Д.А. Зайцев, С.М. Вороной, Т.Р. Шмелёва

Трассировка процессов функционирования сетевых операционных систем

Методические указания к лабораторным и практическим занятиям

Донецьк
ІПШІ «Наука і освіта»
2007

УДК 004.4’6
Т 65

Рецензенти:
Кандидат технічних наук, доцент
Донецького національного технічного університету
О.А. Дмітрієва
Кандидат технічних наук, доцент
Донецького державного інституту штучного інтелекту
Р.М. Бабаков

Зайцев Д.А. , Вороной С.М., Шмелёва Т.Р.
Т 65 Трассировка процессов функционирования сетевых операционных систем: Методические указания к лабораторным и практическим занятиям. – Донецьк: ІПШІ «Наука і освіта», 2007. – 68 с.

Представлены индивидуальные задания для проведения лабораторных и практических занятий по основным разделам теории сетевых операционных систем. Каждая тема начинается кратким изложением теоретического материала, затем следует общее описание задания, пример его выполнения, контрольные вопросы; в приложениях приведены варианты заданий.

УДК 004.4’6

Рекомендовано до видання Вченою радою
Донецького державного інституту штучного інтелекту

Д.А. Зайцев, С.М. Вороной, Т.Р. Шмелёва

Трассировка процессов функционирования сетевых операционных систем

Методические указания к лабораторным и практическим занятиям

Рекомендовано
Ученым советом ДонГИИИ,
протокол № 8 от 29.05.2007 г.

Донецк
ИПИИ «Наука і освіта»
2007

Навчальне видання

Дмитрий Анатольевич Зайцев
Сергей Михайлович Вороной
Татьяна Рудольфовна Шмелёва

ТРАССИРОВКА ПРОЦЕССОВ ФУНКЦИОНИРОВАНИЯ СЕТЕВЫХ ОПЕРАЦИОННЫХ СИСТЕМ

Методические указания
к лабораторным и практическим занятиям

Відповідальний редактор С.Б. Іванова
Технічний редактор В.М. Пігуз
Комп'ютерна верстка Т.Є. Нікітіна
Коректор К.С. Хлиста

Здано до набору 00.00.2007. Підписано до друку 00.00.2007. Формат 60×84/16.
Обл.-вид. арк. 0,00. Тираж 000 прим. Зам. 00/07 від 00.00.2007.
Ціна договірна

Віддруковано в Інституті проблем штучного інтелекту МОН і НАН України
Свідоцтво № 444, серія ДК від 08.05.2001 р.
Україна, 83050, м. Донецьк, пр. Б. Хмельницького, 84
тел. (062) 337-01-70
edoffice@iai.donetsk.ua www.iai.donetsk.ua

Содержание

Введение

Большинство современных сетевых операционных систем обладают практически одинаковым набором возможностей и отличаются лишь особенностями их реализации, а также организацией интерфейсов пользователя и администратора. Кроме того, конкретные операционные системы зачастую вводят собственную терминологию, что затрудняет понимание общих принципов функционирования сетевых операционных систем.

В настоящем методическом пособии изучение основных возможностей современных сетевых операционных систем выполнено в соответствии с классической терминологией теории операционных систем, что позволяет затем применить полученные знания и навыки к анализу конкретных операционных систем на лабораторных занятиях. Рекомендуемыми к дальнейшему изучению на лабораторных занятиях являются операционные системы семейств MS Windows, Unix, QNX. Кроме того, возможно изучение специализированных операционных систем коммутаторов, маршрутизаторов и других сетевых устройств, например операционных систем IOS компании Cisco. Целесообразным представляется также проведение семинарских занятий по ретроспективному изучению операционных систем, внесших значительный вклад в развитие теории операционных систем, таких как MULTICS, OS/360, RSX11M, VAS/VMX.

Существуют различные уровни освоения предмета «Сетевые операционные системы», такие как уровень квалифицированного пользователя, уровень администратора и уровень проектировщика (разработчика). Базовым уровнем для направления «Телекоммуникации» является уровень администратора. Однако квалифицированное администрирование в настоящее время практически невозможно без освоения основ проектирования операционных систем, предполагающих знание структур данных и алгоритмов

работы компонентов операционной системы. Задания по трассировке процессов функционирования ОС представляют собой основу для изучения алгоритмов работы компонентов сетевых операционных систем.

Структура и организация функционирования современных сетевых операционных систем

Операционная система – это комплекс программ для управления ресурсами компьютера, организации интерфейсов с пользователем и обеспечения защиты ресурсов.

Таким образом, ОС, решает две группы задач:

- управление ресурсами;
- организация интерфейсов.

Различают 4 типа ресурсов:

- процессоры;
- оперативная память;
- устройства ввода/вывода;
- информация.

Рассматривают 4 типа операций по управлению ресурсом:

- выделение;
- освобождение;
- отслеживание;
- планирование.

Таким образом, управление ресурсами представлено 16 компонентами операционной системы (4 типа ресурсов х 4 типа операций). Кроме того, ввиду существенных отличий устройств ввода/вывода (например, сетевая карта и магнитный диск) выполняют дальнейшее структурирование средств управления устройствами. Компонент операционной системы, организующий ввод/вывод для конкретного типа устройств называют драйвером.

Среди интерфейсов операционной системы наиболее известным является интерфейс с пользователем. Как правило, современные ОС реализуют две разновидности пользовательского интерфейса: текстовый командный интерфейс и графический. Лаконичность текстового командного интерфейса обусловливает его широкое применение в сетях для удалённого доступа.

Запущенная программа представляет собой единицу работы для операционной системы и именуется процессом. Интерфейсы процессов и оборудования недоступны непосредственно конечному пользователю. Интерфейс процессов представляет собой набор системных вызовов, с помощью которых выполняемая программа обращается за сервисом к операционной системе, например, для выполнения операций ввода/вывода. Исполняя системные вызовы, операционная система взаимодействует с оборудованием, используя его физический интерфейс, представленный в архитектуре компьютера, например, множеством портов ввода/вывода и аппаратными прерываниями.

Таким образом, операционная система запрещает непосредственное обращение пользовательских программ к оборудованию и выступает в качестве посредника, выполняя конкретные операции на устройствах от имени пользовательских процессов. Описанная технология является основой для реализации функций защиты пользовательских процессов друг от друга и самой операционной системы от пользовательских процессов. Обеспечение защиты возможно лишь при соответствующей аппаратной поддержке: наличии нескольких режимов работы процессора с различным множеством доступных команд. В простейшем случае используется 2 режима: «система» для исполнения ОС и «пользователь» для исполнения процессов пользователя. Попытка выполнения процессом пользователя привилегированной команды (например, записи в порт устройства) вызывает специальное прерывание, перехватываемое ОС.

Классической является реализация компонентов операционной системы в форме ядра и множества систем-

ных процессов. Ядро представляет собой часть операционной системы, которая постоянно находится в оперативной памяти и не является процессом. Фактически ядро представляет собой множество обработчиков прерываний: аппаратных – для управления вводом/выводом, памятью и процессорами; программных – для реализации системных вызовов. В системные процессы выделяют компоненты ОС, требующие значительных временных затрат в процессе выполнения, например, файловые операции, командный и графический интерфейсы пользователя.

Как правило, при загрузке операционной системы в оперативную память загружается ядро, перехватывающее все прерывания, и таким образом захватывающее полный контроль над оборудованием, а также несколько начальных процессов, обеспечивающих интерфейс с пользователем. Далее, пользователь может запускать собственные процессы, а операционная система инициировать запуск требуемых системных процессов.

Общая характеристика заданий по трассировке процессов функционирования сетевых ОС

Материал методического пособия структурирован в соответствии с основными ресурсами операционной системы: процессоры (процессы), оперативная память, устройства, информация. Большинство занятий предполагает изучение простейших операций ОС: выделение и освобождение соответствующих ресурсов. При изучении средств управления информацией особое внимание уделено операциям отслеживания состояния ресурса, реализуемым с помощью специальной файловой структуры диска. Наиболее сложная операция планирования ресурса изучена на примере планирования очереди запросов к магнитному диску.

В качестве набора основных возможностей современных операционных систем, изучаемых в настоящем пособии, выбраны следующие:

- мультипрограммирование;
- циклическое квантование времени;
- приоритетные дисциплины;
- динамические разделы;
- свопинг;
- виртуальная (страничная) память;
- буферизация ввода/вывода;
- технология клиент-сервер;
- планирование дисковых операций;
- файловая структура диска.

Типовое задание состоит в выполнении ручной трассировки процессов функционирования компонентов ОС, реализующих указанные возможности, при обработке заданной последовательности процессов (запросов). Заполняемые трассировочные таблицы представляют собой упрощённое представление фрагментов базы данных операционной системы. Трассировочная таблица отображает процессы изменения базы данных ОС во времени в результате выполнения операций. Она является основой для дальнейшего анализа преимуществ и недостатков изучаемых возможностей операционных систем.

В процессе выполнения заданий студент выполняет действия в точности таким же образом, как это делает операционная система, что позволяет ему понять сложные алгоритмы функционирования современных операционных систем. Ручная трассировка может рассматриваться также как первый шаг к изучению основ проектирования операционных систем. Её логическим продолжением является описание алгоритмов функционирования компонентов операционной системы, взаимосвязей компонентов и детализированного представления базы данных ОС. Примеры описания алгоритмов и объектов базы данных ОС приведены в Приложении 2.

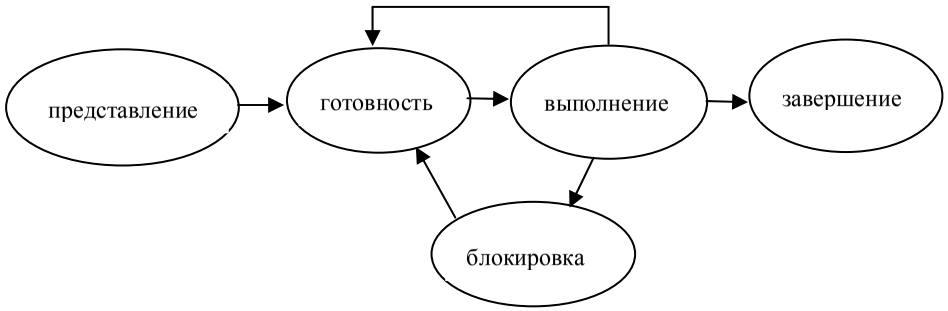
Раздел 1. Управление процессами

Ключом к пониманию функционирования средств управления процессорами (процессами) современных ОС является диаграмма состояний процесса. Модули ОС обеспечивают изменение состояний процессов в соответствии с диаграммой, переключение контекста процессора для запуска текущего процесса, вызов подсистемы ввода-вывода для инициирования операций на внешних устройствах.

Основными состояниями процесса являются:

- представление;
- готовность;
- выполнение;
- блокировка;
- завершение.

В состоянии представления запущенному процессу выделяются требуемые ресурсы; в состоянии готовности процесс обладает всеми необходимыми для выполнения ресурсами за исключением центрального процессора (ЦП); в состоянии выполнения программный код процесса фактически выполняется на процессоре; в состоянии блокировки процесс ожидает события (например, завершения ввода-вывода); в состоянии завершения процесс освобождает ресурсы ОС. Абстрактная диаграмма состояний имеет следующий вид:



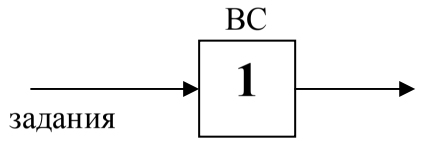
При проектировании ОС абстрактная диаграмма дополняется множеством состояний, отображающих различные причины блокировок и этапы выделения (освобождения) ресурсов.

Занятие 1.1. Мультипрограммирование

Цель занятия: освоить основы организации мультипрограммного режима работы операционных систем, оценить преимущества мультипрограммирования

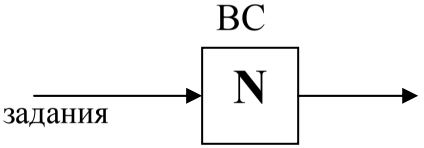
Краткое изложение теоретического материала

Первые вычислительные системы (ВС) работали в однопрограммном режиме. ОС начинала выполнение нового задания лишь в случае полного завершения текущего задания. Схематически это можно представить следующим образом:

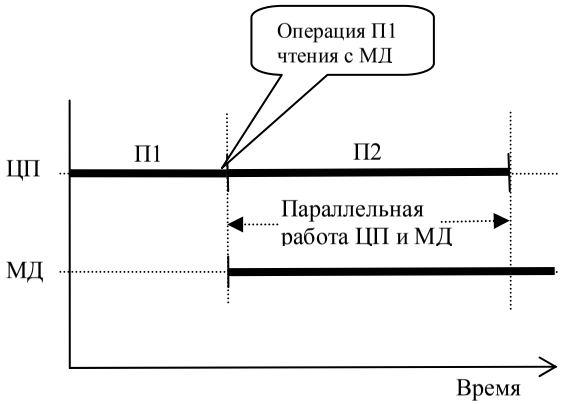


Это приводило к неэффективному использованию ресурсов вычислительной системы в связи с простоем оборудования. Только одно из многочисленных устройств (процессор, магнитный диск, принтер, магнитная лента) работало в конкретный момент времени.

Мультипрограммные ОС позволяют загрузить множество устройств вычислительной системы за счёт одновременного выполнения нескольких заданий, что приводит к сокращению суммарного времени выполнения смеси заданий. Схематически ОС, выполняющую N заданий, можно представить следующим образом:



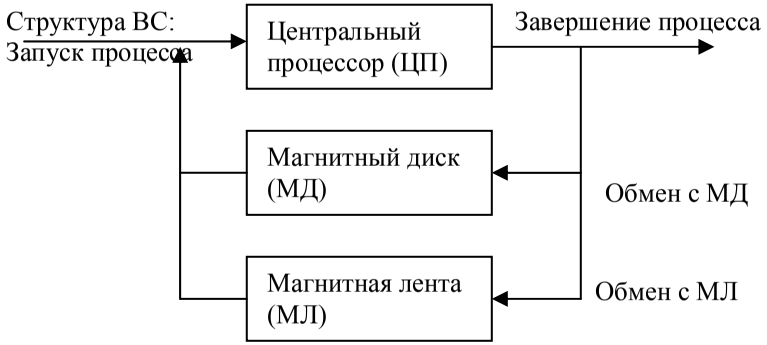
Число N называют коэффициентом мультипрограммирования. Сокращение суммарного времени выполнения достигается за счёт совмещении работы процессора с работой внешних устройств. В то время как Процесс 2 выполняется на процессоре, Процесс1 может обслуживаться на магнитном диске:



Следует отметить, что мультипрограммный режим работы требует соответствующей аппаратной поддержки. Оборудование должно обеспечивать автономную работу внешних устройств, управляемых своим контроллером (каналом), а также систему прерываний для сигнализации внешних устройств о завершении операции.

Задание

Выполнить ручную трассировку работы средств управления процессами мультипрограммной ОС. Оценить эффективность работы мультипрограммной ОС.



Порядок выполнения

1. Выполнить ручную трассировку выполнения указанной смеси процессов:
   а) в однопрограммной вычислительной системе;
   б) в мультипрограммной вычислительной системе.
2. Заполнить трассировочные таблицы
3. Выполнить анализ эффективности мультипрограммного режима:
   3.1. Оценить ускорение выполнения смеси процессов
   3.2. Рассчитать и сравнить коэффициенты загрузки устройств
4. Сформулировать преимущества мультипрограммного режима

Пример выполнения

Характеристики выполняемых процессов

Номер	Время поступления	Последовательность действий
1	0	ЦП(40)-МД(100)-ЦП(20)
2	10	ЦП(20)-МД(100)-ЦП(80)
3	20	ЦП(60)-МД(100)-ЦП(10)

Пример трассировки

Время	Очередь готовых процессов	Процес-сор	Очередь к МД	МД
0		П1(40)		
10	П2(20)	П1(30)		
20	П3(20),П2(20)	П1(20)		
40	П3(20)	П2(20)		П1(100)
60		П3(60)	П2(100)	П1(80)
120			П3(100), 2(100)	П1(20)
140		П1(20)	П3(100)	П2(100)
160			П3(100)	П2(80)
240		П2(80)		П3(100)
320				П3(20)
340		П3(10)		
350				

Ускорение выполнения смеси процессов

Время выполнения смеси в однопрограммном режиме:

T1=(40+100+20)+(20+100+80)+(60+100+10)=530

Время выполнения смеси в мультипрограммном режиме:

Тм=350

Ускорение

A=T1 / Тм = 530 / 350=1,51

Коэффициенты загрузки устройств

K=Тустр / Тобщ

Однопрограммный	Мультипрограммный
Кпроц = 230 / 530=0,45 Кмд=300 / 530=0,57	Кпроц = 230 / 350=0,66 Кмд=300 / 350=0,86

Варианты заданий – Приложение 1.1.

Контрольные вопросы

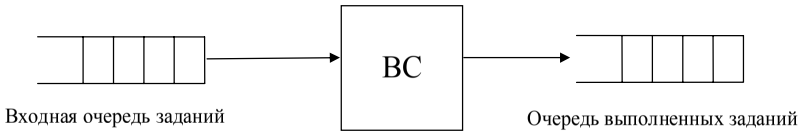
1. В чём состоит мультипрограммный режим работы ОС?
2. За счёт чего сокращается время выполнения смеси процессов в мультипрограммном режиме?
3. Каковы требования к аппаратным средствам компьютера для организации мультипрограммного режима?
4. Каковы накладные расходы организации мультипрограммного режима?

Занятие 1.2. Циклическое квантование времени

Цель занятия: освоить основы организации режима разделения времени работы операционных систем, оценить преимущества режима разделения времени

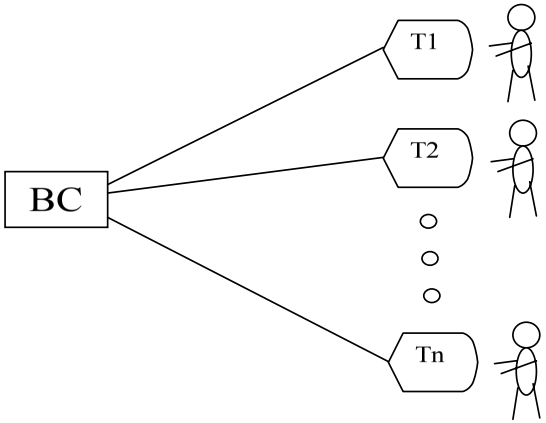
Краткое изложение теоретического материала

Первые мультипрограммные ОС функционировали в *пакетном режиме*, в котором пользователь отделялся от процесса выполнения его заданий. Предварительно подготовленные задания организовывались в очереди на перфокартах либо магнитном диске. Затем смесь заданий подавалась на вход вычислительной системы:

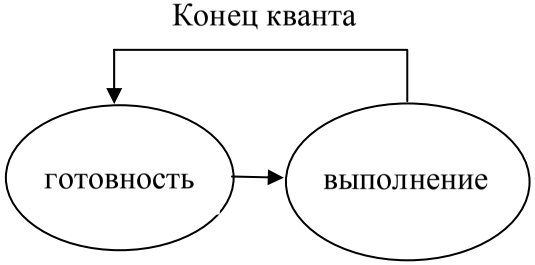


Необходимость взаимодействия пользователя с выполняющимися программами, автоматизация процессов подготовки заданий с помощью текстовых редакторов, а также появление дисплеев с электронно-лучевой трубкой обусловили появление ВС с *диалоговым режимом* работы. В этом режиме несколько пользователей одновременно взаимодействуют с ВС с помощью терминалов:

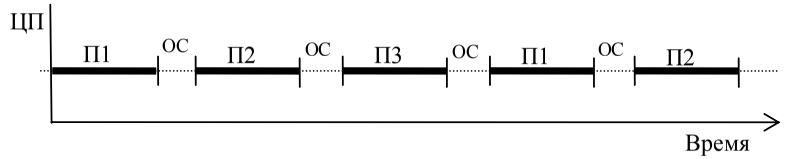
Для обеспечения работы нескольких пользователей в диалоговом режиме особенно важным становится *время реакции ОС* на команды пользователя. При использовании классического мультипрограммирования один из процессов может захватить процессор на неопределённое время и, таким образом, блокировать работу остальных пользователей. Эффективный диалоговый режим возможен при гарантированном времени реакции ОС не превышающем несколько секунд.



Для обеспечения диалоговой работы был предложен *режим разделения времени*, обеспечивающий совместную работу нескольких пользователей. Основой для обеспечения режима разделения времени является *циклическое квантование времени* центрального процессора. Каждому процессу при получении процессора выделяется определённый интервал времени – *квант*, в течение которого он выполняется на процессоре; затем процесс возвращается в конец очереди готовых процессов. Таким образом, процессор циклически переключается между готовыми процессами и каждый из пользователей считает, что ОС работает только с ним одним; далее представлен фрагмент диаграммы состояний процесса, обеспечивающий циклическое квантование:



Следует отметить, что организация квантования влечёт за собой некоторое снижение производительности ВС. Переключение между процессами требует определённого времени, в течение которого работают программы ОС, и представляет собой *накладные расходы* ресурсов ОС при организации режима разделения времени. При реализации циклического квантования встаёт вопрос о выборе *оптимального размера кванта*. Большой размер кванта снижает время реакции ОС, малый – увеличивает накладные расходы из-за частого переключения. Реальные ОС применяют адаптивные механизмы выбора размера кванта.



Квантование трёх процессов

Для реализации квантования времени используется аппаратный таймер. Таймер устанавливается на продолжительность кванта при выделении процессора процессу; прерывание таймера по истечении времени кванта инициирует переключение.

Задание

Выполнить ручную трассировку работы средств управления процессами. Заполнить трассировочную таблицу.

Характеристики ОС: циклическое квантование времени, мультипрограммирование

Порядок выполнения

1. Выполнить ручную трассировку выполнения указанной смеси процессов
2. Заполнить трассировочные таблицы
3. Выполнить анализ эффективности режима разделения времени:

а) при мгновенном переключении процессов;
б) при времени переключения равном 1

4. Сформулировать преимущества и недостатки режима разделения времени

Пример выполнения

Размер кванта – 10

Характеристики выполняемых процессов

Номер	Время поступления	Последовательность действий
1	0	ЦП(40)-МД(100)-ЦП(20)
2	10	ЦП(20)-МД(100)-ЦП(80)
3	20	ЦП(60)-МД(100)-ЦП(10)

Пример заполнения трассировочной таблицы

Время	Очередь готовых процессов	Процессор	Очередь к МД	МД
0		П1(40)		
10	П1(30)	П2(20)		
20	П2(10), П1(30)	П3(60)		
30	П3(50),П2(10)	П1(30)		
40	П1(20),П3(50)	П2(10)		
50	П1(20)	П3(50)		П2(100)
60	П3(40)	П1(20)		П2(90)
70	П1(10)	П3(40)		П2(80)
80	П3(40)	П1(10)		П2(70)
90		П3(40)	П1(100)	П2(60)
100	…	…	…	…

Варианты заданий – Приложение 1.2.

Контрольные вопросы

1. В чём состоит режим разделения времени?
2. Как влияет режим разделения времени на производительность ВС?

3. Каковы требования к аппаратным средствам компьютера для организации квантования?
4. Какие критерии используют при выборе оптимального размера кванта?

Занятие 1.3. Приоритетные дисциплины

Цель занятия: освоить основы организации приоритетных дисциплин работы операционных систем, оценить преимущества приоритетных дисциплин

Краткое изложение теоретического материала

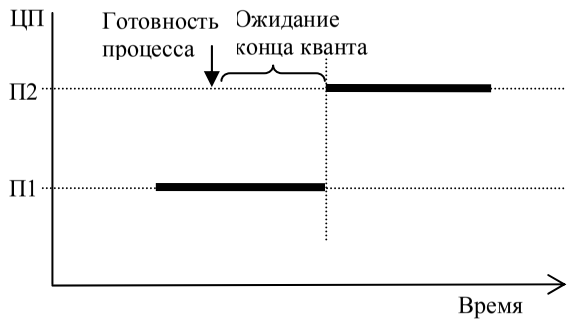
Множество процессов, исполняемых в среде ОС, можно разделить на *системные* и *прикладные*. Системные процессы являются частью операционной системы и обеспечивают работу ВС в целом. Таким образом, во многих случаях требуется обеспечить преимущественное выделение им ресурсов, в особенности, центрального процессора.

Кроме того, прикладные процессы также следует различать на основе предпочтительного выделения ресурсов, особенно при использовании ВС в технологическом управлении. Ввиду жёстких ограничений на время реакции и связанных с этим особенностей построения, операционные системы, предназначенные для технологического управления, выделяют в специальный класс, именуемый *операционные системы реального времени* (ОСРВ).

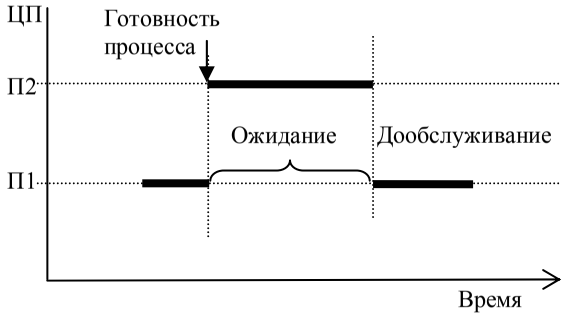
Приоритетные дисциплины планирования процессов были предложены как для оптимизации выполнения смеси заданий, так и для обеспечения оперативности исполнения системных процессов и процессов реального времени.

Уровень приоритета во многих случаях может быть представлен целым числом. Различают два типа приоритетов процессов: *относительные и абсолютные*. Относительные приоритеты используются только при постановке

процессов в очередь, таким образом, появление процесса с более высоким относительным приоритетом не прерывает исполнение на процессоре текущего процесса. Абсолютный приоритет сравнивается с приоритетом текущего выполняемого процесса и, в случае превышения, прерывает исполнение текущего процесса до завершения его кванта:



Относительный приоритет



Абсолютный приоритет

Таким образом, абсолютные приоритеты используются для мгновенного переключения на новый процесс, и применяется для процессов реального времени и наиболее

важных системных процессов, например, самой подсистемы управления процессами.

Различаю *статические и динамические приоритеты*. Статические приоритеты назначаются вручную администратором ОС, либо пользователем в определённых отведенных ему администратором ОС границах и не изменяются во время исполнения процессов. Динамические приоритеты назначаются и изменяются ОС во время выполнения процессов в целях оптимизации функционирования ВС. Например, ОС может повысить приоритет процесса, часто выполняющего ввод/вывод и понизить приоритет процесса, продолжительно занимающего процессор, для повышения загрузки устройств ВС.

Статические абсолютные приоритеты обеспечивают минимальное время реакции ОС РВ в технологическом управлении. Следует отметить, что применение приоритетных дисциплин не всегда оптимально с точки зрения критерия обеспечения максимальной загрузки устройств ВС. Для ОС РВ на первое место выдвигаются критерии минимального времени реакции, что оправдано, так как потери из-за неоперативного технологического управления могут значительно превышать потери из-за простоя устройств ВС.

Задание

Выполнить ручную трассировку работы средств управления процессами. Заполнить трассировочную таблицу.

Характеристики ОС: приоритетная дисциплина, мультипрограммирование

Порядок выполнения

1. Выполнить ручную трассировку выполнения указанной смеси процессов
2. Заполнить трассировочные таблицы
3. Выполнить анализ эффективности приоритетной дисциплины

4. Сформулировать преимущества и недостатки приоритетной дисциплины планирования

Пример выполнения

Характеристики выполняемых процессов

Номер	Время поступления	Приоритет	Последовательность действий
1	0	1	ЦП(40)-МД(100)-ЦП(20)
2	10	2	ЦП(20)-МД(100)-ЦП(80)
3	20	3 абс.	ЦП(60)-МД(100)-ЦП(10)

Пример заполнения трассировочной таблицы

Время	Очередь готовых процессов	Процессор	Очередь к МД	МД
0		П1(40)		
10	П2(20)	П1(30)		
20	П1(20),П2(20)	П3(60)		
80	П1(20)	П2(20)		П3(100)
100	…	…	…	…

Варианты заданий – Приложение 1.3.

Контрольные вопросы

1. Для чего используют приоритеты в операционных системах?
2. Какие специфические требования предъявляют ОС реального времени?
3. В чём состоят различия относительных и абсолютных приоритетов?
4. Для чего применяют динамические приоритеты?

Раздел 2. Управление памятью

Оперативная память (ОП) является вторым после ЦП ресурсом по своей значимости для исполнения процесса. Множество способов управления памятью конкретных ОС можно классифицировать по следующим основным признакам:
- полное либо частичное размещение процесса в ОП;
- связное либо несвязное выделение оперативной памяти;
- выделение участков памяти фиксированной либо переменной длины.

Многие сложные методы управления ОП, обусловленные исторически высокой стоимостью и ограничениями объёма физической памяти, например, оверлейные структуры, практически не используются в настоящее время в связи с развитием концепции виртуальной памяти. Стандартом де-факто для современных ОС является виртуальная сегментно-страничная память.

Занятие 2.1. Динамические разделы

Цель занятия: освоить основы организации управления памяти динамическими разделами, оценить эффективность управления памяти динамическими разделами

Краткое изложение теоретического материала

Управление оперативной памятью (ОП) *динамическими разделами* применялось в ОС ранних поколений; однако и в настоящее время этот способ управления памятью остаётся актуальным, так как используется для управлением пулом памяти базы данных ОС, в котором размещаются управляющие блоки ОС: блок управления

процессом, блок управления устройством, блок управления файлом и другие.

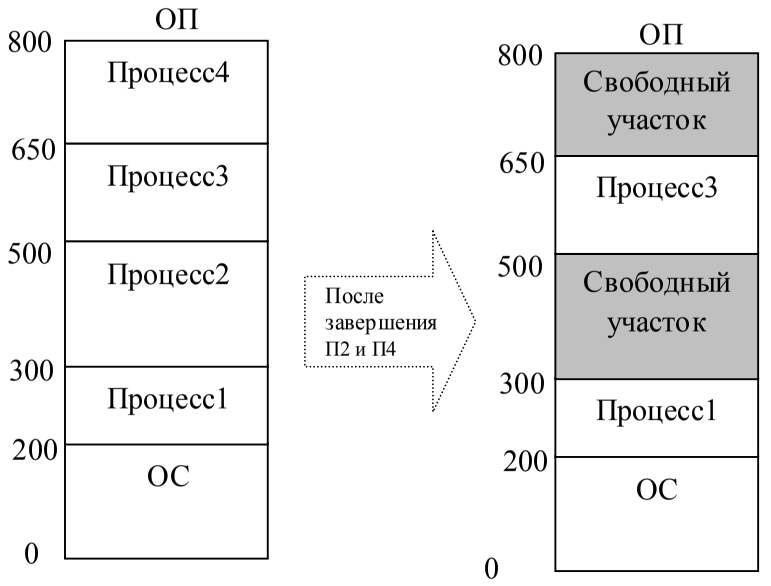
В ранних поколениях ОС оперативная память рассматривалась как одномерный массив байтов (слов). Для выделения процессу ЦП требовалось выполнение двух условий: полное размещение процесса (исполнимого файла) в оперативной памяти; связное выделение памяти процесса в виде одного непрерывного участка.

В этом случае текущее распределение памяти полностью описывается массивом границ (начальный и конечный адреса либо начальный адрес и длина) памяти процессов. Аналогичный описатель можно использовать и для оставшихся свободных участков ОП:

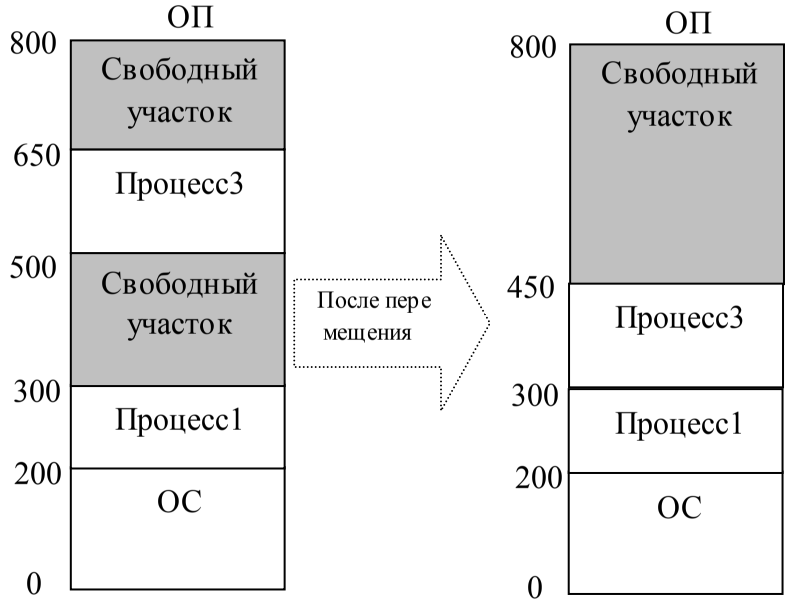
Память процессов		
Процесс	Начало	Длина
П1	200	100
П3	500	150

Свободная память	
Начало	Длина
300	200
650	150

Первоначально загруженные процессы занимают один непрерывный участок памяти вслед за кодом ОС. Однако ввиду случайного времени их завершения, свободные и занятые участки становятся разбросанными в массиве байтов ОП:



Такую ситуацию называют *фрагментацией памяти*. При выборе требуемого участка памяти для нового процесса ОС может применять критерии первого подходящего участка либо наиболее подходящего участка. Однако возможна ситуация, при которой общий объем свободной памяти удовлетворяет требованиям процесса, но составлен суммой размеров несмежных свободных участков. В этом случае был предложен специальный метод реорганизации ОП, называемый *перемещением*. Запускается специальный модуль управления памяти ОС, который сдвигает (копирует) все процессы в ОП для устранения фрагментации и образования одного непрерывного участка свободной ОП:



Следует отметить, что перемещение неизбежно влечёт за собой накладные расходы ОС в виде дополнительного времени копирования процессов в ОП. Хотя архитектура многих процессоров предлагает специальные операции быстрого копирования блоков памяти, эти расходы могут быть существенными для общей производительности ВС.

Мультипрограммный режим делает актуальными аппаратные средства защиты памяти для предотвращения (ошибочного либо намеренного) изменения кодов одного процесса некоторым другим процессом. Заметим, что средства защиты памяти крайне необходимы и однопрограммным ОС для защиты раздела памяти самой ОС от вмешательства исполняемого процесса. В большинстве ОС с управлением памятью динамическими разделами эта проблема решается путём использования *аппаратных регистров границ* текущего процесса. При обращении к адресам памяти вне границ генерируется аппаратное прерывание по защите памяти, которое затем обрабатывается ОС.

Задание

Выполнить ручную трассировку средств управления оперативной памятью. Заполнить трассировочную таблицу. Оценить эффективность управления памятью.

Характеристики ОС: динамические разделы, мультипрограммирование, приоритетная дисциплина.

Порядок выполнения

1. Выполнить ручную трассировку работы средств управления ОП
2. Заполнить трассировочные таблицы
3. Выполнить анализ накладных расходов на перемещение процессов
4. Сформулировать преимущества и недостатки управления памяти динамическими разделами

Пример выполнения

Размер ОП – 70

Характеристики выполняемых процессов

Номер	Время поступления	Время выполнения	Размер в ОП	Приоритет
1	0	30	30	2
2	20	40	20	1
3	30	50	40	3

Пример заполнения трассировочной таблицы

Время	Состояние ОП				Примечание
0	Процесс	П1 (акт.)	Своб.		
	Начало	0	30		
	Длина	30	40		
20	Процесс	П1 (акт.)	П2	Своб.	
	Начало	0	30	50	
	Длина	30	20	20	

Время	Состояние ОП	Примечание
30	Процесс | Своб. | П2 | Своб. Начало | 0 | 30 | 50 Длина | 30 | 20 | 20	Фрагментация
30	Процесс | П2 | Своб. Начало | 0 | 20 Длина | 20 | 50	Перемещение
30	Процесс | П2 | П3 (акт.) | Своб. Начало | 0 | 20 | 60 Длина | 20 | 40 | 10	
80	…	…

Варианты заданий – Приложение 1.4.

Контрольные вопросы

1. Каким образом описывается распределение памяти при использовании динамических разделов?
2. Почему возникает фрагментация памяти при использовании динамических разделов?
3. Каким образом ликвидируется фрагментация памяти?
4. Какие аппаратные средства применяются для защиты оперативной памяти?

Занятие 2.2. Свопинг процессов

Цель занятия: освоить основы организации свопинга процессов, оценить эффективность применения свопинга процессов

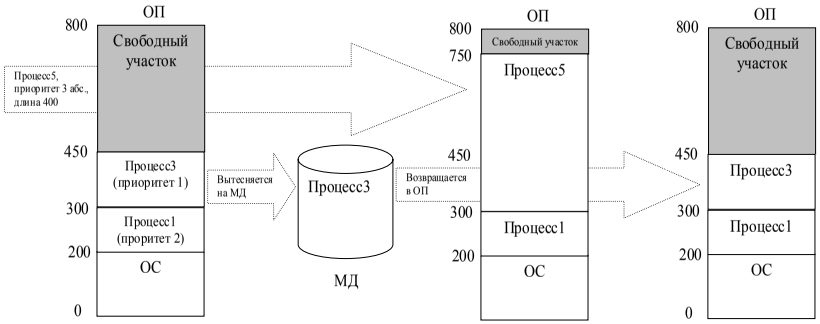
Краткое изложение теоретического материала

Методы *свопинга* (замещения) широко используются в управлении ОП современных ОС. Подвергаться свопингу (вытесняться) могут как процессы целиком, так и

отдельные участки памяти процессов. Современные ОС применяют свопинг страниц и сегментов процессов, предыдущие поколения ОС замещали процессы целиком, либо организовывали специальный набор замещаемых участков (оверлеи) внутри процесса.

Изучим наиболее простую разновидность: *свопинг процессов* при управлении памятью динамическими разделами. Во время длительной операции ввода-вывода на медленном устройстве процессор может выполнить несколько других процессов, но оперативная память занята. Аналогичная ситуация возникает при появлении нового процесса с абсолютным приоритетом. Хотя процесс может захватить процессор, ему недостаточно места в оперативной памяти.

Предложено простое решение: находится процесс с наименьшим приоритетом среди загруженных в ОП; участок памяти процесса целиком копируется на быстрое внешнее устройство (магнитный диск) и освобождается; высокоприоритетный процесс загружается в память и исполняется. При создании благоприятных условий (соотношение приоритетов) вытесненный процесс возвращается в ОП и продолжает выполнение:



Свопинг неизбежно связан с *накладными расходами* на выполнение операций выгрузки и загрузки процессов,

но в ОС РВ такие накладные расходы оправданы. Кроме того, в диалоговой или пакетной ОС свопинг может *повышать общую эффективность* работы ВС. В особенности, в сочетании с динамическими приоритетами, когда вытесняется «плохой» процесс с уровнем приоритета, ставшим ниже некоторой границы, а во время пребывания в вытесненном состоянии приоритет процесса постепенно повышается ОС.

Отметим, что для обеспечения свопинга на время выполнения длительной операции ввода-вывода (принтер, магнитная лента) требуется организация буферов в памяти ОС для промежуточного хранения данных процесса.

Задание

Выполнить ручную трассировку работы средств свопинга процессов. Заполнить трассировочную таблицу.

Характеристики ОС: свопинг процессов, динамические разделы, мультипрограммирование, приоритетная дисциплина

Порядок выполнения

1. Выполнить ручную трассировку работы средств управления ОП и свопинга
2. Заполнить трассировочные таблицы:
   а) без учета времени загрузки/выгрузки;
   б) указать корректировки при учёте времени загрузки/выгрузки
3. Выполнить анализ накладных расходов на свопинг процессов
4. Сформулировать преимущества и недостатки свопинга процессов

Пример выполнения

Размер ОП – 70. Время загрузки/выгрузки процесса – 10.

Характеристики выполняемых процессов

Номер	Время поступления	Время выполнения	Размер в ОП	Приоритет
1	0	100	50	1
2	10	100	40	3 абс.
3	20	100	20	2

Пример заполнения трассировочной таблицы

Время	Состояние ОП	Состояние ВП	Примечание
0	Процесс: П1 (акт.) / Своб. Начало: 0 / 30 Длина: 50 / 40		
10	Процесс: Своб. Начало: 0 Длина: 70	П1 (90)	Выгрузка П1
10	Процесс: П2 (акт.) / Своб. Начало: 0 / 40 Длина: 40 / 30	П1(90)	
20	Процесс: П2 (акт.) / П3 / Своб. Начало: 0 / 40 / 60 Длина: 40 / 20 / 10	П1(90)	
110	Процесс: Своб. / П3 / Своб. Начало: 0 / 40 / 60 Длина: 40 / 20 / 10	П1(90)	Завершение П2
110	Процесс: П3 (акт.) / П1 Начало: 0 / 20 Длина: 20 / 50		Подкачка П1
210	…		

Варианты заданий – Приложение 1.5.

Контрольные вопросы

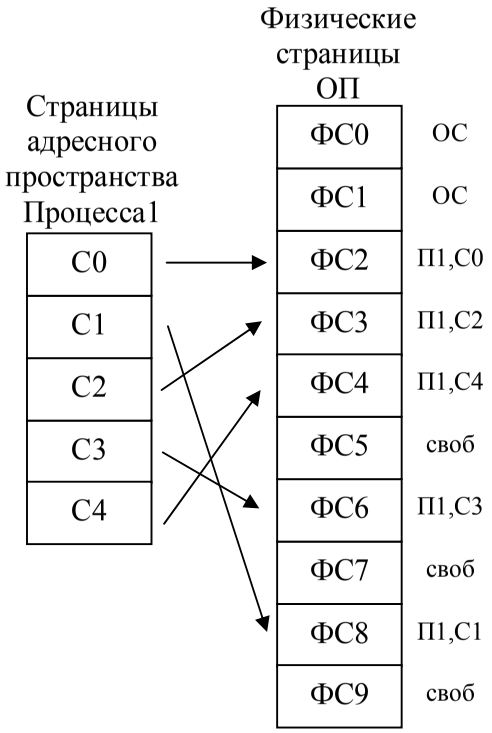
1. В чём заключается свопинг процессов?
2. Для чего применяют свопинг процессов?
3. В каких случаях процесс вытесняется во внешнюю память?
4. Как влияет свопинг на производительность ВС?

Занятие 2.3. Страничная память

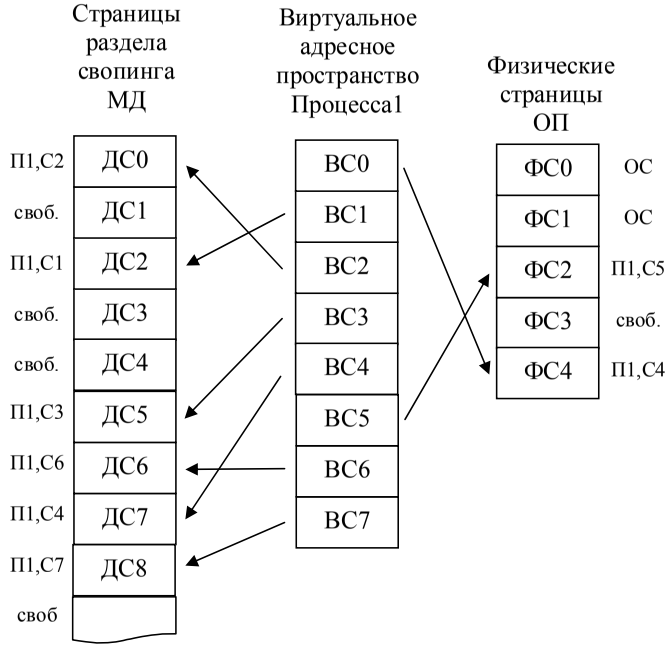
Цель занятия: освоить основы организации виртуальной страничной памяти, оценить эффективность страничного управления памятью

Краткое изложение теоретического материала

Страничная память основана на несвязном выделении ОП процессу. Оперативная память компьютера разбивается на участки фиксированной длины, именуемые *страницами*. Адресное пространство процесса также разбивается на страницы той же самой длины. Каждая из страниц процесса может быть размещена в произвольной странице оперативной памяти:

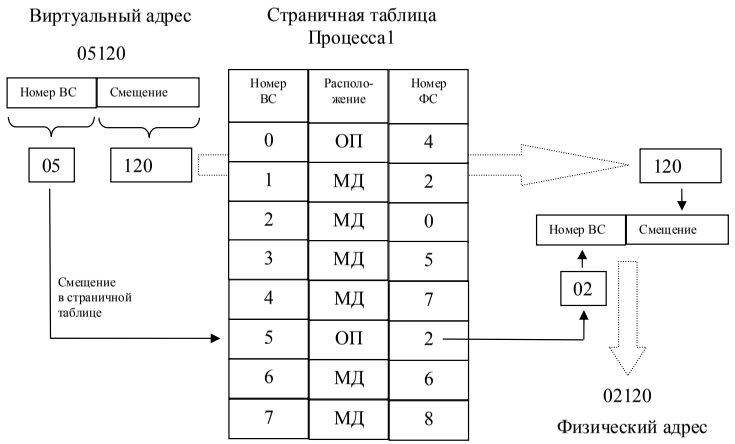


Кроме того, применяется концепция *виртуальной памяти*. Виртуальная (моделируемая ОС) память процесса может превышать размеры физической ОП и ограничена лишь суммой объёмов ОП и внешней (дисковой) памяти компьютера:



Для обеспечения виртуальной памяти было решено отказаться от принципа полного размещения процесса в ОП. В ОП загружается лишь начальная страница (несколько начальных страниц) процесса; остальные страницы подкачиваются по мере обращения к ним в процессе свопинга страниц. Свопинг обеспечивает *замещение страниц* процессов, если свободная страница ОП отсутствует.

Возникает проблема отображения адресов. При каждом обращении к ОП требуется вычислить физический адрес памяти по заданному виртуальному адресу. Указанная проблема решается с помощью набора специальных *страничных таблиц*, обеспечивающих отображение:



Множества свободных страниц оперативной и внешней памяти описываются аналогичными таблицами:

Таблица страниц МД

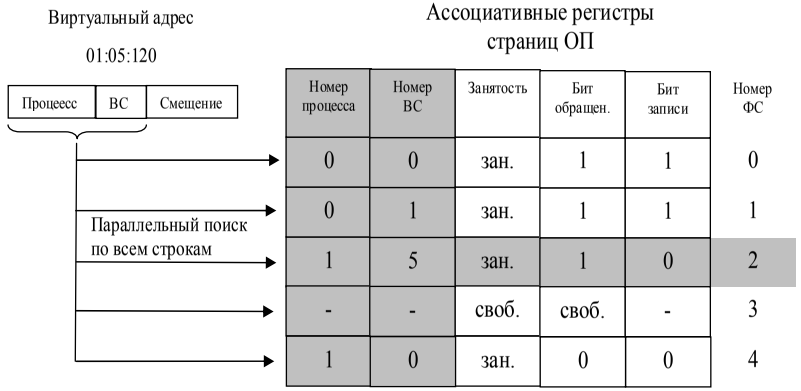
Номер страницы	Занятость	Номер процесса	Номер ВС
0	зан.	1	4
1	своб.	-	-
2	зан.	1	1
3	своб.	-	-
4	своб.	-	-
5	зан.	1	3
6	зан.	1	6
7	зан.	1	4
8	зан.	1	7

Таблица страниц ОП

Номер страницы	Занятость	Номер процесса	Номер ВС
0	зан.	0	0
1	зан.	0	1
2	зан.	1	4
3	своб.	-	-
4	зан.	1	0

При реализации свопинга встаёт вопрос выбора страницы процесса для вытеснения в случае, если все физические страницы ОП заняты. Наиболее распространённым является алгоритм LRU *вытеснения страницы*, которая дольше всего не использовалась. Непосредственная реализация алгоритма требует хранения значения таймера последнего обращения к странице, однако практически применяются более простые модификации, использующие лишь несколько битов информации.

Следует отметить, что страничная память может быть реализована ОС только в случае специальной архитектуры ВС и, в особенности, ЦП. Для быстрого аппаратного отображения адресов используются *страничные регистры*, обеспечивающие параллельный *ассоциативный поиск* требуемой страницы и последующее вычисление физического адреса:



Свопинг страниц инициируется *страничным прерыванием*, которое генерируется процессором в том случае, если требуемая страница отсутствует в ОП. Для экономной реализации алгоритма LRU, как правило, применяют пару бит страничных регистров, которые модифицируются автоматически: бит обращения A и бит записи W. Биты AW всех страниц периодически сбрасываются; затем биты некоторых страниц устанавливаются процессором во время выполнении процессов. В первую очередь вытесняется страница со значениями битов 00, если такие страницы отсутствуют, то с 01, иначе любая из страниц со значениями битов 11. Операция вытеснения страницы с битами 00 не требует её записи во внешнюю память и является наиболее быстрой.

При реализации страничной памяти важным является также вопрос *выбора размера страницы*. Малая длина страницы приводит к увеличению частоты свопинга, большая – к увеличению процента хранимого в ОП, но неиспользуемого кода. Реальная длина страниц колеблется от 4 до 16 Кбайт.

Современные ОС использует более сложную сегментно-страничную организацию памяти. Процесс разбивают на несколько логических частей переменной длины, именуемых сегментами. Каждый из сегментов разбивается на страницы фиксированной длины.

Используются двухуровневые таблицы сегментов и их страниц.

Задание

Выполнить ручную трассировку средств управления виртуальной памятью. Заполнить трассировочную таблицу. Оценить эффективность управления памятью.

Характеристики ОС: страничная память, LRU, мультипрограммирование

Порядок выполнения

1. Выполнить ручную трассировку работы страничной виртуальной памяти
2. Заполнить трассировочные таблицы
3. Выполнить анализ эффективности свопинга страниц
4. Сформулировать преимущества и недостатки страничного управления памятью

Пример выполнения

Количество страниц ОП – 3
Количество страниц ВнП – 11

Характеристики выполняемых процессов

Номер	Время поступления	Количество страниц	Последовательность действий
1	0	2	0(30)-1(20)-0(20)
2	20	2	0(20)-1(20)-0(30)
3	30	3	0(20)-1(20)-2(20)-0(20)

Пример заполнения трассировочной таблицы

t	Страницы ОП			Страницы ВнП			Пр.
0	ФС	Процесс	ВС	ФС	Процесс	ВС	
	0	П1	0(30)а	0			
	1			1			
	2			…	…	…	

20	ФС	Процесс	ВС	ФС	Процесс	ВС	
	0	П1	0(20)	0			
	1	П2	0(20)a	1			
	2			…	…	…	
30	ФС	Процесс	ВС	ФС	Процесс	ВС	
	0	П1	0(20)	0			
	1	П2	0(10)	1			
	2	П3	0(20)a	…	…	…	
50	ФС	Процесс	ВС	ФС	Процесс	ВС	A
	0			0	П1	0(20)	
	1	П2	0(10)	1			
	2	П3	0(20)a	…	…	…	
50	ФС	Процесс	ВС	ФС	Процесс	ВС	B
	0	П3	1(20)a	0	П1	0(20)	
	1	П2	0(10)	1			
	2	П3	0	…	…	…	
70	ФС	Процесс	ВС	ФС	Процесс	ВС	C
	0	П3	1	0	П1	0(20)	
	1			1	П2	0(10)	
	2	П3	0	2	…	…	
70	ФС	Процесс	ВС	ФС	Процесс	ВС	D
	0	П3	1	0	П1	0(20)	
	1	П3	2(20)a	1	П2	0(10)	
	2	П3	0	2	…	…	
90	…			…			…

Обозначения примечаний:

A - Запрос П3 ВС1, выгрузка П1 ВС0;
B - Загрузка П3 ВС1;
C Запрос П3 ВС2, выгрузка П2 ВС;
D - Загрузка П3 ВС2

Варианты заданий – Приложение 1.6.

Контрольные вопросы

1. В чём состоит концепция виртуальной памяти?
2. Что такое страница ОП?
3. Где могут размещаться страницы процесса?
4. Каким образом выполняется отображение страниц?
5. Каким образом ОС учитывает свободные и занятые страницы?
6. Для чего используется свопинг страниц?
7. Каким образом ОС выбирает страницу для вытеснения?
8. Каковы требования к аппаратным средствам ВС для организации страничной памяти?
9. Что такое сегмент?
10. Какие алгоритмы вытеснения страниц применяются в ОС?
11. В чем состоит идея LRU алгоритма?
12. Каковы требования к аппаратным средствам ВС для реализации LRU алгоритма вытеснения страниц?
13. Приведите пример сегментной таблицы.
14. Приведите пример страничной таблицы.

Раздел 3. Управление устройствами

Фактическое выполнение операций ввода-вывода для внешних устройств реализуется специальными моду-лями ОС, именуемыми *драйверами*. Ввиду существенных отличий внешних устройств каждый из типов внешних устройств, подключенных к ВС, имеет свой собственный драйвер в составе ОС. Кроме того, значительно отличаются драйверы устройств с блочным (МД) либо байтовым (клавиатура) вводом-выводом. В своей работе драйверы используют физические регистры контроллеров внешних устройств либо канальные программы (при подключении устройств посредством каналов), а также аппаратные прерывания внешних устройств. Драйверы, как правило, имеют несколько точек входа и одна из них – по аппаратному прерыванию. Подробное изучение организации драйверов выходит за рамки настоящего пособия.

В базе данных ОС каждое устройство представлено специальным блоком управления, в котором хранится его описание. Кроме того, создаются блоки управления контроллером (каналом), которые содержат указатели на блоки управления подключенных к ним устройств и рабочие данные текущей операции.

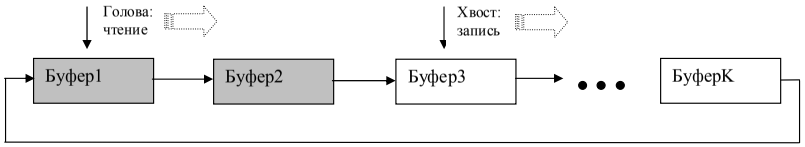
В настоящем разделе изучаются наиболее распространённые методы, используемые большинством модулей управления устройствами, такие как буферизация, кэширование, планирование операций, а также особенности организации взаимодействия ВС в сети.

Занятие 3.1. Циклическая буферизация

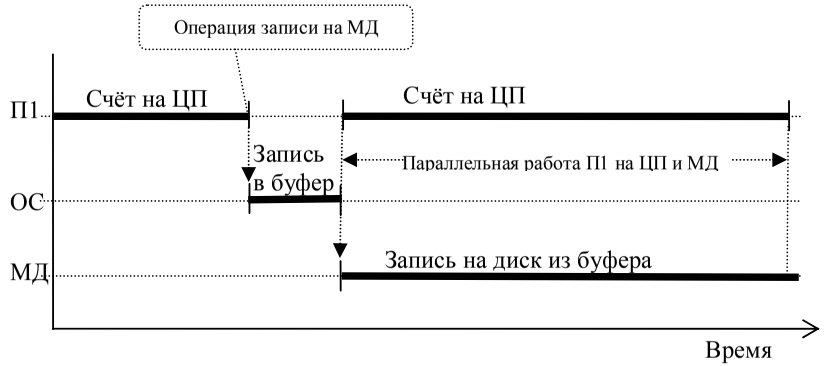
Цель занятия: освоить основы буферизации ввода-вывода, оценить эффективность применения буферизации

Краткое изложение теоретического материала

Буферизация является стандартным средством современных ОС для согласования скоростей работы устройств ВС. *Буфер* представляет собой участок памяти для промежуточного хранения данных. Различают буферы постоянной и переменной длины. Отдельные буферы объединяются в связные списки. Наиболее распространённой является *циклическая буферизация*:



Согласование скоростей обеспечивается, например, при выводе на магнитный диск, за счёт того, что после быстрого копирования ОС выводимых данных в буфер процесс считает операцию завершённой и продолжает своё выполнение. Таким образом, фактический вывод данных на устройство выполняется *параллельно с работой процесса*:



Достаточно большой размер буфера позволяет значительно ускорить выполнение процесса. В том случае, если записанные в буфер данные считываются до их фактической записи на диск, ОС обеспечивает их быстрое

извлечение из буфера. Следует отметить, что интенсивный ввод-вывод может привести к заполнению буферов и неизбежному ожиданию их освобождения, но в среднем буферизация существенно повышает эффективность работы ВС. Для *оценки эффективности* можно использовать сравнение времён работы процесса без буферизации и с использованием буферизации.

К *недостаткам буферизации* можно отнести возможность потери данных при крахах ОС; в современных надёжных ОС при использовании бесперебойного питания ВС вероятность такой ситуации сведена к минимуму. Тем не менее, большинство современных ОС предусматрива-ют операцию принудительной записи содержимого буферов.

Для согласования скоростей и ускорения работы внешних устройств ВС используют также *кэширование*. В этом случае блоки данных размещаются в промежуточном высокоскоростном хранилище (кэше) несвязно; обеспечивается быстрый ассоциативный поиск по ключевой информации, например, номерам логических (физических) блоков внешних устройств.

Задание

Выполнить ручную трассировку работы средств управления внешними устройствами. Заполнить трассировочную таблицу.

Характеристики ОС: циклическая буферизация, мультипрограммирование

Порядок выполнения

1. Выполнить ручную трассировку работы средств буферизации
2. Заполнить трассировочные таблицы
3. Выполнить анализ эффективности буферизации
4. Сформулировать преимущества и недостатки буферизации

Пример выполнения

Размер буфера – 3 блока
Время вывода одного блока на устройство – 10
Последовательность записи блоков процессом (имя блока - задержка)
A-5-B-5-C-5-D-5-E-30-F-5-G

Пример заполнения трассировочной таблицы

Время	Состояние процесса	Состояние буфера			Состояние устройства
		0	1	2	
0	Запись A	A (голова)	(хвост)		Вывод A (10)
5	Запись B	A (голова)	B	(хвост)	Вывод A (5)
10	Запись C	(хвост)	B (голова)	C	Вывод B (10)
15	Запись D	D	B (гол. и хв.)	C	Вывод B (5)
20	Запись E	D	E	C (гол. и хв.)	Вывод C (10)
30	Счёт	D (голова)	E	(хвост)	Вывод D (10)
40	Счёт		E (голова)	(хвост)	Вывод E (10)
50	…	…	…	…	…

Варианты заданий – Приложение 1.7.

Контрольные вопросы

1. Для чего применяют буферизацию?
2. В чём состоит циклическая буферизация?
3. За счёт чего обеспечивается согласование скоростей работы устройств при буферизации?
4. В чём состоят различия буферизации и кэширования?

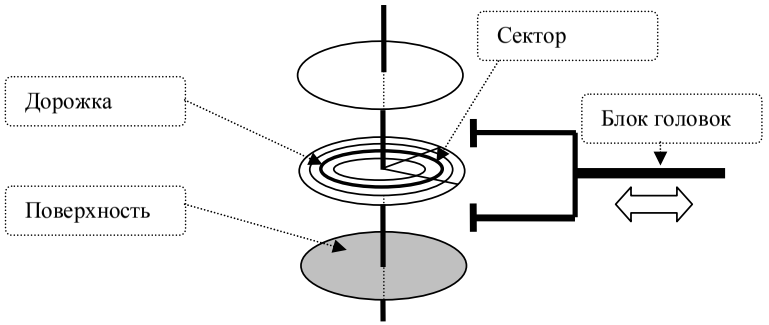
Занятие 3.2. Планирование дисковых операций

Цель занятия: освоить основы планирования дисковых операций, оценить эффективность применения планирования дисковых операций

Краткое изложение теоретического материала

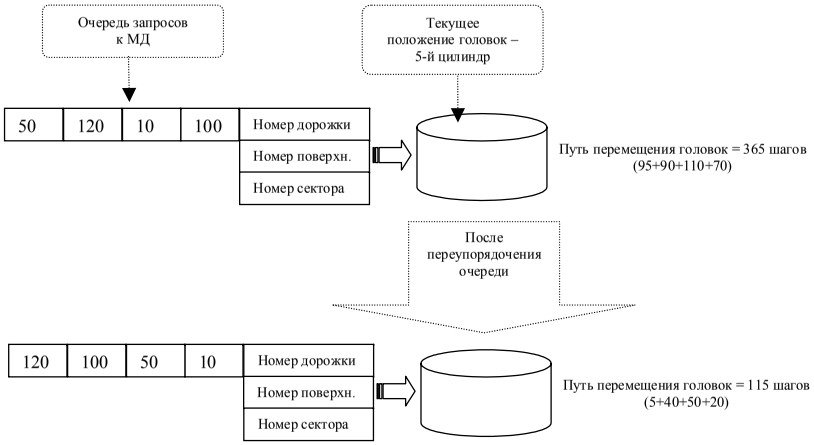
Планирование является наиболее сложной операцией ОС. Большинство ОС ограничиваются реализацией лишь операций выделения, освобождения и отслеживания своих ресурсов. Однако, использование операций планирования может существенно повысить производительность ВС. Изучим средства планирования ОС на примере планирования дисковых операций.

Магнитный диск представляет собой сложное электронно-механическое устройство. На общей оси собраны несколько дисков с магнитными поверхностями; как правило, внешние поверхности нижнего и верхнего дисков не используются. Поверхности дисков разбиты на дорожки в форме концентрических окружностей. Доступ к данным дорожек обеспечивает блок магнитных головок, перемещаемый между дорожками шаговым двигателем. Каждая дорожка разбита на секторы равной длины; сектор является минимальной адресуемой единицей магнитного диска и содержит блок данных. Множество равноудалённых от оси дорожек всех поверхностей именуют цилиндром. Таким образом, физический адрес сектора (блока данных) состоит из номера цилиндра, номера поверхности и номера сектора:



Быстрое вращение диска (1000 об/с) обеспечивает сравнительно высокую скорость чтения-записи для всех секторов текущего цилиндра. Операция перемещения блока головок к другому цилиндру требует включения шагового двигателя и занимает сравнительно большее (10-20 раз) время.

Запросы чтения-записи магнитного диска размещаются ОС в очередь. При отсутствии планирования они выполняются в порядке поступления. Заметим, что такой способ реализации запросов в большинстве случаев является неэффективным, так как приводит к частым перемещениям блока головок. *Переупорядочение очереди* запросов по критерию минимизации суммарного пути перемещения головок позволяет *минимизировать общее время выполнения всех запросов*. Учитывая разницу времён чтения-записи и перемещения головок, можно прийти к выводу, что повышение производительности является существенным:



Заметим, что возможно также дополнительное планирование номеров секторов текущего цилиндра, но оно редко используется ввиду сравнительно высокой скорости вращения диска.

Недостатком описанного простого подхода к планированию является возможное *увеличение времени ввода-вывода отдельных процессов*. Такое увеличение нежелательно для процессов реального времени. Таким образом, при использовании приоритетных дисциплин требуются применение более сложных критериев планирования.

Задание

Выполнить ручную трассировку работы планировщика очереди запросов к магнитному диску. Заполнить трассировочную таблицу.

Характеристики ОС: планирование дисковых операций, мультипрограммирование

Порядок выполнения

1. Выполнить ручную трассировку работы средств планирования дисковых операций

2. Заполнить трассировочные таблицы
3. Выполнить анализ эффективности планирования дисковых операций
4. Сформулировать преимущества и недостатки планирования дисковых операций

Пример выполнения

Время перемещения головки на 1 цилиндр – 1. Время записи – 10.

Последовательность операций ввода/вывода

Время	0	20	40	60	80	100	120
Цилиндр	100	150	120	110	200	10	160

Пример заполнения трассировочной таблицы

Время	Очередь к МД	Операция МД
0		Движение к 100 (100)
20	150	Движение к 100 (80)
40	150, 120	Движение к 100 (60)
60	150, 120, 110	Движение к 100 (40)
80	200, 150, 120, 110	Движение к 100 (20)
100	10, 200, 150, 110	Запись 100 (10)
110	10, 200, 150	Движение к 110 (10)
120	10,200, 160, 150	Запись 110 (10)
130	…	…

Варианты заданий – Приложение 1.8.

Контрольные вопросы

1. Для чего необходимо планирование операций МД?
2. Из чего состоит физический адрес данных на МД?
3. Каким образом выполняется переупорядочение очереди к МД?
4. Как влияет планирование операций МД на производительность ВС?

Занятие 3.3. Взаимодействие компьютеров в сети

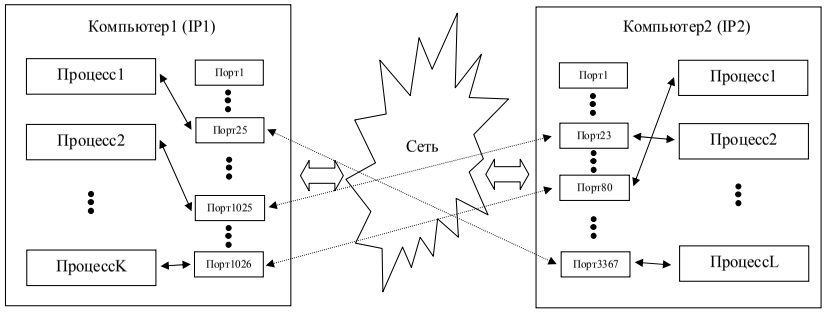
Цель занятия: освоить основы организации взаимодействия компьютеров в сети на основе технологии клиент-сервер

Краткое изложение теоретического материала

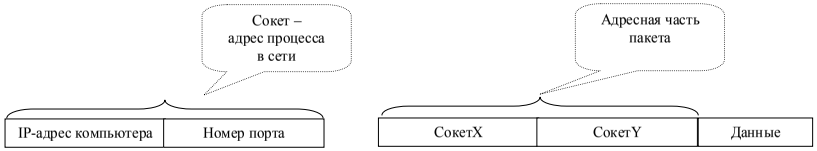
В настоящее время доминирующим *семейством сетевых протоколов* является TCP/IP; стек протоколов TCP/IP реализован во всех современных ОС. Передача информации в сети основана на взаимодействии при-кладных процессов на основе *технологии клиент-сервер*. Клиент инициирует взаимодействие и обращается с запросом на обслуживание к некоторому серверу; сервер выполняет запрос и передаёт результаты клиенту. Клиентами и серверами являются процессы операционной системы.

Таким образом, возникает *задача адресации процессов в сети*. В сети TCP/IP адресом процесса является выделенный ему *сокет* (гнездо). Сокет состоит из IP-адреса компьютера и номера порта. *Порт* является абстрактным объектом и служит для организации взаимодействия. В семействе протоколов TCP/IP порты с номерами 0-1024 зарезервированы и служат для адресации стандартных прикладных серверов, например, порт 25 – сервер электронной почты, протокол SMTP; порт 23 –

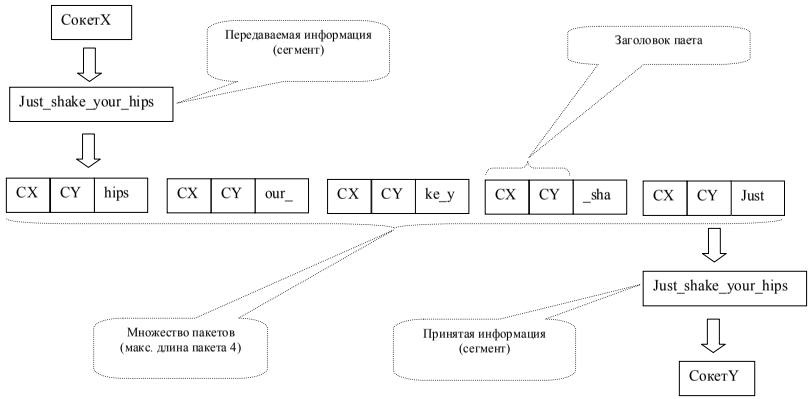
служба удаленного управления, протокол эмуляции терминала telnet; порт 80 – сервер передачи гипертекстовой (WWW) информации, протокол HTTP. При запуске сервер занимает соответствующий порт и «слушает» его. Соответствующему клиенту выделяется порт со случайным номером, превышающим 1024:



Адресную часть пакетов, передаваемых в сети, можно представить как пару: сокет отправителя, сокет получателя. Таким образом, обеспечивается полная адресация сетевых процессов, обеспечивающая доставку и обработку информации.



Кроме того, сеть ограничивает максимальную длину пакета. Поэтому передаваемая информация разбивается на множество пакетов, а после получения собирается из принятых пакетов. Более сложные аспекты *фрагментации*, связанные с возможным изменением порядка доставки пакетов не рассматриваются в настоящей работе.



Заметим, что физически компьютер должен иметь хотя бы один сетевой интерфейс, с которым ассоциирован IP-адрес. Таким интерфейсом может быть Ethernet адаптер при использовании локальной сети, модем, подключенный к интерфейсу USB (RS232), при удалённом доступе.

Задание

Выполнить ручную трассировку процессов взаимодействия компьютеров в сети. Заполнить трассировочную таблицу.

Характеристики ОС: технология «клиент-сервер», мульти-программирование

Порядок выполнения

1. Выполнить ручную трассировку работы сетевых средств ОС
2. Заполнить трассировочные таблицы
3. Оценить дополнительный объем информации заголовков пакетов
4. Сформулировать основные принципы адресации процессов в сети

Пример выполнения

Время выполнения запроса сервером – 40
Сокет сервера (8,21)
Время передачи пакета в сети – 40
Размер пакета – 5. Время формирования пакета – 1.
Клиент 1: сокет (5,6); команда «get f1» – прочесть файл f1
Клиент 2: сокет (2,3); команда «get f2» – прочесть файл f2
F1: “Extremes meet”
F2: “Practice make perfect”

Структура трассировочной таблицы

Время	Клиент 1	Клиент 2	Сеть			Сервер
			Отправ.	Получ.	Данные	
0	“get f1”		(5,6)	(8,21)	“get f”	
1			(5,6)	(8,21)	“1”	
20		“get f2”	(2,3)	(8,21)	“get f”	
21			(2,3)	(8,21)	“2”	
40						Получение “getf”
41						Получение “get f1” и выполнение запроса
60						Получение “get f”
61						Получение “get f2” и Выполнение запроса
81			(8,21)	(5,6)	“Extre”	Начало передачи f1
82			(8,21)	(5,6)	“mes m”	
83	…	…	…	…	…	…

Варианты заданий – Приложение 1.9.

Контрольные вопросы

1. Из чего состоит адрес процесса в сети?
2. Каким образом распределяются порты компьютера?
3. Что такое сокет?
4. В чём заключается технология клиент-сервер?
5. Из чего состоит пакет информации, передаваемый в сети?

Раздел 4. Управление информацией

Подсистема управления устройствами (также как и подсистема управления ОП) является защищённой и скрытой от конечного пользователя и прикладных процессов в современных ОС. Пользователю предоставляются средства управления информацией ОС с единицей хранения – *файл*. Следует отметить также тенденцию представления устройств в виде специальных файлов (например, в ОС Unix) для обеспечения ограниченного (и безопасного) доступа пользователей.

В базе данных ОС объекты файловой системы представлены блоками управления томом, каталогом, файлом, организованные в многосвязные списки. Заметим, что ОС обеспечивает «горячую» защиту информации на фактически подключенных устройствах. Безопасность в случае возможной несанкционированной передачи устройства (диска) злоумышленникам может быть обеспечена только при использовании дополнительных возможностей шифрования хранимой информации.

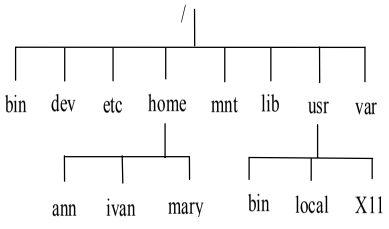
Занятие 4.1. Файловая структура диска

Цель занятия: освоить основы организации управления информацией, выполнить сравнительную оценку различных способов организации файловой структуры диска

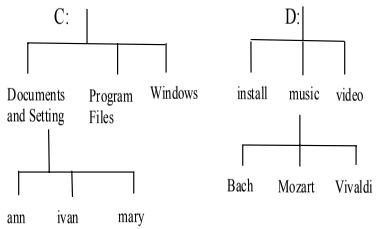
Краткое изложение теоретического материала

Древовидная (иерархическая) файловая система, образованная такими элементами как *том, каталог, файл,* является стандартной для современных ОС. Файл представляет собой поименованную единицу хранения информации; файлы объединяются в каталоги, причём каталог может содержать как файлы, так и другие каталоги; том представляет собой устройство с файловой системой. Единственная разница между семействами доминирующих

в настоящее время ОС Unix и MS Windows состоит в представлении томов: Unix использует общую иерархию всех томов, MS Windows представляет файловую систему с разбивкой по томам (устройствам):



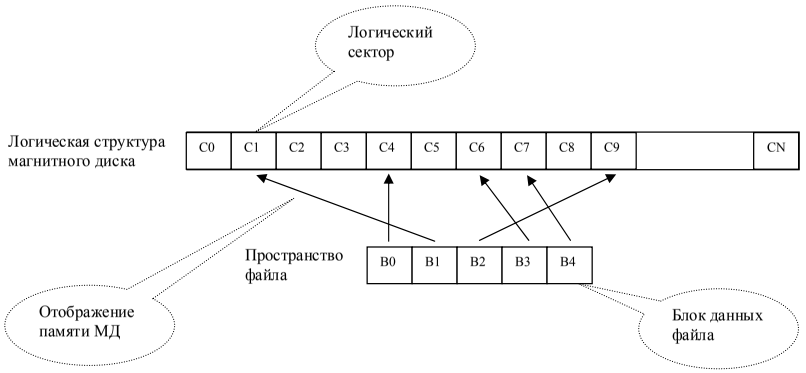
Unix



MS Windows

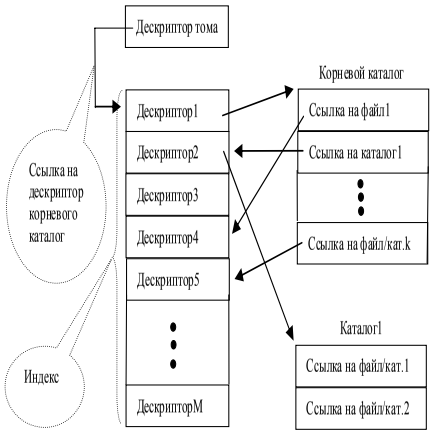
Как правило, современные ОС реализуют такие основные файловые операции как чтение-запись и позиционирование внутри файла, обеспечивая *последовательный и прямой доступ* к информации. Более сложные методы доступа реализуются СУБД на основе указанных операций ОС.

Логическая структура диска может быть представлена одномерным массивом секторов образованным занумерованными физическими секторами всех поверхностей, цилиндров и дорожек. Возникает *проблема отображения* файловой системы на логическую структуру диска для обеспечения доступа к файлам:

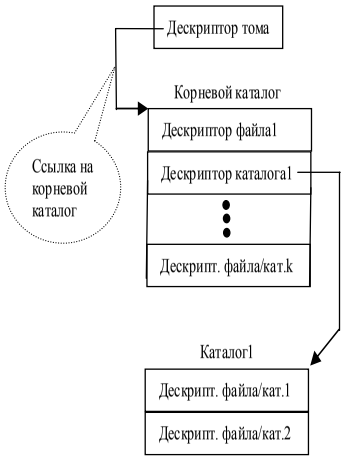


Для обеспечения отображения создаются специальные структуры данных, описывающие отображение и невидимые для конечного пользователя. Способ организации таких данных принято называть *файловой структурой*. Файловая структура обеспечивает решение двух основных задач: нахождение логических секторов заданного файла/каталога на диске; учёт свободных/занятых секторов.

Большинство файловых структур использует специальный блок – *дескриптор* для каждого из своих объектов: том, каталог, файл. Дескриптор файла/каталога содержит имя, владельца, права доступа, даты создания/корректировки, длину. Дескриптор тома содержит также указатель на дескриптор корневого каталога. Дескрипторы файла/каталога имеют фиксированную длину. ОС используют два основных похода к размещению дескрипторов: в ОС семейства Unix дескрипторы файлов/ каталогов размещаются в общем одномерном массиве – *индексе*, в этом случае каталог содержит список указателей на дескрипторы его файлов/каталогов; в ОС семейства MS Windows дескрипторы файлов/каталогов размещаются внутри содержащего их каталога:



Unix



MS Windows

При решении задачи отображения для уменьшения размера структур данных применяют *кластеризацию*. Кластер состоит из нескольких блоков (секторов) и является единицей выделения пространства диска. В настоящее время доминируют два подхода к описанию размещения файла на диске: использование общей таблицы размещения для всех кластеров (в ОС семейства MS Windows); использование массива описателей связных участков в дескрипторе файла (в ОС семейства Unix).

Общая таблица размещения содержит по одной записи для каждого кластера. Запись равняется нулю, если соответствующий кластер свободен. Если кластер принадлежит некоторому файлу, то соответствующая запись содержит номер следующего кластера; запись последнего кластера файла имеет специальное значение – признак конца цепочки (например, все двоичные единицы).

Использование *таблицы связных участков*, описанных указанием начального кластера участка и длины в дескрипторе файла, обеспечивает более быстрое отображение. Однако существенным недостатком этого метода является фиксированное число участков в дескрипторе файла. Размещение больших фрагментированных файлов решается в ОС Unix за счёт выделения дополнительных дескрипторов.

Таблица размещения файлов (File Allocation Table)

0	1	2	3	4	5	6	7	8	9
0	0	3	4	8	0	-	0	9	6

Таблица связных участков (в дескрипторе)

Нач.	2	8	6
Дл.	3	2	1

Использование единой таблицы размещения обеспечивает также решение задачи *учёта свободных/занятых кластеров*: записи свободных кластеров имеют нулевое значение в таблице. К дополнительным способам учёта свободного/занятого пространства можно отнести битовые карты и специальный (фиктивный) файл, занимающий все свободные кластеры. В битовой карте каждый бит соответствует кластеру; значение 0 – кластер свободен, значение 1 – занят. Применение специального файла обеспечивает преимущества быстрого поиска наиболее подходящего участка при выделении пространства и способствует уменьшению фрагментации.

Задание

Выполнить ручную трассировку работы файловой системы. Заполнить трассировочную таблицу.

I.

Характеристики ОС: несвязное выделение дискового пространства, таблица связных участков в дескрипторе файла, специальный файл учёта свободного пространства, мультипрограммирование

II.

Характеристики ОС: несвязное выделение дискового пространства, отдельная таблица размещения файлов, учёт свободного пространства в таблице размещения, мультипрограммирование

Порядок выполнения

1. Выполнить ручную трассировку работы сетевых средств ОС
2. Заполнить трассировочные таблицы
3. Оценить дополнительный объем информации заголовков пакетов
4. Сформулировать основные принципы адресации процессов в сети

Пример выполнения

Последовательность операций:
Создать файл 1. Записать 3 блока в файл 1. Создать файл 2. Записать 2 блока в файл 2. Дописать 1 блок в файл 1. Дописать 2 блока в файл 2. Удалить файл 1.
Доступное пространство диска – 10

I.

Пример заполнения трассировочной таблицы

Операция	Дескрипторы файлов		
	Файл 0	Файл 1	Файл 2
	A, (0,10)		
Создать файл 1	A, (0,10)	F1	
Записать 3 блока в файл 1	A, (3,7)	F1, (0,3)	
Создать файл 2	A, (3,7)	F1, (0,3)	F2
Записать 2 блока в файл 2	A, (5,5)	F1, (0,3)	F2, (3,2)
Дописать 1 блок в файл 1	A, (6,4)	F1, (0,3),(5,1)	F2, (3,2)
…	…	…	…

II.

Пример заполнения трассировочной таблицы

Операция	Код операции
Создать файл 1	O1
Записать 3 блока в файл 1	O2
Создать файл 2	O3
Записать 2 блока в файл 2	O4
Дописать 1 блок в файл 1	O5

Код операции	Дескрипторы файлов		
	Файл 0	Файл 1	Файл 2
	A, (0,10)		
O1	A, (0,10)	F1	
O2	A, (3,7)	F1, (0,3)	
O3	A, (3,7)	F1, (0,3)	F2
O4	A, (5,5)	F1, (0,3)	F2, (3,2)
O5	A, (6,4)	F1, (0,3),(5,1)	F2, (3,2)
…	…	…	…

Код операции	Дескрипторы файлов			Таблица размещения файлов							
	0	1	2	0	1	2	3	4	5	6	7
				0	0	0	0	0	0	0	0
O1	F1										
O2	F1, 0			1	2	-	0	0	0	0	0
O3	F1, 0	F2		1	2	-	0	0	0	0	0
O4	F1, 0	F2, 3		1	2	-	4	-	0	0	0
O5	F1, 0	F2, 3		1	2	5	4	-	-	0	0
…	…	…	…	..	..	..	..	..	..	..	..

Варианты заданий – Приложение 1.10.

Контрольные вопросы

1. Каковы основные элементы файловой системы?
2. Для чего необходима специальная файловая структура диска?
3. Какую информацию содержит дескриптор файла (тома, каталога)?
4. Как организуются дескрипторы для описания древовидной файловой системы?
5. Что такое кластер?
6. Каким образом описывается фактическое размещение файла на диске?
7. Какие средства используются для учёта свободного/занятого пространства диска?
8. Каким образом обеспечивается защита файловой системы?

Приложения

Приложение 1. Варианты заданий

П.1.1. Мультипрограммирование

Структура процессов:
З1: С1ЦП-D1МД-С2ЦП
З2: С3ЦП-L1МЛ-С4ЦП
З3: С5ЦП-D2МД-С6ЦП-D3МД-С7ЦП
З4: С3ЦП-L2МЛ-С4ЦП-D2МД-С1ЦП

Ci= (((n+i) mod 5)+1)*10
Di=(((n+i) mod 4)+1)*100
Li=(((n+i) mod 3)+1)*1000

n – номер студента в журнале

Вариант №29.

Таблица данных 1.

№	C	D	L
1	10	300	1000
2	20	400	2000
3	30	100	
4	40		
5	50		
6	10		
7	20		

Структура задания:
З1: 10ЦП-300МД-20ЦП
З2: 30ЦП-1000МЛ-40ЦП
З3: 50ЦП-400МД-10ЦП-100МД-20ЦП
З4: 30ЦП-2000МЛ-40ЦП-400МД-10ЦП

П.1.2. Циклическое квантование времени

Варианты – Приложение 1.1.
Размер кванта:

dt = ((n mod 2) + 1) * 10

Время переключения:

tos = ((n mod 4) + 2)

П.1.3. Приоритетные дисциплины

Варианты – Приложение 1.1.
Приоритеты заданий Зi:

Пi = ((n+i) mod 3) + 1

Приоритет 3 – абсолютный.

П.1.4. Динамические разделы

Размер ОП: V = ((n mod 4) + 7) * 10

Характеристики выполняемых процессов

Номер (i)	Время поступления (t_i)	Время выполнения	Размер в ОП	Приоритет
1	0	(((n+i) mod 4)+2) * 10	(((n+i) mod 3) + 3) * 10	((n+i) mod 3) + 1
2	t_{i-1}+(((n+i) mod 2) +1)* 10	…	…	…
3	…	…	…	…
4	…	…	…	…

П.1.5. Свопинг процессов

Варианты – Приложение 1.4.
Приоритет 3 – абсолютный.
Время загрузки/выгрузки: tio = ((n mod 2) + 1) * 10

П.1.6. Страничная память

Количество страниц ОП: NO = ((n mod 3) + 3
Количество страниц ВнП: NE = ((n mod 4) + 5

Характеристики выполняемых процессов

Номер (i)	Время поступ-ления (t_i)	Кол-во страниц	Последовательность действий
1	0	2	0(C1)-1(C2)-0(C3)
2	t_{i-1}+(((n+i) mod 2) +1)* 10	2	0(C4)-1(C5)-0(C6)
3	…	3	0(C7)-1(C1)-2(C2)-0(C3)
4	…	2	0(C6)-1(C4)-0(C2)

Ci из Приложения 1.1.

П.1.7. Циклическая буферизация

Размер буфера (блоков): BS = (n mod 3) + 2
Время вывода одного блока на устройство: TIO = ((n mod 2) + + 1) * 10
Последовательность записи блоков процессом (имя блока - задержка)
A-t1-B-t2-C-t3-D-t4-E-t5-F-t6-G

ti = ((n mod 4) + 1) * 5

П.1.8. Планирование дисковых операций

Время перемещения головки на 1 цилиндр: TC = (n mod 2) +1
Время записи: TIO = ((n mod 3) + 1) * 5

Последовательность операций ввода/вывода

Номер (i)	1	2	3	4	5	6	7	8
Время (t_i)	0	t_i+((n+ i) mod 3)+1)* 10	…	…	…	…	…	…
Цилиндр	((n+i)*100+i*10+(n+i)) mod 200	…	…	…	…	…	…	…

П.1.9. Взаимодействие компьютеров в сети

Время выполнения запроса сервером: TSer = ((n mod 3) + +2) * 10
Сокет сервера (n,21)
Время передачи пакета в сети: TNet = ((n mod 4) +1) * 10
Размер пакета: Psize = (n mod 3) + 4
Время формирования пакета: Pio = (n mod 2) + 2
Клиент 1: сокет (n+1,2*n); команда «get f1» – прочесть файл f1

Клиент 2: сокет (n+2,3*n); команда «get f2» – прочесть файл f2
F1: “Nothing succeeds like success”
F2: “The bait hides the hook”

П.1.10. Файловая структура диска

Последовательность операций:
Создать файл 1. Записать ((n mod 2) +2) блока в файл 1. Создать файл 2. Записать ((n mod 2)+1) блока в файл 2. Дописать ((n mod 2)+1) блока в файл 1. Дописать ((n mod 3) +1) блока в файл 2. Удалить файл 1.

Доступное пространство диска: V = (n mod 3) + 10

Приложение 2. Описание алгоритмов работы компонентов ОС

Алгоритмы планирования процессов.

Задание:
Разработать алгоритм модулей управления процессами в сетевой ОС.

Внешние спецификации.

1. Точки входа:
   - системные вызовы:
     `StartProcess(FileName, Priority): PID` – запустить процесс;
     `FinishProcess([PID])` – завершить процесс;
     `StartIO (IO_Parameters)` – начать ввод/вывод;
   - по прерыванию:
     `EndofIO(PID)` – завершение ввода/вывода;

`EndofQuant()` – завершение кванта времени;

- внутренние вызовы:

`Schedule()` – вызов диспетчера процессов.

2. Используемые процедуры:

- основные процедуры:

`StartDevice(IO_Parameters)` – запустить внешнее устройство;

`SetQuant()` – установить квант;

`SaveContext(aPCB)` – сохранить контекст процессора в блоке управления процессом с указанным адресом;

`LoadContext(aPCB)` – загрузить контекст процесса на процессор;

- вспомогательные процедуры:

`InQueue(Head, aPCB)` – поставить блок управления процессом с указанным адресом в очередь с указанным заголовком;

`OutQueue(Head, [PID]): aPCB` – извлечь блок управления процессом из указанной очереди.

Считаем, что указанные процедуры работают с очередями, упорядоченными по приоритету.

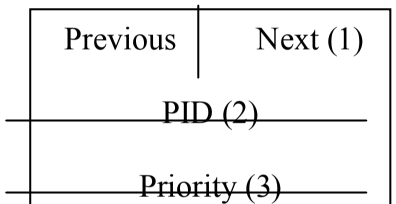
Приоритет – 0…255 (1 байт).

Приоритет 255 – абсолютный по отношению ко всем остальным.

Задача состоит в разработке алгоритмов для указанных точек входа.

Используемые структуры данных.

1. БУП – блок управления процессом (PCB).



1 – для организации связных списков (очередей);

2 – идентификатор процессов;

3 – приоритет;

4 – контекст процессора;

5 – имя файла;

6 – дополнительные параметры;

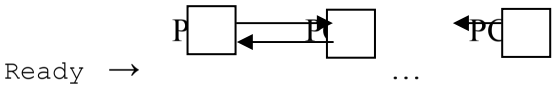
Обозначения:

`aPCB` – адрес PCB.

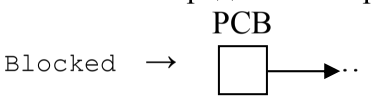
`aPCB → Priority` – поле приоритетов.

2. Заголовки очередей.

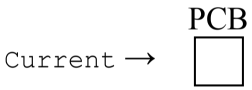
`Ready` – очередь готовых к исполнению процессов:



`Blocked` – очередь заблокированных процессов:



3. Указатель текущего процесса.



Примечания:

- указатель `Current` может быть пустым, если текущий процесс заблокирован/завершён;

- в системе имеется предопределённый процесс ленивец IDLE, priority = 0.

- если требуемое действие не указано в списке используемых процедур и не может быть описано явно на алгоритмическом языке, для его записи будем использовать форму комментариев:

; выделить требуемый объём ОП.

Алгоритмы.

```
1. StartProcess(FileName, Priority): PID
   begin
     SaveContext(Current);
     ; выделить память для БУП: → aPCB
     aPCB → priority := Priority;
     aPCB → PID := next PID;
     ; выделить ОП для процесса
     ; загрузить файл FileName в ОП
     aPCB → FileName:= FileName;
     ; установить начальное значение контекста
  процессора в БУП
     InQueue(aPCB);
     Schedule();
   end;

2. FinishProcess(PID)
   begin
     SaveContext(Current);
     if PID = 0 then
       begin
         aPCB := Current;
         Current := 0;
       end
     else aPCB := OutQueue(Ready, PID);
     ; освободить ОП aPCB
     ; освободить БУП aPCB
     Schedule();
   end;
```

3. Диспетчер процессов (Schedule) – универсальный выход из ядра. Его функция – выбор текущего процесса Current и установка контекста процесса Current на процессоре:

   - если Current = 0, выбрать новый процесс из очереди готовых;
   - если Current ≠ 0, продолжить выполнение текущего процесса, если нет готовых процессов с абсолютным приоритетом.

```
Schedule()
begin
if Current = 0
  then
    begin
      aPCB := OutQueue(Ready);
      Current := aPCB;
      SetQuant();
      LoadContext(Current);
    end;
  else
    begin
      aPCB := Ready;
      if aPCB → priority = 255
        then
          if Current → priority = 255
            then LoadContext(Current);
            else
              begin
                aPCB := OutQueue(Ready);
                InQueue(Ready, Current);
                Current := aPCB;
                SetQuant();
                LoadContext(Current);
              end;
    end;
end;

4. EndofQuant()
   begin
        InQueue(Ready, Current);
        Current := OutQueue(Ready);
        SetQuant();
        LoadContext(Current);
```

```
   end;

5. StartIO(IO_Parameters)
   begin
     InQueue(Blocked, Current);
     Current := 0;
     StartDevice(IO_Parameters);
     Schedule();
   End;

6. EndofIO(PID)
   begin
     aPCB := OutQueue(Blocked, PID);
     InQueue(Ready, aPCB);
     Schedule();
   end;
```


Список основной рекомендуемой литературы

1. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. – Питер, 2001. – 544 с.
2. Дейтел Г. Введение в операционные системы: В 2-х т. – М.: Мир, 1987. – 756 с.
3. Краковяк С. Основы организации и функционирования ЭВМ. – М.: Мир, 1988. – 480 с.
4. Мэдник С., Донован Дж. Операционные системы. – М.: Мир, 1978. – 640с.

Список дополнительной литературы

1. Иртегов Д. Введение в операционные системы. – BHV: Санкт-Петербург, 2002. – 624 с.
2. Столлингс В. Операционные системы. – Вильямс, 2002.– 848 с.
3. Партыка Т.Л., Попов И.И. Операционные системы, среды и оболочки. – Форум, 2003. – 400 с.
4. Гордеев А.В. Молчанов А.Ю Системное программное обеспечение. СПб Питер, 2001.
5. Робачевский А.М. ОС UNIX. BHV: С-Петербург, 2001.
6. Кулаков Ю.А., Омелянский С.В. Компьютерные сети. К.: Юниор,1999.
7. Шоу А. Логическое проектирование операционных систем. – М.: Мир, 1981. – 360 с.
8. Соловьев Г.Н. Операционные системы ЦВМ. – М.: Машиностроение, 1977. – 136 с.
9. Татцан Г. Операционные системы.- М.: Мир, 1976.- 471с.
10. Лихачева Г.Н., Медведев В.Д. Операционные системы. – М.: Статистика, 1980. – 231с.
11. Зелковиц М., Шоу А., Гэннон Дж. Принципы разработки программного обеспечения. – М.: Мир, 1982. – 368 с.
12. Коэн Л.Дж. Анализ и разработка операционных систем. – М.: Наука, 1975. – 190 с.
13. Кристиан К. Введение в операционную систему UNIX. – М.: Финансы и статистика, 1985. – 318 с.

14. Бирюков В.В., Рыбаков А.В., Шикура Ю.П. Введение в систему программирования ОС РВ. – М.: Финансы и статистика, 1986. – 192 с.
15. Кейслер С. Проектирование операционных систем для малых ЭВМ. – М.: Мир, 1986. – 680 с.
16. Методические указания к курсовому проектированию операционных систем (для студентов специальности 22.04) / Сост.: А.В.Григорьев, Д.А.Зайцев, А.И.Слепцов. – Донецк: ДГТУ, 1994. – 29 с.