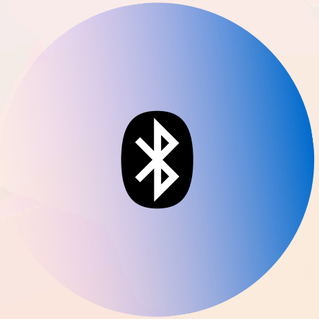
Peripheral Identification and Operations

Github Link

The aim of this project is to create a program in C that helps users detect peripheral devices like USBs, Bluetooth, and Wi-Fi on a Linux system. It allows listing connected devices, showing detailed information, and performing actions like mounting, unmounting, ejecting, and connecting Peripheral devices.



Team 6 :
1.) Jaiswal Aditya Ranjit (B22CS025)
2.) Lakavat Umesh Chandra (B22CS029)
3.) Satyam Sharma (B22CS047)
4.) Neielotpal Rao (B22EE038)

Basic Information

Peripheral Devices

Peripheral devices are external hardware components connected to a computer to extend its capabilities. Examples include:

- USB Devices: Flash drives, external storage, and input devices like keyboards.
- Bluetooth Devices: Wireless headphones, keyboards, and mice.
- Wi-Fi Devices: Adapters and routers for internet connectivity.

Configuring a Device

- Configuration refers to setting up a device to work with the computer. For example, connecting wireless peripherals like a Bluetooth mouse or adjusting network settings for a Wi-Fi adapter.

Mounting and Unmounting

- Mounting: The process of making a storage device accessible by attaching it to the system's file structure. Once mounted, the device becomes ready for reading or writing data.
- Unmounting: Unmounting safely disconnects a device from the OS file system, stopping access to its files. The device remains physically connected and is still tracked by the OS, but it can no longer be used until remounted or ejected.

Ejecting

- Ejecting ensures all ta.sks involving the device are completed. After ejecting, the device is fully removed from the OS's connected devices list, making it safe to physically disconnect. Os will no longer keep track of it

Task 1: Detecting and identifying connected peripheral devices

Key Commands

popen() function

popen() function is used to establish Inter-Process Communication (IPC) between the program and system shell commands like lsusb
It runs the command you pass and allows you to read its output or send input to it. This creates a unidirectional pipe that allows the program to read the output of these commands and process device-related information

popen("lsusb", "r");

popen("lsusb", "r");   "lsusb": This is the command being executed. The lsusb command lists all USB devices connected to the system.

popen("bluetoothctl devices", "r")

the program runs this command and allows it to capture the list of paired Bluetooth devices. bluetoothctl is a command-line utility that interacts with the Bluetooth service outputs information such as the device's address and name.

popen("iwconfig", "r")

iwconfigs used to configure and display information about wireless network interfaces ,information about all the wireless interfaces on the system, such as their status, SSID (network name)
fclose(fp) function is used to close the file pointer and release the resources associated with the pipe created by popen()

Task 2: Displaying detailed information about each device

Key Commands

lsusb -v -d <vendor:product>

lsusb : This is the basic command used to list all USB devices connected to the system.
-v: -v flag stands for "verbose," which tells the lsusb command to provide detailed information about each USB device. Gives much more than just the vendor and product IDs; it shows descriptors, configuration settings, and detailed device attribute
-d <vendor_product_id> : -d option filters the output to show only the device with the specified vendor ID and product ID.The <vendor_product_id> is a combination of the two IDs (e.g., 8087:0a2b)

bluetoothctl info <device_address>

bluetoothctl :A command-line utility to manage Bluetooth devices in Linux,allows you to connect, pair, scan, and retrieve information about Bluetooth devices.
i**nfo** The info subcommand fetches detailed information about a Bluetooth device identified by its device address. The address format is usually XX:XX:XX:XX:XX,Device Name, Device Address, Paired Status, Connected Status etc
<device_address>: This represents the specific Bluetooth device you want to query.bluetoothctl devices command, which lists all paired or visible devices.

popen("iwconfig", "r")

It is same command used in task 1 which gives detailed info about Wi-Fi interface details: signal strength, quality, rate, mode, and ESSID.

Task 3: Allowing users to perform operations such as ejecting, mounting, or configuring these devices.

Key Commands

sudo mount <device_path> <mount_point>

This command is used to mount a device at a specified mount point, making its file system accessible to the OS.

sudo umount <mount_point>

This command is used to mount a device at a specified mount point, making its file system accessible to the OS.

sudo eject <device_path>

This command ejects the device, ensuring all processes are completed before safely removing the device from the OS's tracking.

bluetoothctl pair <device_address>

This command is used to mount a device at a specified mount point, making its file system accessible to the OS.

bluetoothctl trust <device_address>

Trusting a device ensures it can automatically reconnect in the future without manual intervention.

bluetoothctl connect <device_address>

This command connects to the Bluetooth device using its MAC address.

Learnings

Learnings

1. Understanding Peripheral Devices:
   - Gained knowledge about different types of peripherals such as USB, Bluetooth, and Wi-Fi devices.
   - Learned how operating systems interact with these devices.
2. Linux Commands and Utilities:
   - Explored commands like lsusb, iwconfig, bluetoothctl, and their role in detecting and managing devices.
   - Understood how to utilize shell commands programmatically through C code.
3. File System Management:
   - Learned the difference between mounting, unmounting, and ejecting devices.
   - Understood how the OS keeps track of devices and ensures data integrity during operations.
4. System Programming in C:
   - Worked with system calls and external commands using the system() function.
   - Improved coding skills by structuring modular programs with reusable functions.

Thank You.

Thank You.

Thank You.