Первый период 1945–1955 г Ламповые машины. Операционных систем нет

Первые шаги в области разработки электронных вычислительных машин были предприняты в конце ВМВ. В середине 40х были созданы первые ламповые вычислительные устройства и появился принцип программы, хранящейся в памяти в то время одна и та же группа людей участвовала и в проектировании, и в эксплуатации, и в программировании вычислительной машины. Это была скорее научно-исследовательская работа в области вычислительной техники, а не регулярное использование компьютеров в качестве инструмента решения каких-либо практических задач из других прикладных областей. Программирование осуществлялось исключительно на машинном языке. Об операционных системах не было и речи, все задачи организации вычислительного процесса решались вручную каждым программистом с пульта управления. За пультом мог находиться только один пользователь. Программа загружалась в память машины в лучшем случае с колоды перфокарт, а обычно с помощью панели переключателей.

Вычислительная система выполняла одновременно только одну операцию Отладка программ велась с пульта управления с помощью изучения состояния памяти и регистров машины. В конце этого периода появляется первое системное программное обеспечение в 1951–1952 гг возникают прообразы первых компиляторов с символических языков, а в 1954 г разрабатывает Ассемблер для IBM-701.

Существенная часть времени уходила на подготовку запуска программы, а сами программы выполнялись строго последовательно. Такой режим работы называется последовательной обработкой данных. В целом первый период характеризуется крайне высокой стоимостью вычислительных систем, их малым количеством и низкой эффективностью использования.

Второй период (1955 г.–начало 60-х). Компьютеры на основе транзисторов. Пакетные операционные системы

С середины 50-х годов начался следующий период в эволюции вычислительной техники, связанный с появлением новой технической базы – полупроводниковых элементов. 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. Началось использование ЭВМ коммерческими фирмами. Одновременно наблюдается бурное развитие алгоритмических языков 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. Оператор загружает задание в память машины и запускает его на исполнение. Полученные выходные данные печатаются на принтере, и пользователь получает их обратно через некоторое (довольно продолжительное) время.

Смена запрошенных ресурсов вызывает приостановку выполнения программ, в результате процессор часто простаивает. Для повышения эффективности использования компьютера задания с похожими ресурсами начинают собирать вместе, создавая пакет заданий.

Появляются первые системы пакетной обработки, которые просто автоматизируют запуск одной программы из пакета за другой и тем самым увеличивают коэффициент загрузки процессора. При реализации систем пакетной обработки был разработан формализованный язык управления заданиями, с помощью которого программист сообщал системе и оператору, какую работу он хочет выполнить на вычислительной машине. Системы пакетной обработки стали прообразом современных операционных систем, они были первыми системными программами, предназначенными для управления вычислительным процессом.

Третий период (начало 60-х – 1980 г.). Компьютеры на основе интегральных микросхем. Первые многозадачные ОС

Следующий важный период развития вычислительных машин относится к началу 60-х – 1980 г. В это время в технической базе произошел переход от отдельных полупроводниковых элементов типа транзисторов к интегральным микросхемам. Вычислительная техника становится более надежной и дешевой. Растет сложность и количество задач, решаемых компьютерами. Повышается производительность процессоров.

Повышению эффективности использования процессорного времени мешает низкая скорость работы механических устройств ввода-вывода (быстрый считыватель перфокарт мог обработать 1200 перфокарт в минуту, принтеры печатали до 600 строк в минуту). Вместо непосредственного чтения пакета заданий с перфокарт в память начинают использовать его предварительную запись, сначала на магнитную ленту, а затем и на диск. Когда в процессе выполнения задания требуется ввод данных, они читаются с диска. Точно так же выходная информация сначала копируется в системный буфер и записывается на ленту или диск, а печатается только после завершения задания. Вначале действительные операции ввода-вывода осуществлялись в режиме оффлайн, то есть с использованием других, более простых, отдельно стоящих компьютеров. В дальнейшем они начинают выполняться на том же компьютере, который производит вычисления, то есть в режиме оналйн. Такой прием получает название или подкачки-откачки данных. Введение техники подкачки-откачки в пакетные системы позволило совместить реальные операции ввода-вывода одного задания с выполнением другого задания, но потребовало разработки аппарата прерыванийдля извещения процессора об окончании этих операций.

Магнитные ленты были устройствами последовательного доступа, то есть информация считывалась с них в том порядке, в каком была записана. Появление магнитного диска, для которого не важен порядок чтения информации, то есть устройства прямого доступа, привело к дальнейшему развитию вычислительных систем. При обработке пакета заданий на магнитной ленте очередность запуска заданий определялась порядком их ввода. При обработке пакета заданий на магнитном диске появилась возможность выбора очередного выполняемого задания. Пакетные системы начинают заниматься планированием заданий: в зависимости от наличия запрошенных ресурсов, срочности вычислений и т.д. на счет выбирается то или иное задание.

Дальнейшее повышение эффективности использования процессора было достигнуто с помощью мультипрограммирования. 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. Эта идея напоминает поведение преподавателя и студентов на экзамене. Пока один студент программа обдумывает ответ на вопрос операция ввода-вывода, преподаватель процессор выслушивает ответ другого студента естественно, такая ситуация требует наличия в комнате нескольких студентов. Точно так же мультипрограммирование требует наличия в памяти нескольких программ одновременно. При этом каждая программа загружается в свой участок оперативной памяти, называемый разделом, и не должна влиять на выполнение другой программы.

Появление мультипрограммирования требует настоящей революции в строении вычислительной системы. Особую роль здесь играет аппаратная поддержка (многие аппаратные новшества появились еще на предыдущем этапе эволюции), наиболее существенные особенности которой перечислены ниже.

Реализация защитных механизмов. Программы не должны иметь самостоятельного доступа к распределению ресурсов, что приводит к появлению привилегированных и непривилегированных команд. Привилегированные команды, например команды ввода-вывода, могут исполняться только операционной системой. Говорят, что она работает в привилегированном режиме. Переход управления от прикладной программы к ОС сопровождается контролируемой сменой режима. Кроме того, это защита памяти, позволяющая изолировать конкурирующие пользовательские программы друг от друга, а ОС – от программ пользователей.

Наличие прерываний. Внешние прерывания оповещают ОС о том, что произошло асинхронное событие, например завершилась операция ввода-вывода. Внутренние прерывания (сейчас их принято называть исключительными ситуациями ) возникают, когда выполнение программы привело к ситуации, требующей вмешательства ОС, например деление на ноль или попытка нарушения защиты.

--------------------------------------------------------------------------

Прямой доступ к памяти и организация каналов ввода-вывода позволили освободить центральный процессор от рутинных операций

При организации мультипрограммирования операционная система выпол. ряд функций

1.орагизация интерфейса между прикладной программой и оспри помощи сис.вызовов

2.орагизация очереди заданий в памяти и выделение ресурсов процессора одному из заданий (в последствии для этого стали испол. сопроцессоры и соот. планировалась их деятельность

3.переключение выполнения ресурсов процессора с одного задания на другое требовало сохранения содержимого регистров и структур данных необходимых для выполнения соответствующих заданий и обеспечения правильности продолжения вычислений.

4.использовались спец. стратегии управ. Памятью т.к. память являлась огр.ресурсом для этого упорядочивались процессы размещения замещения и выборки информации из памяти

5.использовалась спец. организация хранения информации на внешних носителях в виде файлов при этом обеспечивался доступ к определенным файлам только для опред категорий пользователей.

6.Для организации санкционированного обмена данных были внедрены средства коммуникации (обмена) данными между программами

7.для корректного обмена данными между пк были введены спц. Механизмы (алгоритмы) которые управляли конфликтными ситуациями при работе с различн ресурсами (средства синхронизации ресурсов и данных)

Мультипрограммные сис. Обеспечили возможность более эффективного использования сис. Ресурсов (памяти процессора периферийных устройств), однако данные системы еще достаточно долго оставались *пакетными*

Пользователь не мог непосредственно взаимодействовать с заданием и должен был предусмотреть с помощью управляющих карт все возможные ситуации, отладка программ занимала много времени и требовала изучения (анализа) многостраничных распечаток содержимого памяти и регистров

Появление электронно-лучевых дисплеев и переосмысление возможностей применения клавиатур поставили на очередь решение данных проблем

Расширением сис. Мультипрограм. Стали так называемые «системы разделения времени» в которых процессор переключ. Между задача мине только на время операции ввода-вывода, но и по прошествии определённого момента времени

Данные переключения происходили достаточно часто, и пользователи могли взаимодействовать со своими программами во время их выполнения (те интерактивно) тем самым появилась возможность одновременной работы нескольких пользователи на одной комп. Сис.

Для того чтобы уменьшить ограничение на кол-во работающих пользователей была внедрена идея неполного нахождения испрл. Программы в ОЗУ. Основная часть программы находилась н а диске и фрагмент, который необходимо было в данное время выполнить загружался в ОЗУ, а ненужная (выполненная) выкачивалась обратно на диск. Данный алгоритм был реализован с помощью механизма (виртуальной памяти). Основным результатом и достоинством этого механизма являлось создание иллюзии и пользователя в неограниченности ОЗУ ЭВМ

В сис разделения времени пользователь получил возможность эффективно производить отладку программы в интерактивном режиме и записывать инф на диск, не используя перфокарты, а непосредственно работая с клавиатуры

Параллельно эволюции внутренних систем, происходила и внешняя эволюция

В начале 3его периода появилась идея создания семейств программно-совместимых машин, работающих под управлением одной и той же ОС

Идея стандартизации операционных систем была широко внедрена в сознание пользователей и получила развитие по настоящее время

4 период развития ОС (1980 по настоящее время) ПЕРСОНАЛЬНЫЕ КОМПЬЮТЕРЫ классические, сетевые и распределенный вычислительные системы

Данный период развития был связан с появлением БИС и в эти годы произошло резкое возрастание степени интеграции и снижения стоимости микросхем. ПК со стандартной архитектурой по цене и простоте испл стал доступен отдельному чел, а не отделу предприятия и организации университета, как было раньше, наступила эра ПК и на правом этапе ПК испол одним пользователем в однопрограммном режиме, что привело к деградации архитектуры ЭВМ и соответственно их операционных сис ( в частности исчезла необходимость защ памяти и файлов, необход планирования заданий, распред ресурсов и др), но при этом ПК стали испол не только программистами, но и специалистами других направлений, что потребовало разработки (дружественного ПО и интерфейса)

Рост сложности и разнообразия задач, решаемых на ПК, потребовали необходимости повышения надежности их работы и со временем привели к возрождению практически всех характеристик архитектуры больших вычислительных систем. В середине 80х стали развиваться сети компьютеров, в том числе персональных и они требовали внедрения сетевых(распределенных) ОС. Пользователи должны были ориентироваться в получении доступа к ресурсу другого компьютера, соотв управлять интерфейсом доступа

Распределенная сис ПК внешне выглядела как обычная автономная сис, пользователь не знал, где хранятся его файлы, где они выполняются (на локальной или удаленной машине)

Анализ основных исторических этапов развития ОС позволяет выделить 6 основных функций, которые выполняли ОС в процессе своей эволюции:

1.Планирование заданий и испол процессора

2.Обечпечение программ средствами коммуникации и синхронизации

3.управление памятью

4.управление файловой сис

5.управление ввода-вывода

6.обеспечение безопасности

Традиционна каждая из функций ОС реализуется в виде отдельной подсистемы, явл структурным компонентом ОС. В каждой конкретной ОС они реализуются по разному ( в разном объеме) и в настоящее время до сих пор модернизируются и развиваются

ОС сущ для того, что на данный момент их сущ это наиболее эффективный способ использования вычислительных систем

Лекция 2. Основные понятия и концепции ОС

В процессе эволюции ОС используются несколько важных или основных понятий, которые составляют теорию и практику изучения ОС.

1. системные вызовы

В любой ОС поддерживается механизм, который позволяет пользовательским программам обращаться к услугам ядра ОС. В ОС различных производителей эти механизмы называются по- разному (например в Win такие средства называют сис вызовами, в ОС IBM они назывались сис макрокомандами, в ОС совет.комп такие средства назывались экстракодами).

Сис вызовы – это интерфейс между ОС и пользовательской программой. Они создают, удаляют и используют различные объекты, главными из которых являются процессы и файлы. Пользовательская программа в процессе своей работы запрашивает сервис у ОС, осуществляя сис вызов. Библиотеки процедур, которые загружают машинные регистры опред параметрами и осущ прерывания процессора, после чего управ передается обработчику данного вызова, входящего в ядро ОС. Целью данных библиотек явл – сделать сис вызов, похожим на обычный вызов подпрограммы. Основным отличием явл. То что при сис вызове задача или алгоритм переходят в работу в привилегированном режиме, поэтому сис вызовы называют еще «программными прерываниями» .

Ядро ОС имеет полный доступ к памяти пользовательской программы и при сис вызове достаточно передать адреса одной иди нескольких областей памяти с параметрами вызова для получения результатов вызова

В большинстве современных ОС сис вызов осущ программой программного прерывания INT.

2. Аппаратное прерывание – событие, генерируемое внешним по отношению к процессору устройством. С помощью аппаратных прерываний аппаратура компьютера либо информирует ЦП о том, что случилось какое-то событие, требующее немедленной реакции, либо сообщает о завершении некоторой асинхронной операции (например ввода-вывода) о том что было завершено чтение данных с диска в основную память. Важным типом аппаратных прерываний явл прерывание таймера, который генерируется периодически через фиксированный промежуток времени. Они испл ОС при планирование процессов. Каждый тип апп прерыв имеет собственный номер, однозначно определ источник прерывания. Апп прерыв это асинхронное событие, возникающие вне зависимости от того, какой код испол процессором в данный момент и обработка апп прерыв не должна учитывать, какой процесс явл текущим

3. Исключительные ситуации – события, возникающие в результате попытки выполнения программой команды, которая по каким-либо причинам не может быть выполнена до конца. Например: могут быть попытки доступа к ресурсу при отсудив достаточных привилегий или обращение к отсутствующей странице памяти

ИС как и СВ явл синхронными событиями, возникающими в результате выполнения текущей задачи. ИС можно разделить на исправимые и неисправимые

Исправимые: в случае отсутствия инф в ОЗУ. После устранения причины исчезает сама исключ ситуация, программа может выполнять работу дальше, поэтому возникновения в процессе работы ОС исправимых ИС считается нормальным явлением

Неисправимые: возникают чаще всего в результате возникновения ошибок в программах, а пример деление на 0, обращ к несущ модулям как правило ОС на данные ситуации реагирует завершением работы программы, которые вызвали данную искл ситуацию

4.Файлы – предназначены для хранения информации на внешних носителях. Понимают именованную часть пространства на носители информации. Управление через файловую сис, основн функции которой скрыть особенности ввода-вывода и дать программисту простую абстрактную модель файлов, независимых от устройств. Для управления файлами испол широкая категория сис вызовов.

При организации файловой сис исполз понятие каталога и пути

Архитектурные особенности ОС

1. Монолитное ядро

ОС – это программа состоит из неких процедур и функций

Компоненты явл не самостоятельными модулями, а составными частями одной большой программы. Данная структура ОС называется монолитным ядром. Монолитное ядро представляет собой набор процедур, каждая из кторых может вызывать другую или соседнюю, все процедуры работают в превелигированном режиме, испол и взаимодействуют друг с другом путем испол механизма вызова процедур

Совеременный сис предполагают сборку или компиляцию монолитного ядра отдельно для каждого комп на которую ставят ос при этом пользователь выбирает список оборудования и программных протоколов поддержка которых будет включена в ядро

Тк ядро явл единой программой то перекомпиляция его это едиснтвенный способ добавить в него новые компоненты или исключить неиспользованые. Присутствие в ядре лишних компонентов нежелательно тк оно постоянно в ОЗУ и требует ресурсов

Исключение ненужных компонентов из ядра повышает надежность ОС в целом. Монолитное ядро — это наиболее традиционный способ организации ОС

Допол в монолитном ядре выделяются сис процедуры соотв сис вызовам. Данные вызовы выполняются в привилегированном режиме в то время, как обычные пользовательские работы такого не могу

1. Многоуровневые системы

Вычислительную сис можно разбить на ряд уроней с чотка определенными связями между ними, таким образом шоб объекты уровня м магли визивать толька объекты уровня н-1.

Веохний уровень интерфейс пользователя

Чем ниже уровень выч сис тем более привилигированные команды и действие может выполнять модуль, наход на данном уровне

Впервые данны подход был предложен программистом

Уровень 5. Интерфейс пользователя

Уровень 4. Управление вводом-выводом

Уровень 3. Драйвер устройства связи оператора и консоли

Уровень 2. Управление памятью

Уровень 1. Планирование задачи процессов

Уровень 0. Аппаратные средства

Данная сис относится к слоеным типам достаточно хорошо реализуется при испол операций очередного нижнего слоя не нужно знать как они реализованы, а нужно лишь понимать, что они делают, слоеные или многоуровневые сис хорошо тестируются, отладка начинается с нижнего слоя к верхнему и идет послойно при обнаружении ошибок разработчики могут быть уврены что ошибки находятся в текущем слое, слоеные сис или многоуровневые хорошо модифицируются, при необходимости можно заменить лишь 1 слой, не трогая остальные

НО слоеные сис сложны для разработки, сложно определить порядок слоев и какой параметр к какому слою относится, слоеные сис менее эффектив чем монолитные, например для выпол ввода-вывода программе пользователя придется последовательно пройти все слои от верхнего до нижнего

Пример ОС THE

1. Виртуальные машины

Рассматривая ОС, был базар о возможности рассмотрения ОС, как ВМ, при этом пользователю не нужно знать детали внутреннего устройства ПК. Пользователь работает с файлами, а не с аппаратными средствами. Рассмотрим ОС, которая реализует виртуальную машину для каждого пользователя

Каждая ВМ представляет польхователю аппартаные средства в виде копии железа выч сис включая процессор, (не)привилегированные команды, устройства ввода-вывода, прерывания

При попытки обратиться к такому виртуальному железу или аппаратным средствам на уровне привилегированных команд в действительности происходит сис вызов реальной ОС, которая и производит все необходимые действия. Данный подход позволяет каждому пользователю загрузить свою ОС на ВМ и делать с ней что захочется ваще пох не парились в принципе

Архитектура ВМ такова

|  |  |  |
| --- | --- | --- |
| Программа пользователя | Программа пользователя | Программа пользователя |
| MS-DOS | Linux | Win |
| Вирутальные аппаратные средства |  |  |
| Реальная ОС | | |
| Реальная hardware | | |
|  | | |

Архитектура виртуальных машин

Впервые реальной такой сис стали копьютеры семейства IBM 307

Недостатком такой архитектуры явл снижение эффективности работы виртуальных машин по сравнению с реальным пк, как правило они достаточно громоздкие

Приемущестом явл возможность на одной выч сис программ написанных для разных ОС

Микроядерная архитектура ОС

Современная тендеция в разработке операционных сис состоит в пернесении знач части сис кода на уровень пользователя и одновременно с этим самым минимизации ядра, поэтому данный подход к построению ядра называется микроядерной архитектурой

Большиносто составляющих ОС явл самостоятельными программами, в этом случае взаимодействие между ними обеспечивает спец модуль, назыв микро ядром

Микро ядро наботает в приливигированном режими и обеспеч взаимодейтвие между программаи, планирует испол процессора осущ первичную обработку прерываний оперции ввода вывода и базовое управление памяти

Структура микроядра состоит из слеж элементов

Микро ядро , которое взаимодействует со след компонентами

Менеджер файловой системы, менеджер сети, менеджер памяти, и приложения

Смешанные системы

Все рассмотренные подходы к построению операционных систем имеют свои достоинства и недостатки. В боьшинстве случаев современные операционные системы используют различные комбинации этих подходов. Так, например, ядро операционной системыLinux представляет собой монолитную систему с элементами микроядерной архитектуры. При компиляции ядра можно разрешить динамическую загрузку и выгрузку очень многих компонентов ядра – так называемых модулей. В момент загрузки модуля его код загружается на уровне системы и связывается с остальной частью ядра. Внутри модуля могут использоваться любые экспортируемые ядром функции.

Другим примером смешанного подход может служить возможность запуска операционной системы с монолитным ядром под управлением микроядра. Микроядро обеспечивает управление виртуаьной памятью и работу низкоуровневых драйверов. Все остальные функции, в том числе взаимодействие с прикладными программами, осуществляется монолитным ядром. Данный подход сформировался в результате попыток использовать преимущества микроядерной архитектуры, сохраняя по возможности хорошо отлаженный код монолитного ядра.

Наиболее тесно элементы микроядерной архитектуры и элементы монолитного ядра переплетены в ядре Windows NT. Хотя Windows NT часто называют микроядерной операционной системой, это не совсем так. Микроядро NT слишком велико, чтобы носить приставку "микро". Компоненты ядра Windows NT располагаются в вытесняемой памяти и взаимодействуют друг с другом путем передачи сообщений, как и положено в микроядерных операционных системах. В то же время все компоненты ядра работают в одном адресном пространс и активно используют общие структуры данных, что свойственно операционным системам с монолитным ядром. По мнению специалистов Microsoft, причина проста: чисто микроядерный дизайн коммерчески невыгоден, поскольку неэффективен.

Таким образом, Windows NT можно с полным правом назвать гибридной операционной системой.

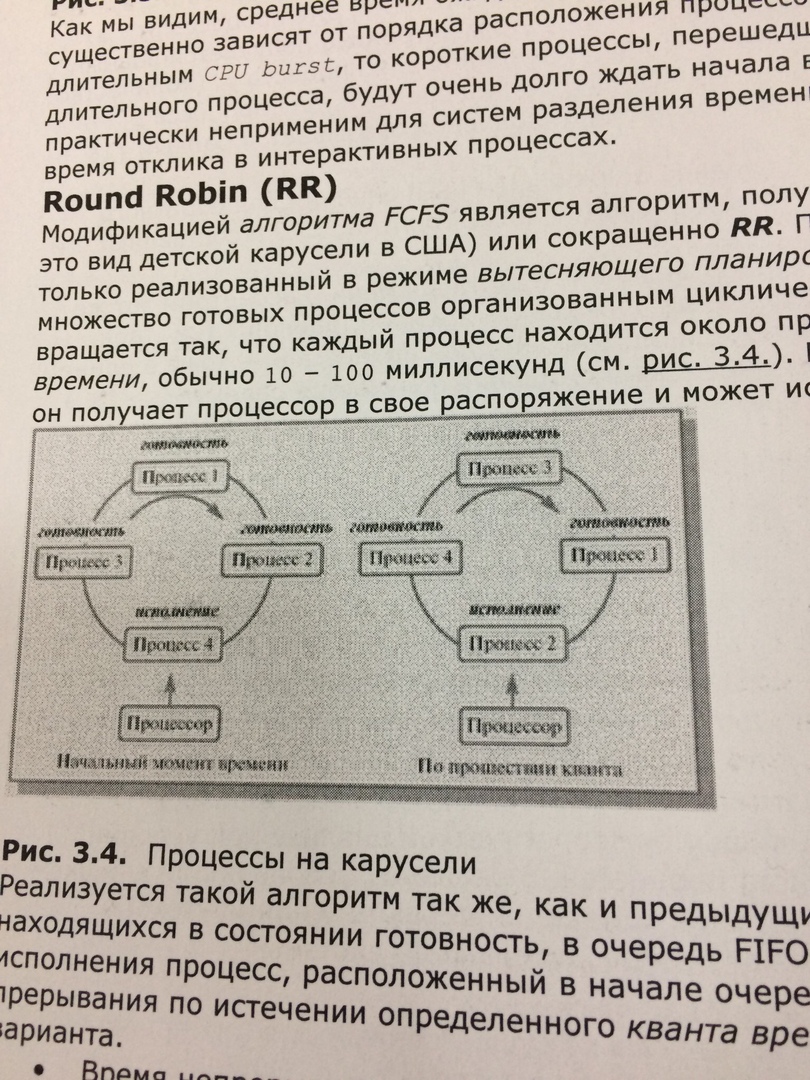
Алгоритмы палнирования

Простейшим алгоритмом планирования является алгоритм, который принято обозначать аббревиатурой FCFS по первым буквам его английского названия – первым пришел, первым обслужен. Представим себе, что процессы, находящиеся в состоянии готовность, выстроены в очередь. Когда процесс переходит в состояние готовность, он, а точнее, ссылка на его PCBпомещается в конец этой очереди. Выбор нового процесса для исполнения осуществляется из начала очереди с удалением оттуда ссылки на его PCB. Очередь подобного типа имеет в программировании специальное наименование – FIFO1, сокращение от первым вошел, первым вышел

Такой алгоритм выбора процесса осуществляет не вытесняющее планирование. Процесс, получивший в свое распоряжение процессор, занимает его до истечения текущего CPU burst . После этого для выполнения выбирается новый процесс из начала очереди.

Round Robin (RR)

Модификацией алгоритма FCFS является алгоритм, получивший название Round Robin –или сокращенно RR. По сути дела, это тот же самый алгоритм, только реализованный в режиме вытесняющего планирования. Можно представить себе все множество готовых процессов организованным циклически – процессы сидят на карусели карусель вращается так, что каждый процесс находится около процессора небольшой фиксированный квант времени, обычно 10 – 100 милсек пока процесс находится рядом с процессором, он получает процессор в свое распоряжение и может исполняться.



процессы на карусели

Реализуется такой алгоритм так же, как и предыдущий, с помощью организации процессов, находящихся в состоянии готовность, в очередь FIFO. Планировщик выбирает для очередного исполнения процесс, расположенный в начале очереди, и устанавливает таймер для генерации прерывания по истечении определенного кванта времени. При выполнении процесса возможны два варианта.

Время непрерывного использования процессора, необходимое процессу (, меньше или равно продолжительности кванта времени. Тогда процесс по своей воле освобождает процессор до истечения кванта времени, на исполнение поступает новый процесс из начала очереди, и таймер начинает отсчет кванта заново.

Продолжительность остатка текущего CPU burst процесса больше, чем квант времени. Тогда по истечении этого кванта процесс прерывается таймером и помещается в конец очереди процессов, готовых к исполнению, а процессор выделяется для использования процессу, находящемуся в ее начале.

Параметр планирования

Для осуществления поставленных целей разумные алгоритмы планирования должны опираться на какие-либо характеристики процессов в системе, заданий в очереди на загрузку, состояния самой вычислительной системы, иными словами, на параметры планирования . В этом разделе мы опишем ряд таких параметров, не претендуя на полноту изложения.

Все параетры планирования можно разбить на две большие группы: статические параметры и динамические параметры. Статические параметры не изменяются в ходе функционирования вычислительной системы, динамические же, напротив, подвержены постоянным изменениям.

К статическм параметрам вычислительной системы можно отнести предельные значения ее ресурсов (размер оперативной памяти, максимальное количество памяти на диске для осуществления свопинга, количество подключенных устройств ввода-вывода и т. п.). Динамические параметры системы описывают количество свободных ресурсов на данный момент.

К статическим параметрам процессов относятся характеристики, как правило присущие заданиям уже на этапе загрузки.

Каким пользователем запущен процесс или сформировано задание.

Насколько важной является поставленная задача, т. е. каков приоритет ее выполнения.

Сколько процессорного времени запрошено пользователем для решения задачи.

Каково соотношение процессорного времени и времени, необходимого для осуществления операций ввода-вывода.

Какие ресурсы вычислительной системы (оперативная память, устройства ввода-вывода, специальные библиотеки и системные программы и т. д.) и в каком количестве необходимы заданию.

Алгоритмы долгосрочного планирования используют в своей работе статические и динамические параметры вычислительной системы и статические параметры процессов (динамические параметры процессов на этапе загрузки заданий еще не известны). Алгоритмы краткосрочного и среднесрочного планирования дополнительно учитывают и динамические характеристики процессов. Для среднесрочного планирования в качестве таких характеристик может использоваться следующая информация:

сколько времени прошло с момента выгрузки процесса на диск или его загрузки в оперативную память;

сколько оперативной памяти занимает процесс;

сколько процессорного времени уже предоставлено процессу

