Измерение времени процесса в *Linux* с использованием *taskset* и *time*

Симоненко Е.А.
easimonenko@mail.ru

13 ноября 2018 г.

Аннотация

Описывается использование утилиты *taskset* в сочетании с утилитой *time* в операционной системе *Linux* для измерения времени работы процесса. Также рассматривается устройство данных утилит.

Введение

Большинство современных компьютеров являются многопроцессорными системами. Однопоточные программы операционная система запускает на одном из процессоров, который уже может быть занят выполнением одной или более других программ. Если перед нами стоит задача измерения времени выполнения определённой программы, то такая ситуация может приводить к получению недостоверных данных о времени работы процесса, превышающие истинную продолжительность.

В Linux для вычисления времени работы запускаемого процесса обычно привлекается стандартная утилита *time*. В сочетании с утилитой *taskset* можно попытаться получить более достоверные данные о времени выполнения нашего процесса.

В качестве примера процесса, время которого будет измерятся, привлечена программа вычисления чисел Фибоначчи рекурсивным методом на языке программирования C++. Ниже приведён листинг этой программы.

Листинг 1: "Программа вычисления чисел Фибоначчи"

```
#include <cstdio>
#include <cstdlib>

long long fibonacci(int n) {
  if (n == 1 || n == 2) {
    return 1;
  } else {
    return fibonacci(n - 2) + fibonacci(n - 1);
  }
}

int main(int argc, char* argv[]) {
  if (argc != 2) {
    printf("Need one argument: the number of Fibonacci numbers.");
    exit(-1);
  }
```

```
    printf("%lld\n", fibonacci(atoi(argv[1])));

    return 0;
}
```

Тестовая система представляется из себя ноутбук с процессором Intel Core i3 с частотой 2.13 ГГц, оперативной памятью объёмом 8 Гб и жёстким диском типа SSD.

1 Использование *taskset* и *time*

Время выполнения кода из примера 1 для нахождения числа Фибоначчи номер 45 составило около 11 сек. (1).

```
1134903170

real 0m11,645s
user 0m11,640s
sys 0m0,001s
```

Рис. 1: Результат с использованием *time*

Для подсчёта времени выполнения процесса используем команду:

```
time ./fibonacci 45
```

При запуске процесса на фиксированном процессоре были получены следующие результаты:

```
1134903170
11.56user 0.00system 0:11.57elapsed 99%CPU

1134903170
11.61user 0.00system 0:11.62elapsed 99%CPU

1134903170
11.80user 0.00system 0:11.84elapsed 99%CPU

1134903170
11.61user 0.00system 0:11.62elapsed 99%CPU
```

Рис. 2: Результат с использованием `taskset`

Для подсчёта времени выполнения процесса использовались команды вида:

```
taskset --cpu-list 0 time ./fibonacci 45
```

Где число 0 в опции `-cpu-time` это номер процессора, на котором будет выполняться процесс.

Из результатов видно, что даже на компьютере с запущенными пользовательским интерфейсом и несколькими приложениями, использование команды taskset ничего не даёт, время выполнения процесса остаётся примерно тем же самым. Что лишь подтверждает указание в справочной странице этой команды на то, что ядро Linux старается выполнять каждый процесс строго на определённом CPU, без перемещения его на другой.

2 Как работает команда time

Утилита time входит в состав пакета time ОС Linux. Главный модуль утилиты time находится в файле src/time.c пакета исходных текстов. Основная работа производится в процедурах run_command() и summarize().

Процедура run_command() получает на вход содержимое командной строки, переданной в функцию main(), и запускает дочерний процесс с помощью fork() и execvp(). Перед запуском дочернего процесса вызывается процедура resuse_start(), определённая в файле src/resuse.c. Эта процедура выясняет текущее время и сохраняет его в поле start структуры resp, описанной в RESUSE. Время может быть получено одним из двух способов: если в системе присутствует процедура gettimeofday(), объявленная в sys/time.h, то используется она, в противном случае выясняется частота процессора и используется системная процедура times(), определённая в sys/times.h.

По окончании работы дочернего процесса вызывается процедура resuse_end(), определённая в файле src/resuse.c, которая вычисляет время, затраченное на выполнение дочернего процесса.

3 Как работает команда taskset

Утилита taskset входит в состав пакета util-linux ОС Linux. Главный модуль этой утилиты находится в файле schedutils/taskset.c пакета исходных текстов. Основная работа производится в процедуре do_taskset().

Процедура sched_getaffinity() возвращает текущие настройки назначения процессора для процесса с заданным идентификатором. Если идентификатор равен нулю, то в качестве процесса рассматривается текущий поток.

Процедура sched_setaffinity() устанавливает новые настройки назначения процессора для процесса с заданным идентификатором. Если идентификатор равен нулю, то в качестве процесса рассматривается текущий поток.

Эти процедуры определены в исходных текстах ядра Linux, написаны на ассемблере и доступны через заголовок sched.h.

На вход эти процедуры получают структуру cpu_set_t, описывающую выбор процессоров для работы процесса. Для манипулирования этими настройками есть набор соответствующих процедур, описанных в CPU_SET(3).

Максимальное число доступных процессоров описывается константой CPU_SETSIZE и на данный момент, как следует из документации, равно 1024.

Ссылки

- GNU time
- util-linux
- Linux kernel