МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)
Кафедра информационных систем и технологий

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__» _____ 2019 г.

СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ

Методические указания к лабораторным работам
для студентов, обучающихся по направлению 02.03.03
Математическое обеспечение и администрирование
информационных систем

Курск 2019

УДК 330.4
Составитель А.И. Катыхин

Рецензент
Кандидат технических наук Ю.А. Халин

Системы реального времени : методические указания к лабораторным работам: / Юго-Зап. гос. ун-т; сост. А.И. Катыхин. Курск, 2019. 55 с. Библиогр.: с. 55.

Рассмотрены вопросы организации и построения систем реального времени. Приведены теоретические положения, практические примеры и задания.

Методические указания предназначены для студентов, обучающихся по направлению 02.03.03 Математическое обеспечение и администрирование информационных систем.

Текст печатается в авторской редакции.

Подписано в печать 27.03.19. Формат 60x84 1/16.
Усл.печ. л. 3,3. Уч.-изд. л. 3,1 . Тираж 260 экз. Заказ.
Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Содержание

АНАЛИЗ ТРЕБОВАНИЙ К СИСТЕМЕ РЕАЛЬНОГО ВРЕМЕНИ

Внешние объекты и характер их взаимодействия с системой являются основой для анализа требований к системе. Как правило, используются две формы описания взаимодействия системы с внешними объектами. Первая форма (контекстная) предполагает описание системы как черного ящика, обменивающегося сообщениями с внешними объектами. Вторая форма (сценарная) предполагает взаимодействие системы с определенным подмножеством объектов при достижении определенной функциональной цели.

Содержание:

1. Контекстные диаграммы

Процесс рассмотрим на примере разработки системы управления лифтом.

Построение контекстных диаграмм на этапе анализа требований к системе позволяет описать объекты, с которыми система должна взаимодействовать, события, происходящие при этом взаимодействии, и сообщения, которыми обмениваются система и внешние объекты.

Применительно к системе лифта можно выделить следующие внешние объекты:

1. Потенциальный пассажир;
2. Пассажир;
3. Обслуживающий персонал.

Потенциальный пассажир формирует команду «Вызов лифта», в ответ на которую (через некоторое время) получает сообщение «Можно войти». Кроме того, потенциальный пассажир наблюдает за положением и направлением перемещения лифта.

Пассажир формирует команду «Перевезти», в ответ на которую (через некоторое время) получает сообщение «Можно выйти». Кроме того, пассажир наблюдает за положением и направлением перемещения лифта и может послать сообщение «Тревога».

Обслуживающий персонал наблюдает за состоянием лифта и принимает сигналы тревоги в случае их появления.

Контекстная диаграмма взаимодействия системы лифта с внешними объектами представлена на рис. 1.

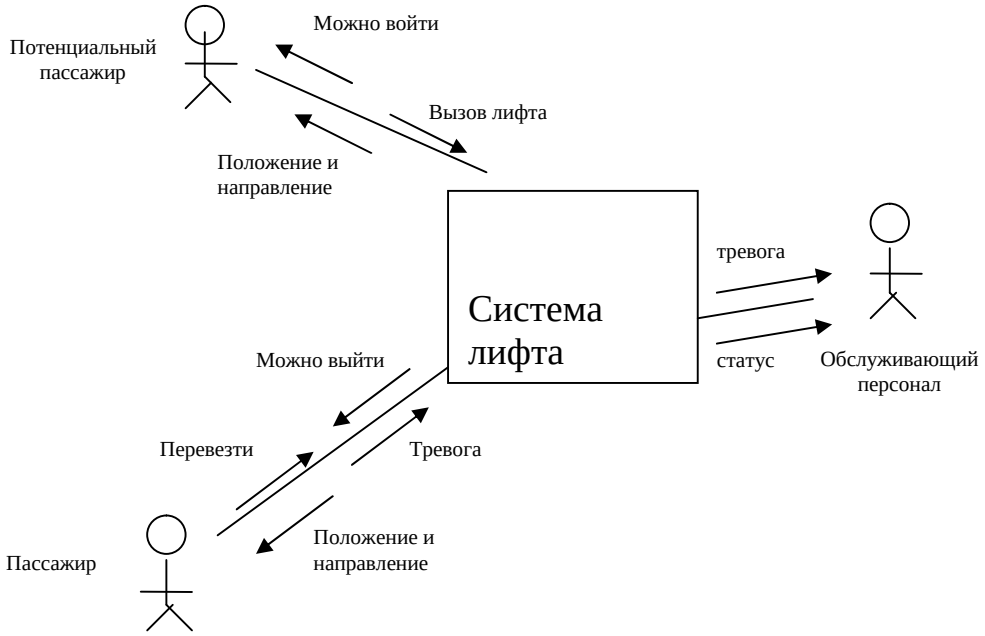


Рис. 1. Контекстная диаграмма взаимодействия системы лифта с внешними объектами

2. Спецификация сообщений и событий

Сообщения являются базовым средством коммуникаций между объектами. На данном этапе определяются наиболее существенные свойства сообщений, к которым относятся следующие свойства:

Содержание данных;

Тип синхронизации (с ожиданием и без ожидания);

Периодичность.

Наиболее важными являются сообщения, переносящие информацию о событиях. Спецификация должна включать перечисление событий и описание отклика на каждое событие.

Кроме того, для СРВ важны временные требования. Поэтому в таблицу описания событий в той или иной форме должны быть включены временные характеристики событий.

Если события периодические, то их временные характеристики специфицируются периодом. Если они эпизодические, то требуется специфицировать средний период и минимальный интервал.

Табл. 1 показывает список внешних событий системы лифта.

Табл. 3.1. Список внешних событий системы лифта

№	Событие	Отклик	Направление	Тип синхронизации	Периодичность	Временные требования
1	Потенциальный пассажир вызывает лифт	Если лифт свободен, то он посылается на этаж. Если лифт занят, то запрос запоминается.	В систему	Без ожидания	Непериодическое	1 с
2	Пассажир запрашивает этаж	Посылка лифта на этаж	В систему	Без ожидания	Непериодическое	1 с
3	Пассажир устанавливает Кнопку «Пуск/Стоп» в положение «Стоп»	Лифт останавливается	В систему	Без ожидания	Непериодическое	1 с
4	Пассажир устанавливает Кнопку «Пуск/Стоп» в положение «Пуск»	Лифт продолжает отрабатывать запросы	В систему	Без ожидания	Непериодическое	1 с
5	Пассажир препятствует закрытию двери	Дверь повторно открывается и запускается таймер закрытия двери	В систему	Без ожидания	Непериодическое	1 с
6	Истекло время нахождения двери в состоянии «Открыта»	Начинается закрытие двери	В систему	Без ожидания	Непериодическое	1 с
7	Пассажир нажимает кнопку «Тревога»	Оповещение обслуживающего персонала	В систему	Без ожидания	Непериодическое	0.5 с
8	Прибытие лифта на этаж	Открытие двери	В систему	Без ожидания	Непериодическое	1 с
9	Начало движения лифта	Ожидание достижения этажа назначения	В систему	Без ожидания	непериодическое	1 с
10	Обрыв троса	Включение механизма блокировки. Генерация сигнала тревоги	Из системы	Без ожидания	Непериодическое	0.25 с
11	Отключение механизма блокировки	Начало отработки запросов лифта, если список запросов не пуст	В систему	Без ожидания	Непериодическое	1 с

3. Выявление вариантов использования системы

Поскольку отдельное внешние событие практически всегда порождает целый поток обмена сообщениями между объектами системы и ее окружения, то после формирования списка внешних событий необходимо более детально рассмотреть отклик системы на каждое событие. Этой цели служат варианты использования системы.

Диаграммы вариантов использования представляют собой одно из средств описания реакции системы на определенные внешние события и являются детализацией контекстных диаграмм, рассмотренных выше.

Для выявления вариантов использования следует пытаться отвечать на вопросы типа:

1.Каковы цели создания системы?

2.Каковы наиболее важные функции системы?

3.Каковы дополнительные функции системы?

4.Как функционировала аналогичная система предыдущего поколения?

Анализ ответов на подобные вопросы позволяет определить:

1. Роли внешних объектов и системы при выполнении системой определенной функции.
2. Последовательности появления событий и сообщений при выполнении системой определенной функции.

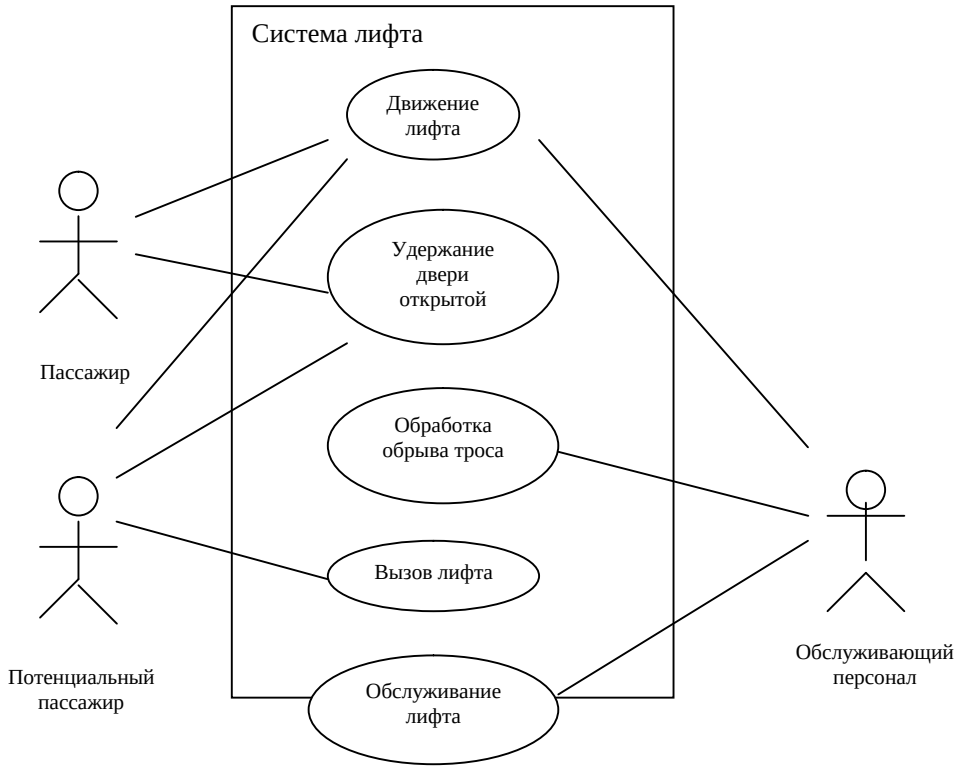


Рис. 2. Диаграмма вариантов использования системы лифта

Представленная диаграмма перечисляет варианты использования системы и показывает объекты, участвующие в каждом из вариантов.

4. Построение сценариев

Дальнейшей детализацией вариантов использования являются сценарии. Любой отдельный вариант использования порождает множество сценариев. Например, вариант «Вызов лифта» может породить следующие сценарии:

1. Лифт уже на этаже;
2. Лифт двигается к этажу, с которого сделан вызов;
3. Лифт имеет запрос, который должен быть обработан.

В начальном этапе анализа в качестве объектов, участвующих в сценариях, выступают система и внешние объекты, определенные в контексте и вариантах использования. При последующем анализе система разбивается на объекты, и процесс построения сценариев может быть продолжен.

Сценарии обеспечивают инструмент для уточнения требований к системе с учетом разнообразия возможный ситуаций, в которых может оказаться система в процессе своего функционирования.

Построение и анализ сценариев – это сложная творческая деятельность.

В табл. 2 показан пример одного из возможных сценариев вызова лифта.

Табл. 2. Пример сценария вызова лифта

Шаг	Событие	Действие
	[Лифт свободен на 1-м этаже]	
1	Запрос лифта на подъем на этаж 6	Лифт начинает подъем на этаж 6
2	[Лифт проходит этаж 2]	
3	Запрос лифта на этаже 2 на спуск	Лифт запоминает запрос
4	Лифт прибыл на этаж 6	
5	Пассажир 1 входит и выбирает этаж 8	Лифт закрывает дверь и начинает движение на этаж 8
6	Лифт прибывает на этаж 8 и открывает дверь	
7	Пассажир 1 выходит	
8	Время таймера закрытия двери истекает	Лифт закрывает дверь и начинает движение на этаж 2
9	Лифт прибывает на этаж 2 и открывает дверь	
10	Пассажир 2 входит и запрашивает этаж 1	Лифт закрывает дверь и начинает движение на этаж 1
11	Лифт прибывает на этаж 1 и открывает дверь	Пассажир 2 выходит
12	Лифт закрывает дверь и остается свободным	

Две описания сценариев используют два варианта диаграмм – последовательные диаграммы и диаграммы сотрудничества.

5. Описание сценариев последовательными диаграммами

Последовательные диаграммы описывают сценарии как последовательности передаваемых и принимаемых сообщений между объектами.

Последовательные диаграммы позволяют специфицировать временные требования к сообщениям и переходы объектов из одних состояний в другие под действием сообщений. Указанные возможности делают последовательные диаграммы особенно ценными при анализе требований к СРВ.

Описание сценария вызова лифта, выполненное в терминах последовательной диаграммы, представлено на рис. 3.

Сообщения могут быть помечены, как это сделано для первых двух сообщений рис. 3. Введение меток позволяет специфицировать временные ограничения, например, запись {b – a <= 1 c} означает, что между событиями b и a должно пройти не более 1 с.

В случае необходимости последовательная диаграмма может быть дополнена состояниями объекта, как это сделано с состояниями «Ожидание» и «Движение» на рис. 3.

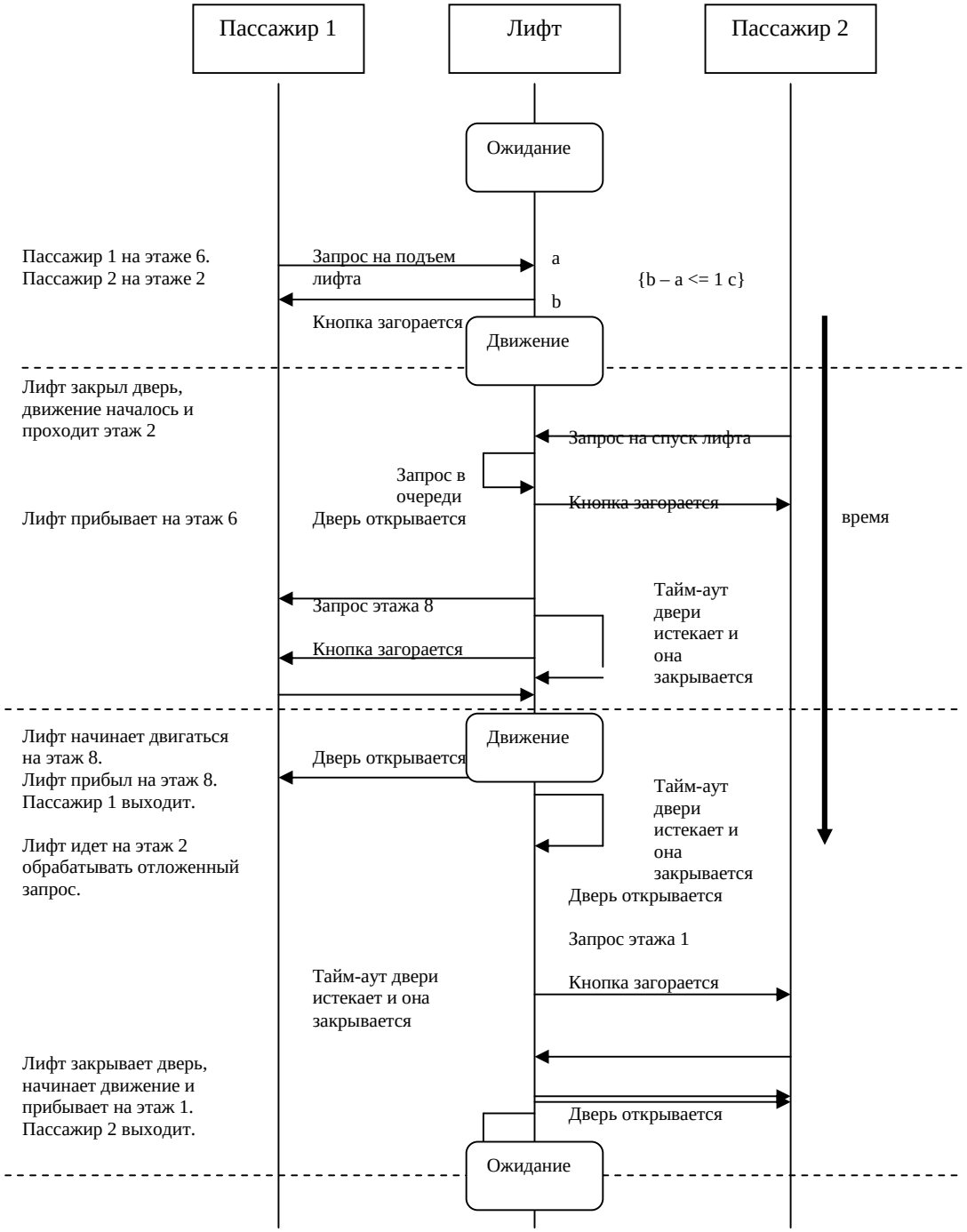


Рис. 3. Последовательная диаграмма, описывающая сценарий вызова лифта

6. Описание сценариев диаграммами сотрудничества

Диаграмма сотрудничества показывает в основном ту же самую информацию, что и последовательная диаграмма, но акцентируется на статической структуре взаимодействующих объектов, а не на очередности сообщений или состояний.

Диаграмма сотрудничества системы лифта, описывающая тот же сценарий, что и рассмотренная выше последовательная диаграмма, представлена на рис. 4.

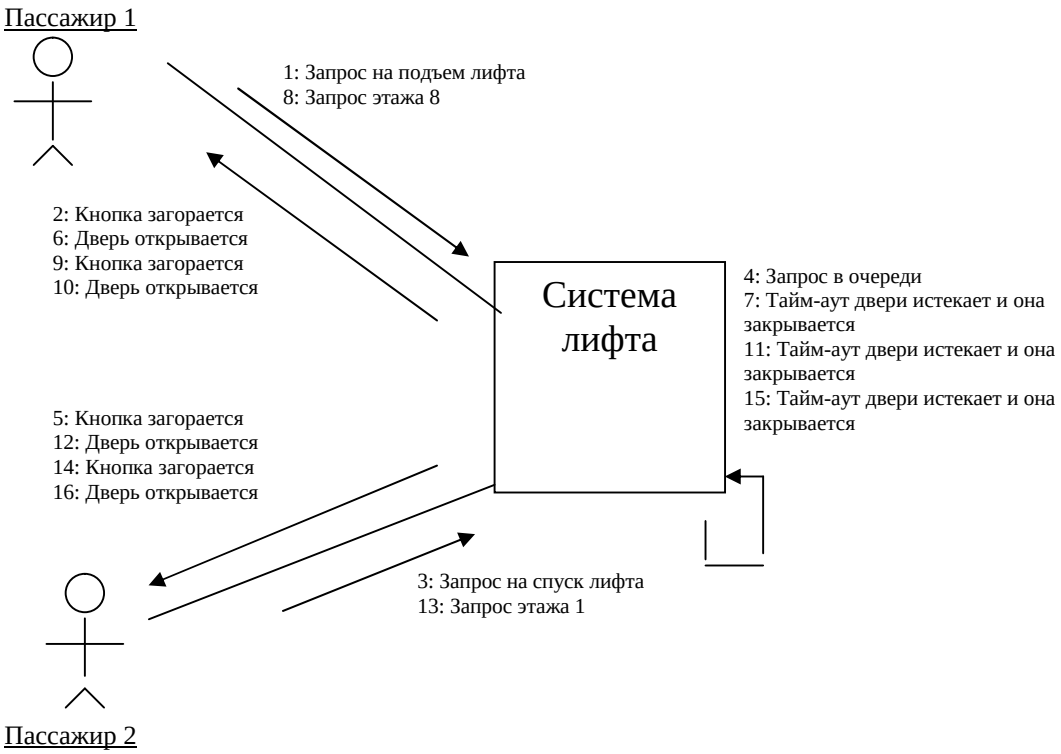


Рис. 4. Диаграмма сотрудничества системы лифта

Выводы

В лабораторной работе показаны этапы анализа системного окружения, с целью понимания и выявления поведения системы в различных обстоятельствах.

Контекстные диаграммы и варианты использования рассматривают различные аспекты этого окружения.

Контекстные диаграммы рассматривают систему как целостный объект и идентифицируют события и сообщения, которыми обменивается система с внешними объектами.

Варианты использования описывают различные аспекты поведения системы безотносительно внутренней структуры системы и сообщений.

Варианты использования декомпозируются на функционально связанные сценарии. Сценарии описываются с использованием последовательных диаграмм и диаграмм сотрудничества.

Построение перечисленного набора диаграмм (контекстной, вариантов использования, последовательных и сотрудничества) завершает внешний обзор системы.

Табл. 3 Задание для выполнения лабораторных работ

№ варианта	Название системы
1	Мобильный телефон
2	Устройство определения местоположения
3	Телевизор

4	Музыкальный центр
5	Стиральная машина
6	Кондиционер
7	Микроволновая печь
8	Фотоаппарат
9	Видеокамера
10	Прибор для измерения давления крови
11	Копировальный аппарат
12	Принтер
13	Сканер
14	Дисплей
15	Телефонный аппарат
16	Факс
17	Квартирная охранная сигнализация
18	Торговый кассовый аппарат
19	Банкомат
20	Автомат для размена денег
21	Турникет для прохождения по магнитной карте

Содержание отчета о лабораторной работе

1. Титульный лист.
2. Цель работы.
3. Таблицы, диаграммы:
    a. контекстная диаграмма взаимодействия;
    b. список внешних событий;
    c. диаграмма вариантов использования;
    d. диаграмма сотрудничества системы.
4. Описание выполнения работы.
5. Выводы.

ОПРЕДЕЛЕНИЕ СТРУКТУРЫ СИСТЕМЫ

В лабораторной работе описывается последовательность определения структуры системы, как совокупности объектов и их отношений.

Содержание лабораторной работы:

1. Основные стратегии определения объектов
2. Определение объектов системы
3. Определение отношений между объектами системы
4. Определение атрибутов объектов
5. Определение классов
6. Выводы

1. Основные стратегии определения объектов

После того, как определено внешнее окружение системы, необходимо внутри самой системы идентифицировать базовые объекты и их взаимоотношения. Существует несколько стратегий, доказавших свою эффективность при разработке СРВ, см. табл. 3.

Табл. 3. Стратегии выделения объектов

№	Стратегия	Краткое описание
1	Подчеркивание существительного	Используется для получения исходного списка объектов. Проектировщик подчеркивает каждое существительное в описании проблемы и оценивает его как потенциальный объект.
2	Идентификация активных объектов	Идентифицируются источники действий, событий и сообщений.
3	Идентификация сервисов (пассивных объектов)	Идентифицируются сущности, которые пассивно обеспечивают сервис, когда тот запрашивается.
4	Идентификация объектов реального мира	К объектам реального мира могут относиться, например, газы, давление, силы, механизмы, химикаты.
5	Идентификация физических устройств	Физические устройства включают в себя датчики, приводы и другие электронные устройства системы, которые выполняют функции наблюдения и управления.
6	Идентификация ключевых концепций	Ключевые концепции могут быть смоделированы как объекты.
7	Идентификация устойчивой информации	Информация, которая является устойчивой в течение значительного интервала времени, может быть объектом или атрибутом объекта.
8	Идентификация визуальных элементов	Элементы пользовательского интерфейса являются объектами.

Нет необходимости использовать все стратегии при проектировании. Стратегии должны находить одно и то же множество объектов. Выбор наиболее подходящей стратегии определяется вкусами проектировщика.

2. Определение объектов системы

Составим список объектов, входящих в систему лифта. Для этого составим описание проблемы и подчеркнем существительные в этом описании.

Контроллер должен управлять лифтом в здании с 10 этажами.

Лифт содержит набор кнопок вызова этажа, каждая из которых соответствует желаемому этажу. Лифт имеет индикатор текущего этажа над дверью. Лифт имеет кнопку тревоги и Кнопка Пуск/Стоп.

Каждый этаж имеет две кнопки для запроса лифта, называемые кнопка вызова лифта. Каждый этаж имеет дверь, организованную так, что две половины двери встречаются в центре, когда закрываются. Когда лифт прибывает на этаж, дверь этажа открывается одновременно с открытием двери лифта.

Этаж имеет датчики давления и оптический датчик для предотвращения закрытия дверей, когда препятствие находится между двумя половинами двери. Если препятствие обнаруживается одним из датчиков, дверь будет открыта.

Дверь будет автоматически закрыта после тайм-аута 5 секунд после открытия двери. Обнаружение препятствия приводит к открытию двери и повторно запускает тайм-аут.

На каждом этаже есть две кнопки вызова лифта, одна вверх и одна вниз. На каждом этаже над дверью есть индикатор, показывающий текущий этаж и текущее направление движения лифта.

Система должна откликаться на вызов лифта, когда лифт свободен или двигается в запрашиваемом направлении.

В других случаях запрос откладывается, пока лифт не отработает предыдущий запрос.

Кнопка вызова лифта при нажатии подсвечивается, чтобы показать, что запрос принят. Нажатие кнопки вызова лифта, когда запрос в том же направлении уже есть, не имеет эффекта.

Когда лифт прибывает, чтобы обработать запрос, подсветка выключается.

Если кнопка вызова лифта нажимается, когда лифт находится на этаже, но планирует двигаться в выбранном направлении, то двери открывается и запускается тайм-аут.

С целью обеспечения безопасности натяжение троса измеряется датчиком натяжения троса. В случае повреждения троса, когда измеренное натяжение падает ниже критического значения, датчик передает информацию в группу обслуживания, а четыре внешних зажима, находящихся в шахте лифта, блокируют движение лифта.

В табл. 4 представлен обработанный перечень подчеркнутых существительных из приведенного описания проблемы.

Табл. 4. Список объектов системы лифта

Лифт	Тайм-аут
Здание	Индикатор положения лифта (на этаже)

Этаж	Индикатор текущего этажа (в лифте)
Кнопки запроса этажа	Подсветка кнопки
Кнопка вызова лифта вверх	Запрос лифта
Кнопка вызова лифта вниз	Шахта
Дверь этажа	Зажимы
Дверь лифта	Повреждение троса
Датчики давления	Натяжение троса
Оптический датчик	Датчик натяжения троса
Препятствие	Критическое натяжение троса
Половины двери	Измеренное натяжение троса
Кнопка Пуск/Стоп	Группа обслуживания
Кнопка тревоги	Контроллер лифта

После определения общего перечня объектов требуется определить перечень самых важных объектов системы.

Для случая с системой управления лифтом к ним следует отнести следующие объекты:

1. Шахта;
2. Зажимы;
3. Датчик натяжения троса;
4. Лифт;
5. Кнопка Пуск/Стоп;
6. Кнопки запроса этажа;
7. Кнопка тревоги;
8. Индикатор этажа;
9. Дверь лифта;
10. Этаж;
11. Индикатор положения лифта;
12. Кнопки вызова лифта;
13. Дверь этажа;
14. Датчики препятствия;
15. Группа обслуживания;
16. Контроллер лифта.

3. Определение отношений между объектами системы

Стратегии определения отношений между объектами в основном базируются на поиске объектов, обменивающихся между собой сообщениями.

В случае с системой управления лифтом результат поиска объектов, обменивающихся сообщениями, представлен в табл. 5.

Табл. 5. Источники и получатели сообщений в системе лифта

Источник сообщения	**Получатель сообщения**	**Характер сообщения**
Контроллер лифта	Лифт	Запрос статуса
Лифт	Группа обслуживания	Статус
Датчик натяжения троса	Группа обслуживания	Тревога
Датчик натяжения троса	Зажимы	Захватить
Группа обслуживания	Зажимы	Отпустить
Кнопка тревоги	Группа обслуживания	Тревога
Кнопка Пуск/стоп	Лифт	Пуск/стоп
Кнопка Пуск/стоп	Группа обслуживания	Пуск/стоп
Кнопка вызова лифта	Контроллер лифта	Запрос лифта
Контроллер лифта	Лифт	Добавить запрос
Кнопка запроса этажа	Контроллеру	Добавить запрос
Датчик препятствий	Дверь	Открыть
Контроллер лифта	Индикатор	Положение

Совокупность объектов и их отношений позволяет построить диаграмму объектов системы лифта. Диаграмма представлена на рис.5.

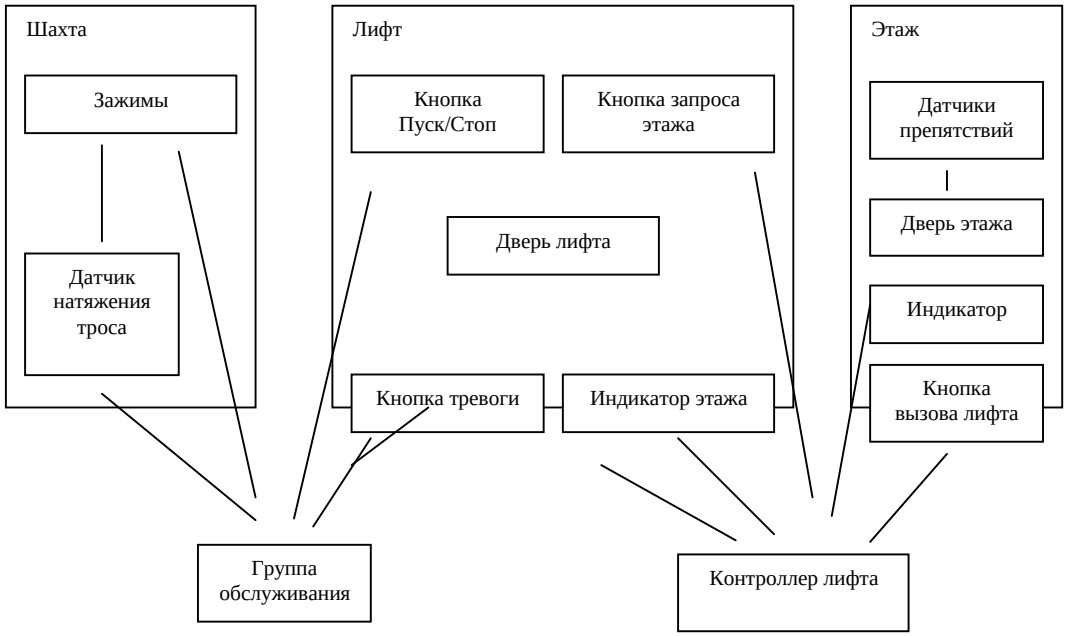


Рис. 5. Диаграмма объектов системы лифта

4. Определение атрибутов объектов

Для определения наиболее важных атрибутов объекта в случаях, когда это не совсем очевидно, рекомендуется ответить на ряд вопросов:

1. Каковы функции объекта?
2. Какая информация необходима объекту для выполнения этих функций?
3. Над какой информацией выполняет объект свои операции?

4. Какой информацией о самом себе должен владеть объект?
5. Не являются ли идентифицированные атрибуты сами объектами?

Попробуем ответить на такие вопросы применительно к лифту.

1. Каковы функции объекта?

Основная функция лифта перевозить пассажиров с одного этажа на другой.

2. Какая информация необходима объекту для выполнения этих функций?

Для этого необходима информация о положении лифта в шахте, о состоянии (остановлен, двигается, занят, свободен), о направлении (если двигается), список запросов.

3. Над какой информацией выполняет объект свои операции?

Лифт выполняет операции над той же самой информацией, которая перечислена в ответе на вопрос 2.

4. Какой информацией о самом себе должен владеть объект?

Лифт должен знать свое положение, состояние (остановлен, двигается, занят, свободен), направление движения.

5. Не являются ли идентифицированные атрибуты сами объектами?

Лифт имеет дверь, которая имеет состояние (открыта, закрыта, открывается, закрывается), связана с дверью этажа, и имеет набор операций. Поэтому наиболее целесообразно определить дверь как объект.

То же самое можно сказать об объекте «Кнопка».

Таким образом, лифт, как объект, обладает следующими атрибутами:

1. Состояние движения (двигается, остановлен);
2. Состояние заполнения (занят, свободен);
3. Положение в шахте;
4. Направление движения (если состояние движения – двигается);
5. Список запросов.

5. Определение классов

Среди объектов, определенных в описании проблемы, обычно существует много идентичных объектов.

В случае лифта, например, существует множество кнопок, но они все идентичны. Каждая кнопка активизируется при отпускании, прием запроса подтверждается подсветкой. Двери лифта и всех этажей также идентичны по своей структуре и поведению.

Объекты, которые идентичны по структуре и поведению, должны быть объединены в классы.

В системе лифта могут быть выделены следующие классы:

1. Шахта;
2. Лифт;
3. Контроллер;
4. Группа обслуживания;
5. Этаж;
6. Индикаторы;
7. Двери.

8. Кнопки;
9. Датчики;
10. Зажим.

Для первой группы перечисленных классов будет создан единственный экземпляр:

1. Шахта;
2. Лифт;
3. Контроллер;
4. Группа обслуживания.

Для второй группы будут отсутствовать классы-наследники:

1. Этаж;
2. Индикаторы;
3. Двери.
4. Зажим.

Для третьей группы будут существовать классы-наследники. Диаграммы классов «Кнопки» и «Датчики» представлены на рис. 6 и рис. 7.

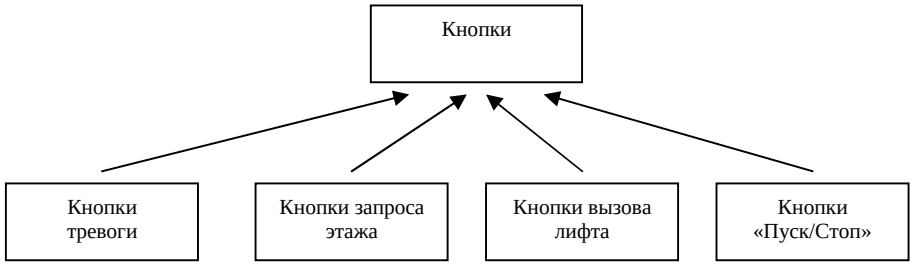


Рис. 6. Диаграмма классов «Кнопки»

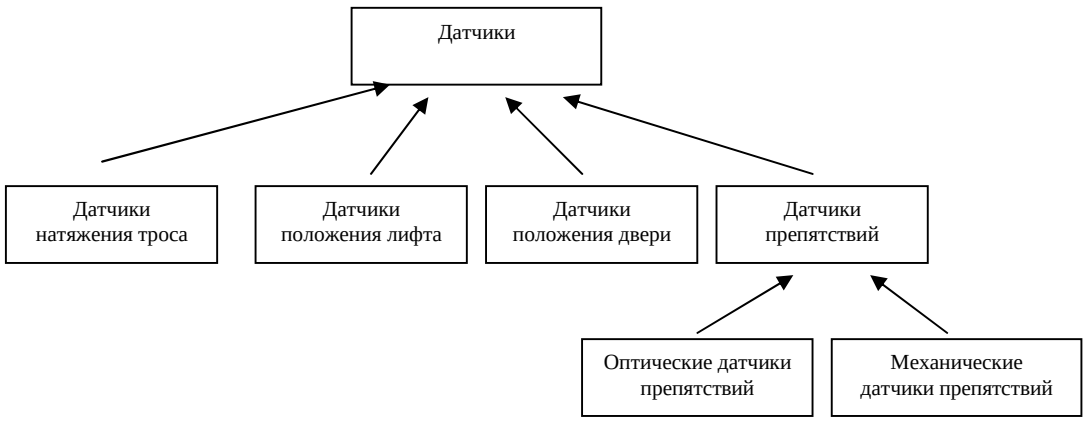


Рис. 7. Диаграмма классов «Датчики»

Первый вариант диаграммы классов системы лифта показан на рис. 8. Диаграмма показывает основные классы и их отношения. Вследствие

ограниченных возможностей графического представления диаграммы классов, типы отношений и их множественность охарактеризованы словесно.

1. Отношение «Шахта – Зажим» - зажим находится в шахте (композиция, 1 - 1);
2. Отношение «Шахта – Датчик» - экземпляр класса «Датчик» – датчик натяжения троса находится в шахте (композиция, 1 - 1);
3. Отношение «Зажим – Контроллер» - зажим управляется контроллером (ассоциация, 1 - 1);
4. Отношение «Контроллер – Группа обслуживания» – контроллер передает информацию группе обслуживания, группа обслуживания управляет системой лифта через контроллер (ассоциация, 1 - 1);
5. Отношение «Датчик – Контроллер» - экземпляры класса «Датчик» передают информацию контроллеру (ассоциация, Много-к-одному);
6. Отношение «Датчик – Лифт» - экземпляры класса «Датчик» находятся в лифте (композиция, Много-к-одному);
7. Отношение «Контроллер – Лифт» - контроллер управляет лифтом (ассоциация, 1 - 1);
8. Отношение «Дверь – Лифт» - экземпляры класса «Дверь» находятся в лифте (композиция, 1 - 1);
9. Отношение «Датчик – Этаж» - экземпляры класса «Датчик» находятся на этаже (композиция, Много-к-одному);
10. Отношение «Дверь – Контроллер» - экземпляры класса «Дверь» управляются контроллером (ассоциация, Много-к-одному);
11. Отношение «Дверь – Этаж» - экземпляр класса «Дверь» находится на этаже (композиция, 1 - 1);
12. Отношение «Кнопка – Лифт» - экземпляры класса «Кнопка» находятся в лифте (композиция, Много-к-одному);
13. Отношение «Кнопка – Этаж» - экземпляры класса «Кнопка» находятся на этаже (композиция, Много-к-одному);
14. Отношение «Индикатор – Лифт» - индикатор положения лифта находится в лифте (композиция, 1 - 1);
15. Отношение «Индикатор – Этаж» - индикатор положения лифта находится на этаже (композиция, 1 - 1);
16. Отношение «Кнопка – Контроллер» - экземпляры класса «Кнопка» передают нажатия контроллеру, а контроллер передает команду «подсветить» кнопкам (ассоциация, Много-к-одному);
17. Отношение «Индикатор - Контроллер» - индикаторы получают информацию от контроллера (ассоциация, Много-к-одному).

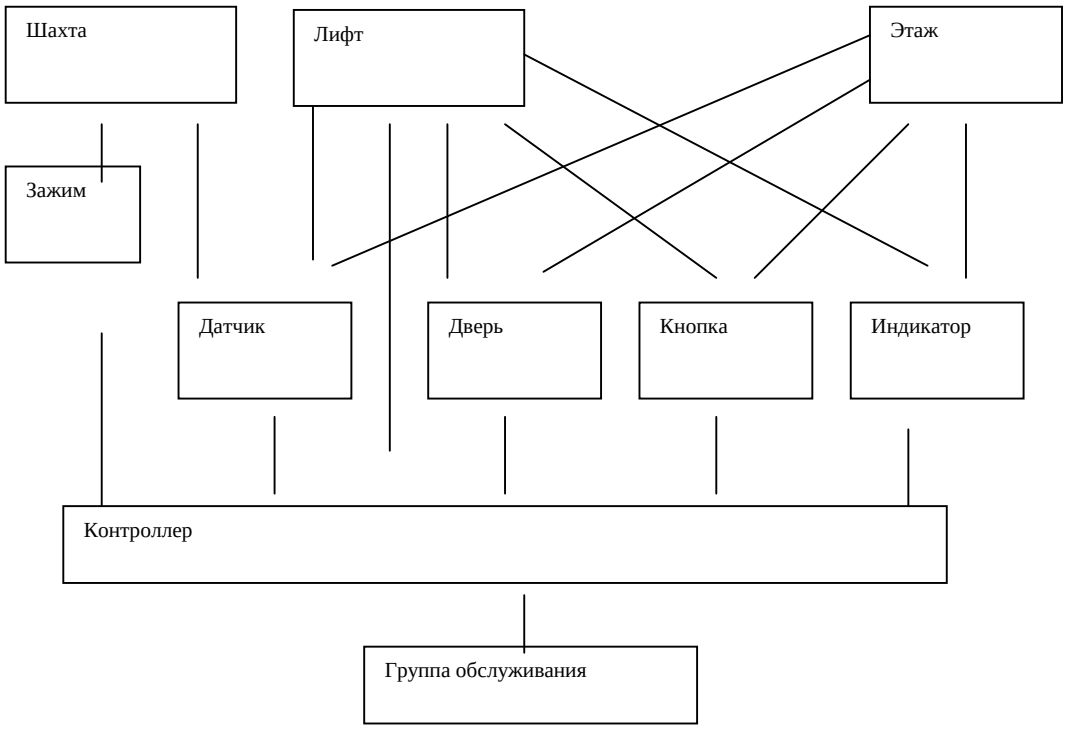


Рис. 8. Первый вариант диаграммы классов лифта

6. Выводы

В лабораторной работе выполнен этап анализа, заключающийся в определении объектов, классов и отношений в системе. Для определения объектов системы была использована стратегия «подчеркивания существительных». Затем объекты, близкие по структуре и поведению были обобщены в классы. А после между классами описаны отношения. Отношения классифицированы и определена их множественность. Итоговым результатом подраздела является диаграмма классов.

В подразделе выполнен только один проход от описания проблемы к построению диаграммы классов. На практике, как правило, необходимо выполнить ряд уточняющих проходов, поскольку после каждого прохода появляются дополнительные сведения и соображения по составу объектов, классов и отношений в системе.

Содержание отчета о лабораторной работе

1. Титульный лист.
2. Цель работы.
3. Таблицы, диаграммы:
    a. определение списка объектов, входящих в систему;
    b. источники и получатели сообщений в системе
    c. диаграмма объектов системы
    d. определение атрибутов и классов системы
    e. построение диаграммы классов
4. Описание выполнения работы.
5. Выводы.

ПОВЕДЕНИЕ СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ

В лабораторной работе необходимо описать поведение системы, выбранной при выполнении первой работы.

Самым важным средством описания поведения системы является описание с помощью диаграмм состояний.

Другим средством является описание сценариев, которые уже были использованы на этапе анализа требований к системе.

Содержание:

1. Построение диаграммы состояний системы
2. Построение диаграмм активности
3. Определение операций классов
4. Выводы

1. Построение диаграммы состояний системы

Лифт представляет собой систему с большим числом возможных состояний. При этом возможна вложенность состояний (подсостояния), параллельность и многоаспектность понятия состояния.

На первом этапе выделим два возможных состояния лифта:

1. Свободен;
2. Занят.

Будем считать, что лифт находится в состоянии «Свободен», если выполняются следующие условия:

1. Лифт не двигается;
2. В лифте отсутствуют пассажиры;
3. Двери лифта закрыты.

Указанные три условия позволяют выделить три подсостояния занятого лифта:

1. Движение – Покой;
2. Отсутствие пассажиров – Наличие пассажиров;
3. Открытые двери – Закрытые двери.

Подсостояния 1 и 2, а также подсостояния 2 и 3 полностью ортогональны, т.е. независимы друг от друга. Движение или Покой могут быть как при Отсутствии, так и при Наличии пассажиров. Открытые или Закрытые двери могут быть как при Отсутствии, так и при Наличии пассажиров. Движение может иметь место только при Закрытых дверях.

Приведенные особенности поведения лифта следующим образом отображаются на диаграмме состояний лифта, рис.9.

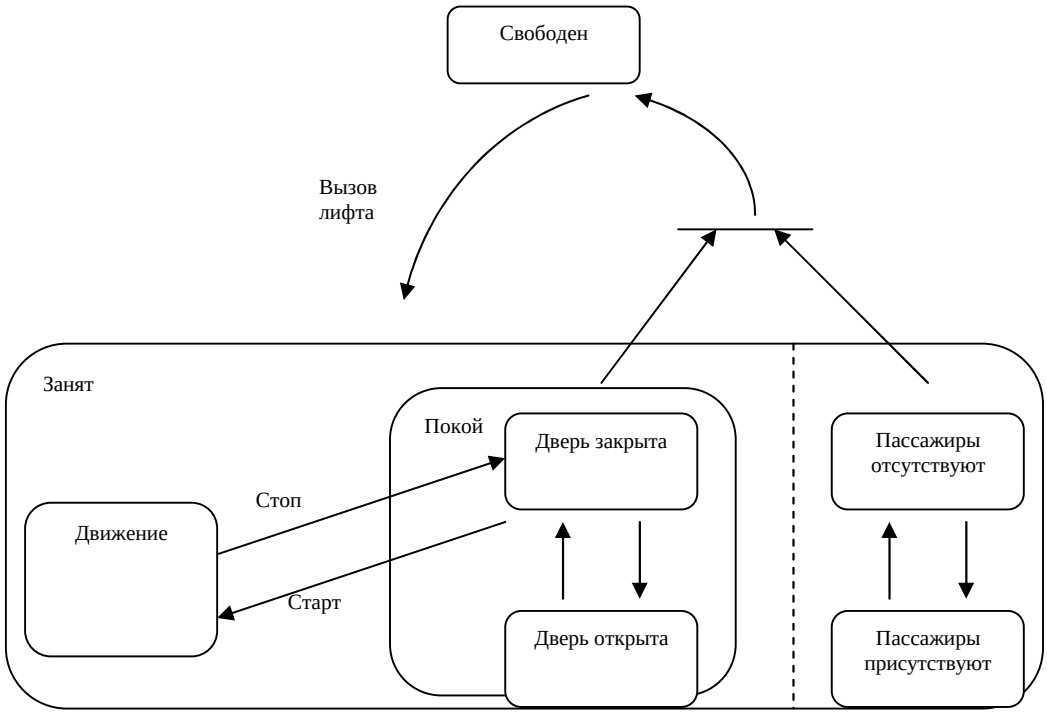


Рис. 9. Диаграмма состояний лифта

2. Построение диаграмм активности

Диаграммы активности приближают нас к построению алгоритмов функционирования системы. Диаграммы активности, так же, как и диаграммы состояний, могут строиться с разной степенью детализации, иметь вложенные состояния и отражать определенные стороны функционирования системы.

Диаграммы активности контроллера лифта при выполнении запроса лифта, когда лифт находится в состоянии «Свободен», представлены на рис. 10.

Ясно, что такой элемент диаграммы, как, например, «Ожидание этажа», сам может представлять собой целую последовательность действий, связанных с определением номера текущего этажа и сравнением текущего этажа с номером этажа, с которого был сделан запрос.

Кроме того, диаграмм, аналогичных рис. 10, может целый ряд, в зависимости от того, в каком состоянии находился лифт, когда делался запрос.

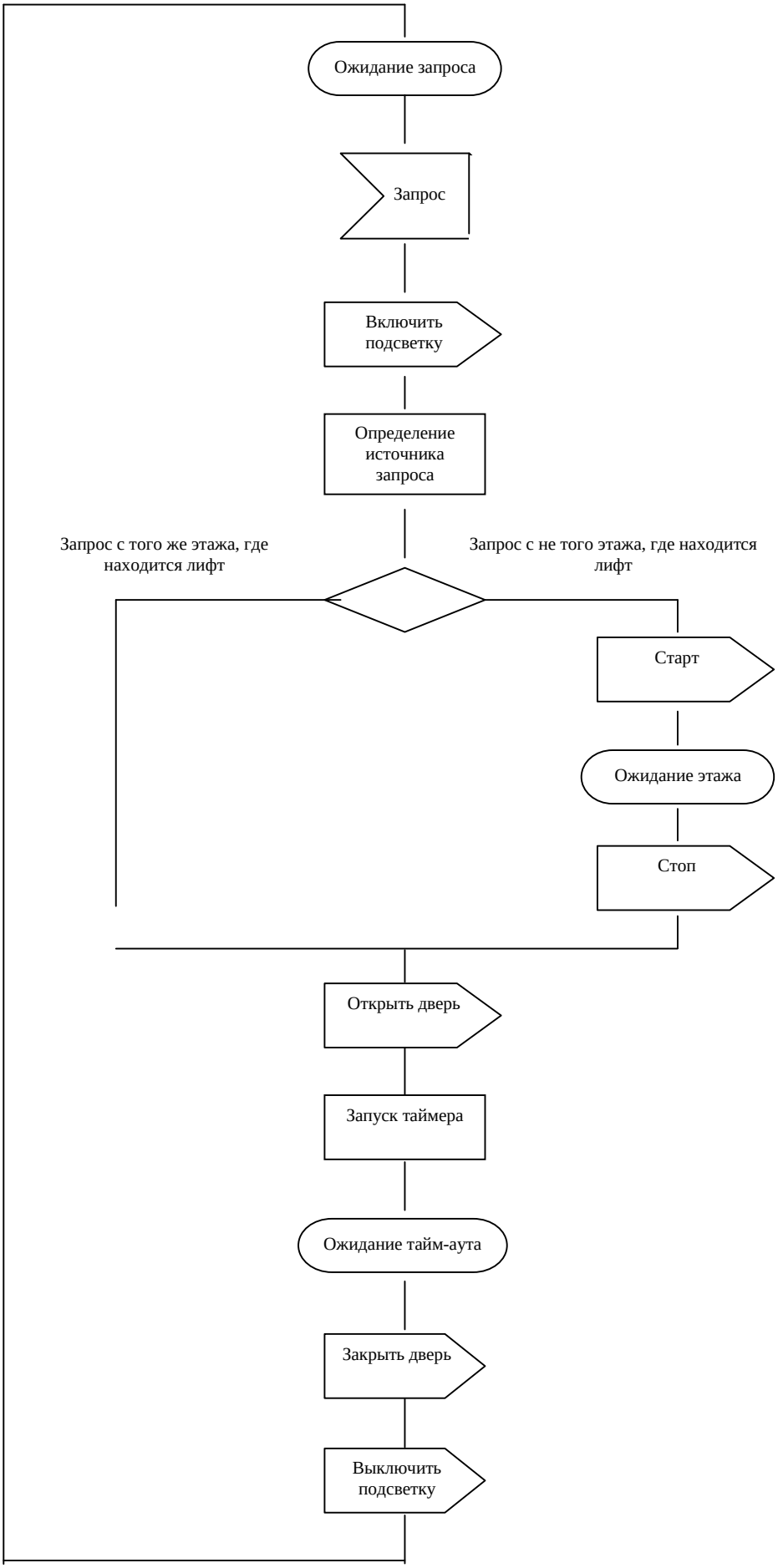


Рис. 10. Диаграмма активности лифта

3. Определение операций классов

Результаты описания поведения системы должны быть использованы для определения операций классов.

Корректное выполнение операций требует специального протокола, который описывает следующие компоненты операций:

1. Предусловия, т.е. условия, которые должны быть выполнены перед тем, как операция будет активизирована;
2. Список передаваемых параметров, их типов, а также тип возвращаемого результата;
3. Постусловия, которые должны быть выполнены, когда операция завершится;
4. Правила взаимодействия потоков - синхронизация.

Существует ряд рекомендаций, которым можно следовать при формировании набора операций класса.

1. Создать ряд ортогональных примитивных интерфейсных операций;
2. Скрыть внутреннюю структуру класса за интерфейсными операциями;
3. Обеспечить ряд непримитивных операций для часто используемых комбинаций операций;
4. Операции внутри класса и иерархии класса должны использовать согласованный ряд типов параметров, где возможно;
5. Родительский класс должен обеспечивать операции, разделяемые классами-наследниками;
6. Операцией должна быть представлена каждая функциональность, выполняемая классом;
7. Все сообщения, направленные объекту, должны быть приняты и обработаны определенным действием;
8. Если необходимо, следует использовать операции Get() и Set() для доступа к атрибутам объекта;
8. Действия, определенные на диаграммах состояний и активности, должны приводить к операциям, определенным на классе, обеспечивающим эти действия;
9. Операции должны проверять свои предусловия.

Определим примерный набор операций классов Лифт, Дверь и Контроллер проектируемой системы.

Количество операций, также как и атрибутов, определяется степенью проработанности класса.

Например. Если различать подсостояния (Пассажиры отсутствуют; Пассажиры присутствуют), то необходимо ввести атрибут, имеющий смысл «Загружен» (“Loaded”). Если количество людей, находящихся в лифте, не различать, то атрибут Loaded может иметь тип Boolean, а если различать (например, чтобы формировать сигнал тревоги при перегрузке), то атрибут Loaded должен иметь тип Integer.

То же самое относится к включению в класс Лифт объектов Дверь, Кнопки.

```
TDirection = (up, down);
TLiftState = (moving, stop);
TPosition = (1..10);

TLift  = object
        State : TLiftState;
        Direction : TDirection;
        Position : TPosition;
        Constructor Create;
        Destructor Done;
        Procedure Start(D : TDirection);
        Procedure Stop;
        Procedure Move;
        Function GetPosition : TPosition;
        Function GetState : TLiftState;
        Function GetDirection : TDirection;
End;
```

Операции Start и Stop носят характер команд, которые контроллер посылает лифту.

Оперции GetPosition, GetState, GetDirection носят характер команд, которые контроллер посылает лифту для чтения его положения, состояния, направления движения.

Операция Move – базовая операция лифта в данной модели.

```
TDoorState = (opened, closed, opening, closing);

TDoor = object
        State : TDoorState;
        Constructor Create;
        Destructor Done;
        Procedure Open;
        Procedure Close;
        Procedure Move;
        Function GetState : TDoorState;
End;
```

Операции Open и Close носят характер команд, которые контроллер посылает двери.

Операция GetState носит характер команды, которую контроллер посылает двери для чтения ее состояния.

Операция Move реализует «процесс» открытия или закрытия двери.

```
TEventType = (Alarm, ButtonPressed);
TButtonType = (LiftButton, FloorButton);
```

```
TEvent = record
        EventType : TEventType;
        ButtonType : TButtonType;
        Message : Integer;
End;

TController = object
        Constructor Create;
        Destructor Done;
        Function GetEvent : TEvent;
        Procedure Dispatcher;
End;
```

Операция Dispatcher носит характер «бесконечного цикла», состоящего из операции GetEvent в начале итерации цикла и, в зависимости от события, посылки соответствующих команд соответствующим объектам.

4. Выводы

В данной лабораторной работе мы исследовали поведение системы путем анализа состояний объектов и анализа действий объектов. Результатом этого анализа является набор операций, выполняемых объектами, построение диаграмм состояний, активности определение набора операций, выполняемых объектами

Содержание отчета о лабораторной работе

1. Титульный лист.
2. Цель работы.
3. Таблицы, диаграммы:
    a. построение диаграмм состояний, активности;
    b. определение набора операций, выполняемых объектами.
4. Описание выполнения работы.
5. Выводы.

ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СИСТЕМЫ

Содержание:

1. Проблемы архитектурного проектирования

В предыдущих лабораторных работах выполнялся анализ системы.

В текущей работе начинается описание этапа проектирования. Проектирование специфицирует решения, которые основаны на результатах анализа.

Проектирование принято делить на этапы в соответствии с масштабами принимаемых решений. Первым этапом является архитектурное проектирование.

Архитектурное проектирование определяет базовые направления организации разрабатываемой системы и детализирует самые большие структуры, такие как:

1. Узлы;
2. Процессоры;
3. Пакеты;
4. Задачи;
5. Протоколы взаимодействия;
6. Пути обеспечения надежности.

Архитектурное проектирование шире по своим масштабам, чем проектирование только программного обеспечения, поскольку учитывает и физическую архитектуру системы.

Во встроенных системах наиболее отчетливо проявляется взаимное влияние физической и программной архитектуры. Программное обеспечение и физическая архитектура должны обязательно подходить друг к другу и эта согласованность должна проявляться уже на архитектурном уровне.

Рациональным способом крупномасштабного проектирования является использование архитектурных образцов, представляющих собой обощение множества полезных решений, найденных в существующих решениях.

2. Выбор архитектурного образца

Опыт архитектурного проектирования информационных систем обобщен в ряде архитектурных образцов.

Существует два аспекта функционирования системы, к которым было применено обобщение с целью построения архитектурных образцов: .аспект управления и аспект обеспечения надежности.

Набор архитектрных образцов, наиболее применимых к систем реального времени, представлен в табл.6.

Табл. 6. Архитектурные образцы, применимые к системам реального времени

Категория	**Название образца**	**Цели**
Управление	Ведущий-ведомый	Координирует деятельность нескольких процессоров.
	Микроядро	Декомпозирует сложные компоненты на множество иерархических абстрактных слоев для облегчения переносимости и повторного использования.
	Прокси	Управляет распределением данных от центрального сервера к множеству удаленных клиентов.
	Брокер	Управляет распределением данных от центрального сервера к множеству клиентов, когда расположение серверов и клиентов неизвестно во время компиляции.
Обеспечение надежности	Однородная избыточность	Повышает надежность в присутствии отдельных отказов.
	Разнородная избыточность	Повышает надежность в присутствии отдельных отказов и систематических ошибок.
	Монитор-привод	Повышает надежность с помощью определенного типа разнородной избыточности.
	«Сторожевая собака»	Механизм обнаружения отказов и сбоев в течение выполнения.
	Безопасное выполнение	Централизованное управление и наблюдение за потенциально сложными и опасными процессами путем комплексирования предыдущих образцов.

Применительно к рассматриваемой системе лифта целесообразным выбором архитектурного образца является образец «Ведущий-Ведомый».

Ведущим (клиентом) будет выступать Контроллер, заявки от которого поступают к Лифту, который будет выступать в качестве Ведомого (сервера).

Указанному образцу будем следовать при формировании облика физической архитектуры и распределении задач по процессорам этой архитектуры.

3. Выявление параллельных задач в системе

Типичная система реального времени имеет множество потоков управления, выполняющихся одновременно.

В системе лифта такие объекты, как Лифт, Контроллер, Дверь, Кнопка, Датчики, могут порождать события, и в этом смысле являются активными объектами системы.

Активные объекты системы в своем составе имеют методы, выполнение которых происходит одновременно. Выявление этих методов и позволяет определить параллельные задачи в системе.

Представление параллельных задач в системе может быть выполнено путем трансформации диаграммы классов (рис. 8) в диаграмму задач.

Диаграмма задач системы лифта представлена на рис. 11.

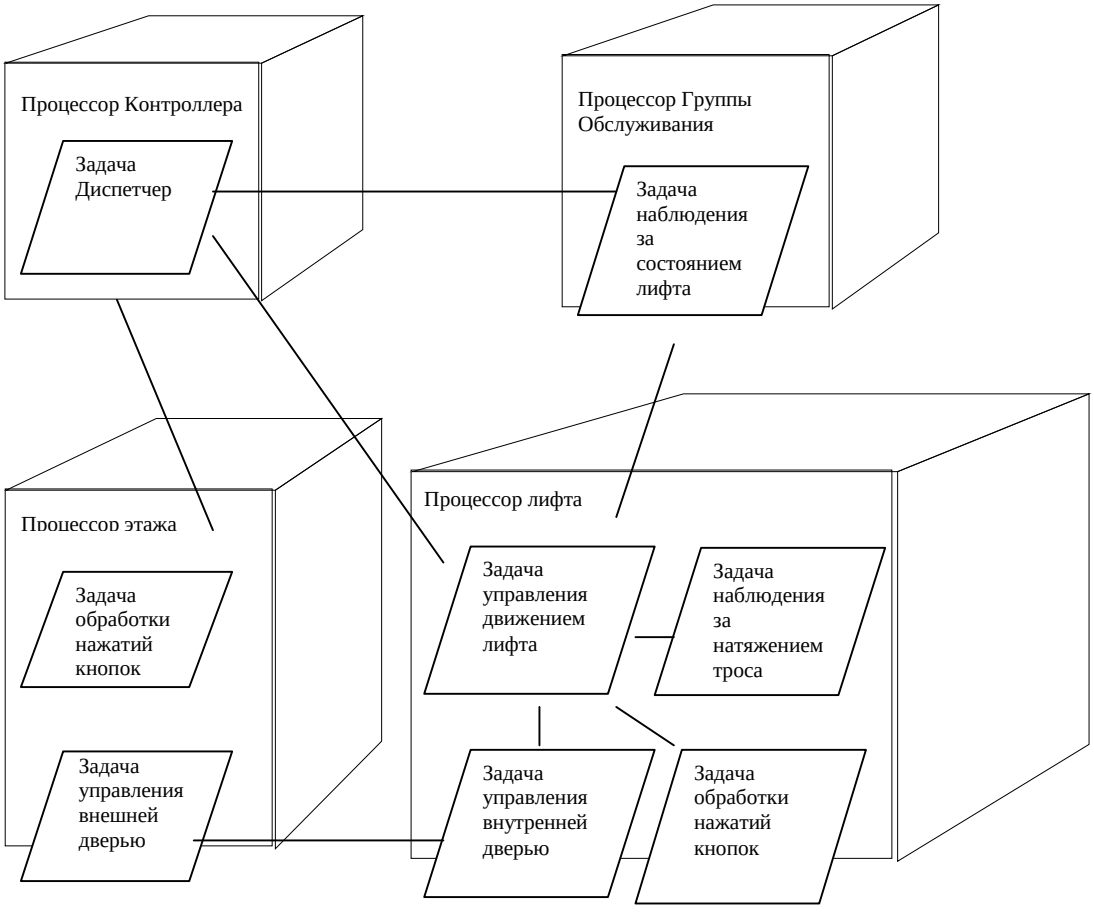


Рис. 11. Диаграмма задач системы лифта.

Перед построением диаграммы задач были определены процессоры. Отметим, что процессоры контроллера, лифта и группы обслуживания – это одиночные процессоры. Количество экземпляров процессора этажа соответствует количеству этажей здания.

Задачи, выполняющиеся внутри каждого процессора, взаимодействуют между собой. Кроме того, взаимодействуют между собой и задачи разных процессоров.

Наличие взаимодействий говорит о том, что необходимо будет выбрать способ обмена сообщениями между задачами.

4. Этап технического проектирования

На предыдущем этапе проектирования – на этапе архитектурного проектирования были приняты решения относительно базовых аппаратных и программных компонентов.

Базовыми аппаратными компонентами выступали процессоры, а программными классы, объекты и задачи.

Итогом предыдущего этапа проектирования явилась диаграмма задач системы.

В данной работе дадим характеристику среднего этапа проектирования, который назовем техническим.

На данном этапе к базовым классам добавляются вспомогательные классы для поддержки функционирования базовых классов.

Существует набор образцов для данного этапа проектирования, некоторые из которых, как наиболее важные для систем реального времени, представлены в табл. 7.

Табл. 7. Образцы технического этапа проектирования

Категория	**Имя образца**	**Цель**
Простые образцы	Наблюдатель	Позволяет группе клиентов разделять сервер.
	Транзакции	Позволяет управлять коммуникациями между объектами.
Повторно используемые образцы	Контейнер	Позволяет агрегировать объекты и эффективно реализовывать ассоциации типа «один ко многим».
	Интерфейс	Обеспечивает отделение типа объекта от его реализации с целью поддержки множественной реализации данного типа.
	Рандеву	Обеспечивает гибкий механизм для межзадачных коммуникаций
Образцы управления состояниями	Состояние	Обеспечивает реализацию машины состояний при редких изменениях состояний.
	Таблица состояний	Обеспечивает эффективные средства поддержки машин с большим числом состояний

Применительно к рассматриваемой системе лифта необходимо воспользоваться образцом «Контейнер» при реализации ассоциаций между Контроллером и Этажами. Так, информацию о положении объекта Лифт в Шахте необходимо рассылать на все этажи и целесообразно это делать одной операцией – итератором, входящей в класс Контейнер.

5. Детальное проектирование

Если на этапе архитектурного проектирования определялись крупномасштабные проектные решения, а на этапе технического проектирования определялись группы взаимодействующих объектов, то на этапе детального проектирования выполняется подробное специфицирование отдельных объектов.

В общем случае на данном этапе определяются решения относительно следующих характеристик объектов:

1. Структуры данных;
2. Реализация ассоциаций;
3. Множество операций, определенных на данных;
4. Видимость данных и операций;
5. Алгоритмы, используемые для реализации этих операций;
6. Возбуждаемые и обрабатываемые исключения.

Структуры данных в объектах обычно просты, поскольку, если это не так, то целесообразно создать отдельный объект, чтобы хранить такие данные. Однако, необходимо определить не только структуры данных, но и диапазоны, точность, предусловия, начальные значения. Это необходимо сделать, поскольку проектируемые системы имеют дело с реальными объектами, которые должны функционировать без ошибок. Так, например, применительно к системе лифта, номер текущего этажа может быть описан типом integer, а можно ограничить этот атрибут диапазоном значений [1..N], где N – количество обслуживаемых этажей.

Реализация ассоциаций позволяет объектам-клиентам активизировать операции объектов-серверов. Простейший способ реализации ассоциации – это прямой вызов операции сервера. Однако, такой способ будет неприемлем, если объекты выполняются в различных потоках, и тем более, на различных процессорах. В этих случаях необходимо использовать средства ОС, такие как очереди сообщений и удаленные вызовы процедур. Применительно к системе лифта (с учетом реализации программных моделей объекта Лифт и объекта Контроллер) метод Лифта, реализующий движение, и метод Контроллера, реализующий прием и диспетчеризацию событий, должны выполняться в разных потоках. Поэтому необходима реализация средств взаимодействия потоков, таких как, например, буфер или очередь сообщений.

Операции, определенные на данных. Архитектурное проектирование специфицирует только базовые операции объектов, доступные из внешнего окружения. Детальное проектирование добавляет еще и внутренние операции, которые появляются в результате функциональной декомпозиции базовых операций.

Видимость данных и операций. Существует несколько принципов управления видимостью данных и операций, а именно:

1. Видимыми надо делать только те данные и операции, которые необходимы объектам-клиентам;
2. Видимые операции должны быть семантически понятными.
3. Атрибуты объекта не должны быть видимы объектам-клиентам.

Алгоритмы, используемые для реализации операций. Существует ряд критериев, по которым можно оценивать качество разрабатываемых алгоритмов. Примерами таких критериев являются следующие критерии:

1. Производительность (средняя и рассчитанная на наихудший случай);
2. Требования памяти;
3. Простоты;
4. Усилия разработки;
5. Повторная используемость;
6. Расширяемость;
7. Надежность.

Перечисленные критерии конфликтны по своей сути. Выбор критериев определяется характером системы. Так, для жестких систем реального времени важным может стать критерий производительности, рассчитанной на наихудший случай. А если СРВ является еще и встроенной системой, то важными могут оказаться и требования к памяти. Для описания алгоритмов может быть использован структурированный английския язык, псевдокод, а также диаграммы активности, рассмотренные выше.

Возбуждаемые и обрабатываемые исключения. Обработка исключений является существенным усложняющим фактором при проектировании алгоритмов. В минимальном варианте реализации алгоритма необходимо специфицировать исключения, которые необходимо обрабатывать, и исключения, которые необходимо возбуждать. Общий подход заключается в том, чтобы обрабатывать те исключения, для обработки которых имеется достаточно информации, а возбуждать все остальные исключения. Серьезность ситуации, которая приводит к исключению, может определить место, в котором это исключение обрабатывается. Так, в случае с лифтом, обрыв троса должен приводить систему в безопасное состояние на уровне системы в целом, а не на уровне отдельных объектов.

6. Реализация системы

Реализация системы должна включать набор объектов (программных моделей реальных объектов), связанных отношением клиент-сервер. Например, для системы управления лифтом такими объектами могут быть Лифт и Контроллер. Объекты, выбираемые для реализации, должны иметь в своем составе методы, выполняемые как потоки.

С добавлением других объектов растет степень проработанности системы, а следовательно, и качество курсового проекта. Например, включение в объект Лифт объекта Дверь позволяет более детально смоделировать работу системы.

Активизация событий в программной модели системы должна выполняться специально выделенными для каждого события клавишами. Например, клавиши “1” – “9” могут имитировать кнопки вызова лифта с соответствующих этажей. Клавиши “q” – “o” могут имитировать кнопки запроса этажа внутри лифта.

Для реализации многопоточной программы в однозадачной операционной системе следует создать ядро управления задачами.

Примерный состав примитивов ядра следующий:

1. Создать ядро;

2. Уничтожить ядро;
3. Начать работу ядра;
4. Завершить работу ядра;
5. Создать задачу;
6. Уничтожить задачу;
7. Приостановить задачу;
8. Возобновить задачу;
9. Создать семафор;
10. Уничтожить семафор;
11. Выполнить операцию P() семафора;
12. Выполнить операцию V() семафора;

Ниже приведены примеры описания объектов “задача”, “семафор” и “ядро”.

```
#include <setjmp.h>
#include <dos.h>

class TProcess : {                  //класс для описания задач
        int stack[1000];            //стек задачи
        jmp_buf ts;                 //средство сохранения контекста задачи
public:
TProcess ( void *p ) {                      //создание задачи из функции с именем p
                struct SREGS segs;
                segread ( &segs );
                ts[0].j_sp    =     FP_OFF(stack) + sizeof stack;
                ts[0].j_ss    =     FP_SEG(stack);
                ts[0].j_flag  =     0x200;
                ts[0].j_cs    =     FP_SEG(p);
                ts[0].j_ip    =     FP_OFF(p);
                ts[0].j_bp    =     ts[0].j_sp;
                ts[0].j_di    =     0;
                ts[0].j_es    =     segs.es;
                ts[0].j_si    =     0;
                ts[0].j_ds    =     segs.ds;
        }//TProcess
};//TProcess

class TSemaphore {
        int count;                  //счетчик семафора
        TCollection *List;          //очередь семафора
public:
        TSemaphore();               //создание семафора
        ~TSemaphore();              //уничтожение семафора
        void P();                   //операция P семафора
        void V();                   //операция V семафора
};//TSemaphore
```

```
#include <tv.h>
#include "proc.h"
#include "sema.h"

class tkernel {
        TCollection *ReadyList;         //очередь готовых задач
        TCollection *KillList;          //очередь уничтожаемых задач
        TCollection *DelayList;         //очередь задержанных задач
        TProcess     *cur;              //дескриптор текущей задачи
        unsigned long count;                    //счетчик “тиков”
        void swt(jmp_buf from,jmp_buf to);  //переключить контекст
public :
        tkernel();                      //создать ядро
        ~tkernel();                     //уничтожить ядро
        void start();                   //начать работу ядра
        void stop();                    //завершить рабору ядра
        void delay(unsigned long tik);  //задержать текущую задачу
        void resume();                          //возобновить задачу
};
```

Представленные фрагменты текстов программы показывают направление, по которому следует идти при реализации ядра.

Выводы

В лабораторной работе рассмотрены вопросы проектирования системы, для которой выполнен анализ требований и определены структура и поведение объектов.

Процесс проектирования включает в себя ряд этапов, которые соответствуют степени детализации принимаемых решений.

Первый этап проектирования имеет дело с самыми общими решениями относительно архитектуры системы. Облегчить выполнение этого этапа призваны архитектурные образцы.

На этапе технического проектирования выделяются группы базовых объектов, взаимодействующих между собой, и добавляются вспомогательные объекты, поддерживающие такое взаимодействие.

На этапе детального проектирования выполняется подробная спецификация отдельных объектов.

Содержание работы:

1. Титульный лист.
2. Цель работы.
3. Таблицы, диаграммы:
    a. выбор архитектурного образца;
    b. диаграмма задач.
4. Описание выполнения работы.
5. Выводы.

РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ СИСТЕМ АВТОМАТИЗАЦИИ НА БАЗЕ ПРОМЫШЛЕННОГО ETHERNET

Необходимо знать:

Принципы программирования для WEB

Организацию интерфейса узлов в сети Ethernet

Состав лабораторного оборудования

Интерфейс между узлами и исполнительными элементами/датчиками

Принципы конфигурирования узлов

Основы программирования для сетевых приложений

Задание на лабораторную работу «Разработка приложений для систем автоматизации на базе промышленного Ethernet»

1. Спроектировать приложение для контроллера
2. Написать программу для IPC Чипа, согласно заданию
3. Загрузить программу на контроллер – IPC Чип
4. Подготовить отчет для сдачи лабораторной работы

Содержание отчета: Описание задания, Предполагаемое решение, Программа с комментариями, Описание основных функций. Описание стандартных функций из библиотек IPC Чип и GGI, применяемых в программе.

Варианты заданий:

1. Разработать утилиту для создания текстовых файлов (telnet)
2. Разработать GGI-программу для создания текстовых файлов
3. Создать программу для чтения и исполнения текстовых файлов с описаниями ввода-вывода в порты
4. Разработать программу для чтения портов. Результаты – в файл. (telnet)
5. Написать GGI-программу для чтения текстовых файлов из браузера
6. Разработать многозадачную программу для работы с портами: задача 1 – запись в порт данных с консоли (telnet); задача 2 – ежесекундный вывод текущего состояния порта (в файл или на консоль)
7. Создать приложение для чтения данных с термометра (шина I2C) и запись в файл
8. Разработать программу управления WEB-камерой из браузера (GGI)
9. Разработать программу управления WEB-камерой по программе из файла
10. Создать приложение - WEB-чат на языке GGI
11. Разработать программу управления сигнализацией в помещении
12. Написать программу – счетчик помещений сайта (GGI), запись в файл статистики

13. Создать программу контроля температуры с применением соединения по telnet
14. Создать программу контроля температуры по программе из файла
15. Создать программу контроля температуры из браузера на языке GGI
16. Построить приложение о принятии решения о вкл/выкл внешнего устройства по голосованию из Интернета с применением языка GGI
17. Разработать программу мониторинга состояния внешних устройств на языке GGI
18. Показать в браузере названия всех запущенных GGI с применением функций языка GGI
19. Разработать WEB-browsing по каталогам чипа (WEB-оболочка для RTOS)
20. Создать программу вывода информации встроенными световыми индикаторами (разработать протокол или использовать азбуку Морзе)
21. Создать программу ввода информации внешними переключателями (разработать протокол или использовать азбуку Морзе)
22. Разработать протокол RTOS для пользователя на базе варианта 20
23. Разработать оболочку RTOS для пользователя на базе варианта 21
24. Разработать программу передачи информации о состоянии контактов IPC@Chip по команде от пользователя (telnet)
25. Разработать программу передачи информации о состоянии контактов IPC@Chip по команде от пользователя (web-интерфейс)
26. Разработать программу синхронизации часов ЧИПА с указанным – ПК (по IP адресу) (из консоли ЧИПА)
27. Разработать комплексное приложение для 2 чипов, обменивающихся информацией о состоянии контактов. (Замыкание контакта на 1 чипе должно сгенерировать включение инверсного контакта на втором чипе) (1-8, 2-7 3-6 5-4)
28. Разработать приложение вывода на экран пользователя состояния контактов в режиме реального времени. (web)
29. Создать программу управления памятью IPC@Chip – (вывод информации о состоянии памяти, сохранение данных, очистка памяти по запросу пользователя) (web)
30. Разработать приложение вывода информации обо всех IPC@Chip найденных в сети (web) (по IP таблице)

Содержание работы:

1. Титульный лист.
2. Цель работы.
3. Описание выполнения работы.
4. Выводы.

РАЗРАБОТКА ПЛАНИРОВЩИКА ПЕРИОДИЧЕСКИХ И СПОРАДИЧЕСКИХ ЗАДАЧ

Описание задания:

Существует структура задачи в виде объекта, обладающего следующими свойствами:

Имя,
Тип,
Период,
Крайний критический срок,
Время запуска,
Время исполнения,
Приоритет.

Необходимо разработать планировщик задач, который на основании таблиц задач строит список исполнения задач и запускает их в соответствии с полученным списком.

В приложении должно существовать от 10 до 100 задач разного типа.

Типы задач определяются пользователем в зависимости от системы управления, определенной заданием. (3-4 типа задач).

Экземпляры задач создаются пользователем и должны сохраняться в конфигурационном файле. Минимальное количество задач должно быть определено по умолчанию.

Каждый тип задач должен в интерфейсе пользователя выводить графическую или текстовую информацию.

В интерфейсе пользователя должны быть реализованы следующие функции управления:

Запустить планировщик
Остановить планировщик
Добавить задачу в таблицу задач
Удалить задачу из списка задач
Изменить свойства задачи
Запустить задачу
Остановить задачу

Планировщик реализуется отдельно для периодических и спорадических задач в соответствии с зданием.

Реализация программы должна осуществляться в любой среде визуального программирования.

Необходимо определить такт функционирования планировщика. Такт его функционирования необходимо разделить на метки. Для каждой метки

устанавливается относительное время запуска (от начала такта планировщика) и список исполняемых задач

Установка списка исполняемых задач осуществляется в зависимости от свойств задач (крайний критический срок исполнения, время запуска, время исполнения)

Для каждой метки должен быть построен собственный список задач, по которому строится общий список и осуществляется исполнение задач.

Постановка задачи

Необходимо разработать приложение для планирования периодических и спорадических задач

Необходимо разработать планировщик периодических задач по алгоритму из задания

Необходимо разработать планировщик спорадических задач по алгоритму из задания

Реализовать интерфейс пользователя для управления задачами

Подготовить отчет для сдачи лабораторной работы:

1. Структура отчета
2. Титульный лист с указанием номера варианта
3. Описание задания.
4. Описание типа «Задача»
5. Описание алгоритма планирования периодических задач
6. Описание алгоритма планирования спорадических задач
7. Описание планировщика заданий.
8. Описание основных функций приложения.

Варианты заданий (в скобках примерные варианты типов задач):

1. Система измерения температуры (измерение температуры, Перевод из мВольт в градусы Цельсия, Контроль температуры)
   a. Планировщик периодических задач - RM
   b. Планировщик спорадических задач - Деферабельный сервер
2. Система измерения освещенности и выдачи управляющих сигналов (Измерение уровня освещенности, выдача сигнала на включение, выдача сигнализации)
   a. Планировщик периодических задач - LSTF
   b. Планировщик спорадических задач - Планирование как задачи фона (динамической)
3. Система измерения давления (Измерение, Контроль, Выдача в 3 цикла 2- предварительное, 3 - настоящее)
   a. Планировщик периодических задач - EDF
   b. Планировщик спорадических задач - Обмен приоритетом

4. Система работы с данными (заполнение массива данных, выполнение однотипной операции над данными, выдача значений)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Спорадический сервер

5. Система перекрестного контроля параметра (Измерение параметра группы 1, Измерение параметра группы 2, сравнение параметров и выдача результата)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Выбор

6. Система изменения состояния объекта (Измерение температуры, построение объекта «квадрат», заполнение объекта в зависимости от температуры соответствующим цветом от синего до красного)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

7. Система контроля климата (включение вентилятора, включение обогревателя, в зависимости от температуры - измеренной)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

8. Система управления светофорным объектом (изменение цвета, переключение режима)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Спорадический сервер

9. Система управления освещением (Включение/Выключение, регулирование яркости, переключение режимов)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Обмен приоритетом

10. Система управления АЗС (вкл/выкл насосов, выдать режим обслуживания, изменить режим обслуживания)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Выбор

11. Система измерения вибраций на объекте (контроль по модели, анализ опасных и повышенных вибраций, сигнализация вибраций)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач — Планирование как задачи фона (динамической)

12. Система управления гирляндой (смена режимов, переключение ламп, смена режимов)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Деферабельный сервер

13. Система управления рекламным щитом (управление подсветкой, изменение рекламы, выдача сообщений)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Обмен приоритетом

14. Система управления питанием на объекте (переключение по 2 источникам и 2 фазам, анализ загруженности, анализ качества)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Выбор

15. Система управления охранной сигнализацией (Анализ состояния датчиков - Движения и Открытия, выдача сигнализации на сирены и пульт)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

16. Система управления камерами наблюдения (задать угол поворота, задать время поворота, вывести состояние)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

17. Система управления автомобилем (управление расходом топлива, анализ скорости движения, анализ степени нажатия на педаль газа, анализ оборотов)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Спорадический сервер

18. Система управления климатом (анализ температуры, анализ влажности, включение/выключение вентилятора и нагревателя)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Деферабельный сервер

19. Система управление выводом и вводом данных (чтение данных с клавиатуры, выдача сигналов на экран, чтение по группам (цифры, буквы), формирование слов)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Спорадический сервер

20. Система управления входом в помещение (выдача состояния дверей, открытие, закрытие, световая сигнализация)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Обмен приоритетом

21. Система измерения освещением (через диммер установка значения 1..12, выдача информации на другие объекты, поддержка освещенности на заданном уровне)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

22. Система управления переключения между камерами наблюдения (переключение режима, включение/выключение, сохранить данные)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

23. Система управления конвейером (Движение, Смена операций, Запуск)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Выбор

24. Система изменения освещенности объекта (Изменение степени освещенности, скорость изменения цвета, вывод состояния)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

25. Система управления движение ж/д транспорта на станции (переключение семафора, перевод стрелок, вывод состояния пути)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Обмен приоритетом

26. Система пожарной сигнализации (анализ датчиков дыма, анализ датчиков огня, выдача световой индикации, вывод состояния на пульт)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Деферабельный сервер

27. Система контроля прохода судов через шлюз (поиск свободного пути, выбор направления суда, вывод информации о судах)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

28. Система формирования «зеленой полосы» на автомобильной дороге по светофорным объектам (Вывод сигнала светофора, оценка потока, вывод информации)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Выбор

29. Система учета электрической энергии (измерение, выявление пиковой нагрузки, выдача сигнализации о пиках)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Обмен приоритетом

30. Система контроля трубопровода нефти (измерение потока в сечениях, включение насосов, выдача информации о рассогласовании потоков)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Спорадический сервер

Содержание работы:

1. Титульный лист.
2. Цель работы.
3. Описание выполнения работы.
4. Выводы.

РАСЧЕТ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ НА ВОЗМОЖНОСТЬ ФУНКЦИОНИРОВАНИЯ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

Необходимо произвести расчет для спроектированной системы управления. В процессе расчета необходимо:

Выделить подсистемы (от 5 до 7). Каждая подсистема представляет собой отдельный модуль управления. Если в системе присутствуют «типовые» узлы, они выделяются в отдельную подсистему.

Для каждой подсистемы определить набор входных и выходных сообщений, а также источники и преемники данной информации.

На основании выделенных входных и выходных данных необходимо построить таблицу сообщений, включающих следующие данные:

Номер сообщения

Источник сообщения (наименование)

Приемник сообщения (наименование)

Размер сообщения (в битах)

Тип сообщения (Периодический, Апериодический, Спорадический)

Периодичность сообщения (в единицах времени, T)

Крайний критический срок исполнения (в единицах времени, D)

Задержка отправки сообщения в очередь (jitter в единицах времени, J)

Размер пакета при передачи данных, включая «накладные расходы»

Приоритет сообщения (вычисляется, в зависимости от используемого метода назначения приоритета)

На основании таблицы исходных данных, производится расчет максимальной задержки передачи сообщения, от источника до приемника для каждого сообщения. (R). Расчет производится для четырех скоростей передачи данных. (По указанном в варианте - протоколу передачи данных)

По полученным результатам строится таблица 2, включающая информацию о задержках передачи сообщения

a. Номер сообщения
b. Источник сообщения (наименование)
c. Приемник сообщения (наименование)
d. Размер сообщения (в битах)
e. Задержка для скорости 1 (в единицах времени, R1)
f. Задержка для скорости 2 (в единицах времени, R2)
g. Задержка для скорости 3 (в единицах времени, R3)
h. Задержка для скорости 4 (в единицах времени, R4)

По результатам таблицы 2 формируется вывод о системе. Для оценки о системе требуется вычисление трех параметров:

i. Message Utilization – Коэффициент передачи полезных данных в сообщении (%)

j. Bus Utilization – Коэффициент передачи всех данных в сообщении (%)

k. Breakdown Utilization – Коэффициент возможности увеличения размеров сообщения. (коэффициент расписабельности)

В случае, если коэффициент Breakdown Utilization<1, необходимо произвести оптимизацию системы методом объединения коротких сообщений.

Этапы выполнения лабораторной работы следующие:

1. Проектирование системы. При проектировании необходимо выделить подсистемы и сообщения в системе, а также произвести словесное описание функционирования системы при заданных условиях.
2. Расчет автоматизированной системы на критерий функционирования в режиме реального времени. Расчет производится по параметру R(J,D,T), а также на основании информации по протоколу передачи данных.
3. Оптимизация автоматизированной системы для увеличения коэффициента Breakdown Utilization. (Оптимизация выполняется методом объединения коротких сообщений)
4. Подготовка отчета. Состав отчета:

a. Проектирование системы с обоснованием разделения системы (почему именно так).

b. Каждое сообщение системы должно содержать информацию. (Между такими подсистемами и зачем оно было нужно).

c. Словесное описание функционирования системы

d. Характеристики протокола передачи данных, с включением общего описания технологии, скоростей передачи данных, накладных расходов, применение протокола передачи данных.

e. Расчет автоматизированной системы, с указанием таблиц 1, таблиц 2, формул расчета.

f. Оптимизация расчета (при необходимости).

g. Выводы по расчетам (возможность функционирования, плюсы автоматизации)

Таблица. Данные по протоколам передачи данных

Наименование протокола	Скорость передачи данных	Накладные расходы + восстановление при сбоях	Разработчик
EIB	9,6 кБит/сек 100 кБит/сек 500 кБит/сек 1200 кБит/сек	70 бит + 6 бит	Siemens
LonWorks	78,6 кБит/сек 125 кБит/сек 250 кБит/сек 1200 кБит/сек	84 бит + 10 бит	Echelon
CAN	125 кБит/сек 250 кБит/сек 500 кБит/сек 1000 кБит/сек	35 бит + 5 бит	Bosh
P-NET	56,6 кБит/сек	24 бит + 5 бит	Process

	78,6 кБит/сек 120 кБит/сек 240 кБит/сек		DATA
TCP-IP	56,6 кБит/сек 1000 кБит/сек 2000 кБит/сек 10000 кБит/сек	48 бит + 5 бит	
Profibus	9,6 кБит/сек 12,4 кБит/сек 56,6 кБит/сек 78,6 кБит/сек	32 бит + 5 бит	

Задание состоит из двух частей (наименование системы автоматизации, в скобках указана базовая технология, на основе которой должна быть реализована система автоматизации.)

Варианты заданий:

1. АЗС (P-NET)
2. Биологическая лаборатория (LonWorks)
3. Тепловая машина (P-NET)
4. Автопилот (CAN)
5. АТС (LonWorks)
6. Здание (EIB)
7. Прогноз погоды (TCP-IP)
8. Сточные воды (P-NET)
9. Управление освещением в помещении (EIB)
10. Управление движением автобуса (LonWorks)
11. Охранная система квартиры (EIB)
12. Система управление движением на перекрестке с разной интенсивностью движения, связь с другими перекрестками (TCP-IP)
13. Безопасность движения на автомобиле (CAN)
14. Климат в помещении (LonWorks)
15. Охрана гаражного кооператива (TCP-IP)
16. Пропускная система предприятия (EIB)
17. Управление светофорным объектом (P-NET)
18. Управление ж/д станцией (P-NET)
19. Управление бильярдом (TCP-IP)
20. Управление мишенями на стрельбище (LonWorks)
21. Автоматизированное кафе (CAN)
22. АЗС (CAN)
23. Биологическая лаборатория (EIB)
24. Тепловая машина (TCP-IP)
25. Автопилот (LonWorks)
26. АТС (EIB)
27. Здание (TCP-IP)
28. Прогноз погоды (CAN)

29. Сточные воды (TCP-IP)
30. Управление освещением в помещении (LonWorks)

Содержание работы:

1. Титульный лист.
2. Цель работы.
3. Описание выполнения работы.
4. Выводы.

РАЗРАБОТКА ИНТЕРФЕЙСА ОПЕРАТОРА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ В SCADA ПРИЛОЖЕНИИ

Исходные данные для выполнения лабораторной работы:

Лабораторная работа выполняется в SCADA приложении Simple-Scada 1.3.3.

Для создания интерфейса оператора необходимо использовать виртуальное внешнее устройство.

Для создания проекта используются программыt компоненты:

Explorer – создание страниц проекта, выбор компонент системы (устройства, переменные, (теги), сервера, платы ввода вывода). Основное средство управления проектом.

Builder – для просмотра и создания всех элементов системы, а также ошибок компиляции.

Runtime – система запуска приложения, разработанного в SCADA и ее проверки функционирования в режиме реального времени и режиме эмуляции.

При использовании не тривиальных функций управления, осуществляется программирование проекта.

Алгоритм создания проекта:

1. Создание проекта, либо создания ссылки на проект. При создании в окне Explorer для заданного проекта формируется шаблон, включающий средство создания страниц, а также всех компонент системы. Компоненты системы – теги переменных, устройства, сервера ввода вывода, системные компоненты.
2. Создается сервер ввода вывода. При выборе данной функции необходимо указать имя.
3. Создается устройство ввода вывода, с которым будет взаимодействовать ПК. Для устройства необходимо указать уникальный идентификатор, адрес (от 1 и выше), тип устройства (Generic), тип памяти, из которой будут приниматься данные (MEMORY).
4. Для всех датчиков (устройства передающие сигналы и управляемые пользователем) создаются теги (переменные). При создании Variable Tag важно указать его имя и тип формируемых данных. Имя – уникальный в системе идентификатор, тип – INT, DECEMAL и пр. Также необходимо для всех элементов указать адрес устройства. При указании адреса следует именовать их начиная с буквы, именующий тип. При типе DECEMAL адрес должен быть D1 D100, при INT – I1 I100 и так далее. Адрес состоит из двух частей – буквенное обозначение (первая буква типа данных), число обозначающее адрес.
5. Для исполнительных устройств также могут быть созданы теги, если вывод информации осуществляется на внешние устройства, либо производится опрос. Если используется виртуальный объект теги указать необязательно.
6. В проекте создаются страницы интерфейса оператора. В проекте может быть одна стартовая страница и несколько вызываемых страниц в процессе

функционирования. (Требуется создать 1 – 3 страницы). Создание осуществляется через Explorer.

7. Переключение между страницами реализуется через кнопки управления, выбираемых в панели управления. Необходимо реализовать кнопку «Выход».

8. Для интерфейса оператора на страницах необходимо разместить все датчики и исполнительные элементы. Добавление осуществляется через панель инструментов.

9. Для всех датчиков (управляются пользователем) в окне свойств необходимо перейти к вкладке Access или Input и указать в окне ввода текста реакцию на действие датчика (например установить значении тега в заданное значение).

10. Для исполнительных элементов реакция указывается на одном из окон (Appearance, Movement, Scaling, Fill, Slider). Реакцией может быть изменение масштаба объекта, изменение цвета заливки, изменение кадра, движение по заданным величинам, задается степень видимости объекта. В окне ввода текста указывается условие, при котором заданное действие наступает.

11. Каждый элемент также может иметь текстовое описание при выводе на экран.

Примечание:

1. При необходимости в проект могут быть включены элементы из внешних графических редакторов.
2. Для каждого тега необходимо указывать устройство ввода вывода.

Задание:

Необходимо разработать интерфейс оператора автоматизированной системы в соответствии с вариантом задания.

1. Для автоматизированной системы предусмотреть 4-8 управляющих воздействий на алгоритм функционирования.
2. Разработать несколько «графических» исполнительных элементов, которые иллюстрируют действия на систему.
3. Создать автономный алгоритм функционирования системы.
4. Подготовить отчет.

Варианты заданий:

1. Модель железнодорожной стрелки
2. Модель 2 перекрестков: равнозначного и с приоритетом
3. Модель химического производства
4. Модель подачи воды в жилой дом, измерение потока, дублирование при аварии
5. Модель системы управления освещением при входе в помещения (2-3 помещения)
6. Модель системы охранной сигнализации на объектах, вывод информации при срабатывании трех типов датчиков
7. Модель системы управления шлюзованием
8. Модель системы отопления помещения в зданиях (2 здания по 5 помещений в каждом)

9. Модель системы подачи электрической энергии на объекты (2-3 объекта, 2-3 станции)

10. Модель системы производства руды

11. Модель системы управления движения скорой помощи по автодорогам (выбор пути, 2-3 пути)

12. Модель системы управления отправкой железнодорожных составов на станции

13. Модель системы управления вылетом и посадкой самолетов

14. Модель конвейера

15. Модель системы пожарной сигнализации в помещениях здания (5-7 помещения)

16. Модель системы коммутации объектов через общую сеть (сетевая активность узлов – 4-5 узлов)

17. Модель системы учета посещаемости

18. Модель АЗС

19. Модель зернохранилища

20. Модель контроля климата в помещениях

21. Модель телефонной станции

22. Модель системы прогноза погоды

23. Модель системы биллинга электрической энергии

24. Модель системы управления зданием

25. Модель системы контроля газовой станции

26. Модель системы управления самолетом

27. Модель системы управления движением автобусом

28. Модель системы управления складом готовой продукции

29. Модель системы управления автомобильным краном

30. Модель системы управления процессов выкачивания нефти

Содержание отчета:

1. Титульный лист с указанием номера варианта.
2. Описание варианта задания.
3. Описание тегов проекта.
4. Описание алгоритмов функционирования.
5. Графическое представление системы.

БАЗОВЫЕ СВЕДЕНИЯ ПО ОПЕРАЦИОННОЙ СИСТЕМЕ QNX

1. Цель работы

Работа в многопользовательской операционной системе (ОС) существенным образом отличается от работы в однопользовательской операционной системе, в такой, например, как ОС MS-DOS. Более того, интерфейс пользователя в многопользовательских операционных системах отличается при переходе от одной ОС к другой. Например, если в ОС MS Windows интерфейс пользователя представляет графический оконный интерфейс, то в таких ОС как Unix, Linux, QNX наряду с графическим интерфейсом имеется и текстовый (командный) интерфейс. Причем подавляющее большинство пользователей предпочитают командный интерфейс.

Командный интерфейс в ОС реального времени QNX, как впрочем, и графический интерфейс, имеют некоторые особенности по сравнению с интерфейсами других многопользовательских ОС общего назначения.

Целью данной лабораторной работы является получение пользователем базовых сведений и приобретение практических навыков работы с командным и графическим интерфейсами ОС QNX.

2. Теоретические сведения

. Инициирование и завершение сеанса работы в ОС QNX

После загрузки ОС QNX, она выдает пользователю приглашение к началу работы. При этом последовательно выполняются две команды операционной системы:

login:

password:

Пользователь соответственно должен указать имя, под которым он зарегистрирован в операционной системе и свой пароль. Имя и пароль предоставляет пользователю системный администратор. Имя и пароль могут состоять из английских букв и цифр, причем имеет значение регистр букв, т.е. заглавные и строчные буквы отличаются.

Пользователь может работать в системе как суперпользователь (обычно это администратор) и простой пользователь. Суперпользователь имеет имя (**login**) **root**, а простой пользователь может иметь любое имя. Если пользователь входит под суперпользователем, то в процессе работы в качестве подсказки выдается символ **#**, а для простого пользователя – **$**.

В многопользовательских ОС пользователь имеет доступ к данным (к части файловой системе) ограниченным его именем. Суперпользователь (администратор) имеет доступ ко всей файловой системе. Поэтому в целях безопасности (предотвращения доступа к данным посторонним пользователям) при завершении сеанса работы необходимо выполнить команду **logout**.

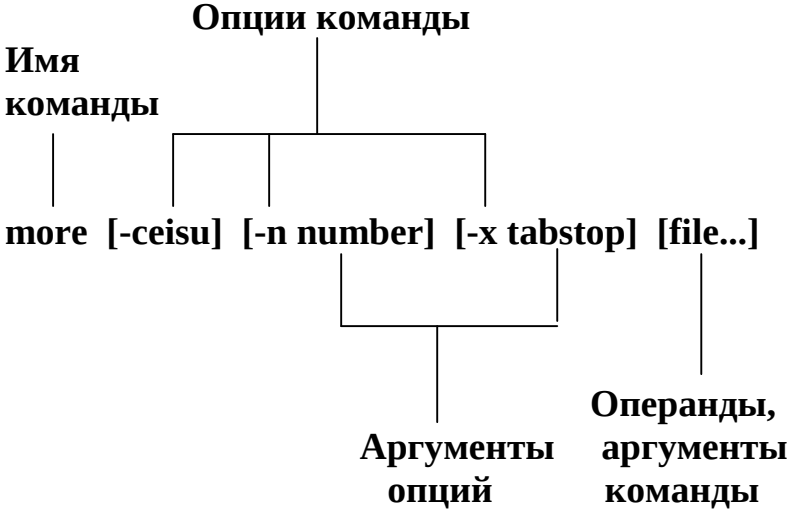
Для системного останова ОС и ее перезагрузки используется команда **shutdown**.

Использование клавиатуры ПК в ОС QNX

При вводе в командной строке можно использовать следующие клавиши:
< > < > <**Home**> <**End**> <**Backspace**> <**Del**> <**Ins**>.
Инициация выполнения команды – <**Enter**>.
<**Ctrl**>+<**U**> – удаление строки.
Вызов истории команд: < > < >.
Переключение виртуальных консолей (**/dev/con1**, ...):
<**Ctrl**>+<**Alt**>+<**Enter**> или <**Ctrl**>+<**Alt**>+<**+**> – переключение на следующую консоль;
<**Ctrl**>+<**Alt**>+<**-**> – переключение на предыдущую консоль;
<**Ctrl**>+<**Alt**>+<**n**> – переключение на консоль с номером **n**.
Переключение шрифтов осуществляется с помощью команды **cfont**.
Быстрое переключение шрифтов:
<**Ctrl**>+<**Alt**>+<**>**> – переключение на следующий шрифт (до **n**);
<**Ctrl**>+<**Alt**>+<**<**> – переключение на предыдущий шрифт (до **0**).
Конец файла – <**Ctrl**>+<**D**>
Завершение процесса (терминального): <**Ctrl**>+<**C**> или <**Ctrl**>+<**Break**>
Приостановка вывода на экран: <**Ctrl**>+<**S**>
Продолжение вывода на экран: <**Ctrl**>+<**D**>
Перезагрузка компьютера (программная):
<**Ctrl**>+<**Alt**>+<**Shift**>+<**Del**>

Основные команды в ОС QNX

При работе пользователя с ОС QNX используется следующий синтаксис команд (на примере команды **more**):



[...] – необязательные опции или аргументы команды.
-a | -f – альтернатива опций.
... – возможно повторение данного аргумента.

Перенаправление ввода-вывода

Большинство команд читает свой ввод из файла стандартного ввода (**stdin**), который обычно является клавиатурой и записывает свой вывод в стандартный

файл вывода (**stdout**), который обычно является экраном дисплея. Однако, возможно:

читать из файла отличного от клавиатуры;

писать в файл отличный от экрана дисплея.

читать из файла или другого устройства < (ввод перенаправленных символов)

писать в файл или в другое устройство > (вывод перенаправленных символов)

перенаправление **stdout** в файл, добавление к файлу >> (добавление выводимых символов)

канал **stdout** прямо в другую команду | (конвейер, канал символов)

Например,

ps > /tmp/pr1

Выполнение команд на других узлах (node) или консолях (tty)

Вы можете выполнять команды на другой машине (другом узле) в локальной сети. Когда команда вызывается на другом узле, **stdin**, **stdout**, **stderr** показываются на консоли вашего компьютера или на терминале, пока вы не перенаправите на другое устройство.

Команда **on node** – выполнение команды на другом узле.

Команда **on tty** – перенаправление **stdin**, **stdout** на различные **tty**.

Использование **on node**.

Пример 1. Запуск **sin** на 4 узле ввод-вывод на вашем экране:

onnode 4 sin

Эквивалент: **//4 sin**

Пример 2. Загрузить **sin** со второго узла из **/bin** и выполнить ее на **4 узле** (потоки ввода-вывода **sin** будут все еще прикреплены на выполняющем: экране):

onnode 4 //2/bin/sin

Пример 3. Запустить **sin** на **4 узле** и перенаправить вывод на **консоль 1 узла 2** (потоки ввода и вывода ошибок будут по-прежнему прикреплены к консоли запуска):

onnode 4 sin >//2/dev/con1

Использование **ontty**.

Пример 1. Запустить команду **ls** с ее вводом и выводом, связанным с **консолью 3**:

ontty /dev/con3 ls

Совместное использование on node и on tty

Пример 1. Запуск сервера имен **nameloc** на **4 узле** и прикрепление

всего своего ввода-вывода к /**dev/con1 узла 4**:

onnode 4 ontty //4/dev/con1 nameloc &

Но в этом случае все равно данная команда остается «ребенком» запускаемого **shell**. Чтобы полностью открепиться от **shell** необходимо использовать команду **nohup**:

onnode 4 ontty //4/dev/con1 nohup nameloc &

Для получения справки о правилах использования команд необходимо использовать команду **use**:

use sin

Использование универсальных символов

***** – замена любого числа (**>= 0**) любых символов.
? – замена любого одиночного символа.

[] - замена любых символов, содержащихся в скобочках. Может быть диапазон, [**1-3] [a-c]**.

! - исключение символов, специфицированных в скобках **[!a]**.

Примеры.
cp f* /tmp
cp *d /tmp
cp freg? /tmp
cp freg[123] /tmp или cp freg[1-3] /tmp
cp *.[ch] /tmp
cp *.[!o] /tmp {копировать все файлы, кроме тех, которые имеют на конце имени **o**}

Ввод множества команд:
cp freg /tmp; ls /tmp

Работа в файловой системе ОС QNX

Имя файла может быть до 48 символов в длину.
Вы можете применять в именах файлов символы из следующих диапазонов:

0x00 … 0x1F
0x2f (/)
0x7F (rubout)
0xFF

Но для переносимости программ желательно в именах файлов использовать символы, которые определены в **POSIX**:

a...z A...Z 0 1 ... 9 _ . -

(нельзя использовать символ «-» в качестве первого символа имени файла).

Суффиксы в именах файлов ничего не означают. По суффиксам имеется локальное соглашение (например, **.txt .c .h .o** и т.д.).

Имена путей (маршруты)

/ – корневой каталог (директорий).

//2/ – корневой каталог на 2 узле.

Текущий директорий – **...**
Домашний директорий (/home/slava).
Возврат в домашний каталог cd.

Типичная файловая система (ФС) QNX

В файловой системе ОС QNX обычно присутствуют следующие директории (каталоги):

Каталог	Описание
/bin	выполняемые файлы
/boot	Makefile образа ОС
/boot/build	make-файлы для построения образов
/boot/images	файл образа ОС
boot/sys	системные процессы, необходимые во время загрузки
/etc	инициализационные и другие файлы
/etc/config	sysinit и конфигурационные файлы
/etc/readme	информационные файлы о версии программного обеспечения
/etc/readme/technote	технические замечания
/tmp	локализация, по умолчанию, временных файлов
/usr/bin	выполняемые файлы
usr/include	файлы заголовков для С компилятора
/usr/lib	библиотеки для С компилятора
/usr/lib/terminfo	файлы описания терминалов
/usr/lib/APPLICATION	инсталлированные QNX приложения
/usr/spool/lp	рабочие файлы для системных спулеров
/home/USERID	домашние каталоги пользователей

Последние 2 директория конфигурируются системным пользователем.

Наиболее часто используемые команды QNX

Работа с директориями

cd – изменение рабочего директория.
mkdir – создание директория.
rmdir – удаление директория.
pwd – печать имени рабочего директория.
ls [-l] – список содержания директория.

Работа с файлами

diff – отчет о различии содержания двух файлов.
cat – конкатенация файлов, вывод содержимого файла на экран.
cp – копирование файлов.
wc – обход структуры директориев и выполнение команд.
more – показ содержимого файла в буферизированном виде.
less – интерактивный (управляемый) просмотр вывода на экран.
lp – вывод содержания файла на принтер.
mv – перенос файла.
rm – удаление файла.
grep – поиск по строке, заданной в виде шаблона.
sort – сортировка, слияние или проверка последовательности текстового файла.

Другие команды

who – вывод на экран пользователей в системе.
ps – вывод на экран отчета о статусе процессов.
sin – вывод системной информации на экран.
find – поиск файлов.
write – посылка сообщения на терминал пользователя.
wall – посылка широковещательного сообщения.

5. Инициирование и завершение работы в графической оболочке Photon

Кроме командной строки в ОС QNX имеется графическая оболочка Photon. Данная графическая оболочка выполнена в стандарте Motif и основана, как и операционная система, на межпроцессных сообщениях.

Запуск графической оболочки Photon осуществляется с помощью команды: **ph**

Выход из графической оболочки, завершение функционирование оболочки происходит при выполнении команды **shutdown** меню оболочки Photon.

Порядок выполнения лабораторной работы

1. Внимательно прочитайте и изучите методические указания (раздел 2). Ответьте на вопросы для самоконтроля (раздел 5).
2. Ознакомьтесь с содержанием раздела 4, и спланируйте написание отчета по выполняемой лабораторной работе.
3. Последовательно выполните задания раздела 6.

4. Проанализируйте полученный в результате работы опыт и сделайте выводы, в том числе о достижении цели работы.
5. Оформите письменный отчет.

Содержание отчета о лабораторной работе

1. Титульный лист.
2. Цель работы.
3. Краткие теоретические сведения об основных командах ОС QNX.
4. Индивидуальное задание к лабораторной работе и описание его выполнения.
5. Выводы.

Варианты индивидуальных заданий по лабораторной работе

1. Освоить корректные начало и завершение работы в командном режиме ОС QNX для простого пользователя и суперпользователя. Сведения об имени и пароле получить у преподавателя.
2. Усвоить назначение основных клавиш и их сочетаний при работе в командном режиме ОС QNX и освоить практическую работу с ними.
3. Усвоить синтаксис команд ОС QNX и освоить работу с командами, указанными в разделе «Основные команды в ОС QNX».
4. Найти в файловой системе директории, указанные в разделе «Работа в файловой системе ОС QNX» и изучить их содержание.
5. Инициировать работу графической оболочки Photon и изучить основы работы с ней, используя для этого документацию по Photon.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Кухаренко, Б. Г. Интеллектуальные системы и технологии [Электронный ресурс]: Учебное пособие / Б. Г. Кухаренко.- Москва: Альтаир|МГАВТ, 2015. - 115 с.

2. Партыка Т. Л. Операционные системы, среды и оболочки [Текст]: учебное пособие /Татьяна Ивановна Партыка, Игорь Иванович Попов.- 4-е изд., перераб. и доп. - М.: ФОРУМ, 2012. - 560 с.

3. Советов, Б. Я. Информационные технологии [Текст] : учебник для прикладного бакалавриата / Б. Я. Советов, В. В. Цехановский. - 6-е изд., перераб. и доп. – Москва : Юрайт, 2015. - 263 с.

4. Назаров, С. В. Современные операционные системы [Текст]: учеб. пособие / С. В. Назаров, А. И. Широков. М. : Интернет-Университет Информационных Технологий: БИНОМ. Лаборатория знаний, 2011-280 с.

5. Онокой, Л. С. Компьютерные технологии в науке и образовании [Текст] : учебное пособие / Л. С. Онокой, В. М. Титов. – Москва : ФОРУМ: ИНФРА-М, 2014. - 223 с.

6. Синицын, С. В. Операционные системы [Текст] : учебник / С. В. Синицын, А. В. Батаев, Н. Ю. Налютин. – 2-е изд., испр. – М. : Академия, 2012.- 304 с.