Подсистема ввода-вывода обеспечивает связь вычислительной системы с внешними устройствами. В современных системах многие из таких устройств бывают представлены на логическом уровне в виде файлов, поэтому понятия **«управление вводом-выводом»** и **«управление файлами»** в значительной мере **синонимичны**.

Подходы к организации ввода-вывода:

- "**Наглый**" (наивный) ввод-вывод: Операции запускаются ЦП без предварительной проверки устройств, что может привести к блокировкам.
- С **предварительной проверкой**: Включает проверки состояния устройства, что универсально, но занимает много времени, что неэффективно для многозадачных систем.
- По **прерываниям**: Транзакции запускаются устройствами на основе событий, требует обработчиков событий, что усложняет программирование.
- С использованием **ПДП** (**DMA**): Использует прямой доступ в память без участия процессора, обеспечивая высокую производительность, но требует специальной аппаратной поддержки и прав для выполнения, что подходит для *драйверов*.

Модели ввода-вывода:

- **Блокирующий**: Поток останавливается и ждет завершения транзакции, что может замедлить выполнение программы.
- **Неблокирующий**: Предварительная проверка состояния устройства позволяет завершить транзакцию с ошибкой, если устройство не готово.
- **Мультиплексированный**: Программа анализирует состояния нескольких устройств и инициирует транзакции только с готовыми к обмену устройствами, обеспечивая параллельный обмен.
- **Управляемый сигналами**: Транзакции инициируются в рамках обработки событий устройства.
- **Многопоточная реализация**: Блокирующие операции выполняются в отдельных потоках, что позволяет избежать блокировок.
- **Асинхронный**: Используются системные механизмы для организации параллельных транзакций и потоков, обеспечивая эффективное управление ресурсами.
- **Отображение файлов в память**: Файлы проецируются в память для прямого доступа к их содержимому, обеспечивая эффективную передачу данных между файлом и буфером в памяти.

Блокирующий ввод-вывод, хоть и является естественным и простым подходом, не раскрывает полностью потенциал многозадачности. Другие, более сложные модели, могут обеспечить более эффективное исполнение операций и оптимальное использование ресурсов системы.

Файл – набор данных (обычно подразумеваются данные на внешнем носителе), пригодный для использования прикладными программами в вычислительной системе. Удобно представлять файл как совокупность **данных** (используются прикладными программами) и **метаданных** (используются системными программами).

Файловая система - способ организации данных на внешних носителях; модули ОС для работы с данными.

MS Windows поддерживает такие файловые системы как FAT (12, 16, 32, exFAT), NTFS, HPFS, и другие как CDFS.

Иерархическая файловая система имеет *корневую директорию*, содержащую файлы и поддиректории. В Microsoft каждый логический диск имел свое дерево директорий, начиная с буквы диска. Win NT ввел общий корневой узел, объединяющий логические диски как директории, но не все программы его отображают явно.

Идентификатор файла - его имя в файловой системе.
Полное имя файла - имя файла и путь к нему (включая вышестоящие директории). Путь может быть **абсолютным** (начиная с корневого каталога) или **относительным** (начиная с текущего каталога). Имя должно быть уникальным в текущем каталоге, а полное имя - во всей файловой системе. Глубина иерархии обычно не ограничена, но общая длина полных имен имеет ограничения.

Имена в файловых системах Microsoft:
- "**короткие**" - до 8 символов в имени и 3 символов в расширении, кодировка - ASCII (FAT12/16)
- "**длинные**" - до 255 символов, кодировка Unicode; поддерживается в FAT32, NTFS, exFAT

Типы файлов:
- **Обычные** (регулярные) файлы: содержат программы или данные.
- **Директории**: содержат списки других файлов и директориев.
- **Файлы-ссылки**: ссылки на другие файлы.
- **Файлы - логические устройства**: символьные (связанные с соответствующими устройствами) и блочные (для размещения на них файловых систем).
- **Файлы - коммуникационные ресурсы**: каналы, "почтовые ящики", сокеты и другие.

Основные атрибуты файлов:
- Тип (директория, системный, скрытый, только для чтения)
- Размер
- Дата и время создания
- Владелец и группа владельцев
- Права доступа

Объект-файл – идентификация файловым дескриптором HANDLE. Особенность: невалидному значению дескриптора соответствует не 0 (NULL), а константа INVALID_HANDLE_VALUE
Группы функций:
- Работа с файлами на диске:
    - Открытие и создание файлов с помощью функции CreateFile.
    - Чтение и запись данных в файлы с помощью функций ReadFile и WriteFile.
- Работа с директориями и другими структурами файловой системы:
    - Функции для работы с директориями, перемещения файлов и т.д.
- Получение информации о файловой системе и другие служебные функции.

Закрытие файла (прекращение действия его Handle) – общая функция CloseHandle(hFile)

Для более сложных сценариев, таких как асинхронный ввод-вывод, существуют расширенные версии функций с возможностью использования **callback-вызовов** по завершении операции.

Многопоточная реализация ввода-вывода подобна блокирующей модели, но длительные операции с файлами выносятся в отдельные потоки. Архитектура приложения включает выделенный поток-монитор и несколько потоков-исполнителей, управляемых монитором. Блокировка затрагивает только исполнительные потоки, сохраняя работоспособность других.

Преимущества:

- Простая логика в потоках-исполнителях, аналогичная блокирующей реализации.
- Эффективное использование ресурсов многопроцессорных систем.

Однако возникают проблемы взаимодействия между потоками и необходимость обеспечения **потокобезопасности** программы.

Отображение файлов в память базируется на управлении виртуальной памятью. Файлы проецируются в оперативную память для прямого доступа к их содержимому, обеспечивая эффективную передачу данных между файлом и буфером в памяти. Отображение файлов в память позволяет работать с файлами так, как если бы они были частью памяти, что обеспечивает эффективный доступ к данным. В Windows для этой операции используются функции CreateFileMapping() (Создает объект отображения файла, который позволяет отображать файл в память) и MapViewOfFile() (Создает отображение области файла в адресном пространстве процесса).

Мультиплексированный ввод-вывод — это способ работы с несколькими источниками данных, используя вызов select(). Этот метод позволяет эффективно проверять, готовы ли данные для чтения или записи, не дожидаясь готовности каждого источника.
Когда вызывается select(), он не ждет, пока все файлы будут готовы, а сразу выполняет доступные действия. Это требует некоторой логики, как у **конечного автомата**.
select() предоставляет интерфейс для работы с множествами дескрипторов.
Существуют и другие способы мультиплексирования ввода-вывода, например, функции WaitFor**().

Асинхронный ввод-вывод позволяет выполнять операции ввода-вывода параллельно основному потоку выполнения программы, улучшая производительность.

1. **Инициализация асинхронного ввода-вывода**:
   При открытии файла с флагом FILE_FLAG_OVERLAPPED операции ReadFile() и WriteFile() выполняются асинхронно.
2. **Структура OVERLAPPED**:

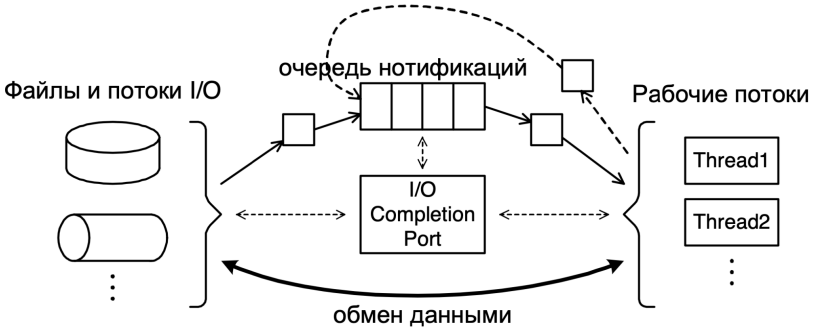
Содержит информацию о каждой операции ввода-вывода, включая позицию начала обмена, количество обрабатываемых байт и код результативности.

3. **Способы контроля состояния операции**:

- "**Ручная**" проверка: использование функций и макросов для анализа структуры OVERLAPPED.
- **Функции ожидания**: WaitFor***(), применяемые к дескриптору файла, позволяют дождаться завершения операции.
- **Использование объектов Event**: События — это синхронизирующие механизмы, которые позволяют потокам взаимодействовать друг с другом и координировать выполнение.
- **Callback-обработчики**: использование расширенных версий функций, таких как Read/WriteFileEx() и WaitFor***Ex(), с возможностью указания callback-функций для обработки завершения операций.

Асинхронный ввод-вывод требует аккуратного управления состоянием операций для обеспечения целостности данных и правильной синхронизации при параллельном выполнении множества операций.

Порты завершения ввода-вывода (I/O Completion Ports) - механизм Windows для управления многопоточной реализацией ввода-вывода. Позволяет выполнять операции ввода-вывода без блокировки потоков.



Использование портов завершения ввода-вывода

Использование файлов устройств позволяет обращаться к физическим устройствам, учитывая их особенности, такие как тайминги, протоколы обмена и инструкции ЦП. Унификация доступа через файлы упрощает задачу для прикладных программ. Примеры таких файлов включают stdin, stdout, stderr.

Для упрощения работы с устройствами, система может предоставлять специализированные API. Например, WinAPI имеет функции для работы с асинхронным последовательным интерфейсом (UART, RS232, COM) через функции -**Comm**-: SetupComm(), GetCommState(), SetCommState(), GetCommTimeouts(), SetCommTimeouts() и другие. Структуры **DCB** и **COMMTIMEOUTS** содержат параметры устройства.