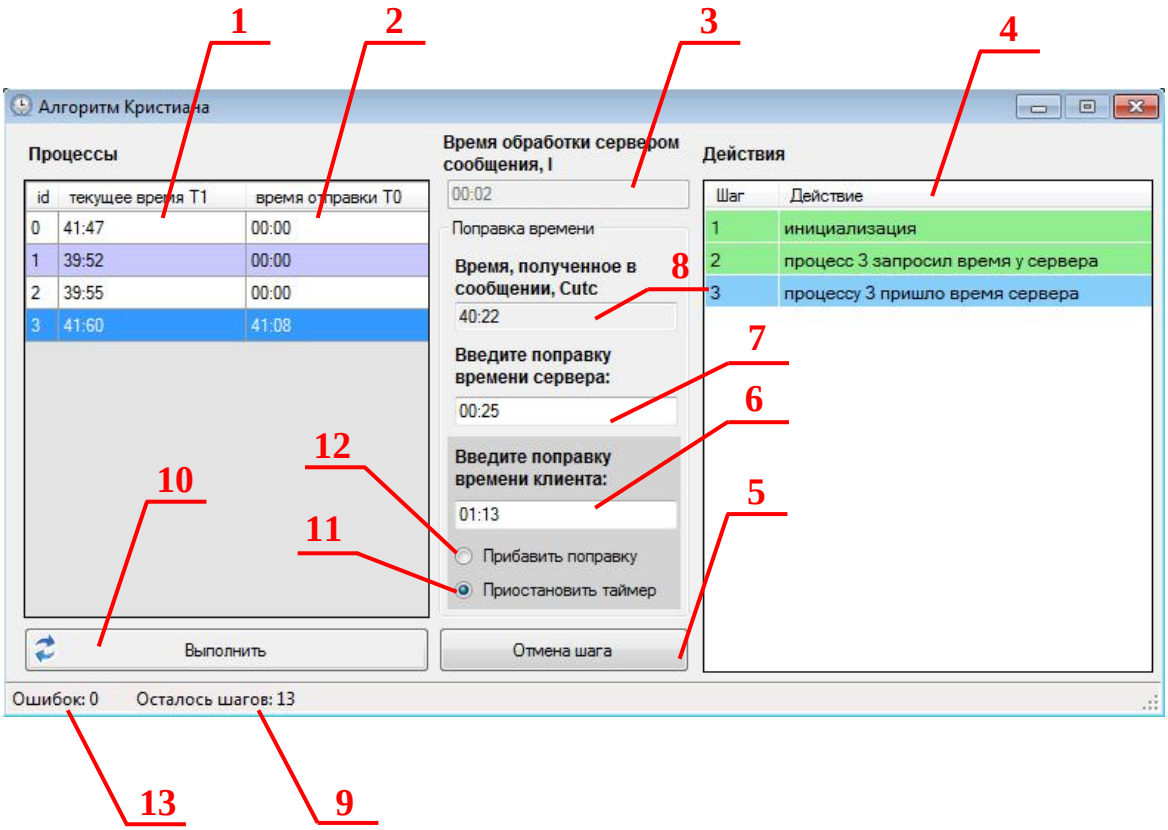
Содержание

Алгоритм Кристиана



- Действие, которое необходимо выполнить представлено в таблице **(4)**. Синий цвет обозначает текущий шаг, зелёный – выполненные правильно шаги, красный – шаг выполнен неверно.
- После выполнения каждого шага необходимо нажать кнопку «Выполнить» **(10)**;
- Для того, чтобы возвратить значения времени на начало шага, необходимо нажать кнопку «Отмена шага» **(5)**;
- Поле **(13)** обозначает количество допущенных ошибок, поле **(9)** – осталось шагов до завершения задания;

1) **Инициализация.** Чтобы проинициализировать значения времени клиента **T1** необходимо щёлкнуть двойным щелчком мыши в поле **(1)** и занести значения из задания. В поле **(3)** необходимо внести время обработки сообщения сервером **I** из задания.
2) **Запрос клиентом времени сервера.** Для сохранения текущего времени клиента **T0** необходимо щёлкнуть двойным щелчком мыши в поле **(2)** и занести в соответствующее поле значение текущего времени.
3) **Получение сообщения о текущем времени сервера.** После получения сообщения о времени сервера, необходимо сосчитать, его поправку **(7)** **(T1-T0-I)/2** *(подробнее в описании метода)*. Причём, если получилось не целое число, дробная часть должна быть отброшена. Следующим шагом является расчёт поправки времени клиента **(6)**, для чего поправка времени сервера **(7)** складывается с временем сервера в сообщении $\mathbf{C_{utc}}$ **(8)** и из неё вычитается текущее время клиента **T1**. Если результат отрицательный таймер необходимо приостановить **(11)**, иначе прибавить поправку **(12).**

Логические часы

Реализация алгоритма «Логические часы» имеет интерфейс, представленный на рисунке

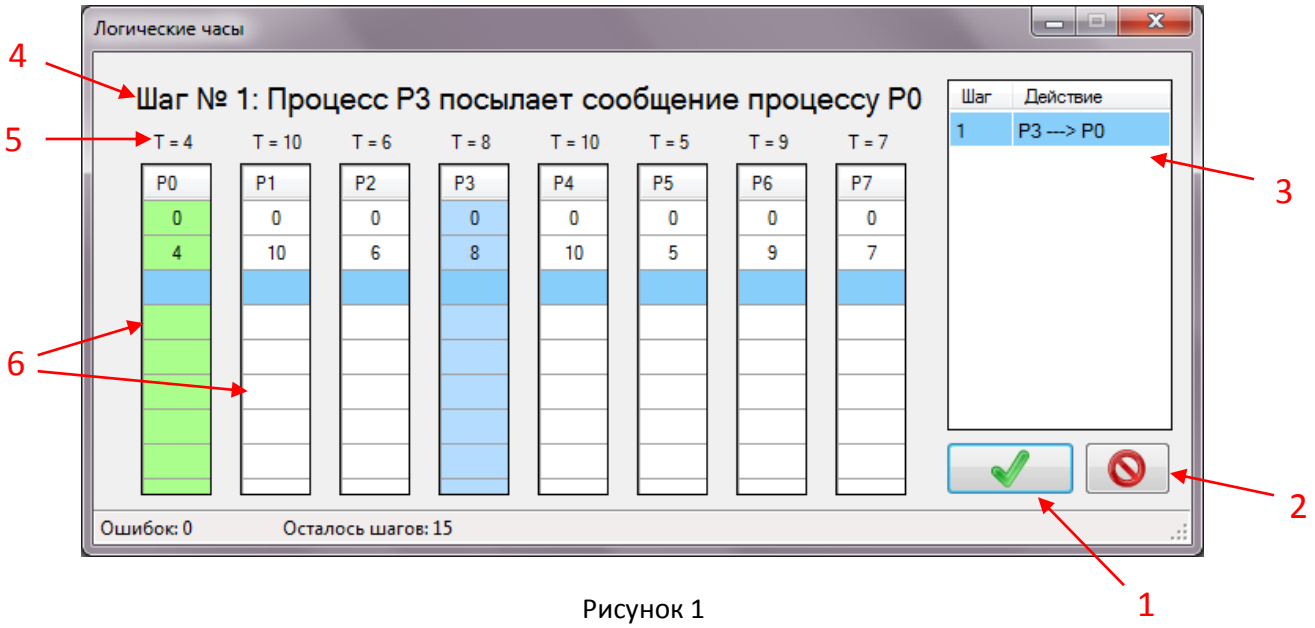


Рисунок 1

Обозначения:

1 – кнопка «Выполнить». Проверяется текущий шаг и в случае правильного выполнения осуществляется переход к следующему.

2 – кнопка «Отмена». Позволяет сделать откат изменений на начало текущего шага.

3 – история выполнения алгоритма. Текущий шаг помечен голубым цветом. Если шаг сделан правильно, то строка закрашивается зеленым цветом, в противном случае закрашивается красным.

4 – Сообщение о том, что нужно сделать в текущем шаге.

5 – показывает время такта для каждого процесса.

6 - В алгоритме рассматриваются 8 процессов, которые обмениваются сообщениями. Процесс-отправитель помечен светло-синим, а процесс-получатель светло-зеленым цветом.

Ход работы:

Пользователь заполняет значения только для процесса-отправителя и процесса-получателя. Остальные времена в строке заполняются автоматически при условии что шаг выполнен верно.

Процесс-отправитель посылает сообщение в текущий момент времени, а получателю оно приходит в следующий такт времени (у каждого процесса он свой).

Время отправления должно быть меньше**(!)** времени получения. В противном случае требуется выполнять синхронизацию. То, что именно нужно сделать, подробно описано в теоретической части методических указаний по лабораторной работе.

Алгоритм Забияки

Реализация «Алгоритма забияки» имеет интерфейс, представленный на рисунке 4

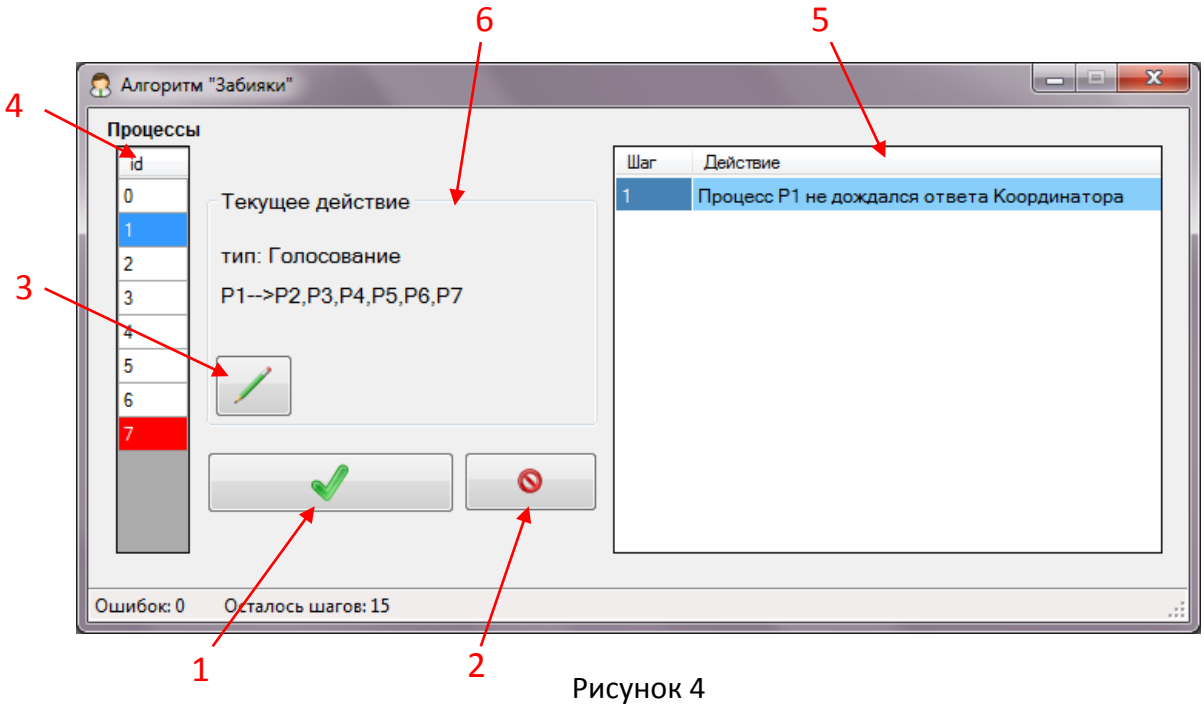


Рисунок 4

Для изменения состояния процесса и отправки сообщений используется контекстное меню (рисунок 5). Форма для отправки сообщений процесса представлена на рисунке 6.

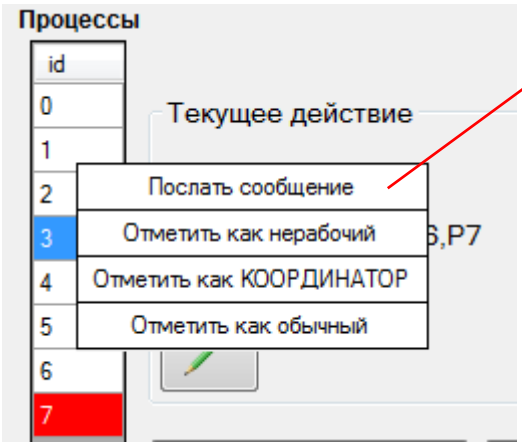


Рисунок 5

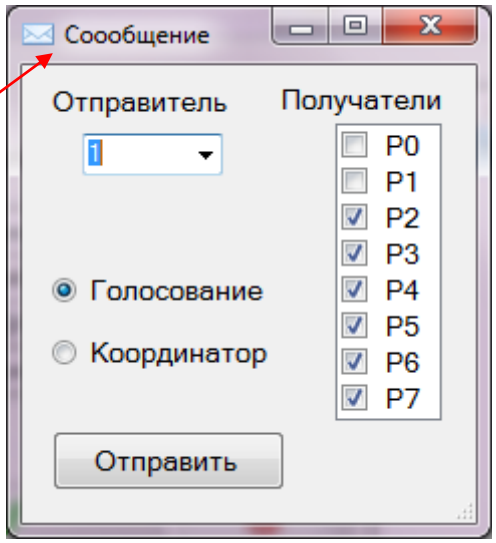


Рисунок 6

Обозначения:

1 - кнопка «Выполнить». Проверяется текущий шаг и в случае правильного выполнения осуществляется переход к следующему.

2 – кнопка «Отмена». Позволяет сделать откат изменений на начало текущего шага.

3 – кнопка «Редактирование». Позволяет редактировать сообщение.

4 – список процессов. Рабочие процессы отображены белым цветом, нерабочие – красным, координатор – синим. Отмечать процессы можно с помощью контекстного меню (кликнуть правой кнопкой по нужному процессу).

5 - история выполнения алгоритма. Текущий шаг помечен голубым цветом. Если шаг сделан правильно, то строка закрашивается зеленым цветом, в противном случае закрашивается красным.

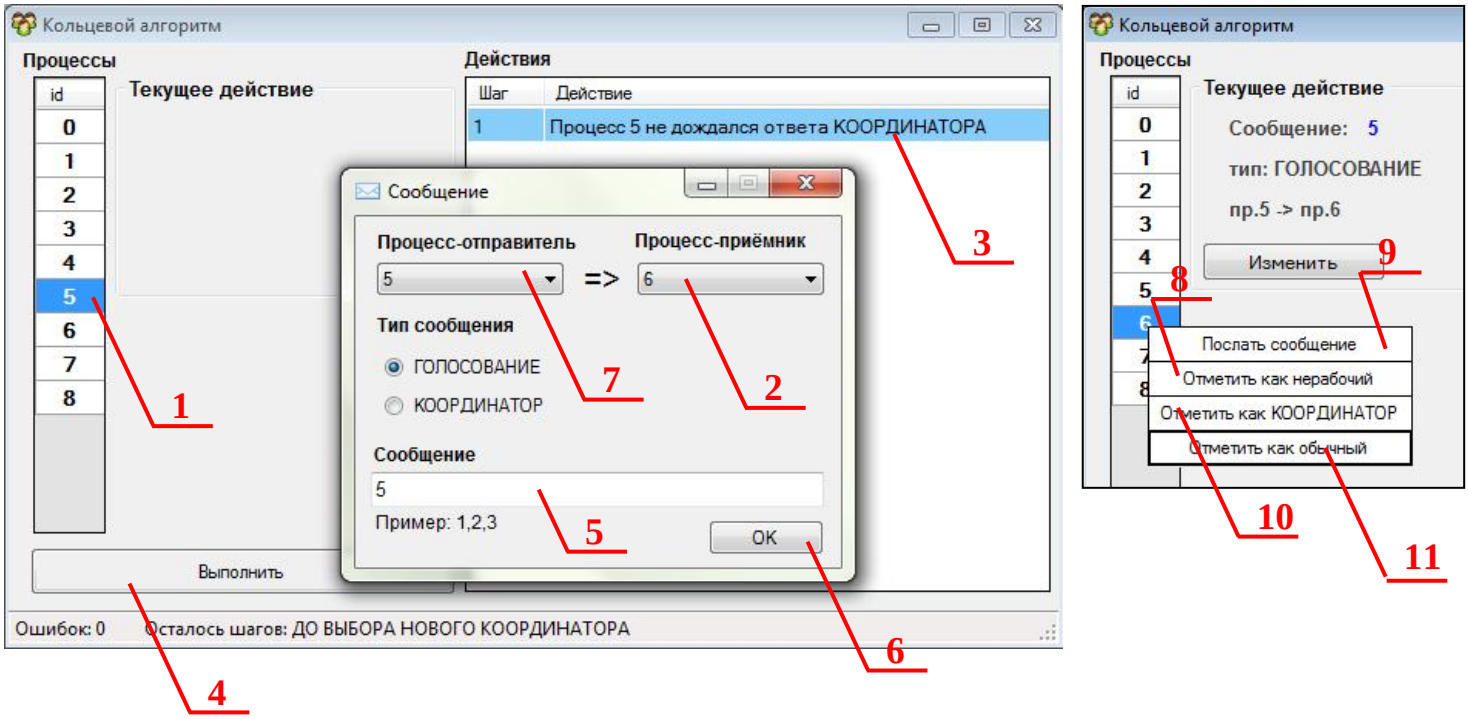
6 – панель «Текущее действие». Показывает какой тип сообщений выбран и каким процессам отправляется сообщение.

Ход работы:

Если координатор перестает замечать, то его нужно отметить как нерабочий. Отсутствие координатора замечает какой-либо процесс и организует процедуру голосования. Он должен послать сообщение с типом «Голосование» всем процессам с номерами, большими чем у него. Если есть ответившие процессы, то они продолжают голосование аналогично, если ни один процесс не ответил, то процесс-отправитель сам становится координатором и посылает **всем кроме себя(!)** сообщение с типом «Координатор».

Нерабочие процесс могут возвращаться в рабочее состояние. Вернувшийся процесс «не знает» есть ли в системе координатор и организует голосование, если ему никто не отвечает, то он сам становится координатором. Если уже есть координатор, и его номер больше, чем у вернувшегося процесса, то он отвечает ему «ОК». Однако в лабораторной установке этот шаг опущен и если вернувшийся процесс не самый «большой», то просто показывается следующий шаг для выполнения.

Кольцевой алгоритм



- Действие, которое необходимо выполнить представлено в таблице **(3)**. Синий цвет обозначает текущий шаг, зелёный – выполненные правильно шаги, красный – шаг выполнен неверно.
- После выполнения каждого шага необходимо нажать кнопку «Выполнить» **(4)**;

1) **Послать сообщение при ГОЛОСОВАНИИ.** Когда процесс заметил нерабочего КООРДИНАТОРА, он должен послать сообщение **(9)** следующему процессу по кольцу. В сообщении необходимо указать процесса-отправителя **(7)**, процесса приемника (следующего по кольцу) **(2)**, указать в сообщении номер текущего процесса (прибавлением через запятую к предыдущему сообщению). Выбрать тип сообщения ГОЛОСОВАНИЕ и подтвердить выбор **(6)**.
2) **Процесс не ответил**. Необходимо отметить этот процесс как не рабочий **(8)**, и послать сообщение следующему процессу по кольцу.
3) **ГОЛОСОВАНИЕ обошло кольцо.** Необходимо выбрать нового КООРДИНАТОРА (максимальный рабочий номер) **(10)**, и послать сообщение по **новому** кольцу (от процесса, с которого началось голосование), чтобы оповестить всех о новом КООРДИНАТОРе. Причем в формате сообщения необходимо указать тип сообщения КООРДИНАТОР с выбранным КООРДИНАТОРОМ.

Централизованный алгоритм

Внешний вид реализации «Централизованного алгоритма» представлен на рисунке 2.

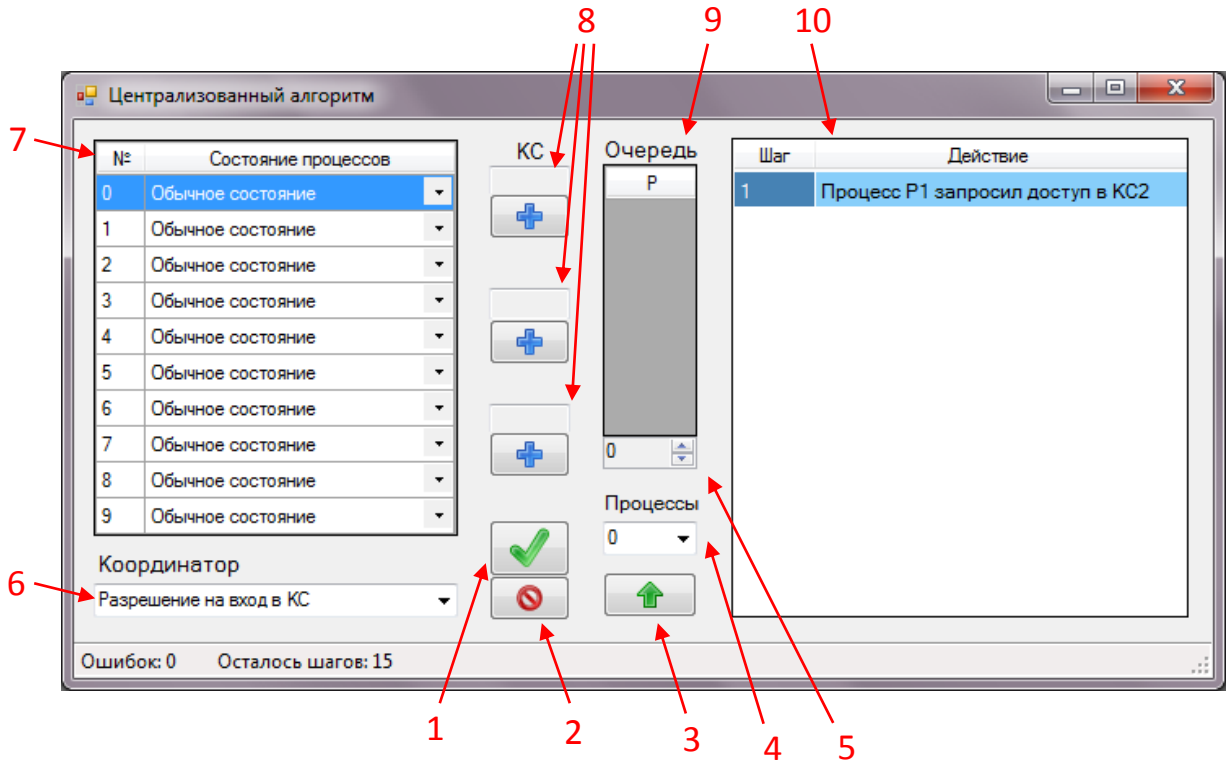


Рисунок 2

Обозначения:

1 - кнопка «Выполнить». Проверяется текущий шаг и в случае правильного выполнения осуществляется переход к следующему.

2 – кнопка «Отмена». Позволяет сделать откат изменений на начало текущего шага.

3 – добавление процесса в очередь

4 – выбор процесса для занесения в очередь

5 – выбор критической секции (КС). Всего в установке три КС.

6 – выбор состояния координатора (разрешение на вход / запрет на вход)

7 – состояния процессов (обычное состояние / в очереди / в критической секции)

8 – критические секции. Нумерация сверху вниз. С помощью кнопки в КС добавляется процесс из выбранной очереди (обозначение 5).

9 - история выполнения алгоритма. Текущий шаг помечен голубым цветом. Если шаг сделан правильно, то строка закрашивается зеленым цветом, в противном случае закрашивается красным.

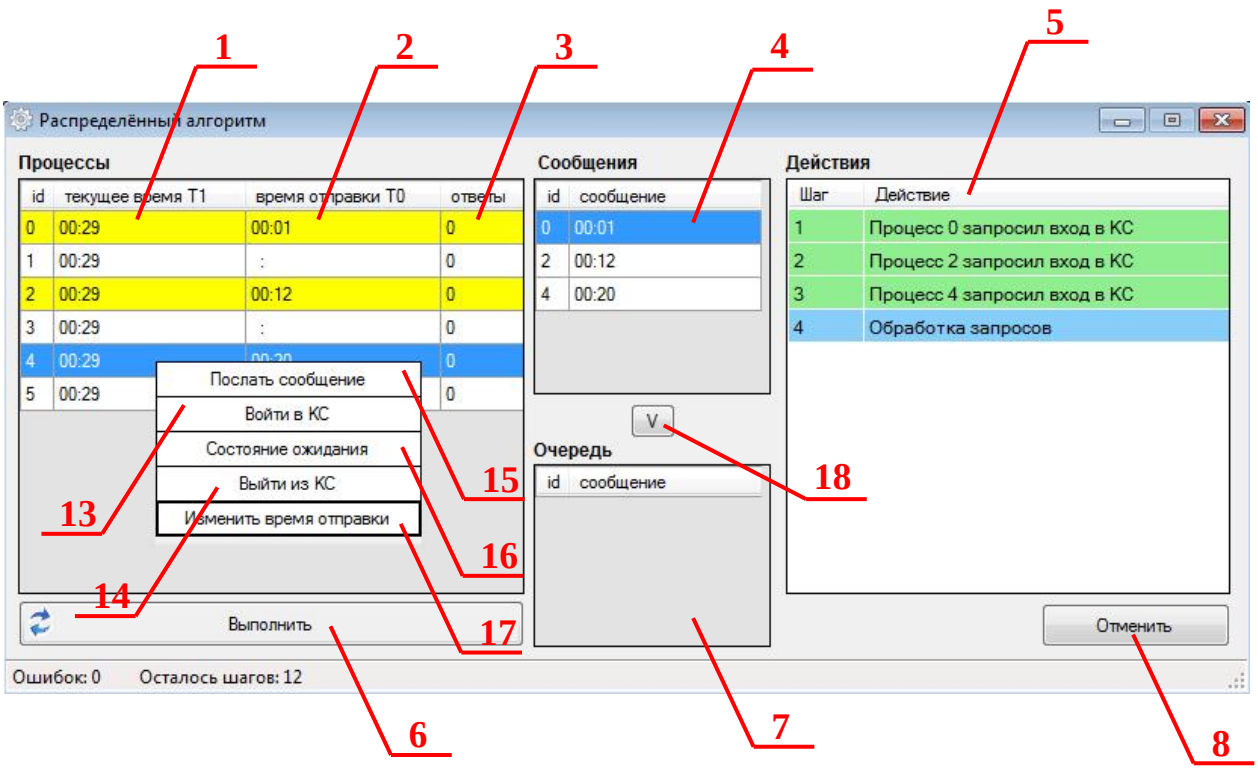
Ход работы:

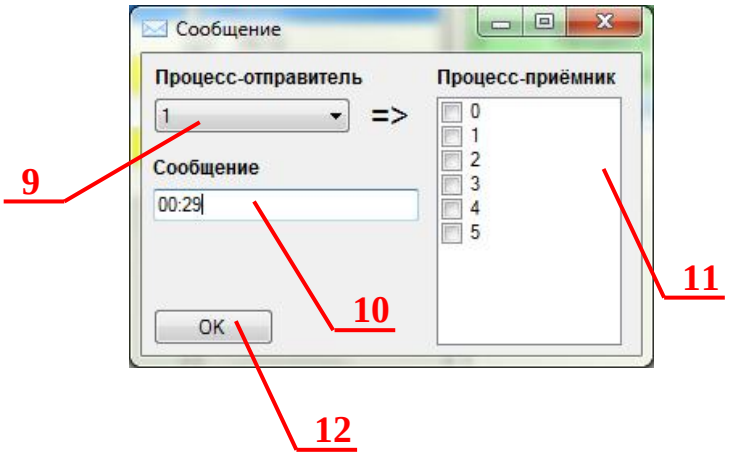
Когда процесс запрашивает вход в КС, сначала его нужно добавить в очередь к нужной КС. После этого, если КС свободна, то процесс в нее входит, изменив свое состояние. Если КС занята, то процесс остается в очереди и также изменяет свое состояние.

Координатор разрешает процессам вход в КС, если она свободна, либо запрещает, если она занята.

В случае, когда процесс выходит из КС, он меняет свое состояние, а на его место поступает процесс, находящийся первым в очереди.

Распределённый алгоритм





- Действие, которое необходимо выполнить представлено в таблице **(5)**. Синий цвет обозначает текущий шаг, зелёный – выполненные правильно шаги, красный – шаг выполнен неверно.
- После выполнения каждого шага необходимо нажать кнопку «Выполнить» **(6)**;
- Для того, чтобы возвратить значения времени на начало шага, необходимо нажать кнопку «Отмена шага» **(5)**;

1) **Запрос процессом КС.** Чтобы запросить КС, процесс должен отправить сообщения остальным клиентам, включая самого себя. Для этого необходимо щёлкнуть правой кнопкой мыши на поле **(1)** и выбрать пункт «Послать сообщение» **(15)**. В окне сообщения необходимо выбрать процесс-отправитель **(9)**, выделить процессы, которым необходимо отправить сообщение **(11)**. После ввода сообщения **(10)** (текущее время процесса) необходимо подтвердить ввод **(12)**. Далее необходимо сохранить время запроса **(17)** и ввести процесс в состояние ожидания **(16)**.

2) **Обработка запросов.** При обработке запросов необходимо ответить на все поступившие сообщения в очередях всех процессов.

- Если процесс запрашивал КС и его время запроса больше, чем в сообщении в очереди **(4),** он может ответить ОК (для этого необходимо щёлкнуть правой кнопкой мыши на сообщении в очереди). Если время запроса меньше, чем в сообщении в очереди сообщений **(4),** он добавляет сообщение в свою очередь **(7),** кнопкой **(18).**

- Если процесс не запрашивал вход в КС, он может ответить на все сообщения в очереди.

После обработки сообщений, у одного из процессов будут собраны все ответы **(3)** . Он может войти в КС **(13)**.

3) **Выход из КС.** Когда процесс выходит из КС, необходимо обозначить его как обычный процесс **(14)**, изменить время отправки **(17)**, просто удалив значение и отправить ОК всем процессам из его очереди **(7)**. В этом же шаге выполняется **обработка запросов и** если есть процесс, у которого будут все ответы, он входит в КС.

Алгоритм маркерного кольца

Реализация алгоритма «Алгоритма маркерного кольца» имеет интерфейс, представленный на рисунке 3.

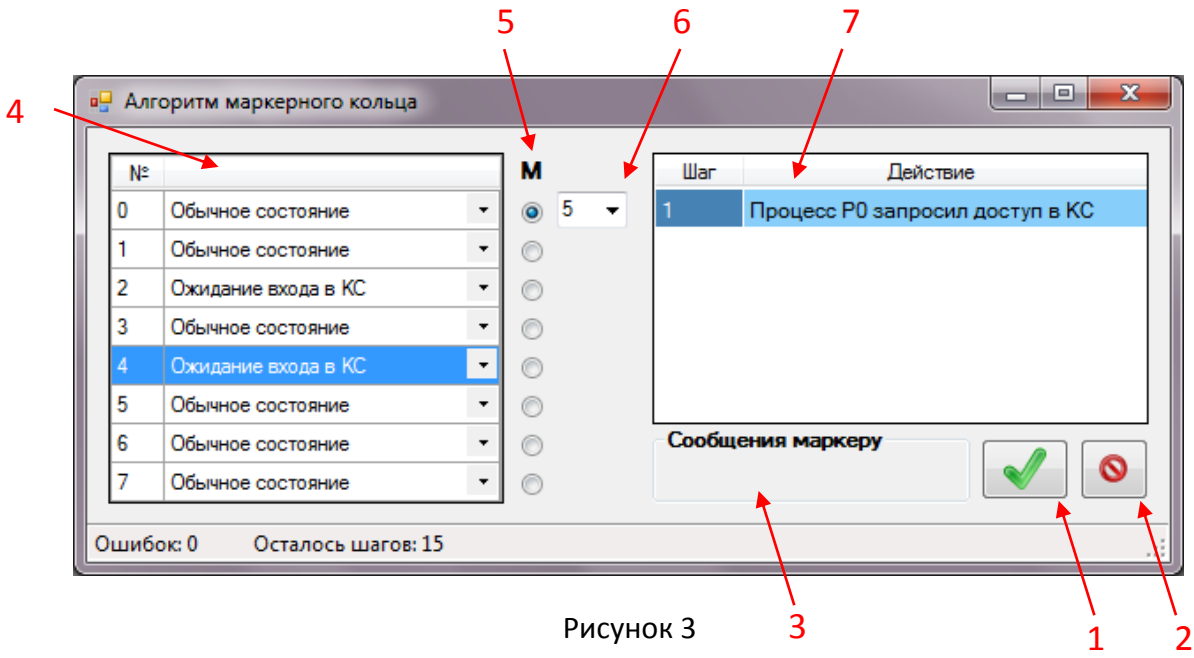


Рисунок 3

Обозначения:

1 - кнопка «Выполнить». Проверяется текущий шаг и в случае правильного выполнения осуществляется переход к следующему.

2 – кнопка «Отмена». Позволяет сделать откат изменений на начало текущего шага.

3 – Панель для отображения, ответил ли процесс маркеру.

4 - состояния процессов (Обычное состояние / Ожидание входа в КС / В критической секции).

5 – маркер

6 – сообщение, которое маркер посылает процессу.

7 - история выполнения алгоритма. Текущий шаг помечен голубым цветом. Если шаг сделан правильно, то строка закрашивается зеленым цветом, в противном случае закрашивается красным.

Ход работы:

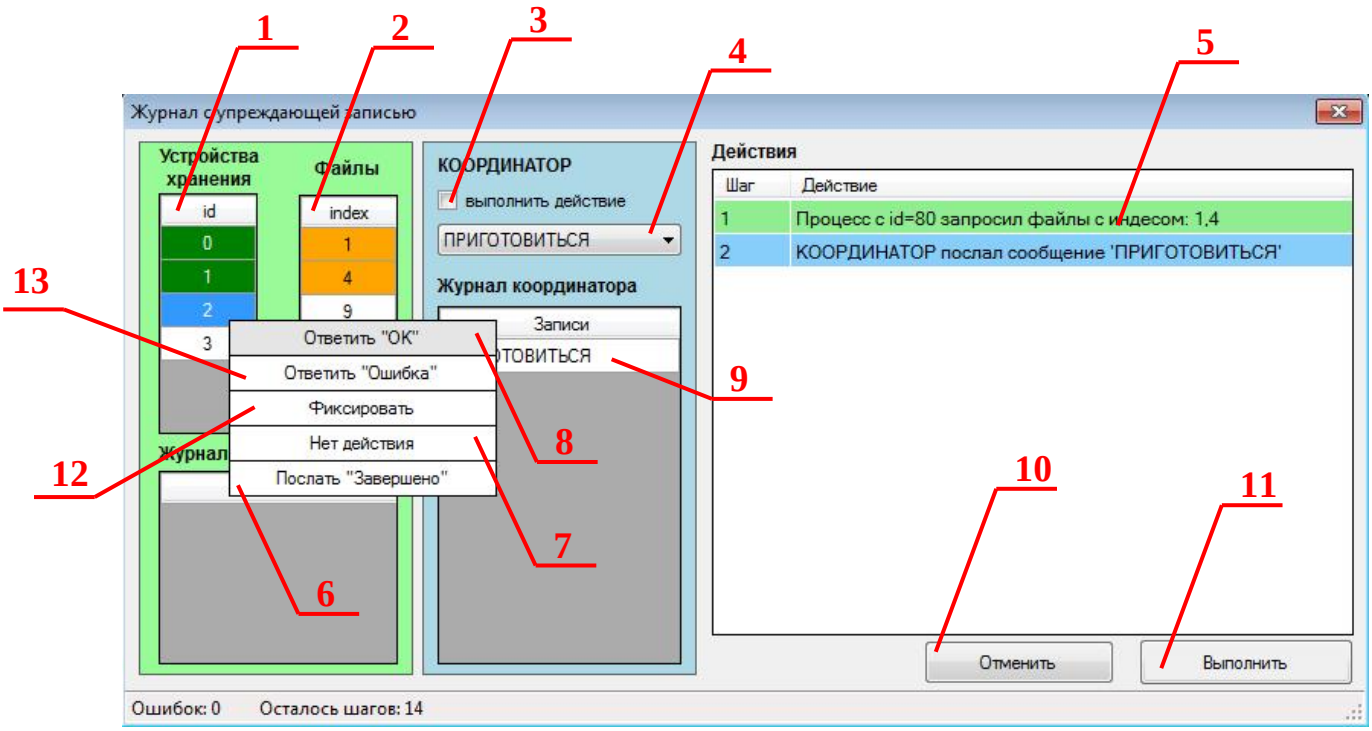
В данном алгоритме процесс может зайти в КС только в случае, если у него сейчас есть маркер.

На каждом шаге маркер посылает сообщение следующему процессу. Если процесс рабочий, то он отвечает маркеру «ОК» и на следующем шаге маркер может на него перейти. Если процесс не отвечает, то маркер остается в том же процессе и посылает сообщение следующему процессу.

Нерабочий процесс помечается красным цветом. Для этого нужно щелкнуть один раз по номеру процесса в таблице состояний процесса.

Возможна ситуация, что маркер отослал сообщение процессу, процесс ответил «ОК» и в это же время процесс, в котором находится маркер, запрашивает вход в КС. В этом случае маркер перемещается в процесс, который ответил «ОК», а предыдущий процесс помечается как ожидающий КС.

Журнал с упреждающей записью



- Действие, которое необходимо выполнить представлено в таблице **(5)**. Синий цвет обозначает текущий шаг, зелёный – выполненные правильно шаги, красный – шаг выполнен неверно.
- После выполнения каждого шага необходимо нажать кнопку «Выполнить» **(11)**;
- Для того, чтобы возвратить значения времени на начало шага, необходимо нажать кнопку «Отмена шага» **(10)**;
- Для очистки полей устройств хранения, необходимо нажить кнопку «Нет действия» **(7)**.

Алгоритм состоит из следующих шагов:

1) **Запрос процессом файлов с определёнными индексами.** Когда процесс запросил изменение файлов с определёнными индексами, КООРДИНАТОР должен запросить **(3)** готовность этих файлов, которые могут храниться на различных устройствах. Для этого необходимо послать команду ПРИГОТОВИТЬСЯ **(4)**, она автоматически запишется в журнал **(9)**.
2) **Ответ от устройств хранения.** После того, как послана команда ПРИГОТОВИТЬСЯ, файлы, с запрошенными индексами проверяются на ошибки, и если они доступны для изменения, устройство отвечает ОК. Необходимо для каждого устройства хранения **(1)** выбрать файлы **(2)** с запрошенными индексами и ответить ОК КООРДИНАТОРУ **(8)**, если файлы присутствуют и нет ошибки (выделенный файл не помечен красным цветом), иначе послать сигнал ошибки **(13).**
3) **Все устройства ответили ОК.** В этом случае КООРДИНАТОР разрешает процессу изменять файлы, посылая команду ФИКСИРОВАТЬ **(4)**.
   Какое-либо устройство ответило сигналом об ошибке. В этом случае КООРДИНАТОР должен сделать ОТКАТ, посылая всем соответствующую команду **(4)**. На этом процесс обработки запроса прерывается.
4) **Фиксирование изменений.** После команды ФИКСИРОВАТЬ, каждое устройство, содержащее файлы с запрошенными индексами фиксирует изменения, которые вносит процесс. Для этого необходимо отметить соответствующие устройства хранения командой ФИКСИРОВАТЬ **(12)**.
5) **Завершение работы**. После того, как изменения завершены, каждое устройство хранения посылает сообщение о том, что оно завершило работу **(6)**.

6) **Завершение транзакции**. После того, как каждое устройство послало сообщение о завершении работы, КООРДИНАТОР должен завершить транзакцию соответствующей командой **(4)**.