Построение статического расписания передачи данных

Практическое задание по курсу
«Архитектура управляющих систем реального времени»
Преподаватель: Балашов Василий Викторович
E-mail: hbd@cs.msu.su
Вариант 1

1 Контекст

В вычислительной системе реального времени используется канал передачи данных с централизованным управлением. Каждая передача сообщения по каналу инициируется контроллером канала. Вытеснение сообщений недопустимо, передача следующего сообщения может начаться только после завершения передачи предыдущего сообщения. Контроллер работает в соответствии со статическим расписанием передачи сообщений.

Каждая передача сообщения называется *работой*. Работе сопоставляются *директивные сроки*:

- директивный срок начала (раньше него недопустимо начинать выполнение работы);
- директивный срок завершения (позже него недопустимо завершать выполнение работы).

Таким образом, каждая работа должна полностью быть выполнена в рамках своего *директивного интервала*, задаваемого директивными сроками начала и завершения.

Длительность выполнения каждой работы считается известной и фиксированной.

Статическое расписание представляет собой набор *цепочек работ*. Работы в цепочке выполняются одна за другой, без пауз. Для каждой цепочки в расписании задано время её начала и последовательность выполняемых работ.

Нагрузка на канал в формальном виде представляет собой набор периодических *заданий*, для каждого из которых определены:

- F: частота (величина, обратная периоду);
- c: длительность выполнения задания;
- φ_1: начальный фазовый сдвиг;
- φ_2: конечный фазовый сдвиг.

Содержательно, периодическому заданию соответствует периодически передаваемое сообщение.

Расписание строится для *интервала планирования* – конечного интервала времени от момента 0 до наименьшего общего кратного периодов всех заданий.

Каждому периодическому заданию соответствует набор работ. Директивный интервал k-й работы задания равен: $[k*T + \varphi_1; k*T + \varphi_2]$, где T – период задания ($T = 1/F$). При построении расписания рассматриваются только работы, директивные интервалы которых входят в интервал планирования. Длительность каждой работы задания равна длительности задания.

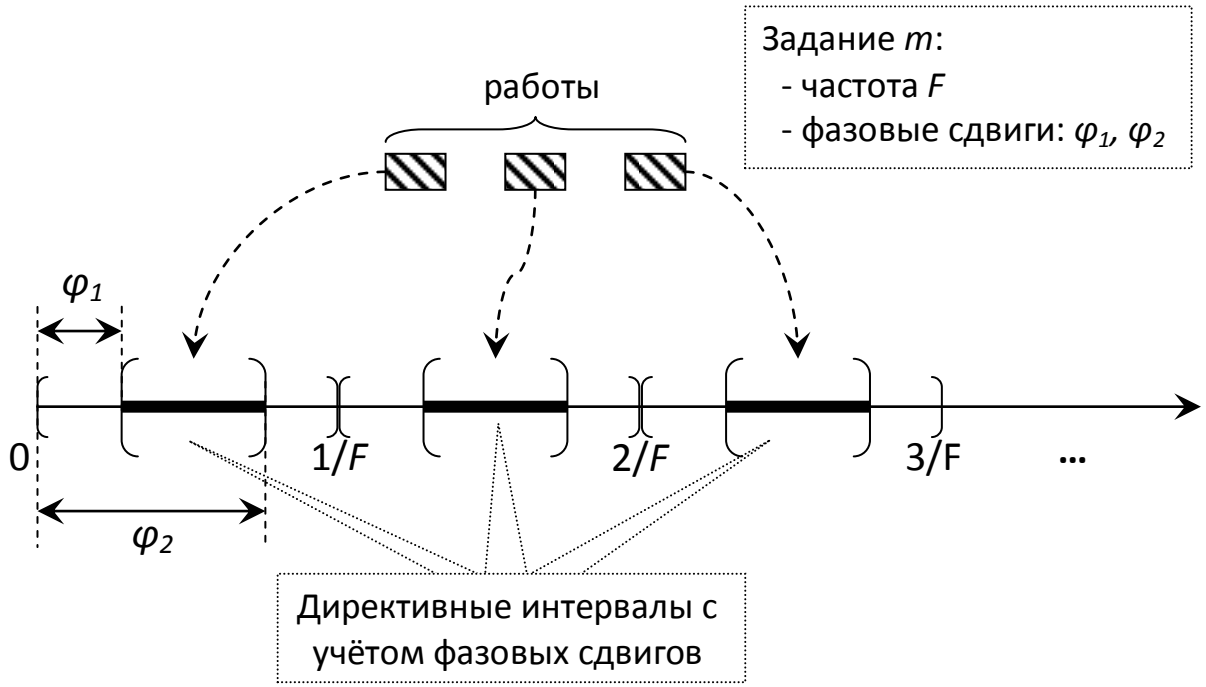

Рисунок 1. Набор работ для периодического задания

Помимо соблюдения директивных сроков работ, расписание должно удовлетворять ряду *технологических ограничений*, связанных со спецификой работы контроллера канала.

Введём два набора технологических ограничений, соответствующих двум видам статических расписаний.

Расписание с подциклами:

- момент начала каждой цепочки должен быть кратен заданному значению l_{sc} (длительность подцикла);
- число работ в цепочке не должно превышать заданного значения r_{mcc} (максимальное число работ в цепочке);
- длительность цепочки не должна превышать $l_{sc} * (1 - r_{rf})$, где $r_{rf} \in [0;1)$ – зарезервированная доля длительности подцикла.

Расписание без подциклов:

- число работ в цепочке не должно превышать заданного значения r_{mcc} (максимальное число работ в цепочке);
- интервал времени от завершения цепочки до начала следующей цепочки не должен быть меньше заданного значения r_{btw} (минимальный интервал между цепочками).
- длительность цепочки не должна превышать r_{mct} (максимальная длительность цепочки).

В конкретном варианте практического задания будет указано, с каким из двух видов расписаний необходимо работать.

На рисунках 2 и 3 показаны фрагменты расписаний с подциклами и без подциклов. Каждый прямоугольник соответствует интервалу от старта соответствующей работы до её завершения.

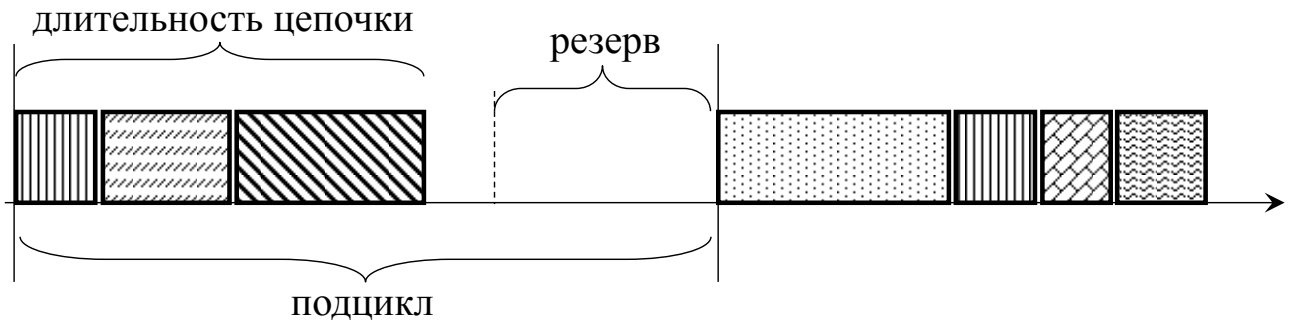

Рисунок 2. Расписание с подциклами

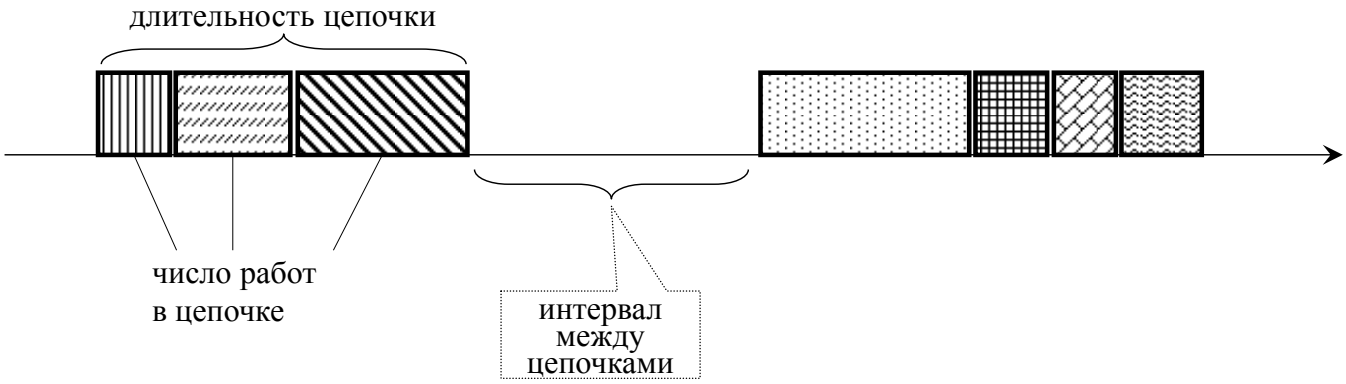

Рисунок 3. Расписание без подциклов

2 Алгоритм построения расписания

Для построения статического расписания используется описанный ниже алгоритм. Он основан на последовательном сдвиге вправо по оси времени так называемой «точки планирования» $t_{пл}$, и размещении работ на оси времени по мере сдвига точки планирования.

Алгоритм получает на вход набор работ с длительностями и директивными интервалами, а также значения технологических ограничений.

Схема выполнения алгоритма следующая:

1) $t_{пл} := 0$;
2) создать новую (первоначально пустую) цепочку работ;

// при определении, является ли $t_{пл}$ допустимым временем начала цепочки работ,
// учитываются ограничения на интервал между цепочками (в схеме без подциклов),
// ограничения на резерв времени в конце подцикла и на время начала подцикла (в схеме
// с подциклами)

3) если $t_{пл}$ не является допустимым временем начала цепочки работ, увеличить $t_{пл}$ до минимального (но не меньшего, чем $t_{пл}$) значения, являющегося допустимым временем начала цепочки работ;
4) Если $t_{пл}$ превысило правую границу интервала планирования, то **СТОП**.

// обновление директивных интервалов

5) Для каждой работы, у которой директивный срок начала меньше, чем $t_{пл,}$ установить директивный срок начала, равный max ($t_{пл,}$ *директивный_срок_начала_работы*);
6) Исключить из дальнейшего рассмотрения все работы, директивные интервалы которых короче, чем сами работы;
7) Если не осталось ни одной работы, то **СТОП.**
8) Если $t_{пл}$ не попадает в директивный интервал ни одной работы, то:
   а) выбрать работу с самым ранним директивным сроком начала, и установить $t_{пл}$ равным её директивному сроку начала (при этом $t_{пл}$ может только увеличиваться);

// учитываем, что в цепочке работ не может быть разрывов

   б) если текущая цепочка работ не пуста, завершить ее формирование и перейти к шагу 2, иначе перейти к шагу 3;

// точка планирования попадает в директивный интервал => добавляем работу

9) Если есть работы, в директивные интервалы которых попадает $t_{пл}$, и которые можно добавить в конец текущей цепочки работ без превышения ограничения на длительность цепочки, то:
   а) Выбрать из этих работ одну в соответствии с определённым **критерием**;
   б) Добавить выбранную работу в конец текущей цепочки работ (если текущая цепочка пуста, то считаем, что её «конец» соответствует времени $t_{пл}$);
   в) Установить $t_{пл}$ на время завершения добавленной работы:
   $t_{пл} := t_{пл} +$ *длительность_выбранной_работы*;
   г) Исключить выбранную работу из дальнейшего рассмотрения;
   д) Если в текущей цепочке достигнуто максимальное число работ, завершить её формирование и перейти к шагу 2;
10) Перейти к шагу 4.

На шаге 9.а обычно используется один из следующих критериев выбора работы:

а) выбрать работу с наименьшим директивным сроком завершения – Earliest Deadline First, EDF;

б) выбрать работу с наименьшим «запасом» в директивном интервале, т.е. разностью между длительностью директивного интервала и длительностью работы – Least Slack First, LSF;

в) выбрать работу с наименьшим временем завершения при условии старта в момент $t_{пл}$, т.е. с наименьшим значением ($t_{пл}$+*длительность_работы*) – Earliest Completion First, ECF.

При выдаче задания будет указано, какой из критериев необходимо использовать.

3 Входные данные

Дана информация о наборе периодических заданий для канала с централизованным управлением. Для каждого задания указаны следующие числовые параметры (натуральные числа):

- числовой идентификатор (номер);
- число слов данных;
- частота, в Гц;
- начальный фазовый сдвиг, в **милли**секундах (мс; 1 мс = 1000 мкс);
- конечный фазовый сдвиг, в мс (если задано значение 0, считать, что правый фазовый сдвиг равен периоду задания).

Число слов данных (N_w) пересчитывается в длительность задания *(t)* по формуле:

$$t = N_w * 20 \text{ (мкс)}$$

Информация о наборе периодических заданий предоставляется в виде текстового файла со строками вида:

<идент. задания> <число слов> <частота> <начальный фаз.сдвиг> <конечный фаз.сдвиг >

Например:

157 28 1 440 460

Числа в строке разделяются пробелами и символами табуляции в произвольном сочетании.

Файл также может содержать строки-комментарии, начинающиеся с «#». Эти строки следует пропускать при разборе файла.

Имя файла передаётся через параметр командной строки.

Для построения расписания с подциклами также задана длительность подцикла l_{sc}, в мс. Она передаётся через параметр командной строки.

4 Содержание задания

Необходимо написать программу, которая:

1) Получает входные данные из параметров командной строки и из входного файла;
2) Выполняет следующие расчёты для полученного набора входных данных:
   а) Для расписания с подциклами:
      i) Поиск максимального значения r_{rf}, при котором возможно успешное построение расписания описанным выше алгоритмом (хотя бы при каких-то значениях r_{mcc});
      ii) Для найденного значения r_{rf}: поиск минимального значения r_{mcc}, при котором расписание успешно строится;
   б) Для расписания без подциклов:
      i) Поиск максимального значения r_{btw}, при котором возможно успешное построение расписания описанным выше алгоритмом (хотя бы при каких-то значениях r_{mcc} и r_{mct});
      ii) Для найденного значения r_{btw}: поиск минимального значения r_{mcc}, при котором расписание успешно строится, хотя бы при каких-то значениях r_{mct};
3) Выводит в консоль:
   а) найденные значения параметров (технологических ограничений) в формате «<имя параметра> = <значение>», например: «r_mcc = 35»;
   б) расписание (построенное при найденных значениях параметров) в виде набора цепочек работ; каждая цепочка выводится на отдельной строке, в формате «<время старта цепочки> <идент.задания 1> <идент.задания 2> …», где время старта цепочки совпадает с временем начала выполнения первой работы из состава цепочки, а идентификаторы заданий определяют задания, которым соответствуют работы в составе цепочки.

Под успешным построением расписания понимается нахождение такого расписания, в которое входят все работы всех заданий, с соблюдением директивных сроков и технологических ограничений.

Значение параметра r_{btw} достаточно определить с точностью до 1 мс (1000 мкс); значение параметра r_{rf} достаточно определить с точностью до 0.01.

Комплекты входных данных прилагаются к заданию.

5 Требования к программной реализации и оформлению результата

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 [1], [2].
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 [3]).
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 8.0 «Jessie»). Если же у вас в распоряжении имеется лишь компьютер с Windows, для проверки работоспособности решения на linux'е имеет смысл воспользоваться эмулятором [4] 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_02100243_521.
6. Имя входного txt-файла, а также длительность подцикла (для расписания с подциклами) должно передаваться через аргументы командной строки, например:
   prog_02100243_521 input.txt 20 – для расписания с подциклами;
   prog_02100243_521 input.txt – для расписания без подциклов;
7. Результат работы программы должен быть выведен в стандартный поток вывода (на консоль). Сначала должны быть напечатаны найденные значения параметров, а затем расписание.
8. Архив с решением должен содержать текстовый файл «readme.txt», содержащий:
   а) ФИО сдающего задание;
   б) номер группы;
   в) список использованных библиотек, если функционала, предоставляемого стандартной, было недостаточно;
   г) краткое описание реализованной схемы поиска значений параметров расписания (технологических ограничений).
9. Архив с решением должен иметь формат zip и имя ФамилияИО.zip (например, IvanovAP.zip). Глубина вложенности — один уровень (т.е. в самом архиве уже должны лежать все файлы, а не отдельная папка с файлами).

6 Процесс сдачи задания

- Задание должно быть прислано на электронную почту hbd@cs.msu.su (тема письма должна быть по шаблону: «[ICS][Task1] ФамилияИО»; ФамилияИО писать кириллицей) не позднее 23:59:59 6 декабря 2016 года (мягкий дедлайн). Если задание будет прислано позднее 00:00 6 декабря 2015 года, но до 23:59:59 11 декабря 2016 года (жёсткий дедлайн), то получаемая за него оценка умножается на коэффициент 0.7.
- Задания, присланные позднее 00:00 10 11 декабря 2016, 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 преподаватель имеет право аннулировать все реализации, содержащие общий фрагмент. Критерий значительности общего фрагмента — на усмотрение преподавателя.

Литература

[1] *90 рекомендаций по стилю написания программ на C++* [HTML] (http://habrahabr.ru/post/172091/)

[2] *Google C++ Style Guide* [HTML] (https://googlestyleguide.googlecode.com/svn/trunk/cppguide.html)

[3] *Makefile для самых маленьких* [HTML] (http://habrahabr.ru/post/155201/)

[4] *Cygwin* [HTML] (www.cygwin.com)

[5] *Google* [HTML] (www.google.com)