Билет №1

1. Классификация ОС по способам распределения процессорного времени. Вытесняющие и не вытесняющие алгоритмы.

a) Вытесняющая многозадачность (preemptive multitasking) :

- Процесс может быть **прерван** планировщиком ОС в любой момент.
- Переключение задач происходит **по таймеру** или приоритету.
- Обеспечивает **справедливое распределение времени** , подходит для интерактивных задач.
- Примеры: Windows NT, UNIX, Linux.

b) Невытесняющая многозадачность (non-preemptive multitasking) :

- Процесс работает до тех пор, пока **сам не передаст управление** ОС.
- Зависит от корректной работы программ.
- Менее надежна — если программа зависает, система блокируется.
- Примеры: Windows 3.x.

c) Алгоритмы распределения процессорного времени :

- **Round Robin (циклический)** – каждому процессу выделяется фиксированный временной квант.
- **FCFS (First Come First Served)** – первый пришел — первый обслужен.
- **Приоритетное планирование** – процессы с высоким приоритетом получают время первыми.
- **SJF (Shortest Job First)** – сначала выполняются самые короткие задачи.

2. Логическая структура диска

Логическая структура диска определяет, как данные организованы и хранятся на устройстве. В системах DOS/Windows она включает:

a) Загрузочный сектор (Boot Sector) :

- Содержит код загрузчика и информацию о файловой системе.
- Первый сектор диска (LBA 0).

b) Таблица размещения файлов (FAT) :

- Указывает, какие кластеры заняты, какие свободны.
- Может быть FAT12, FAT16, FAT32.

c) Корневой каталог (Root Directory) :

- Хранит список файлов и каталогов верхнего уровня.
- Имеет фиксированную длину.

d) Область данных (Data Area) :

- Физическое место хранения файлов.
- Разделена на кластеры.

3. Создать командный файл `SUMMA.BAT`

@echo off

echo Объединение и переименование файлов

copy A:\D1\ANEW.TXT + A:\D1\BNEW.TXT A:\D2\CNEW.TXT

Объяснение:

- `@echo off` — отключает вывод команд на экран.
- `echo ...` — вывод текста на экран.
- `copy ... + ... ...` — объединяет два файла и сохраняет результат в третий.

Билет №2

1. Основные типы многозадачных ОС: пакетной обработки, разделения времени и реального времени.

a) ОС пакетной обработки :

- Задачи группируются в **пакеты** и выполняются последовательно.
- Цель: максимальная **пропускная способность** .
- Примеры: ранние ЭВМ.

b) ОС разделения времени (time-sharing) :

- Каждому пользователю предоставляется **временной квант** .
- Цель: быстрая реакция системы.
- Примеры: UNIX, Linux.

c) ОС реального времени (real-time OS) :

- Гарантированное время выполнения задач.
- Используется в автоматизации, авиации, медицине.
- Примеры: VxWorks, QNX.

2. Утилита `msconfig.exe`. Назначение и возможности.

Назначение :

`msconfig.exe` — это **системная утилита Windows** , предназначенная для **настройки параметров загрузки системы** , управления службами и автозапуском.

Основные вкладки и возможности :

a) Boot (Загрузка) :

- Настройка мультизагрузки.
- Параметры: количество процессоров, объем памяти, безопасный режим.

b) Services (Службы) :

- Отключение/включение системных и сторонних служб.
- Возможность скрыть Microsoft-службы.

c) Startup (Автозагрузка) :

- Управление программами автозапуска.

- Можно отключить ненужные программы для ускорения загрузки.

d) Tools (Инструменты) :

- Доступ к диагностическим инструментам Windows.

3. Создать командный файл `BATCH1.BAT`

@echo off

color 0a

echo Копирование и удаление файла

md DIR\DIR2

Объяснение:

- `@echo off` — отключает вывод команд на экран.
- `color 0a` — устанавливает цвет текста (черный фон, зеленый текст).
- `echo ...` — вывод текста на экран.

`md DIR\DIR2` — создает каталог `DIR`, а внутри него — `DIR2`.

Зачетный билет №3

1. Способы построения ядра ОС: монолитное ядро, микро ядро.

a) Монолитное ядро (monolithic kernel) :

- Все основные функции ОС (управление процессами, памятью, файловой системой, устройствами ввода-вывода) реализуются **в одном адресном пространстве** — в режиме ядра.
- Высокая производительность из-за отсутствия межпроцессного взаимодействия при вызовах функций.
- Менее устойчиво к сбоям: ошибка в любом модуле может повалить всю систему.
- Примеры: **Linux, UNIX, MS-DOS** .

b) Микроядро (microkernel) :

- В режиме ядра остаются только минимально необходимые функции:
- Планирование потоков.
- Обработка прерываний.
- Межпроцессное взаимодействие (IPC).
- Остальные компоненты (драйверы, файловая система, сетевые протоколы) работают в **пользовательском режиме** .
- Более надежная архитектура: ошибки в отдельных компонентах не влияют на работу ядра.
- Несколько медленнее из-за необходимости обмена сообщениями между пользовательским и ядерным режимом.
- Примеры: **MINIX, QNX, AmigaOS (v4), L4** .

c) Гибридное ядро (hybrid kernel) :

- Компромисс между монолитным и микроядерным подходами.

- Некоторые компоненты работают в ядерном режиме, другие — в пользовательском.
- Пример: **Windows NT (включая Windows 10/11)** .

2. Программа Regedit

`Regedit.exe` — это **редактор реестра Windows** , предназначенный для просмотра, редактирования, импорта и экспорта записей реестра.

Функциональные возможности:

- **Просмотр структуры реестра** :
  Реестр состоит из разделов:
- `HKEY_CLASSES_ROOT` – типы файлов и COM-объекты.
- `HKEY_CURRENT_USER` – настройки текущего пользователя.
- `HKEY_LOCAL_MACHINE` – параметры всей системы.
- `HKEY_USERS` – данные всех пользователей.
- `HKEY_CURRENT_CONFIG` – конфигурация оборудования.
- **Редактирование ключей и значений** :
- Создание новых ключей и параметров.
- Изменение значения существующих параметров.
- Удаление ненужных записей.
- **Импорт/экспорт разделов** :
- Сохранение части реестра в `.reg`-файл.
- Восстановление из файла.
- **Поиск по реестру** :
- Поиск ключей, подключей или значений.
- **Настройка автозапуска программ** :
- Редактирование раздела:
  `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
- **Диагностика и восстановление системы** :
- Настройка параметров безопасности, драйверов, служб и т.д.

⚠ **Важно:** Работа с реестром требует осторожности, так как некорректные изменения могут привести к нестабильной работе системы.

3. Создать командный файл `BATCH1.BAT`

@echo off

:: Отключение вывода команд на экран

color 0a

:: Установка цвета текста

cls

:: Очистка экрана

```
echo До выполнения операций:
dir TEXT1.TXT
:: Показываем, существует ли исходный файл

if not exist TEXT1.TXT (
 echo Файл TEXT1.TXT не найден в текущем каталоге!
 pause
 exit
)

copy TEXT1.TXT ..\TEXTNEW.TXT
:: Копируем файл в другой каталог
del TEXT1.TXT
:: Удаляем оригинал

echo.
echo Файл скопирован и удален
echo.
echo После выполнения операций:
dir ..\TEXTNEW.TXT
:: Проверяем наличие нового файла

pause
:: Пауза до нажатия клавиши
```

Объяснение:

- `@echo off` — скрывает выполнение команд.
- `color 0a` — зеленый текст на черном фоне.
- `cls` — очищает экран.
- `if not exist ...` — проверяет наличие файла перед удалением.
- `copy` — копирует файл.

- `del` — удаляет оригинальный файл.
- `dir` — показывает содержимое до и после операции.
- `pause` — ожидает нажатия клавиши.

Зачетный билет №4

1. Архитектура ядра ОС. Основные подсистемы ядра.

a) Ядро ОС (kernel) :

- Центральный компонент ОС, управляющий всеми ресурсами компьютера.
- Работает в **привилегированном режиме (kernel mode)** .
- Ядро обеспечивает:
- Управление процессами и потоками.
- Управление памятью.
- Управление вводом-выводом.
- Файловую систему.
- Безопасность и защиту.
- Поддержку сетевых соединений.

b) Основные подсистемы ядра :

Управление процессами	Создание, планирование, переключение контекста, завершение процессов
Управление памятью	Распределение и освобождение памяти, виртуализация, страничная/сегментная организация
Управление вводом-выводом	Работа с драйверами устройств, буферизация данных
Файловая система	Организация хранения данных, работа с дисками
Сеть	Обеспечение сетевого взаимодействия, маршрутизация, протоколы TCP/IP
Безопасность	Управление доступом, шифрование, контроль прав

c) Архитектуры ядра :

- **Монолитное ядро** — все компоненты в одном адресном пространстве.
- **Микроядро** — минимум функций в ядре, всё остальное в пользовательском режиме.
- **Гибридное ядро** — смесь двух предыдущих.

2. Назначение реестра

Реестр Windows — это центральная **база данных параметров и настроек** операционной системы и установленных программ.

Основное назначение:

- Хранение информации о:
- Конфигурации оборудования.
- Настройках ОС.
- Параметрах учетных записей.
- Информации о приложениях.
- Автозапуске программ.

Особенности:

- Доступ к реестру возможен через `regedit.exe`.
- Содержит **ключи (keys)** и **значения (values)** .
- Иерархическая структура напоминает файловую систему.
- Реестр используется для:
- Настройки интерфейса Windows.
- Управления службами и драйверами.
- Регистрации COM-объектов.
- Запуска программ при старте системы.

3. Диаграмма выполнения трех процессов P1, P2, P3 при невытесняющем алгоритме

Дано:

- P1 = 3
- P2 = 5
- P3 = 8

Диаграмма:

Время: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

P1 [И][И][И]

P2 [Г][Г][Г][И][И][И][И][И]

P3 [Г][Г][Г][Г][Г][Г][Г][Г][И][И][И][И][И][И][И][И]

Расчеты:

P1 0 3
P2 3 5
P3 8 8
Sw=0+3+8/3=3.67

Ss=3+8+16/3=9

Зачетный билет №5

1. Основные режимы процесса. Состояние процессов. Взаимосвязи между родительскими и порожденными процессами.

а) Основные режимы процесса:

- **Пользовательский режим (user mode)** :
- Процесс выполняет пользовательский код.
- Не имеет доступа к защищенным ресурсам напрямую.
- **Режим ядра (kernel mode)** :
- Процесс вызывает системные функции (например, ввод-вывод).
- Может обращаться ко всем ресурсам компьютера.

b) Состояния процессов:

Процесс может находиться в одном из следующих состояний:

Выполнение (Running)	Процесс исполняется на процессоре.
Готовность (Ready)	Процесс готов к выполнению, но ждет своего времени.
Ожидание (Waiting / Blocked)	Процесс ожидает завершения внешнего события (например, ввода-вывода).

Переходы между состояниями осуществляются с помощью планировщика и системных вызовов.

c) Взаимосвязь между родительскими и порожденными процессами

- **Родительский процесс** — это процесс, который создал другой процесс.
- **Дочерний процесс** — это процесс, созданный другим процессом.

Пример:

```
pid = fork(); // Создание дочернего процесса

if (pid == 0) {

// Это дочерний процесс

} else {

// Это родительский процесс

}
```

Особенности взаимосвязи:

- Родитель может управлять дочерним процессом (например, ожидать его завершения через `wait()`).
- При завершении дочернего процесса он отправляет сигнал родителю (`SIGCHLD`), чтобы освободить ресурсы.
- Дочерний процесс наследует:
- Копию памяти родителя (в момент `fork()`).
- Открытые файловые дескрипторы.
- Текущую директорию.

2. Файловые системы. Файловая система FAT.

a) Файловая система FAT (File Allocation Table)

FAT — одна из первых популярных файловых систем, использовавшихся в MS-DOS и ранних версиях Windows.

Версии FAT:

- **FAT12** — до 32 Мб.
- **FAT16** — до 2 Гб.
- **FAT32** — до 2 Тб.

Структура FAT:

1. **Загрузочный сектор (Boot Sector)** – содержит информацию о структуре тома и загрузчик.
2. **Таблица размещения файлов (FAT)** – указывает, какие кластеры заняты или свободны.
3. **Корневой каталог (Root Directory)** – список файлов верхнего уровня.
4. **Область данных (Data Area)** – место хранения содержимого файлов.

Преимущества:

- Простота реализации.
- Совместимость с большинством устройств (USB, камеры, навигаторы).

Недостатки:

- Ограниченный размер раздела (для FAT12/16).
- Нет поддержки прав доступа.
- Подвержена фрагментации.
- Нет журналирования.

3. Создать файл, ввести в файл текст из 6-ти строк. Убедиться, что файл создан; просмотреть его содержимое и права доступа.

Решение (в командной строке Windows):

@echo off

:: Создаем файл с 6 строками

echo Это строка 1 > myfile.txt

echo Это строка 2 >> myfile.txt

echo Это строка 3 >> myfile.txt

echo Это строка 4 >> myfile.txt

echo Это строка 5 >> myfile.txt

echo Это строка 6 >> myfile.txt

:: Проверяем, создан ли файл

dir myfile.txt

:: Выводим содержимое файла

type myfile.txt

:: Просматриваем атрибуты файла

attrib myfile.txt

Объяснение:

- `echo ... > myfile.txt` — создает файл и записывает первую строку.

- `>> myfile.txt` — добавляет строки в конец файла.
- `dir myfile.txt` — проверяет наличие и размер файла.
- `type myfile.txt` — выводит содержимое файла на экран.
- `attrib myfile.txt` — показывает атрибуты файла (архивный, скрытый, только чтение и т.д.).

Если нужно изменить атрибуты:

attrib +r myfile.txt :: Установить "только чтение"

attrib -r myfile.txt :: Снять "только чтение"

Зачетный билет №6

1. Синхронизация процессов. Понятие критической секции. Семафоры. Тупики.

a) Синхронизация процессов

Цель синхронизации — обеспечить корректное совместное использование общих ресурсов несколькими процессами или потоками.

b) Критическая секция

- Участок кода, в котором происходит **доступ к общему ресурсу** (например, переменной, файлу, устройству).
- В каждый момент времени **только один процесс** может находиться внутри критической секции.

Требования:

1. **Взаимное исключение** – не более одного процесса в критической секции.
2. **Прогресс** – если никто не находится в критической секции, то запрос должен быть удовлетворен.
3. **Ограничение времени ожидания** – процесс не должен бесконечно ждать входа в критическую секцию.

c) Семафоры

Семафор — это целочисленная переменная, которая используется для контроля доступа к общим ресурсам.

Основные операции:

- `wait(S)` – уменьшает значение семафора. Если S=0, процесс блокируется.
- `signal(S)` – увеличивает значение семафора и, возможно, возобновляет ожидание.

Пример:

semaphore mutex = 1;

process P1 {

wait(mutex);

// Критическая секция

signal(mutex);

}

process P2 {

wait(mutex);

// Критическая секция

signal(mutex);

}

d) Тупики (Deadlock)

Тупик — ситуация, при которой несколько процессов **ожидают ресурсов, занятых друг другом** , и не могут продолжить выполнение.

Условия возникновения:

1. **Взаимное исключение** – ресурс нельзя использовать одновременно.
2. **Удержание и ожидание** – процесс удерживает ресурсы и ожидает другие.
3. **Невозможность принудительного изъятия** – ресурсы можно освободить только добровольно.
4. **Циклическое ожидание** – цепочка процессов, где каждый ожидает ресурс, занятый следующим.

Методы предотвращения:

- Избегать циклического ожидания.
- Ограничивать количество используемых ресурсов.
- Использовать таймеры ожидания.

2. Создать командный файл, выполняющий действия:

- Объединение содержимого 2-х файлов из разных каталогов.
- Вывод содержимого нового файла на экран.
- Переименование исходных файлов.
- До и после каждого действия показать результат.

Пример командного файла:

@echo off

cls

echo === Начало работы ===

:: Проверка наличия файлов до объединения

echo [До] Содержимое файлов:

dir C:\folder1\file1.txt

dir D:\folder2\file2.txt

:: Объединение файлов

copy C:\folder1\file1.txt + D:\folder2\file2.txt combined.txt

echo.

:: Проверка результата объединения

echo [После] Содержимое combined.txt:

type combined.txt

echo.

:: Переименование исходных файлов

ren C:\folder1\file1.txt file1_old.txt

ren D:\folder2\file2.txt file2_old.txt

:: Проверка после переименования

echo [После] Список файлов:

dir C:\folder1\file1_old.txt

dir D:\folder2\file2_old.txt

pause

Объяснение:

- `copy A + B C` — объединяет два файла в третий.
- `ren` — команда переименования файлов.
- `dir` и `type` — используются для проверки.

3. Команды фильтры MS DOS / Примеры.

Фильтры — это команды, которые принимают данные из одного источника, обрабатывают их и выводят на другой.

Примеры фильтров:

`find`	Поиск строк по ключевому слову	`type log.txt
`findstr`	Расширенный поиск с регулярными выражениями	`findstr /i "warning" log.txt`
`sort`	Сортировка данных	`type names.txt
`more`	Постраничный вывод	`type bigfile.txt
`tee` (в UNIX)	Одновременный вывод на экран и в файл	`command

Пример объединения фильтров:

type data.txt | find "error" | sort > errors_sorted.txt

Эта команда:

1. Читает файл `data.txt`.
2. Выбирает строки с "error".
3. Сортирует их.
4. Сохраняет в `errors_sorted.txt`.

Зачетный билет №7

1. Управление памятью в ОС. Страничное распределение памяти

a) Страничная организация памяти

- Память делится на **равные блоки** — **страницы** .
- Программа также разбивается на части — **виртуальные страницы** .
- Физическая память делится на **рамки (frame)** — ячейки, равные по размеру страницам.

b) Как работает страничная система?

1. При обращении к адресу процессор использует **виртуальный адрес** , который состоит из:
- **номера виртуальной страницы**
- **смещения внутри страницы**
1. Номер виртуальной страницы используется как индекс в **таблице страниц** , где хранится номер соответствующей **физической рамки** .
2. Физический адрес вычисляется как:
3. 1
4. физический адрес = номер_физической_рамки × размер_страницы + смещение

c) Преимущества страничного распределения:

- Упрощает управление памятью.
- Минимизирует внешнюю фрагментацию.
- Поддерживает **виртуальную память** — возможность использовать диск как продолжение оперативной памяти.

d) Недостатки:

- Требуется дополнительная память для хранения таблицы страниц.
- Может быть внутренняя фрагментация (если последняя страница не заполнена полностью).

e) Реализация в реальных системах:

- Используется во многих современных ОС: **Windows, Linux, UNIX** .
- Аппаратная поддержка в процессорах Intel (например, **i386 и выше**) — регистры управления таблицами страниц.

2. Структура реестра

Реестр Windows — это **централизованное хранилище конфигурационной информации** об операционной системе, оборудовании, пользователях и приложениях.

a) Основные разделы реестра (ключи верхнего уровня):

`HKEY_CLASSES_ROOT`	Связь между типами файлов и программами, COM-объекты
`HKEY_CURRENT_USER`	Настройки текущего пользователя
`HKEY_LOCAL_MACHINE`	Конфигурация всей системы
`HKEY_USERS`	Настройки всех пользователей
`HKEY_CURRENT_CONFIG`	Информация о текущей аппаратной конфигурации

b) Ключи и значения

- **Ключи** — аналогичны папкам.
- **Значения (value)** — данные, связанные с ключом (DWORD, QWORD, строка, бинарные данные и др.)

c) Пример структуры:

HKEY_CURRENT_USER

└── Software

└── Microsoft

└── Windows

└── CurrentVersion

└── Run

└── myapp.exe = "C:\Program Files\MyApp\myapp.exe"

d) Изменение реестра:

- Через редактор реестра: `regedit.exe`
- Через командную строку: `reg add`, `reg delete`, `reg query`

3. Основные команды DOS, примеры использования

`dir`	Выводит список файлов и каталогов	`dir C:\`
`cd`	Переход в другой каталог	`cd folder`
`md` или `mkdir`	Создание каталога	`md newfolder`
`rd` или `rmdir`	Удаление каталога	`rd oldfolder`
`copy`	Копирование файлов	`copy file.txt newfile.txt`
`move`	Перемещение файла	`move file.txt D:\newfolder`
`del` или `erase`	Удаление файла	`del temp.txt`
`type`	Вывод содержимого файла	`type file.txt`
`ren` или `rename`	Переименование файла	`ren old.txt new.txt`
`find`	Поиск строки в файле	`type log.txt

`echo`	Вывод текста или запись в файл	`echo Hello > hello.txt`

Пример скрипта:

@echo off

echo Листинг текущего каталога:

dir

echo Копируем файл:

copy source.txt backup.txt

echo Удаляем исходный:

del source.txt

pause

Зачетный билет №8

1. Управление памятью в ОС. Сегментное распределение памяти

a) Что такое сегментное распределение?

- Программа делится на **логические блоки — сегменты** : код, данные, стек, куча и т. д.
- Каждый сегмент имеет имя и длину.
- Адресация: `(номер_сегмента, смещение)`.

b) Как происходит преобразование адреса?

- Используется **таблица сегментов** , которая содержит:
- Базовый адрес сегмента в физической памяти.
- Длину сегмента.
- Права доступа.
- При обращении к памяти:

1. Проверяется, чтобы смещение не выходило за границы длины сегмента.
2. Физический адрес = база_сегмента + смещение.

c) Преимущества:

- Удобство защиты и совместного использования данных.
- Гибкость в использовании памяти.

d) Недостатки:

- Сложность управления свободным пространством.
- Возможна **внешняя фрагментация** .
- Медленнее, чем страничная организация.

e) Сравнение со страничной организацией:

Размер блока	Разный	Фиксированный
Фрагментация	Внешняя	Внутренняя
Защита	На уровне	На уровне

	сегментов	страниц
Преобразование адреса	Таблица сегментов	Таблица страниц

2. Файловая система NTFS. Основные особенности NTFS

a) NTFS (New Technology File System) — основная файловая система Windows.

b) Основные особенности:

Длинные имена файлов	До 255 символов
Поддержка больших объемов	До 256 ТБ на один том
Права доступа (ACL)	Гибкая система разрешений на файлы и папки
Шифрование (EFS)	Шифрование файлов и папок
Журналирование (логирование)	Все изменения записываются в журнал для восстановления
Сжатие файлов	Поддержка сжатия без потерь
Самовосстановление	Автоматическое восстановление после сбоев
Символические ссылки	Поддержка точек монтирования и ссылок
MFT (Master File Table)	Центральная структура, где хранится информация обо всех файлах

c) MFT — Master File Table

- Это **главная таблица** , где каждая запись описывает:
- Имя файла/каталога.
- Расположение данных на диске.
- Права доступа.
- Даты создания, изменения, чтения.
- Каждому файлу или каталогу соответствует **запись в MFT** .
- Первая запись относится к самой таблице MFT.

- **Загрузочный сектор (Boot Sector)** – содержит информацию о структуре тома и загрузчик.
- **Master File Table (MFT)** – главная таблица файлов, где хранятся все данные о файлах и каталогах.
- **Площадка MFT (MFT Zone)** – область вокруг MFT, зарезервированная для ее роста.
- **Область данных** – место, где физически хранятся данные файлов.
- **Служебные файлы NTFS** – системные файлы, такие как $MFT, $LogFile, $Bitmap, $Volume и другие.

3. Открыть дисковый файл текстового формата. Вставить в его начало данные с другого файла.

a) Задача:

Добавить содержимое одного файла в начало другого файла.

b) Решение (в командной строке Windows):

```
@echo off

:: Проверяем наличие файлов

if not exist file1.txt (

 echo file1.txt не найден!

 exit /b

)

if not exist file2.txt (

 echo file2.txt не найден!

 exit /b

)


:: Создаем временный файл и добавляем содержимое file1 в начало file2

type file1.txt > temp.txt

type file2.txt >> temp.txt


:: Перезаписываем file2

move /Y temp.txt file2.txt


echo Данные добавлены в начало файла file2.txt

pause
```

c) Объяснение:

- `type file1.txt > temp.txt` — перезаписывает `temp.txt` содержимым `file1.txt`.
- `type file2.txt >> temp.txt` — дополняет `temp.txt` данными из `file2.txt`.
- `move /Y temp.txt file2.txt` — заменяет `file2.txt` на объединенный файл.
- `/Y` — отключает запрос на подтверждение замены.

Зачетный билет №9

1. Понятие ОС. Задачи ОС. ОС как виртуальная машина. ОС как система управления ресурсами.

a) Понятие операционной системы (ОС) :

- ОС — это комплекс системных программ, управляющий работой компьютера и обеспечивающий взаимодействие между пользователем и аппаратурой.
- Является **интерфейсом** между приложениями и оборудованием.

b) Основные задачи ОС :

1. **Предоставление удобного интерфейса** для пользователя.
2. **Управление ресурсами** :
   - процессором,
   - памятью,
   - устройствами ввода/вывода,
   - файловой системой.
1. **Обеспечение безопасности и защиты данных** .
2. **Поддержка многозадачности и многопользовательской работы** .

c) ОС как виртуальная машина :

- ОС предоставляет пользователю и приложениям абстрактную, "виртуальную машину", которая проще и удобнее реального оборудования.
- Программист работает с логическими представлениями ресурсов: файлы, процессы, устройства, не заботясь об их аппаратной реализации.

d) ОС как система управления ресурсами :

- ОС распределяет ресурсы между процессами.
- Управляет доступом к ресурсам и разрешает конфликты.
- Обеспечивает эффективное использование вычислительных ресурсов.

2. Файлы. Структура тома с файловой системой NTFS.

a) NTFS (New Technology File System) :

- Современная файловая система Windows.
- Предназначена для обеспечения надежности, производительности и гибкости.

b) Структура тома NTFS :

Загрузочный сектор (Boot Sector)	Содержит информацию о структуре тома и загрузчик.
Master File Table (MFT)	Главная таблица, где хранится информация обо всех файлах и каталогах. Каждый файл имеет свою запись в MFT.
Площадка MFT (MFT Zone)	Область вокруг MFT, зарезервированная для ее роста.
Область данных	Физическое место хранения содержимого файлов.
Служебные файлы NTFS	Такие как `$LogFile`, `$Bitmap`, `$Volume` и др., обеспечивают целостность и функциональность системы.

c) Особенности MFT :

- В каждой записи MFT хранятся атрибуты файла: имя, размер, права доступа, расположение данных.
- Если файл маленький, его данные могут храниться прямо в MFT (резидентное хранение).
- Для больших файлов указывается только адрес в области данных (нерезидентное хранение).

3. Создать командный файл, выполняющий действия:

В случае отсутствия указанного файла в текущей директории, скопировать его туда из другой, где он по условию существует. Если файл уже есть, вывести соответствующее сообщение

Решение:

```
@echo off

setlocal


:: Параметры

set "filename=data.txt"

set "source_path=C:\source\%filename%"

set "target_path=%cd%\%filename%"


echo === Проверка наличия файла ===


:: Проверяем, существует ли файл в текущей директории

if exist "%target_path%" (

 echo Файл %filename% уже существует в текущей директории.

 goto end

)


:: Если файла нет — копируем его из источника

if exist "%source_path%" (

 echo Копирую файл из %source_path%

 copy "%source_path%" "%target_path%" >nul
```

```
 echo Файл успешно скопирован.
) else (
 echo Исходный файл %filename% не найден в %source_path%
)

:end
pause
```

Объяснение:

- `if exist` — проверяет наличие файла.
- `%cd%` — текущая директория.
- `goto end` — переход к метке для завершения программы.
- `copy` — команда копирования.
- `>nul` — подавляет вывод результата копирования.

Зачетный билет №10

1. История и эволюция ОС

а) Этапы развития ОС :

1945–1955 Нулевое поколение ЭВМ. Программы запускались вручную. Нет ОС.
1955–1965 Появились первые ОС, реализующие **пакетную обработку заданий** .
1965–1980 Мультипрограммирование, разделение времени, появление UNIX.
С 1980 г. Графические интерфейсы, сетевые возможности, персональные компьютеры (MS-DOS, Windows, Linux).

1-е поколение (конец 1940-х — начало 1950-х годов)

- **Аппаратная база:** электронно-вакуумные лампы.
- **Особенности программирования:**
- Программы загружались вручную.
- Не существовало понятия ОС.
- Каждая программа была уникальной и требовала ручного управления оборудованием.
- **Режим работы:** однопрограммный, без разделения времени.
- **Примеры компьютеров:** ENIAC, EDSAC.

2-е поколение (середина 1950-х — начало 1960-х годов)

- **Аппаратная база:** транзисторы.
- **Начало появления первых управляющих программ.**
- **Появились:**
- Библиотеки подпрограмм.
- Программы-мониторы, которые обеспечивали автоматическую последовательную обработку заданий.
- **Режим работы:** пакетная обработка заданий.

- **Цель:** уменьшить время между выполнением программ.
- **Примеры ОС:** IBM FORTRAN Monitor System.

3-е поколение (1960-е — 1970-е годы)

- **Аппаратная база:** интегральные схемы.
- **Основные достижения:**
- Внедрение мультипрограммирования (одновременное хранение нескольких программ в памяти).
- Появление систем разделения времени (time-sharing), позволяющих нескольким пользователям работать с одной машиной одновременно.
- Развитие файловых систем и защиты данных.
- **Примеры ОС:**
- **IBM OS/360** — одна из первых универсальных операционных систем, созданная для широкого спектра оборудования.
- **Multics** — экспериментальная система, оказавшая влияние на развитие UNIX.
- **Изменения в пользовательском интерфейсе:** командный интерфейс стал более развитым.

4-е поколение (с 1980-х годов по настоящее время)

- **Аппаратная база:** микропроцессоры, большие интегральные схемы.
- **Важнейшие изменения:**
- Распространение персональных компьютеров (ПК).
- Появление графического интерфейса пользователя (GUI) — например, Apple Macintosh, Microsoft Windows.
- Развитие сетевых технологий и поддержка многозадачности и многопользовательской работы.
- Интеграция с интернетом и облачными технологиями.
- **Примеры ОС:**
- **MS-DOS** — популярная однозадачная ОС для ПК.
- **Windows** — семейство графических ОС от Microsoft.
- **UNIX/Linux/macOS** — мощные многозадачные, многопользовательские системы.
- **Android/iOS** — мобильные ОС.
- **Современные тенденции:**
- Поддержка виртуализации.
- Облачные технологии.
- Высокий уровень безопасности и автоматизации.
- Искусственный интеллект в управлении системами.

b) Современные направления :

- Поддержка **реального времени** .
- **Распределенные ОС** и **сетевые технологии** .
- **Мобильные и облачные ОС** .

2. Хранение данных реестра

a) Что такое реестр Windows?

- Центральное хранилище параметров и настроек ОС, оборудования и программ.
- Реестр представляет собой **иерархическую базу данных** .

b) Где хранится реестр ?

Файлы реестра находятся в:

1

C:\Windows\System32\config\

Основные файлы:

- `SYSTEM` – параметры системы.
- `SOFTWARE` – настройки установленных программ.
- `SECURITY` – политики безопасности.
- `SAM` – учетные записи локальных пользователей.
- `DEFAULT` – шаблон профиля по умолчанию.

c) Как происходит хранение данных ?

- Данные реестра загружаются в память при старте Windows.
- Изменения временно хранятся в оперативной памяти и записываются на диск при необходимости или при завершении работы.
- Используется механизм **журналирования** , чтобы предотвратить повреждение данных при сбое.

d) Редактирование реестра :

- Через редактор: `regedit.exe`.
- Через командную строку: `reg add`, `reg query`, `reg delete`.

3. Создать командный файл, который:

Используя функцию FOR, в цикле выводит на экран содержимое указанного в командной строке файла четыре раза.

Решение:

```
@echo off

setlocal enabledelayedexpansion


:: Проверяем, передан ли файл

if "%~1"=="" (

 echo Использование: %0 имя_файла

 exit /b

)
```

```
:: Проверяем существование файла
if not exist "%~1" (
 echo Файл %~1 не найден.
 exit /b
)

:: Четыре раза выводим содержимое файла
for /L %%i in (1,1,4) do (
 echo --- Вывод файла: %%i ---
 type "%~1"
 echo.
)

pause
```

Как использовать:
my_script.bat myfile.txt

Объяснение:

- `%~1` — первый аргумент командной строки.
- `for /L %%i in (1,1,4)` — цикл от 1 до 4.
- `type` — вывод содержимого файла.
- `--- Вывод файла: %%i ---` — разделитель для наглядности.

Зачетный билет №11

1. Классификация ОС по особенности управления ресурсами. Режимы многозадачности и многопользовательности.

а) Классификация ОС по управлению ресурсами :

- **Однозадачные ОС** : выполняют только одну задачу в момент времени (например, MS DOS).
- **Многозадачные ОС** : поддерживают выполнение нескольких программ одновременно.

- Вытесняющая многозадачность — процесс может быть прерван системой (Windows NT, UNIX).
- Невытесняющая многозадачность — процесс сам передает управление (Windows 3.x).
- **Однопользовательские ОС** : рассчитаны на одного пользователя (MS Windows до XP).
- **Многопользовательские ОС** : поддерживают работу нескольких пользователей одновременно (UNIX, Linux).

b) Режимы многозадачности :

- **Вытесняющая многозадачность** :
- Процессорное время распределяется автоматически планировщиком.
- Обеспечивает справедливое распределение времени.
- Пример: Windows, Linux.
- **Невытесняющая многозадачность** :
- Процесс работает до тех пор, пока не завершится или сам не отдаст управление.
- Менее устойчива к зависанию.
- Пример: Windows 3.x.

c) Многопользовательский режим :

- Позволяет нескольким пользователям одновременно работать с одной системой через терминалы или сетевые подключения.
- Реализуется через механизмы учетных записей, прав доступа и защиты данных.
- Пример: серверные версии UNIX/Linux.

2. Понятие вычислительного ресурса

a) Что такое вычислительный ресурс?

Это любой элемент компьютерной системы, который может быть использован программой для выполнения задачи.Понятие ресурса строго не определено. Будем считать, что всякий потребляемый объект (независимо от формы его существования), обладающий некоторой практической ценностью для потребителя, является *ресурсом* [12].

Ресурсы различаются по запасу выделяемых единиц ресурса и бывают в этом смысле *исчерпаемыми* и *неисчерпаемыми*.

К *исчерпаемым* ресурсам относится, например, центральный процессор.

В качестве *неисчерпаемого* ресурса можно представить, например, память, выделяемую программе, если рассматривать ее как совокупность всех имеющихся в компьютере запоминающих устройств. В то же время, запоминающее устройство, состоящее только из оперативной памяти с единственным трактом записи/считывания, представляет собой исчерпаемый ресурс.

Исчерпаемость ресурса, как правило, приводит к конфликтам среди потребителей этого ресурса. Для регулирования конфликтов ресурсы должны распределяться между потребителями по каким-то правилам, в наибольшей степени их удовлетворяющим.

b) Виды вычислительных ресурсов :

Процессор	Время работы ЦП для выполнения инструкций программы
Оперативная память	Объем доступной RAM
Дисковое пространство	Объем свободного места на жестком диске
Устройства ввода-вывода	Принтеры, сканеры, клавиатура, мышь и т.д.
Сетевые ресурсы	Доступ к сети, сетевые порты, интернет-соединение
Программные ресурсы	Файлы, семафоры, потоки, библиотеки

c) Задачи управления ресурсами :

- Учет занятых и свободных ресурсов.
- Распределение ресурсов между процессами.
- Разрешение конфликтов между процессами.
- Защита ресурсов от несанкционированного использования.

3. Создать командный файл, выполняющий различные действия в зависимости от переданного параметра в строке вызова:

Создание каталога и копирование в него всех .txt файлов
Вывод на экран содержимого каталога и переименование всех .txt файлов в .asm файлы

Решение:

```
@echo off

setlocal


:: Проверяем наличие аргумента

if "%~1"=="" (

 echo Использование: %0 [create ^| rename]

 exit /b

)


:: Если первый аргумент = "create"

if /i "%~1"=="create" (
```

```
 echo === Создание каталога и копирование .txt файлов ===
 if not exist txt_files mkdir txt_files
 copy *.txt txt_files\ >nul
 echo Каталог создан, .txt файлы скопированы.
 dir txt_files\*.txt
 goto end
)

:: Если первый аргумент = "rename"
if /i "%~1"=="rename" (
 echo === Переименование .txt файлов в .asm ===
 if not exist txt_files (
 echo Каталог txt_files не найден!
 exit /b
 )
 cd txt_files
 for %%f in (*.txt) do (
 ren "%%f" "%%~nf.asm"
 )
 cd ..
 dir txt_files\*.asm
 goto end
)

:: Если неверный параметр
echo Неверный параметр. Используйте 'create' или 'rename'.
:end
pause
```

Как использовать:

my_script.bat create – создаст каталог и скопирует все .txt файлы

my_script.bat rename – переименует .txt в .asm

Объяснение:

- `if "%~1"==""` — проверяет, передан ли параметр.
- `copy *.txt folder\` — копирует все текстовые файлы.
- `for %%f in (*.txt)` — перебирает файлы.
- `ren "%%f" "%%~nf.asm"` — переименовывает файл, меняя расширение.

Зачетный билет №12

1. Классификация ОС по архитектуре вычислительных систем. Сетевые, многопроцессорные и кластерные ОС.

a) ОС по архитектуре вычислительных систем :

Сетевые ОС	Предоставляют средства для подключения к сети, обмена данными, удаленного доступа (Windows Server, Novell NetWare).
Многопроцессорные ОС	Управляют несколькими процессорами внутри одного компьютера. Поддерживают параллелизм (Solaris, Windows Server).
Кластерные ОС	Объединяют несколько компьютеров в единый вычислительный ресурс. Используются для повышения отказоустойчивости и производительности (Hadoop, Beowulf).

b) Отличия :

- Сетевые ОС — ориентированы на коммуникации.
- Многопроцессорные ОС — на параллелизм внутри одной машины.
- Кластерные ОС — на объединение нескольких машин в одно целое.

2. Структура и назначение реестра

a) Что такое реестр Windows?

- Центральная база данных конфигурационной информации ОС.
- Хранит настройки оборудования, параметры ОС, пользовательские данные и приложения.

b) Основные разделы реестра :

`HKEY_CLASSES_ROOT`	Связь типов файлов с программами
`HKEY_CURRENT_USER`	Настройки текущего пользователя
`HKEY_LOCAL_MACHINE`	Конфигурация всей системы
`HKEY_USERS`	Настройки всех пользователей
`HKEY_CURRENT_CONFIG`	Информация о текущей аппаратной конфигурации

c) Структура реестра :

- Ключи (keys) — аналогичны папкам.
- Подключи и значения (values) — данные в формате "ключ-значение".

d) Назначение :

- Хранение информации о:
- Установленных программах.
- Параметрах автозапуска.
- Настройках безопасности.
- Драйверах устройств.
- Политиках системы и пользователей.

e) Редактирование реестра :

- Через `regedit.exe`.
- Через командную строку: `reg add`, `reg query`, `reg delete`.

3. Создать дисковый файл, занести в файл информацию с клавиатуры размером не менее 128 байт

Решение:

```
@echo off

setlocal


:: Создаем файл и просим ввести данные

echo Введите не менее 128 символов:

set /p input=>


:: Сохраняем в файл

echo %input% > user_input.txt


:: Проверяем размер файла

for %%F in (user_input.txt) do set size=%%~zF


if %size% geq 128 (

 echo Данные сохранены в файл user_input.txt

) else (

 echo Введено меньше 128 байт!

)
```

:: Показываем содержимое

type user_input.txt

pause

Как использовать:

1

Введите текст вручную при запуске скрипта.

Объяснение:

- `set /p input=>` — считывает ввод от пользователя.
- `echo %input% > file.txt` — записывает в файл.
- `set size=%%~zF` — получает размер файла.
- `geq` — сравнение «больше или равно».

Для ввода более 128 байт можно ввести длинную строку, например:

Зачетный билет №13

1. Контекст процесса

a) Что такое контекст процесса?

Контекст процесса — это **состояние процессора и среды** , необходимое для возобновления выполнения процесса после его прерывания.
Это **полная информация** , которая позволяет системе приостановить выполнение процесса и позже продолжить его с того же места.

b) Что включает контекст процесса?

- Содержимое регистров процессора (включая **регистры общего назначения**).
- Состояние флагов (**EFLAGS / RFLAGS**).
- Указатель команд (**EIP / RIP**) — адрес следующей инструкции.
- Указатель стека (**ESP / RSP**).
- Таблицы страниц (для страничной организации памяти).
- Открытые файловые дескрипторы.
- Информация о состоянии ввода-вывода.
- Текущее состояние процесса (готовность, выполнение, ожидание).

c) Зачем нужен контекст процесса?

- Чтобы обеспечить **многозадачность** : переключаться между процессами без потери данных.
- Для корректного **восстановления работы** процесса после прерывания.
- Для **отладки и анализа состояния** программы.

d) Как происходит сохранение и восстановление контекста?

При переключении задач:

1. Контекст текущего процесса **сохраняется** в специальной области памяти (например, в **TSS — Task State Segment** в архитектуре x86).
2. Контекст нового процесса **загружается** из его собственного хранилища.
3. Выполнение продолжается с точки, где процесс был прерван.

2. Файловая система FAT. Структура и назначение корневого каталога

a) Файловая система FAT

FAT (File Allocation Table) — одна из первых популярных файловых систем, использовавшихся в MS-DOS и ранних версиях Windows.

Основные компоненты FAT:

- **Boot Sector** – загрузочный сектор.
- **FAT таблица (2 копии)** – содержит информацию о занятых, свободных и поврежденных кластерах.
- **Корневой каталог** – список файлов и подкаталогов верхнего уровня.
- **Область данных** – место хранения содержимого файлов.

b) Корневой каталог в FAT

- **Расположение** : сразу после таблиц FAT.
- **Фиксированный размер** : состоит из 32-байтных записей.
- **Максимальное количество записей** : зависит от версии FAT:
- FAT12: до 512 записей.
- FAT16: может быть больше, но всё равно ограничен.

c) Структура одной записи корневого каталога (32 байта) :

Имя файла	8 байт	В формате 8.3
Расширение	3 байта	Например, TXT, EXE
Атрибуты	1 байт	Архивный, скрытый, системный, только чтение и т.д.
Зарезервировано	10 байт	Не используется или для LFN
Время последней модификации	2 байта	Формат: ННММSS
Дата последней модификации	2 байта	Формат: ГГГГММДД
Начальный кластер	2 байта	Номер первого кластера файла
Размер файла	4 байта	В байтах

d) Назначение корневого каталога

- Хранение информации о файлах и подкаталогах, находящихся в корне диска.
- Является **фиксированной областью** , поэтому имеет ограничения по количеству элементов.
- Обеспечивает начальную точку доступа к файловой системе.

3. Последовательность действий сохранения и восстановления реестра Windows

a) Что такое реестр Windows?

Реестр — это **центральная база данных конфигурационной информации** операционной системы и установленных программ.

b) Как сохранить реестр

1. **Открыть редактор реестра** :
   - Win + R → `regedit` → Enter
1. **Выбрать раздел для сохранения** :
   - Например, `HKEY_CURRENT_USER\Software\MyApp`
1. **Экспорт раздела** :
   - ПКМ на ключе → **Экспорт раздела реестра**
   - Сохранить как `.reg`-файл

c) Резервное копирование всего реестра через командную строку :

reg export HKLM C:\backup\HKLM_backup.reg

reg export HKCU C:\backup\HKCU_backup.reg

d) Восстановление реестра

1. **Через редактор реестра** :
   - Открыть `regedit`
   - Файл → Импорт → выбрать `.reg`-файл
1. **Через командную строку** :

reg import C:\backup\HKLM_backup.reg

e) Резервное копирование реестра через System File Checker (SFC) :

sfc /scannow

f) Важные замечания:

- Перед изменением реестра **всегда делайте резервную копию** .
- Работа с реестром требует **прав администратора** .
- Некорректные изменения могут привести к **нестабильной работе системы** .

Зачетный билет №14

1. Основные режимы процесса. Состояние процессов

a) Основные состояния процесса :

Выполнение (Running)	Процесс исполняется на процессоре
Готовность (Ready)	Процесс готов к выполнению, ждет очереди
Ожидание (Waiting/Blocked)	Процесс ожидает завершения внешнего события (например, ввода-вывода)

b) Переходы между состояниями :

1. **Новый → Готовность** : процесс создан и добавлен в очередь готовых.

2. **Готовность → Выполнение** : процесс получил процессорное время.
3. **Выполнение → Готовность** : истек временной квант (при вытесняющем алгоритме).
4. **Выполнение → Ожидание** : процесс запрашивает ресурс, который недоступен.
5. **Ожидание → Готовность** : ресурс стал доступен, процесс снова в очереди.

c) Режимы процесса :

Пользовательский режим (User Mode)	Процесс выполняет пользовательский код. Не имеет доступа к защищенным ресурсам напрямую.
Режим ядра (Kernel Mode)	Процесс вызывает системные функции, может обращаться ко всем ресурсам компьютера.

2. Основные файловые операции

a) Что такое файловая операция?

Это действие над файлом или каталогом, реализованное через интерфейс файловой системы.

b) Основные файловые операции :

Создание файла	Создание нового файла в файловой системе
Открытие файла	Получение дескриптора файла для дальнейшей работы
Чтение файла	Чтение данных из файла
Запись в файл	Запись новых данных в файл
Закрытие файла	Завершение работы с файлом
Удаление файла	Удаление файла из файловой системы
Переименование файла	Изменение имени файла
Перемещение файла	Перемещение файла в другую директорию
Копирование файла	Создание копии файла
Поиск файлов	Поиск файлов по имени, расширению, дате и т.д.

c) Примеры в командной строке Windows :

```
copy source.txt destination.txt :: копирование

move file.txt folder\ :: перемещение

del oldfile.txt :: удаление

ren file.txt newfile.txt :: переименование

type file.txt :: вывод содержимого

dir :: просмотр содержимого каталога
```

3. Последовательность действий нахождения цепочки кластеров в файловой системе FAT12

a) Что такое FAT12?

FAT12 — это файловая система, использующая **12-битные элементы таблицы размещения файлов (FAT)** . Применялась на дискетах.

b) Как устроена файловая система FAT12?

- Все пространство диска делится на **кластеры** .
- Первые два кластера (0 и 1) зарезервированы.
- Кластер 2 и далее используются для хранения файлов и каталогов.
- **Таблица FAT** показывает, какие кластеры заняты, какие свободны, какие являются последними в цепочке.

c) Формат элемента FAT12

- 12 бит на один кластер.
- Если кластер является последним в цепочке — значение `0xFF8-0xFFF`.
- Если кластер свободен — значение `0x000`.

d) Алгоритм нахождения цепочки кластеров:

Дано:

- Начальный номер кластера `N`, например, `N = 3`.

Шаги:

1. Найдите **номер сектора** , в котором находится FAT.
   - Обычно FAT начинается сразу после загрузочного сектора.
1. **Вычислите смещение в FAT** :
   - Каждый элемент FAT занимает 12 бит.
   - Для кластера N: `offset = N * 1.5 байта` (т.к. 12 бит = 1.5 байта).
   - Это значит, что данные о кластере могут находиться в двух смежных байтах.
1. **Прочитайте значение из FAT** :
   - Если значение = `0FF8h-0FFFh`: это **конец цепочки** .
   - Если значение = `000h`: кластер **свободен** .
   - Иначе: это номер **следующего кластера** в цепочке.
1. **Перейдите к следующему кластеру** и повторите шаги 2–3.
2. **Продолжайте до тех пор, пока не найдете конец цепочки** (`0FF8h-0FFFh`).

e) Пример :

Предположим, начальный кластер — 3. FAT содержит:

Кластер | Значение

3 | 0x004

4 | 0x005

5 | 0x000

6 | 0xFFF

Тогда цепочка будет:

1

3 → 4 → 5 → 6 → конец

f) Зачем это нужно?

- Для определения местоположения файла на диске.
- Для восстановления поврежденных файлов.
- Для диагностики и проверки целостности дискового пространства.

Зачетный билет №15

1. MFT (Master File Table) в NTFS

a) Что такое MFT?

- **MFT (Master File Table)** — это центральная структура файловой системы **NTFS** , представляющая собой базу данных всех файлов и каталогов на томе.
- Каждый файл или каталог имеет свою запись в MFT.

b) Структура MFT :

- Каждая запись содержит **атрибуты файла** :
- Имя файла
- Размер
- Дата создания, изменения, последнего доступа
- Права доступа (ACL)
- Указатель на данные (адрес и размер)
- Тип объекта: файл или каталог

c) Особенности MFT :

- Первая запись MFT описывает саму таблицу MFT.
- Если файл маленький (до ~700 байт), его содержимое хранится прямо в MFT (**резидентное хранение**).
- Для больших файлов MFT содержит только адреса кластеров, где хранятся данные (**нерезидентное хранение**).

d) Преимущества MFT :

- Быстрый поиск информации о файле.
- Поддержка длинных имен, прав доступа, сжатия, шифрования.
- Журналирование изменений MFT обеспечивает устойчивость к сбоям.

2. Алгоритм исключения «гонок» с помощью критической секции

a) Что такое "гонки" (race condition)?

Это ситуация, при которой результат выполнения программы зависит от порядка выполнения процессов, обращающихся к общим данным.

b) Понятие критической секции :

- Это участок кода, в котором происходит **доступ к разделяемым ресурсам** .
- В каждый момент времени **в критической секции может находиться только один процесс** .

c) Цель :

Обеспечить **взаимное исключение (mutual exclusion)** , чтобы исключить гонки.

d) Алгоритм Петерсона (пример решения задачи взаимного исключения) :

```
int turn; // чья очередь ждать

int interested[2]; // кто хочет войти в критическую секцию


void enter_region(int process) {

int other = 1 - process;

interested[process] = TRUE;

turn = other;

while (interested[other] && turn == other);

}


void leave_region(int process) {

interested[process] = FALSE;

}
```

e) Как работает :

- `enter_region` проверяет, не находится ли другой процесс в критической секции.
- Если да — текущий процесс ждет.
- После завершения работы вызывается `leave_region`, освобождающий возможность входа другому процессу.

f) Результат :

- Гарантировано, что **только один процесс** будет выполнять критический участок кода.
- Таким образом, исключаются ситуации гонок.

3. Написать программу, которая добавит в текстовый файл данные, считываемые с клавиатуры, в конец файла

□ Пример на языке Python:

```
filename = "output.txt"


print("Введите данные для добавления в файл (введите 'exit' для завершения):")

with open(filename, "a") as file:
```

```
while True:
user_input = input()
if user_input.lower() == "exit":
break
file.write(user_input + "\n")

print(f"Данные добавлены в {filename}")
```

□ Пример на языке C:

```
#include <stdio.h>

int main() {
FILE *fp;
char data[100];

fp = fopen("output.txt", "a");
if (fp == NULL) {
printf("Не удалось открыть файл\n");
return 1;
 }

printf("Введите данные для добавления в файл (введите 'exit' для завершения):\n");

while (1) {
fgets(data, sizeof(data), stdin);
if (strcmp(data, "exit\n") == 0)
break;
fputs(data, fp);
 }
```

```
fclose(fp);

printf("Данные успешно добавлены.\n");

return 0;

}
```

Объяснение:

- Файл открывается в режиме `"a"` (append), чтобы данные добавлялись в конец.
- Пользователь вводит строки до тех пор, пока не наберет `exit`.
- Введенные данные сохраняются в файл.

Зачетный билет №16

1. Синхронизация процессов. Понятие критической секции

a) Критическая секция :

- Это участок кода, в котором осуществляется доступ к **разделяемым ресурсам** (например, общей памяти, устройству ввода-вывода).
- В каждый момент времени **в критической секции должен быть максимум один процесс** .

b) Задача синхронизации :

- Обеспечить **взаимное исключение** .
- Предотвратить **занос в состояние ожидания навсегда** (ограничить время ожидания).
- Не допустить **голодания** (каждый процесс должен иметь возможность попасть в критическую секцию).

c) Методы реализации критических секций :

- **Блокирующие переменные (флаги)** – простой, но неэффективный метод.
- **Семафоры** – более универсальный механизм.
- **Мьютексы** – специализированные семафоры для защиты одного ресурса.
- **Мониторы** – конструкции языка программирования, обеспечивающие безопасный доступ к ресурсам.

2. Файловые системы. Файловая система NTFS

a) NTFS (New Technology File System) — современная файловая система Windows.

b) Основные особенности NTFS :

Длинные имена файлов	До 255 символов
Поддержка больших объемов	До 256 ТБ на один том
Права доступа (ACL)	Гибкая система разрешений
Шифрование (EFS)	Шифрование файлов и папок
Журналирование (логирование)	Все изменения записываются в журнал для восстановления
Сжатие файлов	Без потерь качества
Самовосстановление	Автоматическое восстановление после сбоев

Символические ссылки	Поддержка точек монтирования и ссылок

c) MFT — главная структура NTFS :

- Хранится в начале тома.
- Каждая запись описывает файл или каталог.
- Поддерживает быстрое перемещение по файловой системе и защиту данных.

3. Построить диаграмму выполнения трех процессов P1, P2 и P3 при невытесняющем алгоритме

Дано:

- Время выполнения:
- P1 = 3
- P2 = 5
- P3 = 8

a) Невытесняющий алгоритм планирования :

- Процесс сам передает управление ОС.
- Выполняется до завершения или добровольной передачи управления.

b) Диаграмма выполнения :

1

2

3

Время: 0 3 8 16

|-----|---------|-----------|

Процесс: P1 P2 P3

c) Расчет средних значений :

i) Sw — среднее время ожидания процессами предоставления процессора

P1 0
P2 3
P3 8
Sw=0+3+8/3=311≈3.67

ii) Ss — среднее время выполнения заданиями начального запуска

P1 3
P2 3 + 5 = 8
P3 8 + 8 = 16
Ss=3+8+16/3=327=9

d) Таблица состояний процессов (по временным интервалам)

1 И Г Г
2 И Г Г

3 И Г Г
4 И Г
5 И Г
6 И Г
7 И Г
8 И Г
9 И
10 И
11 И
12 И
13 И
14 И
15 И
16 И

Обозначения:

- И — исполнение
- Г — готовность

Зачетный билет №17

1. Логическая структура диска в DOS

a) Что такое логическая структура диска?

Логическая структура диска — это способ организации данных на диске, при котором физическое устройство представляется как последовательность секторов, кластеров и файловых структур.

b) Основные компоненты логической структуры диска в FAT:

Загрузочный сектор (Boot Sector)	Содержит код загрузчика и информацию о структуре тома.
Таблица размещения файлов (FAT)	Указывает, какие кластеры заняты, свободны или повреждены. Хранится в двух копиях.
Корневой каталог (Root Directory)	Список файлов верхнего уровня (до 512 записей в FAT12).
Область данных (Data Area)	Место хранения содержимого файлов и подкаталогов.

c) Как работает логическая структура:

- Диск делится на **секторы** (обычно по 512 байт).
- Секторы объединяются в **кластеры** — минимальную единицу выделения памяти.
- Файл хранится в виде цепочки кластеров, информация о которой находится в **FAT** .
- Каждая запись корневого каталога занимает **32 байта** и содержит:
- Имя файла
- Атрибуты
- Дату и время создания
- Начальный кластер
- Размер файла

2. Структура реестра Windows

a) Что такое реестр Windows?

Реестр — это **централизованная база данных параметров и настроек** операционной системы и установленных программ.

b) Основные разделы реестра:

`HKEY_CLASSES_ROOT`	Связь между типами файлов и программами
`HKEY_CURRENT_USER`	Настройки текущего пользователя
`HKEY_LOCAL_MACHINE`	Конфигурация всей системы
`HKEY_USERS`	Параметры всех пользователей
`HKEY_CURRENT_CONFIG`	Информация о текущей аппаратной конфигурации

c) Структура записи в реестре:

- **Ключи** — аналогичны папкам.
- **Подключи** — вложенные ключи.
- **Параметры (values)** — данные, связанные с ключом:
- **REG_SZ** – строка
- **REG_DWORD** – 32-битное число
- **REG_QWORD** – 64-битное число
- **REG_BINARY** – бинарные данные
- **REG_MULTI_SZ** – список строк

d) Пример пути в реестре:

1

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

3. Дата создания файла в байтах 18h..19h содержат числа ADh и 30h. Формат даты: ГГГГГГГММММДДДДД (Год, Месяц, День), где год начинается с 1980. Определить дату.

Дано:

- Байты даты: `ADh`, `30h`
- В шестнадцатеричном виде: `30ADh`
- В двоичном виде: `0011 0000 1010 1101`

Распределяем биты по полям:

ГГГГГГГ ММММ ДДДДД

(7 бит) (4 бита) (5 бит)

- `30h = 0011 0000`
- $30_{16} = (3 \times 16^1) + (0 \times 16^0) = 48 + 0 = 48_{10}$

Деление	Целое частное	Остаток
48 ÷ 2	24	0

24 ÷ 2	12	0
12 ÷ 2	6	0
6 ÷ 2	3	0
3 ÷ 2	1	1
1 ÷ 2	0	1

- `ADh = 1010 1101`
- $AD_{16} = (10 \times 16^1) + (13 \times 16^0) = 160 + 13 = 173_{10}$

Деление	Целое частное	Остаток
173 ÷ 2	86	1
86 ÷ 2	43	0
43 ÷ 2	21	1
21 ÷ 2	10	1
10 ÷ 2	5	0
5 ÷ 2	2	1
2 ÷ 2	1	0
1 ÷ 2	0	1

```
      30ADh → 0011 0000 1010 1101

0011 000] [0 1010] [1101]

    ↑           ↑         ↑

   Год         Месяц     День

 (7 бит)      (4 бита)  (5 бит)
```

Шаг 3: Разбираем на поля

Поле "Год" (7 бит): **0011000**

$0011000_2 = (0 \times 2^6) + (0 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 0 + 0 + 16 + 8 + 0 + 0 + 0 = 24_{10}$

$0 \times 64 = 0$

$0 \times 32 = 0$

$1 \times 16 = 16$

$1 \times 8 = 8$

$0 \times 4 = 0$

$0 \times 2 = 0$

$0 \times 1 = 0$

Итого: $16 + 8 = 24$

Реальный год = `1980 + 24 = 2004`

Поле "Месяц" (4 бита): **`0101`**

1012 = (0 × 23) + (1 × 22) + (0 × 21) + (1 × 20) = 0 + 4 + 0 + 1 = 510

0 × 8 = 0

1 × 4 = 4

0 × 2 = 0

1 × 1 = 1

Итого: 4 + 1 = 5

Месяц = **Май**

Поле "День" (5 бит): **`1101`**

11012 = (1 × 23) + (1 × 22) + (0 × 21) + (1 × 20) = 8 + 4 + 0 + 1 = 1310

Добавляем один ноль слева, чтобы получилось 5 бит: `01101`

0 × 16 = 0

1 × 8 = 8

1 × 4 = 4

0 × 2 = 0

1 × 1 = 1

Итого: 8 + 4 + 1 = 13

День = **13**

ИТОГО:

Дата из байтов `18h..19h = ADh и 30h`:

13 мая 2004 года

Зачетный билет №18

1. Основные режимы процесса. Состояние процессов.

а) Режимы процесса :

Пользовательский режим (User Mode)	Процесс выполняет пользовательский код. Не имеет доступа к защищенным ресурсам напрямую.
Режим ядра (Kernel Mode)	Процесс вызывает системные функции (например, ввод-вывод). Может обращаться ко всем ресурсам компьютера.

b) Состояния процесса :

Выполнение (Running)	Процесс исполняется на процессоре
Готовность (Ready)	Процесс готов к выполнению, но ждет своего времени
Ожидание (Waiting / Blocked)	Процесс ожидает завершения внешнего события (например, ввода-вывода)

c) Переходы между состояниями:

1. **Новый → Готовность** – процесс создан и добавлен в очередь готовых.
2. **Готовность → Выполнение** – процесс получил процессорное время.
3. **Выполнение → Готовность** – истек временной квант (при вытесняющем алгоритме).
4. **Выполнение → Ожидание** – процесс запрашивает ресурс, который недоступен.
5. **Ожидание → Готовность** – ресурс стал доступен.

2. Файловая система NTFS. Структура тома с файловой системой NTFS. Структура MFT (Master File Table) в NTFS.

a) NTFS — New Technology File System

- Современная файловая система Windows.
- Поддерживает длинные имена, большие объемы, права доступа, шифрование, сжатие и самовосстановление.

b) Структура тома NTFS :

Загрузочный сектор (Boot Sector)	Содержит данные о структуре тома и загрузчик.
MFT (Master File Table)	Главная таблица, где хранится информация обо всех файлах и каталогах.
Площадка MFT	Область вокруг MFT, зарезервированная для ее роста.
Область данных	Физическое место хранения содержимого файлов.
Служебные файлы NTFS	`$LogFile`, `$Bitmap`, `$Volume` и др., обеспечивают целостность и функциональность системы.

c) MFT — Master File Table

- Это **главная структура NTFS** , где каждая запись описывает:
- Имя файла/каталога
- Размер
- Даты создания, изменения, последнего доступа
- Права доступа (ACL)
- Расположение данных на диске
- Если файл маленький, его данные могут храниться прямо в MFT (**резидентное хранение**).
- Для больших файлов указывается только адрес в области данных (**нерезидентное хранение**).

3. Время последней модификации файла в байтах 16h...17h содержат числа 4Ah и 74h. Формат поля времени: ЧЧЧЧЧММММММССССС (Часы, Минуты, Секунды). Определить время.

Дано:

- Байты времени: `4A 74h`
- В шестнадцатеричном виде: `744Ah`
- В двоичном виде: `0111 0100 0100 1010`

Распределяем биты:

- 5 бит — часы
- 6 бит — минуты
- 5 бит — секунды (умножаются на 2, т.к. точность до 2 секунд)

a) `4A h` → `0100 1010`

$$4A_{16} = (4 \times 16^1) + (10 \times 16^0) = 64 + 10 = 74_{10}$$

Деление	Целое частное	Остаток
74 ÷ 2	37	0
37 ÷ 2	18	1
18 ÷ 2	9	0
9 ÷ 2	4	1
4 ÷ 2	2	0
2 ÷ 2	1	0
1 ÷ 2	0	1

b) `74 h` → `0111 0100`

$$74_{16} = (7 \times 16^1) + (4 \times 16^0) = 112 + 4 = 116_{10}$$

Деление	Целое частное	Остаток
116 ÷ 2	58	0
58 ÷ 2	29	0
29 ÷ 2	14	1
14 ÷ 2	7	0
7 ÷ 2	3	1
3 ÷ 2	1	1
1 ÷ 2	0	1

74h 4Ah → 0111 0100 0100 1010

ЧЧЧЧЧ ММММММ ССССС

5 бит 6 бит 5 бит

01110 100010 01010

↑ ↑ ↑

Часы Минуты Секунды

(5 бит) (6 бит) (5 бит)

Шаг 3: Переведем каждую часть в десятичную систему

a) Часы : `01110`

$01110_2 = (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 0 + 8 + 4 + 2 + 0 = 14_{10}$

$0 \times 16 = 0$

$1 \times 8 = 8$

$1 \times 4 = 4$

$1 \times 2 = 2$

$0 \times 1 = 0$

Всего: $8 + 4 + 2 = 14$

Часы = 14

b) Минуты : `100010`

$100010_2 = (1 \times 2^5) + (0 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 32 + 0 + 0 + 0 + 2 + 0 = 34_{10}$

$1 \times 32 = 32$

$0 \times 16 = 0$

$0 \times 8 = 0$

$0 \times 4 = 0$

$1 \times 2 = 2$

$0 \times 1 = 0$

Всего: $32 + 2 = 34$

 Минуты = 34

c) Секунды : `01010`

$01010_2 = (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 0 + 8 + 0 + 2 + 0 = 10_{10}$

$0 \times 16 = 0$

$1 \times 8 = 8$

$0 \times 4 = 0$

$1 \times 2 = 2$

0 × 1 = 0

Всего: 8 + 2 = 10

Но помни: **секунды хранятся с шагом в 2 секунды** , поэтому умножаем на 2:

10 × 2 = 20

 Секунды = 20

Итоговое время:

14 часов : 34 минуты : 20 секунд

Зачетный билет №19

1. Синхронизация процессов. Понятие критической секции. Семафоры. Тупики.

a) Синхронизация процессов

- Это механизм, обеспечивающий **корректное совместное использование общих ресурсов** несколькими процессами или потоками.
- Цель: предотвратить **состояние гонок (race condition)** — ситуацию, когда результат зависит от порядка выполнения операций.

b) Критическая секция

- Участок кода, в котором происходит **доступ к общему ресурсу** (например, переменной, файлу, устройству).
- В каждый момент времени **только один процесс** может находиться внутри критической секции.

Требования:

1. **Взаимное исключение** – не более одного процесса в критической секции.
2. **Прогресс** – если никто не находится в критической секции, то запрос должен быть удовлетворен.
3. **Ограничение времени ожидания** – процесс не должен бесконечно ждать входа в критическую секцию.

c) Семафоры

- Это целочисленная переменная, которая используется для контроля доступа к общим ресурсам.
- Основные операции:
- `wait(S)` – уменьшает значение семафора. Если S=0, процесс блокируется.
- `signal(S)` – увеличивает значение семафора и, возможно, возобновляет ожидание.

Пример:

semaphore mutex = 1;

```
process P1 {
wait(mutex);
// Критическая секция
signal(mutex);
}

process P2 {
wait(mutex);
// Критическая секция
signal(mutex);
}
```

d) Тупики (Deadlock)

- Ситуация, при которой несколько процессов **ожидают ресурсов, занятых друг другом** , и не могут продолжить выполнение.

Условия возникновения:

1. **Взаимное исключение** – ресурс нельзя использовать одновременно.
2. **Удержание и ожидание** – процесс удерживает ресурсы и ожидает другие.
3. **Невозможность принудительного изъятия** – ресурсы можно освободить только добровольно.
4. **Циклическое ожидание** – цепочка процессов, где каждый ожидает ресурс, занятый следующим.

Методы предотвращения:

- Избегать циклического ожидания.
- Ограничивать количество используемых ресурсов.
- Использовать таймеры ожидания.

2. Структура диска UNIX. Распределение файла на диске в UNIX.

a) Структура диска в UNIX

Диск делится на **цилиндры, дорожки и секторы** .

Boot Block	Содержит загрузчик и информацию о структуре тома.
Superblock	Хранит метаданные о файловой системе: размер блока, количество inode, свободные блоки и inode.
Inode Table	Таблица inode, где хранится информация обо всех файлах (размер, права доступа, указатели на блоки).

Data Blocks	Область данных — физическое место хранения содержимого файлов.

b) inode

- Каждому файлу соответствует **inode** , который содержит:
- Права доступа
- Владельца и группу
- Размер файла
- Даты создания/модификации
- Указатели на блоки данных

c) Распределение файла на диске

- Файл разбивается на **блоки** (обычно 512 байт или больше).
- Inode содержит список указателей на эти блоки.
- Для больших файлов используются **косвенные блоки** (указатели на другие блоки).

Например:

- 12 прямых указателей
- 1 косвенный указатель → указывает на блок с адресами других блоков
- 1 двойной косвенный → блок указателей на блоки указателей
- 1 тройной косвенный → три уровня указателей

d) Файловая система UFS (Unix File System) :

- Поддерживает длинные имена, права доступа, символические ссылки.
- Использует **индексное распределение** (через inode).

3. Создать командный файл, выполняющий действия:

Объединение содержимого 2-х файлов, находящихся в разных каталогах; вывод содержимого нового файла на экран; переименование исходных файлов. До и после каждого действия показать результат.

Решение:

```
@echo off

cls


:: Проверяем наличие файлов до объединения

echo [До] Состояние файлов:

dir C:\folder1\file1.txt

dir D:\folder2\file2.txt


:: Объединяем файлы
```

```
copy C:\folder1\file1.txt + D:\folder2\file2.txt combined.txt

echo.


:: Проверяем результат объединения

echo [После] Содержимое combined.txt:

type combined.txt

echo.


:: Переименовываем исходные файлы

ren C:\folder1\file1.txt file1_old.txt

ren D:\folder2\file2.txt file2_old.txt


:: Проверяем после переименования

echo [После] Список файлов:

dir C:\folder1\file1_old.txt

dir D:\folder2\file2_old.txt


pause
```

Как использовать:

- Сохраните как `MERGE.BAT`.
- Убедитесь, что файлы `file1.txt` и `file2.txt` существуют в своих каталогах.
- При запуске скрипт объединит их в `combined.txt`, выведет содержимое, а затем переименует исходные файлы.

Зачетный билет №20

1. Основные режимы процесса. Состояние процессов.

a) Режимы процесса :

Пользовательский режим (User Mode)	Процесс выполняет пользовательский код. Не имеет доступа к защищенным ресурсам напрямую.
Режим ядра (Kernel Mode)	Процесс вызывает системные функции (например, ввод-вывод). Может обращаться ко всем ресурсам компьютера.

b) Состояния процесса :

Выполнение (Running)	Процесс исполняется на процессоре

Готовность (Ready)	Процесс готов к выполнению, но ждет своего времени
Ожидание (Waiting / Blocked)	Процесс ожидает завершения внешнего события (например, ввода-вывода)

Переходы между состояниями:

1. **Новый → Готовность** : процесс создан и добавлен в очередь готовых.
2. **Готовность → Выполнение** : процесс получил процессорное время.
3. **Выполнение → Готовность** : истек временной квант (при вытесняющем алгоритме).
4. **Выполнение → Ожидание** : процесс запрашивает ресурс, который недоступен.
5. **Ожидание → Готовность** : ресурс стал доступен.

2. Файловая система NTFS. Структура тома с файловой системой NTFS. Структура MFT (Master File Table) в NTFS.

a) NTFS — New Technology File System

- Современная файловая система Windows.
- Поддерживает:
- Длинные имена файлов
- Шифрование и сжатие
- Журналирование
- Права доступа
- Самовосстановление

b) Структура тома NTFS :

Загрузочный сектор	Содержит данные о структуре тома и загрузчик
MFT (Master File Table)	Главная таблица, где хранится информация обо всех файлах и каталогах
Площадка MFT	Область вокруг MFT, зарезервированная для ее роста
Область данных	Физическое место хранения содержимого файлов
Служебные файлы NTFS	`$LogFile`, `$Bitmap`, `$Volume` и др., обеспечивают целостность и функциональность системы

c) MFT (Master File Table)

- Это **главная структура NTFS** , где каждая запись описывает:
- Имя файла/каталога
- Размер
- Даты создания, изменения, последнего доступа
- Права доступа (ACL)
- Расположение данных на диске
- Если файл маленький (до ~700 байт), его содержимое хранится прямо в MFT (**резидентное хранение**).
- Для больших файлов MFT содержит только адреса кластеров, где хранятся данные (**нерезидентное хранение**).

3. Привести пример возникновения тупиков

***!** Пример:*

Пусть два процесса — A и B — используют два ресурса: **принтер** и **сканер** .

a) Процесс A:

1. Запрашивает **принтер**
2. Запрашивает **сканер**
3. Использует оба ресурса
4. Освобождает оба ресурса

b) Процесс B:

1. Запрашивает **сканер**
2. Запрашивает **принтер**
3. Использует оба ресурса
4. Освобождает оба ресурса

⚠ Возможная ситуация тупика:

1. Процесс A занимает **принтер**
2. Процесс B занимает **сканер**
3. Процесс A пытается занять **сканер** , но он занят → **ждёт**
4. Процесс B пытается занять **принтер** , но он занят → **ждёт**

⚠ Получаем **тупик** — оба процесса ждут, пока другой освободит ресурс, которого не произойдет никогда.

🛠 Способы предотвращения тупиков:

Исключение циклического ожидания	Назначить единый порядок запроса ресурсов (например, сначала принтер, потом сканер)
Выделение всех ресурсов сразу	Процесс должен запросить все нужные ресурсы за один раз
Принудительное освобождение ресурса	Прервать процесс и освободить ресурсы
Использование таймаутов	Ограниченное время ожидания ресурса

Пример

Сначала строим и заполняем таблицу (табл.3-2). Столбцы соответствуют

моментам времени. Строки – процессам. Обозначение **И** используется для процессов, находящихся в состоянии исполнение, обозначение **Г** – для процессов, находящихся в состоянии готовность, пустые ячейки соответствуют завершившимся процессам. Состояния процессов показаны на протяжении соответствующей единицы времени, т.е. колонка с номером 1 соответствует промежутку от 0 до 1.

Таблица 3-2. Решение задание 3.1

Т 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

1 И И И И

2 Г Г Г Г И ИИ

3 Г Г Г Г Г ГГ И ИИ ИИ

4 Г Г Г Г Г ГГ Г ГГ ГГ И И

5 Г Г Г Г Г ГГ Г ГГ ГГ ГГ И И И И ИИ И И И

2.

Для определения среднего времени ожидания подсчитывается количество

клеток в таблице, заполненных состоянием Г - готовность. Например, для первого процесса время ожидания равно 0, а для третьего процесса – 7 единицам времени. Полученные значения суммируются, и результат делится на количество процессов. Среднее время ожидания СВО= (0+4+7+12+14)/5 = 7,4 единиц времени.

3.

Для определения среднего времени выполнения подсчитывается общее

количество заполненных клеток в таблице для каждого процесса. Например, для первого процесса время выполнения равно 4, а для третьего процесса – 12 единицам времени. Полученные значения суммируются, и результат делится на количество процессов. Среднее время выполнения СВВ=(4+7+12+14+23)/5 = 12 единиц времени.