

# Анализ динамической планируемости наборов задач

*Автор курса: Балашов Василий Викторович*

*E-mail: hbd@cs.msu.ru*

Вариант 2

## 1 Контекст

В однопроцессорной вычислительной системе реального времени (ВС РВ) используется планирование выполнения задач по схеме EDF. Выполнение задач происходит с вытеснением. Для оценки планируемости набора задач используются формулы, описанные в лекциях [1]. Требуется реализовать программу, оценивающую планируемость по этим формулам.

## 2 Входные данные

1. Дана информация о наборе задач для выполнения на ВС РВ с динамическим планированием. Про каждую задачу известно имя (текстовая строка) и следующие числовые параметры (натуральные числа):

- период;
- относительный директивный срок, не превосходящий период;
- длительность выполнения.

Информация о наборе задач содержится в XML-файле, структура которого описана ниже.

2. ВС РВ использует схему динамического планирования EDF с вытеснением.

Примечание: для указанной схемы планирования, при наличии задач с директивными сроками, меньшими чем периоды, для планируемости набора задач *не достаточно* того, что загрузка процессора не превышает 1.

## 3 Требования

Необходимо написать программу, которая:

1. Считывает входные данные из xml-файла;

2. Проверяет описанные в [1] условия планируемости и выдаёт диагноз по планируемости (ответ «YES», если гарантируется выполнение всех задач с соблюдением директивных сроков, или «NO» — в противном случае);
3. В случае отрицательного ответа выводит левую и правую границы участка, на котором потребность в процессорном времени выше, чем 100%; достаточно обнаружить и вывести один такой участок.

### 3.1 Требования к программной реализации и оформлению ответа

К заданию прилагается несколько наборов входных данных, оформленных в виде XML-файлов. Для возможности самопроверки эти файлы сгруппированы в папки YES и NO в соответствии с правильными ответами на поставленную задачу.

В качестве языка программирования необходимо использовать C или C++. Полученная реализация должна удовлетворять следующим требованиям:

1. Код программы должен быть чистым и аккуратным. Он должен содержать комментарии в количестве, необходимом для его понимания. Подробнее про стиль кодирования можно прочитать, например, в [2], [3].
2. Архив с решением должен содержать Makefile, необходимый для сборки и запуска программы. В списке целей должны присутствовать цели *all* и *clean* (подробнее про написание и структуру Makefile'ов можно почитать, например, в [4]). Также допустимо использовать систему сборки *stake* [7] (в этом случае архив с решением должен содержать файл *CMakeLists.txt*, на основе которого с помощью утилиты *stake* генерируется Makefile описанного выше вида).
3. Архив с решением должен содержать всё необходимое для сборки и запуска программы в linux-based операционной системе (в частности, проверяться задание будет на компьютере под управлением Debian GNU/Linux 9.2 (stretch)). Если же у вас в распоряжении имеется лишь компьютер на Windows, для проверки работоспособности решения на linux'е имеет смысл воспользоваться эмулятором [5] и/или виртуальной машиной.
4. Все исходные файлы (за исключением Makefile'а) должны находиться в папке *src*, включённой в архив с решением.

5. Получаемый исполняемый файл должен быть назван по шаблону: `prog_<studnum>_<groupnumber>`, где `studnum` — номер студенческого билета. Например, `prog_02100242_621`.
6. Имя входного XML-файла должно передаваться через аргументы командной строки. Результат работы программы должен быть выведен на стандартный поток вывода. Сначала должен быть напечатан ответ на задачу (либо YES, либо NO), а на следующей строке в случае отрицательного ответа должны присутствовать разделённые пробелом левая и правая границы участка, на котором потребность в процессорном времени выше, чем 100%. Пример запуска приложения:

`prog_02100242_621 input.xml`

7. Архив с решением должен содержать текстовый файл `readme`, содержащий:
- (a) ФИО сдающего задание;
  - (b) номер группы;
  - (c) список использованных библиотек, если функционала, предоставляемого стандартной, было недостаточно;
  - (d) любую другую информацию на выбор сдающего, которая могла бы упростить и/или ускорить процесс приёма его задания.
8. Архив с решением должен иметь формат `zip` и имя `<ФамилияИО латиницей>.zip` (например, `IvanovAB.zip`). Глубина вложенности — один уровень, т.е. в самом архиве уже должны лежать все файлы, а не отдельная папка с файлами.

### 3.2 Формат входного XML-файла

XML-файл с описанием набора задач имеет следующую структуру:

- `<system>` — корневой элемент в описании ВС:
  - содержит элементы `<task>`;
- `<task>` — описание задачи, имеет атрибуты:
  - `name` (строка) — имя задачи;
  - `period` (целое число) — период задачи;

- deadline (целое число) — относительный директивный срок;
- duration (целое число) — длительность выполнения задачи.

### 3.2.1 Пример входного файла

```
<system>
  <task name="task1" period="100" deadline="30" duration="15"/>
  <task name="task2" period="195" deadline="20" duration="20"/>
  <task name="task3" period="250" deadline="10" duration="30"/>
</system>
```

## 4 Процесс сдачи задания

- Задание должно быть прислано на электронную почту [hbd@cs.msu.ru](mailto:hbd@cs.msu.ru) (тема письма должна быть по шаблону: «[ICS][Task1] ФамилияИО». ФамилияИО писать по-русски) не позднее 23:59:59 29 ноября 2021 года (мягкий дедлайн). Если задание будет прислано позднее 00:00 30 ноября 2021 года, но до 23:59:59 6 декабря 2021 года (жёсткий дедлайн), то получаемая за него оценка умножается на коэффициент 0.7.
- Задания, присланные позднее 00:00 7 декабря 2021 года, проверяться не будут.
- Задания, требования по оформлению которых были нарушены, также проверяться не будут (информация о нарушении придёт в ответном письме).
- В случае выявления значительных общих фрагментов программного кода в двух или более сданных реализациях проверяющие имеют право аннулировать все реализации, содержащие общий фрагмент. Критерий значительности общего фрагмента — на усмотрение проверяющих.

## Литература

- [1] В. В. Балашов *Информационно-управляющие системы реального времени* (слайды лекций).
- [2] 90 рекомендаций по стилю написания программ на C++ [HTML]  
[\(http://habrahabr.ru/post/172091/\)](http://habrahabr.ru/post/172091/)

- [3] *GoogleC++ Style Guide* [HTML] (<https://googlestyleguide.googlecode.com/svn/trunk/cppguide.html>)
- [4] *Makefile для самых маленьких* [HTML] (<http://habrahabr.ru/post/155201/>)
- [5] *Cygwin* [HTML] ([www.cygwin.com](http://www.cygwin.com))
- [6] *Google* [HTML] ([www.google.com](http://www.google.com))
- [7] *Cmake* [HTML] (<https://cmake.org/>)