Тема: Операционное окружение (операционная среда)

1. **Введение**
   - **1.1. Системное программное обеспечение**
   - **1.2. Состав СПО**
   - **1.3. ОС**
   - **1.4. Система управления файлами СУФ**
   - **1.5. Операционная среда**
   - **1.6. Система программирования**
   - **1.7. Утилиты**
2. **Понятие операционной среды**
3. **Понятие вычислительного процесса и ресурса**
   - **3.1. Программа**
   - **3.2. Процесс (задача)**
   - **3.3. Ресурс**
   - **3.4. Интерфейс**
4. **Основные виды ресурсов**
   - **4.1. Процессор**
   - **4.2. Оперативная память**
   - **4.3. Внешняя память**
   - **4.4. Программные модули**
   - **4.5. Информационные ресурсы**
5. **Функциональные компоненты операционной системы**
   - **5.1. Управление процессами**
   - **5.2. Управление памятью**
   - **5.3. Управление файлами и внешними устройствами**
   - **5.4. Защита данных и администрирование**
   - **5.5. Интерфейс прикладного программирования**
   - **5.6. Пользовательский интерфейс**
6. **ОС как виртуальная (расширенная) машина**
7. **Режим супервизора и пользователя**
8. **Требования к современным ОС**
9. **Выводы**

1. Введение

1.1. **Системное программное обеспечение** – это программы и комплексы программ, общие для всех, кто совместно использует технические средства компьютера, и применяемые для выполнения существующих программ и автоматизации разработки новых.

1.2. Состав СПО

Системное программное обеспечение состоит из 5-ти групп:

1. операционные системы;
2. системы управления файлами (СУФ); системы управления базами данных (СУБД)
3. интерфейсные оболочки для взаимодействия пользователя с ОС и программные среды;

4. системы программирования;
5. утилиты.

Рассмотрим краткое описание каждой из групп.

1.3. ОС

В данном случае **ОС выступает как** комплекс **управляющих** и **обрабатывающих программ**, интерфейс между аппаратурой компьютера и пользователем с его задачами, предназначенный для эффективного использования ресурсов вычислительной системы.

Тогда формализованный с точки зрения операционной среды **перечень функций ОС** выглядит следующим образом:

- **прием от пользователя заданий или команд**, выданных в виде командной строки или с помощью манипулятора (мыши);
- **прием и исполнение программных запросов** на запуск, приостановку или остановку других программ;
- **загрузка в оперативную память подлежащих исполнению программ**;
- **инициирование программы** (передачи управления на ее выполнение);
- **идентификация** всех программ и данных;
- **обеспечение работы систем управления файлами** (СУФ) и/или систем управления базами данных (СУБД);
- **обеспечение режима мультипрограммирования**, выполнение двух и более задач на одном процессоре;
- **организация и управление операциями ввода/вывода**;
- обеспечение минимального времени ответа в системах реального времени;
- **распределение памяти**, организация виртуальной памяти;
- **планирование и диспетчеризация заданий** в соответствии с заданной дисциплиной обслуживания;
- **обмен сообщениями и данными** между выполняющимися программами;
- **защита одной программы от влияния другой**, сохранность данных;
- предоставление **услуг на случай частичного сбоя системы**;
- обеспечение работы **систем программирования**.

1.4. **Система управления файлами (СУФ)** предназначена для организации более удобного доступа к данным, организованным в файлы.

Все современные ОС имеют соответствующие системы управления файлами, однако, ряд ОС позволяют работать с несколькими файловыми системами (даже одновременно). Эта возможность обеспечивается монтированием файловых систем.

1.5. **Интерфейсная оболочка** предназначена для удобства взаимодействия пользователя с ОС. Назначение – расширить возможности по управлению ОС или изменить встроенные в систему возможности. Примеры: Explorer, X Window, эмуляторы).

Операционная среда – интерфейс, необходимый программам для обращения к ОС с целью получить определенный сервис.

1.6. Система программирования

Системой программирования называется комплекс программ, предназначенный для автоматизации программирования задач на ЭВМ. Система программирования освобождает проблемного пользователя или прикладного программиста от необходимости написания программ решения своих задач на неудобном для него языке машинных команд и предоставляют им возможность использовать специальные языки более высокого уровня. Для каждого из таких языков, называемых входными или исходными, система программирования имеет программу, осуществляющую автоматический перевод (трансляцию) текстов программы с входного языка на язык машины. Обычно система программирования содержит описания применяемых языков программирования, программы-трансляторы с этих языков, а также развитую библиотеку стандартных подпрограмм.

Поэтому можно записать компоненты СП:

- транслятор с соответствующего языка программирования (транслятор – программа, переводящая исходную программу в объектный модуль – на машинный язык);
- библиотеки подпрограмм;
- редакторы;
- компоновщики (программа, которая производит *компоновку* — принимает на вход один или несколько объектных модулей и собирает по ним исполнимый модуль);
- отладчики ((debugger) является модулем среды разработки или отдельным приложением, предназначенным для поиска ошибок в программе)

Самостоятельных (вне ОС) систем программирования не бывает.

1.7. Утилиты – это специальные системные программы, с помощью которых можно как обслуживать саму ОС, так всю вычислительную систему.

Функции утилит (например):

- подготовка для работы носителей данных;
- перекодировка;
- оптимизация размещения данных на диске;
- разбиение накопителя на магнитных дисках на разделы;
- форматирование;
- архивирование данных.

2. Понятие операционной среды

Напомним, что

Назначение операционной системы:

- управление вычислительными процессами в вычислительной системе;
- распределение ресурсов вычислительной системы между различными вычислительными процессами;
- образование программной (операционной) среды, в которой выполняются

прикладные программы пользователей.

Т.е. ОС - программная подсистема, при обращении к которой посредством соответствующих вызовов пользователь получает функции и сервисы.

Та вот:

Операционная среда – набор функций и сервисов ОС и правила обращения к ним. Операционная среда – набор интерфейсов, необходимый программам и пользователям для обращения к ОС с целью получить определенные сервисы. Это совокупность компьютерных программ, обеспечивающая оператору возможность управлять вычислительными процессами и файлами

Операционная система в общем случае может содержать несколько операционных сред.

Операционная среда может включать несколько интерфейсов: пользовательские и программные.

Операционная среда – системное программное окружение, в котором могут выполняться программы, созданные по правилам работы этой среды.

Стандартом на операционные системы (ОС) определены синтаксис и семантика языка оболочки и утилит, составляющих операционную среду компьютера, работающего под управлением такой ОС.

Есть операционные среды, позволяющие управлять вычислительными процессами и файлами в стандартной операционной системе посредством графического пользовательского интерфейса, такие как Enlightenment, GNOME, KDE и пр.

Существуют также операционные среды, предназначенные для работы под альтернативными операционными системами (например, GEOS, Microsoft Windows 1.0-3.x, 95, 98 и ME на базе MS-DOS), причем сама ОС может включаться в поставку операционной среды, а некоторые альтернативные ОС (например, Windows NT, Mac OS), наоборот, включают графические операционные среды в свой состав.

Проявляется тенденция включать в операционные среды также нетрадиционные средства ввода-вывода данных (голосовой ввод, синтез голоса, распознавание рукописного ввода и др.).

3. Понятие вычислительного процесса и ресурса

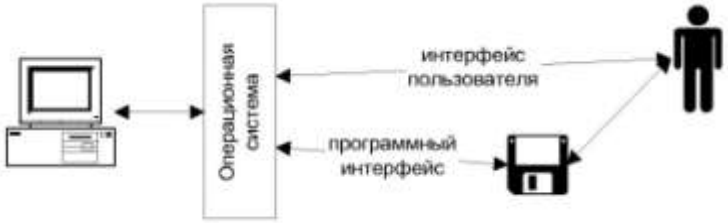
3.1. Одной из основных функций ОС является организация работы программного обеспечения ЭВМ. Под **программой** понимается логически завершенный набор команд процессора..

Особенности Программ:

- Программа хранится на внешнем носителе в виде файла или совокупности файлов
- В процессе выполнения программа создает в памяти различные данные.
- Для обеспечения работы программы ОС создает структуры, содержащие различную информацию об этой программе. Это позволяет рассматривать выполняющуюся программу как объект управления со стороны ОС.
- Для обозначения программы, находящейся в стадии выполнения используется понятие процесса или задачи.

3.2. **Процесс (или задача**) - последовательность команд, выполняемых программой или логически выделенной ее частью, и совокупность используемых ею данных. В этом смысле процесс можно понимать как выполняемую программу. При этом выполнение одной программы может порождать несколько процессов. Процесс является наименьшей единицей работы, для которой выделяются ресурсы компьютера.

Основными задачами, решаемыми ОС в рамках управления ресурсами, являются упрощение доступа к ресурсам и распределение ресурсов между конкурирующими за них



процессами и устройствами. Для решения этих задач ОС организует т.н. интерфейсы (рис 3).

С процессом связано понятие ресурса. Термин **ресурс** относится к используемым, относительно стабильным и часто недостающим объектам, которые запрашиваются, используются и освобождаются процессами в период их активности.

3.3. **Ресурс** -логический или физический компонент ЭВМ и предоставляемые им возможности. Основные ресурсы: процессор, ОП, внешняя память, порты ввода-вывода, номера обработчиков прерываний, каналы DMA.
Распределение ресурсов процессора между несколькими процессами является одной из основных зададач т.н. многозадачных ОС. Назначение различным ПУ (периферийные стройства) различных адресов портов ввода-вывода, различных номеров прерываний является необходимым условием успешной, бесконфликтной работы этих ПУ.

Ресурсы могут быть:

- разделяемыми;
- неделимыми.

Разделяемые ресурсы могут использоваться:

- одновременно (в один и тот же момент времени);
- параллельно (в течение некоторого отрезка времени процессы используют ресурс попеременно).

Разделяемые ресурсы (пример)

В современном операционном окружении программист не может быть уверен и не должен полагаться на то, что коды его программы будут выполняться в тон же последовательности, в какой они написаны. Выполнение одной из функций программы может быть остановлено системой и возобновлено позднее, причем это может произойти даже при выполнении тела какого-либо цикла. При проектировании многопотоковых приложений следует иметь в виду, что ресурсы, разделяемые потоками (блоки памяти или файлы), можно неосознанно повредить. Чтобы показать, как это происходит, рассмотрим пример, который приведен в книге Jesse Liberty «Beginning Object-Oriented Analysis and Design with C++» (Дж. Либерти «Начало

объектно-ориентированного анализа и проектирования с помощью C++»), доступной в MSDN.

Представьте себе пассажирский авиалайнер в полете, а в нем такой разделяемый всеми ресурс, как туалетная комната. Создатели самолета предполагали, что только одна персона может занимать эту комнату. Первый, кто ее занял, закрывает (lock) доступ к пей для всех остальных. Следующий пассажир, желающий воспользоваться этим ресурсом, может либо терпеливо ожидать освобождения, либо по истечении какого-то времени (time out) вернуться на свое сиденье и продолжать заниматься тем, чем он был занят до этого события. Решение о том, что выбрать и как долго ждать, принимает пассажир. Блокирование ресурса порождает неэффективное проведение времени второго пассажира как ожидающего очереди, так и избравшего другую тактику.

Возвращаясь к многопотоковым процессам, отметим, что если не блокировать ресурс, то становится возможным повреждение данных. Представьте, что один поток процесса проходит по записям базы данных, повышая зарплату каждому сотруднику на 10%, а другой поток в это же время изменяет почтовые индексы в связи с введением нового стандарта. Согласитесь с тем, что разумно совместить эти две работы в одном процессе с целью повышения производительности. Что может произойти, если не блокировать доступ к записи при ее модификации? Первый поток прочел запись (все ее поля), и занят вычислением повышения (предположим, с $80 000 до $85 000). В это время второй поток читает эту же запись с целью изменения почтового индекса. В этой ситуации может произойти следующее: первый поток сохраняет измененную запись с новым значением зарплаты, а второй, возвращая запись с измененным индексом, реставрирует значение зарплаты и данный сотрудник останется без повышения. Это происходит по причине того, что оба потока не могут обратиться к части записи и поэтому работают со всей записью, хотя модифицируют только отдельные ее поля.
Для того чтобы исключить подобный сценарий, автор многопотокового приложения должен решать **проблему синхронизации при попытке одновременного доступа к разделяемым ресурсам**

Система запросов на ресурс

Мультипрограммный режим работы вычислительной системы заключается в том, что пока одна программа (процесс, задача) ожидает завершения очередной операции ввода/вывода, другая программа (задача) может быть поставлена на выполнение.

При мультипрограммировании повышается пропускная способность системы, но отдельный процесс никогда не может быть выполнен быстрее, чем, если бы он выполнялся в однопрограммном режиме.

ОС поддерживает мультипрограммирование (многопроцессность) и старается эффективно использовать ресурсы путем организации очередей запросов.

При необходимости использовать какой-либо ресурс процесс обращается к супервизору ОС и сообщает ему свои требования (вид ресурса, объем и т.д.).

Эта директива переводит процессор в **привилегированный режим**, если он есть.

Ресурс будет выделен обратившемуся за ним процессу, если:

- он свободен и нет задач с более высоким приоритетом, обратившимся за этим ресурсом;
- текущий запрос и ранее выданные запросы допускают совместное использование ресурсов;
- ресурс используется задачей с более низким приоритетом и может быть временно отобран.

Если ресурс занят, ОС ставит задачу в очередь к ресурсу, переводя ее в состояние ожидания. Очередь к ресурсу может быть организована различными способами, но обычно с помощью списковой структуры.

После завершения работы с ресурсом задача с помощью системного вызова супервизора сообщает ОС об отказе от ресурса.

Супервизор ОС, получив управление, освобождает ресурс и проверяет, есть ли очередь к этому ресурсу. Если очередь есть, то в зависимости от **дисциплины обслуживания** и приоритетов задач, ожидающих данный ресурс, супервизор выбирает задачу и переводит ее в состояние готовности к выполнению. Управление будет передано либо этой выбранной задаче, либо той, которая только что освободила ресурс.

При выдаче запроса задача может указать, хочет ли она владеть ресурсом монопольно или совместно с другими задачами (файл).

При организации управления ресурсами требуется принять решение о том, что в данной ситуации выгоднее:

- быстро обслуживать отдельные наиболее важные запросы;
- предоставлять всем процессам равные возможности;
- обслуживать максимально возможное количество процессов;
- наиболее полно использовать ресурсы.

3.4. **Интерфейсом** называется набор средств и правил, обеспечивающих взаимодействие физических или логических участников взаимодействия. Выделяют пользовательский и программный интерфейсы ОС.

Пользовательский интерфейс включает в себя набор средств, предназначенных для упрощения взаимодействия пользователя с ЭВМ. ОС осуществляет сложный процесс управления ресурсами компьютера, скрывая от пользователя все его детали.

Важную роль в организации пользовательского интерфейса играет способ представления информации на мониторе. В зависимости от этого различают символьный и графический интерфейсы. Символьный интерфейс обычно реализуется посредством командного языка - набора инструкций (команд), осуществляющих управление системой. Графический интерфейс включает в себя набор наглядных графических средств, позволяющих в интуитивно понятной форме воспроизводить состояние тех или иных объектов (аппаратных компонентов ЭВМ, выполняемых программ) и осуществлять операции по управлению ими.

Для реализации программного интерфейса в ОС включается набор функций, осуществляющих выполнение рутинных операций по управлению ПУ, и обеспечивается доступ к этим функциям со стороны прикладных программ.

4. Основные виды ресурсов

Виды ресурсов

- Процессор (процессорное время)
- Оперативная память
- Внешняя память
- Программные модули
- Информационные ресурсы

4.1. Процессор

Одним из важнейших ресурсов является сам **процессор**, точнее **процессорное время – время на выполнение процессором некоторой задачи. Измеряется в тактах или секундах.**

Например, график зависимости процессорного времени на выполнение расчетной задачи в зависимости от количества сложности прикладной задачи (количество узлов) и типа процессора.

4.2. Вторым важным ресурсом является **оперативная память**. В оперативной памяти может располагаться одновременно несколько процессов (точнее фрагментов, участвующих в вычислении), а может вся оперативная память предоставляться процессам попеременно.

В конкретный момент времени процессор при выполнении вычислений обращается к очень небольшому числу ячеек оперативной памяти, поэтому память желательно разделить для возможно большего числа параллельно исполняемых процессов.

Проблема разделения оперативной памяти между параллельно выполняемыми процессами является наиболее актуальной.

4.3. **Внешняя память**, например магнитный диск, является двумя видами ресурсов:

- собственно память;
- доступ к ней.

Каждый из этих ресурсов может предоставляться независимо друг от друга, но для работы с внешней памятью необходимы оба вида ресурсов:

- собственно память используется одновременно;
- доступ к внешней памяти попеременный.

Если обращение к внешнему устройству использует механизм прямого доступа, то такие устройства разделяются параллельно. Если устройство работает с последовательным доступом, то оно не относится к разделяемым ресурсам, например, принтер или накопитель на магнитной ленте.

4.4. Программные модули

Важным видом ресурсов являются **программные модули -** представляют собой функционально законченный фрагмент программы, оформленный в виде отдельного файла с

исходным кодом, предназначенный для использования в других программах. Системные программные модули рассматриваются как ресурсы, которые могут быть разделены между параллельно выполняемыми процессами.

Программные модули могут использоваться:

- однократно;
- многократно.

Однократно исполняемые модули, как правило, могут быть выполнены только один раз, поскольку в процессе своего исполнения они могут:

- повредить часть кода;
- повредить исходные данные, от которых зависит ход вычислений.

Однократно исполняемые программные модули вообще не распределяются как ресурс системы, они, как правило, используются только на этапе загрузки системы.
Повторно используемые программные модули делятся на следующие виды:

- непривилегированные;
- привилегированные;
- реентерабельные.

Привилегированные программные модули работают в привилегированном режиме, при отключенной системе прерываний, т.е. никакие внешние события не могут нарушить естественный порядок вычислений. Привилегированный программный модуль всегда выполняется до конца и представляет собой попеременно разделяемый ресурс. Структура привилегированного программного модуля включает следующие секции:

- отключение прерываний;
- собственно тело программного модуля;
- включение прерываний.

Непривилегированные программные модули – это обычные программные модули, которые могут быть прерваны во время своей работы. В общем случае их нельзя считать разделяемыми, потому что если его прервать в рамках одного процесса и запустить еще раз в рамках другого процесса, то промежуточные результаты для первого процесса могут быть потеряны.

Реентерабельные программные модули допускают повторное многократное прерывание своего исполнения и повторный их запуск при обращении из других задач, т.е. реентерабельные программные модули должны сохранять промежуточные значения для прерываемых вычислений и их восстановление, когда вычислительный процесс возобновляется с прерванной точки.

Это можно реализовать двумя способами:

- *с помощью статических методов выделения памяти под сохраняемые значения;*
- *с помощью динамических методов выделения памяти под сохраняемые значения. Этот метод используется чаще.*

-

Реентерабельный программный модуль **делится на следующие секции**:

- привилегированный модуль, заказывающий в системной области памяти блок ячеек для хранения текущих (промежуточных) данных;
- основное тело реентерабельного модуля, которое и может быть прервано. Работает в непривилегированном режиме;
- привилегированный модуль, освобождающий в системной области памяти блок памяти, использованный для хранения промежуточных результатов.

*При помещении всех промежуточных данных в системную область, на вершину **стека** помещается указатель на начало области данных и ее объем. Во время исполнения центральной секции реентерабельного программного модуля возможно ее прерывание. Если прерывание не возникло, то в последней секции производится освобождение использованного блока системной области памяти. Если во время исполнения центральной части произошло прерывание и другой вычислительный процесс обращается к тому же реентерабельному модулю, то для этого нового процесса выделяется новый блок памяти в системной области и на вершину стека записывается новый указатель. Повторное вхождение возможно, пока не израсходуется область системной памяти, выделенной специально для реентерабельной обработки.*

При статическом способе выделения памяти резервируется область памяти для фиксированного числа вычислительных процессов, в которых будут располагаться переменные реентерабельных программных модулей, для каждого процесса своя область памяти. К таким процессам относятся драйверы ввода/вывода.

*Кроме реентерабельных программных модулей еще имеются **повторно входимые модули**. Повторно входимые программные модули допускают многократное параллельное исполнение, но их нельзя прерывать. Повторно входимые программные модули состоят из привилегированных секций и повторное обращение к ним возможно только при завершении работы какой-либо секции. После выполнения какой-либо секции управление передается супервизору, который определит, какой процесс будет использовать этот модуль и с какой точки. В повторно входимых программных модулях определены все допустимые (возможные) точки входа. Повторно входимые модули встречают гораздо чаще, чем реентерабельные.*

4.5. Информационные ресурсы

К ресурсам относятся также информационные ресурсы, т.е. данные. Информационные ресурсы включают в себя:

- переменные, находящиеся в оперативной памяти;
- файлы.

Если процессы используют данные только для чтения, то такие нформационные ресурсы можно разделять.

Если процессы могут изменять данные, то работы с такими данными должна быть организована специальным образом, в т.ч. с учетом вопросов синхронизации.

5. Функциональные компоненты операционной системы

Функции операционной системы автономного компьютера обычно группируются либо в соответствии с типами локальных ресурсов, которыми управляет ОС, либо в соответствии со специфическими задачами, применимыми ко всем ресурсам. Иногда такие группы функций называют подсистемами.

Наиболее **важными подсистемами управления ресурсами** являются подсистемы управления процессами, памятью, файлами и внешними устройствами, а подсистемами, общими для всех ресурсов, являются подсистемы пользовательского интерфейса, защиты данных и администрирования.

- Управление процессами
- Управление памятью
- Управление файлами и внешними устройствами
- Защита данных и администрирование
- Интерфейс прикладного программирования
- Пользовательский интерфейс

5.1.Управление процессами

Важнейшей частью операционной системы, непосредственно влияющей на функционирование вычислительной машины, является подсистема управления процессами.

Для каждого вновь создаваемого процесса ОС генерирует системные информационные структуры, которые содержат данные о потребностях процесса в ресурсах вычислительной системы, а также о фактически выделенных ему ресурсах. Таким образом, **процесс можно также определить как некоторую заявку на потребление системных ресурсов.**

Чтобы процесс мог быть выполнен, операционная система должна назначить ему область оперативной памяти, в которой будут размещены коды и данные процесса, а также предоставить ему необходимое количество процессорного времени. Кроме того, процессу может понадобиться доступ к таким ресурсам, как файлы и устройства ввода-вывода.

В информационные структуры процесса часто включаются вспомогательные данные, характеризующие историю пребывания процесса в системе (например, какую долю времени процесс потратил на операции ввода-вывода, а какую на вычисления), его текущее состояние (активное или заблокированное), степень привилегированности процесса (значение приоритета). Данные такого рода могут учитываться операционной системой при принятии решения о предоставлении ресурсов процессу.

В мультипрограммной операционной системе одновременно может существовать несколько процессов. Часть процессов порождается по инициативе пользователей и их приложений, такие процессы обычно называют пользовательскими. Другие процессы, называемые системными, инициализируются самой операционной системой для выполнения своих функций.

Поскольку процессы часто одновременно претендуют на одни и те же ресурсы, то в обязанности ОС входит поддержание очередей заявок процессов на ресурсы, например очереди к процессору, к принтеру, к последовательному порту.

Важной задачей операционной системы является **защита ресурсов**, выделенных данному процессу, от остальных процессов. Одним из наиболее тщательно защищаемых ресурсов

процесса являются области оперативной памяти, в которой хранятся коды и данные процесса. Совокупность всех областей оперативной памяти, выделенных операционной системой процессу, называется его адресным пространством. **Говорят, что каждый процесс работает в своем адресном пространстве, имея в виду защиту адресных пространств, осуществляемую ОС.** Защищаются и другие типы ресурсов, такие как файлы, внешние устройства и т. д. Операционная система может не только защищать ресурсы, выделенные одному процессу, но и организовывать их совместное использование, например разрешать доступ к некоторой области памяти нескольким процессам.

На протяжении периода существования процесса его выполнение может быть многократно прервано и продолжено. Для того чтобы возобновить выполнение процесса, необходимо восстановить состояние его операционной среды. Состояние операционной среды идентифицируется состоянием регистров и программного счетчика, режимом работы процессора, указателями на открытые файлы, информацией о незавершенных операциях ввода-вывода, кодами ошибок выполняемых данным процессом системных вызовов и т. д. Эта информация называется контекстом прогресса. Говорят, что при смене процесса происходит переключение контекстов.

Операционная система берет на себя также **функции синхронизации процессов**, позволяющие процессу приостанавливать свое выполнение до наступления какого-либо события в системе, например завершения операции ввода-вывода, осуществляемой по его запросу операционной системой.

В операционной системе нет однозначного соответствия между процессами и программами. Один и тот же программный файл может породить несколько параллельно выполняемых процессов, а процесс может в ходе своего выполнения сменить программный файл и начать выполнять другую программу.

Для реализации сложных программных комплексов полезно бывает организовать их работу в виде **нескольких параллельных процессов**, которые периодически взаимодействуют друг с другом и обмениваются некоторыми данными. Так как операционная система защищает ресурсы процессов и не позволяет одному процессу писать или читать из памяти другого процесса, то для оперативного взаимодействия процессов ОС должна предоставлять особые средства, которые называют средствами межпроцессного взаимодействия.

Таким образом, подсистема управления процессами:

- планирует выполнение процессов, то есть распределяет процессорное время между несколькими одновременно существующими в системе процессами,
- занимается созданием и уничтожением процессов,
- обеспечивает процессы необходимыми системными ресурсами,
- поддерживает синхронизацию процессов
- обеспечивает взаимодействие между процессами.

5.2.Управление памятью

Память является для процесса таким же важным ресурсом, как и процессор, так как процесс может выполняться процессором только в том случае, если его коды и данные (не обязательно все) находятся в оперативной памяти.

Управление памятью включает распределение имеющейся физической памяти между всеми существующими в системе в данный момент процессами, загрузку кодов и данных процессов в отведенные им области памяти, настройку адресно-зависимых частей кодов

процесса на физические адреса выделенной области, а также защиту областей памяти каждого процесса.

Существует большое разнообразие **алгоритмов распределения памяти**. Они могут отличаться, например, количеством выделяемых процессу областей памяти (в одних случаях память выделяется процессу в виде одной непрерывной области, а в других — в виде нескольких несмежных областей), степенью свободы границы областей (она может быть жестко зафиксирована на все время существования процесса или же динамически перемещаться при выделении процессу дополнительных объемов памяти). В некоторых системах распределение памяти выполняется страницами фиксированного размера, а в других — сегментами переменной длины.

Одним из наиболее популярных способов управления памятью в современных операционных системах является так называемая **виртуальная память**. Наличие в ОС механизма виртуальной памяти позволяет программисту писать программу так, как будто в его распоряжении имеется однородная оперативная память большого объема, часто существенно превышающего объем имеющейся физической памяти. В действительности все данные, используемые программой, хранятся на диске и при необходимости **частями (сегментами или страницами) отображаются в физическую память**. При перемещении кодов и данных между оперативной памятью и диском подсистема виртуальной памяти выполняет трансляцию виртуальных адресов, полученных в результате компиляции и компоновки программы, в физические адреса ячеек оперативной памяти. Очень важно, что все операции по перемещению кодов и данных между оперативной памятью и дисками, а также трансляция адресов выполняются ОС прозрачно для программиста.

Защита памяти — это избирательная способность предохранять выполняемую задачу от записи или чтения памяти, назначенной другой задаче. Правильно написанные программы не пытаются обращаться к памяти, назначенной другим. Однако реальные программы часто содержат ошибки, в результате которых такие попытки иногда предпринимаются. Средства защиты памяти, реализованные в операционной системе, должны пресекать несанкционированный доступ процессов к чужим областям памяти.

Таким образом, функциями ОС по управлению памятью являются:

- отслеживание свободной и занятой памяти;
- выделение памяти процессам и освобождение памяти при завершении процессов;
- защита памяти;
- вытеснение процессов из оперативной памяти на диск, когда размеры основной памяти недостаточны для размещения в ней всех процессов, и возвращение их в оперативную память, когда в ней освобождается место
- настройка адресов программы на конкретную область физической памяти.

5.3.Управление файлами и внешними устройствами

Способность ОС к «экранированию» сложностей реальной аппаратуры очень ярко проявляется в одной из основных подсистем ОС — файловой системе. Операционная система виртуализирует отдельный набор данных, хранящихся на внешнем накопителе, в виде файла — простой неструктурированной последовательности байтов, имеющей символьное имя. Для удобства работы с данными файлы группируются в каталоги, которые, в свою очередь, образуют группы — каталоги более высокого уровня. Пользователь может с помощью ОС выполнять над файлами и каталогами такие действия, как поиск по имени, удаление, вывод содержимого на внешнее устройство (например, на дисплей), изменение и сохранение содержимого.

Чтобы представить **большое количество наборов данных, разбросанных случайным образом по цилиндрам и поверхностям дисков различных типов, в виде хорошо всем знакомой и удобной иерархической структуры файлов и каталогов, операционная система должна решить множество задач**. Файловая система ОС выполняет преобразование символьных имен файлов, с которыми работает пользователь или прикладной программист, в физические адреса данных на диске, организует совместный доступ к файлам, защищает их от несанкционированного доступа.

При выполнении своих функций файловая система тесно взаимодействует с подсистемой управления внешними устройствами, которая по запросам файловой системы осуществляет передачу данных между дисками и оперативной памятью.

Подсистема управления внешними устройствами, называемая также подсистемой ввода-вывода, исполняет роль интерфейса ко всем устройствам, подключенным к компьютеру. Спектр этих устройств очень обширен. Номенклатура выпускаемых накопителей на жестких, гибких и оптических дисках, принтеров, сканеров, мониторов, плоттеров, модемов, сетевых адаптеров и более специальных устройств ввода-вывода, таких как, например, аналого-цифровые преобразователи, может насчитывать сотни моделей. Эти модели могут существенно отличаться набором и последовательностью команд, с помощью которых осуществляется обмен информацией с процессором и памятью компьютера, скоростью работы, кодировкой передаваемых данных, возможностью совместного использования и множеством других деталей.

Программа, управляющая конкретной моделью внешнего устройства и учитывающая все его особенности, обычно называется **драйвером этого устройства** (от английского drive — управлять, вести). Драйвер может управлять единственной моделью устройства или же группой устройств определенного типа. Для пользователя очень важно, чтобы операционная система включала как можно больше разнообразных драйверов, так как это гарантирует возможность подключения к компьютеру большого числа внешних устройств различных производителей. От наличия подходящих драйверов во многом зависит успех операционной системы на рынке (например, отсутствие многих необходимых драйверов внешних устройств было одной из причин низкой популярности OS/2).

Созданием драйверов устройств занимаются как разработчики конкретной ОС, так и специалисты компаний, выпускающих внешние устройства. Операционная система должна поддерживать хорошо определенный интерфейс между драйверами и остальной частью ОС, чтобы разработчики из компаний-производителей устройств ввода-вывода могли поставлять вместе со своими устройствами драйверы для данной операционной системы.

Прикладные программисты могут пользоваться интерфейсом драйверов при разработке своих программ, но это не очень удобно — такой интерфейс обычно представляет собой низкоуровневые операции, обремененные большим количеством деталей.

Поддержание высокоуровневого унифицированного интерфейса прикладного программирования к разнородным устройствам ввода-вывода является одной из наиболее важных задач ОС. Со времени появления ОС UNIX такой унифицированный интерфейс в большинстве операционных систем строится на основе концепции файлового доступа. Эта концепция заключается в том, что обмен с любым внешним устройством выглядит как обмен с файлом, имеющим имя и представляющим собой неструктурированную последовательность байтов. В качестве файла может выступать как реальный файл на диске, так и алфавитно-цифровой терминал, печатающее устройство или сетевой адаптер. **Здесь мы опять имеем дело со свойством операционной системы подменять реальную аппаратуру удобными для пользователя и программиста абстракциями.**

5.4.Защита данных и администрирование

Безопасность данных вычислительной системы обеспечивается средствами отказоустойчивости ОС, направленными на защиту от сбоев и отказов аппаратуры и ошибок программного обеспечения, а также средствами защиты от несанкционированного доступа. В последнем случае ОС защищает данные от ошибочного или злонамеренного поведения пользователей системы.

Первым рубежом обороны при защите данных от несанкционированного доступа является **процедура логического входа**. Операционная система должна убедиться, что в систему пытается войти пользователь, вход которого разрешен администратором. Функции защиты ОС вообще очень тесно связаны с функциями администрирования, так как именно администратор определяет права пользователей при их обращении к разным ресурсам системы — файлам, каталогам, принтерам, сканерам и т. п. Кроме того, администратор ограничивает возможности пользователей в выполнении тех или иных системных действий. Например, пользователю может быть запрещено выполнять процедуру завершения работы ОС, устанавливать системное время, завершать чужие процессы, создавать учетные записи пользователей, изменять права доступа к некоторым каталогам и файлам. Администратор может также урезать возможности пользовательского интерфейса, убрав, например, некоторые пункты из меню операционной системы, выводимого на дисплей пользователя.

Важным средством защиты данных являются **функции аудита ОС, заключающиеся в фиксации всех событий, от которых зависит безопасность системы**. Например, попытки удачного и неудачного логического входа в систему, операции доступа к некоторым каталогам и файлам, использование принтеров и т. п. Список событий, которые необходимо отслеживать, определяет администратор ОС.

Поддержка отказоустойчивости реализуется операционной системой, как правило, на основе резервирования. Чаще всего в функции ОС входит поддержание нескольких копий данных на разных дисках или разных дисковых накопителях. Резервируются также принтеры и другие устройства ввода-вывода. При отказе одного из избыточных устройств операционная система должна быстро и прозрачным для пользователя образом произвести реконфигурацию системы и продолжить работу с резервным устройством. Особым случаем обеспечения отказоустойчивости является использование нескольких процессоров, то есть мультипроцессирование, когда система продолжает работу при отказе одного из процессоров, хотя и с меньшей производительностью. (Необходимо отметить, что многие ОС использует мультипроцессорную конфигурацию компьютера только для ускорения работы и при отказе одного из процессоров прекращают работу.)

Поддержка отказоустойчивости также входит в обязанности системного администратора. В состав ОС обычно входят утилиты, позволяющие администратору выполнять регулярные операции резервного копирования для обеспечения быстрого восстановления важных данных.

5.5.Интерфейс прикладного программирования

Прикладные программисты используют в своих приложениях обращения к ОС, когда для выполнения тех или иных действий им требуется особый статус, которым обладает только операционная система. Например, в большинстве современных ОС все действия, связанные с управлением аппаратными средствами компьютера, может выполнять только ОС. Помимо этих функций прикладной программист может воспользоваться набором сервисных функций ОС, которые упрощают написание приложений. Функции такого типа реализуют универсальные действия, часто требующиеся в различных приложениях, такие, например, как обработка текстовых строк. Эти функции могли бы быть выполнены и самим приложением,

однако гораздо проще использовать уже готовые, отлаженные процедуры, включенные в состав операционной системы. В то же время даже при наличии в ОС соответствующей функции программист может реализовать ее самостоятельно в рамках приложения, если предложенный операционной системой вариант его не вполне устраивает.

Возможности операционной системы доступны прикладному программисту в виде набора функций, называющегося интерфейсом прикладного программирования (Application Programming Interface, API). От конечного пользователя эти функции скрыты за оболочкой алфавитно-цифрового или графического пользовательского интерфейса.

Для разработчиков приложений все особенности конкретной операционной системы представлены особенностями ее API. Поэтому **операционные системы с различной внутренней организацией, но с одинаковым набором функций API кажутся им одной и той же ОС, что упрощает стандартизацию операционных систем и обеспечивает переносимость приложений между внутренне различными ОС, соответствующими определенному стандарту на API**. Например, следование общим стандартам API UNIX, одним из которых является стандарт Posix, позволяет говорить о некоторой обобщенной операционной системе UNIX, хотя многочисленные версии этой ОС от разных производителей иногда существенно отличаются внутренней организацией.

Приложения выполняют обращения к функциям API с помощью системных вызовов. Способ, которым приложение получает услуги операционной системы, очень похож на вызов подпрограмм. Информация, нужная ОС и состоящая обычно из идентификатора команды и данных, помещается в определенное место памяти, в регистры и/или стек. Затем управление передается операционной системе, которая выполняет требуемую функцию и возвращает результаты через память, регистры или стеки. Если операция проведена неуспешно, то результат включает индикацию ошибки.

Способ реализации системных вызовов зависит от структурной организации ОС, которая, в свою очередь, тесно связана с особенностями аппаратной платформы. Кроме того, он зависит от языка программирования. При использовании ассемблера программист устанавливает значения регистров и/или областей памяти, а затем выполняет специальную инструкцию вызова сервиса или программного прерывания для обращения к некоторой функции ОС. При использовании языков высокого уровня функции ОС вызываются тем же способом, что и написанные пользователем подпрограммы, требуя задания определенных аргументов в определенном порядке.

5.6.Пользовательский интерфейс

Операционная система должна обеспечивать удобный интерфейс не только для прикладных программ, но и для человека, работающего за терминалом. Этот человек может быть конечным пользователем, администратором ОС или программистом.

В ранних операционных системах пакетного режима функции пользовательского интерфейса были сведены к минимуму и не требовали наличия терминала. Команды языка управления заданиями набивались на перфокарты, а результаты выводились на печатающее устройство.

Современные ОС поддерживают развитые функции пользовательского интерфейса для интерактивной работы за терминалами двух типов: алфавитно-цифровыми и графическими.

При работе **за алфавитно-цифровым терминалом пользователь** имеет в своем распоряжении систему команд, мощность который отражает функциональные возможности данной ОС. Обычно командный язык ОС позволяет запускать и останавливать приложения,

выполнять различные операции с файлами и каталогами, получать информацию о состоянии ОС (количество работающих процессов, объем свободного пространства на дисках и т. п.), администрировать систему. Команды могут вводиться не только в интерактивном режиме с терминала, но и считываться из так называемого командного файла, содержащего некоторую последовательность команд.

Программный модуль ОС, ответственный за чтение отдельных команд или же последовательности команд из командного файла, иногда называют командным интерпретатором.

Ввод команды может быть упрощен, если операционная система поддерживает графический пользовательский интерфейс. В этом случае пользователь для выполнения нужного действия с помощью мыши выбирает на экране нужный пункт меню или графический символ.

6. ОС как виртуальная (расширенная) машина

Для того чтобы успешно решать свои задачи, современный пользователь или даже прикладной программист может обойтись без досконального знания аппаратного устройства компьютера. Ему не обязательно быть в курсе того, как функционируют различные электронные блоки и электромеханические узлы компьютера. Более того, очень часто пользователь может не знать даже системы команд процессора. **Пользователь-программист привык иметь дело с мощными высокоуровневыми функциями, которые ему предоставляет операционная система**.

Так, например, **при работе с диском программисту, пишущему приложение для работы под управлением ОС, или конечному пользователю ОС достаточно представлять его в виде некоторого набора файлов, каждый из которых имеет имя.** Последовательность действий при работе с файлом заключается в его **открытии, выполнении одной или нескольких операций чтения или записи, а затем в закрытии файла**. Такие частности, **как используемая при записи частотная модуляция или текущее состояние двигателя механизма перемещения магнитных головок чтения/записи, не должны волновать программиста**. Именно операционная система скрывает от программиста большую часть особенностей аппаратуры и предоставляет возможность простой и удобной работы с требуемыми файлами.

Если бы программист работал непосредственно с аппаратурой компьютера, без участия ОС, то для организации чтения блока данных с диска программисту **пришлось бы использовать более десятка команд с указанием множества параметров: номера блока на диске, номера сектора на дорожке и т. п. А после завершения операции обмена с диском он должен был бы предусмотреть в своей программе анализ результата выполненной операции**. Учитывая, что контроллер диска способен распознавать более двадцати различных вариантов завершения операции, можно считать программирование обмена с диском на уровне аппаратуры не самой тривиальной задачей. Не менее обременительной выглядит и работа пользователя, если бы ему для чтения файла с терминала потребовалось задавать числовые адреса дорожек и секторов.

Операционная система избавляет программистов не только от необходимости напрямую работать с аппаратурой дискового накопителя, предоставляя им простой файловый интерфейс, но и **берет на себя все другие рутинные операции, связанные с управлением другими**

аппаратными устройствами компьютера: физической памятью, таймерами, принтерами и т. д.

В результате **реальная машина, способная выполнять только небольшой набор элементарных действий, определяемых ее системой команд, превращается в виртуальную машину, выполняющую широкий набор гораздо более мощных функций**. **Виртуальная машина тоже управляется командами, но это уже команды другого, более высокого уровня: удалить файл с определенным именем, запустить на выполнение некоторую прикладную программу, повысить приоритет задачи, вывести текст из файла на печать.** Таким образом, назначение ОС состоит в предоставлении пользователю/программисту некоторой расширенной виртуальной машины, которую легче программировать и с которой легче работать, чем непосредственно с аппаратурой, составляющей реальный компьютер или реальную сеть.

7. Режим супервизора и пользователя

Режим супервизора — привилегированный режим работы процессора, как правило используемый для выполнения ядра операционной системы.

В данном режиме работы процессора доступны привилегированные операции, как то:

- операции ввода-вывода к периферийным устройствам,
- изменение параметров защиты памяти, настроек виртуальной памяти, системных параметров и прочих параметров конфигурации.

Как правило, в режиме супервизора или вообще не действуют ограничения защиты памяти или же они могут быть произвольным образом изменены, поэтому код, работающий в данном режиме, как правило, имеет полный доступ ко всем системным ресурсам (адресное пространство, регистры конфигурации процессора и т. п.). Во многих типах процессоров **это наиболее привилегированный режим** из всех доступных режимов.

Известно одно исключение из данного правила: у некоторых современных процессоров может присутствовать ещё более привилегированный режим **гипервизора**, как правило, используемый с целью виртуализации, то есть обеспечения параллельной работы сразу нескольких операционных систем на одном процессоре. В этом случае настройки, сделанные из режима гипервизора могут вносить некоторые ограничения на прямой доступ к системным ресурсам и периферии из режима супервизора с целью предоставить гипервизору возможность арбитража и разграничения доступа к системным ресурсам и периферии незаметно для работающих параллельно операционных систем.

В нормальном состоянии ЦП находится в **режиме пользователя**. Переход из этого режима в режим супервизора возможен только при нарушении нормальной работы специальной инструкцией или внешним событием. Такая ситуация называется **исключением**, а сама **процедура перехода - обработкой исключения**.

Исключение - это любое нарушение нормальной работы МП.

Исключения могут вызываться:

- внутренними (адресные ошибки, неправильные результаты обработки и выполнения инструкций, трассировка) причинами

- внешними (сигнал сброса, ошибка магистрали, прерывания) причинами.

Исключения разделяются по приоритетам.

Их обработка осуществляется подпрограммами, адреса которых вычисляет ЦП с использованием номера вектора исключения, генерируемого самим ЦП или передаваемым ему в цикле подтверждения прерывания.

Прерывания являются частным случаем исключений.

Режим работы ЦП определяется специальным битом в регистре состояния (бит S), переключение которого возможно только в режиме супервизора.

Переход из режима супервизора в режим пользователя происходит только по инструкции, воздействующей на бит S регистра состояния.

В режимах пользователя и супервизора различаются:

- адресные пространства, в которых работает ЦП;
- программные модели;
- набор допустимых инструкций;
- активные стеки.

8. Требования к современным операционным системам

Главным требованием, предъявляемым к операционной системе, является выполнение ею основных **функций эффективного управления ресурсами** и обеспечение удобного интерфейса для пользователя и прикладных программ. Современная ОС, как правило, должна поддерживать мультипрограммную обработку, виртуальную память, свопинг, многооконный графический интерфейс пользователя, а также выполнять многие другие необходимые функции и услуги.

Кроме этих требований функциональной полноты к операционным системам предъявляются не менее важные эксплуатационные требования, которые перечислены ниже.

- **Расширяемость**. В то время как аппаратная часть компьютера устаревает за несколько лет, полезная жизнь операционных систем может измеряться десятилетиями. Примером может служить ОС UNIX. Поэтому операционные системы всегда изменяются со временем эволюционно, и эти изменения более значимы, чем изменения аппаратных средств. Изменения ОС обычно заключаются в приобретении ею новых свойств, например поддержке новых типов внешних устройств или новых сетевых технологий. Если код ОС написан таким образом, что дополнения и изменения могут вноситься без нарушения целостности системы, то такую ОС называют расширяемой. Расширяемость достигается за счет модульной структуры ОС, при которой программы строятся из набора отдельных модулей, взаимодействующих только через функциональный интерфейс.
- **Переносимость**. В идеале код ОС должен легко переноситься с процессора одного типа на процессор другого типа и с аппаратной платформы (которые различаются не только типом процессора, но и способом организации всей аппаратуры компьютера) одного типа на аппаратную платформу другого типа. Переносимые ОС имеют несколько

вариантов реализации для разных платформ, такое свойство ОС называют также многоплатформенностъю.

- **Совместимость**. Существует несколько «долгоживущих» популярных операционных систем (разновидности UNIX, MS-DOS, Windows 3.x, Windows NT, OS/2), для которых наработана широкая номенклатура приложений. Некоторые из них пользуются широкой популярностью. Поэтому для пользователя, переходящего по тем или иным причинам с одной ОС на другую, очень привлекательна возможность запуска в новой операционной системе привычного приложения. Если ОС имеет средства для выполнения прикладных программ, написанных для других операционных систем, то про нее говорят, что она обладает совместимостью с этими ОС. Следует различать совместимость на уровне двоичных кодов и совместимость на уровне исходных текстов. Понятие совместимости включает также поддержку пользовательских интерфейсов других ОС.
- **Надежность и отказоустойчивость**. Система должна быть защищена как от внутренних, так и от внешних ошибок, сбоев и отказов. Ее действия должны быть всегда предсказуемыми, а приложения не должны иметь возможности наносить вред ОС. Надежность и отказоустойчивость ОС прежде всего определяются архитектурными решениями, положенными в ее основу, а также качеством ее реализации (отлаженностью кода). Кроме того, важно, включает ли ОС программную поддержку аппаратных средств обеспечения отказоустойчивости, таких, например, как дисковые массивы или источники бесперебойного питания.
- **Безопасность**. Современная ОС должна защищать данные и другие ресурсы вычислительной системы от несанкционированного доступа. Чтобы ОС обладала свойством безопасности, она должна как минимум иметь в своем составе средства аутентификации — определения легальности пользователей, авторизации — предоставления легальным пользователям дифференцированных прав доступа к ресурсам, аудита — фиксации всех «подозрительных» для безопасности системы событий. Свойство безопасности особенно важно для сетевых ОС. В таких ОС к задаче контроля доступа добавляется задача защиты данных, передаваемых по сети.
- **Производительность.** Операционная система должна обладать настолько хорошим быстродействием и временем реакции, насколько это позволяет аппаратная платформа. На производительность ОС влияет много факторов, среди которых основными являются архитектура ОС, многообразие функций, качество программирования кода, возможность исполнения ОС на высокопроизводительной (многопроцессорной) платформе.

9. Выводы

- ОС — это комплекс взаимосвязанных программ, предназначенный для повышения эффективности аппаратуры компьютера путем рационального управления его ресурсами, а также для обеспечения удобств пользователю путем предоставления ему расширенной виртуальной машины.
- К числу основных ресурсов, управление которыми осуществляет ОС, относятся процессоры, основная память, таймеры, наборы данных, диски, накопители на магнитных лентах, принтеры, сетевые устройства и некоторые другие. Ресурсы распределяются между процессами. Для решения задач управления ресурсами разные ОС используют различные алгоритмы, особенности которых в конечном счете и определяют облик ОС.
- Наиболее важными подсистемами ОС являются подсистемы управления процессами, памятью, файлами и внешними устройствами, а также подсистемы пользовательского интерфейса, защиты данных и администрирования.

- Прикладному программисту возможности ОС доступны в виде набора функций, составляющих интерфейс прикладного программирования (API).
- В число требований, предъявляемых сегодня к сетевым ОС, входят: функциональная полнота и эффективность управления ресурсами, модульность и расширяемость, переносимость и многоплатформенность, совместимость на уровне приложений и пользовательских интерфейсов, надежность и отказоустойчивость, безопасность и производительность.