

Отчёт по лабораторной работе №11

Имитационное моделирование

Ганина Таисия Сергеевна, НФИбд-01-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	Задание деклараций системы	10
4.2	Параметры элементов основного графа системы обработки заявок в очереди	11
4.3	Параметры элементов генератора заявок системы	12
4.4	Параметры элементов обработчика заявок системы	13
4.5	Запуск системы обработки заявок в очереди	13
4.6	Функция Predicate монитора Ostanovka	14
4.7	Функция Observer монитора Queue Delay	14
4.8	Файл Queue_Delay.log	15
4.9	График изменения задержки в очереди	16
4.10	Функция Observer монитора Queue Delay Real	16
4.11	Содержимое Queue_Delay_Real.log	17
4.12	Функция Observer монитора Long Delay Time	17
4.13	Определение longdelaytime в декларациях	18
4.14	Содержимое Long_Delay_Time.log	19
4.15	Периоды времени, когда значения задержки в очереди превышали заданное значение	20

Список таблиц

1 Цель работы

Реализовать модель $M|M|1$ в CPN tools.

2 Задание

- Реализовать в CPN Tools модель системы массового обслуживания $M|M|1$.
- Настроить мониторинг параметров моделируемой системы и нарисовать графики очереди.

3 Теоретическое введение

CPN Tools — специальное программное средство, предназначенное для моделирования иерархических временных раскрашенных сетей Петри. Такие сети эквивалентны машине Тьюринга и составляют универсальную алгоритмическую систему, позволяющую описать произвольный объект. CPN Tools позволяет визуализировать модель с помощью графа сети Петри и применить язык программирования CPN ML (Colored Petri Net Markup Language) для формализованного описания модели.

Назначение CPN Tools:

- разработка сложных объектов и моделирование процессов в различных прикладных областях, в том числе:
- моделирование производственных и бизнес-процессов;
- моделирование систем управления производственными системами и роботами;
- спецификация и верификация протоколов, оценка пропускной способности сетей и качества обслуживания, проектирование телекоммуникационных устройств и сетей.

Основные функции CPN Tools:

- создание (редактирование) моделей;
- анализ поведения моделей с помощью имитации динамики сети Петри;
- построение и анализ пространства состояний модели.

[1,2].

4 Выполнение лабораторной работы

Постановка задачи

В систему поступает поток заявок двух типов, распределённый по пуассоновскому закону. Заявки поступают в очередь сервера на обработку. Дисциплина очереди - FIFO. Если сервер находится в режиме ожидания (нет заявок на сервере), то заявка поступает на обработку сервером.

Будем использовать три отдельных листа: на первом листе опишем граф системы, на втором — генератор заявок, на третьем — сервер обработки заявок.

Сеть имеет 2 позиции (очередь — Queue, обслуженные заявки — Complited) и два перехода (генерировать заявку — Arrivals, передать заявку на обработку серверу — Server). Переходы имеют сложную иерархическую структуру, задаваемую на отдельных листах модели (с помощью соответствующего инструмента меню — Hierarchy).

Зададим декларации системы (рис. 4.1).

Определим множества цветов системы (colorset):

- фишки типа UNIT определяют моменты времени;
- фишки типа INT определяют моменты поступления заявок в систему.
- фишки типа JobType определяют 2 типа заявок — A и B;
- кортеж Job имеет 2 поля: jobType определяет тип работы (соответственно имеет тип JobType, поле AT имеет тип INT и используется для хранения времени нахождения заявки в системе);
- фишки Jobs — список заявок;

- фишки типа `ServerxJob` — определяют состояние сервера, занятого обработкой заявок.

Переменные модели:

- `proctime` — определяет время обработки заявки;
- `job` — определяет тип заявки;
- `jobs` — определяет поступление заявок в очередь.

Определим функции системы:

- функция `expTime` описывает генерацию целочисленных значений через интервалы времени, распределённые по экспоненциальному закону;
- функция `intTime` преобразует текущее модельное время в целое число;
- функция `newJob` возвращает значение из набора `Job` — случайный выбор типа заявки (А или В).

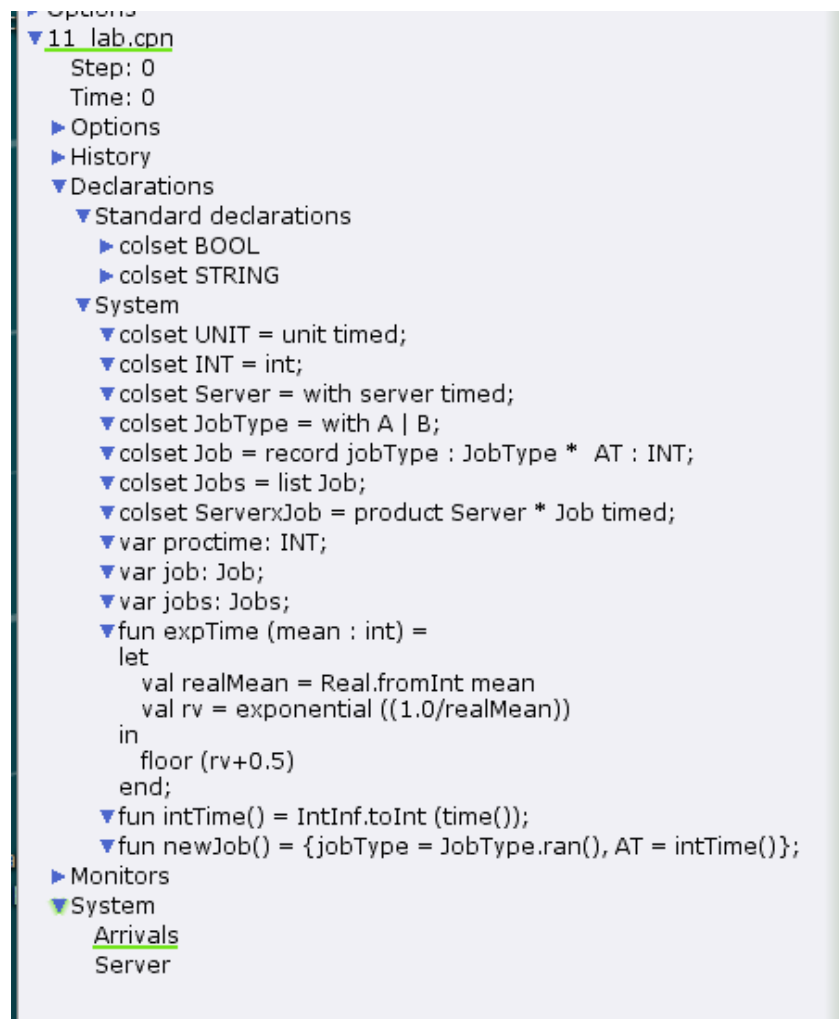


Рис. 4.1: Задание деклараций системы

Зададим параметры модели на графах сети.

На листе System (рис. 4.2):

- у позиции Queue множество цветов фишек — Jobs; начальная маркировка 1 ``[] определяет, что изначально очередь пуста.
- у позиции Completed множество цветов фишек — Job.

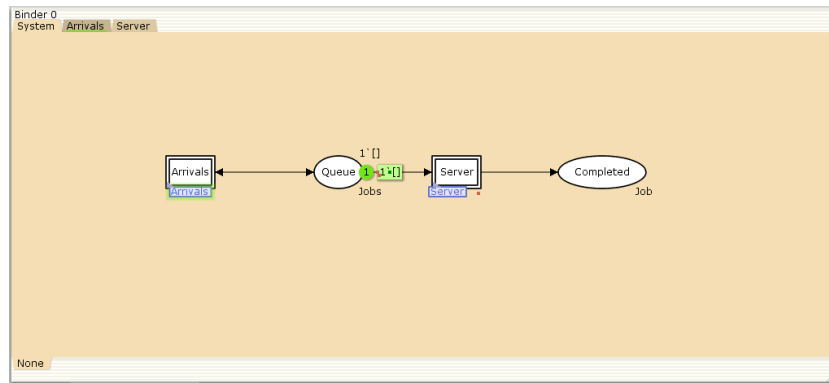


Рис. 4.2: Параметры элементов основного графа системы обработки заявок в очереди

На листе Arrivals (рис. 4.3):

- у позиции `Init`: множество цветов фишек — `UNIT`; начальная маркировка `1'`()`@0 определяет, что поступление заявок в систему начинается с нулевого момента времени;
- у позиции `Next`: множество цветов фишек — `UNIT`;
- на дуге от позиции `Init` к переходу `Init` выражение `()` задаёт генерацию заявок;
- на дуге от переходов `Init` и `Arrive` к позиции `Next` выражение `()@+expTime(100)` задаёт экспоненциальное распределение времени между поступлениями заявок;
- на дуге от позиции `Next` к переходу `Arrive` выражение `()` задаёт перемещение фишки;
- на дуге от перехода `Arrive` к позиции `Queue` выражение `jobs^[job]` задает поступление заявки в очередь;
- на дуге от позиции `Queue` к переходу `Arrive` выражение `jobs` задаёт обратную связь.

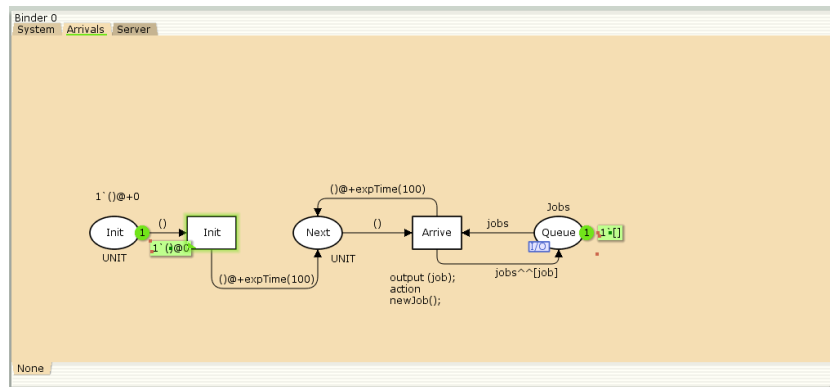


Рис. 4.3: Параметры элементов генератора заявок системы

На листе Server (рис. 4.4):

- у позиции Busy: множество цветов фишек — Server, начальное значение мар-кировки — $1 \cdot \text{server} @ 0$ определяет, что изначально на сервере нет заявок на обслуживание;
- у позиции Idle: множество цветов фишек — $\text{Server} \times \text{Job}$;
- переход Start имеет сегмент кода `output (proctime); action expTime(90);` определяющий, что время обслуживания заявки распределено по экспоненциальному закону со средним временем обработки в 90 единиц времени;
- на дуге от позиции Queue к переходу Start выражение `job : : jobs` определяет, что сервер может начать обработку заявки, если в очереди есть хотя бы одна заявка;
- на дуге от перехода Start к позиции Busy выражение `(server , job) @ + proctime` запускает функцию расчёта времени обработки заявки на сервере;
- на дуге от позиции Busy к переходу Stop выражение `(server , job)` говорит о завершении обработки заявки на сервере;
- на дуге от перехода Stop к позиции Completed выражение `job` показывает, что заявка считается обслуженной;
- выражение `server` на дугах от и к позиции Idle определяет изменение состояние сервера (обрабатывает заявки или ожидает);

- на дуге от перехода Start к позиции Queue выражение jobs задаёт обратную связь.

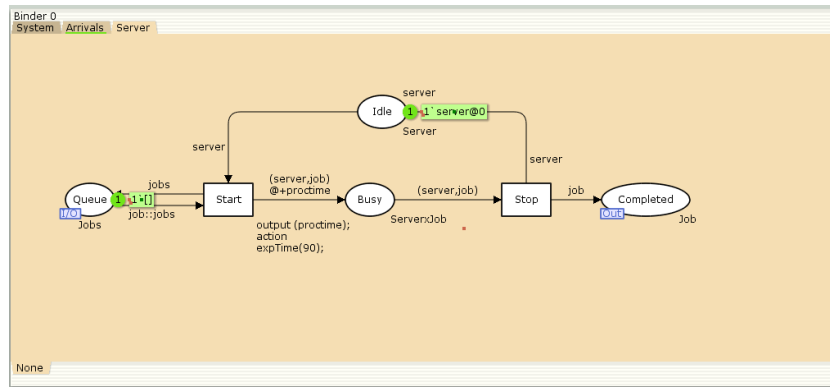


Рис. 4.4: Параметры элементов обработчика заявок системы

После этого можно запустить модель и мы получим следующую картину (рис. 4.5)

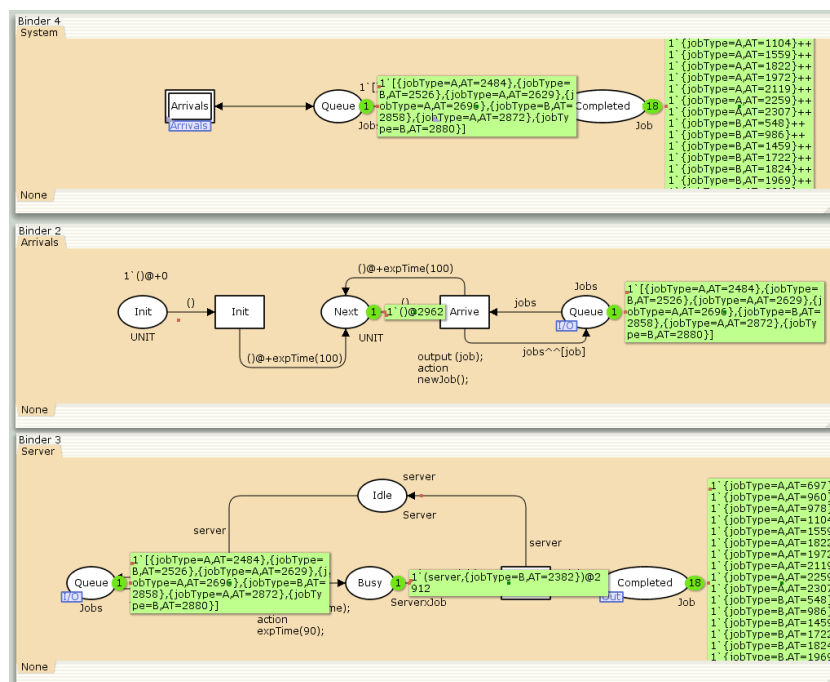
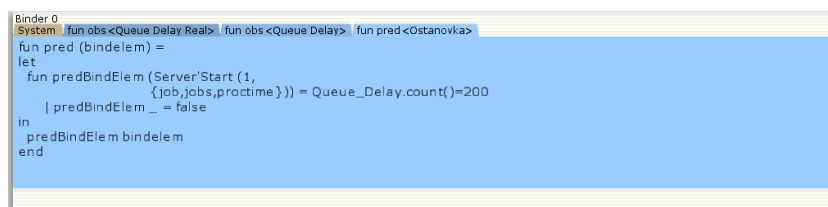


Рис. 4.5: Запуск системы обработки заявок в очереди

Мониторинг параметров моделируемой системы

Потребуется палитра Monitoring. Выбираем Break Point (точка останова) и устанавливаем её на переход Start. После этого в разделе меню Monitor появится новый подраздел, который назовём Ostanovka. В этом подразделе необходимо внести изменения в функцию Predicate, которая будет выполняться при запуске монитора. Зададим число шагов, через которое будем останавливать мониторинг. Для этого true заменим на `Queue_Delay.count()=200`.

В результате функция примет вид (рис. 4.6):



```

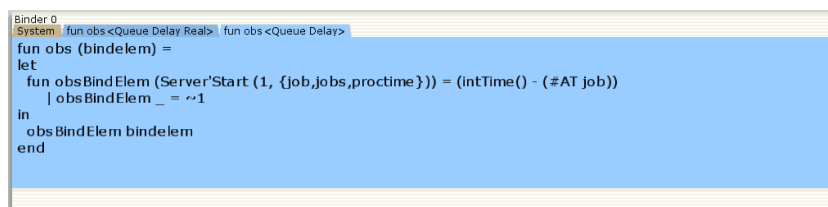
Binder 0
System fun obs<Queue Delay Real> fun obs<Queue Delay> fun pred<Ostanovka>
fun pred (bindelem) =
let
  fun predBindElem (Server'Start (1, {job,jobs,proctime})) = Queue_Delay.count()=200
  | predBindElem _ = false
in
  predBindElem bindelem
end

```

Рис. 4.6: Функция Predicate монитора Ostanovka

Необходимо определить конструкцию `Queue_Delay.count()`. С помощью палитры Monitoring выбираем Data Call и устанавливаем на переходе Start. Появившийся в меню монитор называем Queue Delay (без подчеркивания). Функция Observer выполняется тогда, когда функция предикатора выдаёт значение true. По умолчанию функция выдаёт 0 или унарный минус (~1), подчёркивание обозначает произвольный аргумент. Изменим её так, чтобы получить значение задержки в очереди. Для этого необходимо из текущего времени `intTime()` вычесть временную метку AT, означающую приход заявки в очередь.

В результате функция примет вид (рис. 4.7):



```

Binder 0
System fun obs<Queue Delay Real> fun obs<Queue Delay>
fun obs (bindelem) =
let
  fun obsBindElem (Server'Start (1, {job,jobs,proctime})) = (intTime() - (#AT job))
  | obsBindElem _ = ~1
in
  obsBindElem bindelem
end

```

Рис. 4.7: Функция Observer монитора Queue Delay

После запуска программы на выполнение в каталоге с кодом программы появится файл Queue_Delay.log (рис. 4.8), содержащий в первой колонке — значение задержки очереди, во второй — счётчик, в третьей — шаг, в четвёртой — время.

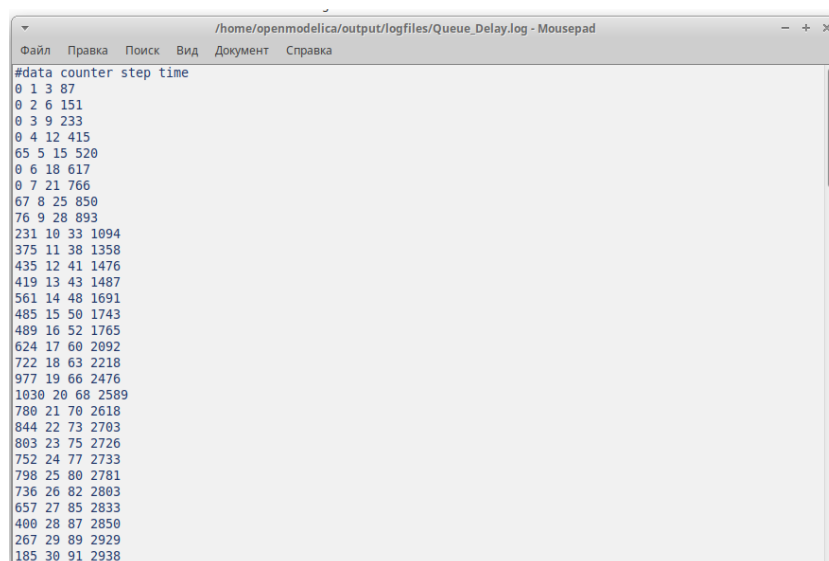


Рис. 4.8: Файл Queue_Delay.log

С помощью gnuplot можно построить график значений задержки в очереди (рис. 4.9), выбрав по оси x время, а по оси y — значения задержки:

```
#!/usr/bin/gnuplot -persist
# задаём текстовую кодировку,
# тип терминала, тип и размер шрифта

set encoding utf8
set term pngcairo font "Helvetica,9"

# задаём выходной файл графика
set out 'window_1.png'
plot "Queue_Delay.log" using ($4):($1) with lines
```

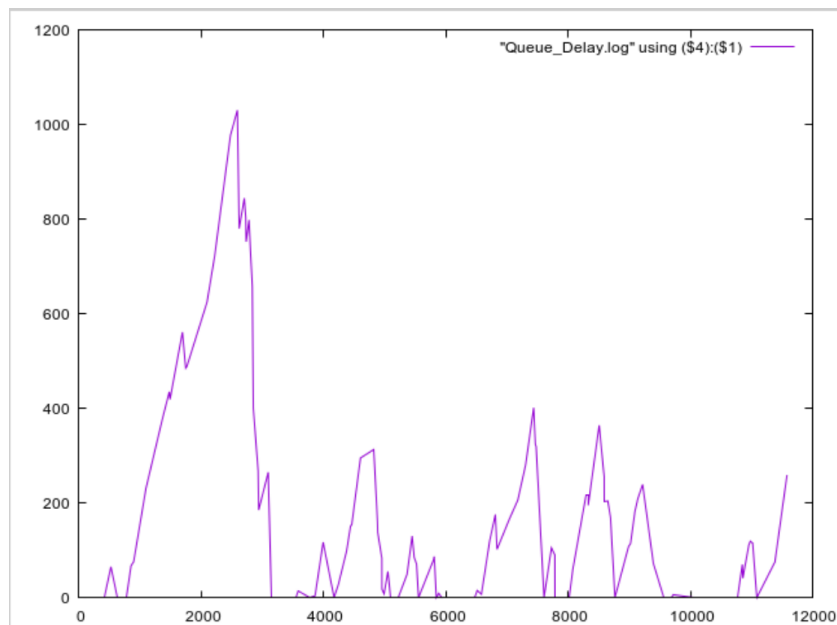


Рис. 4.9: График изменения задержки в очереди

Посчитаем задержку в действительных значениях. С помощью палитры Monitoring выбираем Data Call и устанавливаем на переходе Start. Появившийся в меню монитор называем Queue Delay Real. Функцию Observer изменим следующим образом (рис. 4.10):

```

Binder 0
System fun obs<Queue Delay Real>
fun obs (bindelem) =
let
  fun obsBindElem (ServerStart (1, {job,jobs,proctime})) = Real.fromInt(IntTime() - (#AT job))
  | obsBindElem _ = ~1.0
in
  obsBindElem bindelem
end

```

Рис. 4.10: Функция Observer монитора Queue Delay Real

По сравнению с предыдущим описанием функции добавлено преобразование значения функции из целого в действительное, при этом obsBindElem _ принимает значение ~1.0. После запуска программы на выполнение в каталоге с кодом программы появится файл Queue_Delay_Real.log с содержимым, аналогичным содержимому файла Queue_Delay.log, но значения задержки имеют действитель-

ный тип (рис. 4.11):

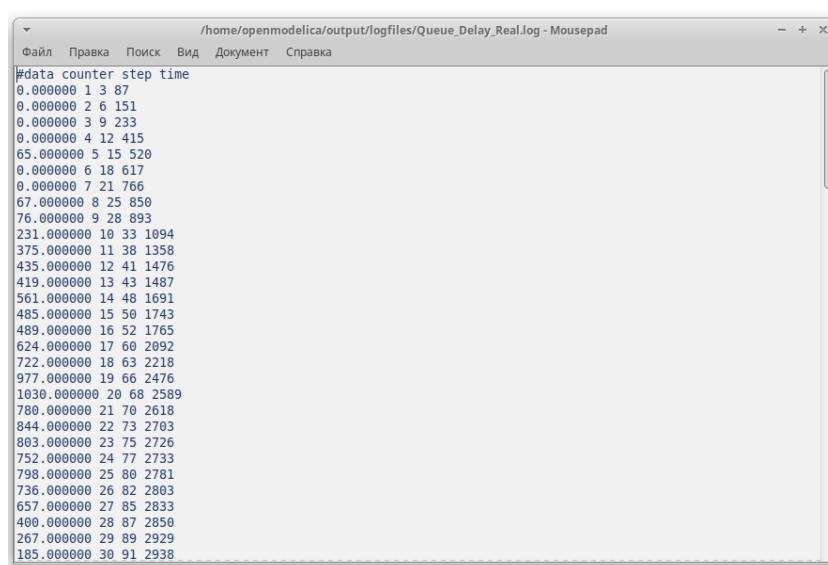


Рис. 4.11: Содержимое Queue_Delay_Real.log

Посчитаем, сколько раз задержка превысила заданное значение. С помощью палитры Monitoring выбираем Data Call и устанавливаем на переходе Start. Монитор называем Long Delay Time. Функцию Observer изменим следующим образом (рис. 4.12):

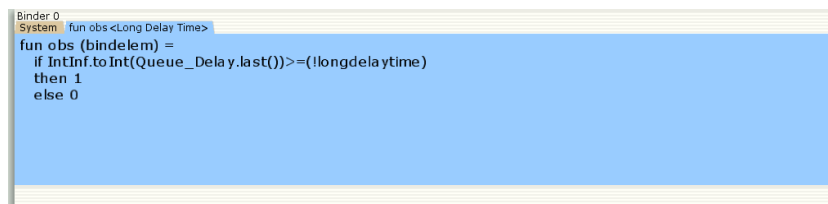


Рис. 4.12: Функция Observer монитора Long Delay Time

При этом необходимо в декларациях задать глобальную переменную (в форме ссылки на число 200): longdelaytime (рис. 4.13).

```
▼ 11 lab.cpn
  Step: 0
  Time: 0
  ▶ Options
  ▶ History
  ▼ Declarations
    ▶ System
    ▶ Standard declarations
    ▶ globref longdelaytime
  ▼ Monitors
    ▶ Queue Delay
    ▶ Queue Delay Real
    ▶ Long Delay Time
    ▶ Ostanovka
  ▼ System
    Arrivals
    Server
```

Рис. 4.13: Определение longdelaytime в декларациях

После запуска программы на выполнение в каталоге с кодом программы появится файл Long_Delay_Time.log (рис. 4.14)

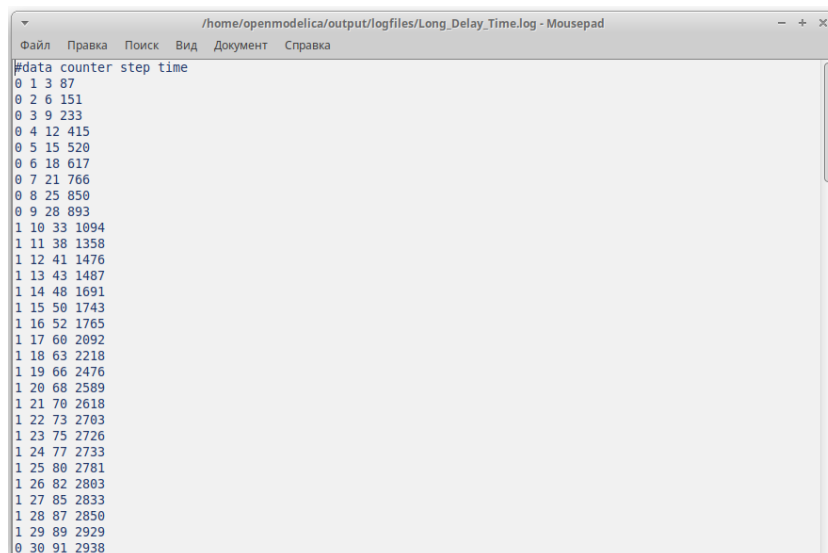


Рис. 4.14: Содержимое Long_Delay_Time.log

С помощью gnuplot можно построить график (рис. 4.15), демонстрирующий, в какие периоды времени значения задержки в очереди превышали заданное значение 200.

```
#!/usr/bin/gnuplot -persist
# задаём текстовую кодировку,
# тип терминала, тип и размер шрифта

set encoding utf8
set term pngcairo font "Helvetica,9"

# задаём выходной файл графика
set out 'window_2.png'
set style line 2
plot [0:] [0:1.2] "Long_Delay_Time.log" using ($4):($1) with lines
```

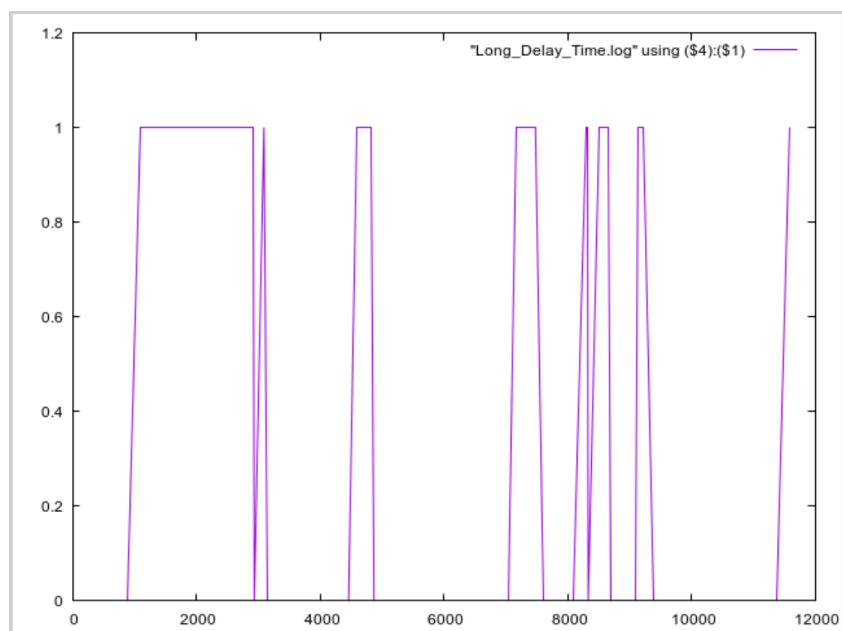


Рис. 4.15: Периоды времени, когда значения задержки в очереди превышали заданное значение

5 Выводы

В процессе выполнения данной лабораторной работы я реализовала модель системы массового обслуживания $M|M|1$ в CPN Tools.

Список литературы

1. Цветные сети Петри и язык распределенного программирования UPL: их сравнение и перевод, Аркадий Валентинович Климов [Электронный ресурс]. URL: https://psta.psiras.ru/read/psta2023_4_91-122.pdf.
2. CPN Tools, Michael Westergaard, August 2010, Eindhoven, Netherlands [Электронный ресурс]. URL: <https://westergaard.eu/wp-content/uploads/2010/09/CPN-Tools.pdf>.