МГТУ им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

УДК 519.876.5

ПСЕВДОЯЗЫК ОПИСАНИЯ СЦЕПЛЕННЫХ ПРОЦЕССОВ (ПОСП)

Электронное учебное издание

Учебное пособие по дисциплине
«Описание параллельных процессов»

Автор Черненький В.М.

Рекомендуется НМС МГТУ им. Н.Э. Баумана в качестве учебного пособия

МОСКВА
2012

Аннотация

Пособие предназначено для студентов, изучающих проблемы описания совокупности взаимодействующих последовательно-параллельных процессов. Пособие содержит изложение языковой метасистемы, позволяющей на достаточно абстрактом уровне описывать функционирование информационных систем. Полученное описание с помощью макросредств легко отображается на большинство алгоритмических систем имитационного моделирования. В заключении приведены примеры заданий для закрепления материала учащимися.

Введение

В пособии приводится лингвистическая система, позволяющая описать взаимодействие объектов в составе сложной системы в ходе ее функционирования. Лингвистическая система предлагается в форме псевдоязыка, поскольку она ориентирована на создание аппарата описания, а не на машинную реализацию. Машинные реализации должны носить предметно-ориентированный характер. Однако было бы желательно, чтобы эти реализации опирались на некоторую инвариантную к предметной области концепцию описания. В качестве такого универсального аппарата описания функционирования сложных систем предлагается изложенный в главе Псевдоязык Описания Сцепленных Процессов (ПОСП).

Объекты языка, описания

Основные положения

В языке ПОСП разрешается использование букв и символов любых алфавитов, в выражениях могут использоваться знаки и символы любых операций из области математики, лингвистики и пр.

Идентификатор - последовательность букв, цифр и некоторых символов, начинающаяся с буквы. Набор допустимых символов определяется самим пользователем.

Типы идентификаторов:

а) *простой* - последовательность букв и цифр, начинающаяся с буквы;

б) *составной* - последовательность простых идентификаторов, соединенных символом подчеркивания;

в) *стандартный* - фиксированный простой или составной идентификатор. Стандартные идентификаторы могут быть пользовательскими и системными. Перечень пользовательских стандартных идентификаторов определяется самим пользователем.

В арифметических выражениях в качестве стандартных пользовательских идентификаторов часто используются имена функций таких, как SIN, COS, EXP, SQRT и т.д. Перечень таких идентификаторов может быть задан пользователем для каждой конкретной программы.

В качестве системных стандартных идентификаторов в языке ПОСП определены следующие:

ВРЕМЯ - переменная типа скаляр, ее значение - текущее время в модели;

RAND - стандартная функция (оператор). Вычисляет очередное значение псевдослучайной переменной **RAND** в [0,1] с равномерным законом распределения.

ИНИЦИАТОР - переменная типа ссылки, принимающая значение ссылки на локальную среду текущего инициатора.

Запись операторов языка сопровождается служебными словами, которые пишутся русскими буквами и выделяются ***жирным шрифтом*** *(в рукописных текстах – подчеркиваются)*. Запись любого оператора завершается символом “**;**” **.**

Запись чисел соответствует общепринятым в математике правилам.

Любой оператор может быть помечен меткой.

Метка - идентификатор; метка отделяется двоеточием от оператора.

Список - линейная последовательность элементов, разделенных запятой.

Строка символов - любая последовательность символов, кроме кавычек, помещенная в кавычки.

Блоки типа агрегат и контроллер содержат единственный инициатор по умолчанию. Начальное местонахождение инициатора указывается в блоке в разделе "Описание".

Объекты, типы объектов

Объекты языка:

простая переменная; переменная; блок; инициатор.

Каждый объект имеет тип, имя и значение.

Имя объекта есть идентификатор.

Значение объекта есть его содержание:

для простой переменной и переменной - логические, арифметические, текстовые или адресные значения;

для инициатора - адрес локальной среды процесса;

для блока - описание параметров блока и его алгоритм.

Тип простой переменной:

скаляр; ссылка; метка.

Тип переменной:

скаляр; ссылка; метка; вектор; пространство.

Тип блока:

агрегат; процессор; контроллер, параметры.

Тип инициатора: **ссылка.**

Описание типов

Скаляр - одно неделимое значение: число, логическое значение либо строка символов.

Синтаксис:

<список имен простых переменных> **– скаляр(ы)**

Ссылка - простая переменная типа скаляра, значением которой является адрес объекта.

Синтаксис:

<список имен простых переменных> **– ссылка(и)**

Метка - простая переменная типа скаляр, значением которой является адрес помеченного ею оператора.

Синтаксис:

<список имен простых переменных> **– метка(и)**

Вектор – линейно-упорядоченная совокупность скаляров либо векторов (определение рекурсивно).

Синтаксис:

<список имен переменных>**– вектор (**<список элементов описания вектора>)

<элемент описания вектора> **::=** <левая граница> **–**<правая граница> **–**< тип>

<левая граница>,<правая граница> **::=** целое число

< тип> **::= скаляр** | **ссылка** | **метка** | **вектор** | **пространство**

Если левая и правая границы совпадают, то указание правой границы может быть опущено.

Примеры

1. Структура данных, приведенная на рисунке 1, может быть описана как: **вектор** (1-3 - **скаляр**, 4-5 - **ссылка**, 6-7 - **скаляр**)

1	2	3	4	5	6	7
скаляры			ссылки		Скаляры	

Рисунок 1. Структура данных – **вектор**

2. Структура данных (рисунок 2) может быть описана как:

вектор (1**-вектор (** 1**-скаляр ,** 2**-вектор (**1-2**-ссылки) ,** 3**-ссылка),** 2-3**-скаляры,** 4**-вектор (**1**-вектор (**1-3**-скаляры),** 2**-ссылка),** 5**-ссылка,** 6**-вектор (** 1-4**-скаляры))**

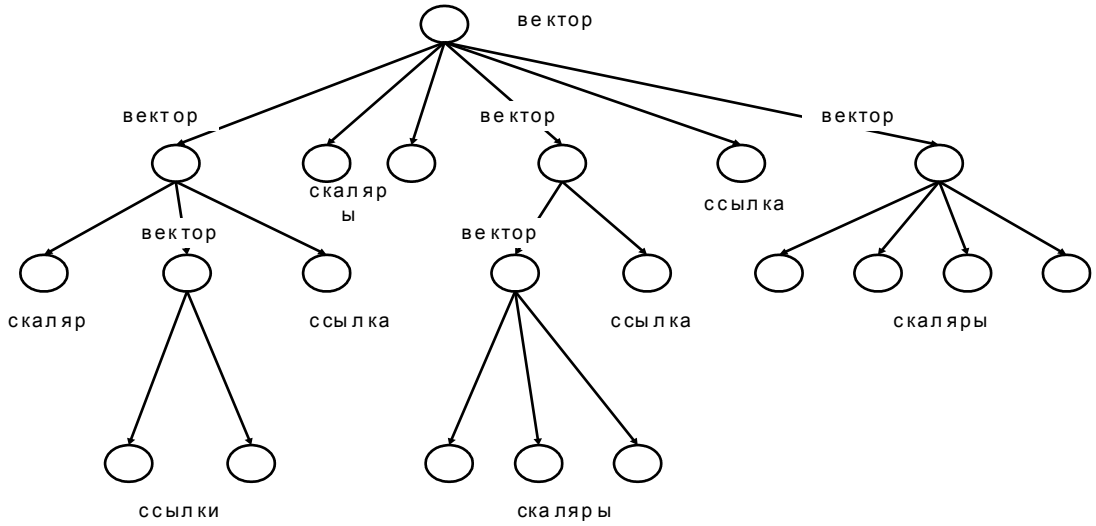

Рисунок 2. Структура данных - **вектор**

Пример. N-мерное пространство скаляров, где N=5, может быть описано, как:

пространство (1-10, 1-3, 1-5, 1-2, 1-4 - скаляры)

Такое пространство содержит 10×3×5×2×4 = 1200 скаляров.

Обращение к конкретному элементу вектора или пространства осуществляется заданием его координат в соответствие с описанием.

Описание блока

Описание блока близко к описанию класса.

Синтаксис:

блок – <тип блока> <имя блока>

описание

<список групп описаний>

все описание;

алгоритм

<программа>

все алгоритм;

все блок.

< тип блока> **::= агрегат** | **процессор** | **контроллер** | **параметры**

Список групп описаний содержит группы описания параметров с указанием: **внешние, внутренние, типы, внешние блоки, метки**, а также необходимые комментарии и договоренности. При описании типов внешних переменных в блоке допустимо после указания типа добавить имя блока, которому они принадлежат. Например: A,B,C - **скаляры блока** ЦЕНТР.

<программа> - последовательность операторов ПОСП.

Операторы ПОСП

Операции над параметрами

Над параметрами могут выполняться любые арифметические и логические операции.

Арифметическое (*логическое*) *выражение* есть последовательность арифметических (логических) операций над параметрами, имеющая целью получить некоторое конкретное числовое (логическое) значение.

Полученное значение присваивается переменной с помощью ***оператора присваивания***:

<список переменных> **:=** <выражение>**;**

Операции над инициатором

Безусловный навигационный оператор

Синтаксис:

направить инициатор на <метка> [**блок** <имя блока>]**;**

Последняя часть может быть опущена, если эта операция происходит в одном и том же блоке.

Условный навигационный оператор

Синтаксис:

если <логическое выражение> **то направить инициатор на** <метка> [**иначе на** <метка>]**;**

Последняя часть может быть опущена, если инициатор продолжает движение по программе.

Векторная форма условного навигационного оператора

Синтаксис:

если

B1 **направить инициатор на** <метка>

:

BN **направить инициатор на** <метка>

[иначе направить на <метка>**];**

где B1… BN - логические выражения.

Оператор задержки инициатора *(оператор условия продвижения инициатора)*

Синтаксис:

ждать <логическое выражение>**;**

Оператор задерживает инициатор до выполнения логического условия. Условие может содержать и переменную **ВРЕМЯ**. Таким образом, этот оператор выполняет функции условного элементарного оператора, совмещая временной, логический и смешанный виды.

Векторная форма оператора задержки инициатора

Синтаксис:

ждать

B1 **направить инициатор на** <метка>

:

BN **направить инициатор на** <метка>**;**

Здесь B1…BN - условные выражения.

Этот оператор совмещает функции двух последовательных операторов: оператора условия продвижения инициатора и векторного навигационного оператора. Его использование позволяет сократить запись в достаточно типичных конструкциях.

Оператор активизации инициатора

Синтаксис:

активизировать инициатор из <имя простой переменной типа ссылки > **в блок** < имя блока> **на метку** <метка>**;**

Оператор активизирует параметр типа ссылки, превращая его в инициатор и направляя на сцепление с помеченным оператором указанного блока.

Оператор пассивизации инициатора

Синтаксис:

пассивизировать инициатор в параметр <имя параметра>**;**

Этот оператор выполняет действие, обратное предыдущему оператору, лишая текущий инициатор свойства инициализации и направляя оставшуюся от него ссылку на локальную среду в указанный параметр ссылочного типа.

Операции над объектами

Оператор создания объекта

Синтаксис:

создать <имя объекта> **типа** <тип>**;**

Оператор создает объект указанного типа и вводит его в программу.

Оператор уничтожения объекта

уничтожить <тип объекта> <имя объекта>**;**

Оператор уничтожает объект с заданным именем и выводит его из программы.

Операции над ссылочными переменными

Оператор присваивания значения ссылке

Синтаксис:

<имя ссылки> **:= ссылка на** [тип объекта] <имя объекта>

Ссылке присваивается адрес объекта.

Оператор разыменования ссылки

Синтаксис:

<имя ссылки> → <тип переменной>[<местонахождение>]

Операция позволяет по ссылке определять значение переменной указанного типа. *Значение переменной* есть результат операции. Местонахождение необходимо указывать лишь для переменных типа **вектор** и **пространство**. Если операция встречается в выражении, то вышеуказанная запись заключается в круглые скобки.

Примеры

1. Вызов значения простой переменной типа скаляр, на которую ссылается ссылка S, имеет вид:

S → **скаляр**

2. Вызов значения 5-го элемента вектора, на который ссылается ссылка S, имеет вид:

S → **вектор(5)**

3. Вызов значения 3-го элемента локальной среды текущего процесса имеет вид:

ИНИЦИАТОР → **вектор**(3)

Напомним, что значение инициатора есть ссылка на локальную среду процесса.

Определение комментария.

Комментарии могут вводиться в любом месте программы и отделяться от операторов двойным слешем (\\ ).

Подпроцессный граф состояний

При составлении алгоритма блока удобно предварительно построить граф состояний блока. В этом графе каждой вершине соответствует некоторый функционально - завершенный подпроцесс, а дуге - путь продвижения инициатора. Назовем такой граф *подпроцессным графом состояний* (ПГС). Степень подробности описания может быть произвольной. Если описание выполняется на уровне обобщенных операторов, то все вершины графа представляются в виде прямоугольников. Условие продвижения инициатора может быть указано как в вершине, так и на дуге. Если же мы хотим задать этот граф на уровне элементарных операторов, то операторы состояний будем помещать в прямоугольники, условные операторы – в овалы, навигационные операторы – в треугольники или ромбы. При использовании векторных форм условных и навигационных операторов условные выражения указываются на дугах. В случае необходимости внесения каких-либо пояснений возможно выделение части графа пунктирным прямоугольником с указанием соответствующего комментария.

Для агрегатов и контроллеров, имеющих по определению один инициатор, будем считать, что этот инициатор априорно помещен в блок и в начальном состоянии находится в вершине, помеченной стрелкой с символом **I**.

Пример 1

На рисунке 3 представлен ПГС блока-агрегата, описывающего файл-сервер в составе локальной вычислительной сети.

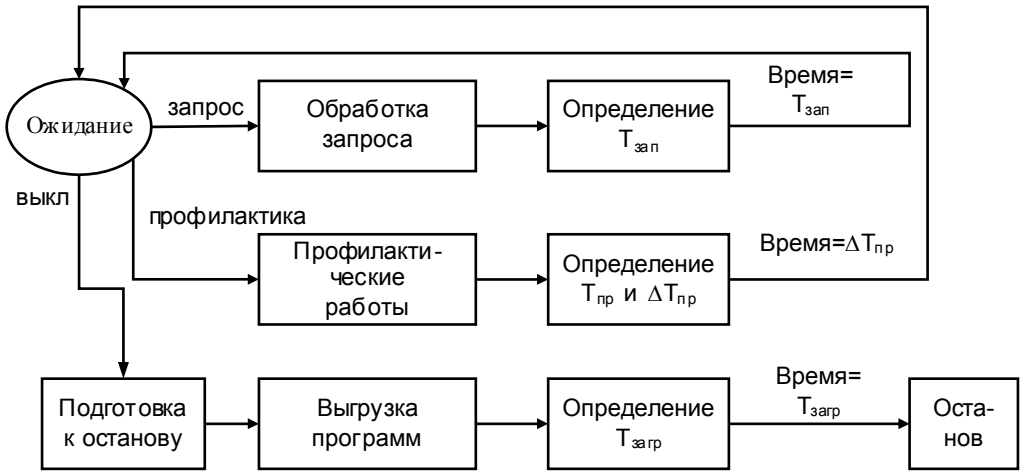

Рисунок 3. ПГС блока-агрегата ФАЙЛ-СЕРВЕР

В начальном положении инициатор размещается в состоянии “ожидание”. Из этого состояния он может быть выведен сигналом “АВОСТ”, началом профилактической работы, запросом из ЛВС на поиск файлов. Дальнейшее функционирование агрегата понятно из ПГС.

Пример 2

Генератор инициаторов

Пусть необходимо создать блок, генерирующий инициаторы, интерпретируемые как документы, и отправляющий их в блок с именем ОБРАБОТКА на оператор с меткой ВХОД. Документы обладают локальной средой, организованной в виде вектора, содержащего 2 скалярных параметра. Первый параметр содержит время появления документа, второй параметр - тип документа, принимающий значение 1 или 2 с равной вероятностью. Значения времени между поступлениями документов распределено равномерно на интервале [20, 50].

ПГС блока приведен на рисунке 4.

Рисунок 4. ПГС блока-агрегата ГЕНЕРАТОР

Программа блока на ПОСП:

блок-агрегат ГЕНЕРАТОР;

описание

ТГЕН- **скаляр;** \\начальное значение равно 0;

ОБРАБОТКА - **внешний блок**;

НАЧАЛО - **метка блока** ОБРАБОТКА;

\\начальное положение инициатора на метке НАЧ;

все описание;

алгоритм

НАЧ**: ждать** ВРЕМЯ = ТГЕН**;**

\\генерация локальной среды;

создать I **типа ссылка**;

создать S **типа вектор (**1-2**-скаляры);**

S(1) **:=** ВРЕМЯ**;**

S(2) **:=** ЦЕЛОЕ (RAND+0.5)+1;

I **:= ссылка на** S**;**

\\выдача инициатора;

активизировать инициатор из I **в блок** ОБРАБОТКА **на метку** ВХОД**;**

ТГЕН **:=** ВРЕМЯ+RAND*30+20**;** \\ вычисление ТГЕН;

направить инициатор на НАЧ**;**

все алгоритм;

все блок.

Пример 3.

Описать процесс функционирования следующей системы:

в компьютерном центре работают круглосуточно 60 компьютеров, интервал их работоспособного состояния задан, после поломки компьютер поступает на ремонт к мастеру, мастер работает с 8 до 17 часов, время ремонта компьютера задано.

Блочная схема модели приведена на рисунке 5.

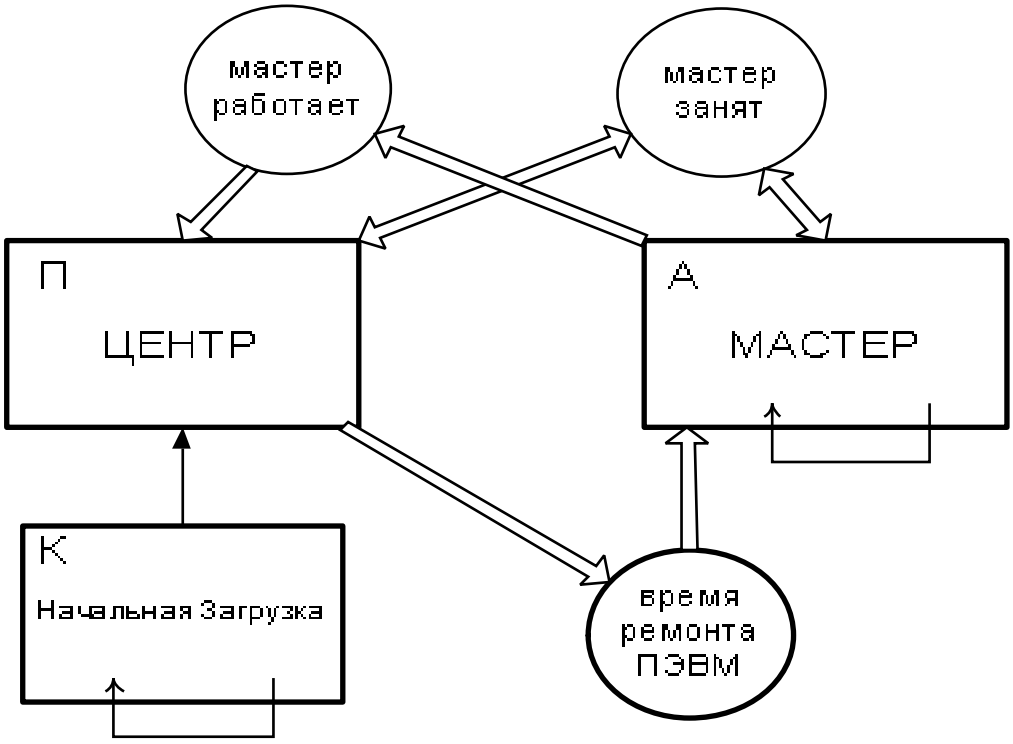

Рисунок 5. Блочная схема модели системы

Блок НАЧАЛЬНАЯ ЗАГРУЗКА генерирует 60 инициаторов в блок-процессор ЦЕНТР. ПГС блока НАЧАЛЬНАЯ ЗАГРУЗКА приведен на рисунке 6.

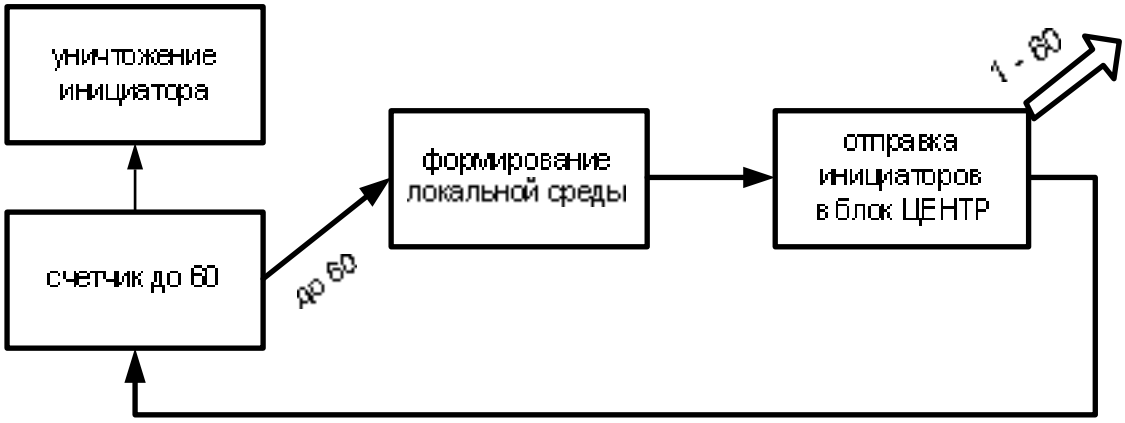

Рисунок 6. ПГС блока НАЧАЛЬНАЯ ЗАГРУЗКА

Описание блока НАЧАЛЬНАЯ ЗАГРУЗКА:

блок-контроллер НАЧАЛЬНАЯ_ЗАГРУЗКА;

описание

N – **скаляр**; \\ начальное значение равно 0;

\\ инициатор на метке НАЧ;

все описание;

алгоритм

НАЧ **:** N **:=** N+1; \\ счетчик;

если N>60 **то направить инициатор на** КОНЕЦ;

создать S **типа ссылка**;

создать W **типа вектор** (1-2-**скаляры**);

W(1) **:=** N;

S **:= ссылка на** W;

активизировать инициатор из S **на** НАЧАЛО **блока** ЦЕНТР;

направить инициатор на НАЧ;

КОНЕЦ **:** **уничтожить** ИНИЦИАТОР;

все алгоритм;

все блок.

Обобщенный ПГС блока Центр приведен на рисунке 7. Он описывает цикл функционирования одного компьютера.

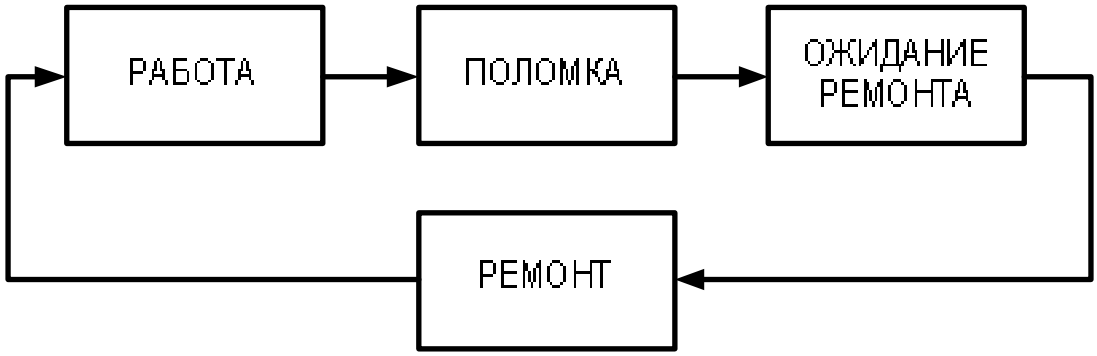

Рисунок 7. Обобщенный ПГС блока ЦЕНТР

ПГС блока ЦЕНТР с комментариями приведен на рисунке 8.

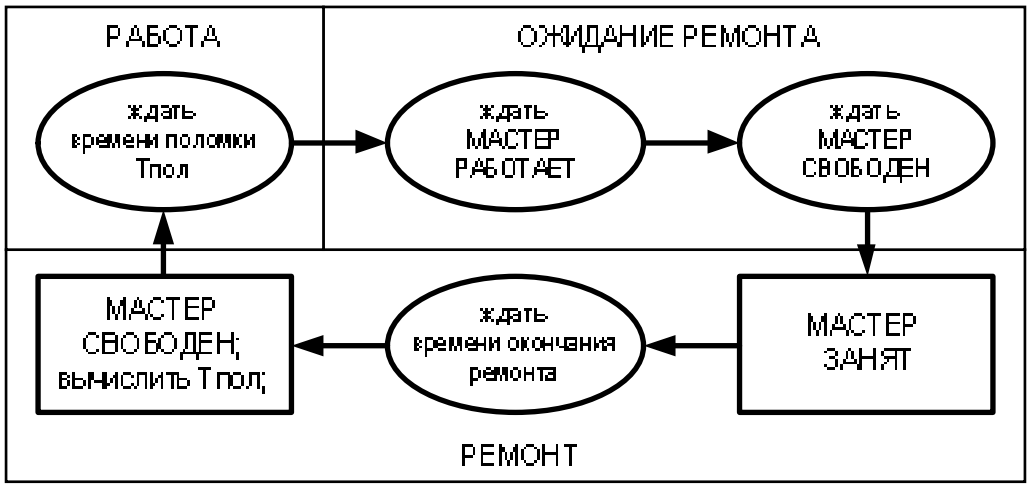

Рисунок 8. ПГС блока ЦЕНТР

Описание блока ЦЕНТР:

блок-процессор ЦЕНТР;

описание

МАСТЕР_РАБОТАЕТ, МАСТЕР_СВОБОДЕН **-скаляры блока** МАСТЕР;

ТРЕМ -скаляр;

\\ FRAB(…)- функция времени работы компьютера;

\\ FREM(…)- функция времени ремонта компьютера;

все описание;

алгоритм

НАЧАЛО: (ИНИЦИАТОР → **вектор**(2)):= ВРЕМЯ+FRAB(…);

ждать ВРЕМЯ = (ИНИЦИАТОР → **вектор**(2));

ждать МАСТЕР_РАБОТАЕТ= “ДА”; \\ мастер на работе;

ждать МАСТЕР_СВОБОДЕН= “ДА”; \\ мастер не занят;

МАСТЕР_СВОБОДЕН := ”НЕТ”;

ТРЕМ := ВРЕМЯ+FREM(…);

ждать ВРЕМЯ = ТРЕМ;

МАСТЕР_СВОБОДЕН := ”ДА”;

направить инициатор на метку НАЧАЛО;

все алгоритм;

все блок.

Блок-агрегат МАСТЕР описывает процесс изменения состояния мастера. ПГС блока МАСТЕР приведен на рисунке 9.

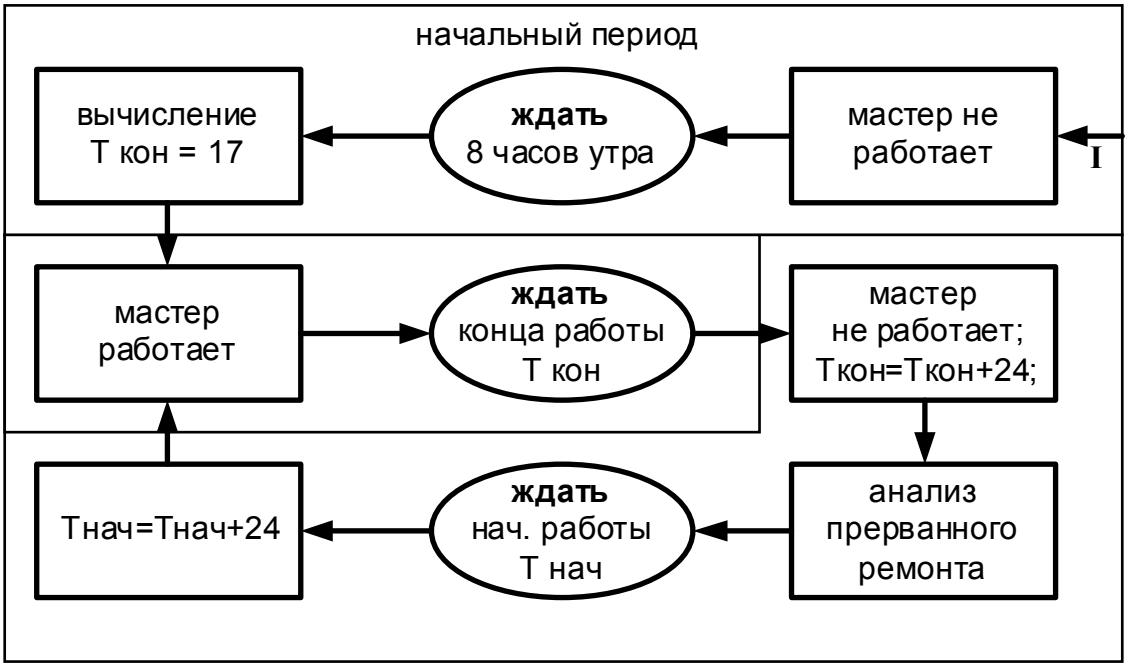

Рисунок 9. ПГС блока МАСТЕР

Описание блока МАСТЕР:

блок-агрегат МАСТЕР;

описание

МАСТЕР_РАБОТАЕТ, МАСТЕР_СВОБОДЕН -**скаляры**;

ТКР, ТНР -**скаляры**;

ТРЕМ- **скаляр блока** ЦЕНТР;

\\начальное положение инициатора на метке НАЧАЛО;

все описание;

алгоритм

\\ начальный сегмент от 0 до 8 часов

НАЧАЛО **:** МАСТЕР_РАБОТАЕТ, МАСТЕР_СВОБОДЕН:=”НЕТ”;

ТКР:=17; ТНР:=32;

ждать ВРЕМЯ=8;

МАСТЕР_РАБОТАЕТ, МАСТЕР_СВОБОДЕН:=”ДА”;

ВКР **:** **ждать** ВРЕМЯ=ТКР;

\\ окончание работы;

ТКР:=ТКР+24;

МАСТЕР_РАБОТАЕТ:=”НЕТ”;

ПРОД1 **:** **ждать** ВРЕМЯ=ТНР;

ТНР:=ТНР+24;

\\ начало работы;

МАСТЕР_РАБОТАЕТ:=”ДА”;

если МАСТЕР_СВОБОДЕН=”ДА” **то направить инициатор на метку** ПРОД2;

\\продление обслуживания отложенного компьютера;

ТРЕМ:=ТРЕМ+15;

ПРОД2**:** **направить инициатор на метку** ВКР;

все алгоритм;

все блок.

Макрорасширения

Предлагаемая лингвистическая система может включать любые расширения и макросы. Поскольку система имеет характер псевдоязыка, то описание макроса может быть выполнено достаточно свободно. Однако, желательно в таком описании максимально близко придерживаться синтаксиса ПОСП.

Принцип построения макросов не ограничен. Например, в основу построения макросов может быть положен ключевой принцип. Причем ключи могут быть оформлены в виде служебных слов.

Служебные слова макроса будем выделять **жирным шрифтом с подчеркиванием**. Макропеременные префиксируются символом ‘&’. Кроме того, вполне допустимо словесное описание макроса, которое, при необходимости, может быть оформлено средствами ПОСП.

Пример 1

Задержка инициатора на время, распределенное по экспоненциальному закону, может быть описана макросом:

задержать инициатор экспоненциально с параметром &A

Его расширение имеет вид:

алгоритм
&B **:=** ВРЕМЯ **-** &A*****LN(RAND**);**
ждать ВРЕМЯ = &B**;**
все алгоритм;

Пример 2

Запись скалярного параметра в однонаправленный список. При создании списка необходимо создать первый элемент &A типа **вектор** (1-2-**ссылки**). Он является головой будущего списка. В список заносится скаляр из параметра &B. Признаком пустого списка является нулевое значение &A(1).

Макрорасширение макрооператора имеет вид:

записать &B **в список** &A;

алгоритм

создать &C **типа вектор** (1-2-**скаляры**).

&C(1) := &B;

если &A(1) ≠ 0 **то направить инициатор на** &M1;

&A(1), &A(2) := **ссылка на вектор** &C;

направить инициатор на &M2;

&M1: (&A(1)→**вектор**(2)) := **ссылка на вектор** &C;

&A(1) := **ссылка на вектор** &C;

&M2:

все алгоритм;

Пример 3

Считывание скалярного параметра из однонаправленного списка в скаляр &B. Начальные условия такие же, как и в примере 2. Признаком пустого списка являются нулевые значения &A(1) и &A(2).

Макрооператор имеет вид:

считать из списка &A **в** &B;

Макрорасширение (предполагается, что &C есть внутренний рабочий параметр типа скаляр):

алгоритм

если &A(2) = 0 **то направить инициатор на** &M2;

&B := (&A(2)→**вектор**(1));

если &A(1) ≠ &A(2) **то направить инициатор на** &M1;

уничтожить (&A(2)→**вектор**)

&A(1), &A(2) := 0;

направить инициатор на &M2;

```
&M1: &C := &A(2);
& A(2) := (&C→вектор(2))
уничтожить (&C→вектор)
&M2:
все алгоритм;
```

Как видно из примеров, использование макрооператоров дает существенную экономию в записи программ. Кроме того, использование макрооператоров позволяет отобразить на ПОСП операторы специализированных языков моделирования таких, как GPSS, SOL, SIMULA и т.д. С помощью макрооператоров можно описать операторы вновь разрабатываемого языка моделирования и провести макетирование новой системы моделирования.

Пример 4

Покажем использование макрооператоров на примере описания функционирования вычислительного комплекса автоматизированной системы управления технологическими процессами (АСУТП) компрессорной станции.

Структура вычислительного комплекса АСУТП приведена на рисунке 10.

Моделируемый комплекс включает:

- центральный вычислительный комплекс (ЦВК), обеспечивающий прием и обработку информации от терминальных устройств;
- терминальные устройства (терминалы), поставляющие технологическую информацию от различных компонент компрессорной станции;
- оконечные устройства;
- каналы связи.

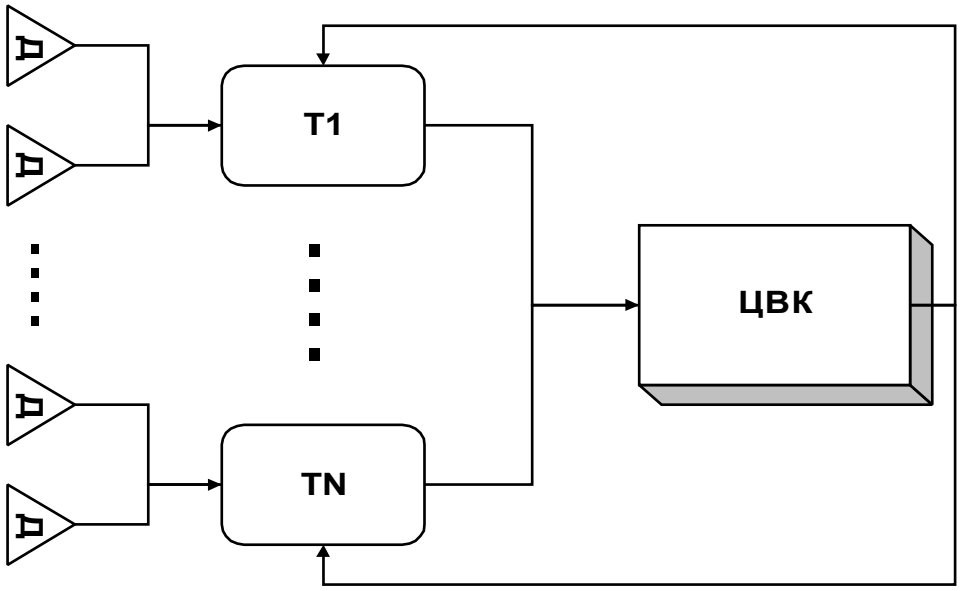

Рисунок 10. Структура вычислительного комплекса

ЦВК решает прикладные задачи по каждому терминалу, необходимые для обеспечения управления технологическими процессами.

Каждый терминал связан с оконечными устройствами, представляющими собой либо датчики, измеряющий технологические параметры некоторой подсистемы компрессорной станции, либо исполнительные механизмы. Для каждого оконечного устройства терминал генерирует запрос к ЦВК для выполнения некоторой обрабатывающей процедуры. Получив ответ, терминал, спустя некоторое время, генерирует новый запрос к ЦВК для следующего оконечного устройства и т.д. ЦВК работает в режиме разделения времени между запросами от терминалов, обрабатывая каждый запрос в течение времени не более, чем заданный временной квант. ЦВК обрабатывает одновременно не более наперед заданного количества запросов, остальные запросы выстраиваются в очередь к ЦВК.

Свернем процессы взаимодействия терминала с оконечными устройствами в один процесс терминала, представляющий собой циклические обращения терминала к ЦВК (рисунок 11). Таким образом, в

системе протекает N параллельных процессов с локальной средой, оформленной в виде вектора и содержащей:

- время появления нового запроса от терминала к серверу - TP;
- рабочий параметр для сохранения времени задержки инициатора в различных процессах – TЗ.

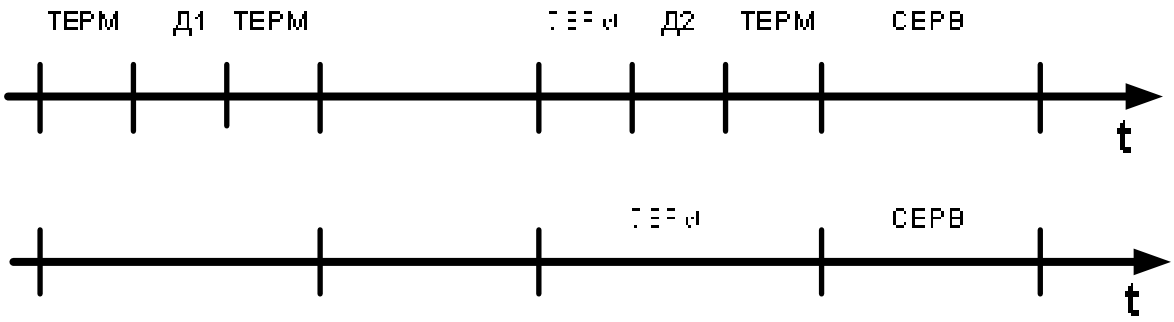

Рисунок 11. Преобразование терминальных процессов

Структура моделируемой системы приведена на рисунке 12. Прерванные задачи помещаются в промежуточный буфер емкостью QS. Задачи, не принятые в ЦВК, помещаются во входной буфер.

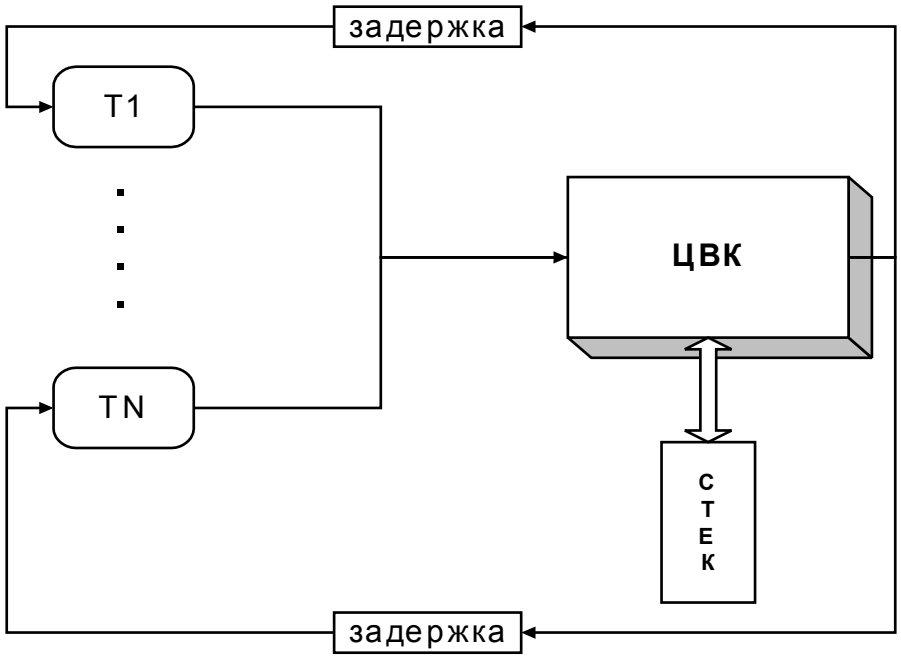

Рисунок 12. Структура модели

Для упрощения описания алгоритма функционирования введем ряд макросов.

Макрос 1. Описание локальной среды процесса:

&W-**векторная локальная среда** &A1, &A2,.. &An;

\\ &A1, &A2,.. &An – список идентификаторов размерностью n элементов.

алгоритм

&W-**вектор** (1-n - **скаляры**);

все алгоритм;

Макрос 2. Обращение к локальной среде процесса:

локальная среда &B;

алгоритм;

\\ k-порядковый номер идентификатора &B в списке описания вектора &W.

(ИНИЦИАТОР → **вектор** (k));

все алгоритм;

Макрос 3. Генерация инициатора:

образовать процесс с векторной локальной средой из &N **элементов по метке** &B **блока** &C;

алгоритм

создать &S **типа ссылки**;

создать &W – **типа вектор** (1-&N-**скаляры**);

&S **:= ссылка на** &W;

активизировать инициатор из &S **на метку** &B **блока** &C;

все алгоритм;

В описании функционирования АСУТП определим локальную среду каждого инициатора, как:

&W – **векторная локальная среда** ТР, ТЗ, Тобраб;

где параметр ТР сохраняет момент появления запроса от терминала,

параметр ТЗ – интервал времени задержки между получением ответа от ЦВК и генерацией нового запроса от терминала,

параметр Тобраб – время , необходимое ЦВК для обработки данного запроса.

Параметр КВАНТ равен шагу квантования ЦВК.

Параметр ΔТП равен затратам времени ЦВК на переключение после окончания времени квантования.

Представим описание в виде двух блоков: агрегата ГЕНЕРАТОР и процессора СЕТЬ. ГЕНЕРАТОР формирует N инициаторов в блок СЕТЬ, запуская, таким образом, N терминальных процессов. ПГС блока ГЕНЕРАТОР идентичен рисунку 4.

Алгоритм блока ГЕНЕРАТОР.

блок-контроллер ГЕНЕРАТОР;
описание
M – **скаляр**; \\ начальное значение равно 0;
N – **скаляр**; \\ начальное значение задается в исходных данных;
\\ инициатор на метке НАЧ;
все описание;
алгоритм
НАЧ **:** M **:=** M+1; \\ счетчик;
если M>N **то направить инициатор на метку** КОНЕЦ;
образовать процесс с векторной локальной средой в 3 **элемента в по метке** НАЧАЛО **блока** СЕТЬ;
направить инициатор на метку НАЧ;
КОНЕЦ **:** **уничтожить** ИНИЦИАТОР;
все алгоритм;
все блок.

ПГС блока – процессора СЕТЬ приведен на рисунке 13. В блок на метку НАЧАЛО в начале моделирования введено N инициаторов из блока ГЕНЕРАТОР. Каждый инициатор имеет локальную среду, структура которой описана выше. В ходе функционирования инициаторы не покидают блок, реализуя замкнутый цикл функционирования системы.

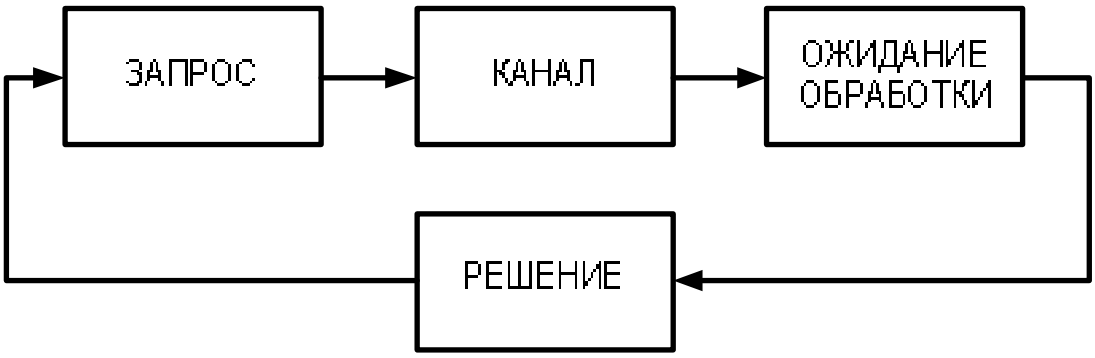

Рисунок 13. ПГС блока СЕТЬ

Описание блока:

блок-процессор СЕТЬ**;**

описание

ТЦВК, Тобр, SERV, Тдообр, Тперекл, ΔТП, КВАНТ **- скаляры;**

ΔТП, КВАНТ **– скаляры;** \\ исходные данные**;**

QS, WS – **скаляры;** \\QS-начальное значение равно 0, WS – объем промежуточного буфера, задается как исходное данное;

Fтерм(…), Fобраб(...) **–** функции времени задержки в терминалах и ЦВК соответственно**;**

\\ инициаторы имеют локальную среду, описанную как

векторная локальная среда ТР, ТЗ, Тобраб;

все описание;

алгоритм

НАЧАЛО: **локальная среда** ТЗ:= Fтерм(...)+ВРЕМЯ**;**

\\ генерация нового запроса;

ждать ВРЕМЯ= **локальная среда** ТЗ;

локальная среда ТР := ВРЕМЯ;

локальная среда Тобраб := Fобраб(...);

\\очередь на вход в ЦВК;

ждать QS<WS;

\\ вход в ЦВК разрешен;

QS := QS+1;

ПРОДОЛЖ**:** **ждать** SERV=’свободен’**;**

SERV := ’занят’**;**

Тобраб := **локальная среда** Тобраб;

если КВАНТ≥ Тобраб **то направить инициатор на** МК1;

ТЦВК := КВАНТ+ВРЕМЯ;

Тдообр := Тобраб-КВАНТ;

направить инициатор на МК2;

МК1: ТЦВК := Тобраб +ВРЕМЯ;

Тдообр := 0;

МК2**:** **ждать** ВРЕМЯ = ТЦВК;

локальная среда Тобраб := Тдообр**;**

Тперекл := ВРЕМЯ+ΔТП;

ждать ВРЕМЯ = Тперекл**;**

SERV := ’свободен’;

если Тдообр>0 **то направить инициатор на** ПРОДОЛЖ;

QS := QS-1;

направить инициатор на НАЧАЛО;

все алгоритм;

все блок.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ ПРОРАБОТКИ

1. Поток задач заданной интенсивности поступает на сервер, сервер тратит на решение одной задачи заданное время, сервер периодически переключается на самотестирование. Использовать блок ГЕНЕРАТОР и блок СЕРВЕР.

2. Поток пакетов заданной интенсивности поступает на канал, канал тратит на передачу одного пакета заданное время, на канале периодически возникают сбои, задано время их устранения. Использовать блок ГЕНЕРАТОР и блок КАНАЛ.

3. На коммутатор поступают три потока пакетов, коммутатор передает их в канал, используя способ временной синхронизации: в моменты времени 1, 4, 7,….передается в канал 1-й поток, в моменты времени 2, 5, 8,….передается в канал 2-й поток, в моменты времени 3, 6, 9,….передается в канал 3-й поток. В один момент времени может быть передан только один пакет. Коммутатор имеет отдельный буфер для каждого потока. Использовать блок ГЕНЕРАТОР и агрегат КОММУТАТОР. Предварительно создать агрегат, выдающий сигналы С1, С2, С3 в соответствующие моменты времени для управления потоками.

4. Система включает CPU и 2 дисковода, в системе одновременно не может находиться более 3-х задач. Задача решается в CPU и затем поступает на тот дисковод, который в данный момент не занят. Каждая задача проходит таких 4 цикла и затем покидает систему. Все времена заданы. Использовать блок ГЕНЕРАТОР и процессор CPU.

5. Система включает 1 дисковод, на входной параметр которого извне поступают заявки на считывание информации. Заявка содержит указание на местоположение информации на дисководе. Дисковод записывает заявку во входную очередь и, если он свободен, приступает к ее выполнению. В соответствии с заявкой дисковод считывает информацию на диске с учетом местоположения и передвижения головки, объем считываемой информации распределен по равномерному закону. После окончания считывания информации дисковод выдает заявку в выходной параметр и считывает очередную заявку из входной очереди. Все времена заданы. Использовать блок ГЕНЕРАТОР и блок ДИСК.

6. ЛВС включает 20 рабочих станций и один сервер, работающие в режиме диалога (вопрос-ответ-вопрос), сервер обрабатывает запрос и отсылает на рабочую станцию ответ, на сервере происходят отказы, на их устранение выделяется время. Все времена заданы. Использовать процессор ЛВС и блок СЕРВЕР.

7. Система включает CPU, 1 дисковод, 1 панель отображения, 1 печатающее устройство. На каждое из устройств извне поступает поток заявок на их использование, между собой потоки независимы (параметры потоков можно задать произвольно). Приоритетность потоков соответствует их порядку в задании. Необходимо описать алгоритм операционной системы в виде агрегата ОС. Все потоки генерируются одним блоком ГЕНЕРАТОР (с вероятностной раскидкой по типам). Все внешние устройства представить только параметрами, связанными с ОС (имитация сигналов прерываний): параметр на прием заявки и параметр об исполнении заявки.

8. Сеть передачи данных включает два узла коммутации и один полудуплексный канал между ними. Узлы обмениваются между собой

пакетами (параметры потоков произвольные) с посылкой подтверждений. Задан размер окна (кредита). Использовать один блок ГЕНЕРАТОР и несколько блоков узлов.

ЛИТЕРАТУРА

1. Черненький В.М. Процессно - ориентированная концепция системного моделирования АСУ: Дисс. док. тех. наук. - М., 2000.- 350 с.