МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Кафедра автоматизированных систем управления (АСУ)

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Тема 1. Назначение и функции ОС

Учебно-методическое пособие

для студентов уровня основной образовательной программы: бакалавриат
направление подготовки: **09.03.01 - Информатика и вычислительная техника**
направление подготовки: **09.03.03 - Прикладная информатика**

Разработчик
доцент кафедры АСУ

В.Г. Резник

2022

Резник В.Г.

Операционные системы. Тема 1. Назначение и функции ОС. Учебно-методическое пособие. – Томск, ТУСУР, 2022. – 33 с.

Учебно-методическое пособие предназначено для изучения теоретической части и выполнения лабораторной работы №1 по теме «Назначение и функции ОС» учебной дисциплины «Операционные системы» для студентов кафедры АСУ ТУСУР уровня основной образовательной программы бакалавриат направлений подготовки: «09.03.01 - Информатика и вычислительная техника» и «09.03.03 - Прикладная информатика».

Оглавление

Введение

Дисциплина «*Операционные системы* (ОС)» изучается студентами кафедры АСУ ТУСУР на третьем курсе обучения уровня основной образовательной программы бакалавриат. Программа этого курса предполагает:

- лекционные занятия в объёме 36 часов;
- лабораторные занятия в объёме 36 часов;
- самостоятельную работу, консультации и экзамен.

Целью дисциплины является обучение студентов современным представлениям об основах построения операционных систем. Эта цель достигается формированием у обучающихся адекватных представлений об архитектуре ОС, а также получением и закреплением практических навыков работы с ними. В результате, студенты должны знать теоретические концепции, состав и взаимодействие компонент современных ОС и уметь использовать полученные навыки в своей профессиональной деятельности.

Сам процесс обучения проводится в учебных классах кафедры АСУ ТУСУР, которые оборудованы проекторами для демонстрации теоретического материала, а также вычислительной техникой, для выполнения лабораторных работ.

Учебно-методическая база данного курса реализована в виде «Учебного программного комплекса АСУ (УПК АСУ)», который представляет специальный дистрибутив ОС Arch Linux с необходимым методическим материалом и инструментальными средствами.

Методические материалы дисциплины представлены:

- учебно-методическим пособием «Самостоятельная и индивидуальная работа студента» [1, Резник В.Г.];
- электронным вариантом учебника [2, Гордеев А.В.];
- электронным вариантом учебника [3, Таненбаум А.В.];
- учебно-методическим пособием «Учебный программный комплекс кафедры АСУ на базе ОС ArchLinux» [4, Резник В.Г.];
- учебно-методическими пособиями по отдельным темам дисциплины «Операционные системы», список которых можно найти в [1].

Все документы, кроме четвёртого, становится доступными только после запуска ОС УПК АСУ и подключения соответствующего курса обучения. Четвёртый документ, в виде файла *upk_asu.pdf*, находится среди файлов дистрибутива ОС УПК АСУ на компьютерах, где это программное обеспечение (ПО) установлено.

Учебный материал данного пособия может использоваться для изучения дисциплины «***Операционные системы***», в пределах уровня основной образовательной программы бакалавриат направлений подготовки 09.03.03 «Прикладная информа-

тика» и 09.03.01 «Информатика и вычислительная техника». Он содержит шесть тем, структура и перечень которых подробно изложены в методическом пособии [1, «Самостоятельная и индивидуальная работа студента»].

Данное пособие является первой частью учебно-методического материала, включённого в ПО ОС УПК АСУ для данной дисциплины.

Весь учебный материал изложен в виде двух разделов:

- Тема 1. Теоретическая часть;
- Лабораторная работа №1.

Подобное методическое разделение учебного материала является типичным и для остальных тем данной дисциплины.

Теоретическая часть изучаемой темы содержит описание ряда концепций и представлений, которые сформировались в процессе становления и развития данного направления вычислительной техники. Подобные представления формируют некоторый целостный образ ОС и намечают направления для дальнейшего более детального изучения предмета. Изложение теоретического материала ведётся без привязки к каким-то конкретным разделам учебников или монографий, не отдаёт преимущество каким-то корпоративным представлениям и не противоречит сложившимся взглядам на данный предмет. Для более детального изучения теоретической части дисциплины предусмотрены учебники [2-3].

Практическая часть изучаемой темы предполагает выполнение лабораторной работы №1, цель которой — общее изучение ПО ОС УПК АСУ, как примера современной ОС рабочей станции (***desktop***). Учебные задания этой части работы предполагают дополнительное изучение первой части методического материала, изложенного в пособии [4].

1 Тема 1. Назначение и функции ОС

Кроме наиболее широко известных ОС – **MS Windows** и **Linux**, имеется целый ряд весьма интересных и распространённых линий развития ОС. Это - прежде всего операционные системы крупных фирм:

- **IBM** – в 1960-х – 1970-х годах разработала OS IBM 360/370; затем – ОС для персональных компьютеров OS/2; в настоящее время наиболее современными ОС этой фирмы являются ***z/OS*** и ***z/VM***;
- **Apple** – с начала 1980-х годов развивает семейство ОС ***MacOS***, которые характеризуются улучшенным графическим пользовательским интерфейсом;
- **Oracle/Sun** – с начала 1980-х годов фирма Sun развивает ОС ***Solaris*** (диалект UNIX);
- **Hewlett-Packard** – развивает собственный диалект UNIX – систему ***HP/UX***;
- **Novell** – одна из ведущих фирм в области сетевых технологий; развивает семейство сетевых ОС ***NetWare***; в настоящее время - ***Open Enterprise Server*** (сетевая ОС, включающая все сетевые возможности NetWare и возможности распространённого диалекта Linux - ***openSUSE***).

Это далеко не полный перечень коммерческих и исследовательских ОС, включающий сотни наименований.

В данной теме рассматривается ряд концепций и представлений, которые сформировались в процессе становления и развития дисциплины «*Операционные системы* (ОС)». Хотя отдельные такие представления отражают разные взгляды, сформулированные в различных терминах, их общий набор должен сформировать целостный образ предмета изучения и обеспечить студента базовыми понятиями, необходимыми для дальнейшего более подробного изучения материала.

1.1 ОС как базовая часть систем обработки данных (СОД)

Прежде чем изучать ОС, необходимо ответить на вопрос: «*Зачем нужны компьютеры?*».

Одним из возможных ответов: «*Компьютеры нужны для обработки данных*». Известный русский учёный, Ларионов А.М., анализируя возможности обработки данных на ЭВМ, даёт 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 классификации СОД, данная Ларионовым А.М., приведена на рисунке 1.1, которую 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 ведётся отдельной ЭВМ, вычислительным комплексом или вычислительной системой;
- *распределенные системы*, в которых процессы обработки данных рассредоточены по многим компонентам: системам телеобработки или вычислительным сетям.

Нетрудно догадаться, что каждый элемент такой классификации имеет свою ОС, хотя их функциональные возможности могут сильно отличаться.

С другой стороны, мы понимаем, что *ОС — это программное обеспечение (ПО)*, которое устанавливается на аппаратную часть вычислительной техники. С этой точки зрения, рассматривая отдельную ЭВМ, мы будем различать:

а) *техническую часть* — аппаратное обеспечение компьютера, упрощённая архитектура которого представлена на рисунке 1.2;
б) *программную часть* — программное обеспечение компьютера, общая классификация которого показана на рисунке 1.3.

Архитектура современного общедоступного компьютера представляет набор

функциональных компонент, во многом работающих независимо друг от друга, но согласованно взаимодействующих через общую системную магистраль.

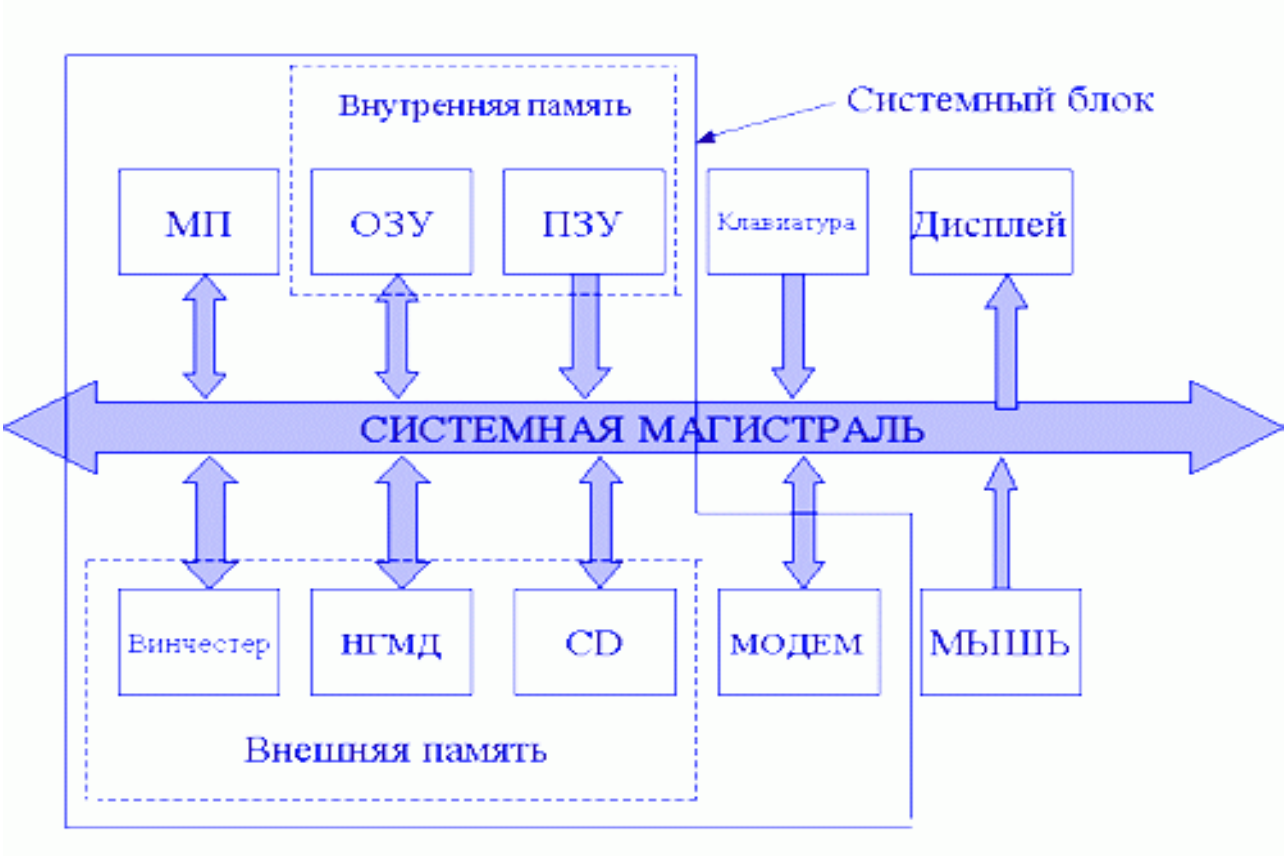

Рисунок 1.2 - Архитектура аппаратной части ЭВМ

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

1) микропроцессор (**МП**);
2) блок оперативного запоминающего устройства (**ОЗУ**);
3) микросхемы постоянного запоминающего устройства (**ПЗУ**);
4) устройства долговременной памяти на «жёстком диске» (**Винчестер**);
5) 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: клавиатура, дисплей, "мышь", принтер и другие устройства.

Программная часть ЭВМ (рис. 1.3) условно разделяется на три категории:

- *системное ПО* - ОС и программы общего пользования, выполняющие различные вспомогательные функции, например, создание копий используемой информации, выдачу справочной информации о компьютере, проверку работоспособности устройств компьютера и другие;
- *прикладное ПО* - программы, обеспечивающие выполнение необходимых работ на ЭВМ: редактирование текстовых документов, создание рисунков или картинок, обработка информационных массивов и другие;

- *инструментальное ПО* - программы, обеспечивающие разработку новых программ для компьютера на различных языках программирования.

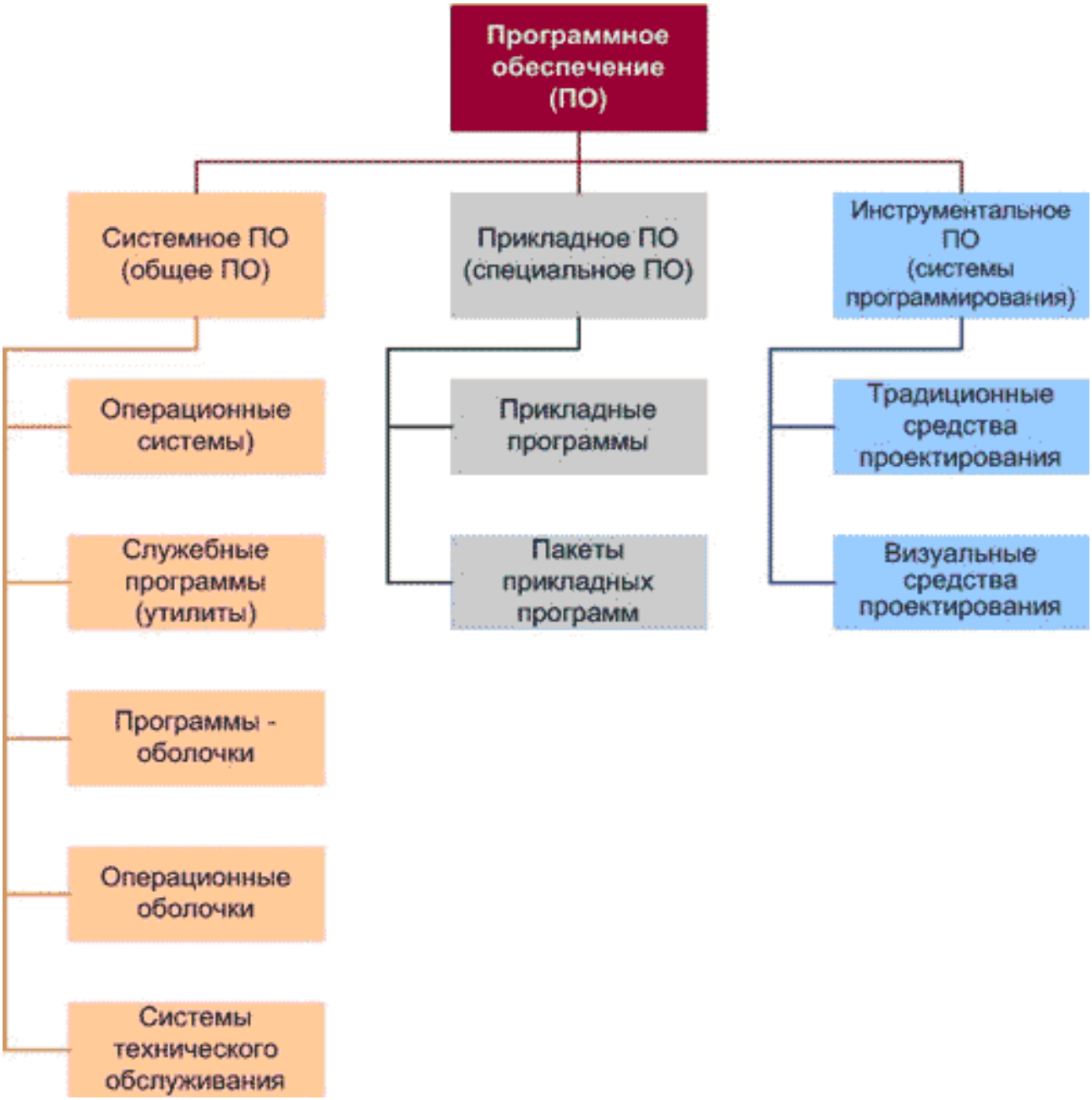

Рисунок 1.3 - Классификация ПО ЭВМ

Хорошо видно, предметом нашей дисциплины является категория системного ПО.

Системное ПО - это программы общего назначения, не связанные с конкретным применением ЭВМ, и выполняющие традиционные функции: планирование и управление задачами, управление вводом-выводом и другие. К ним относятся:

1) *операционные системы* — программа, которая загружается в ОЗУ ЭВМ при включении компьютера;
2) *программы–оболочки* - обеспечивают более удобный и наглядный способ общения с компьютером, по сравнению с командной строкой DOS, например, Far, Total Commander;

3) *операционные оболочки* – интерфейсные системы, которые используются для создания графических интерфейсов, мультипрограммирования и другие;
4) *драйверы* - программы, предназначенные для управления портами периферийных устройств; обычно загружаются в оперативную память при запуске компьютера;
5) *утилиты* - вспомогательные или служебные программы, которые представляют пользователю ряд дополнительных услуг.

Замечание

Часто бывает сложно разделить ПО, относящееся к утилитам, и прикладное ПО, поскольку часть утилит входит в состав ОС, а другая часть поставляется и функционирует автономно. Общепринято считать, что к утилитам относятся:

- *диспетчеры файлов* или *файловые менеджеры*;
- *средства динамического сжатия данных*, которые позволяют уменьшить размеры файлов и увеличить количество информации на диске за счёт ее динамического сжатия;
- *средства просмотра* и *воспроизведения*;
- *средства диагностики* и *контроля*, которые позволяют проверить конфигурацию компьютера и работоспособность его устройств, прежде всего «жёстких дисков»;
- *средства коммуникаций* (коммуникационные программы), предназначенные для организации обмена информацией между компьютерами;
- *средства обеспечения компьютерной безопасности*: резервное копирование, антивирусное ПО и другие.

1.2 Серверные ОС и рабочие станции

Несмотря на кажущуюся простоту понятий *сервер* и *рабочая станция*, необходимо внимательно относиться к контексту, в котором эти термины употребляются. Прежде всего, следует уточнить идёт ли речь об аппаратном или о программном обеспечении.

В контексте аппаратного обеспечения ЭВМ:

- *сервер* — существительное от глагола ***to serve*** — служить; специализированный компьютер или оборудование для выполнения на нем сервисного ПО; ЭВМ с повышенной надёжностью исполнения, имеющее сетевые устройства и предназначенное для непрерывной работы в течении длительного без выключения или перезагрузки ОС;
- *рабочая станция* — ***workstation*** — ЭВМ, оборудование которой расширено устройствами мультимедиа и другими системами, предназначенная для решения определённого круга задач; наличие оборудования для работы в сети является необязательным требованием, но требования к возможности интерактивного взаимодействия с пользователем являются определяющими.

В контексте программного обеспечения.

Фактически, сервер и рабочая станция могут иметь одинаковую аппаратную конфигурацию, что во многом ограничивает возможности контекста аппаратного обеспечения.

В контексте программного обеспечения подразумевается использование парадигмы «*клиент-сервер*»:

- *сервер* — любая запущенная программа, ориентированная в прикладном пла-

не на обслуживание запросов от других программ — *клиентов*;
- *рабочая станция* — ЭВМ для интерактивной работы с пользователем, на которой установлено *клиентское программное обеспечение.*

На ранней стадии развития, многие ОС не поддерживали работу в сети. Например, MS DOS и MS Windows, первоначально создавались как рабочие станции, предполагающие автономную работу ЭВМ. Со временем, такие ОС стали использовать сетевое ПО сторонних разработчиков, а те ОС, которые имели собственное сетевое ПО, стали называться сетевыми ОС.

В настоящее время практически все ОС способны работать в сети, поэтому необходимость в дополнительной классификации отпала сама собой. Тем не менее градация ЭВМ осталась, но перешла в область системного ПО и дистрибутивов ОС:

- **сервер** (*server*) — дистрибутив ОС или ЭВМ, с установленным системным и прикладным ПО, ориентированные на выполнение функций сервера;
- **рабочая станция** (*desktop*) — дистрибутив ОС или ЭВМ, предназначенная для интерактивной работы с пользователем, на которой установлено соответствующее клиентское прикладное программное обеспечение.

Замечание

Применительно к ограничениям уровня изучения нашей дисциплины, различия между серверами и рабочими станциями являются не существенными, поскольку определяются ***специализацией системного и прикладного ПО ЭВМ***. ПО ОС УПК АСУ создано на основе дистрибутива типа *desktop*, поэтому для работы в качестве сервера требуется установка дополнительного ПО.

В целом, объем нашего курса предполагает изучение ПО ОС, на уровне ***desktop***, ограниченной отдельной ЭВМ.

1.3 Многослойная структура ОС

Проблема. Первоначально, ОС были монолитными и не имели архитектуры. Например, корпорация IBM в 1964 году стала разрабатывать первую версию ОС OS/360 и за 5 лет, коллектив из 5000 человек написал более 1 млн строк кода.

Постепенно стало ясно, что разработка ОС должна:

1) *вестись* на основе модульного программирования;
2) *иметь* иерархическую структуру.

В результате были разработаны концептуальные требования к архитектуре ОС.

Классическая архитектура ОС основана на:

1) концепции иерархической многоуровневой машины;
2) концепции привилегированного ядра ОС;
3) концепции пользовательского режима работы транзитных модулей.

Модули ядра выполняют базовые функции ОС:

1) управление процессами, памятью;
2) устройствами ввода-вывода и тому подобное.

В концепции многоуровневой (многослойной) иерархической машины, структура ОС представляется рядом слоёв, показанных на рисунке 1.4. Здесь, каждый внутренний слой обслуживает вышележащий слой через межслойный интерфейс.

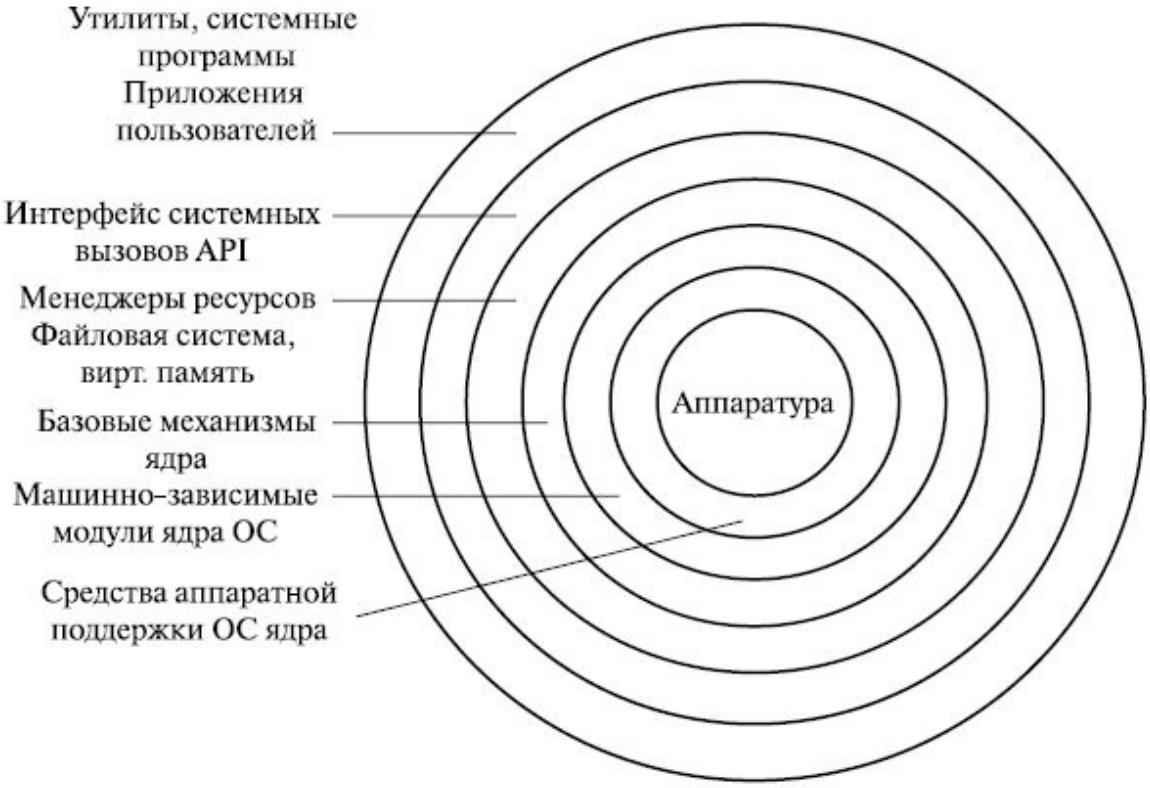

Рисунок 1.4 - Иерархическая архитектура ОС

Такая организация существенно упрощает разработку системы и позволяет:

1) сначала, "***сверху вниз***" определить функции слоёв и межслойные интерфейсы;
2) при детальной реализации, двигаясь "***снизу вверх***", – можно наращивать мощность функций слоёв;
3) модули каждого слоя можно изменять без необходимости изменений в других слоях, но не меняя межслойных интерфейсов!

В общем случае повышение устойчивости работы ОС обеспечивается переходом ядра ОС в *привилегированный режим*.

Привилегированный режим — особый режим работы процессора, поддерживаемый аппаратурой ЭВМ, в котором никакая программа, работающая в других режимах, не может прервать работу процессора.

1.4 ОС как базовая часть ПО ЭВМ

Из рисунка 1.4 хорошо видно, что в классической архитектуре ОС прикладным программам пользователей отводится верхний, последний слой. Все остальные слои архитектуры составляют *ядро ОС*, которое непосредственно взаимодействует с аппаратным обеспечением ЭВМ.

Ядро ОС составляет сердцевину системного ПО ЭВМ, без которого это программное обеспечение является *полностью неработоспособным* и *не может выполнить ни одну из своих функций*.

В ядре решаются *внутрисистемные задачи организации вычислительного процесса*, недоступные для приложений.

Таким образом, ядро ОС является *базовым ПО ЭВМ*.

Особый класс функций ядра служит для поддержки приложений, создавая для них так называемую *прикладную программную среду*. Все приложения обращаются к ядру со специальными запросами – *системными вызовами*.

Примеры системных вызовов:

1) *открытие* и чтение файла;
2) *получение* системного времени;
3) *вывод* информации на дисплей компьютера и другие вызовы.

Функции ядра, которые могут вызываться приложениями, образуют *интерфейс прикладного программирования* – **API** (*Application Programming Interface*).

Хотя архитектура аппаратной части многих ЭВМ может быть адекватно представлена рисунком 1.2, конкретные реализации такой архитектуры могут быть различны:

1) *различаются* процессора и поддерживаемый набор команд;
2) *различаются* шины компьютера и устройства подключения к ним;
3) *постоянно идёт развитие* и *изменение* конструктивных особенностей всех внешних устройств.

В такой ситуации, каждое ядро ОС реализует некоторую абстрактную и более упрощённую архитектуру ЭВМ, доступ которой реализуется через стандартный набор функций, называемый функциями системных вызовов.

В результате прикладной программист рассматривает ядро ОС как некоторую *абстрактную (**виртуальную**) машину*, которая является средой для выполнения его программ.

Систематизируя различные системные вызовы и развивая идею виртуальной машины, мы с точностью до терминологии можем утверждать, что каждое ядро ОС, абстрагирует три базовых концепции: ***файл***, ***пользователь*** и ***процесс***.

Но прежде чем обосновать это, рассмотрим более подробно взаимодействие прикладных программ пользователя с защищённым ядром ОС, через интерфейс API. Кроме того, следует также учесть, что на основании рисунка 1.4 можно формировать различные типы ядер ОС.

1.5 Режимы ядра и пользователя

Чтобы повысить надёжность работы ОС, её ядро работает в специальном привилегированном (***защищённом***) режиме, а режим, в котором работают утилиты и остальное прикладное ПО ОС, называется *режимом пользователя*.

Сначала рассмотрим взаимодействие ПО ЭВМ для ***классической архитектуры ядра ОС UNIX***, которое показано на рисунке 1.5.

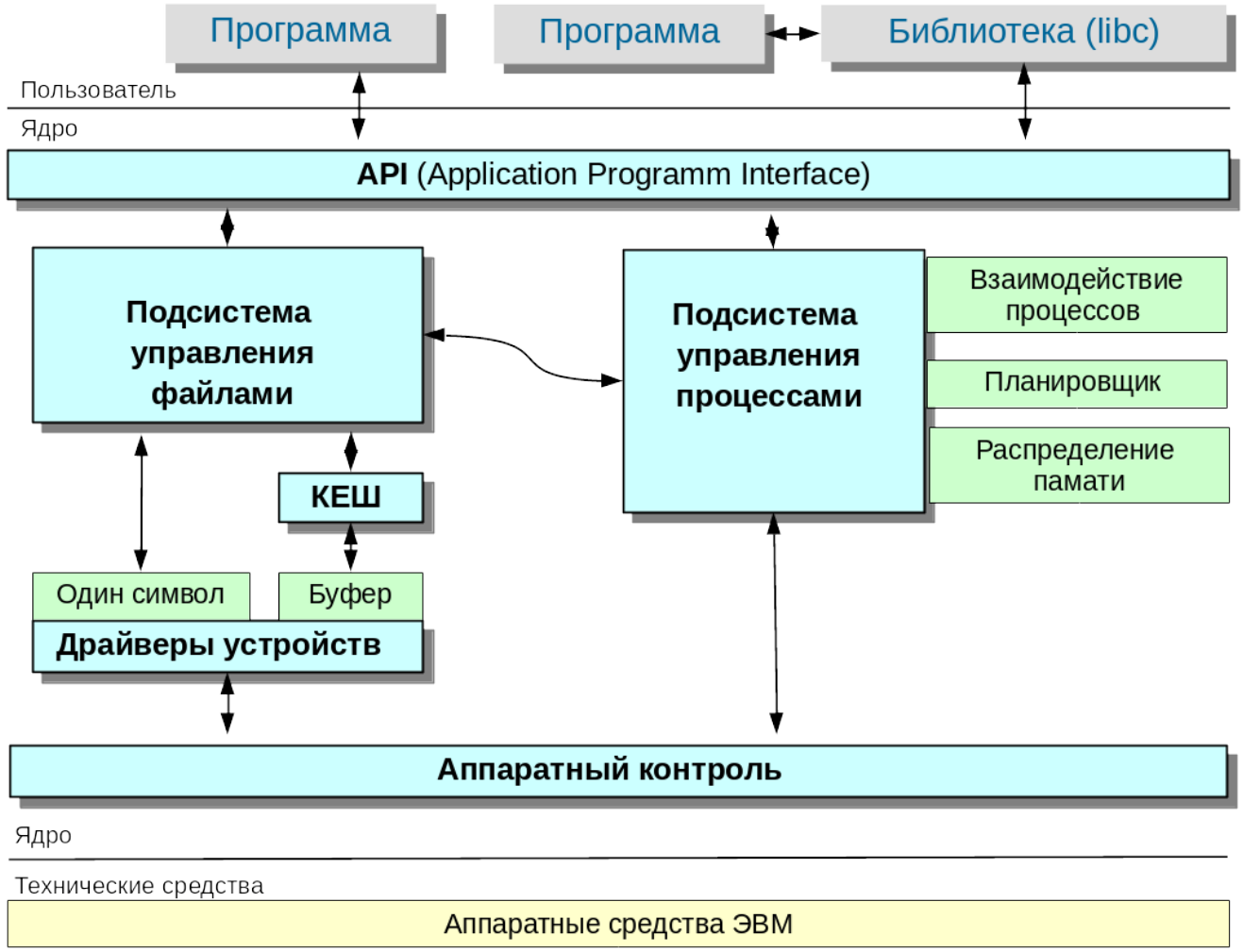

Рисунок 1.5 - Классическая архитектура ПО ОС UNIX

Классическая архитектура ПО ОС UNIX содержит *монолитное ядро ОС*, которое согласно рисунку 1.4 содержит ПО всех уровней, кроме последнего.

Системный вызовов такого ядра ОС происходит в два этапа, показанного на рисунке 1.6:

а) *системный вызов* привилегированного ядра ОС инициирует переключение процессора из пользовательского режима в привилегированный;

б) *при возврате к приложению*, происходит обратное переключение.

За счёт ***времени переключения 2t*** возникает дополнительная задержка в обработке системного вызова. Однако такое решение стало классическим и используется во многих ОС: ***UNIX***, ***VAX***, ***VMS***, ***IBM OS/390***, ***OS/2*** и других.

Рисунок 1.6 - Переключение режимов монолитного ядра ОС

Замечание

Многослойная классическая многоуровневая архитектура ОС не лишена своих проблем:

а) *значительные изменения одного из уровней* могут иметь трудно предвидимое влияние на смежные уровни;

б) *многочисленные взаимодействия между соседними уровнями* усложняют обеспечение безопасности работы ОС.

Альтернативой классическому варианту архитектуры ОС является *микроядерная архитектура ОС*.

Суть этой архитектуры состоит в следующем:

а) в привилегированном режиме работает только очень небольшая часть ОС, называемая *микроядром*.

б) *микроядро защищено* от остальных частей ОС и приложений.

в) в состав микроядра входят *машинозависимые модули*, а также *модули, выполняющие базовые механизмы обычного ядра*.

г) все остальные, более высокоуровневые функции ядра, оформляются как *модули, работающие в пользовательском режиме*.

На рисунке 1.7 представлено сравнение классической и микроядерной архитектур ОС:

1) *менеджеры ресурсов*, являющиеся неотъемлемой частью обычного ядра, становятся "периферийными" модулями, работающими в пользовательском режиме;

2) *внешние по отношению к микроядру компоненты ОС* реализуются как обслуживающие процессы;

3) *между собой* эти модули взаимодействуют как равноправные партнёры с помощью обмена сообщениями, которые передаются через микроядро.

Менеджеры ресурсов, вынесенные в пользовательский режим, называются ***серверами ОС.*** Схематично механизм обращений к функциям ОС, оформленным в виде серверов, выглядит, как показано на рисунке 1.8.

Для микроядра, схема смены режимов, при выполнении системного вызова в ОС, показана на рисунке 1.9. Хорошо видно, что выполнение системного вызова сопровождается *четырьмя переключениями режимов (4t)*, а в классической архитектуре – *двумя (2t)*. Следовательно, производительность ОС с микроядерной архитектурой, при прочих равных условиях, будет ниже, чем у ОС с классическим ядром.

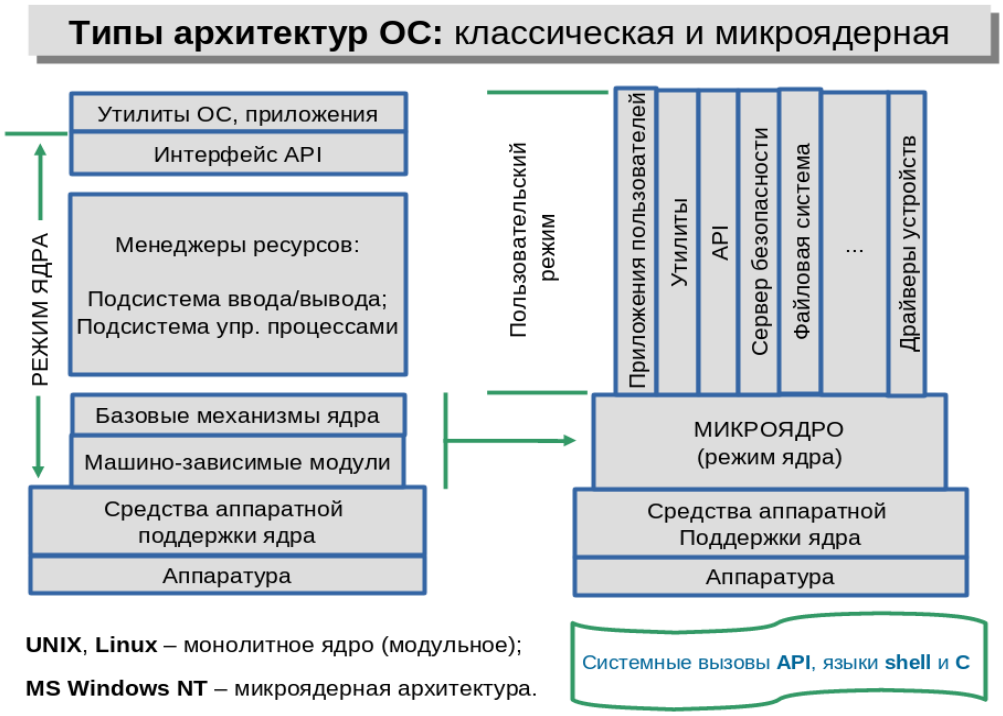

Рисунок 1.7 - Сравнение классической и микроядерной архитектур ОС

Замечание

По многим литературным источникам, вопрос масштабов потери производительности в микроядерных ОС является спорным. *Большинство ядер дистрибутивов ОС Linux являются монолитными.*

Рисунок 1.8 — Системные вызовы через микроядро ОС

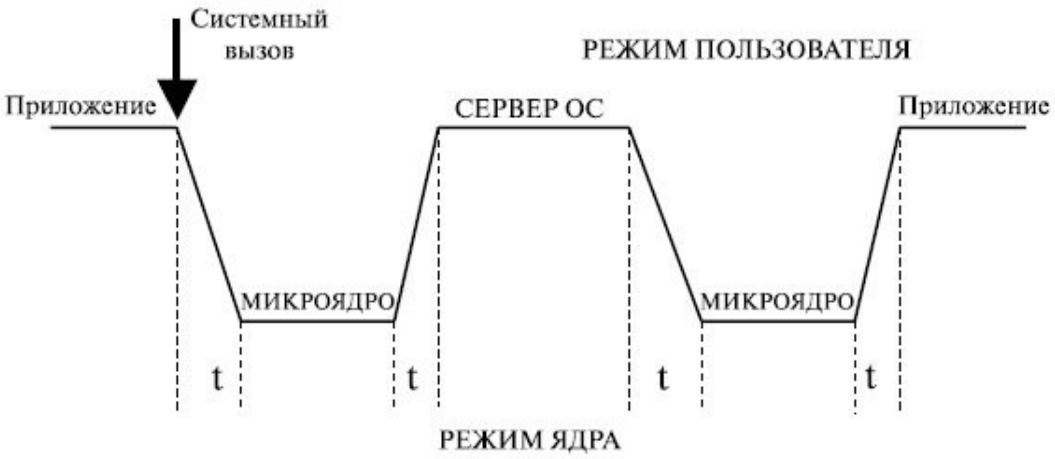

Рисунок 1.9 - Схема переключения режимов микроядерной архитектуры ОС

С 1990 года, в рамках проекта GNU, ведётся разработка микроядерная архитектура Hurd ОС Linux, основанная на микроядре ***GNU Mach***. Ричард Столлман, руководитель проекта GNU, в 2002 году заявил о скором выходе производственной версии ***Hurd***, однако его обещания не оправдались.

Микроядерными являются ядра ***ОС Minix*** и ядро ***систем семейства BSD***.

Windows NT часто также называют микроядерной ОС, однако многие считают, что:

а) *микроядро NT слишком велико* (более 1 Мбайт), чтобы носить приставку "микро";
б) *все компоненты ядра работают в одном адресном пространстве* и активно используют общие структуры данных, что свойственно операционным системам с монолитным ядром.

1.6 Ядро и модули ОС

Как было отмечено выше, модули ядра выполняют основные базовые функции ОС:

1) *управление* процессами и памятью;
2) *управление* устройствами ввода-вывода и другими элементами.

Хотя модульная организация ОС характерна для микроядерной архитектуры, монолитные ядра также используют модули. Вызвано это тем, что аппаратная часть ЭВМ настолько многообразна, что практически неэффективно создавать ядро на все возможные варианты конфигурации модулей.

Традиционно, взаимодействие ядра ОС с аппаратной частью ЭВМ осуществляется через специальное ПО, которое называется ***драйверами***.

Любое ядро ОС является одной большой программной, которая:

1) выполняется ***в защищённом режиме*** (режиме ядра);
2) выполняется ***в своём собственном адресном пространстве***.

Поэтому, перед компиляцией ядра запускается *программа-конфигуратор*, в которой можно указать какие драйвера включаются в ядро статически, а какие будут присутствовать как модули.

Когда ядро ОС загружено в память ЭВМ и начинает работать, запускается *первый пользовательский процесс* ***init*** (или скрипт ***init***), имеющий ***PID=1***, который обеспечивает дальнейшую загрузку необходимых модулей.

Для работы с модулями ОС Linux имеет специальные утилиты:

1) **lsmod** — просмотр списка модулей;
2) **insmod** — инсталляция модулей;
3) **rmmod** — удаление модуля;
4) **modprobe** — может выполнять функции ***insmod*** и ***rmmod***;
5) **modinfo** — получение информации о модуле.

1.7 Три базовых концепции ОС: файл, пользователь, процесс

Подведём итог изученного ранее учебного материала. Нами были рассмотрены различные концепции и представления:

а) модель СОД раскрывает различные архитектурные возможности применения средств вычислительной техники; на уровне отдельной ЭВМ можно выделить аппаратную и программную части; в программной части выделяется системное ПО, которое содержит объект нашего изучения — ОС;

б) концепции серверных ОС и ОС рабочих станций в большей степени отражают особенности применения прикладного ПО, что находит своё отражение в дистрибутивах ОС и их использования в пределах СОД;

в) концепция многослойной структуры ОС отражает современную парадигму построения сложных программных систем; стремление повысить надёжность работы системного ПО приводит к выделению ядра ОС, работающего в привилегированном режиме; стремление повысить эффективность работы системного ПО приводит к созданию различных моделей ядер ОС, среди которых были выделены: монолитное ядро ОС и микроядро ОС;

г) концепция базового ПО ЭВМ является попыткой формализовать и стандартизировать наиболее важные понятия ОС, к которым в первую очередь относится её ядро; формализуется интерфейс прикладного программирования (API) и идея абстрактной (виртуальной) машины;

д) модели взаимодействия режимов ядра и пользователя наглядно показывают архитектурные возможности построения ОС; демонстрируются архитектуры классического (монолитного) ядра ОС и альтернативного — микроядра ОС;

е) идея модульности ОС является техническим решением, обеспечивающим устранение основного недостатка монолитных ядер — привязка к аппаратной конфигурации конкретной ЭВМ; в частности, ядро ОС УПК АСУ является монолитным и модульным.

Таким образом, нами изучены основные архитектурные особенности ОС, *кроме представлений прикладного уровня*, которые характерны для ПО, работающего в режиме пользователя. Данный подраздел посвящён именно этому вопросу.

Современный пользователь воспринимает ОС с внешней стороны — на уровне графической оболочки ОС. Эта оболочка представлена в виде *рабочего стола — **Desktop***.

На рабочем столе имеются: окна, панели, меню, курсор мыши и другие элементы. Некоторые ОС, например, *MS Windows и MacOS*, не могут запуститься без графической оболочки. В других ОС, например, *UNIX* и *Linux*, графическая оболочка является пользовательским приложением: *X Window (X-сервер)*.

Рассматривая работу ПО ЭВМ на профессиональном уровне, то можно утверждать, что:

а) в *привилегированном* режиме процессора (*режим ядра*) работает ядро ОС;

б) в *не привилегированном* режиме процессора (*режим пользователя*) работают утилиты, инструментальное и прикладное ПО ЭВМ;

в) *для прикладного ПО* — режим пользователя представляет некоторую *среду исполнения ОС*, которой управляет ядро ОС;

г) *прикладное ПО реализует некоторую модель СОД* и обращается к ядру ОС за дополнительными функциями, в основном связанными с доступом к аппаратным средствам ЭВМ;

д) *обращение прикладного ПО к ядру ОС* осуществляется посредством *системных вызовов*;

е) *классификация и абстрагирование* системных вызовов функций ядра ОС на более высоком уровне приводит к трём базовыми концепциям ОС: *файл, пользователь и процесс*.

Таким образом, мы приходим к понятию ***среды исполнения ОС***, которая опирается на три базовых концепции: ***файл***, ***пользователь*** и ***процесс***.

Эти концепции образуют иерархию отношений, показанную на рисунке 1.10.

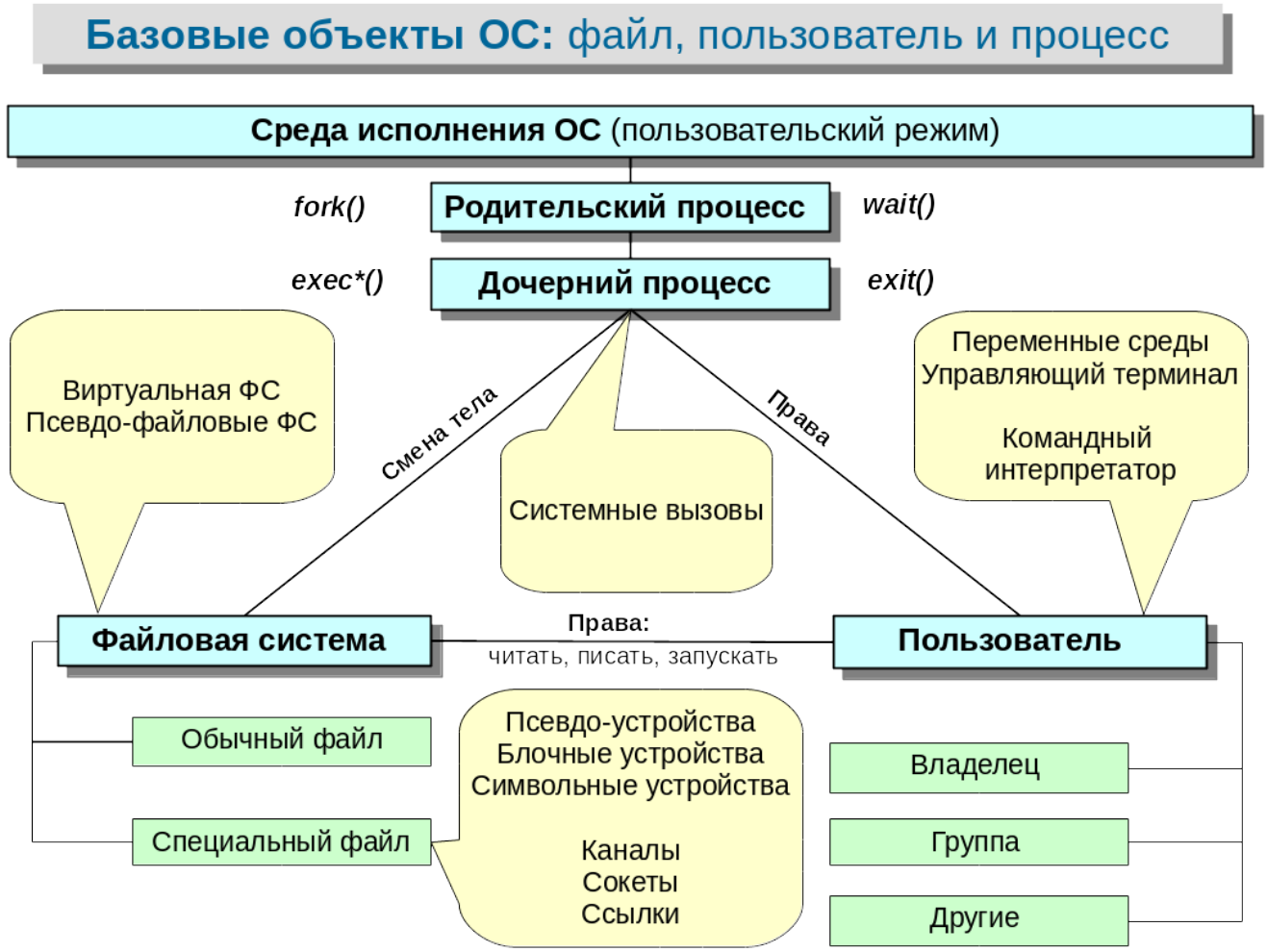

Рисунок 1.10 - Базовые концепции пользовательского режима ОС

Все концепции, структурно представленные на рисунке 1.10, будут подробно изучаться в последующих темах. Здесь мы рассмотрим лишь основные понятия и идеи, которые уже известны из теории других дисциплин или без которых невозможно обойтись при выполнении лабораторных работ.

Первое понятие, которое следует обсудить, является ***концепция файла***.

ОС UNIX (***Linux***) прямо декларируют парадигму: «*Все есть файл*».

К любому файлу потенциально применимы три операции: **r - *чтения***, **w - *записи*** и **x - *запуска***.

Конкретизация понятия файл, в аспекте хранилища данных, формализуется в понятие *файловой системы*, которая представляет собой *поименованную совокупность обычных и специальных файлов*.

Обычный файл — именованная упорядоченная последовательность байт.

Специальный файл имеет *имя* и *специализацию* по назначению:

а) ***устройства*** — отображение аппаратных средств компьютера в файловую систему ОС;

б) ***директории*** — файлы, представляющие список имён файлов и директорий; обеспечивают иерархическую структуру файловой системы ОС. Операция запуск, применительно к директории означает возможность «войти в неё» пользователю;

в) ***ссылки*** — именованные указатели на другие файлы, позволяющие работать с ними по имени ссылки.

г) ***именованные каналы*** — точки доступа ОС для передачи данных;

д) ***сокеты*** — точки доступа, через которые работает сетевое обеспечение ЭВМ.

Все операции с файлами интерпретируются через концепцию пользователя.

Понятие пользователя интерпретируется через свои элементы - ***владелец***, ***группа*** и ***другие***:

а) ***владелец*** — именованный и индексированный объект ОС, права которого интерпретируются на операции с файлами; индекс владельца или UID — User Identification — целое число (от нуля и выше), которое присутствует в каждом файле и интерпретируется как «хозяин файла»; чем меньше значение UID, тем более «важным» является пользователь; например, пользователь root имеет UID=0 и наивысшие права на файлы ОС;

б) ***группа*** — именованный и индексированный объект ОС, предназначенный для объединения владельцев ОС; каждый владелец должен входить хотя бы в одну группу; индекс группы или GID — Group Identification, аналогичен UID и выполняет те же функции для групп; владелец root имеет собственную группу с именем root и идентификатором GID=0; все администраторы ОС обычно входят в группу root;

в) ***другие*** — это пользователи без имени и идентификатора, которые дополняют концепцию пользователя и права которых также отображены в каждом файле.

Третья концепция - процесс, который обычно интерпретируется как *запущенная программа* или *задача*.

Процесс — это элементарный управляемый объект ОС, имеющий целочисленный идентификатор ***PID — Process Identification***, обеспечивающий функциональное преобразование файлов (данных) с правами, которые определяются объектами *пользователь*.

Значения **PID** начинаются с 1 (обычно - это процесс ***init***) — *главный родительский процесс*, и увеличиваются по мере *порождения дочерних процессов*.

Новому процессу присваивается номер на 1 больше, чем максимальный номер существующего или существовавшего с момента запуска ОС процесса.

1.8 Системные вызовы fork(...) и exec(…)

Ядро ОС самостоятельно запускает только один процесс init. Все остальные процессы режима пользователя являются дочерними относительно процесса init.

Запуск любой программы в режиме пользователя осуществляется с помощью двух системных вызовов: *fork(...)* и ***exec*(...)***.

Вызов функции *fork(...)* из программы на языке С, имеет вид:

```
#include <unistd.h>
pid_t fork(void);
```

Функция fork(...) полностью дублирует существующий процесс, вместе со всеми открытыми файлами, порождая новый (дочерний) процесс с новым PID. Программист различает родительский и дочерний процессы только по целочисленному значению, которое возвращает функция fork(...):

-1 — *ошибка*, дочерний процесс не создан**;**

0 - дочерний процесс;

> 0 — *родительский процесс*, которому передано значение ***PID*** дочернего процесса.

Замечание

Родительский процесс обязан дождаться завершения дочернего процесса, иначе контроль передаётся по иерархии выше. Процесс ***init*** является родителем для всех остальных процессов.

Если дочерний процесс создан для запуска некоторой программы, то используется одна из разновидностей системной функции *exec*(...)*:

```
#include <unistd.h>
extern char **environ;

int execl(const char   *path, const char *arg, ...);
int execp(const char   *file, const char *arg, ...);
int execle(const char  *path, const char *arg, …, char * const envp[]);
int execv(const char   *path, const *char argv[]);
int execvp(const char  *file, const *char argv[]);
int execvpe(const char *file, const *char argv[], char * const envp[]);
```

Хорошо видно, что любой вызов имеет ссылку на файл, в качестве аргумента:

а) ***после всех проверок*** на права запуска, указанный файл загружается в пространство дочернего процесса.

б) ***загруженной программе*** передаются все ресурсы дочернего процесса, включая открытые и созданные файлы.

Замечание

Обратите внимание, что родительский процесс продолжает контролировать работу уже новой программы и *имеет право принудительно завершить её работу*.

Замечание.

Эти функции (exec*(...)) дублируют действия оболочки при поиске исполняемого файла, если указанное имя файла не содержит символ косой черты (/). Файл ищется в разделенном двоеточием списке путей к каталогам, указанном в переменной среды PATH. Если эта переменная не определена, список путей по умолчанию представляет собой список, включающий каталоги, возвращаемые функцией confstr(_CS_PATH) (которая обычно возвращает значение "*/bin:/usr/bin*"), а также, возможно, текущий рабочий каталог; см. ПРИМЕЧАНИЯ для получения дополнительной информации.

ПРИМЕЧАНИЯ.

Путь поиска по умолчанию (используемый, когда среда не содержит переменной PATH) показывает некоторые различия в разных системах. Обычно он включает */bin* и */usr/bin* (именно в таком порядке), а также может включать текущий рабочий каталог.

В некоторых других системах *текущий каталог* включается уже после */bin* и */usr/bin*, *в качестве меры защиты от троянских коней*. Реализация *glibc* долгое время следовала традиционному стандарту по умолчанию, *когда текущий рабочий каталог включается в начало пути поиска*. Однако некоторый рефакторинг кода, во время разработки *glibc 2.24*, привёл к тому, что *текущий рабочий каталог был полностью исключён из пути поиска по умолчанию*. Это введённое изменение поведения считается немного полезным и не может быть отменено.

Все перечисленные функции *exec*(...)*, в конечном итоге, преобразуются в системную функцию *execve(...)*.

#include <unistd.h>
int execve (const char *pathname, char *const argv[], char *const envp[]);

execve() выполняет программу, на которую ссылается путь (*pathname*). Это приводит к тому, что программа, которая в настоящее время выполняется вызывающим процессом, заменяется *новой программой*, *с вновь инициализированным стеком*, *кучей* и (инициализированными и неинициализированными) *сегментами данных*.

Обратите внимание, что ***argv[0]*** — *имя файла* запускаемой программы.

1.9 Дистрибутивы ОС

Когда говорят, что на ЭВМ установлена некоторая ОС, то обычно подразумевают некоторый её дистрибутив, включающий конкретное ядро ОС и другое системное ПО, а также прикладное ПО и системы разработки.

Выбор конкретного дистрибутива предполагает учёт многих факторов:

а) *производитель* дистрибутива;
б) *тип процессора*, на который рассчитан дистрибутив;
в) *лицензия дистрибутива* и ценовая политика дистрибьютора;
г) *поддержка национальных языков*;
д) *типы носителей*, на которых распространяется дистрибутив;
е) *особенности инсталляции*;
ж) *сопровождение дистрибутива*;
з) *наличие документации*.

Учёт всех перечисленных факторов может оказаться довольно сложной задачей и выходит за рамки нашей дисциплины. Более того, каждому дистрибутиву и его версиям посвящены отдельные сайты, а многие дистрибутивы постоянно обсуждаются на форумах в Интернете.

Для целей обучения выбран 64-битный базовый дистрибутив ***Arch Linux***. На его основе создан набор ПО, организованный как учебный программный комплекс кафедры АСУ (УПК АСУ).

Замечание

Структура ***ОС УПК АСУ*** ориентирована не только на задачи курса «***Операционные системы***», но имеет всё необходимое ПО для изучения этой дисциплины и организации проведения лабораторных работ.

2 Лабораторная работа №1

Данный учебный материал является методическим пособием по проведению лабораторной работы №1 по дисциплине «***Операционные системы***».

Работа проводится в рамках темы №1 «*Назначение и функции ОС*».

Цель работы — получение практических навыков использования ОС УПК АСУ, применительно к изучаемой дисциплине.

Указанная цель достигается посредством:

- *изучения структуры* ПО ОС УПК АСУ;
- *создания индивидуального загрузочного устройства* flashUSB;
- *освоения процедуры* запуска ОС УПК АСУ с загрузочного flashUSB;
- *получения навыков работы* в среде пользователя ***asu***; в частности: подключение к ОС УПК АСУ личного архива студента, содержащего рабочую область пользователя ***upk***; выход из сессии пользователя ***asu*** и вход в сессию пользователя ***upk***;
- *изучения рабочей среды* (рабочего стола) пользователя ***upk***, содержащего учебный материал и инструментальные средства для выполнения всех лабораторных работ по изучаемой дисциплине;
- *выполнения учебных заданий* данного раздела пособия;
- *оформления отчёта* по выполненным заданиям;
- *освоением процедур* создания личного архива на flashUSB и завершения работы с ОС УПК АСУ.

Замечание

Данное учебно-методическое пособие становится доступным только после запуска ОС УПК АСУ, подключения личного архива студента и входа в сеанс пользователя ***upk***, поэтому:

- ***основная часть работ*** выполняется под непосредственным руководством преподавателя;
- ***основной учебный материал*** по данной лабораторной работе содержится в методическом пособии [4, раздел 1, «Назначение и использование ОС УПК АСУ»];
- ***учебный материал данного раздела*** только дополняет и уточняет [4], в плане особенностей изучаемой дисциплины.

Учитывая указанные выше ограничения, лабораторная работа №1 выполняется в три этапа.

Этап 1, студент:

- *передаёт* преподавателю личное устройство ***flashUSB*** для установки на него загрузочного ПО;
- *изучает* учебный материал первого раздела пособия [4], доступный на ЭВМ учебных классов кафедры АСУ как файл: *upk_asu.pdf*.

Этап 2, студент:

- *получает* от преподавателя личное устройство ***flashUSB*** с установленным на нем загрузочным ПО и выполняет загрузку ОС УПК АСУ;

- *выполняет* учебные задания первого раздела пособия [4], доступного на рабочем столе пользователя ***asu*** как файл: *upk_asu.pdf*.

Этап 3, студент:

- *подключает* к ОС УПК АСУ личный архив со своего устройства ***flashUSB***, используя учебный материал [4, подраздел 1.1]; архив должен находиться в корне файловой системы ***flashUSB***: */asu64upk/themes/os-home.ext4fs.gz*;
- при подключении архива будет запрошен пароль пользователя **asu**; следует ввести **upkasu**;
- *выходит* из сеанса пользователя ***asu*** и *входит* в сеанс пользователя ***upk***;
- *запускает* на чтение данное руководство, читает раздел 2 и выполняет задания лабораторной работы №1.

Замечание

В процессе выполнения лабораторных работ, студент использует сеансы пользователей *asu* и *upk*. Оба пользователя имеют пароль: ***upkasu***

2.1 Рабочий стол УПК АСУ

Рабочий стол пользователя ***upk***, для дисциплины «***Операционные системы***», имеет оригинальную для данного курса обучения заставку с надписью в верхней части экрана: «*Операционные системы. Тема os*», показанное на рисунке 2.1.

Наличие такого стилизованного фона говорит о правильном подключении рабочей среды пользователя ***upk*** и служит для визуального контроля выбора нужной учебной среды ОС УПК АСУ.

Замечание

В случае, когда архив рабочей среды пользователя ***upk*** создавался на ЭВМ с другим графическим адаптером, вместо указанной заставки может появиться изображение рабочего стола пользователя ***asu***, которое в ОС УПК АСУ установлено по умолчанию. Следует восстановить нужное изображение и пересоздать личный архив.

Как это сделать? Обратитесь к преподавателю!

Кроме стилизованного изображения, на рабочем столе находится множество значков, часть из которых обозначают файловые системы компьютера.

Другие, например, «*Домашний каталог*», «*Корзина*» и «*Файловая система*» находятся на рабочем столе всегда, а значок устройства ***flashUSB*** появляется только после подключения этого устройства.

Нас, в первую очередь, должны интересовать *специальные значки* для данной дисциплины, перечень которых представлен в таблице 2.1.

Рисунок 2.1 — Правильное изображение рабочего стола

Таблица 2.1 — Специальные значки рабочего стола ОС УПК АСУ

Название значка	***Назначение***
Начало работы	Значок для запуска данного учебного пособия.
os_self_19_090303.pdf os_self_19_090301.pdf	Учебное пособие [1] для самостоятельной и индивидуальной работы студента.
upk_asu.pdf	Учебное пособие [4].
ОтчетОС1.odt	Шаблон единого отчёта для данной дисциплины, который студент должен регулярно и самостоятельно заполнять.
Учебный материал	Значок ссылки на директорию, содержащую весь учебный материал по данной дисциплине. Его активация запускает файловый менеджер Thunar.

Учебное задание

1. Запустить на редактирование файл *ОтчетОС1.odt*, а на просмотр «*Начало работы*» и *upk_asu.pdf*.
2. Зафиксировать в отчёте выполнение задания по [4, подраздел 1.1].
3. Перейти к изучению подраздела 2.2 данного методического пособия.

2.2 Работа с личным архивом студента на flashUSB

Личный архив студента — файл с именем *os-home.ext4fs.gz*, имеющий специальный формат хранения файловой системы ОС Linux, типа *ext4*.

Место хранения архива — личный ***flashUSB*** студента.

Каталог хранения архива — директория */asu64upk/themes* в файловой системе ***flashUSB***, типа *FAT32*.

Студент запускает ОС УПК АСУ, используя ПО GRUB, как это описано в учебном руководстве [4].

ПО GRUB передаёт ядру ОС *набор параметров*, включая *UUID раздела* блочного устройства flashUSB, имеющего тип файловой системы *FAT32* и отмеченного как архивное устройство.

ОС УПК АСУ читает параметры ядра, в процессе запуска ОС, и сохраняет их для дальнейшего использования в файле */etc/upkasu/upkasu.conf*.

В частности, ОС запоминает *UUID раздела архивного блочного устройства*, который и использует для последующего поиска и подключения архива студента.

После нормального запуска, ОС УПК АСУ *автоматически подключает к системе* (***login***) пользователя с именем *asu*, как это описано в руководстве [4].

На рабочем столе пользователя отображается множество значков, среди которых имеется стилизованный значок личного ***flashUSB*** студента. Наведя курсор мыши на значок блочного устройства, можно узнать его состояние: подключено или не подключено.

Блочное устройство (съёмный том) может находиться в двух состояниях:

- *подключено* — блочное устройство *подмонтировано* к некоторой директории файловой системы ОС и его содержимое *доступно* ПО ОС;
- *не подключено* — блочное устройство *не подмонтировано* к файловой системе ОС и его содержимое *не доступно* ПО ОС;

Подключение блочного устройства осуществляется левой кнопкой мыши, при этом запускается файловый менеджер ***Thunar***.

Меню работы с блочным устройством активируется правой кнопкой мыши, в котором имеются пункты:

- *Открыть* — при необходимости подключает устройство и запускает файловый менеджер Thunar;
- *Подключить том/Отключить том* — появляются в зависимости от состояния устройства, причём *подключение тома не запускает Thunar*;
- *Извлечь…* - появляется только для съёмных устройств типа ***flashUSB***;
- *Свойства* — доступно только для подключённых устройств;
- *Меню приложений* — второй уровень меню.

Замечание

«Корневая» файловая система не отображается значком на рабочем столе пользователя, поэтому flashUSB не отображается при аварийном варианте запуска.

Рабочая среда пользователя *asu* находится в памяти ЭВМ и расходует её в процессе осуществления всех действий студента. Поэтому она используется только для выполнения следующих служебных операций:

- *подключение/отключение* ***flashUSB*** студента по отношению к ОС ЭВМ;
- *подключение* архива темы обучения к рабочей среде пользователя *upk*;
- *отключение* архива темы обучения от рабочей среды пользователя *upk*;
- *выключение* компьютера;
- *проведение служебных операций*: форматирование ***flashUSB***, создание файловой системы на ***flashUSB***, установка на ***flashUSB*** ПО GRUB и ПО аварийного варианта загрузки ОС.

При запуске ОС УПК АСУ, рабочая среда пользователя *asu* восстанавливается из архива ОС.

Рабочая среда пользователя *upk* присутствует и используется во множестве вариантах:

- *базовый вариант* - рабочая среда пользователя *upk*, которая восстанавливается из архива ОС, во время её запуска;
- *учебные варианты* — рабочие среды, находящиеся в личных архивах студентов, которые подключаются и отключаются от системы из среды пользователя *asu*.

Назначение базового варианта рабочей среды — предупреждение студента, что он или забыл подключить тему обучения, или — при подключении темы возникли проблемы.

Назначение учебных вариантов рабочих сред — подключение их к ОС УПК АСУ с целью:

- *обеспечение студента* учебным материалом и инструментами, во время проведения занятий по конкретной дисциплине;
- *оперативное сохранение* данных в личном архиве студента, во время проведения учебного занятия.

Замечание

Личный архив студента, на его личном flashUSB, имеет ограниченный объём файловой системы, большая часть которой занята учебным материалом и системным ПО ОС.

Первоначальный размер файла архива *os-home.ext4fs* - 300 Мбайт, поэтому:

- ***не следует хранить*** в личном архиве пользователя *upk* посторонние файлы, кроме тех, что предусмотрены для проведения занятий по изучаемой дисциплине;
- ***обязательно сохранять*** на личном flashUSB студента, за пределами архива, копию отчёта по даной дисциплине, шаблон которого представлен на рабочем столе пользователя *upk*, в виде файла *ОтчетОС1.odt*;
- ***в случае*** повреждения архива или его полного заполнения, *следует обратиться к преподавателю* для консультаций по устранению возникших проблем.

Перед окончанием занятия, студент должен:

- *закрыть* все окна и остановить все приложения в среде пользователя *upk*;
- *выйти* из среды пользователя *upk* и зайти в среду пользователя *asu*;

- *отключить* личный архив, воспользовавшись значком на рабочем столе;
- *провести архивирование* личной рабочей среды на личный ***flashUSB***;
- *отключить* личный flashUSB, воспользовавшись значком его блочного устройства, также расположенном на рабочем столе;
- *выключить* компьютер.

2.3 Изучение рабочей среды пользователя upk

Учебное задание

Находясь в середе пользователя ***upk***, выполнить задания подразделов 1.2 и 1.3 из учебно-методического пособия [4].

Прочитать и усвоить учебный материал раздела 3 из пособия [4].

Отразить результаты работы в своём личном отчёте.

Для изучения рабочей среды пользователя имеется три основных инструмента:

- *командная строка* (виртуальный терминал, консоль), в которой можно запускать команды языка ***shell***;
- *текстовый файловый менеджер* (***Midnight Commander***), который запускается в окне виртуального терминала (командной строке) командой **mc**;
- *графический файловый менеджер* (***Thunar***), который запускается разными способами, например, активацией левой кнопкой мыши значка «***Домашний каталог***», расположенного на рабочем столе любого пользователя.

Инструмент командной строки является самым главным в нашем курсе обучения, поскольку командные языки ***shell*** (*sh*, *bash* и другие) являются основой управления системным ПО любой ОС.

В частности, изучению языка ***sh*** посвящена «*Тема 3*» нашего курса обучения.

В качестве примера, рассмотрим один из способов текущего контроля *размера использованной области* пользователя *upk*:

```
du -hs /home/upk
```

Вывод этой команды показан на рисунке 2.2.

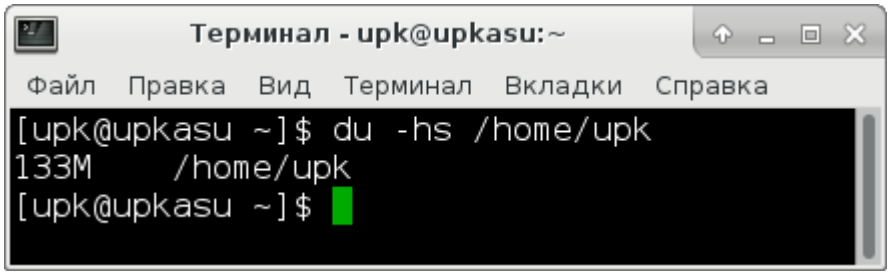

Рисунок 2.2 — Контроль размера содержимого отдельного каталога

Другой пример. На рисунке 2.3 показан вывод команды: ***ls -la***.

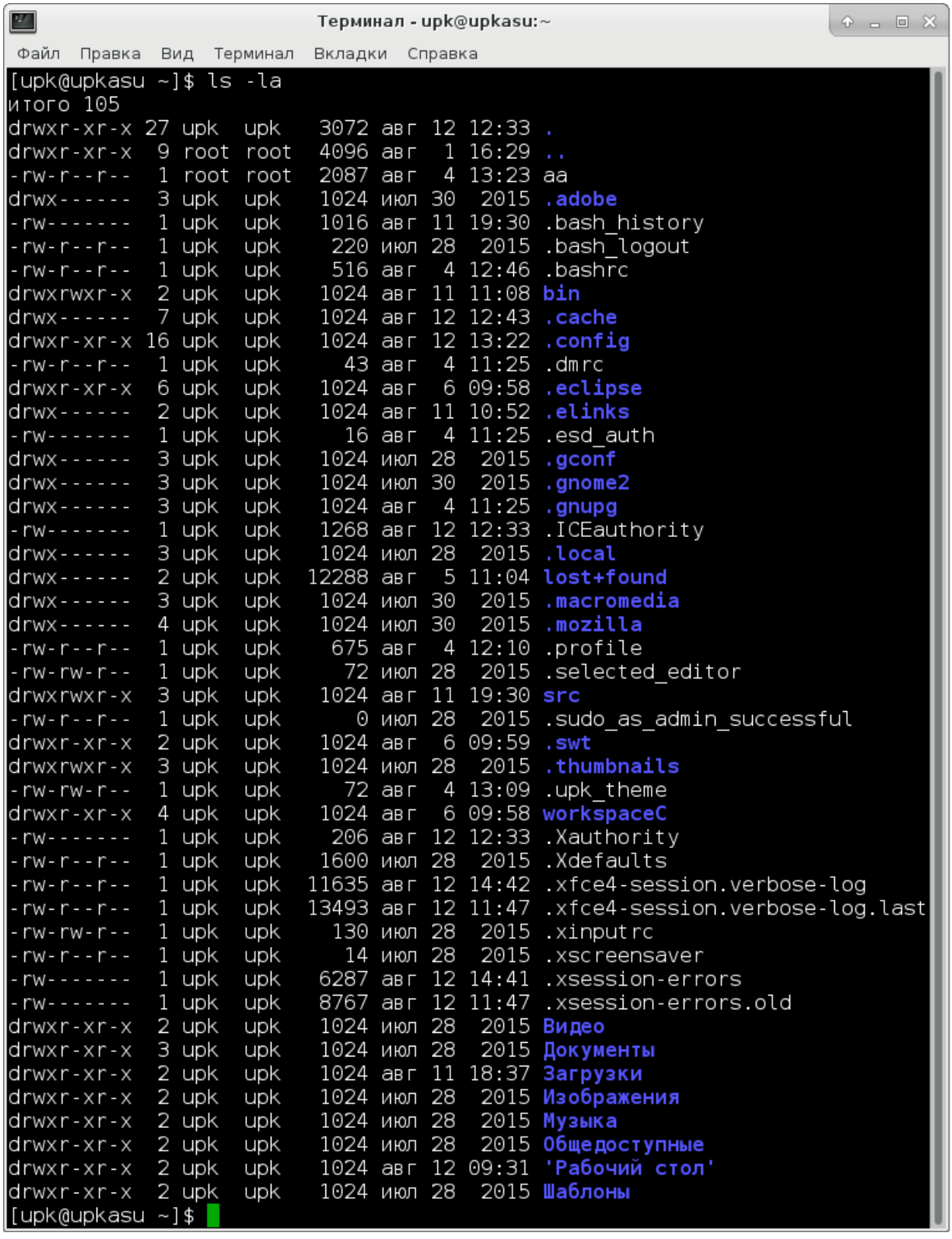

Рисунок 2.3 — Контроль содержимого домашнего каталога пользователя upk

Завершая первую лабораторную работу, рассмотрим основное место хранения учебного материала по данной дисциплине. Оно расположено в домашней директории пользователя *upk*: *~/Документы*.

Доступ к этой директории проще всего получить, активировав левой кнопкой мыши значок «*Учебный материал*», расположенный на рабочем столе пользователя.

В результате, запустится файловый менеджер ***Thunar***, который покажет содержимое директории */home/upk/Документы*, как показано на рисунке 2.4.

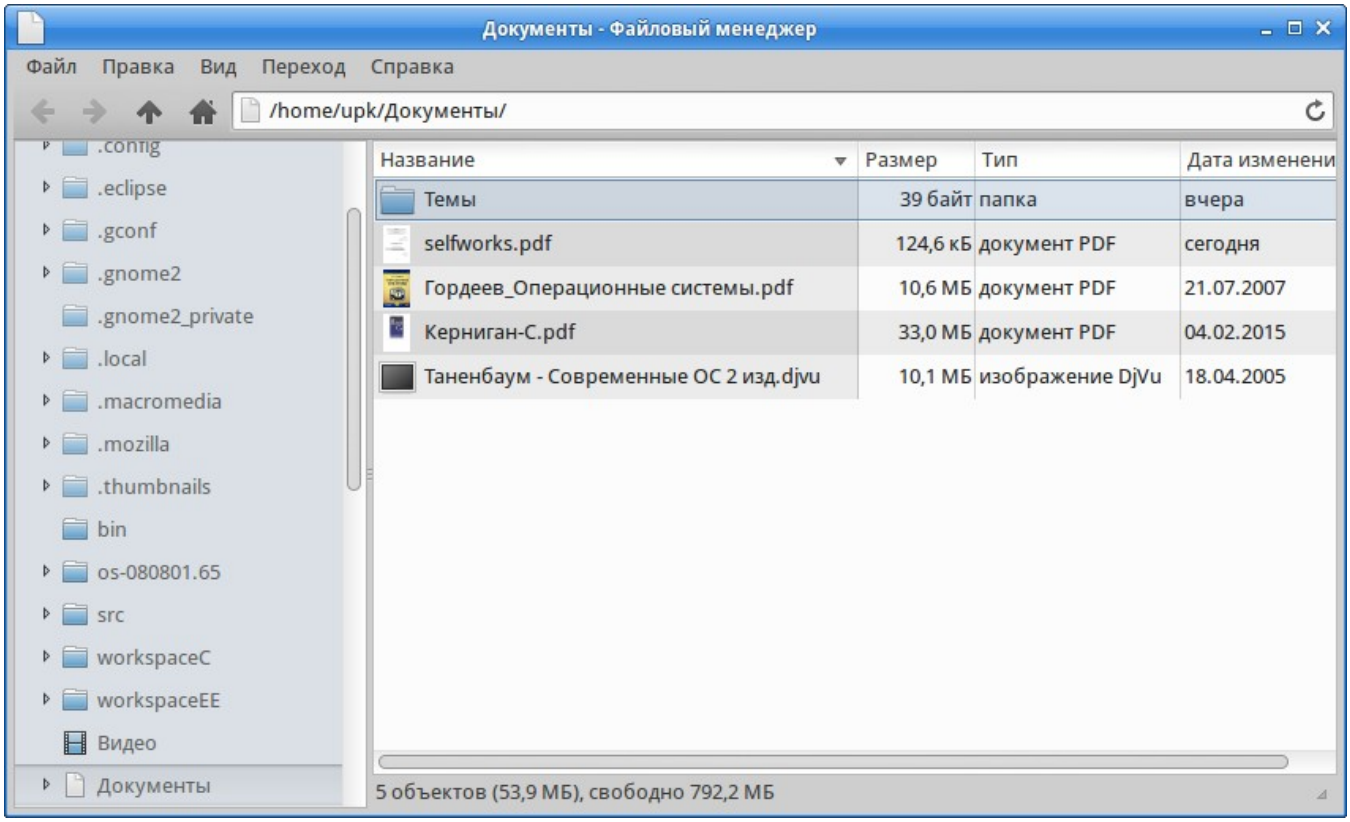

Рисунок 2.4 — Директория «Документы» пользователя upk

В этой директории находятся:

- *общая учебная литература* по данной дисциплине;
- *директория* «*Темы*», в которой находятся файлы учебно-методических пособий.

Обозначения файлов директории «Темы»:

```
    ТемаX_os.pdf,
```

где ***X*** — номер изучаемой темы по дисциплине «*Операционные системы*».

Далее, следует:

- отобразить в отчёте изученный материал;
- выключить компьютер, завершив лабораторную работу №1.

Список использованных источников

1 Резник В.Г. Операционные системы. Самостоятельная и индивидуальная работа студента. Учебно-методическое пособие. – Томск, ТУСУР, 2016. – 13 с.

2 Гордеев А.В. Операционные системы: учебное пособие для вузов. ─ СПб.: Питер, 2004. ─ 415с.

3 Таненбаум Э., Бос Х. Современные операционные системы. - СПб.: Питер, 2015. - 1120с.

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

5 Поль Коббо. Фундаментальные основы Linux, 2014/Перевод А. Панина. - [Электронный ресурс]. - Режим доступа: - Фундаментальные основы Linux__by Paul Cobbaut .pdf.