МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»

Кафедра автоматизированных систем управления (АСУ)

УТВЕРЖДАЮ
Зав. кафедрой АСУ, профессор
[redacted] А.М. Кориков

АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ КОМПЛЕКСОВ

Тема 1. Состояние и тенденции развития АВК

Учебно-методическое пособие

для студентов уровня основной образовательной программы: **магистратура**
направление подготовки: **09.04.01 - Информатика и вычислительная техника**

Разработчик
доцент кафедры АСУ

В.Г. Резник

2017

Резник В.Г.

Архитектура вычислительных комплексов. Тема 1. Состояние и тенденции развития АВК. Учебно-методическое пособие. – Томск, ТУСУР, 2017. – 49 с.

Учебно-методическое пособие предназначено для изучения темы №1 по дисциплине «Архитектура вычислительных комплексов» для студентов уровня основной образовательной программы магистратура направления подготовки: 09.04.01 «Информатика и вычислительная техника».

Оглавление

Введение.....4
1 Тема 1. Состояние и ненденции развития АВК.....5
1.1 Основные понятия архитектуры ЭВМ.....7
1.2 Многоуровневая компьютерная организация.....9
1.3 Историческое развитие архитектуры ЭВМ.....11
1.3.1 Нулевое поколение (1492-1945).....11
1.3.2 Первое поколение (1937,1945 -1953,1955).....11
1.3.3 Второе поколение (1954,1955 -1962,1965).....12
1.3.4 Третье поколение (1963,1965 – 1972,1980).....13
1.3.5 Четвертое поколение (1972,1980 - 1984).....13
1.3.6 Пятое поколение(1984-1990).....14
1.3.7 Шестое поколение (1990-) – невидимые компьютеры.....14
1.4 Процессоры и шины ЭВМ.....15
1.5 Структуры взаимосвязей устройств ЭВМ.....17
1.6 Пути повышения быстродействия (мощности) ЭВМ.....18
1.6.1 Параллельные сумматоры.....18
1.6.2 Периферийные процессоры и шины.....18
1.6.3 Многопрограммный режим работы.....20
1.6.4 Внедрение в процессор нескольких специальных АЛУ.....20
1.6.5 Системы обработки данных (СОД).....21
2 Лабораторная работа №1.....27
2.1 Структура учебной части дистрибутива ОС УПК АСУ.....27
2.2 Настройка сети ОС УПК АСУ.....29
2.3 Контроль выполнения лабораторной работы №1.....30
3 Лабораторная работа №2.....31
3.1 Работа со средой кластера ЭВМ кафедры АСУ.....31
3.2 Настройка сетевого обеспечения ОС.....31
3.3 Консольный доступ к кластеру.....33
3.4 Графический доступ к кластеру.....36
4 Лабораторная работа №3.....43
4.1 Удаленная разработка приложений.....43
4.2 Контроль выполнения работ по теме 1.....47
Список использованных источников.....48

Введение

Данное методическое пособие содержит учебный материал по первой теме дисциплины «Архитектура вычислительных комплексов », - сокращенно АВК.
Изложенный материал является обязательной частью процесса обучения магистранта по направлению подготовки 09.04.01 «Информатика и вычислительная техника» и содержит как теоретическую часть, так и методические указания по выполнению лабораторных работ.
Последовательность и тематическая направленность учебного материала данного пособия предполагает, что магистрант:

- прошел обучение по курсу «Современные операционные системы»;
- имеет на собственном flash-носителе загрузчик ОС УПК АСУ;
- умеет запускать ОС УПК АСУ, как в текстовом, так и в графическом режимах;
- имеет личный личный защищенный доступ к вычислительному кластеру кафедры АСУ.

Изложенный материал разбит на ряд разделов, последовательность которых определяет сам порядок процесса обучения.

В первом разделе представлен торетический материал по объявленной теме «Состояние и тенденции развития АВК», который рассчитан на 4 академических часа и, в дальнейшем, закрепляется с помошью трех лабораторных работ.

Второй, третий и четвертый разделы содержат методические описания самих лабораторных работ, выполняемых по данной теме.

Последний раздел описывает требования по контролю выполняемых работ, применительно к тематике всего курса.

Во втором разделе, последовательно излагаются действия:

- по настройке сетевого обеспечения ОС рабочей станции студента;
- правила установки защищенного терминального соединения с кластером ЭВМ кафедры АСУ;
- особенности архитектуры X Window System, а также параметры доступа, обеспечивающие возможность подключения программ, запущенных на кластере, к собственному X-серверу.

Замечание

Технология проведения всех работ, предполагает, что каждый студент должен иметь flash-память не менее 2 Гб для сохранения личного материала и запуска УПК АСУ.

1 Тема 1. Состояние и ненденции развития АВК

Прежде чем начать изучение «Архитектуры вычислительных комплексов (АВК)», необходимо ответить на вопрос: «*Зачем нужны компьютеры?*».
Одним из возможных ответов: «*Компьютеры нужны для обработки данных*».

Известный русский ученый, Ларионов А.М., анализируя возможности обработки данных на ЭВМ, дает определение и классификацию различных систем обработки данных.
Система обработки данных (СОД) — совокупность технических средств и программного обеспечения, предназначенная для информационного обслуживания пользователей и технических объектов.

Общая структура классификации СОД, данная Ларионовым А.М., приведена на рисунке 1.1. Ее с успехом можно использовать в качестве ориентира при изучении различных дисциплин, связанных с применением вычислительной техники.

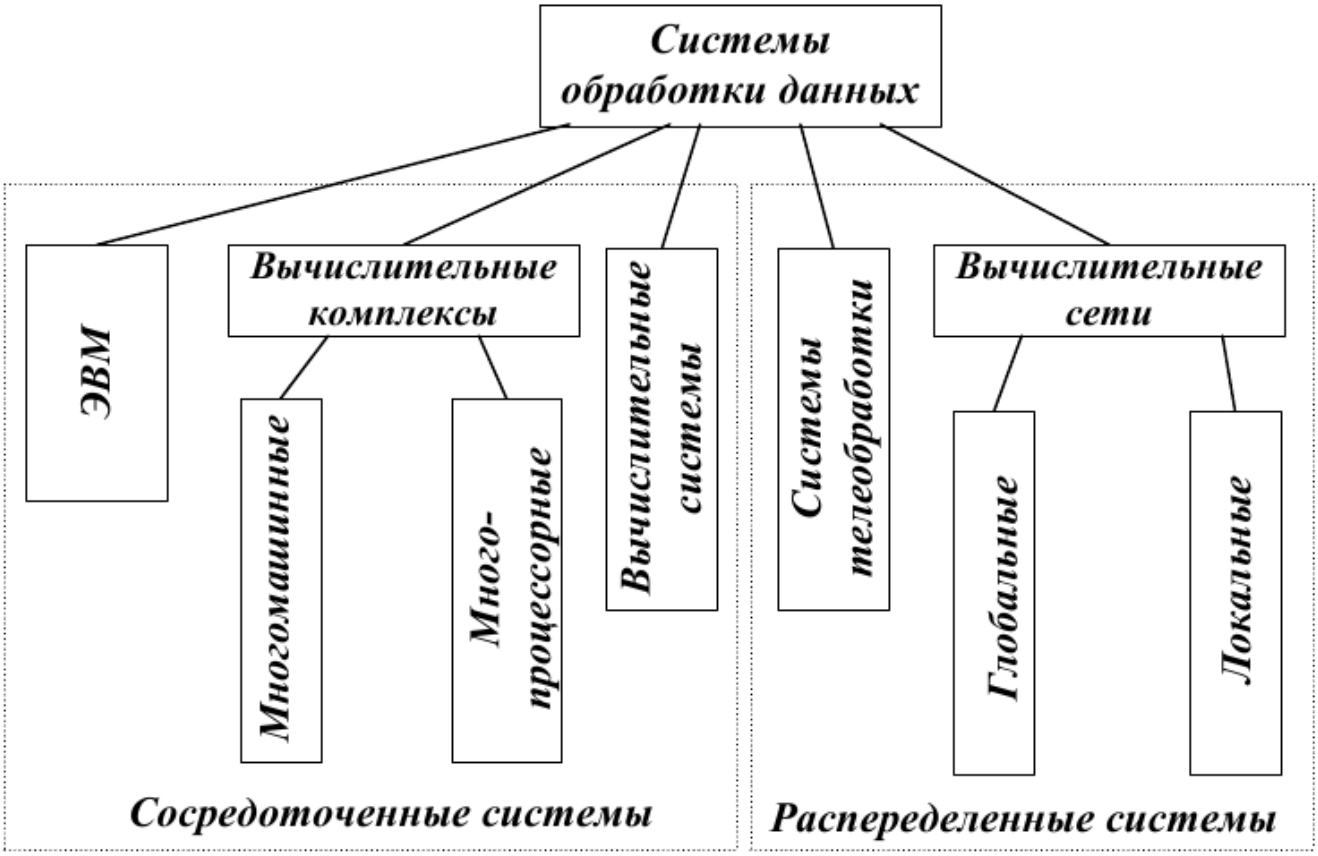

Рисунок 1.1 — Структура классификации СОД (Ларионов А.М.)

Хорошо видно, что вся классификация разделена на два больших класса:

- *сосредоточенные (централизованные) системы*, в которых обработка данных ведется отдельной ЭВМ, вычислительным комплексом или вычислительной системой;

- *распределенные системы*, в которых процессы обработки данных рассредоточены по многим компонентам: системам телеобработки или вычислительным сетям.

С другой стороны, архитектура современного общедоступного компьютера, представленного на рисунке 1.2, является набором функциональных компонент, которые во-многом работают независимо друг от друга. Чтобы эти компоненты работали согласованно, используется *общая системнная манистраль*.

Рисунок 1.2 - Архитектура аппаратной части ЭВМ

Основу архитектуры аппаратной части ЭВМ составляет *системный блок*, в котором размещены:

- микропроцессор (**МП**);
- блок оперативного запоминающего устройства (**ОЗУ**);
- микросхемы постоянного запоминающего устройства (**ПЗУ**);
- устройства долговременной памяти на «жестком диске» (**Винчестер**);
- устройства для запуска компакт-дисков (**CD**) и дискет (**НГМД**).

В системном блоке находятся также *интерфейсные платы*: сетевая, видеопамяти, обработки звука, модем (модулятор-демодулятор), платы, обслуживающие устройства ввода-вывода: клавиатуры, дисплея, "мыши", принтера и других устройств.

1.1 Основные понятия архитектуры ЭВМ

Вычислительная машина (ЭВМ, ВМ, ПК) — это комплекс технических и программных средств, предназначенный для автоматизации подготовки и решения задач пользователей.

Вычислительная система - это совокупность взаимосвязанных и взаимодействующих процессоров или вычислительных машин, периферийного оборудования и программного обеспечения, предназначенную для подготовки и решения задач пользователей.

Под **архитектурой ЭВМ** обычно понимается логическое построение ВМ, то есть то, ***какой машина представляется программисту***.

Замечание

Здесь необходимо отметить, что из рассмотрения выпадают вопросы физического построения вычислительных средств:

- состав устройств,
- число регистров процессора,
- емкость памяти,
- наличие специального блока для обработки вещественных чисел,
- тактовая частота центрального процессора и тому подобное.

Весь этот круг вопросов принято определять понятием ***организация вычислительной машины***.

Можно выделить четыре основных уровня детализации *архитектуры ЭВМ*, представленные на рисунке 1.3:

а) - *уровень «черного ящика»*, когда ЭВМ рассматривается как устройство, способное хранить и обрабатывать информацию, а также обмениваться данными с внешним миром. Например, ЭВМ может быть подключена к коммуникационной сети или к ней могут подключаться периферийные устройства.

б) - *уровень общей архитектуры*, когда ЭВМ представляется в виде четырех составляющих: центрального процессора (ЦП), основной памяти (ОП), устройств ввода/вывода (УВВ) и системы шин.

в) - *уровень архитектуры центрального процессора*, когда детализируется каждое из устройств второго уровня. Например, для центрального процессора можно выделить:

- **арифметико-логическое устройство** (АЛУ), обеспечивающее обработку целых чисел;
- **блок обработки чисел** в формате с плавающей запятой (БПЗ);
- **регистры процессора**, использующиеся для краткосрочного хранения команд, данных и адресов;
- **устройство управления** (УУ), обеспечивающее совместное функционирование устройств ВМ;
- **внутренние шины**.

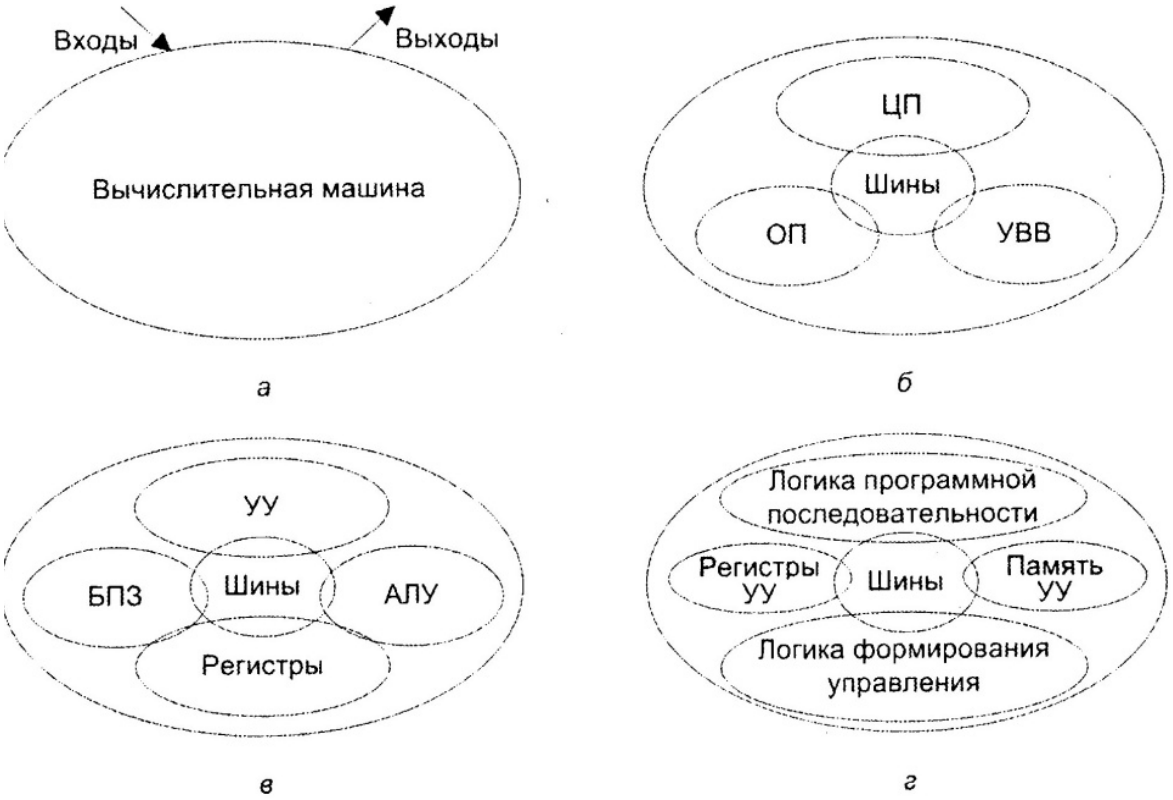

Рисунок 1.3 - Уровни детализации вычислительной машины

г) — *уровень архитектуры устройства управления*, когда детализируются элементы третьего уровня. Например, может быть раскрыта *структура устройства управления* (**УУ**), представленная в виде четырех составляющих:

- *логики программной последовательности* - электронных схем, обеспечивающих выполнение команд программы в последовательности, предписываемой программой;
- *регистров и дешифраторов* устройства управления; управляющей памяти;
- *логики формирования управления*, генерирующей все необходимые управляющие сигналы.

Замечание

Применительно к параллельным и распределенным многопроцессорным и многомашинным вычислительным системам зачастую вводят понятие «**метауровня**».

1.2 Многоуровневая компьютерная организация

Существует разница между тем, *что удобно людям, и тем, что могут делать компьютеры*. Одно из решений данной проблемы – *многоуровневая организация ЭВМ и языков управления компьютером*.

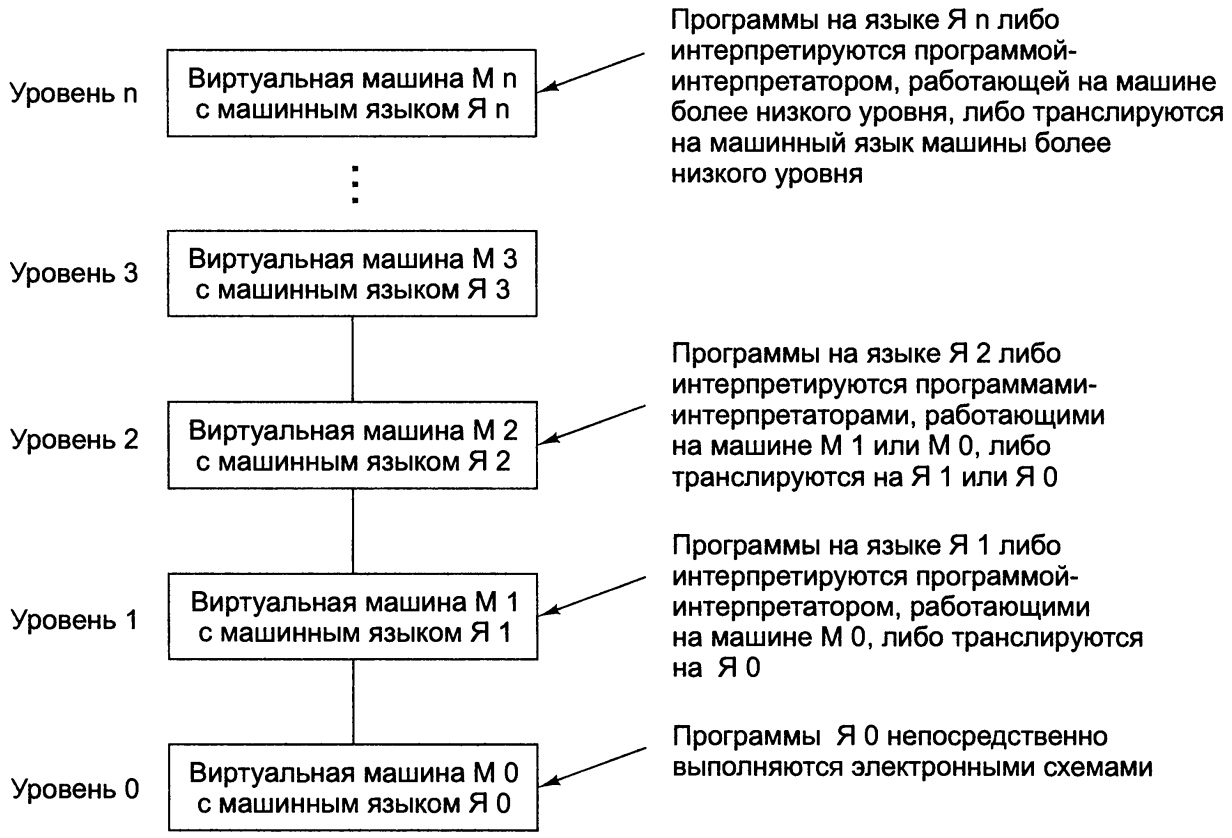

Рисунок 1.4 - Многоуровневая организация ЭВМ

Уровень 0 – это аппаратное обеспечение машины.
Это – *цифровой логический уровень*, объектами которого являются *вентили*.
Сами вентили состоят из аналоговых компонентов, таких как транзисторы, но они могут быть точно смоделированы как цифровые устройства.
Каждый вентиль:

- *имеет* один или несколько цифровых входов (сигналов), представляющих 0 или 1;
- *вычисляет* простые функции таких сигналов как «И» или «ИЛИ».

Несколько вентилей формируют *1 бит памяти*, который может содержать 0 или 1.
Биты памяти, объединенные в группы, например, 8, 16, 32 или 64, формируют *регистры*.
Каждый регистр может содержать одно двоичное число до определенного предела.

Замечание

Из вентилей также может состоять и сам компьютер.

Таким образом, современную *6-уровневую организацию компьютера* можно представить иерархической схемой, показанной на рисунке 1.5:

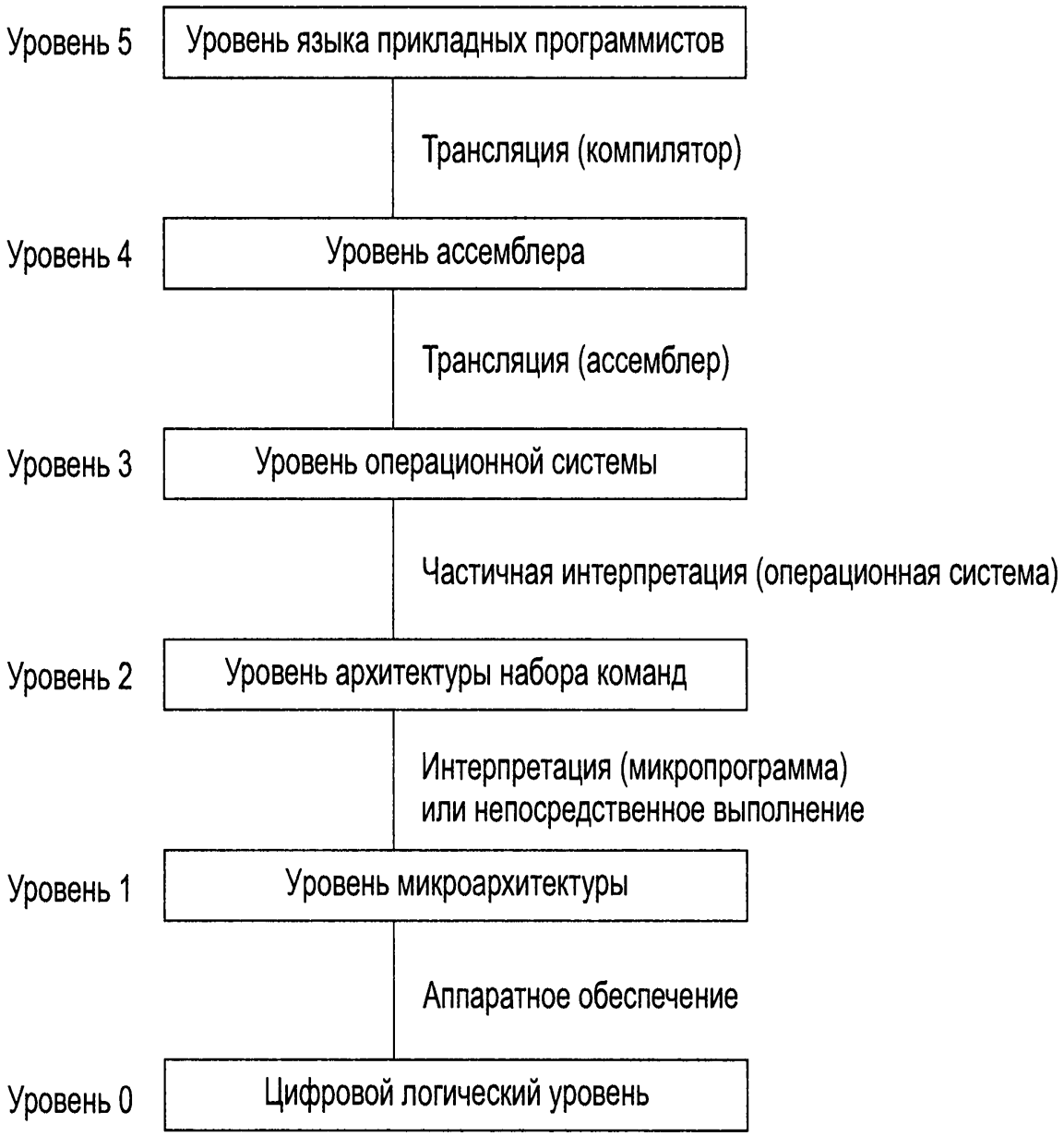

Рисунок 1.5 - Шестиуровневая органзация ЭВМ

1.3 Историческое развитие архитектуры ЭВМ

Историческое развитие архитектуры ЭВМ дает нам представление об этапах формирования архиектурных СОД.

1.3.1 Нулевое поколение (1492-1945)

Для полноты картины упомянем два события, произошедшие до нашей эры:

- **первые счеты** - абак, изобретенные в древнем Вавилоне за 3000 лет до н.э.;
- более «современный» вариант, ***с «косточками на проволоке»***, появился в Китае, примерно, за 500 лет до н.э.

«Механическая» эра (нулевое поколение) в эволюции ВТ связана с механическими, а позже — электромеханическими вычислительными устройствами.

Основным элементом механических устройств было *зубчатое колесо*.

Начиная с XX века, роль базового элемента переходит к *электромеханическому реле*.

1.3.2 Первое поколение (1937,1945 -1953,1955)

На роль первой в истории электронной вычислительной машины в разные периоды претендовало несколько разработок.
Общим у них было использование схем на базе электронно-вакуумных ламп вместо электромеханических реле.

Предполагалось, что электронные ключи будут значительно надежнее, поскольку в них отсутствуют движущиеся части.

Однако технология того времени была настолько несовершенной, что по надежности электронные лампы оказались ненамного лучше, чем реле.

У электронных компонентов имелось одно ***важное преимущество***: выполненные на них ключи могли переключаться примерно в тысячу раз быстрее своих электромеханических аналогов.

Сначала появилась *гарвардская архитектура ЭВМ*, отличительными признаками которой являются:

- хранилище инструкций и хранилище данных представляют собой разные физические устройства;
- канал инструкций и канал данных так же физически разделены.

Позже, появилась концепция машины *фон-неймановской архитектуры*, показанная на рисунке 1.6, особенностями которой является:

- общая и однотипная для команд и данных память ЭВМ (*принцип однородности* памяти);
- *принцип адресности* - память состоит из пронумерованных ячеек с произвольным доступом;
- *принцип программного управления* — АЛУ реализует набор команд, на основе котрых пишется программа;
- *принцип двоичного кодирования* - формате команды выделяются два поля: поле кода операции и поле адресов, которые можно представить в двоичном виде: последовательностью 0 и 1.

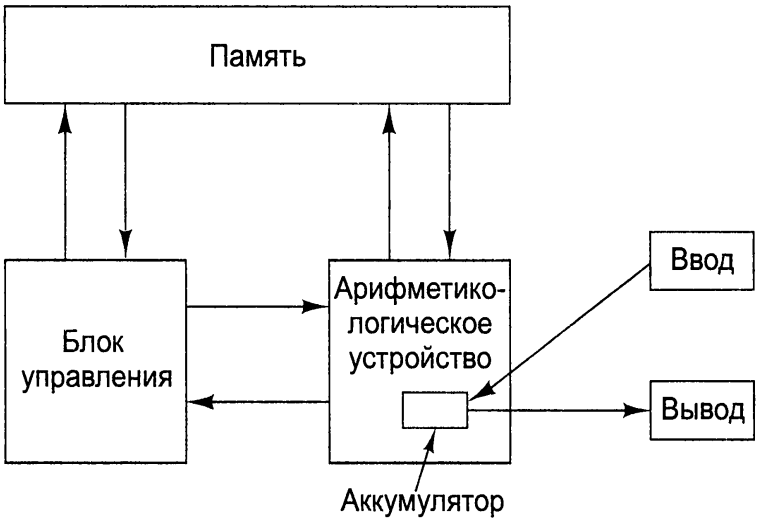

Рисунок 1.6 - Архитектура Фон-Неймана

1.3.3 Второе поколение (1954,1955 -1962,1965)

Второе поколение характеризуется *рядом достижений в элементной базе*, структуре и программном обеспечении.

Принято считать, что поводом для выделения нового поколения ВМ стали технологические изменения, и, главным образом, *переход от электронных ламп к полупроводниковым диодам и транзисторам* со временем переключения порядка 0,3 мс.

Появился язык Algol 60, предшественник языков C и Java.

Появились вычислительные машины PDP с шинной архитектурой, выпускаемые компанией DEC (рисунок 1.7):

Рисунок 1.7 - Архитектура ЭВМ с общей шиной

1.3.4 Третье поколение (1963,1965 – 1972,1980)

Третье поколение ознаменовалось *резким увеличением вычислительной мощности ВМ*, ставшим следствием больших успехов в области архитектуры, технологии и программного обеспечения.

Основные технологические достижения:

- переход от дискретных полупроводниковых элементов к *интегральным микросхемам*;
- начало применения *полупроводниковых запоминающих устройств*, начинающих вытеснять ЗУ на магнитных сердечниках.

Существенные изменения произошли и в **архитектуре ВМ**:

- *микропрограммирование* - как эффективная техника построения устройств управления сложных процессоров;
- *наступление эры конвейеризации* и **параллельной обработки**.

В области программного обеспечения определяющими вехами стали:

- первые *операционные системы*;
- реализация *режима разделения времени*.

Компания IBM заменила свои устаревшие модели 1401 и 7094 на новые машины System/360 (соответственно, модель 30 и модель 75).

1.3.5 Четвертое поколение (1972,1980 - 1984)

Связывается с переходом на интегральные микросхемы различной степени интеграции:

- *большой* (***large-scale integration***, LSI) , содержащие около *1000 транзисторов* на кристалле;
- *сверхбольшой* (***very large-scale integration***, VLSI), имеющие порядок *100000 транзисторов* на одном кристалле.

При таких уровнях интеграции стало возможным уместить в одну микросхему не только центральный процессор, но и вычислительную машину (ЦП, основную память и систему ввода/вывода).

Появились ***первые персональные компьютеры*** в виде *комплектов*.
Каждый комплект содержал:

- *печатную плату*;
- *набор интегральных схем*, обычно включающий схему Intel 8080, несколько кабелей, источник питания и иногда 8-дюймовый дисковод.
- *программное обеспечение не прилагалось*, требовалось писать самому.

Позднее, появилась *операционная система CP/M*, написанная Гарри Килдаллом (Gary Kildall) для Intel 8080.

Затем появились *MS-DOC, OS/2, Windows, …*

В 1981 году правительство Японии объявило о намерениях выделить национальным компаниям 500 миллионов долларов на разработку компьютеров *пятого поколения* на основе *технологий искусственного интеллекта*, которые должны были потеснить «*тугие на голову*» машины четвертого поколения.

1.3.6 Пятое поколение(1984-1990)

Главным поводом для выделения вычислительных систем второй половины 80-х годов в самостоятельное поколение стало *стремительное развитие ВС с сотнями процессоров*, ставшее побудительным мотивом для прогресса в области параллельных вычислений.

Ранее, параллелизм вычислений выражался лишь в виде:

- *конвейеризации*;
- *векторной обработки*;
- распределения работы между *небольшим числом процессоров*.

Вычислительные системы пятого поколения обеспечивают такое распределение задач по множеству процессоров, при котором *каждый из процессоров может выполнять задачу отдельного пользователя*.

1.3.7 Шестое поколение (1990-) – невидимые компьютеры

На ранних стадиях эволюции вычислительных средств смена поколений ассоциировалась с *революционными технологическими прорывами*:

- *Каждое из первых четырех поколений* имело четко выраженные отличительные признаки и вполне определенные хронологические рамки.

- *Последующее деление на поколения* уже не столь очевидно и может быть понятно лишь при ретроспективном взгляде на развитие вычислительной техники.

Пятое и шестое поколения в эволюции ВТ — это *отражение нового качества*, возникшего в результате последовательного накопления частных достижений, главным образом в *архитектуре вычислительных систем*.

С другой стороны, — *компьютеры начали стремительно уменьшаться*.

В 1993 году появилась *модель Apple Newton*, которая наглядно доказала, что компьютер можно уместить в корпусе размером с кассетный плеер. Все усложнял лишь **Рукописный ввод**, реализованный в *Newton*.
Впоследствии, пользовательский интерфейс подобных машин был изменен, и появилось то, что теперь называется *персональными электронными секретарями* – PDA.
PDA (*Personal Digital Assistants*), или просто карманный компьютер, был усовершенствован и приобрел широкую популярность.

По-настоящему революционной разработкой настоящего времени, которой придается очень большее значение, называется «*невидимые» компьютеры* — те, что встраиваются в бытовую технику, часы, банковские карточки и огромное количество других устройств.
Процессоры этого типа предусматривают широкие функциональные возможности и не менее широкий спектр вариантов применения за весьма умеренную цену.
Вопрос о том, можно ли свести эти микросхемы в одно полноценное поколение (а существуют они с 1970-х годов), *остается дискуссионным*.

1.4 Процессоры и шины ЭВМ

На уровене общей архитектуры - ЭВМ представляется в виде рисунка 1.8:

- центрального процессора (ЦП);
- основной памяти (ОП);
- устройств ввода/вывода (УВВ);
- системы шин.

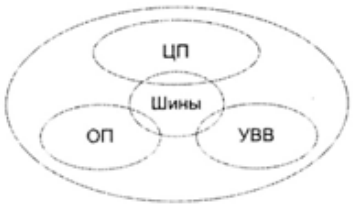

Рисунок 1.8 — Обшая архитектура компьютера

Шина - совокупность трактов, объединяющих между собой основные устройства ВМ: *центральный процессор, память и модули ввода/вывода*.

Шина обеспечивает *структуру взаимосвязей* вычислительной машины.

Революционное значение второго поколения ЭВМ (1954 – 1965 г.г.) – *концепция общей шины*, с которой связано *понятие общего адресного пространства ЭВМ*.

Шина Omnibus, разработанная компанией DEC, показала, что можно унифицировать взаимодействие основных блоков ЭВМ, выделив каналы:

- *адреса*;
- *данных*;
- *управления*.

В результате, архитектуру ЭВМ можно свести к *трем функциональным модулям*, показанным на рисунке 1.9.

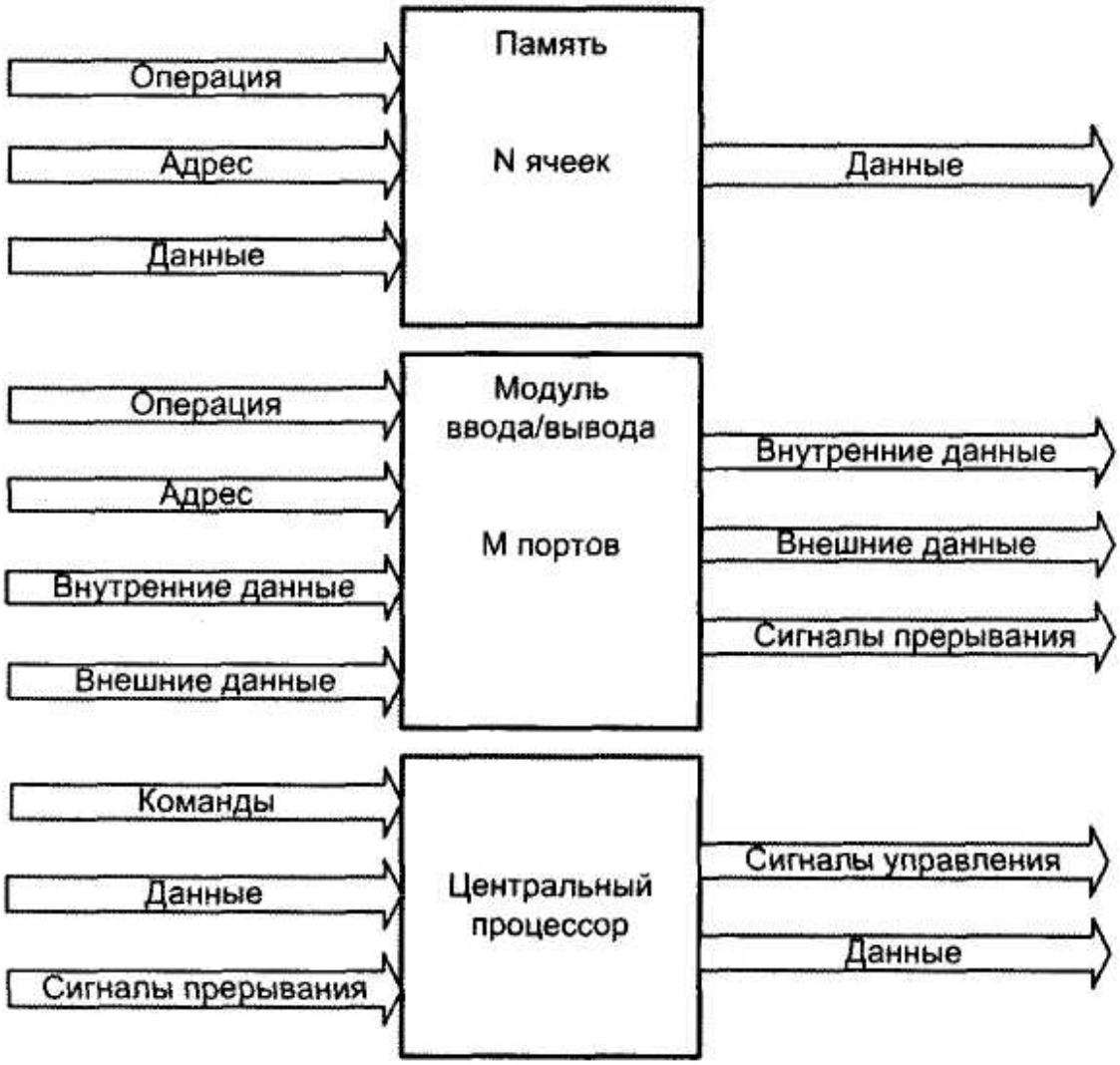

Рисунок 1.9 — Стандартизация компонентов ЭВМ

1.5 Структуры взаимосвязей устройств ЭВМ

С развитием средств вычислительной техники менялась структура взаимосвязей устройств ЭВМ.
В историческом плане, можно выделить четыре таких структуры, показанные на рисунке 1.10.

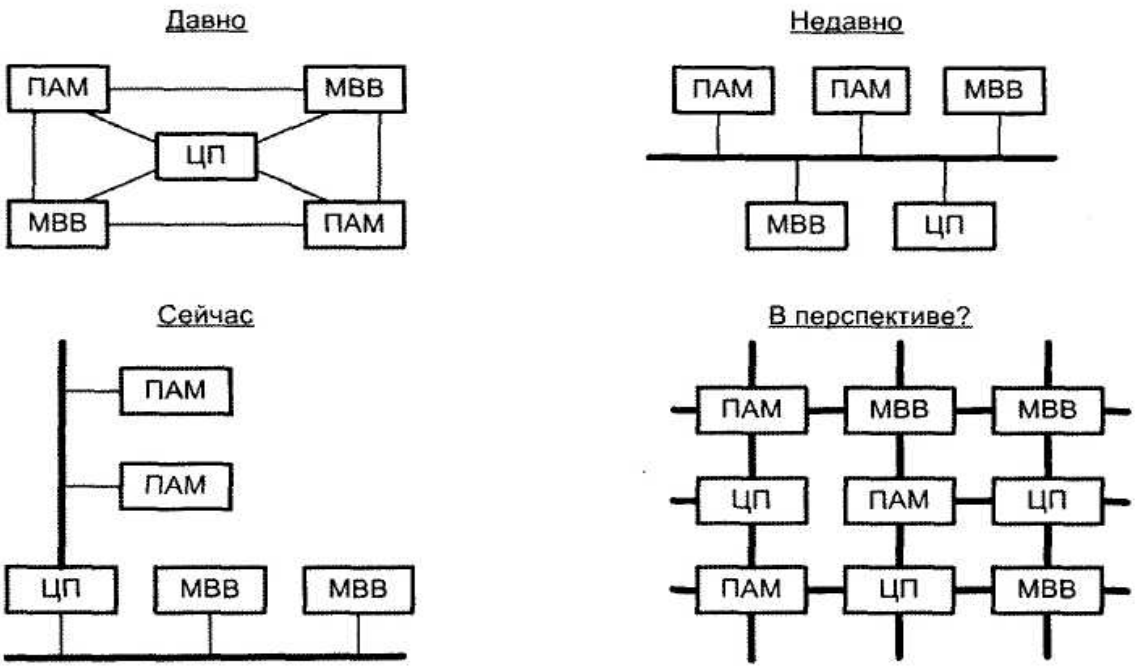

Рисунок 1.10 — Структура взаимосвязей устройств ЭВМ

На начальной стадии (давно) преобладали непосредственные связи между взаимодействующими устройствами ЭВМ.

С появлением мини-ЭВМ (недавно), и особенно первых микро-ЭВМ, все более популярной становится ***схема с одной общей шиной***. Последовавший за этим быстрый рост производительности практически всех устройств ЭВМ привел к *неспособности единственной шины справиться с возросшим трафиком*.

На смену единственной шине (сейчас) пришли ***структуры взаимосвязей на базе нескольких шин***.

Дальнейшие развитие (в перспективе) повышения производительности вычислений связаны с многопроцессорными вычислительными системами (***транспьютерная взаимосвязь***).
Способы взаимосвязей в таких системах значительно разнообразнее.

1.6 Пути повышения быстродействия (мощности) ЭВМ

Рассмотрим основные архитектурные моменты, повышающие быстродействие, а следовательно, мощность ЭВМ.

1.6.1 Параллельные сумматоры

Сумматором – называется комбинационное логическое устройство, предназначенное для выполнения операции арифметического сложения чисел в двоичном коде.
Таблица истинности для сложения одноразрядных двоичных кодов имеет вид:

Слаг. 1	***Слаг. 2***	***Сумма***	***Перенос***
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Следовательно, реализовать сумматор через **схемы ИЛИ** нельзя, поскольку существует отличие в четвертой строке и результат сложения не может быть представлен двоичным кодом, разрядность которого равна разряду.

Трудность в достижении быстродействия связана с тем, что *процесс распространения переносов носит последовательный характер*.
Следовательно, задержка в распространении переносов увеличивается при получении последнего переноса.

Частичное решение проблемы – *построение сумматоров с групповым переносом*.

1.6.2 Периферийные процессоры и шины

Цель такой конфигурации - повышение общей производительности системы за счет ***перераспределения выполняемых процессов*** между *центральным* и *периферийными* процессорами.

Из практики построения операционнных систем (ОС) известно, что:

- *для надежной работы* вычислительной системы необходима работа ядра ОС в *привелигерованном режиме процессора*;
- *частое переключение процессора* (из режима в режим) существенно *замедляет работу всей вычислительной системы*.

Чтобы устранить указанный недостаток, предлагается структура системы, показанная на рисунке 1.11.

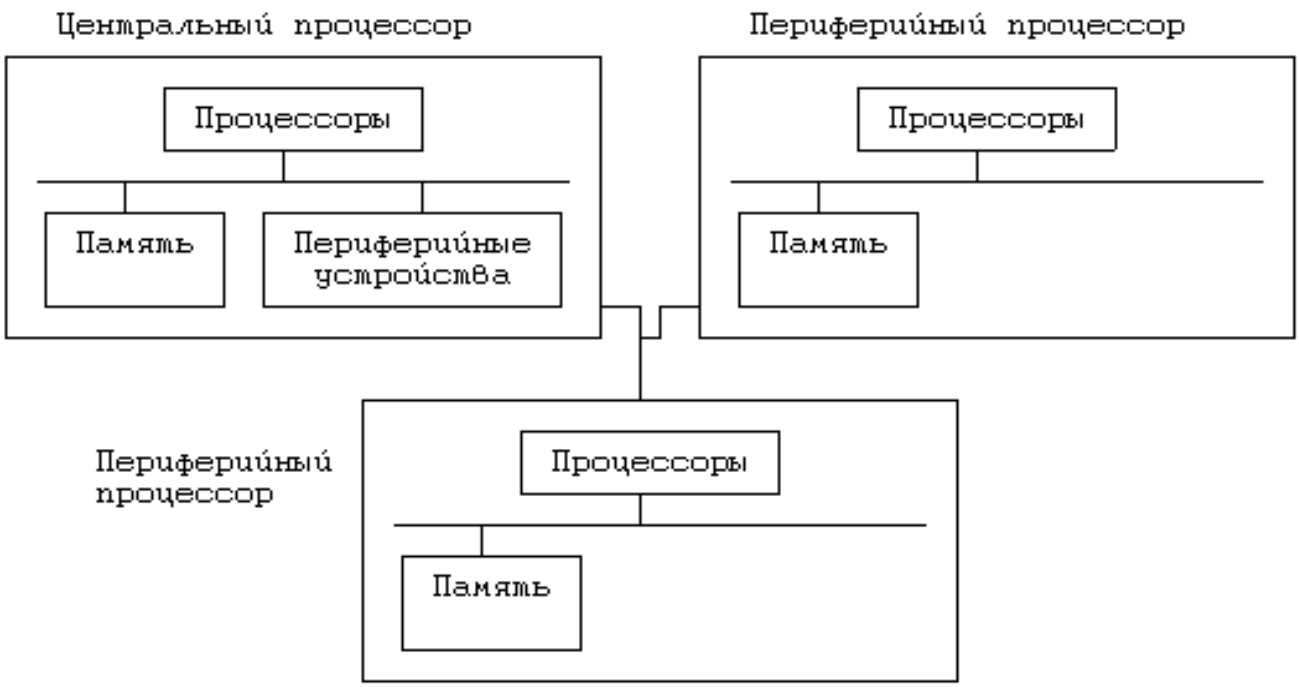

Рисунок 1.11 — Структура с периферийными процессорами

Особенности такой структуры:

- *Каждый периферийный процессор* имеет только средства связи с центральным процессором.
- *Файловая система и все устройства* находятся в распоряжении центрального процессора.

Каждый периферийный процессор содержит облегченный вариант операционной системы, *предназначенный для обработки локальных обращений*:

- к системе *управления прерываниями*,
- к системе *распределения памяти*,
- *к программам работы* с сетевыми протоколами и драйвером устройства связи с центральным процессором.

Каждый процессор имеет свою ***внутреннюю шину***.

Передача данных между процессорами производится через ***внешние шины***, обычно посредством ***передачи сообщений***, как показано на рисунке 1.12.

Формат сообщения

Признак вызова системной функции	Параметры системной функции	Данные о среде выполнения процесса	Составное имя -------- или -------- поток данных

Ответ

Результаты выполнения системной функции	Сообщение об ошибке	Номер сигнала	--------- Поток данных -------

Рисунок 1.12 – Структура сообщений через внешние шины

1.6.3 Многопрограммный режим работы

Многопрограммный режим работы системы обычно обеспечивается операционной системой (ОС).
Необходимость такого режима работы – эффективность использования вычислительной мощности процессора.

Что задерживает процессор?
Как правило:

1. *шина* – скорость шины ниже скорости работы процессора;
2. *память* – ожидание загрузки страниц;
3. *периферийные устройства*.

Все современные процессоры способны переключаться между программами, загруженными в память компьютера.
Эффективный режим переключения между программами – *сложная задача*.

1.6.4 Внедрение в процессор нескольких специальных АЛУ

АЛУ – арифметико-логическое устройство.
ALU *- arithmetic and logic unit.*

По своим функциям АЛУ является *операционным блоком, выполняющим микрооперации*, обеспечивающие приём из других устройств (например, памяти) операндов, их преобразование и выдачу результатов преобразования в другие устройства.
Само АЛУ управляется *управляющим блоком*.

Управляющий блок генерирует управляющие сигналы, которые инициируют выполнение в АЛУ определённых *микроопераций*.

По способу действия над операндами АЛУ делятся на *последовательные* и *параллельные*:

- *В последовательных АЛУ* операнды представляются в последовательном коде, а операции производятся последовательно во времени над их отдельными разрядами.
- *В параллельных АЛУ* операнды представляются параллельным кодом и операции совершаются параллельно во времени над всеми разрядами операндов.

По способу представления чисел различают АЛУ:

1) для чисел *с фиксированной точкой*;
2) для чисел *с плавающей точкой*;
3) для *десятичных чисел*.

По характеру использования элементов и узлов АЛУ делятся на *блочные* и *многофункциональные*.

- *В блочном АЛУ* операции над числами с фиксированной и плавающей точкой, десятичными числами и алфавитно-цифровыми полями выполняются в отдельных блоках. При этом повышается скорость работы, так как блоки могут параллельно выполнять соответствующие операции, но значительно возрастают затраты оборудования.
- *В многофункциональных АЛУ* операции для всех форм представления чисел выполняются одними и теми же схемами, которые коммутируются нужным образом в зависимости от требуемого режима работы.

1.6.5 Системы обработки данных (СОД)

Подведем итоги изложенного материала, уточнив классификацию на уровне СОД. *Система обработки данных* - совокупность технических средств и ПО, предназначенных для обработки данных.

По способам построения выделяются:

- одномашинные СОД;
- вычислительные комплексы;
- вычислительные системы;
- системы телеобработки;
- вычислительные сети.

Одномашинные СОД имеют:

1. *Ограничения по* надежности и производительности.
2. *Повышение производительности* достигается за счет совершенствования элементной технической базы (близок физический предел).

Существенной проблемой является использование общей шины, которая тормозит весь вычислительный процесс, поэтому современные ЭВМ используют специальный набор микросхем (*чипсет*), показанный на рисунке 1.13.

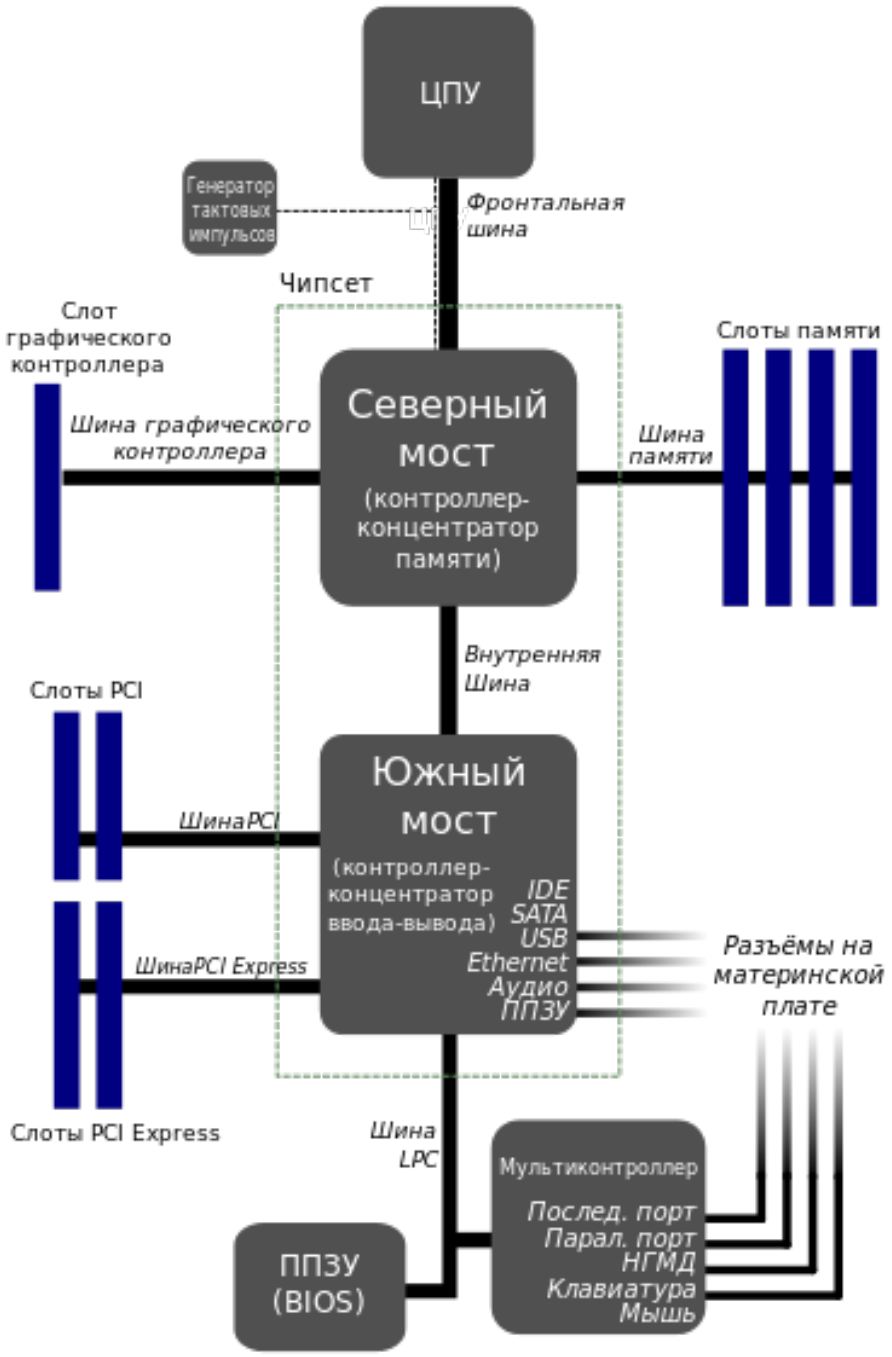

Рисунок 1.13 — Архитектура шин современной ЭВМ

Вычислительные комплексы призваны снять ограничения одномашинных СОД. *Вычислительный комплекс* - совокупность технических средств, включающих в себя *несколько ЭВМ* или *процессоров* и общесистемного (*базового*) ПО (*прикладное ПО не рассматривается*).

Развитие архитектур ВК идет в двух направлениях:

- *многомашинные* ВК;
- *многопроцессорные* ВК.

Многомашинные вычислительные комплексы имеют два способа объединения аппаратных средств ЭВМ:

- *косвенная связь ЭВМ*, показанная на рисунке 1.14, выполняется через внешние устройства хранения информации, что эффективно при создании высоконадежных систем *ассинхронной обработки информации*;
- *прямая связь ЭВМ*, показанная на рисунке 1.15, выполняется через специальные каналы связи (*адаптеры*), что эффективно присоздании систем, требующих *синхронное взаимодействие процессов обработки информации*.

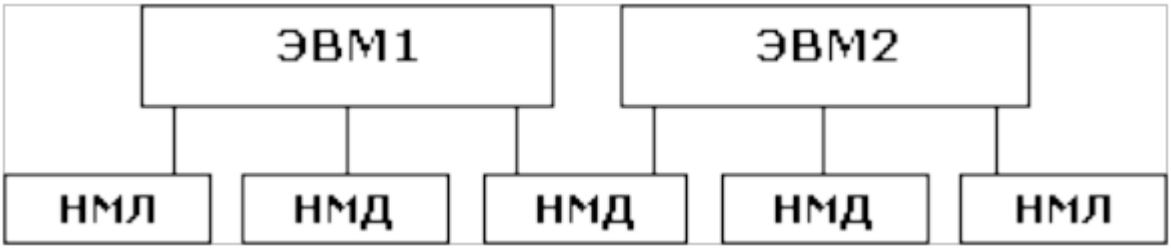

Рисунок 1.14 — Косвенная связь многомашинного ВК

Рисунок 1.15 — Прямая связь многомашинного ВК

Многопроцессорный вычислительный комплекс, показанный на рисунке 1.16, использует множество процессоров, объединенных *внутренней системной шиной* (*средва коммуникации*).

В настоящее время — это один из трендов развития архитектуры ЭВМ.

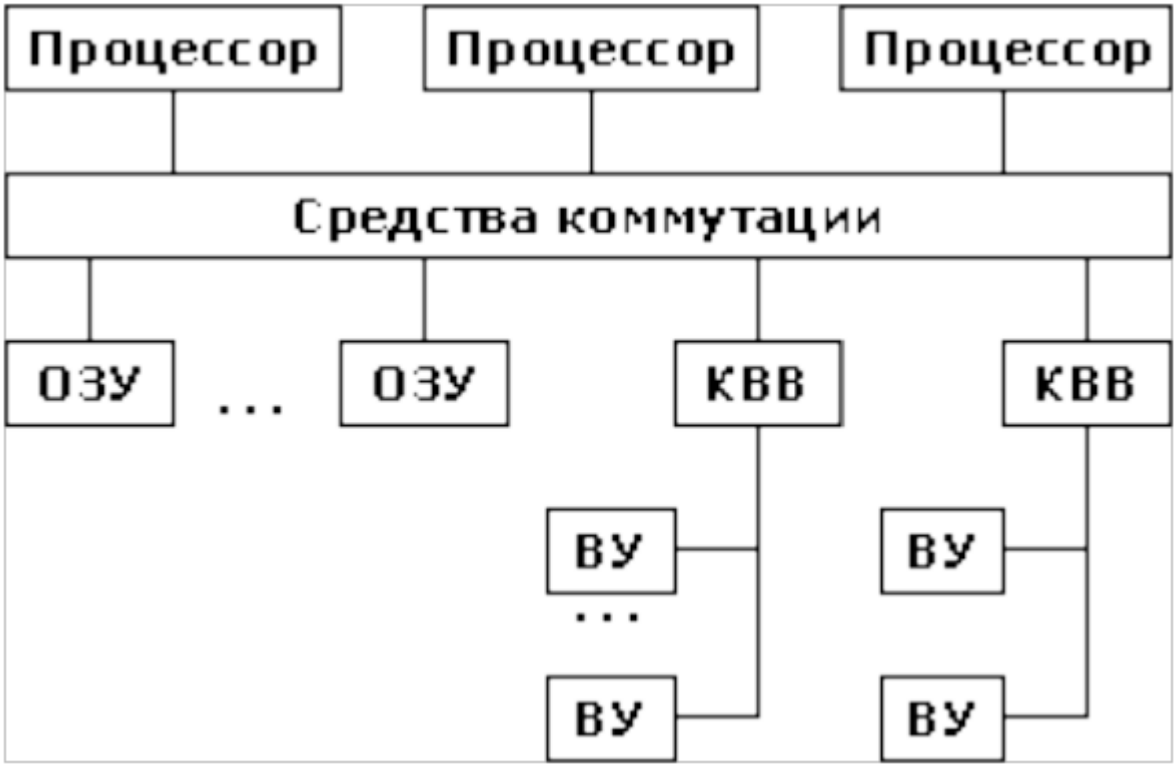

Рисунок 1.16 — Архитектура многопроцессорного ВК

Архитектурное развитие сосредоточенных систем, заканчиваемся моделью вычислительной системы.

Вычислительная система (ВС) - СОД, настроенная на решение задач конкретной области применения, поскольку ее технические средства и ПО направлены на решение определенного класса задач.

Способы ориентации этого направления:

- *ЭВМ или ВК общего применения* + проблемно-ориентированное ПО;
- *Специализированные* ЭВМ или ВК.
- *Наиболее перспективными* являются *адаптивные ВС*, имеющие динамическую структуру.

Многоуровневая архитектура вычислительной системы, показанная на рисунке 1.17, демонстрирует наличие системного ПО, предназначенного для организации взаимодействия между элементами ВС.

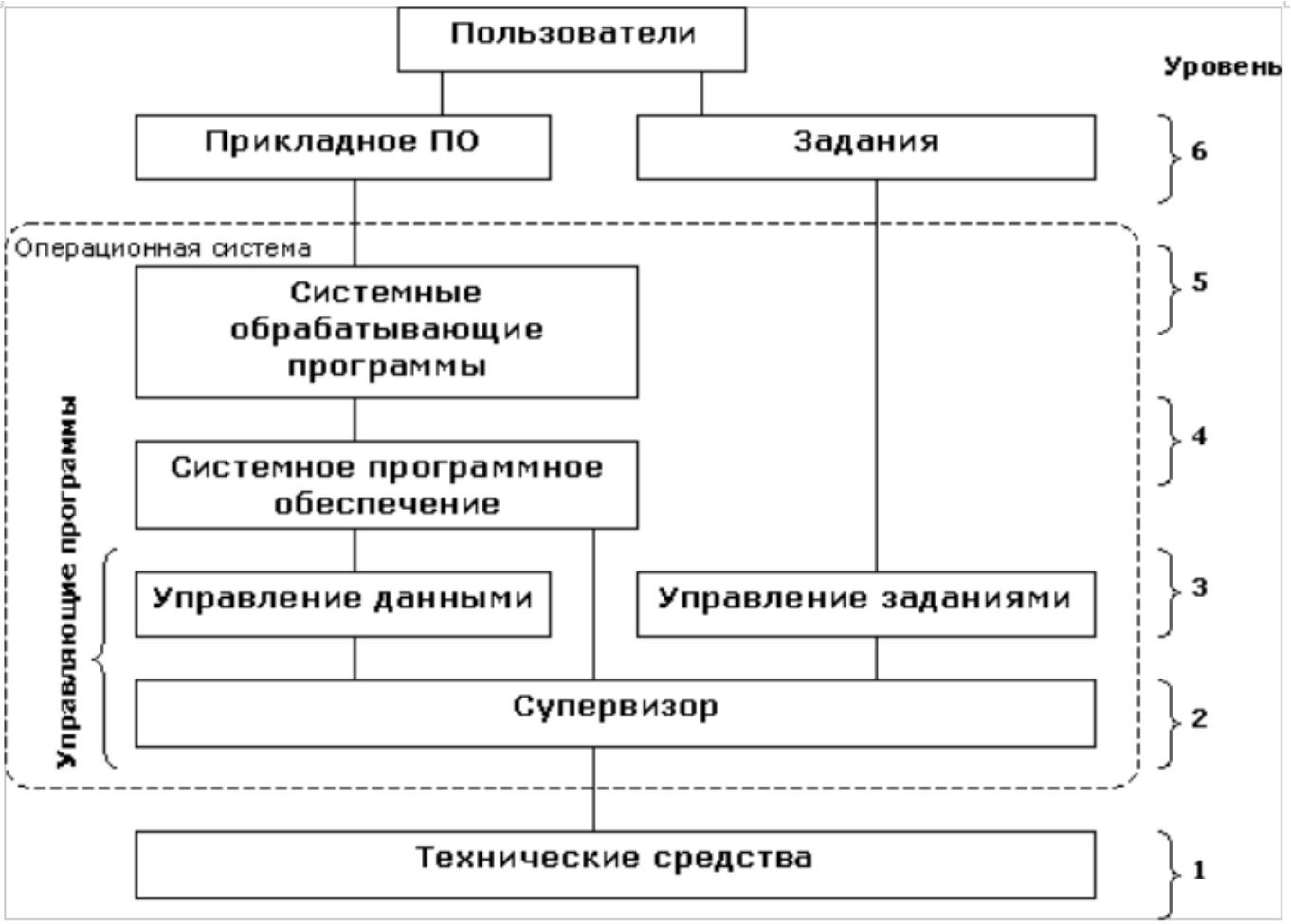

Рисунок 1.17 — Общая архитектура вычислительной системы

Классификация распределенных систем, в общем случае, ограничивается двумя архитектурами: *системами телеобработки* и *вычислительными сетями*.

Системы телеобработки (СТО) - системы, предназначенные для обработки данных, передаваемых по каналам связи.

Рисунок 1.18 — Схема СТО

Вычислительные сети, архитектура которых показана на рисунке 1.19, определяются как последовательное объединение следующих понятий:

- *каналы связи* — элементы объединения средств вычислительной техники;
- *устройства связи (УС)* — коммутаторы каналов связи;
- *Сеть передачи данных (СПД)* - устройства связи + каналы связи.
- *Сеть ЭВМ* - совокупность ЭВМ, объединенных сетью передачи данных.
- Сети делятся на ***глобальные*** и ***локальные***.
- **Вычислительная сеть** - СПД + сеть ЭВМ + ТС.

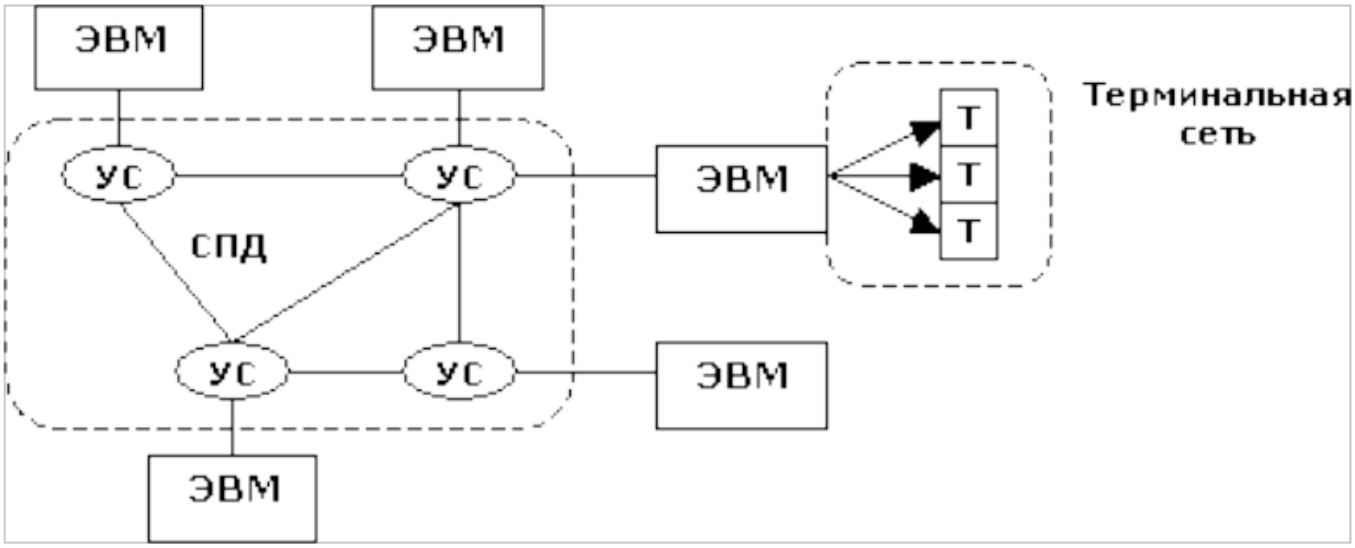

Рисунок 1.19 — Архитектура вычислительной сети

Таким образом, общая классификация СОД по способу построения, объединяется в структуру, показанную на рисунке 1.20.

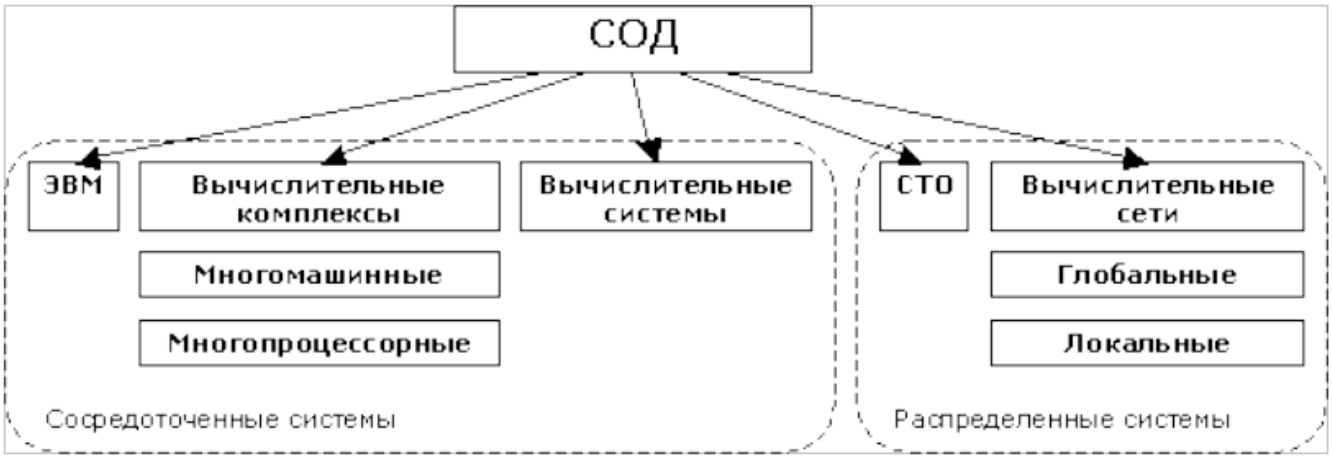

Рисунок 1.20 — Общая классификация СОД

2 Лабораторная работа №1

По традиции, первая лабораторная работа содержит описание среды ОС, в которой они проводятся.
Такая среда содержит:

- компьютеры учебных классов кафедры АСУ, полностью обеспечивающие индивидуальное обучение магистранта;
- общее программное обеспечение каждого компьютера с установленной ОС MS Windows, подключенной к общей локальной сети кафедры АСУ;
- специальное программное обеспечение: ОС УПК АСУ, содержащее обучающие программы и учебно-методический материал по изучаемой дисциплине.

Для выполнения данной и последующих лабораторных работ, магистрант должен изучить:

- состав тем вопросов и лабораторных работ изучаемой дисциплины, описанных во вспомогательном руководстве [1];
- местоположение и файловую структуру ***ОС УПК АСУ*** в пределах файловой системы ***ОС MS Windows***, используя методическое руководство [2];
- местоположение и состав учебно-методического материала данной и последующих лабораторных работ, используя файл *avk-selfworks.pdf*, размещенный на рабочем столе пользователя *upk*;
- подготовить личный накопитель flashUSB для выполнения лабораторных работ, хранения личных данных и отчета о проделанных работах.

2.1 Структура учебной части дистрибутива ОС УПК АСУ

Общая структура ОС УПК АСУ достаточно подробно описана в методическом руководстве [2], которым первоначально и следует воспользоваться.
Далее, необходимо изучить:

- *правила* общей загрузки ОС и вход в систему пользователем ***asu***;
- *местоположение* рабочей области студента на его личном flashUSB;
- *подключение* рабочей области пользователя ***upk*** к среде ОС;
- *переход* в сессию пользователя ***upk***;
- *правила завершения работы студента*, включающие: возврат в сессию пользователя ***asu***, отключение рабочей области, сохранение рабочей об-ласти на личном flashUSB;
- *отключение flashUSB* от среды ОС и завершение работы (выключение системы).

Рабочий стол сессии пользователя ***upk***, для изучаемой дисциплины, имеет стилизованную фоновую заставку, что показано на рисунке 2.1.

Рисунок 1.1 — Рабочий стол пользователя upk для изучаемой дисциплины «Архитектура вычислительных комплексов»

На рабочем столе имеются:

- значки *доступных файловых систем* (возможно отключенных);
- *avk-selfworks.pdf* — перечень тем и вопросов по обучаемой дисциплине;
- *Отчет.doc* — шаблон отчета студента;
- *upk_asu.pdf* — методическое руководство [2];
- значек перехода в *директорию с учебным материалом*;
- значек запуска *системы разработки* языка С на базе IDE Eclipse.

Основной учебный материал по данной дисциплине расположен в директории *~/Документы*. Здесь располагаются:

- *публикации* и другой учебный материал изучаемой дисциплины;
- *Темы* — каталог для размещения методических пособий по темам изучаемой дисциплины, содержащие как теоретический материал, так и описания лабораторных работ.

Замечание

В связи с обновлением версий ОС УПК АСУ и другого, сопутствующего, программного обеспечения, студенту следует убедиться в работоспособности всех компонет системы и, при необходимости, устранить имеющиеся недостатки, руководствуясь указаниями преподавателя.

2.2 Настройка сети ОС УПК АСУ

Современные компьютерные комплексы широко используют сети ЭВМ, в среде которых они установлены. В связи с этим, необходимо настроить сетевое ПО ОС УПК АСУ, используя знания, полученные при иучении курса «Современные операционные системы».

Дополнительно, магистр, проходящий обучение по дисциплине «*Архитектура вычислительных комплексов*» должен:

- *иметь личные данные* для авторизации на компьютерах кафедры АСУ;
- *уметь включать компьютеры*, расположенные в учебных классах кафедры АСУ, а также проходить процедуры авторизации на ОС УПК АСУ;
- *иметь личные данные* для авторизации ***на сервере кластера*** кафедры АСУ.

Замечание

Первоначальная установка дистрибутива ОС УПК АСУ не имеет привязки к локальной сети кафедры АСУ, поэтому необходимо научится делать это самостоятельно!

Современные дистрибутивы ОС имеют достаточно развитые средства настройки и работы с сетями ЭВМ. Используемый для обучения дистрибутив ОС УПК АСУ создан на базе дистрибутива ***Arch Linux***, который имеет сетевое ПО, основанное на сетевом пакете ***netctl***, совместимым с управляющим менеджером ОС — ***systemd***.

Замечание

Старые версии ОС Linux, для обозначения сетевых интерфейсов устройств, использовали *статические* имена:

- *ethX* — проводное устройство ethernet;
- *wlanX* — беспроводное устройство Wi-Fi,

где *X* — номер устройства, начинающийся со значения *0*.

Новые версии ОС Linux, для обозначения сетевых интерфейсов устройств, используют *динамические* имена, привязанные к узлам шины *pci* (см. рисунок 2.2):

- *enpYYYY* — проводное устройство ethernet;
- *wlpYYYY* — беспроводное устройство Wi-Fi,

где *YYYY* — обозначение узла шины *pci*.

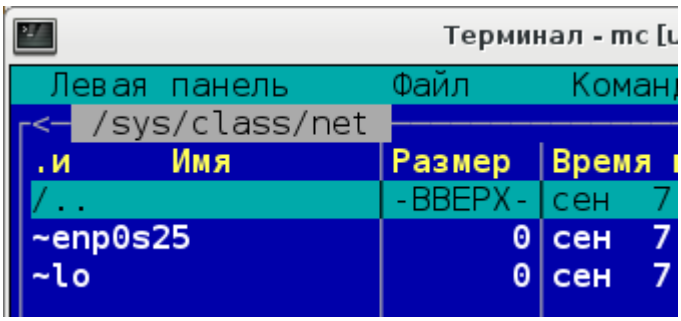

Рисунок 2.2 — Интерфесы компьютера с одним адаптером сети Ethernet

Для проверки готовности компьютера к работе в сети следует воспользоваться

утилитами ***ip*** или ***ifconfig***. На рисунке 2.3 представлен пример вывода утилиты ***ip***, которая показывает, что сетевой интерфейс ***enp0s25*** готов к работе.

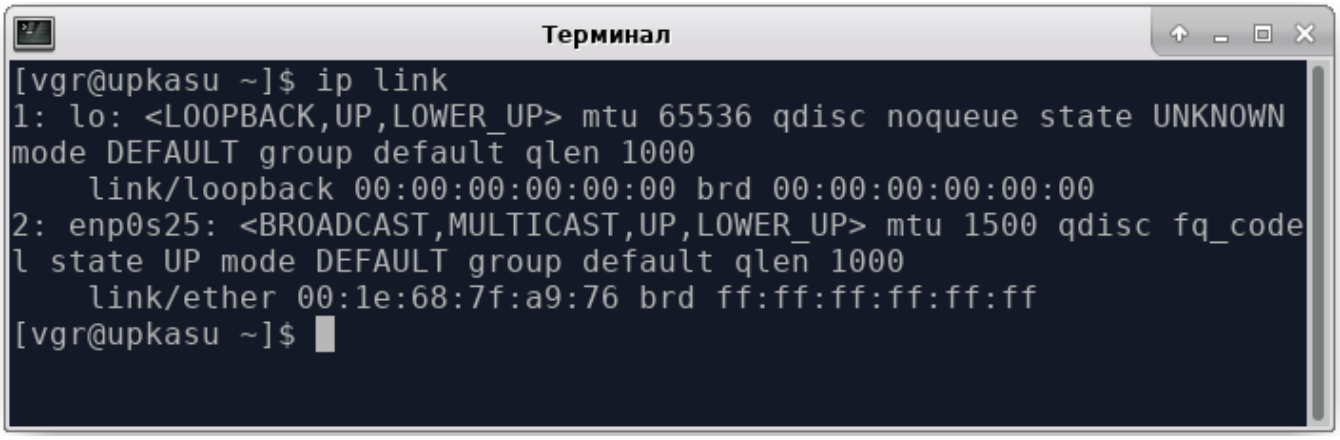

Рисунок 2.3 — Проверка наличия и рабочего состояния сетевых интерфейсов рабочей станции в среде ОС УПК АСУ

2.3 Контроль выполнения лабораторной работы №1

В результате выполнения лабораторной работы №1, магистрант должен:

- *проверить* запуск работу и полный комплект ПО по изучаемой дисциплине, а также соответствие установленного ПО требованиям методического пособия [2];
- *изучить* и *освоить* работу утилит ***ip*** и ***ifconfig***, а также правила настройки сети ОС УПК АСУ с помощью средств пакета ***netctl***;
- *описать* результаты проделанной работы в личном отчете;
- *отметить* завершение работы у преподавателя;
- *сохранить* резервную копию отчета.

3 Лабораторная работа №2

Цель лабораторной работы №2 — освоить консольный и графический интерфесы доступа к кластеру ЭВМ кафедры АСУ.

Учитывая поставленную цель, выполнение данной лабораторной работы выполняется под общим рабочим названием «***Работа со средой кластера кафедры АСУ***».

3.1 Работа со средой кластера ЭВМ кафедры АСУ

Цели и назначение кластера ЭВМ кафедры АСУ достаточно хорошо описаны в статье И.В. Бойченко [3]. Студенту следует познакомиться с этой статьей, чтобы иметь общее представление о конкретном варианте архитектуры вычислительного комплекса.

Все сетевое ПО кафедры АСУ настроенно таким образом, что компьютеры получают **IP**-адрес автоматически по протоколу **DHCP**.

При старте ОС УПК АСУ должен использоваться протокол DHCP, обеспечивающий получение адреса, маски и основного маршрутизатора компьютера.
Если используются другие альтернативные настройки, то результирующим критерием может быть «видимость» сайта: ***`cluster.asu.tusur.ru`***

Замечание

Если при запуске ОС УПК АСУ сеть не установилась, следует обратиться к преподавателю, обеспечивающему проведение лабораторных работ.

3.2 Настройка сетевого обеспечения ОС

Проверка работоспособности настроек сети проводится разными способами, один из вариантов которых - следующий:

- запустив из меню рабочего стола «Эмулятор терминала», можно выполнить команду ***ifconfig***; тогда появится информация, например, как показано на рисунке 3.1; сообщение, выделенное как ***enp0s25***, покажет IP адрес компьютера, его маску и другие параметры;
- убедившись подключении рабочей станции к сети, следует проверить доступность кластера ЭВМ кафедры АСУ; для этого необходимо, как показано на рисунке 3.2, выполнить команду **ping** по адресу кластера:

```
sudo ping cluster.asu.tusur.ru
```

Естественно, что результаты вывода будут зависеть от конкретного компьютера.

```
Терминал
[vgr@upkasu ~]$ ifconfig
enp0s25: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 78.140.20.13  netmask 255.255.255.0  broadcast 78.140.20.255
        inet6 fe80::21e:68ff:fe7f:a976  prefixlen 64  scopeid 0x20<link>
        ether 00:1e:68:7f:a9:76  txqueuelen 1000  (Ethernet)
        RX packets 8202  bytes 579700 (566.1 KiB)
        RX errors 0  dropped 165  overruns 0  frame 0
        TX packets 74  bytes 5735 (5.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 20  memory 0xefc00000-efc20000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[vgr@upkasu ~]$
```

Рисунок 3.1 - Сообщения команды ifconfig

```
Терминал
[vgr@upkasu ~]$ sudo ping cluster.asu.tusur.ru
PING cluster.asu.tusur.ru (88.204.75.195) 56(84) bytes of data.
From dl-ft-4.SUR.net.ru (88.204.72.250) icmp_seq=1 Packet filtered
From dl-ft-4.SUR.net.ru (88.204.72.250) icmp_seq=2 Packet filtered
From dl-ft-4.SUR.net.ru (88.204.72.250) icmp_seq=4 Packet filtered
From dl-ft-4.SUR.net.ru (88.204.72.250) icmp_seq=5 Packet filtered
From dl-ft-4.SUR.net.ru (88.204.72.250) icmp_seq=6 Packet filtered
^C
--- cluster.asu.tusur.ru ping statistics ---
6 packets transmitted, 0 received, +5 errors, 100% packet loss, time
5023ms

[vgr@upkasu ~]$
```

Рисунок 3.2 - Проверка наличия кластера командой ping

Убедившись, что кластер обнаружен, необходимо остановить работу **ping**, используя комбинацию клавиш ***<Ctrl>+C***.

Замечание

Если кластер ЭВМ не обнаруживается, как показано на рисунке 3.2, то следует обратиться к преподавателю, обеспечивающему проведение лабораторных работ.

3.3 Консольный доступ к кластеру

Консольный доступ к удаленным ЭВМ является базовой основой устаревшей технологии взаимодействия, известной как системы телекоммуникаций.
На базе этой технологии разрабатывалась модель терминала.

Учебная цель данного этапа, научиться *устанавливать защищенное соединение* с кластером и настраивать личные переменные среды окружения удаленной ОС.

Для обеспечения такого доступа к удаленному компьютеру, существует множество программ, как для ОС MS Windows, так и для ОС Linux. Наиболее известной из них является программа ***PuTTY***, которая имеется для обоих ОС.
Мы рассмотрим наиболее простой способ, который осуществляется с помощью «Эмулятора терминала» ОС Linux посредством вызова программы **ssh** (*security shell*). Чтобы выполнить такое подключение, необходимо набирать команду:

```
ssh <имя пользователя>@asu.local@cluster.asu.tusur.ru
```

где

- **cluster.asu.tusur.ru** - адрес главного компьютера кластера кафедры АСУ;
- **<имя пользователя>** - имя пользователя, под которым он имеет регистрацию на сервере кластера.

Пример такого действия показан на следующем рисунке 3.3:

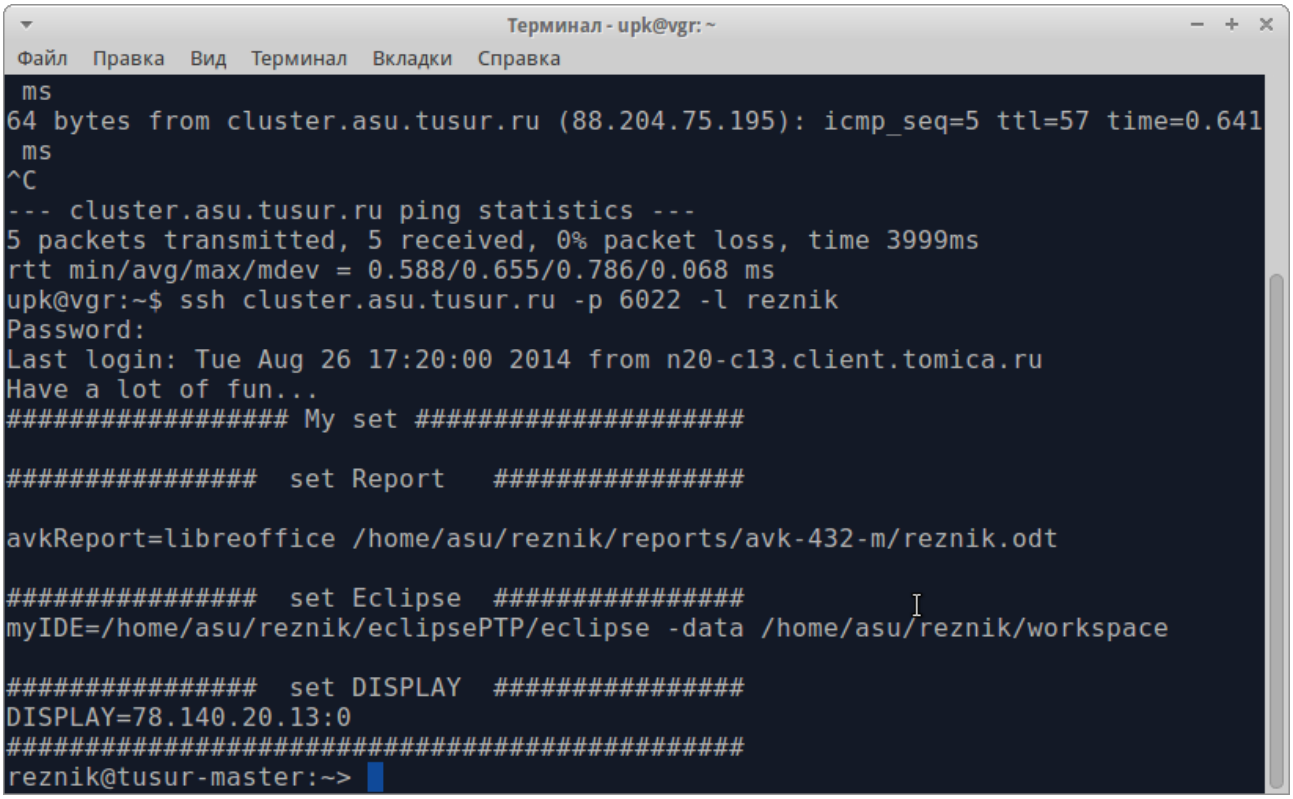

Рисунок 3.3 - Пример терминального соединения с кластером

Замечание

Удаленный сервер может запросить согласие на генерацию ключей. Нужно ответить: **yes**.

Затем, сервер запросит пароль, который студент должен правильно ввести и нажать клавишу «***Enter***».

Замечание

Имя и ***пароль*** для доступа на кластер кафедры АСУ студент получает у преподавателя, ведущего лабораторные занятия.

Если студент неправильно введет пароль, то удаленный сервер предложит повторный ввод. После нескольких неудачных попыток, сервер разрывает соединение.

Если пароль введен правильно, сервер выдает подсказку, характерную для консоли (терминала) ОС.

Далее, командой **mc** следует запустить ***Midnight Commander*** и изучить свою домашнюю директорию пользователя, как показано на рисунке 3.4.

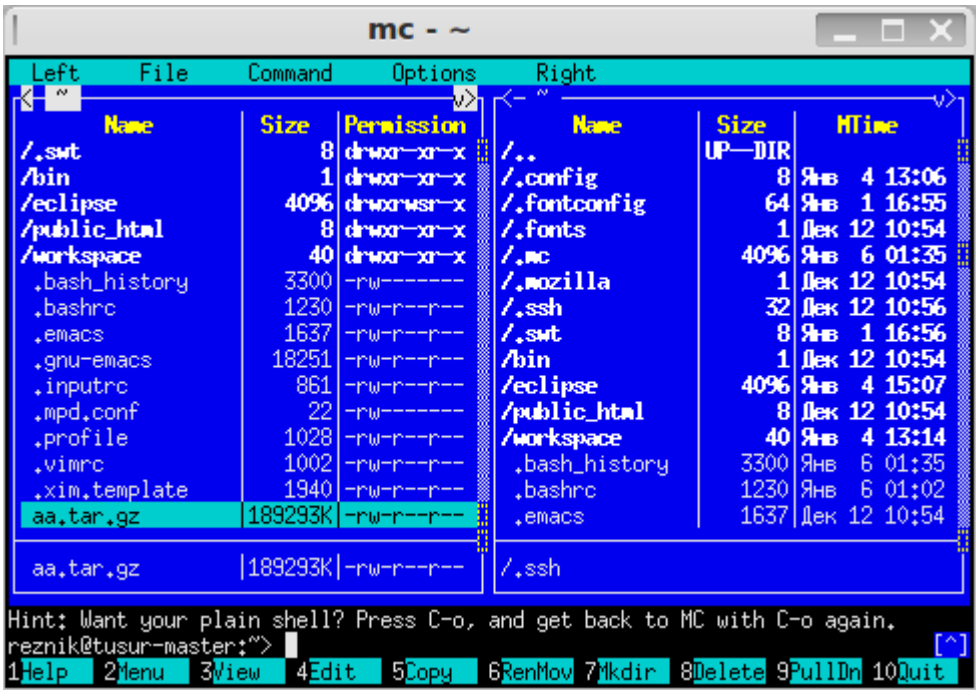

Рисунок 3.4 - Домашняя директория пользователя на кластере кафедры АСУ

Получив доступ к ЭВМ кластера, следует изучить среду пользователя, под именем которого был осуществлен вход.
Наиболее важными для изучения являются файлы сценариев ***.profile*** и ***.bashrc***.
Эти сценарии ***запускаются***, когда пользователь «***успешно сделал login***» в системе, и ***создают*** переменные окружения (переменные среды) ОС.
Основной список переменных среды студент изучал в ***теме 3*** бакалаврского курса

«Операционные системы».

Наиболее важными являются переменные, которые представлены в таблице 3.1.

Таблица 3.1 - Наиболее важные переменные среды ОС

DISPLAY	Указывает адрес и экран ***локальной*** или ***удаленной ЭВМ***, на которую буде осуществляться графический вывод приложений. *Подробно рассмотрена в следующем подразделе*.
HOME	Директория пользователя.
PATH	Список путей, в которых ищутся запускаемые программы.
USER	Имя пользователя, под которым осуществленн вход в систему.

Чтобы проверить значения этих переменных, можно воспользоваться командами:

echo $USER<Enter>
echo $<Enter>
echo $PATH<Enter>

В большинстве случаев, чтобы установить нужные значения переменных среды, достаточно провести редактирование файла ***.bashrc***.
Закончив его редактирование, необходимо выполнить команду:

. ./.bashrc

Затем, следует повторно проверить значения этих переменных и завершить удаленный доступ к кластеру.

Замечание

Закончив изучение файловой системы удаленного компьютера, студент должен корректно выйти из удаленного соединения (***просто закрывать окно терминала не следует!***).
Для этого, обычно достаточно последовательности команд **exit**.
Например, в рассмотренном случае достаточно двух команд **exit:**

- ***первая*** — закроет приложение ***Midnigth Commander***;
- ***вторая*** — разорвет соединение с удаленным сервером.

3.4 Графический доступ к кластеру

Учебная цель этого подраздела:

- *изучить* архитектуру X Window System;
- *научится* запускать X-сервер с нужными параметрами доступа;
- *проверить* возможность подключения к X-серверу программ, запущенных на сервере кластера.

Замечание

*В отличие от MS Windows, имеющей встроенное в ядро ОС графическое ПО, без которого она не может работать, Linux для этой цели имеет набор приложений, взаимодействующих на основе модели «**клиент/сервер**».*

*Базовая часть графического ПО Linux назвается **X Window System** (или просто - **X Window**).*

***Начиная с 1988 года**, этот стандарт поддерживался консорциумом X, созданным с целью унификации графического интерфейса для ОС UNIX.*

***В 1997 году** консорциум X был преобразован в X Open Group (**http://www.x.org**).*

X Window — достаточно сложная система и подробно описана большим количеством первоисточников. Мы рассмотрим только основные элементы архитектуры этой системы, опираясь на материал Интернет-статьи «Графический интерфейс Linux» [4].
Согласно этой статье, общая архитектура X Window может быть представлена рисунком 3.5.

«Сердцем» этой графической системы является программа X-сервер, который через драйверы устройств взаимодействует с видеоплатой, клавиатурой, мышью и монитором компьютера. Именно X-сервер устанавливает и переключает графические режимы видеплаты, рисует элементы изображений, определяет координаты мыши и формирует программные прерывания при нажатии кнопок мыши и клавиатуры. Все остальные программы, включая менеджер окон, взаимодействуют с X-сервером по особому протоколу, который называется X-протокол, или протокол сетевой связи (X Network Protocol).

Для написания программ, поддерживающих ***X-протокол***, имеется базовая библиотека ***X-lib***. На основе этой библиотеки пишутся дополнительные графические библиотеки более высокого уровня, например, ***GTk+***, ***Qt***, ***Motif*** и другие. Обычно, менеджеры окон, рабочие столы и сложные графические приложения пишутся с использованием этих библиотек.

Рисунок 3.5 - Общая архитектура X Window

В общем случае, X-сервер обрабатывает ***4 типа сообщений***:

- **Запрос** – клиент требует нарисовать что-либо в окне или запрашивает у сервера информацию;
- **Ответ** – сервер отвечает на запрос;
- **Событие** – сервер сообщает клиенту о событии (например, о нажатии клавиши пользователем);
- **Ошибка** – сервер сообщает об ошибке.

Замечание

Важной для нас особенностью является возможность X-сервера работать на стеке протоколов TCP/IP с программами, запущенными на удаленных компьютерах. Обычно используется асинхронная связь (протокол UDP), но возможна и синхронная связь по протоколу TCP, которая работает в 30 раз медленнее.

Кроме того, на компьютере может быть запущено несколько X-серверов, которые выводят графическую информацию на разные дисплеи с разными номерами: 0, 1, 3 и т.д.

Чтобы прикладные программы знали на каком компьютере и на каком дисплее находится X-сервер, используется переменная окружения ***DISPLAY***, которая содержит информацию в формате (см. также таблицу 3.2):

```
<IP-адрес X-сервера>:<номер дисплея>.<номер экрана дисплея>
```

Таблица 3.2 - Примеры задания значений переменной DISPLAY

Значение переменной DISPLAY	**Номер порта соединения**	**Пояснение**
:0	--	X-сервер на локальном компютере, дисплей №0, экран №0
:1.0	--	X-сервер на локальном компютере, дисплей №1, экран №0
asu.tusur.ru:4	6004	X-сервер на компютере asu.tusur.ru, дисплей №4, экран №0, номер UDP-порта - 6004
192.168.1.17:2	6002	X-сервер на компютере 192.168.1.17, дисплей №0, экран №0, номер UDP-порта - 6002

Многие прикладные программы имеют специальный параметр ***-display***, после которого можно указать указать ***IP-адрес*** и ***номер дисплея*** вывода.
Например, запуск программы **xterm** можно осуществить в виде:

```
xterm -display foo:1<Enter>
```

выведет консольную программу на дисплей **№1** компьютера **foo**.

Если на компьютере запущено несколько X-серверов, то следует пользоваться «горячими клавишами»:

- <Ctrl>+<Alt>+<Backspact> - приведет к остановке X-сервера;
- <Ctrl>+<Alt>+<F#> - переключение между консолями Linux.

Замечание

Запустить X-сервер может только пользователь **root**. Запуск производится с помощью программы **xinit**, общий формат запуска которой имеет вид:

```
xinit <клиент> -- <дисплей> <сервер>
```

где <**клиент**> - параметры, определяющие клиентскую часть;
<**дисплей**> - адрес компьютера и номер дисплея, где загрузится X-сервер;
<**сервер**> - серверная часть параметров.

Следует также заметить, что структура запуска программы **xinit** — довольно сложна и требует высокой квалификации.

На практике используются два основных варианта запуска X-сервера:

- **Вариант 1**. X-сервер запускается менеджером дисплея, например, ***gdm*** (*gnome display manadger*), во время загрузки ОС в графическом режиме.
- **Вариант 2**. X-сервер запускается пользователем из текстового режима посредством скрипта ***startx***.

В любом из используемых вариантов, конечным результатом этих усилий будет

запуск программы **xinit** с нужными параметрами.
Здесь следует учесть, что основными файлами, содержащими параметры запуска X-сервера, являются:

- **/etc/X11/Xsession;**
- **/etc/xinit/xinitrc;**
- **/etc/xinit/xserverrc.**

Замечание

Если эти файлы скопировать в домашнюю директорию пользователя с именами **.Xsession**, **.xinitrc** и **.xserverrc**, а затем отредактировать нужным образом, то X-сервер будет запускаться с параметрами указанными в этих файлах.

Кроме перечисленных свойств, X-сервер имеет набор механизмов различного уровня защиты от несанкционированного доступа. Современные дистрибутивы Linux, такие как ОС Xubuntu, поставляются с параметрами, запрещающими X-серверу подключать программы удаленных компьютеров.

Замечание

Поскольку подробное обсуждение средств защиты выходит за рамки данного пособия, студенту следует использовать интерактивное справочное руководство **man** с параметром **Xsecurity**.

В дистрибутивах ***Linux Xubuntu*** используется оконный менеджер ***LightDM***.
Чтобы разрешить подключение сети, необходимо:

- открыть файл **/etc/lightdm/lightdm.conf**;
- в секцию **[SeatDefaults]** добавить строку:

```
xserver-allow-tcp=true
```

- создать еще одну секцию:

```
[XDMCPServer]
enabled=true
```

Перезапустить оконный менеджер командой:

```
sudo restart lightdm
```

Замечание

В дистрибутиве ОС УПК АСУ, ***все необходимые настройки X-сервера уже проведены***.

Изучив основные элементы архитектуры X Window System, перейдем к пошаговому изложению операций, которые студент должен выполнять на каждой лабораторной работе от момента запуска дистрибутива ОС УПК АСУ до момента запуска на кластере кафедры АСУ интегрированной среды разработки IDE Eclipse.

Шаг 1. Загрузить ОС УПК АСУ, в графическом режиме, и выполнить все необходимые действия по настройке сети компьютера согласно подразделам 3.1 и 3.2 данного руководства.

Шаг 2. С помощью команды ***ifconfig*** или ***ip***, определить и запомнить **IP**-адрес компьютера, на котором проводится лабораторная работа.

Шаг 3. В консоли, от имени пользователя **root**, следует выполнить команду:

```
xhost +cluster.asu.tusur.ru
```

которая разрешит пользователям компьютера **cluster.asu.tusur.ru** подключаться к вашему X-серверу.

Замечание

Если в команде **xhost** заменить '+' на '-', то доступ с компьютера **cluster.asu.tusur.ru** к вашему X-серверу будет запрещен.

Соответственно, для проверки, что на компьютере разрешено подключение к X-серверу, используется команда:

```
xhost
```

Эта команда покажет список и уровень защищенности всех разрешенных соединений.

Шаг 4. Командой **ssh** подключиться к серверу **cluster.asu.tusur.ru**, используя правила, изложенные в разделе 3.2 данного пособия.

Шаг 5. Находясь на консоли удаленного сервера, отредактировать файл **.bashrc**, установив переменную ***DISPLAY***, как показано на рисунке 3.6:

Замечание

В качестве параметра IP-адреса, студент должен указать **IP-адрес** своего компьютера, определенного ранее на **шаге 1**.

Переменную **DISPLAY** всегда нужно проверять, поскольку IP-адрес компьютера студента может изменяться.

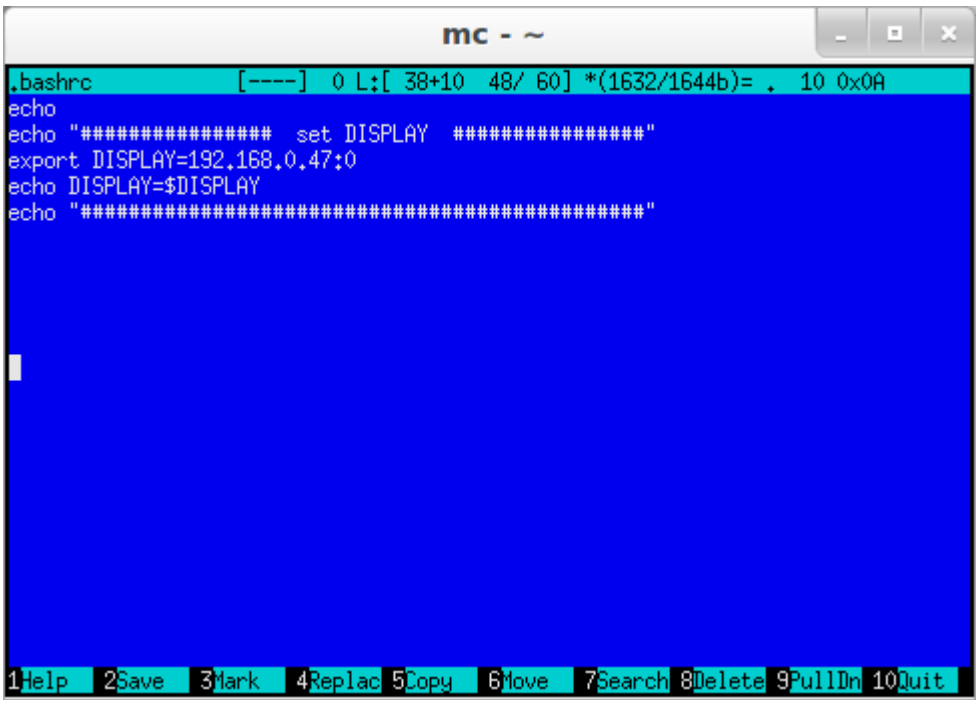

Рисунок 3.6 - Редактирование системной переменной DISPLAY

Далее, следует выйти из редактора и перезапустить скрипт **.bashrc**, как показано на рисунке 3.7, чтобы убедиться в правильности установки переменной **DISPLAY**.

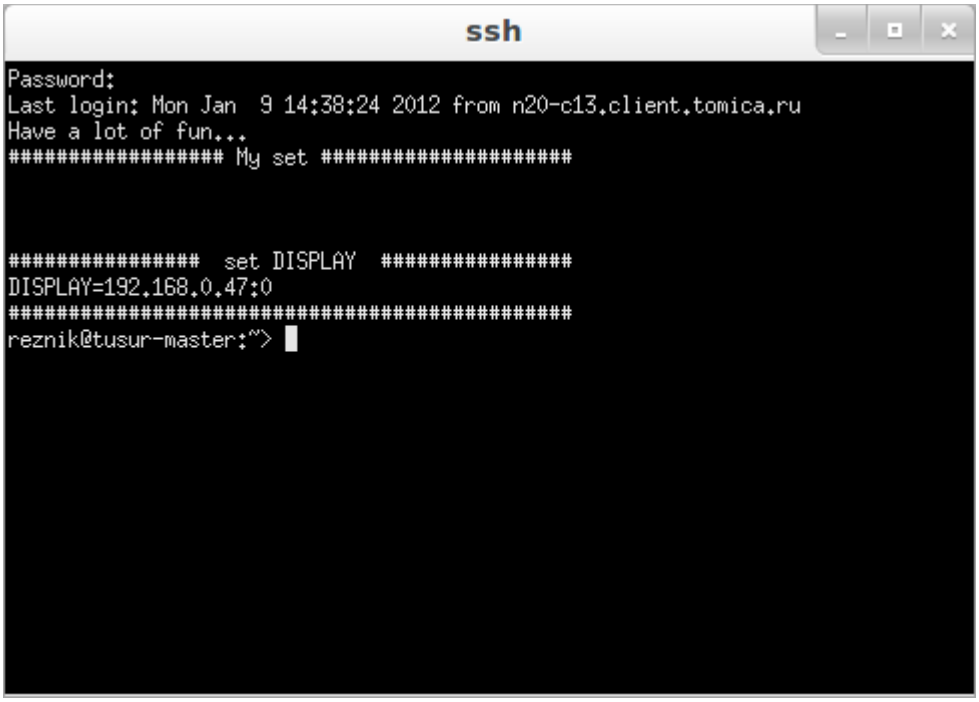

Рисунок 3.7 - Проверка установки переменной DISPLAY

Шаг 6. Нужно дополнительно убедиться в правильности настроек, запустив на удаленном компьютере команду **xterm**. При правильных настройках, которые были проведены на предыдущих шагах, на рабочем столе должно появиться новое окно консоли, озаглавленное ***xterm***.

Выйти из этой консоли можно командой **exit**.

Замечание

Все перечисленные ***шесть шагов*** следует выполнять каждый раз перед самым ***началом выполнения работ на кластере кафедры АСУ***.

4 Лабораторная работа №3

Обычно, разработка приложений осуществляется на локальном компьютере. Для этой цели используются интегрированные среды разработки (***IDE***).
В ряде случаев, такая разработка затруднительна или невозможна по различным причинам. Тогда используются различные варианты удаленной разработки.

Цель лабораторной работы №3 — освоить удаленную разработку приложений на кластере ЭВМ кафедры АСУ.

Учитывая поставленную цель, выполнение данной лабораторной работы выполняется под общим рабочим названием «***Удаленная разработка приложений***».

Обоснованием указанной цели служат две причины:

- невозможность изпользования на локальной рабочей станции инструмента ***Eclipse PTP*** — платформы для разработки параллельных вычислений;
- возможность организации удаленной разрабоки ПО с использованием сетевых возможностей ***X Window System*** доступных в ОС UNIX/Linux.

4.1 Удаленная разработка приложений

Основное учебное задание:

- подключиться к кластеру кафедры АСУ и запустить интегрированную среду разработки (IDE) EclipsePTP;
- создать типовой проект на языке С, который выводит на консоль строку сообщения: **«Hello, world!!!»**.

Навыки, полученные студентом во время выполнения данной лабораторной работы, обеспечат его умениями, необходимыми для реализации программного обеспечения, требующего параллельных вычислений.

Запустив систему ОС УПК АСУ, настроив и подключившись к кластеру кафедры АСУ, как описано в лабораторных работах №1 и №2, студент запускает Midnight Commander и переходит к работе с интегрированной средой разработки IDE Eclipse.
Для этого, перемещаясь по дереву файловой системы, студент заходит в директорию ***/home/asu/reznik/eclipsePTP/*** и запускает на выполнение файл ***eclipse***.
В результате, на экране монитора появится окно ***«Select a workspase»***, которое предлагает определить директорию рабочей области (см. рисунок 4.1).
Далее, следует отредактировать строку ввода «**Workspace:**», придерживаясь формата ***<Домашняя директория студента>/workspace*** и, воздействуя на кнопку **OK**... Загрузить среду разработки Eclipce.

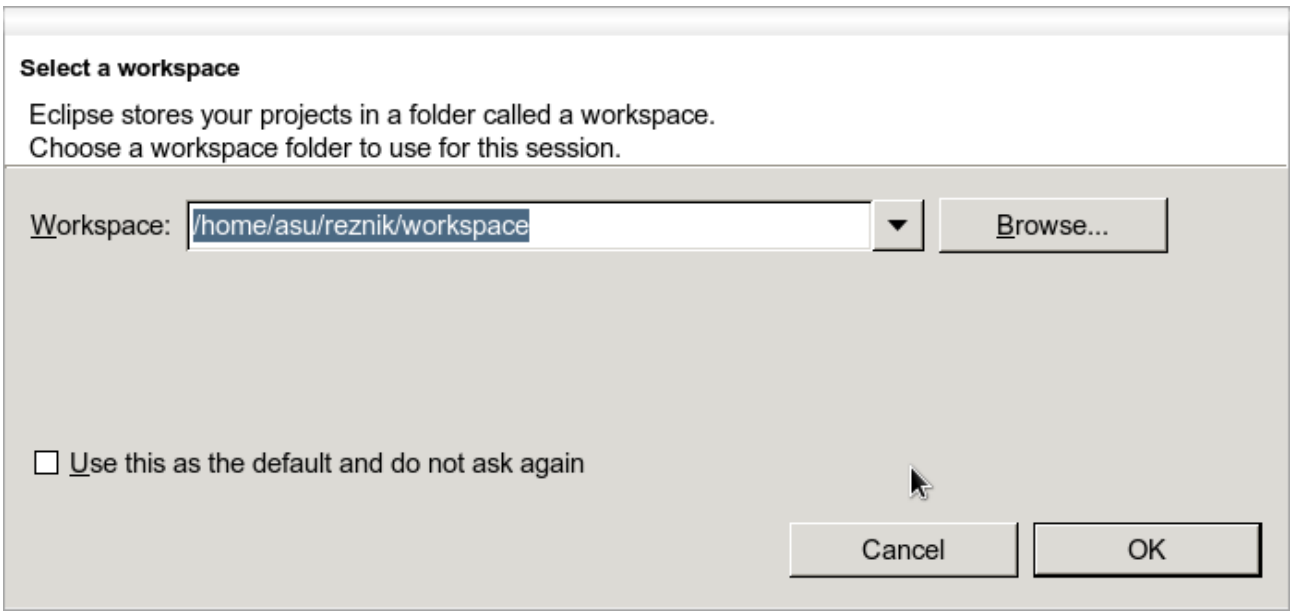

Рисунок 4.1 - Окно выбора директории проектов Eclipse

Замечание

Если рабочая директория среды разработки выбирается первый раз, то появится окно с предложением выбрать разделы инструментальной среды.
Следует выбрать раздел «**Workbench**» и появится ***окно среды разработки***.
При последующих выборах этой директории, рабочая среда разработки Workbench будет показана сразу.

Ниже, на рисунках 3.2 и 3.3, показаны оба окна запуска среды EclipsePTP.

Приступаем к созданию контрольного примера на языке **C**, учитывая что интегрированная среда разработки IDE EclipsePTP, как и большинство других современных сред разработки, делит создаваемые программные продукты на проекты. ***Все создаваемые проекты***, в пределах рабочей области Workspace, ***должны иметь уникальные имена***.

С целью упорядочения процесса проведения занятий, а также последующей отчетности и контроля, студент на каждой лабораторной работе создает один проект. Соответственно, проекты именуются по порядку: lab1, lab2, lab3 и т.д.

Задание на разработу ПО

В данной лабораторной работе, студент должен создать проект с именем **lab1** для типовой программы на языке C.
Эта программа должна выводить на стандартное устройство вывода (консоль) текст:

```
Hello, world!!! <Фамилия студента> <Имя студента>
```

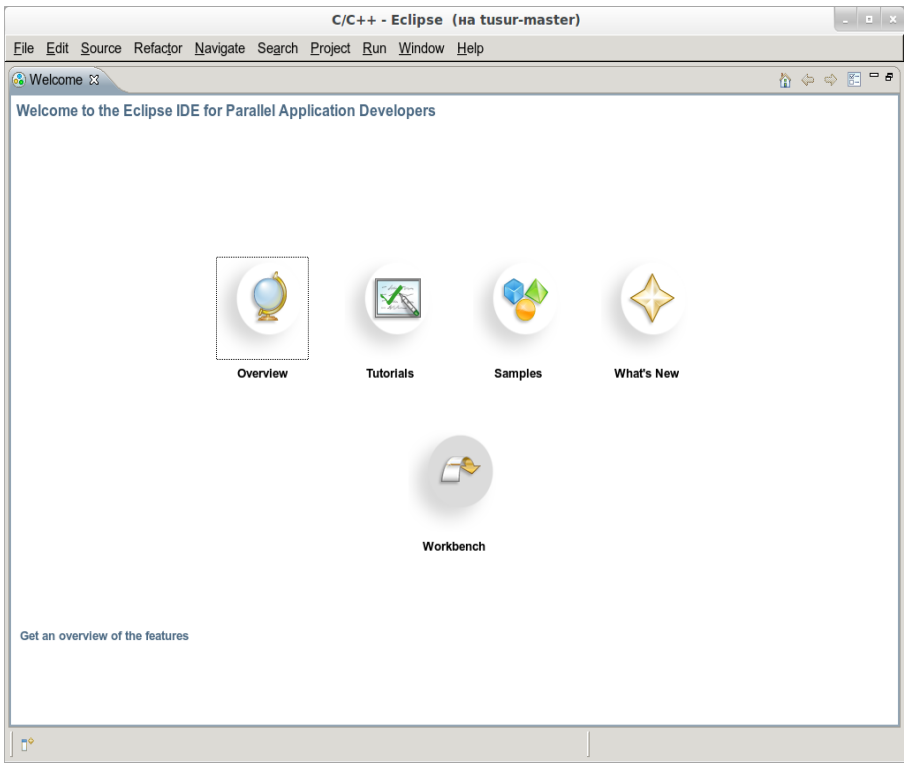

Рисунок 4.2 - Окно Eclipse при первом создании рабочей области

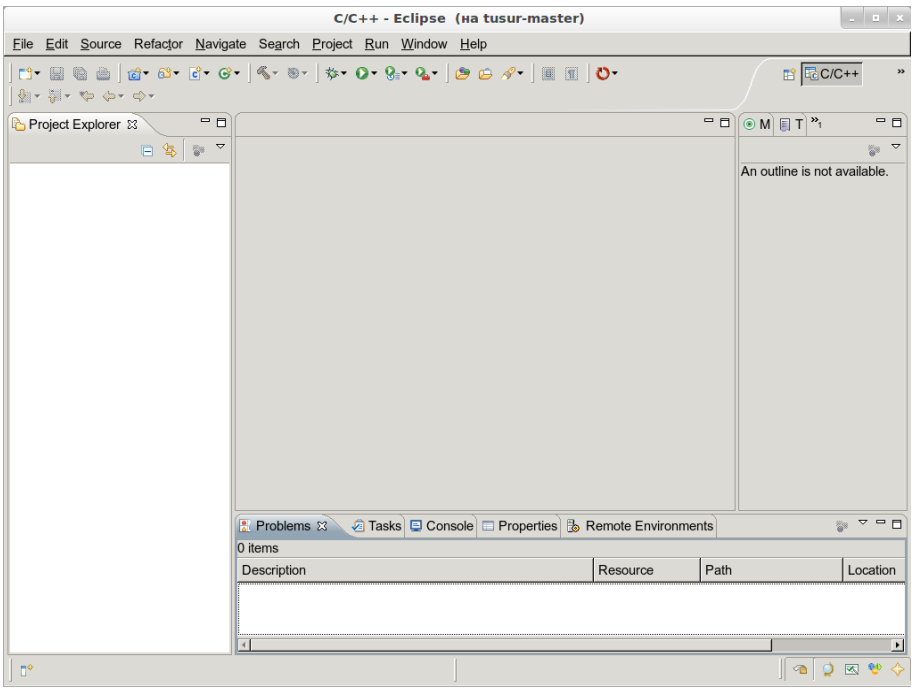

Рисунок 4.3 - Рабочее окно среды разработки Eclipse

Учебная цель данного задания:

- *закрепление* знаний работы с интегрированной IDE Eclipse;
- *уточнение настроек рабочей среды разработки* под локальную ОС;
- *практическое закрепление* технологического цикла удаленной разработки ПО.

Чтобы максимально упростить рутинные действия, при запуске IDE Eclipse, следует дополнительно отредактировать файл **.bashrc**, как показано на рисунке 3.4. **Тогда** запуск Eclipse можно будет производить с консоли командой: **$myIDE**

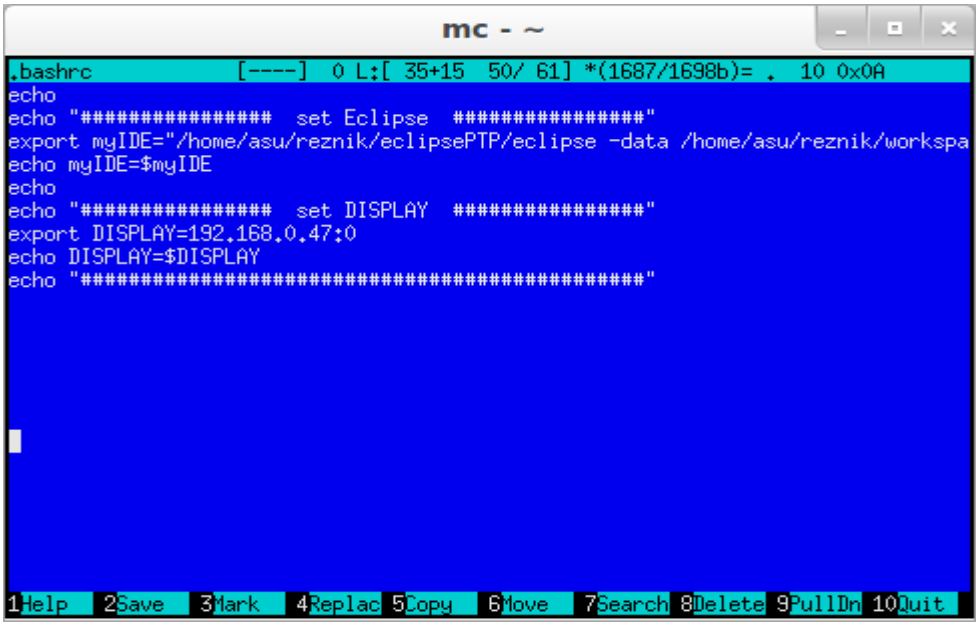

Рисунок 4.4 - Редактирование переменной среды myIDE

Замечание

Программа ***eclipse***, в качестве значения ключа **-data** принимает значение директории, которая будет рассматриваться как ***Workspace***.
Студент должен правильно установить значение этого параметра.

Поскольку IDE Eclipse — сложная модульная система, первоначально созданная для разработки ПО на языке Java, то основная масса литературы, посвященная Eclipse и примерам разработки на ней, содержит контекст для языка Java.

В последнее время, некогда единый проект интегрированной среды Eclipse, возглавляемый Eclipse Foudation, стал делиться на специализированные проекты, среди которых появилась среда разработки для ПО на C/C++ (среда CDT — C Development Tool).

Позже появился еще один специализированный проект, поддерживащий разработу приложений с параллельными вычислениями (Eclipse IDE for Parallel Application Developers).
Другое название этого проекта — **Eclipse PTP** (***Eclipse Parallel Tools Platform***).

На кластере ЭВМ кафедры АСУ установлена среда разработки Eclipse PTP версии 3.7 (проект Indigo).
Эта среда обеспечивает разработку ПО на C, C++ и Fortran.

4.2 Контроль выполнения работ по теме 1

Текущий и заключительный контроль выполнения лабораторных работ является обязательной составляющей процесса обучения.

Текущий контроль преподаватель проводит следующими способами:

- индивидуальная регистрация посещения каждой лабораторной работы каждым студентом;
- наблюдение общего хода выполнения отдельной лабораторной работы в течение отведенного для ее выполнения времени;
- просмотр индивидуальных отчетов магистрантом и регистрация объема выполненной работы;
- оценка результатов, выполненных каждым магистрантом работ, по двум контрольным точкам.

Заключительный контроль проводится общей оценкой результатов обучения по всей дисциплине в виде ***экзамена***.

За время обучения по дисциплине «Архитектура вычислительных комплексов», магистрант должен подготовить единый отчет.

Шаблон отчета расположен на рабочем столе пользователя ***upk***. Копию отчета следует скопировать на личный flashUSB, отредактировать титульную страницу и регулярно заполнять в процессе выполнения всех лабораторных работ по данной дисциплине.

В целом, текст отчета заполняется ***в произвольной форме***, выбранной самим обучающимся.

Обязательным является:

- деление отчета на разделы, соответствующие отдельной работе;
- отчет должен иметь «Оглавление» с перечисленными разделами и указанием страниц на каждый раздел;
- разделы должны быть пронумерованы; номер раздела должен быть равен номеру соответствующей лабораторной работы;
- допускается, по желанию магистранта, разбивать разделы на подразделы; в таком случае, эти подразделы должны быть отражены на странице «Оглавление»;
- магистрант несет полную ответственность за сохранность отчета.

По требованию преподавателя, отчет должен быть предоставлен для проверки и, при необходимости, скопирован в указанное преподавателем место.

Список использованных источников

1. Резник В.Г. Архитектура вычислительных комплексов. Самостоятельная и индивидуальная работа студента. Учебно-методическое пособие. – Томск, ТУСУР, 2017. – 13 с.

2. Резник В.Г. Учебный программный комплекс кафедры АСУ на базе ОС ArchLinux. Учебно-методическое пособие. – Томск, ТУСУР, 2017. – 38 с.

3. Бойченко И.В. ВЫЧИСЛИТЕЛЬНЫЙ КЛАСТЕР ТУСУРа: СТАРТ. Труды постоянно действующего научно-технического семинара «Информационные системы», 2008. - 9 с. [Электронная копия]: HPC-TUSUR-START-2008.pdf.

4. Костромин В. Графический интерфейс Linux: Интернет-ресурс - http://rus-linux.net/papers/xwin/X-Window.html.

Учебное издание

Резник Виталий Григорьевич

АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ КОМПЛЕКСОВ

Учебно-методическое пособие предназначено для изучения темы №1 по дисциплине «Архитектура вычислительных комплексов» для студентов уровня основной образовательной программы магистратура направления подготовки: 09.04.01 «Информатика и вычислительная техника».

Учебно-методическое пособие

Усл. печ. л. . Тираж . Заказ .
Томский государственный университет
систем управления и радиоэлектроники
634050, г. Томск, пр. Ленина, 40