clock();  long long resIterative = FibIterative(n);  end = clock();  double timeIterative = (double)(end-start)/CLOCKS\_PER\_SEC;  printf("Iterative: %d = %11d, Time: %f sec\n", n, resIterative, timeIterative);    return 0;  } |

Листинг 1 – Исходный код программы.

Для получения данных профилирования понадобится скомпилировать программу с флагом *-pg*, выполнить их, после чего выполнить следующую команду, преобразующую файл, сгенерированный в результате выполнения программ, в читабельный для человека текстовый файл:

*$ gprof {программа} gmon.out > {выходной\_файл}.txt*

Где *gmon.out* – файл, сгенерированный в результате выполнения программы.

На рисунке 1 представлен результат работы программы.

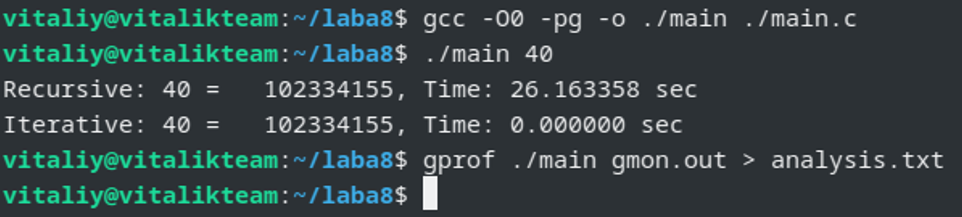


Рисунок 1 – Запуск программы.

На рисунке 2 представлен результат профилирования.

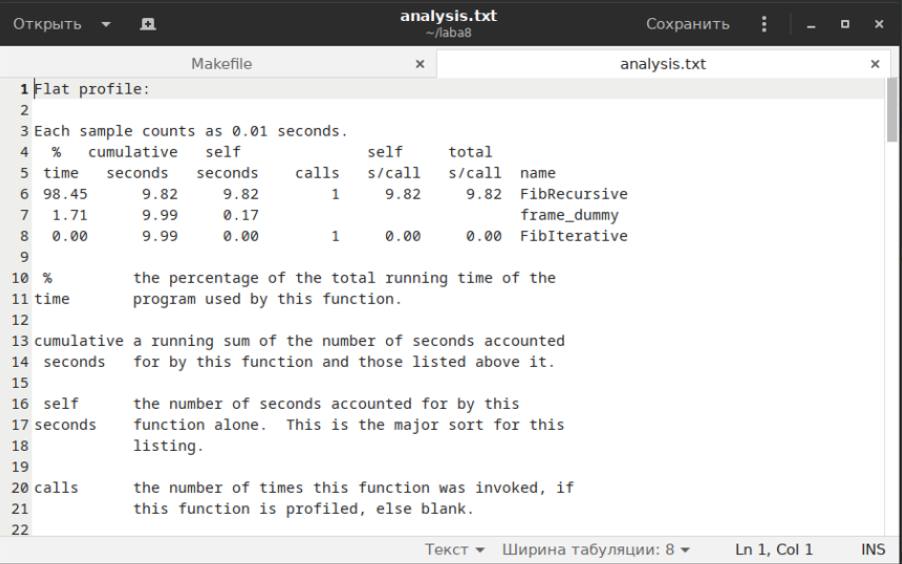


Рисунок 2 – Результат профилирования программы.

Итеративный подход показал значительно меньшее время выполнения за счёт отсутствия избыточных вызовов функций. Рекурсивный подход привел к экспоненциальному росту количества вызовов из-за многократного пересчета одних и тех же значений.

gprof использует **дискретизацию**, чтобы измерять производительность. Это значит, что инструмент периодически фиксирует текущую функцию, выполняемую в программе, например, каждые несколько миллисекунд. Однако:

* **Короткие вызовы между отсчетами пропускаются**. При экспоненциальной рекурсии функции вызываются так часто, что многие из них завершаются между моментами дискретизации, особенно если вызовы функций очень быстрые и короткие. gprof фиксирует только активную функцию в момент отсчета, что ведет к неполной картине времени выполнения для рекурсивных функций.
* **Экспоненциальное количество вызовов** означает, что суммарное реальное время, затраченное на рекурсию, оказывается намного больше, чем зафиксированное gprof. gprof не успевает учесть все вызовы из-за ограниченной частоты дискретизации и пропускает множество коротких вызовов.

**Вывод:**

Итеративный подход более эффективен для вычисления чисел Фибоначчи, поскольку снижает количество вычислительных операций и избегает избыточных вызовов функций