**1 аттестация (16.03.2022)**

|  |  |
| --- | --- |
| **1** | **Эволюция ОС и ЭВМ.** |
| **2** | **Определение ОС в качестве различных компонентов.** |
| **3** | **Способы построения ядра ОС.** |
| **4** | **Классификация ОС.** |
| **5** | **Требования, предъявляемые к ОС.** |
| **6** | **Основные принципы построения ОС.** |
| **7** | **Процессы. Классификация процессов.** |
| **8** | **Диаграмма состояний процесса.** |
| **9** | **Планирование процессов. Виды планирования.** |
| **10** | **Алгоритмы планирования пакетной обработки данных.** |
| **11** | **Алгоритмы планирования в интерактивной системе.** |

1. **Эволюция ОС и ЭВМ.**

***I поколение (1945 – 1955, электронные лампы).***

В первом поколении ОС не существовало. Программы формировались с помощью коммутационных панелей. Всё программирование велось исключительно на машинном языке. Одна группа людей выполняла все операции над машиной.

***II поколение (1955 – 1965, транзисторы и системы пакетной обработки).***

***IBM 1401, IBM 7094***

Применение транзисторов вместо часто перегоравших электронных ламп привело к **повышению надежности** компьютеров. Теперь машины могут непрерывно работать достаточно долго, чтобы на них можно было возложить выполнение практически важных задач. Снижается потребление вычислительными машинами электроэнергии, совершенствуются системы охлаждения. Размеры компьютеров уменьшились. Снизилась стоимость эксплуатации и обслуживания вычислительной техники.

Появляются первые настоящие **компиляторы**, **редакторы** **связей**, библиотеки математических и служебных подпрограмм. Упрощается процесс программирования. Пропадает необходимость взваливать на одних и тех же людей весь процесс разработки и использования компьютеров. Именно в этот период происходит разделение персонала на **программистов** и **операторов**, специалистов по эксплуатации и разработчиков вычислительных машин.

Изменяется сам процесс прогона программ. Теперь пользователь приносит программу с входными данными в виде **колоды перфокарт** и указывает необходимые ресурсы. Такая колода получает название **задания**.

***III поколение (1965 – 1980, интегральные схемы, многозадачность).***

В **IBM/360** использовались малые интегральные схемы, что давало преимущество в цене и качестве. Вычислительная техника становится более надежной и дешевой. Растет сложность и количество задач, решаемых компьютерами. Повышается производительность процессоров.

Дальнейшее повышение эффективности использования процессора было достигнуто с помощью мультипрограммирования (**многозадачности**). Идея мультипрограммирования заключается в следующем: **пока одна программа выполняет операцию ввода-вывода, процессор не простаивает**, как это происходило при однопрограммном режиме**, а выполняет другую программу.** Когда операция ввода-вывода заканчивается, процессор возвращается к выполнению первой программы.

***IV поколение (1980 – н.д., БИС, персональные компьютеры).***

Следующий период в эволюции вычислительных систем связан с появлением **больших интегральных схем** (БИС). Компьютер по цене и простоте эксплуатации стал доступен отдельному человеку. Наступила эра персональных компьютеров.

Компьютеры стали использоваться не только специалистами, что потребовало разработки **«дружественного»** программного обеспечения. На рынке операционных систем в эти годы доминировали две системы: MS-DOS и Unix-RISC.

***V поколение (1990 – н.д., мобильные технологии).***

За 30 с небольшим лет мобильные операционные системы эволюционировали от скромной вычислительной среды для работы простых программ-помощников (контакты, записная книжка, калькулятор и др.) до мощного комплекса, обеспечивающего полноценную работу специализированных приложений и развлекательных сервисов. Сегодня мобильные ОС дают возможность снимать, редактировать и просматривать видео в высоком разрешении, играть в 3D-игры, использовать искусственный интеллект и технологии дополненной реальности.

1. **Определение ОС в качестве различных компонентов.**

* **ОС как виртуальная машина**

Предоставляется уровень абстракции между аппаратным обеспечением и прикладным ПО, предоставляя вверх сервисы, которых не существует внизу.

Например, **файловая система** – это уровень абстракций, позволяющий унифицировать работу с разнообразных оборудованием.

* **ОС как менеджер ресурсов**

ОС управляет выделением различных ресурсов процессам (процессорное время, ячейки памяти).

* **ОС как защита пользователей и их программ**

Многопользовательские системы должны иметь функции защиты одного пользователя от другого.

Многозадачные системы должны иметь функции защиты данных одного процесса от другого.

**Степень** **мультипрограммирования** – максимальное количество процессов, которые могут одновременно выполняться в системе.

* **ОС как постоянно функционирующее ядро**

**Операционная система** – это программа, постоянно работающая на компьютере и взаимодействующая со всеми прикладными программами.

1. **Способы построения ядра ОС.**
2. **Монолитное ядро**

Включает все функции ОС с поддержкой всего возможного оборудования. Для того, чтобы подстроить ядро под конкретное оборудование, необходима перекомпиляция.

**Достоинства**: всё всегда в памяти, ядро работает быстро.

**Недостатки**: большой размер, занимает много памяти.

1. **Модульное ядро**

Усовершенствованная структура монолитных ядер. Модули загружаются при необходимости без перегенерации системы.

**Достоинства**: меньший объем.

**Недостатки**: некоторый функционал необходимо подгружать отдельно.

1. **Микроядро**

Большинство составляющих операционной системы являются самостоятельными программами. В этом случае взаимодействие между ними обеспечивает специальный модуль ядра, называемый микроядром. Микроядро работает в привилегированном режиме и обеспечивает взаимодействие между программами,.

**Достоинства**: повышается надежность системы.

**Недостатки**: микроядерная архитектура вносит дополнительные расходы, связанные с передачей сообщений, что существенно влияет на производительность

1. **Классификация ОС.**

* **По многозадачности:**

**- однозадачные** (однопользовательский режим, MS DOS)

**-** **многозадачные** (поддержка многопользовательского режима)

* **По количеству поддерживаемых процессоров:**

- **однопроцессорные**

- **многопроцессорные** (поддерживают режим распределения ресурсов нескольких процессоров для решения той или иной задачи)

* **По критерию эффективности:**

1. **Системы пакетной обработки.**

Задания выполняются друг за другом последовательно, процессы работают, пока не решат свою задачу.

**Критерий эффективности**: количество задач, выполняемых в единицу времени (пропускная способность).

**Цель** – максимальная загруженность.

1. **Системы разделения времени.**

Процессорное время разделяется между множеством пользователей, каждый из которых работает о своего терминала. При этом для пользователей и процессов создаётся иллюзия монопольного владения системой.

**Критерий эффективности:** соответствие ожиданиям пользователя.

1. **Система реального времени.**

Предназначена для обслуживания приложений реального времени, которые обрабатывают данные по мере их поступления, как правило, без задержек в буфере. Система переключается между задачами либо в зависимости от их приоритетов, либо на основе тактовых прерываний.

**Критерий эффективности:** реактивность (гарантированное время реакции системы на то или иное событие).

1. **Требования, предъявляемые к ОС.**
2. **Расширяемость**

Код должен быть написан таким образом, чтобы можно было легко внести дополнения и изменения, если это потребуется, и не нарушить целостность системы.

1. **Переносимость**

Код должен легко переноситься с процессора одного типа на процессор другого типа. Для легкого переноса ОС д.б. соблюдены следующие требования:

* Переносимый язык высокого уровня.

Большинство программ написаны на языке С, т.к. он стандартизирован и С-компиляторы широко доступны. Непереносимый код должен быть тщательно изолирован внутри тех компонентов, где он используется.

* Изоляция процессора

Некоторые низкоуровневые части ОС должны иметь доступ к процессорно-зависимым структурам данных и регистрам.

Код должен содержаться в небольших модулях, которые могут быть заменены аналогичным модулем для других процессоров.

* Изоляция платформы

Должен быть программный уровень, абстрагирующий аппаратуру.

1. **Совместимость**

ОС должна иметь средства для выполнения прикладных программ, написанных для других ОС.

Бывает двоичная и на уровне исходных кодов.

1. **Надёжность и отказоустойчивость**

Система должна быть защищена как от внутренних, так и от внешних сбоев и отказов. В случае ошибки в программе или аппаратуре система должна обнаружить ошибку и попытаться исправить положение или, по крайней мере, постараться свести к минимуму ущерб, нанесенный этой ошибкой пользователям.

1. **Безопасность**

ОС должна защищать пользователей и от воздействия чужих ошибок, и от попыток злонамеренного вмешательства (несанкционированного доступа). С этой целью в ОС как минимум должны быть средства аутентификации – определения легальности пользователей, авторизации – предоставления легальным пользователям установленных им прав доступа к ресурсам, и аудита – фиксации всех потенциально опасных для системы событий.

1. **Производительность**

На показатели эффективности ОС влияет много различных факторов, среди которых основными являются архитектура ОС, многообразие ее функций, качество программного кода, аппаратная платформа (компьютер) и др.

1. **Основные принципы построения ОС.**
2. **Частотный принцип**

Основан на выделении в алгоритмах в обрабатываемых массивах действий данных по частоте использования. Для наиболее частых создается условие быстрого выполнения. Оптимизация кода частых операций.

1. **Принцип наращивания**

ОС должна быть открыта и использована для изучения.

ОС открыта, если описан интерфейс взаимодействия с ней.

1. **Принцип модульности**

Система должна разбиваться на отдельные части по функциональному назначению, которые должны добавляться или удаляться не нарушая системы.

1. **Принцип генерируемости**

ОС должна настраиваться на конкретное программное обеспечение.

1. **Принцип функциональной избыточности**

Любое действие в ОС можно выполнить несколькими путями.

1. **Принцип функциональной избирательности**

Функции, которые используются чаще всего, должны быть под рукой.

1. **По умолчанию**

Любые настраиваемые значения в ОС должны иметь значения по умолчанию.

1. **Принцип перемещаемости**

Приложения должны работать одинаково вне зависимости от их расположения в оперативной памяти.

1. **Принцип защиты**
2. **Принцип независимости программ от внешних устройств**
3. **Процессы. Классификация процессов.**

* **По динамическим признакам:**

**-** последовательные

**-** параллельные

**-** комбинированные

* **По принадлежности к ЦП:**

**-** **внутренние** (если процесс порождается внутри процессора)

**-** **внешние** (если процесс порождается внутри ОС)

* **По результативности:**

**- эквивалентные** (имеют одинаковый результат, но процессы могут реализовываться как на одном, так и на многих **процессорах по одному или разным алгоритмам**, то есть они имеют **разные трассы**)

**-** **тождественные** (имеют одинаковый результат, процессы **реализуются по одной и той же программе**, но имеют **разные трассы**)

**-** **равные** (имеют одинаковый результат, процессы **реализуются по одной программе** и имеют **одинаковые трассы**)

**-** **различные** (во всех остальных случаях)

* **По связности:**

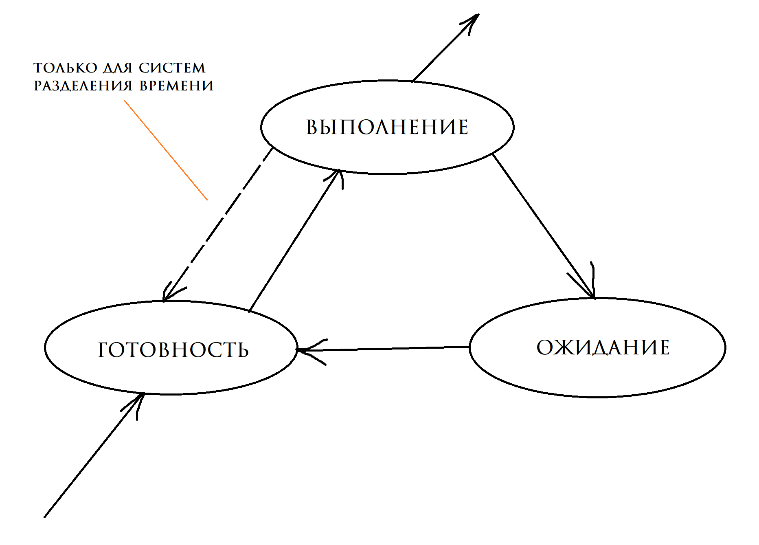
**- взаимосвязанные** (это процессы, между которыми существует хотя бы один тип связи)

***Типы связи:***

* функциональные
* связи по управлению
* информационные связи
* пространственно-временные

**-** **изолированные** (нет связей между процессами)

1. **Диаграмма состояний процесса.**

****

Активный процесс может быть в одном из следующих состояний:

* Выполнения –все затребованные процессом ресурсы выделены. В этом состоянии в каждый момент времени может находиться только один процесс, если речь идёт об однопроцессорной системе;
* Готовности к выполнению – ресурсы могут быть предоставлены, тогда процесс перейдёт в состояние выполнения;
* Ожидания– затребованные ресурсы не могут быть предоставлены, или не завершена операция ввода/вывода.

Процесс **из состояния бездействия** может перейти **в состояние готовности** в сле­дующих случаях:

♦ по команде оператора (пользователя)

♦ при выборе из очереди планировщиком (характерно для операционных сис­тем, работающих в пакетном режиме);

♦ по вызову из другой задачи

♦ по прерыванию от внешнего устройства

♦ при наступлении запланированного времени запуска программы

**Из состояния выполнения** процесс может выйти по одной из следующих при­чин:

♦ процесс завершается

♦ процесс переводится в состояние готов­ности к исполнению в связи с появлением более приоритетной задачи или в связи с окончанием выделенного ему кванта времени;

♦ процесс блокируется (переводится в состояние ожидания) либо вследствие запроса операции ввода/вывода, либо в силу невозможности предо­ставить ему ресурс, запрошенный в настоящий момент.

1. **Планирование процессов. Виды планирования.**

Планирование процессов - это деятельность диспетчера процессов, которая обрабатывает удаление запущенного процесса из ЦП и выбор другого процесса на основе конкретной стратегии.

1. **Долгосрочное планирование**

Решается, какие новые задачи будут добавлены.

1. **Среднесрочное планирование**

Решается, нужно ли временно выгружать программу во вторичную память.

1. **Краткосрочное планирование**

Принятие решения о выборе процесса из очереди готовых процессов, которые поступят на выполнение (диспетчеризация). Решается, какому потоку дать следующий квант процессорного времени и какой длины.

1. **Алгоритмы планирования пакетной обработки данных.**
2. **FCFS (First Come – First Served)**

**Достоинства:** простота реализации.

**Недостатки:** время ожидания и время выполнения зависит от очереди поступления процессов.

1. **SJN (Shortest Job Next) –** самый короткий поступает на обработку.

**Не вытесняющий алгоритм.**

**Достоинства:** практически оптимальный алгоритм.

**Недостатки:** сложность реализации; невозможность предугадать, сколько понадобится процессу времени.

1. **SRN (Shortest Remain Next) или SRTN**

**Вытесняющий алгоритм, модификация SJN.**

Решение о том, какой процесс будет выполняться следующим принимается в следующих случаях:

**-** при освобождении процессорного устройства (завершение либо ожидание)

- при поступлении нового процесса в систему

**Недостаток**: более длительные процессы будут иметь даже большее среднее время ожидания и большую дисперсию времени ожидания

1. **Алгоритмы планирования в интерактивной системе.**

**RR (Round Robin) или карусель**

В циклическом планировании каждая готовая задача выполняется по очереди, только в циклической очереди в течение ограниченного промежутка времени (квант времени).

Чем **меньше квант** времени, тем **меньше среднее полное время выполнения**, тем больше накладные расходы на переключение контекста.

При больших квантах времени RR вырождается в FCFS.