Автономная некоммерческая образовательная организация
высшего образования
«Воронежский экономико-правовой институт»
(АНОО ВО «ВЭПИ»)

АНОО ВО "ВОРОНЕЖСКИЙ ЭКОНОМИКО-ПРАВОВОЙ ИНСТИТУТ", АНОО ВО "ВЭПИ", ВЭПИ

Подписано цифровой подписью: АНОО ВО "ВОРОНЕЖСКИЙ ЭКОНОМИКО-ПРАВОВОЙ ИНСТИТУТ", АНОО ВО "ВЭПИ", ВЭПИ
Дата: 2021.01.01 13:55:17 +03'00'

УТВЕРЖДАЮ
Проректор
по учебно-методической работе
_____ А.Ю. Жильников
«___» _____ 2018 г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Б1.О.13 Операционные системы
(наименование дисциплины (модуля))

09.03.03 Прикладная информатика
(код и наименование направления подготовки)

Направленность (профиль) Прикладная информатика в экономике
(наименование направленности (профиля))

Квалификация выпускника Бакалавр
(наименование квалификации)

Форма обучения Очная, заочная
(очная, заочная)

Рекомендованы к использованию Филиалами АНОО ВО «ВЭПИ»

Воронеж 2018

Методические рекомендации по выполнению лабораторных работ по дисциплине (модулю) рассмотрены и одобрены на заседании кафедры прикладной информатики.

Протокол от «13» декабря 2018 г. № 5

Заведующий кафедрой  Г.А. Курина

Разработчики:

Доцент  А. И. Кустов

ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа № 1
«Общие сведения об операционных системах, средах и оболочках»

Цель работы: знать общие сведения об операционных системах, средах и оболочках.

1. Краткие теоретические сведения

ЭВМ первого поколения (40 и начало 50 годов) практически не имели ОС. Программы писались непосредственно в машинных кодах, что, в частности, требовало поддержки доступа программы к памяти на этапе ее написания. Поэтому разработка программного продукта наталкивалась на ряд сложностей. Для ЭВМ второго поколения были созданы простейшие ОС, которые отчасти позволили "разделить" среду разработки программ и аппаратные средства. Однако, достигнутого на этом этапе «уровня абстрагирования» было явно не достаточно для разработки и сопровождения относительно сложных задач. По структуре и функциям эти ОС существенно отличались от современных.

Современные контуры ОС стали приобретать в конце 60 годов, когда появились достаточно мощные ЭВМ третьего поколения. Становление ОС на этом этапе ниже будет рассмотрено более подробно.

Позднее (через полтора десятка лет) ЭВМ третьего поколения стали вытесняться более мобильными ЭВМ 4-ого поколения. К их числу, в частности, относятся самые распространенные в настоящее время персональные компьютеры семейства IBM PC. При разработке ОС для этих компьютеров были учтены не только опыт, но и горькие уроки, полученные в результате эксплуатации первых операционных систем и требовала хорошего знания аппаратных средств.

Основные функции операционных систем.

Современные ОС - широко распространенные системы - во многом похожи друг на друга. Прежде всего, это определяется требованием переносимости программного обеспечения. Именно для обеспечения этой переносимости был принят POSIX (Portable OS Interface based on uniX) - стандарт, определяющий минимальные функции по управлению файлами, межпроцессному взаимодействию и т.д., которые должна уметь выполнять система.

Кроме того, за четыре с лишним десятилетия, прошедших с момента разработки первых ОС, сообщество программистов достигло определенного понимания того, что: при разработке ОС возникает много стандартных проблем и вопросов; для большинства из этих проблем и вопросов существует набор стандартных решений; некоторые из этих решений намного лучше, чем все альтернативные.

По современным представлениям, ОС должна уметь делать следующее:

Обеспечивать загрузку пользовательских программ в оперативную память и их исполнение.

Обеспечивать работу с устройствами долговременной памяти, такими как магнитные диски, ленты, оптические диски и т.д. Как правило, ОС управляет свободным пространством на этих носителях и структурирует пользовательские данные.

Предоставлять более или менее стандартный доступ к различным устройствам ввода/вывода, таким как терминалы, модемы, печатающие устройства.

Предоставлять некоторый пользовательский интерфейс. Слово не который здесь сказано не случайно - часть систем ограничивается командной строкой, в то время как другие на 90% состоят из средств интерфейса пользователя.

Более развитые ОС предоставляют также следующие возможности:

1. Параллельное (точнее, псевдопараллельное, если машина имеет только один процессор) исполнение нескольких задач.
2. Распределение ресурсов компьютера между задачами.
3. Организация взаимодействия задач друг с другом.
4. Взаимодействие пользовательских программ с нестандартными внешними устройствами.

Организация межмашинного взаимодействия и разделения ресурсов.

Защита системных ресурсов, данных и программ пользователя, исполняющихся процессов и самой себя от ошибочных и зловредных действий пользователей и их программ.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Основные функции операционных систем, сред и оболочек.
2. История развития и поколения ОС.
3. Требования к современным ОС.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Основные функции операционных систем, сред и оболочек.
2. Функциональные компоненты ОС. Требования к современным ОС.

Лабораторная работа № 2
«Основные функции операционных систем, сред и оболочек»

Цель работы: знать основные функции операционных систем, сред и оболочек.

1. Краткие теоретические сведения

ОС как менеджер ресурсов должна обеспечивать:

• загрузку пользовательских программ в оперативную память;

• выполнение этих программ путем организации работы процессора;

• работу с устройствами долговременной памяти, такими как магнитные диски, ленты, оптические диски и т.д. (как правило, ОС управляет свободным пространством на этих носителях и структурирует пользовательские данные.);

• стандартный доступ к различным устройствам ввода/вывода, таким как терминалы, модемы, печатающие устройства.

При этом в современных вычислительных системах реализуются следующие возможности:

• параллельное (или псевдопараллельное, если машина имеет только один процессор) исполнение нескольких задач;

• распределение ресурсов компьютера между задачами;

• организация взаимодействия задач друг с другом;

• взаимодействие пользовательских программ с нестандартными внешними устройствами;

• организация межмашинного взаимодействия и разделения ресурсов;

• защита системных ресурсов, данных и программ пользователя, исполняющихся процессов и самой себя от ошибочных и зловредных действий пользователей и их программ.

Операционная оболочка (operation shell) - комплекс программ, ориентированных на определенную операционную систему и предназначенный для облегчения диалога между пользователем и компьютером при выполнении определенных видов деятельности на компьютере.

Операционные оболочки дополняют и расширяют пользовательский интерфейс ОС за счет наглядного представления объектов (файлов, каталогов, дисков), использования систем меню и горячих клавиш.

Операционные оболочки предоставляют следующие услуги:

• работа с дисками (просмотр дерева каталогов, получение информации о состоянии диска, форматирование дисков);

• работа с файлами и каталогами (создание, просмотр содержимого, копирование, перенос, переименование, удаление, изменение атрибутов файлов и каталогов; редактирование текстовых файлов; создание архивов);

• дополнительные возможности (подключение к сети, создание пользовательских меню, подключение внешних редакторов и др.).

Под операционной средой (operating environment) понимается комплекс средств, обеспечивающих разработку и выполнение прикладных программ и представляющих собой набор функций и сервисов операционной системы и правил обращения к ним. В общем случае операционная среда включает операционную систему, программное обеспечение, интерфейсы прикладных программ, сетевые службы, базы данных, языки программирования и другие средства выполнения работы на компьютере - в зависимости от решаемых задач. В такой трактовке примерами операционных сред могут служить следующие:

• ОС Windows + Delphi + вспомогательные средства - операционная среда разработчика прикладных приложений;

• ОС Windows + Adobe Photoshop + Adobe Illustrator + Macromedia Dreamweaver + Internet Explorer + вспомогательные средства-операционнаясреда WEBразработчика;

• ОС FreeBSD + WEB-серверApache + сервер СУБД MySQL + интерпретатор PHP + программы защиты + вспомогательныесредства-операционнаясреда для создания приложений, работающих на стороне сервера.

История развития и поколения ОС.

Первый период (1945 -1955).Второй период (1955 - 1965).

Третий период (1965 - 1980).

Четвертый период (1980 - настоящее время).

Классификация ОС.

Для построения классификации ОС, прежде всего, необходимо выбрать основание классификации. Таких оснований множество, но наиболее существенными можно считать следующие:

1. Область использования ОС.
2. Типы аппаратной платформы.
3. Методы проектирования.
4. Реализация внутренних алгоритмов управления ресурсами.

Классификация по области использования:

1. Настольные ОС (Desktop Operating System) - ОС, ориентированные на работу отдельного пользователя в различных предметных областях (разработка программ, работа с документами и т.п.); основными чертами настольных ОС являются универсальность и ориентированность на пользователя; представители-MacOS,Windows.

2. Серверные ОС, использующиеся в серверах сетей как центральное звено, а также в качестве элементов систем управления; основной чертой серверных ОС является надежность; представители-семействоUNIX, Windows NT.

Классификация по области использования:

1. Специализированные ОС, ориентированные на решение узких классов задач с жестким набором требований (высокопроизводительные вычисления, управление в реальном времени); системы такого рода практически неразрывно связаны с аппаратной платформой; представители - QNX,

редуцированные и специализированные версии UNIX, системы собственной разработки

2. Мобильные ОС - вариант развития настольных ОС на аппаратной платформе КПК; основные черты - удобство использования и компактность; представители -PalmOS,Windows CE.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. История развития и поколения ОС.
2. Функциональные компоненты ОС.
3. Мультипрограммирование и распределение ресурсов.
4. Понятие процессов и потоков.
5. Алгоритмы планирования процессов и потоков.
6. Синхронизация процессов.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Мультипрограммирование и распределение ресурсов.
2. Синхронизация процессов.

Лабораторная работа № 3
«Управление памятью»

Цель работы: знать, как правильно управление памятью.

1. Краткие теоретические сведения

Под памятью (memory) здесь подразумевается оперативная память компьютера. В отличие от памяти жесткого диска, которую называют внешней памятью (storage), оперативной памяти для сохранения информации требуется постоянное электропитание.

Память является важнейшим ресурсом, требующим тщательного управления со стороны мультипрограммной операционной системы. Особая роль памяти объясняется тем, что процессор может выполнять инструкции протравы только в том случае, если они находятся в памяти. Память распределяется как между модулями прикладных программ, так и между модулями самой операционной системы.

В ранних ОС управление памятью сводилось просто к загрузке программы и ее данных из некоторого внешнего накопителя (перфоленты, магнитной ленты или магнитного диска) в память. С появлением мультипрограммирования перед ОС были поставлены новые задачи, связанные с распределением имеющейся памяти между несколькими одновременно выполняющимися программами.

Функциями ОС по управлению памятью в мультипрограммной системе являются:

1) отслеживание свободной и занятой памяти;

2) выделение памяти процессам и освобождение памяти по завершении процессов;

3) вытеснение кодов и данных процессов из оперативной памяти на диск (полное или частичное), когда размеры основной памяти не достаточны для размещения в ней всех процессов, и возвращение их в оперативную память, когда в ней освобождается место;

4) настройка адресов программы на конкретную область физической памяти.

Помимо первоначального выделения памяти процессам при их создании ОС должна также заниматься динамическим распределением памяти, то есть выполнять запросы приложений на выделение им дополнительной памяти во время выполнения. После того как приложение перестает нуждаться в дополнительной памяти, оно может возвратить ее системе. Выделение памяти случайной длины в случайные моменты времени из общего пула памяти приводит к фрагментации и, вследствие этого, к неэффективному ее использованию. Дефрагментация памяти тоже является функцией операционной системы.

Во время работы операционной системы ей часто приходится создавать новые служебные информационные структуры, такие как описатели процессов и потоков, различные таблицы распределения ресурсов, буферы,

используемые процессами для обмена данными, синхронизирующие объекты и т. п. Все эти системные объекты требуют памяти»» В некоторых ОС заранее (во время установки) резервируется некоторый фиксированный объем памяти для системных нужд. В других же ОС используется более гибкий подход, при котором память для системных целей выделяется динамически. В таком случае разные подсистемы ОС при создании своих таблиц, объектов, структур и т. п. обращаются к подсистеме управления памятью с запросами.

Защита памяти — это еще одна важная задача операционной системы, которая состоит в том, чтобы не позволить выполняемому процессу записывать или читать данные из памяти, назначенной другому процессу. Эта функция, как правило, реализуется программными модулями ОС в тесном взаимодействии с аппаратными средствами.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Функции ОС по управлению памятью.
2. Типы адресов.
3. Виды алгоритмов распределения памяти.
4. Виртуализация памяти.
5. Классы виртуальной памяти.
6. Кэширование данных.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Функции ОС по управлению памятью.
2. Типы адресов.
3. Виды алгоритмов распределения памяти.

Лабораторная работа № 4
«Ввод-вывод и файловая система»

Цель работы: рассмотреть ввод-вывод и файловую систему

1. Краткие теоретические сведения

Одной из главных задач ОС является обеспечение обмена данными между приложениями и периферийными устройствами компьютера. Собственно ради выполнения этой задачи и были разработаны первые системные программы, послужившие прототипами операционных систем. В современной ОС функции обмена данными с периферийными устройствами выполняет подсистема ввода-вывода. Клиентами этой подсистемы являются не только пользователи и приложения, но и некоторые компоненты самой ОС, которым требуется получение системных данных или их вывод, например подсистеме управления процессами при смене активного процесса необходимо записать на диск контекст приостанавливаемого процесса и считать с диска контекст активизируемого процесса.

Основными компонентами подсистемы ввода-вывода являются драйверы, управляющие внешними устройствами, и файловая система. К подсистеме ввода-вывода можно также с некоторой долей условности отнести и диспетчер прерываний, рассмотренный выше. Условность заключается в том, что диспетчер прерываний обслуживает не только модули подсистемы ввода-вывода, но и другие модули ОС, в частности такой важный модуль, как планировщик/диспетчер потоков. Но из-за того, что планирование работ подсистемы ввода-вывода составляет основную долю нагрузки диспетчера прерываний, его вполне логично рассматривать как ее составную часть (к тому же первопричиной появления в компьютерах системы прерываний были в свое время именно операции с устройствами ввода-вывода).

Файловая система ввиду ее сложности, специфичности и важности как основного хранилища всей информации вычислительной системы заслуживает рассмотрения в отдельной главе. Тем не менее, здесь файловая система рассматривается совместно с другими компонентами подсистемы ввода-вывода по двум причинам. Во-первых, файловая система активно использует остальные части подсистемы ввода-вывода, а во-вторых, модель файла лежит в основе большинства механизмов доступа к устройствам, используемых в современной подсистеме ввода-вывода.

Задачи ОС по управлению файлами и устройствами.

Подсистема ввода-вывода (Input-Output Subsystem) мультипрограммной ОС при обмене данными с внешними устройствами компьютера должна решать ряд общих задач, из которых наиболее важными являются следующие:

1) организация параллельной работы устройств ввода-вывода и процессора;

2) согласование скоростей обмена и кэширование данных;

3) разделение устройств и данных между процессами;

4) обеспечение удобного логического интерфейса между устройствами и остальной частью системы;

5) поддержка широкого спектра драйверов с возможностью простого включения в систему нового драйвера;

6) динамическая загрузка и выгрузка драйверов;

7) поддержка нескольких файловых систем;

8) поддержка синхронных и асинхронных операций ввода-вывода.

В следующих разделах все эти задачи рассматриваются подробно.

Организация параллельной работы устройств ввода-вывода и процессора.

Каждое устройство ввода-вывода вычислительной системы — диск, принтер, терминал и т. п. — снабжено специализированным блоком управления, называемым контроллером. Контроллер взаимодействует с драйвером — системным программным модулем, предназначенным для управления данным устройством. Контроллер периодически принимает от драйвера выводимую на устройство информацию, а также команды управления, которые говорят о том, что с этой информацией нужно сделать (например, вывести в виде текста в определенную область терминала или записать в определенный сектор диска). Под управлением контроллера устройство может некоторое время выполнять свои операции автономно, не требуя внимания со стороны центрального процессора. Это время зависит от многих факторов — объема выводимой информации, степени интеллектуальности управляющего устройством контроллера, быстродействия устройства и т. п. Даже самый примитивный контроллер, выполняющий простые функции, обычно тратит довольно много времени на самостоятельную реализацию подобной функции после получения очередной команды от процессора. Это же справедливо и для сложных контроллеров, так как скорость работы любого устройства ввода-вывода, даже самого скоростного, обычно существенно ниже скорости работы процессора.

Процессы, происходящие в контроллерах, протекают в периоды между выдачами команд независимо от ОС. От подсистемы ввода-вывода требуется спланировать в реальном масштабе времени (в котором работают внешние устройства) запуск и приостановку большого количества разнообразных драйверов, обеспечив приемлемое время реакции каждого драйвера на независимые события контроллера. С другой стороны, необходимо минимизировать загрузку процессора задачами ввода-вывода, оставив как можно больше процессорного времени на выполнение пользовательских потоков.

Данная задача является классической задачей планирования систем реального времени и обычно решается на основе многоуровневой приоритетной схемы обслуживания по прерываниям. Для обеспечения приемлемого уровня реакции все драйверы (или части драйверов) распределяются по нескольким приоритетным уровням в соответствии с требованиями ко времени реакции и временем использования процессора.

Для реализации приоритетной схемы обычно задействуется общий диспетчер прерываний ОС.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Файловая система ОС.
2. Логическая организация файловой системы.
3. Физическая организация файловой системы.
4. Подсистема ввода-вывода.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Файловая система ОС.
2. Подсистема ввода-вывода.

Лабораторная работа № 5
«Архитектура операционных систем»

Цель работы: знать архитектуру операционных систем.

1. Краткие теоретические сведения

Операционная система – это программа, контролирующая работу прикладных программ и системных приложений и исполняющая роль интерфейса между приложениями и аппаратным обеспечением компьютера.

Операционная система выполняет функции управления вычислительными процессами в вычислительной системе, распределяет ресурсы вычислительной системы между различными вычислительными процессами и образует программную среду, в которой выполняются прикладные программы пользователя.

Основная цель ОС, обеспечивающей работу ЭВМ в любом из описанных режимов, - динамическое распределение ресурсов и управление ими в соответствии с требованиями вычислительных процессов (задач).

Операционная система является посредником между ЭВМ и её пользователем. Она делает работу с ЭВМ более простой, освобождая пользователя от обязанностей распределять ресурсы и управлять ими. Операционная система осуществляет анализ запросов пользователя и обеспечивает их выполнение. Запрос отражает необходимые ресурсы и требуемые действия ЭВМ и представляется последовательностью команд на особом языке директив операционной системы. Такая последовательность команд называется заданием.

Основными функциями, которые выполняются операционной системой, являются:

1) приём от пользователя заданий или команд, сформулированных на соответствующем языке и их обработка;

2) приём и исполнение программы запроса на запуск или приостановку других программ;

3) загрузка в оперативную память программ, подлежащих исполнению;

4) инициализация программ (передача им управления), в результате чего процессор использует программы;

5) идентификация программ;

6) обеспечение работы системы управления файлами базы данных, что позволяет резко увеличить эффективность программного обеспечения;

7) обеспечение режима мультипрограммирования, т.е. выполнение двух или более программ на одном процессоре, создающая видимость их одновременного исполнения;

8) обеспечение функции по организации и управлению всеми операциями ввода и вывода;

9) удовлетворение жёстким ограничениям на время выполнения в режиме реального времени;

10) распределение памяти:

а) организация виртуальной памяти;

б) в большинстве современных систем.

11) планирование и диспетчеризация в соответствии с заданием на выполнение;

12) организация обмена сообщениями и данными между выполняющимися программами;

13) защита одной программы от влияния других программ, обеспечение сохранности данных;

14) предоставление услуг на случай частичного сбоя системы;

15) обеспечение работы систем программирования, с помощью которых пользователи готовят свои программы.

Ядро и вспомогательные модули операционной системы.

Любая сложная система должна иметь понятную и рациональную структуру, то есть разделяться на части — модули, имеющие вполне законченное функциональное назначение с четко оговоренными правилами взаимодействия. Ясное понимание роли каждого отдельного модуля существенно упрощает работу по модификации и развитию системы.

Функциональная сложность операционной системы неизбежно приводит к сложности ее архитектуры, под которой понимают структурную организацию ОС на основе различных программных модулей. Обычно в состав ОС входят исполняемые и объектные модули стандартных для данной ОС форматов, библиотеки разных типов, модули исходного текста программ, программные модули специального формата (например, загрузчик ОС, драйверы ввода-вывода), конфигурационные файлы, файлы документации, модули справочной системы и т. д.

Ядро и вспомогательные модули ОС.

Наиболее общим подходом к структуризации операционной системы является разделение всех ее модулей на две группы:

1) ядро — модули, выполняющие основные функции ОС;

2) модули, выполняющие вспомогательные функции ОС.

Модули ядра выполняют такие базовые функции ОС, как управление процессами, памятью, устройствами ввода-вывода и т. п. Ядро составляет сердцевину операционной системы, без него ОС является полностью неработоспособной и не сможет выполнить ни одну из своих функций.

В состав ядра входят функции, решающие внутрисистемные задачи организации вычислительного процесса, такие как переключение контекстов, загрузка/выгрузка станиц, обработка прерываний. Эти функции недоступны для приложений. Другой класс функций ядра служит для поддержки приложений, создавая для них так называемую прикладную программную среду. Приложения могут обращаться к ядру с запросами — системными вызовами — для выполнения тех или иных действий, например для открытия и чтения файла, вывода графической информации на дисплей, получения системного времени и т. д. Функции ядра, которые могут вызываться приложениями, образуют интерфейс прикладного программирования — API.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Архитектура на базе ядра в привилегированном режиме.
2. Микроядерная архитектура.
3. Переносимость ОС.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Архитектура на базе ядра в привилегированном режиме.
2. Переносимость ОС.

Лабораторная работа № 6
«История развития операционных систем и эволюция их функциональных характеристик»

Цель работы: знать историю развития операционных систем и эволюцию их функциональных характеристик.

1. Краткие теоретические сведения

История операционной системы UNIX началась в 1969 году с совместного проекта Массачусетского технологического института, исследовательской лаборатории Bell Labs и корпорации General Electric – системы MULTICS (Multiplexed Information and Computing Service – информационно-вычислительная служба с мультиплексированием каналов передачи данных). Ее разработчики поставили своей задачей создание большой и сложной системы общего назначения с разделением времени. Лаборатория Bell Labs вскоре вышла из числа участников проекта, после чего один из ее сотрудников, Кен Томпсон (Ken Thompson), к которому затем присоединился Денис Ритчи (Dennis Ritchie), создал усеченный вариант системы MULTICS для миникомпьютера PDP-7. Эта система в шутку была названа UNICS (UNiplexed Information and Computing Service – примитивная информационная и вычислительная служба). В дальнейшем, сохранив то же произношение, система UNICS получила несколько «сокращенное» написание в виде UNIX. UNIX привлекла внимание других специалистов, группа разработчиков увеличилась, и система была перенесена на более современные вычислительные машины PDP-11 (доминирующие в нише миниВМ в 1970-е годы), причем ее переписали на языке программирования высокого уровня С, специально созданном для этой цели Денисом Ритчи. Это событие произошло в 1973 году, когда еще не было принято писать операционные системы на высокоуровневых языках, но время для этого уже пришло, поскольку проблема переносимости системного программного обеспечения между различными аппаратными платформами стояла довольно остро. UNIX обладала огромным преимуществом перед другими ОС – она была переносимой и могла быть установлена, по крайней мере потенциально, на ВМ любой архитектуры.

Поначалу UNIX была сравнительно маленькой, очень простой в использовании и понятной системой. Первая ее версия, однопользовательская, вскоре была расширена и стала поддерживать многопользовательский доступ. Разработчики стремились максимально упростить структуру операционной системы, пусть даже за счет снижения эффективности работы и удаления некоторых функций, которыми на то время обладало большинство подобных систем. UNIX быстро приобрела популярность не только в Bell Labs, но и за ее пределами. Корпорация AT&T (учредитель лаборатории Bell Labs) стала вкладывать средства в развитие UNIX, в результате чего было выпущено несколько новых версий ОС, архитектура которых все более усложнялась. Конечным результатом усилий

корпорации AT&T стал продукт под названием System V, развитие которого затем было продолжено уже другими компаниями.

Параллельно с указанной ветвью операционной системы UNIX специалисты Калифорнийского университета в Беркли разрабатывали еще одну, названную BSD (Berkeley Software Distribution – программное изделие Калифорнийского университета). Она тоже получила широкое распространение, а впоследствии на основе обеих ветвей был создан ряд новых версий ОС.

Классической UNIX принято считать так называемую седьмую версию ОС, которая является исходной точкой двух основных ветвей развития данной архитектуры: System V и BSD.

Третья самостоятельная ветвь развития UNIX началась с попытки вернуться к исходному свойству этой операционной системы – предельной простоте. Данная ветвь начинается с микроядерной системы MINIX, за которой последовала значительно более мощная система Linux, разработанная Линусом Торвальдсом (Linus Torvalds) и представленная им в 1991 г. в качестве первой официальной версии 0.02. В настоящее время Linux – это полноценная многозадачная многопользовательская операционная система семейства UNIX. Одним из наиболее удачных вариантов реализации Linux является дистрибутивный комплект Red Hat Linux. Главными особенностями, выделяющими его на фоне других версий Linux, являются, во-первых, наличие средств управления пакетами (Red Hat Package Manager, RPM), служащих для установки программ, проверки их целостности, обновления и удаления программного обеспечения, а во-вторых, наличие графической панели управления (Control panel), с помощью которой можно осуществлять контроль практически за всеми ресурсами ВМ.

Для совместимости различных версий операционной системы UNIX и производных от нее систем Международный институт инженеров по электротехнике и электронике IEEE разработал стандарт системы UNIX, называемый POSIX, который теперь поддерживают большинство версий UNIX. Стандарт POSIX определяет минимальный интерфейс системного вызова, необходимый для совместимости UNIX-систем. Некоторые другие операционные системы в настоящее время тоже поддерживают интерфейс POSIX.

На сегодняшний день существуют версии UNIX для многих ВМ – от персональных компьютеров до суперВМ. Независимо от версии, общими для UNIX чертами являются:

– многопользовательский режим со средствами защиты данных от несанкционированного доступа;

– реализация мультипрограммного режима разделения времени, основанного на использовании алгоритмов вытесняющей многозадачности;

– использование механизмов виртуальной памяти и свопинга;

– унификация операций ввода-вывода на основе расширенного использования понятия «файл»;

– иерархическая файловая система, образующая единое дерево каталогов независимо от количества физических устройств, используемых для размещения файлов;

– переносимость системы за счет написания ее основной части на языке C;

– разнообразные средства взаимодействия процессов, в том числе и через сеть;

– кэширование диска для уменьшения среднего времени доступа к файлам.

Хотя многие пользователи UNIX, особенно опытные программисты, предпочитают командный интерфейс графическому, почти все UNIX-системы поддерживают оконную систему, созданную в Массачусетсском технологическом институте. Она называется X Windows. Эта система оперирует основными функциями окна, позволяя пользователю создавать, удалять, перемещать окна и изменять их размеры с помощью мыши. Часто поверх системы X Windows может быть установлен полный графический интерфейс, например Motif, придающий системе UNIX внешний вид системы типа Microsoft Windows или системы компьютеров Macintosh.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Операционные системы разных этапов разработки вычислительных машин.
2. Характеристики операционных систем UNIX.
3. Характеристики операционных систем семейства Windows.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Операционные системы разных этапов разработки вычислительных машин.
2. История развития и характеристики операционных систем UNIX.
3. История развития и характеристики операционных систем семейства Windows.