УДК 681.3

DOI: http://dx.doi.org/10.20535/2219-3804172017123927

Зинченко С. В. [1], *научный сотрудник*, **Зинченко В. П.** [2], *доцент, к.т.н.*

ПЛАНИРОВАНЕ ЗАДАЧ В СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

En As is known, the main task of the RTS is to react in time to events at the site. Therefore, the establishment of some sequence of tasks or mutual exclusion is an urgent problem for the RTS.

The RTS of a rigid RT must ensure that each critical task is completed for all possible scenarios for its operation. Such guarantees can be provided on the basis of: exhaustive testing of all possible scenarios of the behavior of the managed object and control application programs (AP); constructing a static schedule; choosing a mathematically based dynamic planning algorithm.

It is difficult to plan the work of dependent tasks, and its optimal solution requires large computational resources. This problem is solved by dividing the planning of tasks into two parts so that one part is performed in advance, before the launch of the RTS, and the second, a more simple part, during the operation of the RTS.

When planning, such time parameters of tasks are taken into account: a). Time points: release time - when there was a need to transfer control to the task (the time of the event); absolute deadline - to which the task must complete the next work; start time - when the task was started on the microprocessor (MP); Completion time - when the task has finished working by processing the event; b). Time parameters: relative deadline; execution time - the time when the task is executed while performing the next work; response time.

A system of independent periodic problems is considered, where there may be still aperiodic and sporadic problems as some special cases.

It is shown that the static schedule for periodic tasks is a table in which it is indicated at which point in time which task to run for execution. Such a schedule is compiled on the interval of the hyperperiod. The scheduler periodically repeats the task execution sequence specified in the schedule. Schedulers usually work on interrupts from the timer, and tasks are AP, which are called at the appropriate time by the interrupt handler. Such algorithms are used in high-reliability RTS.

Two types of schedulers with dynamic priorities are considered: EDF (earliest dead-line first); LLF (least laxity first). Theorems / algorithms for their optimality are given. The notion of time reserve is explained. An example of a correct timetable is given. It is shown that in the analysis of planning algorithms, it is necessary to take into account the overhead costs for planning, by including them during the execution of tasks.

Dynamic planning with static priorities is considered: RMS (rate monotonic scheduling); DMS (deadline monotonic scheduling). An example of a schedule composed by RMS and DMS planners is given. For the considered rules for assigning priorities, the statement / theorems of their optimality are given. Also, for

[1] *Институт кибернетики имени В. М. Глушкова НАН Украины*
[2] *НТУУ «Киевский политехнический институт им. Игоря Сикорского», кафедра автоматизации экспериментальных исследований*

systems with arbitrary relations between task periods, a sufficient, but not necessary, test / criterion for testing the feasibility of the RMS algorithm is given.

Conclusions.The features of task scheduling in real-time systems, the conditions of the guaranteed completion of task times and task parameters; algorithms for static and dynamic scheduling; criteria/ the theorem of existence of planning objectives; examples.

Ua Розглянуті особливості планування задач у системах реального часу, умови гарантованого завершення задач, часові моменти та параметри задач; алгоритми статистичного і динамичного планування; критерії/ теореми планування; приклади.

Введение

Как известно главной задачей систем реального времени (СРВ) является вовремя реагировать на события на объекте, которые регистрируются датчиками и передаются в модули ввода-вывода СРВ. Получив сигнал, СРВ должна запустить прикладную программу (ПП) обработки этого события. Поэтому ключевым параметром является время реакции СРВ на прерывание. Известно, что время реакции СРВ на события зависит от области применения и варируются от мкс до ч. Общепринятого метода определения этого параметра нет, но его можно предсказать/вычислить по значению его составляющих [1, 2].

Время выполнения действий не зависит от операционных систем реального времени (ОСРВ) и целиком определяется аппаратурой, а интервал времени от возникновения запроса на прерывание до выполнения первой инструкции обработчика – определяется целиком свойствами операционной системы (ОС) и архитектурой компьютера. Необходимо также учитывать оценки времени реакции ОСРВ на прерывание и обработки параллельных событий.

Постановка задачи

Цель работы является анализ особенностей выполнения задач и функций СРВ, планирование задач в СРВ; выполнение прикладных програм; временные характеристики и параметры задач; проблемы построения расписаний [3, 4].

Планирование задач

СРВ жесткого РВ должна гарантировать завершения выполнения каждой критической задачи для всех возможных сценариев ее работы. Такие гарантии можно обеспечить на основании: исчерпывающего тестирования всех возможных сценариев поведения управляемого объекта и управляющих ПП; построения статического расписания; выбора матема-

тически обоснованного динамического алгоритма планирования (АП). При этом, при выборе АП следует учитывать зависимость задач (имеются: критические секции; ограничения на порядок исполнения).

С практической точки зрения АП зависимых задач более важны, чем АП независимых задач. В случе использования простых микроконтроллеров нет смысла организовывать мультипрограммное выполнение большого количества независимых задач на одном компьютере в силу сложности ПП. Одновременно выполняемые зависимые задачи должны обмениваться информацией и получать доступ к общим данным для достижения общей цели системы. Поэтому установление некоторой последовательности выполнения задач или взаимного исключения является актуальной проблемой для СРВ [4, 5].

Планировать работу зависимых задач сложно, и ее оптимальное решение требует больших вычислительных ресурсов. Решается эта проблема путем разделения проблемы планирования на две части так, что одна часть выполнятся заранее, до запуска СРВ, а вторая, более простая, часть – во время работы СРВ. Например, анализ набора задач с взаимными исключениями может состоять в: выявлении запрещенных областей времени, в течение которых нельзя назначать выполнение задач, содержащих критические секции; введении ограничений на поведения набора задач. При таком подходе планирование приближается к статическому [4, 6].

Параметры задач

Работа задач характеризуется такими моментами времени (t): r (*release time*) – когда возникла необходимость в передаче управления задаче (момент события); d (absolute deadline/абсолютный крайний срок) – к которому задача должна завершить очередную работу; s (*start time*) – когда задача начала исполняться на микропроцессоре (МП); c (*completion time*) – когда задача закончила работу, обработав событие. И такими временными параметрами: D (*relative deadline*) – относительный крайний срок, $D = d - r$; e (*execution time*) – время исполнения задачи при выполнении очередной работы, $e = c - s$; R (*response time*) – время отклика, $R = c - r$.

На рис. 1 представлена диаграмма работы задачи с такими параметрами: $r = 2$, $d = 11$, $s = 5$, $\mathrm{c} = 9$, $D = 11 - 2 = 9$, $e = 9 - 5 = 4$, $R = 9 - 2 = 7$.

Параметры определяются следующим образом. Время перехода задачи в состояние готовности определяется природой управляемого объекта. Крайние сроки (d, D) определяет разработчик СРВ, исходя из свойств управляемого объекта. Время выполнения задач e определяются архитектурой МП, его тактовой частотой и конкретной реализацией алгоритма. Для этого используются два подхода: подсчет тактов МП, необходимых на

выполнение конкретной задачи; непосредственное измерения времени исполнения.

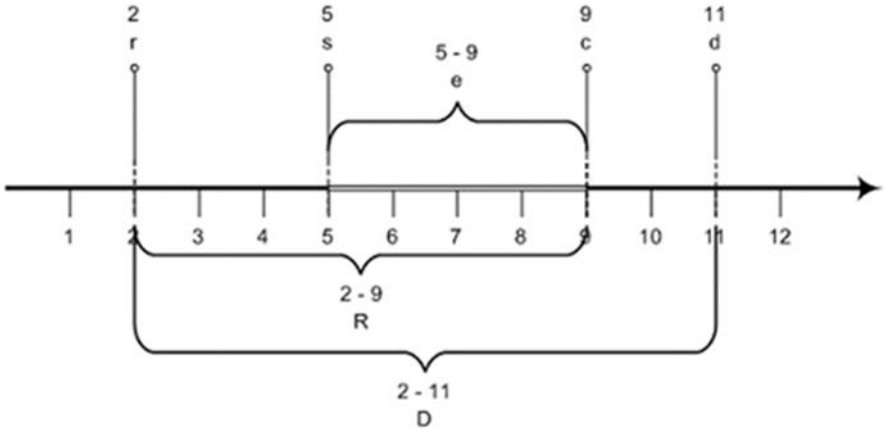

Рис. 1. Параметры задачи

Характер возникновения событий на управляемом объекте определяется типом задач: периодические; спорадические (непериодические с жестким крайним сроком); апериодические (непериодические с мягким крайним сроком).

Обозначим $\{T_j\}$ – набор задач; $r_{jk} - t$, когда возникает необходимость в выполнении j–й задачи; $T_j[\varphi_j, p_j, e_j, D_j]$ – периодическая задача, где r_{j1} является фазой задачи φ_j. Для периодической задачи возможны такие соотношения между p_j и D_j: $p_j = D_j$; $p_j > D_j$; $p_j < D_j$ (одновременно могут $\exists$: $\{\bar{T}_j\}$ несколько экземпляров T_j). Тогда, коэффициент использования времени МП периодической задачей будет таким $u_j = e_j/p_j$, а коэффициент использования МП времени системы периодических задач таким $U = \sum u_j = \sum e_j/p_j$ [7].

Статическое планирование

Расписание является корректным, если соблюдены все крайние сроки. Набор задач $\{T_j\}$ называется планируемым некоторым АП, если этот АП всегда дает корректное расписание для этого набора задач.

АП является оптимальным, если для $\forall\{T_j\}$ дает корректное расписание, если таковое вообще существует для данного набора задач.

Рассмотрим систему независимых периодических задач $\{T_j\}$, где могут иметься еще апериодические и спорадические задачи как некоторые особые случаи.

Расписание для периодических задач представляет собой таблицу, в которой указано, в какой момент времени какую задачу запускать на исполнение. Такое расписание составляется на промежутке гиперпериода

$P = lcm_j(\{p_j\})$. Таким образом, планировщик периодически повторяет последовательность запуска задач, заданную в расписании.

Например, для набора из четырех задач: $T_1[*,4,*,1]$ $T_2[*,5,*,1.8]$ $T_3[*,20,*,1]$ $T_4[*,20,*,2]$ $P = 20$. На его протяжении T_1 должна получить управление 5 раз, T_2 – 4, а T_3 и T_4 – по одному разу. При этом должны быть соблюдены все крайние сроки. Статическое расписание для этой системы задач будет таким (табл.), где *I* – «задача» простоя (*Idle task*).

***Таблица* 1.**

Пример расписания

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
время	0	1	2	3,8	4	5	6	8	9,8	10,8	12	13,8	14,8	16	17	18	19,8
относительное	0,2	1	1	1,8	0,2	1	1	2	1,8	1	1,2	1,8	1	1,2	1	1	1,8
задача	T1	T3	T2	I	T1	I	T4	T2	T1	I	T2	T1	I	T1	I	T2	I

Планировщики, обычно, работают по прерываниям от таймера, а задачи представляют собой ПП, которые вызываются в нужные t обработчиком прерываний. Так как таймер удобно программировать на генерацию прерывания не через какой-то промежуток времени от начала отсчета времени, а на прерывание через какой-то промежуток времени от текущего момента времени, то «время» в табл. 1 целесообразно представить в ином виде: вместо промежутков времени от начала цикла помещать промежутки относительного время от одного прерывания до другого, что и приведено в строке «относительное».

Преимущества таких алгоритмов: простота, вызванная отсутствием понятий «процесс»/«поток»; передача управления задаче – это вызов ПП; результаты тестирований и проверок весьма надежны. Благодаря этим качествам, такие алгоритмы применяются в СРВ высокой надежности.

Недостатки: негибкость (изменение числа задач, временных параметров и т.п., требует остановки СРВ, пересчета расписания и перекомпиляции ПП); планировщик «отвязан» от внешнего мира, так как работает по прерываниям от таймера; сложно учитывать спорадические задачи; возможный большой размер таблицы с расписанием.

Динамическое планирование с динамическими приоритетами

Рассмотрим два типа планировщиков с динамическими приоритетами:

- *EDF* (*earliest deadline first*) – наивысший приоритет имеет та задача, у которой осталось меньше всего времени до крайнего срока»; модификации – с/без вытеснением/я задач.
- *LLF* (*least laxity first*) – наивысший приоритет имеет задача с наименьшим запасом времени.

Известно, что (теорема) алгоритм *EDF* оптимален для $\forall\{T_j\}$ (периодических с любым соотношением между p_j и D_j, спорадических), если $all\ T_j$: независимы; возможно вытеснение; вытеснение не требует временных затрат.

Ее доказательство основывается на том, что если для какого-то набора задач {можно составить корректное расписание, то это расписание всегда можно привести к такому виду при котором оно останется корректным, что порядок исполнения задач будет таким, какой получился бы при использовании *EDF*. Для алгоритма *LLF* имеет место подобная теорема.

Под понятием резерв/запас времени (*laxity*) понимается разность между временем, оставшимся до крайнего срока, и временем, которое задаче еще нужно проработать, то есть $L(t) = (d - t) - e^{(rem)}$. На рис. 2 показана некая работа T_j, которая стала готовой в $t = 2$ и имеет крайний срок в $t = 12$. Задача отработала с вытеснениями. Пока T_j выполняет свою работу, запас времени остается постоянным, так как уменьшается и время до D_j, и количество времени, которое еще нужно проработать – разумеется, на одну и ту же величину. Если же T_j простаивает по причине вытеснения ее другими, более приоритетными задачами, то D_j приближается, а время, которое еще нужно проработать, остается постоянным, поэтому в такие промежутки запас времени, естественно, уменьшается.

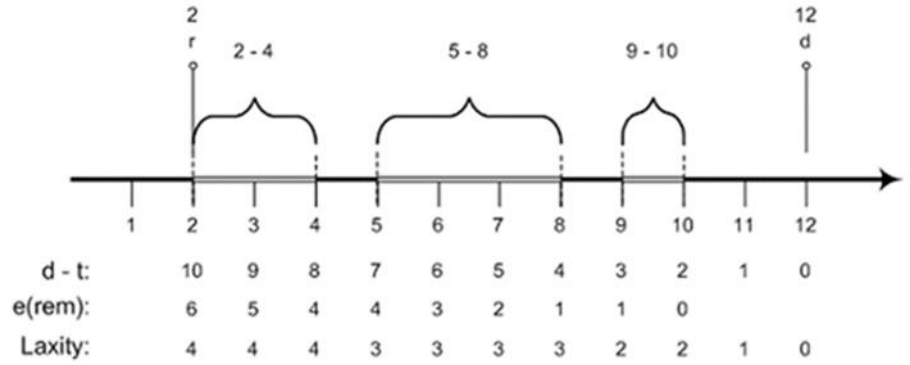

Рис. 2. Резерв времени

Алгоритм *EDF* без вытеснения не оптимален (теорема). Доказать эту теорему можно, приведя такой $\{T_j\}$ для которого можно составить корректное расписание, а АП *EDF* без вытеснения даёт некорректное расписание.

Пусть имеем три задачи со такими параметрами $T_j[r, e, d]$: T_1[0, 3, 10]; T_2[2, 6, 14]; T_3[4, 4, 12], и таким, например, корректным расписанием (рис. 3).

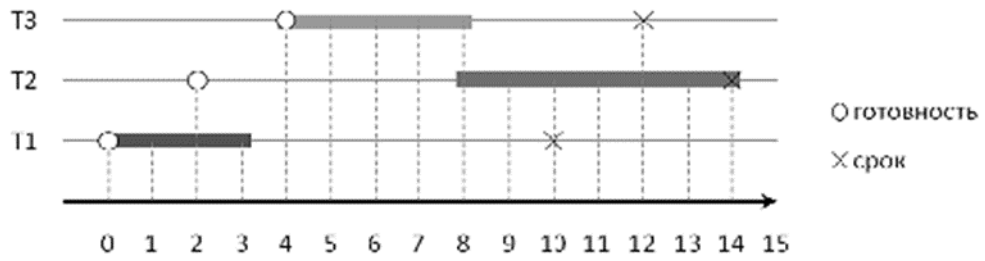

Рис. 3. Расписание для системы трех задач

Алгоритм *EDF* без вытеснения даст следующее расписание (отметим, что моменты принятия решения планировщиком – это t, когда-либо появляется задача в состоянии готовности, либо какая-то задача завершает свою очередную работу) (рис. 4).

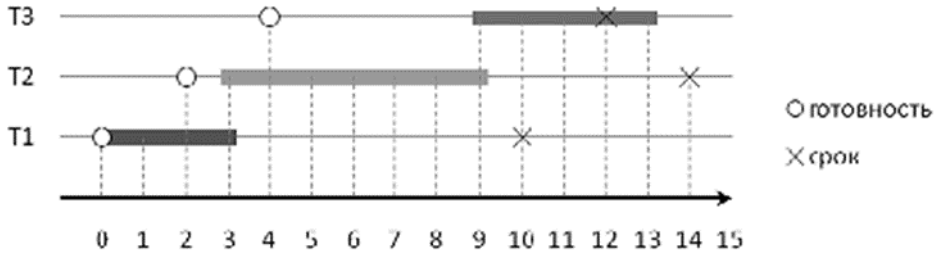

Рис. 4. Расписание *EDF*-планировщика без вытеснения

В $t = 3$ готова только T_2, поэтому она и выбирается для исполнения и не вытесняется до тех пор, пока полностью не сделает свою работу несмотря на то, что в $t = 4$ в состояние готовности переходит T_3, у которой крайний срок наступает раньше, чем у T_2. Как видим, из-за этого T_3 пропускает свой крайний срок. Таким образом, алгоритм EDF без вытеснения оптимальным нее является.

Для сравнения приведем расписание, которое составил бы EDF- планировщик с возможностью вытеснения задач (рис. 5).

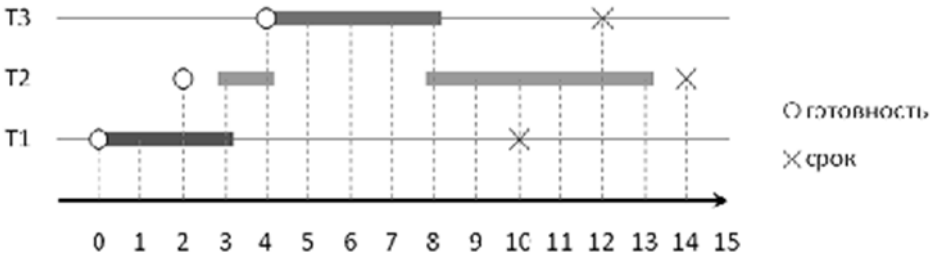

Рис. 5. Расписание *EDF*-планировщика с вытеснением

Как видно из рис. 5 $all\ T_j$ успели завершить свои работы до D_j. В $t = 2$ стала готовой T_2, но она не вытесняет T_1, так как у T_2 крайний срок наступает позже, чем у T_1. T_2 получает управление только в $t = 3$. Но появившаяся в $t = 4$ T_3 вытесняет T_2, так как у T_3 крайний срок наступает

раньше, чем у T_2. В $t = 8$, когда T_3 завершит свою работу, управление вновь получает T_2.

Из вышесказанного, однако, не следует делать вывод, что *EDF* без вытеснения безусловно хуже *EDF* с вытеснением, поскольку доказательство оптимальности последнего проведено в предположении, что собственно само вытеснение задач (например, переключение контекстов) не требует никакого времени, что, подчеркнем, никак с действительностью не согласуется. Поэтому при анализе АП нужно учитывать накладные расходы на планирование, например, включать их во времена исполнения задач e.

Для *EDF*-планировщика с вытеснением имеет место утверждение о проверке $\{T_j\}$ на предмет возможности составления расписания для нее таким планировщиком (теорема). Для того чтобы система $\{T_j\}$ независимых вытесняемых периодических задач (таких, что для $\forall_j : D_j \gg= p_j$), была планируема *EDF*-планировщиком, необходимо и достаточно, чтобы $U \leq 1$.

Если же относительные $D_j \leq p_j$, то данное условие является необходимым, но не достаточным. Например, набор из двух задач $T_j(p, e, D)$: $T_1(2;\ 1;\ 1{,}9)$; $T_2(2;\ 1;\ 1{,}9)$ заведомо не может быть корректно распланирована никаким алгоритмом, хотя для нее $U = 1$. В таких случаях вместо понятия «коэффициент использования процессорного времени» используется понятие «плотность δ задачи», определяемую как $\delta = e/min(D, p)$.

Суммарная плотность $\{T_j\}$ определяется как сумма плотностей $all\ T_j$ системы: $\Delta = \sum \delta_j = \sum e_j / min(p_j, D_j)$.

Для систем такого рода имеет место несколько иное утверждение относительно возможности составления корректного расписания (теорема). Для того чтобы $\{T_j\}$ независимых вытесняемых периодических задач (таких, что для $\forall j$: $D_j < p_j$), была планируема *EDF*-планировщиком, достаточно, чтобы $\Delta < 1$.

Отметим, что это условие не является необходимым. Это значит, что если оно выполняется, то для СРВ можно составить корректное расписание; если же это условие не выполняется, то корректное расписание составить нельзя. Например, для $T_1(2;\ 0{,}6;\ 1)$ и $T_2\ (5;\ 2{,}3)$ условие $\Delta \leq 1$ (Δ=0,6/1+2,3/5=1,06>1) не выполняется, но корректное расписание составить можно [8, 9].

Динамическое планирование со статическими приоритетами

Существуют два распространенных способа назначения приоритетов:

– *RMS* (*rate monotonic scheduling*) – чем меньше период задачи, тем выше у нее приоритет или чем чаще/*rate* задача переходит в состояние готовности, тем ее приоритет выше;

– *DMS* (*deadline monotonic scheduling*) – чем меньшее относительный крайний срок задачи, тем выше ее приоритет.

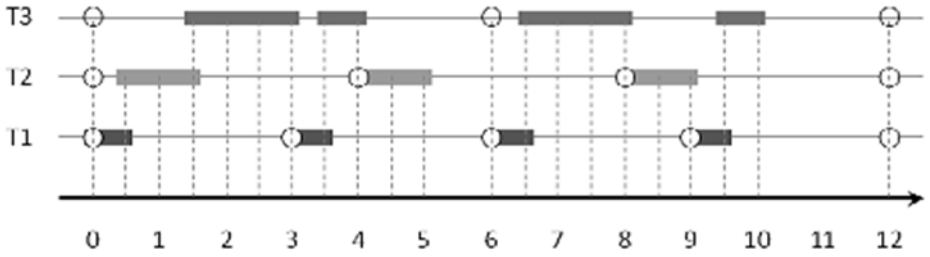

Рис. 6. Расписание *RMS*-планировщика

На рис. 6 приведен пример расписания, составленного *RMS*-планировщиком для синхронной системы 3-х периодических задач с такими параметрами: $T_1(3;\ 0{,}5)$; $T_2(4;\ 1)$; $T_2(6;\ 2)$.

В соответствии с вышеприведенным правилом самый высокий приоритет имеет T_1, а самый низкий – T_3. В $t = 0$ имеется 3 готовых задачи, МП получает T_1. В $t = 0{,}5$ готовы 2 задачи (T_2 и T_3), управление получает T_2. В $t = 3$ снова становится готовой T_1, поэтому T_3 ею вытесняется. В $t = 3{,}5$ управление вновь получает T_3 как единственно готовая. В $t = 4{,}0$ готова только T_2, она и получает управление.

В $t = 6$ готовы две задачи (T_1 и T_3), управление отдается T_1. Далее работает T_3, но в $t = 8$ она вытиснятся T_2. В $t = 9$ опять готова T_1, она отрабатывает и, наконец, в $t = 9{,}5$ получает управление T_3. Затем все повторяется сначала (при условии, что число задач в системе неизменное).

Рассмотрим пример расписания, составленного *DMS*-планировщиком (рис. 7) для следующей системы задач: $T_1(3;\ 0{,}5)$; $T_2(4;\ 1;\ 2)$; $T_3(6;\ 2)$. Отличие этой системы от предыдущей только в том, что у T_2 относительный крайний срок нее равен периоду, а < в 2 раза. Согласно правилу назначения приоритетов, самый высокий приоритет будет иметь T_2, а самый низкий – T_3.

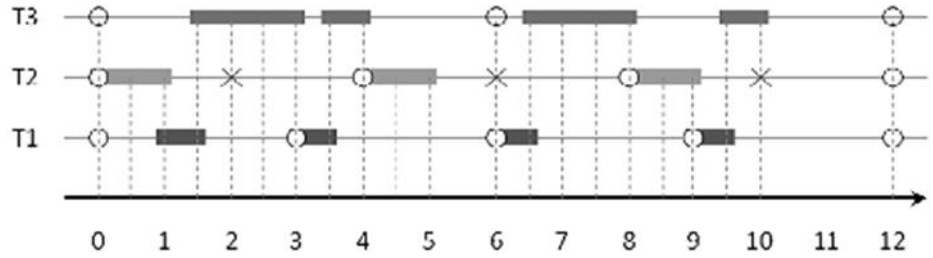

Рис. 7. Расписание *DMS*-планировщика

Для рассмотреных правил назначения приоритетов существует следующее утверждение (теорема): алгоритмы *DMSS* и *RMS* не являются оптимальными.

Система $\{T_j\}$ периодических задач называется СРВ с простой периодичностью, если для $\forall T_j$ и T_k при $p_i < p_k$ имеет место равенство $p_k \; mod \; p_i = 0$. Иными словами, периоды всех задач системы попарно делятся нацело друг на друга. Например, $\{T_1(2,1), T_2\,(4,1), T_3\,(8,11)\}$ является системой с простой периодичностью, а система $\{T_1(2,1), T_2(5,11), T_3(10,1)\}$ таковой не является ($5 \; mod \; 2 = 1 \neq 0$).

Известо, что для систем с простой периодичностью справедливо такое утверждение (теорема). Система $\{T_j\}$ независимых вытесняемых задач с простой периодичностью с периодами, не превышающими относительные крайние сроки, планируется на одном МП тогда и только тогда, когда для этой системы $U < 1$.

Для систем с произвольными соотношениями между периодами задач справедливо следующее утверждение (теорема/ критерий проверки планируемости алгоритмом RMS). Система $\{T_j\}$ n независимых вытесняемых периодических задач с $p_j = D_j$ планируется на одном МП, если для этой системы $U \leq n(\frac{21}{n} - 1)$. Это условие является достаточным, но не необходимым [10].

Выводы. Рассмотрены особенности планирования задач в СРВ, условия гарантированого завершения задач, временные моменты и параметры задач; алгоритмы статического и динамического планирования; критерии/ теоремы планируемости; примеры.

Список использованной литературы

1. *Зинченко С. В.* Особенности систем реального времени / С. В. Зинченко, В. П. Зинченко // Інформаційні системи, механіка та керування, 2017. - Вып. 16. – С. 11–23. DOI: https://doi.org/10.20535/2219-3804162017114105.
2. *Зинченко С. В.* Особенности систем реального времени / С. В. Зинченко, В. П. Зинченко // XI Междунар. конф. “Гиротехнологии, навигация, управление движением и конструирование авиационно-космической техники”. Сб. докл. Секция - Информационные технологии. – К.: “КПИ им. Игоря Сикорского”, 13-14 апреля 2017. – С. 48-57.
3. *Зинченко С. В.* К вопросу планирования задач в системах реального времени /С. В. Зинченко, В. П. Зинченко // XI Междунар. конф. “Гиротехнологии, навигация, управление движением и конструирование авиационно-космической техники”. Сб. докл. Секция - Информационные технологии. – К.: “КПИ им. Игоря Сикорского”, 13-14 апреля 2017. – С. 40-47.
4. *Зинченко С. В.* Программная поддержка и планирование задач в системах реального времени / С. В. Зинченко, В. П. Зинченко // Комп’ютерна математика, 2017. – Вып. 1. – С. 73–87.

5. Теория расписаний и вычислительные машины/ Под ред. Э. Г. Коффмана. - М.: Наука, 1984.
6. *Костенко В. А.* Алгоритм построения расписаний обменов по шине с централизованным управлением и исследование его эффективности / В. А. Костенко, Е. С. Гурьянов // Программирование, 2005., № 6.
7. Организация вычислительных процессов: конспект лекций/ сост. Б. М. Степанов. – Улан-Удэ: Изд-во ВСГТУ, 2001.
8. Теория и реализация вычислительных систем реального времени: сборник / Рос. АН ВЦ. – М.: ВЦ РАН, 1999.
9. *Balashov V. V.* A Tool System for Automatic Scheduling of Data Exchange in Real-Time Distributed Avionics Systems / V. V. Balashov, V.A. Kostenko, R.L. Smeliansky // Proceedings of the 2nd EUCASS European Conference for Aerospace Sciences (EUCASS'2007), 2007.
10. *Конвей Р. В.* Теория расписаний / Р. В. Конвей, В. Л. Максвелл, Л. В. Миллер // М.: Гл. ред. физ.-математ. лит-ры изд-ва «Наука», 1975.