

Лабораторная работа №11

Дисциплина: Имитационное моделирование

Пронякова Ольга Максимовна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	17
	Список литературы	18

Список иллюстраций

2.1	Граф сети системы обработки заявок в очереди	7
2.2	Граф генератора заявок системы	8
2.3	Граф процесса обработки заявок на сервере системы	8
2.4	Декларации системы	9
2.5	Декларации системы	10
2.6	Декларации системы	10
2.7	Декларации системы	11
2.8	Декларации системы	11
2.9	Декларации системы	12
2.10	Параметры элементов основного графа системы обработки заявок в очереди	13
2.11	Параметры элементов генератора заявок системы	14
2.12	Параметры элементов обработчика заявок системы	15
2.13	Запуск модели	15
2.14	Запуск модели	16

Список таблиц

1 Цель работы

Научиться работать с Моделью системы массового обслуживания $M | M | 1$

2 Выполнение лабораторной работы

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 (рис. 11.1), на втором — генератор заявок (рис. 11.2), на третьем — сервер обработки заявок (рис. 11.3). 1.1. 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)(рис.2.1).

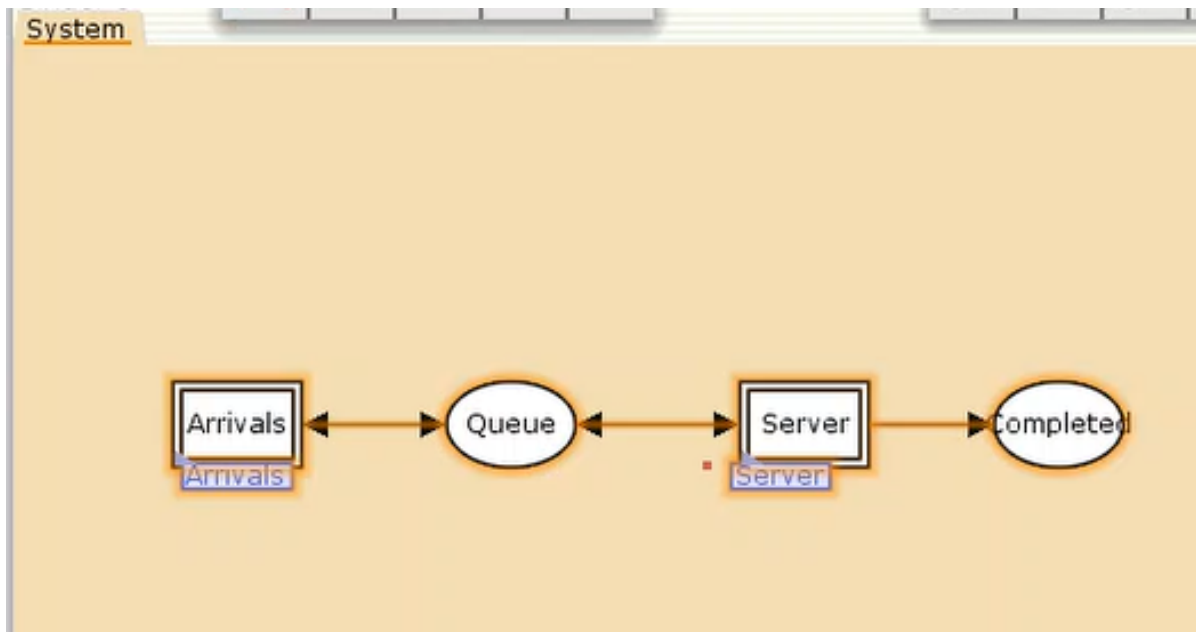


Рис. 2.1: Граф сети системы обработки заявок в очереди

граф генератора заявок имеет 3 позиции (текущая заявка — Init, следующая заявка — Next, очередь — Queue из листа System) и 2 перехода (Init — определяет распределение поступления заявок по экспоненциальному закону с интенсивностью 100 заявок в единицу времени, Arrive — определяет поступление заявок в очередь)(рис.2.2).

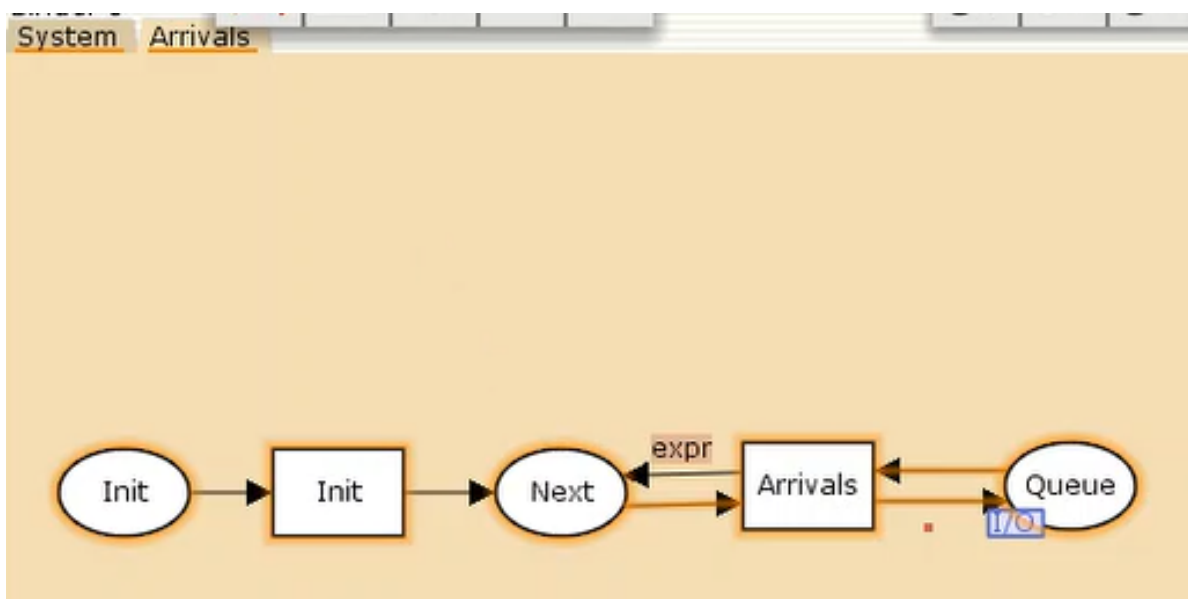


Рис. 2.2: Граф генератора заявок системы

Граф процесса обработки заявок на сервере имеет 4 позиции (Busy — сервер занят, Idle — сервер в режиме ожидания, Queue и Completed из листа System) и 2 перехода (Start — начать обработку заявки, Stop — закончить обработку заявки)(рис.2.3).

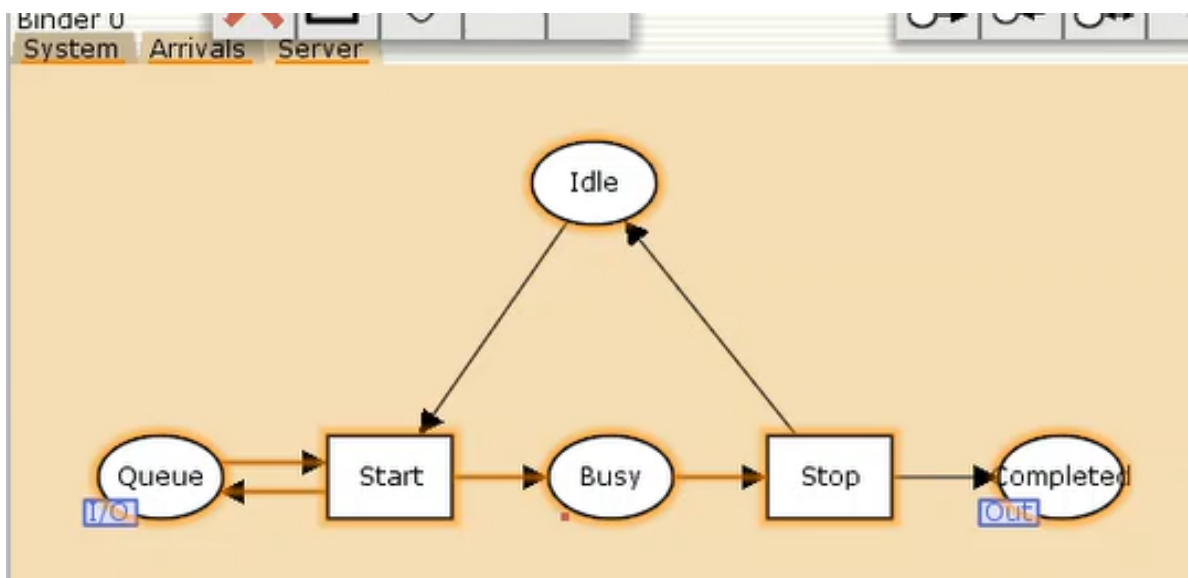


Рис. 2.3: Граф процесса обработки заявок на сервере системы

Зададим декларации системы. 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(рис.2.4).

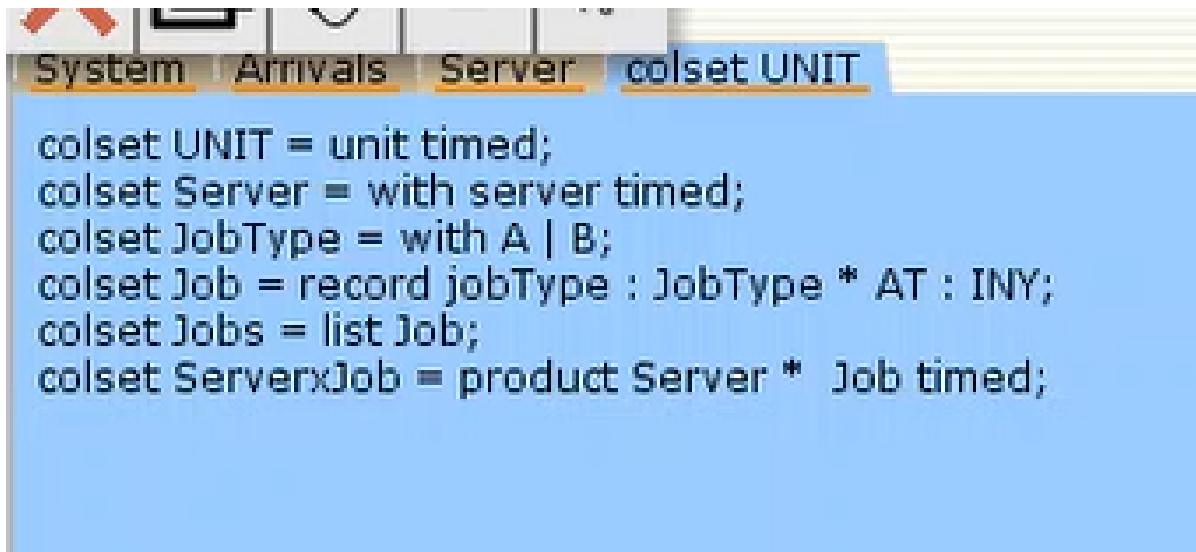
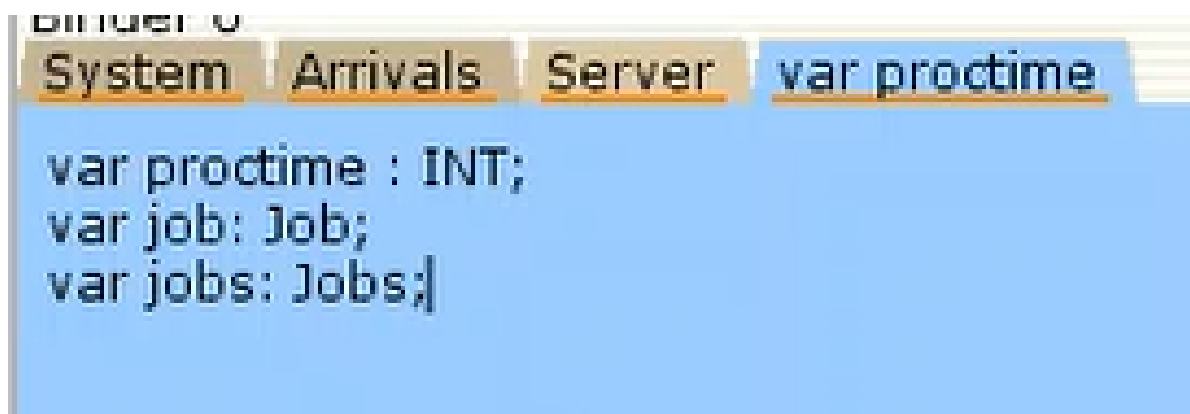


Рис. 2.4: Декларации системы

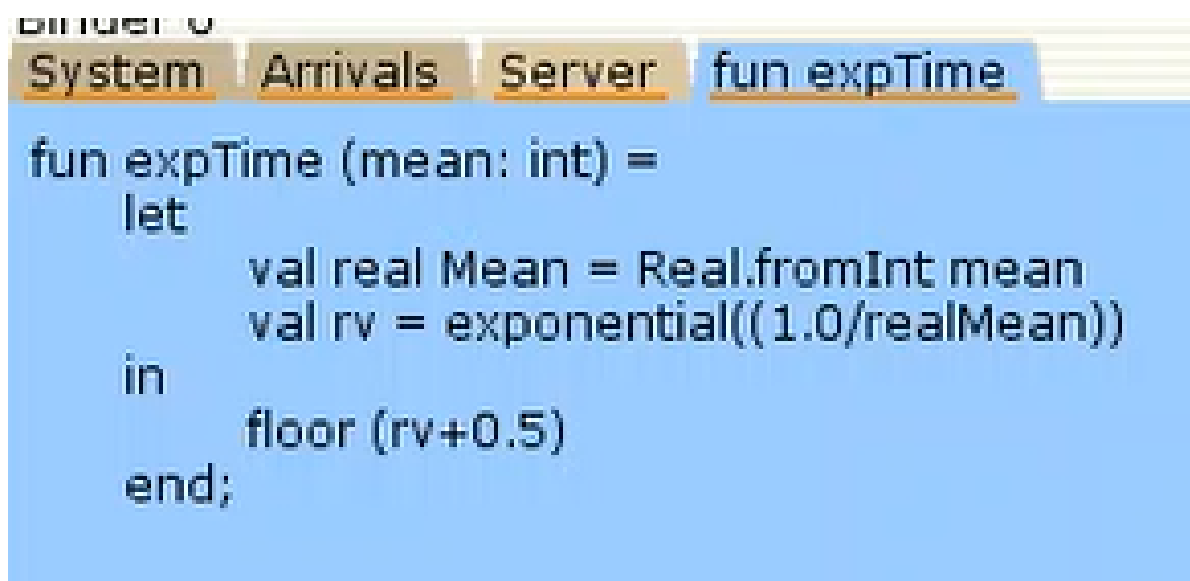
Переменные модели: – proctime – определяет время обработки заявки; – job – определяет тип заявки; – jobs – определяет поступление заявок в очередь(рис.2.5).



```
System Arrivals Server var proctime  
var proctime : INT;  
var job: Job;  
var jobs: Jobs;
```

Рис. 2.5: Декларации системы

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.2.6), (рис.2.7), (рис.2.8), (рис.2.9).



```
System Arrivals Server fun expTime  
fun expTime (mean: int) =  
  let  
    val real Mean = Real.fromInt mean  
    val rv = exponential((1.0/realMean))  
  in  
    floor (rv+0.5)  
  end;
```

Рис. 2.6: Декларации системы

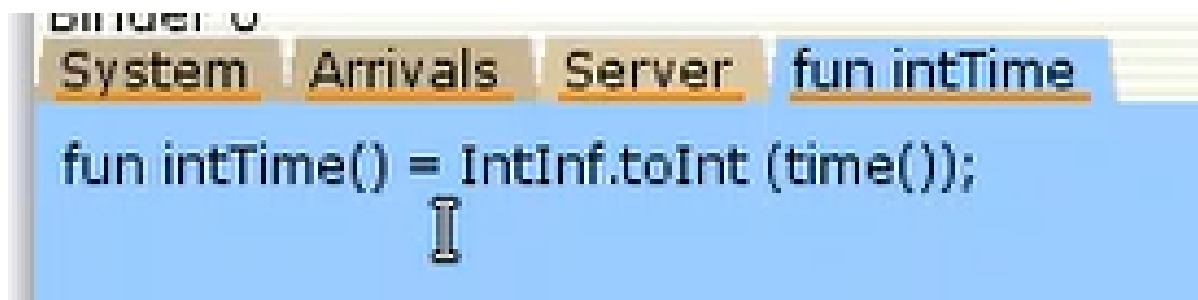


Рис. 2.7: Декларации системы

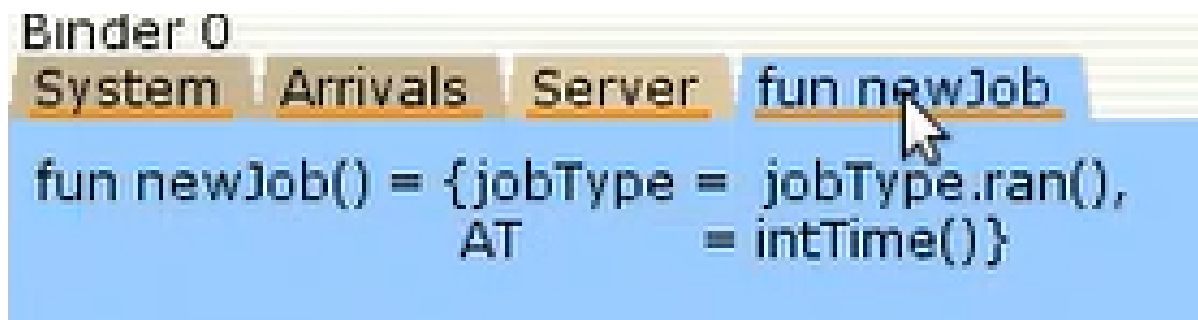


Рис. 2.8: Декларации системы

- ▼ Declarations
 - ▼ Standard declarations
 - ▶ colset BOOL
 - ▶ colset STRING
 - ▼ System
 - ▶ colset UNIT
 - ▶ colset INT
 - ▼ colset Server = with server
 - ▼ colset JobType = with A | B;
 - ▶ colset Job
 - ▼ colset Jobs = list Job;
 - ▶ colset ServerxJob
 - ▶ var proctime
 - ▶ var job
 - ▶ var jobs
 - ▶ fun expTime
 - ▶ fun intTime
 - ▶ fun newJob
 - ▶ Monitors
 - ▼ System
 - Arrivals
 - Server

Рис. 2.9: Декларации системы

Зададим параметры модели на графах сети. – у позиции Queue множество цветов фишек – Jobs; начальная маркировка $1'[]$ определяет, что изначально очередь пуста. – у позиции Completed множество цветов фишек – Job(рис.2.10).

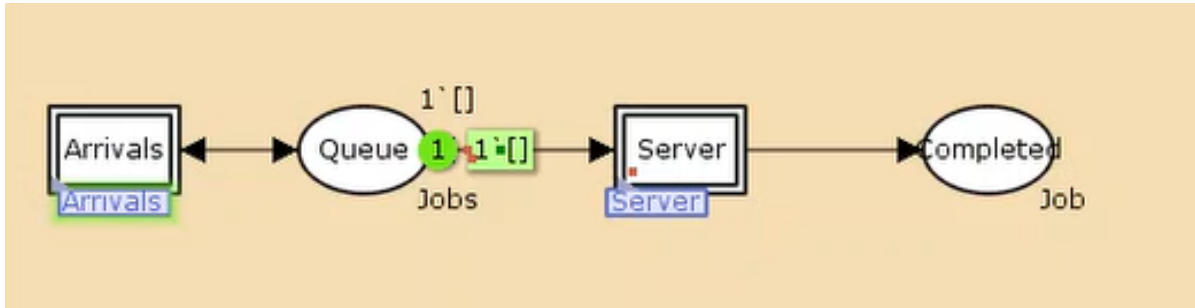


Рис. 2.10: Параметры элементов основного графа системы обработки заявок в очереди

На листе Arrivals: – 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 $jobs^1$ задает поступление заявки в очередь; – на дуге от позиции Queue к переходу Arrive выражение $jobs$ задаёт обратную связь(рис.2.11).

¹job

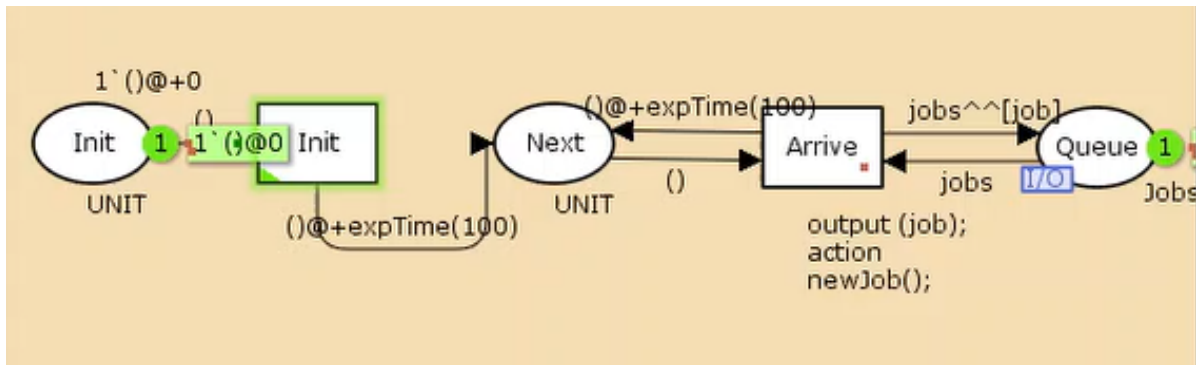


Рис. 2.11: Параметры элементов генератора заявок системы

На листе Server: – у позиции Busy: множество цветов фишек – Server, начальное значение мар-кировки – 1'server@0 определяет, что изначально на сервере нет заявок на обслуживание; – у позиции Idle: множество цветов фишек – ServerxJob; – 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.2.12).

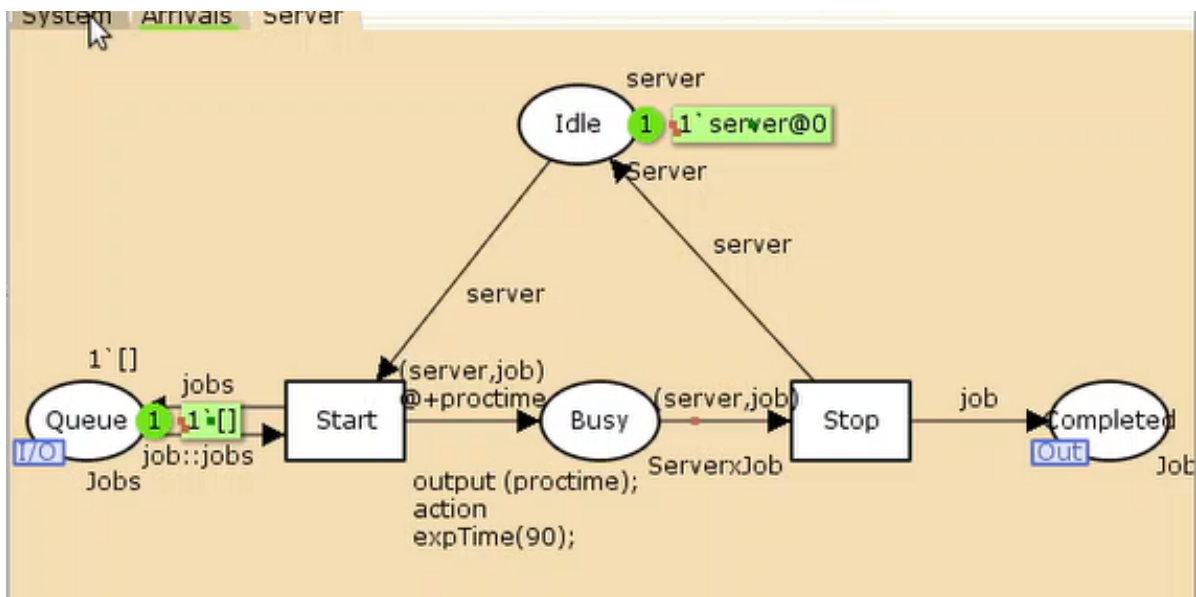


Рис. 2.12: Параметры элементов обработчика заявок системы

Запуск модели(рис.2.13),(рис.2.14).

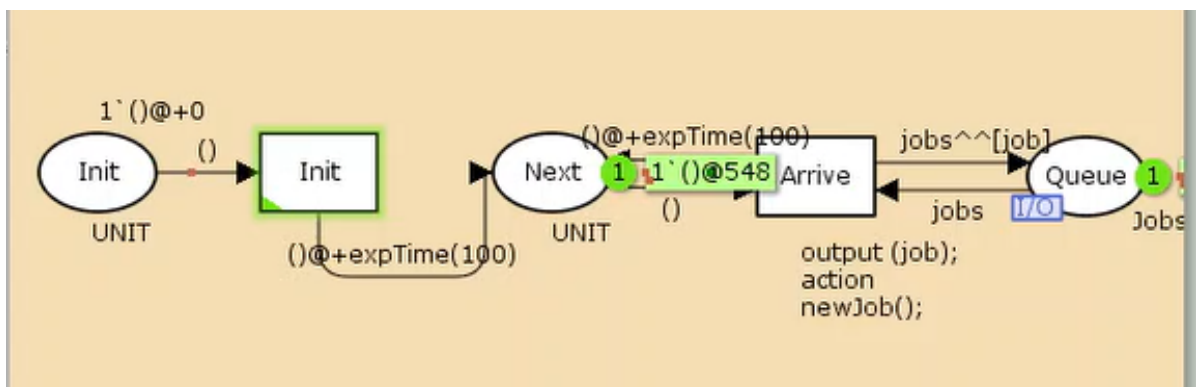


Рис. 2.13: Запуск модели

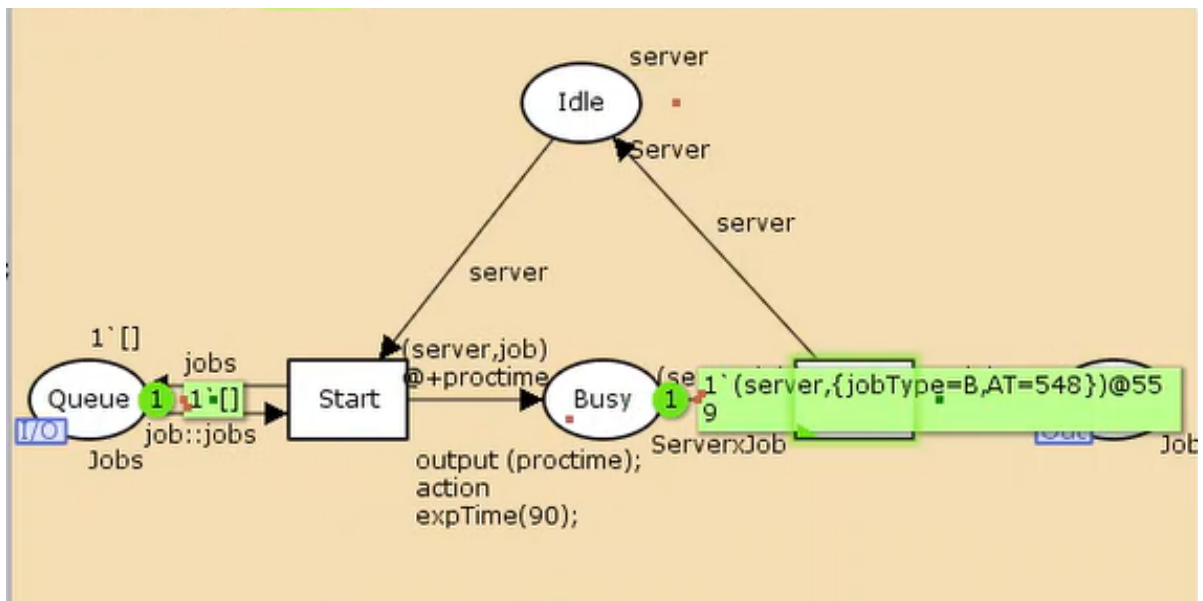


Рис. 2.14: Запуск модели

3 Выводы

Научилась работать с Моделью системы массового обслуживания $M | M | 1$

Список литературы