**USB** расшифровывается как **Universal Serial Bus** (универсальная последовательная шина). Это стандартный интерфейс, который позволяет подключать различные устройства к компьютеру и другим устройствам для передачи данных и электропитания.

1. БИБЛИОТЕКА LIBUSB USB

(Universal Serial Bus) – последовательный интерфейс передачи данных для среднескоростных и низкоскоростных периферийных устройств. Стандарт USB ввел единообразие в работу с широким спектром периферийных устройств. До его появления использовалось множество разных интерфейсов для подключения периферийных устройств (последовательный порт RS-232C для модема, мыши, параллельный порт для принтера и дисководов IOMEGA ZIP, PS/2 для мыши и клавиатуры, специализированные интерфейсы для подключения сканеров и т. д.).

Программирование взаимодействия с USB-устройствами достаточно трудоемко. Для упрощения работы с USB-устройствами из прикладных программ была создана многоплатформенная библиотека libusb. Она реализует универсальный низкоуровневый интерфейс, который позволяет взаимодействовать со всеми возможными USB-устройствами. С помощью библиотеки libusb можно получить список доступных USB-устройств, узнать их параметры, обмениваться данными с устройствами в синхронном и асинхронном режимах, реагировать на подключение и отключение устройств.

Каждое конкретное устройство имеет свой специальный протокол взаимодействия, и прикладная программа должна взаимодействовать с устройством в соответствии с его протоколом, используя низкоуровневые операции библиотеки libusb.

1. ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ДОСТУПНЫХ USB-УСТРОЙСТВАХ С ПОМОЩЬЮ БИБЛИОТЕКИ LIBUSB

Библиотека libusb предоставляет следующие функции, которые позволяют получить информацию об имеющихся в системе USBустройствах:

• libusb\_init – инициализация работы с libusb,

• libusb\_exit – завершение работы с libusb,

• libusb\_set\_debug – установка уровня подробности отладочных сообщений (рекомендуется использовать уровень 3),

• libusb\_get\_device\_list – получение списка подключенных к машине USB устройств,

• libusb\_free\_device\_list – освобождение памяти, выделенной в функции libusb\_get\_device\_list для хранения данных со списком устройств,

• libusb\_get\_device\_descriptor – получение дескриптора USB устройства,

• libusb\_get\_config\_descriptor – получение дескриптора конфигурации USB устройства,

• libusb\_free\_config\_descriptor – освобождение памяти, выделенной в функции libusb\_get\_config\_descriptor,

• libusb\_ref\_device – увеличение счетчика числа пользователей устройства на 1 (при первом вызове функции libusb\_get\_device\_list после подключения устройства его счетчик устанавливается в 1),

• libusb\_unref\_device – уменьшение счетчика числа пользователей устройства на 1 (если счетчик уменьшается до нуля, дескриптор устройства удаляется),

• libusb\_open – открыть устройство (начать работать с устройством) и получить дескриптор устройства, который далее можно использовать для ввода/вывода данных,

• libusb\_open\_device\_with\_vid\_pid – открыть устройство по его идентификаторам производителя и изделия,

• libusb\_close – закрыть устройство после его использования,

• libusb\_get\_string\_descriptor\_ascii – получить дескриптор устройства в виде строки символов,

• lib\_usb\_error\_name – преобразование кода ошибки библиотеки libusb в строковое сообщение об ошибке.

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Что такое libusb? Какие задачи решает libusb?

**libusb** - это кроссплатформенная библиотека с открытым исходным кодом, которая предоставляет API для взаимодействия с USB-устройствами на низком уровне. Она позволяет разработчикам создавать программы, которые могут обнаруживать, конфигурировать, передавать данные и управлять USB-устройствами без необходимости знания специфичных для операционной системы деталей.

**Основные особенности libusb:**

* **Кроссплатформенность:** libusb поддерживает Windows, macOS, Linux и другие Unix-подобные операционные системы.
* **Открытый исходный код:** Библиотека распространяется под лицензией LGPL, что позволяет свободно использовать её в коммерческих и некоммерческих проектах.
* **Низкоуровневый доступ:** libusb предоставляет прямой доступ к функциям USB-устройств, что позволяет реализовывать сложные сценарии взаимодействия.
* **Поддержка широкого спектра устройств:** libusb может работать с большинством USB-устройств, включая HID, CDC, Mass Storage и другие.

1. На какие группы можно разбить функции libusb?

Функции libusb можно разделить на несколько групп в зависимости от их назначения:

* **Инициализация и завершение работы:**
  + libusb\_init() - Инициализация библиотеки.
  + libusb\_exit() - Завершение работы библиотеки.
* **Обнаружение устройств:**
  + libusb\_get\_device\_list() - Получение списка подключенных USB-устройств.
  + libusb\_get\_device\_descriptor() - Получение дескриптора устройства.
  + libusb\_get\_config\_descriptor() - Получение дескриптора конфигурации устройства.
* **Открытие и закрытие устройств:**
  + libusb\_open() - Открытие устройства для взаимодействия.
  + libusb\_close() - Закрытие устройства.
* **Конфигурирование устройств:**
  + libusb\_set\_configuration() - Установка активной конфигурации устройства.
  + libusb\_claim\_interface() - Захват интерфейса устройства.
  + libusb\_release\_interface() - Освобождение интерфейса устройства.
* **Передача данных:**
  + libusb\_control\_transfer() - Выполнение управляющего передачи данных.
  + libusb\_bulk\_transfer() - Выполнение блочной передачи данных.
  + libusb\_interrupt\_transfer() - Выполнение прерывистой передачи данных.
  + libusb\_get\_string\_descriptor() - Получение строкового дескриптора устройства.
* **Обработка событий:**
  + libusb\_handle\_events() - Обработка событий USB.
  + libusb\_set\_pollfd\_notifiers() - Установка обработчиков событий для файловых дескрипторов.

1. Как с помощью библиотеки libusb получить состав и конфигурацию USB-устройств?

Для получения состава и конфигурации USB-устройств с помощью libusb можно использовать следующий алгоритм:

1. **Инициализация библиотеки:**

libusb\_context \*context;

libusb\_init(&context);

1. **Получение списка устройств:**

libusb\_device \*\*device\_list;

ssize\_t device\_count = libusb\_get\_device\_list(context, &device\_list);

1. **Перебор устройств и получение их дескрипторов:**

for (ssize\_t i = 0; i < device\_count; i++) {

libusb\_device \*device = device\_list[i];

libusb\_device\_descriptor device\_desc;

libusb\_get\_device\_descriptor(device, &device\_desc);

// Вывод информации о устройстве (например, VID, PID, производитель, продукт)

printf("Device %04x:%04x\n", device\_desc.idVendor, device\_desc.idProduct);

// Получение дескриптора конфигурации

libusb\_config\_descriptor \*config\_desc;

libusb\_get\_config\_descriptor(device, 0, &config\_desc);

// Перебор интерфейсов и их эндпоинтов

for (uint8\_t i = 0; i < config\_desc->bNumInterfaces; i++) {

const libusb\_interface \*interface = &config\_desc->interface[i];

for (int j = 0; j < interface->num\_altsetting; j++) {

const libusb\_interface\_descriptor \*interface\_desc = &interface->altsetting[j];

printf("Interface %d, Alt setting %d\n", interface\_desc->bInterfaceNumber, interface\_desc->bAlternateSetting);

for (uint8\_t k = 0; k < interface\_desc->bNumEndpoints; k++) {

const libusb\_endpoint\_descriptor \*endpoint\_desc = &interface\_desc->endpoint[k];

printf("Endpoint %02x, Address %02x, Type %d\n", endpoint\_desc->bEndpointAddress, endpoint\_desc->bmAttributes & 0x03);

}

}

}

// Освобождение дескриптора конфигурации

libusb\_free\_config\_descriptor(config\_desc);

}

1. **Освобождение списка устройств и завершение работы библиотеки:**

libusb\_free\_device\_list(device\_list, 1);

libusb\_exit(context);

Коды классов USB устройств

00h – код отсутствует (информацию о классе нужно получать в дескрипторе интерфейса) 01h – аудиоустройство (если код получен из дескриптора интерфейса, а не устройства)

02h – коммуникационное устройство (сетевой адаптер)

03h – устройство пользовательского интерфейса

05h – физическое устройство 06h – изображения

07h – принтер 08h – устройство хранения данных

09h – концентратор

0Ah – CDC-Data

0Bh – Smart Card

0Dh – Content Security

0Eh – видеоустройство

0Fh – персональное медицинское устройство

10h – аудио- и видеоустройства

DCh – диагностическое устройство

E0h – беспроводный контроллер

EFh – различные устройства

FE – специфическое устройство

Дескрипторы конфигурации содержат сведения о каждой конфигурации устройства. Дескрипторы строк содержат текстовые строки в Юникоде. Каждое USB-устройство предоставляет дескриптор устройства, который указывает сведения о классе устройства, идентификаторы поставщика и продукта, а также количество конфигураций.

Дескриптор – это некоторое описание, в нашем случае – описание конкретного свойства устройства.

Корневые хабы являются важной частью архитектуры USB, так как они обеспечивают подключение и управление другими устройствами.

Libusb отвечает за работу на низком уровне с usb устройствами, не требует какой-то установки драйверов, оно непосредственно считывает устройства, может считать информацию о нем, может отправить ему данные, может получить с него данные (не требует url самого устройства?)

Нужно создать контекст – условие для работы libusb и инициализация libusb, после получаем список всех устройств (функция определенная) и кол-во устройств. Для каждого устройства инфа:получаю дескриптор устройства - уникальный номер каждого устройства, содержащий информацию о том, как к нему подключаться и как с ним работать(vendor id – идентификатор производителя, product id – идентификатор продукции этого производителя), дальше открытие устройства, в котором пытаемся получить мануфактуру, производителя, саму продукцию и серийный номер. Функция с ascii тоже по сути получение дескриптора, но получение дескриптора в виде строки, потом надо обязательно закрыть устройство, которое мы открывали. Дескриптор возвращал класс устройства(получение класса словарь и по номеру, которое возвращает я получаю что это за класс). В конце программы освободить список весь устройств и закрыть libusb.

Что полезного можно получить через usb устройства – получение движения курсора мыши к примеру.

**1. Данные с устройств ввода**

* **Клавиатуры и мыши**: Можно получать данные о нажатиях клавиш и движениях мыши.
* **Геймпады и джойстики**: Можно получать данные о положении осей и нажатиях кнопок.
* **Сканеры**: Можно получать изображения и данные с сканируемых документов.

**2. Данные с мультимедийных устройств**

* **Веб-камеры**: Можно получать видеопотоки и изображения.
* **Микрофоны**: Можно получать аудиоданные.
* **Наушники и колонки**: Можно получать данные о состоянии и настройках аудиоустройств.

**3. Данные с накопителей**

* **USB-флешки и внешние жесткие диски**: Можно получать доступ к файлам и данным, хранящимся на этих устройствах.
* **Картридеры**: Можно получать данные с карт памяти.

Шина в usb:

Универсальная последовательная шина (USB) Dell USB — это тип входа, который позволяет подключать периферийные устройства (например, мыши, клавиатуры, флешки, внешние жесткие диски...) к компьютеру или другим устройствам . В настоящее время мы работаем с ПК, которые оснащены портами USB в моделях 2.0, 3.1 и Type C.

**USB** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Universal Serial Bus* — «универсальная последовательная шина») — последовательный [интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) для подключения периферийных устройств к [вычислительной технике](https://ru.wikipedia.org/wiki/%D0%AD%D0%92%D0%9C). Получил широчайшее распространение и стал основным интерфейсом подключения периферии к бытовой цифровой технике.

Интерфейс позволяет не только обмениваться данными, но и обеспечивать электропитание периферийного устройства. Сетевая архитектура позволяет подключать большое количество периферии даже к устройству с одним разъёмом USB.

Разработка спецификаций USB производится в рамках международной некоммерческой организации [USB Implementers Forum](https://ru.wikipedia.org/wiki/USB_Implementers_Forum) (USB-IF), объединяющей разработчиков и производителей оборудования с шиной USB. В процессе развития выработано несколько версий спецификаций

В контексте библиотеки libusb, **хэндл** (handle) — это указатель на открытое устройство, который используется для выполнения операций с этим устройством. Хэндл позволяет вам взаимодействовать с устройством, отправляя запросы, получая данные и управляя его состоянием.

Когда вы открываете устройство с помощью функции libusb\_open, вы получаете хэндл, который затем используется для всех последующих операций с этим устройством. После завершения работы с устройством, хэндл должен быть закрыт с помощью функции libusb\_close.

### Как получается класс устройств?

Класс устройства (Device Class) — это параметр, который определяет общий тип устройства в соответствии со стандартами USB. Класс устройства указывает на функциональность устройства, например, является ли оно клавиатурой, мышью, веб-камерой, накопителем и т.д.

Класс устройства можно получить из дескриптора устройства (device descriptor), который содержит информацию о характеристиках устройства.

Класс устройства определяется в поле bDeviceClass дескриптора устройства.

OpenCV и libusb — это две разные библиотеки, которые решают разные задачи и работают на разных уровнях абстракции. Давайте разберем, почему OpenCV считается более высоким уровнем общения с периферийными устройствами по сравнению с libusb.

### 1. ****Уровень абстракции****

* **libusb**: Это низкоуровневая библиотека, которая предоставляет прямой доступ к USB-устройствам. Она работает на уровне USB-протокола и позволяет вам управлять устройствами, отправляя и получая данные напрямую через USB-интерфейс. libusb требует от разработчика глубокого понимания USB-протокола, структуры устройств и управления передачей данных.
* **OpenCV**: Это высокоуровневая библиотека для компьютерного зрения и обработки изображений. Она предоставляет готовые функции и алгоритмы для работы с изображениями, видеопотоками и другими данными. OpenCV абстрагирует сложные операции, такие как чтение видеопотока с камеры, обработка изображений и распознавание объектов, делая их доступными для разработчиков без необходимости понимать низкоуровневые детали.

### 2. ****Функциональность****

* **libusb**: Предоставляет функции для открытия устройств, отправки и получения данных, управления конфигурациями и интерфейсами. Разработчик должен самостоятельно реализовывать все операции, связанные с устройством, включая обработку ошибок и управление ресурсами.
* **OpenCV**: Предоставляет готовые функции для работы с видеопотоками, изображениями и другими данными. Например, для чтения видеопотока с веб-камеры достаточно вызвать функцию cv2.VideoCapture(0), которая скрывает все сложности работы с USB-интерфейсом и предоставляет удобный интерфейс для работы с видеоданными.

### 3. ****Примеры использования****

#### Пример с libusb

Чтобы прочитать данные с USB-устройства, например, веб-камеры, с использованием libusb, вам нужно:

1. Инициализировать библиотеку.
2. Найти устройство по его Vendor ID и Product ID.
3. Открыть устройство.
4. Установить конфигурацию и запросить права на интерфейс.
5. Отправить запрос на чтение данных.
6. Обработать полученные данные.

#### Пример с OpenCV

Чтобы прочитать видеопоток с веб-камеры с использованием OpenCV, достаточно вызвать несколько функций:

### 4. ****Удобство использования****

* **libusb**: Требует от разработчика глубокого понимания USB-протокола и работы с низкоуровневыми функциями. Это может быть сложно и требовательно к ресурсам.
* **OpenCV**: Предоставляет высокоуровневый интерфейс, который скрывает сложности работы с USB-устройствами и предоставляет удобные функции для работы с изображениями и видеопотоками. Это делает OpenCV более удобным и быстрым в использовании для задач компьютерного зрения.

### Заключение

OpenCV считается более высоким уровнем общения с периферийными устройствами по сравнению с libusb, потому что она предоставляет высокоуровневые функции и абстракции, которые скрывают сложности работы с низкоуровневыми протоколами и интерфейсами. Это делает OpenCV более удобным и эффективным инструментом для задач компьютерного зрения и обработки изображений.

### Как работает вывод серийного номера устройства

Вывод серийного номера устройства через libusb включает несколько шагов:

1. **Получение дескриптора устройства**: Сначала необходимо получить дескриптор устройства, который содержит информацию о характеристиках устройства, включая серийный номер.
2. **Открытие устройства**: Для работы с устройством его необходимо открыть с помощью функции libusb\_open.
3. **Получение строки серийного номера**: Используя функцию libusb\_get\_string\_descriptor\_ascii, можно получить серийный номер устройства в виде строки.
4. **Закрытие устройства**: После завершения работы с устройством его необходимо закрыть с помощью функции libusb\_close.