Project Proposal: Chimeric File System Device (CFSD): Lyell Read

Tl;dr (Summary)

CFSD (Chimeric File System Device) is a concept USB[1] mass storage device that addresses a multitude of serious issues with current FS[2] specifications. CFSD aims to replace the incompatibility of different FS's on removable drives.

Existing Problems

Removable storage devices work great as long as you only use them between computers that share a FS standard. Sometimes, you just need a USB stick to get a file from your computer to one that you do not natively share FS drivers with. In that case, you're either out of luck, or you'll find yourself installing drivers or third party packages to access your USB stick.

OS/FS Compatibility[3]						
File System	**Windows XP**	**Windows 7/8/10**	**macOS (<10.6.4)**	**macOS (>10.6.5)**	**Ubuntu Linux**	**Android**
NTFS	Yes	Yes	No	No	Yes	No
FAT32	Yes	Yes	Yes	Yes	Yes	Yes[4]
exFAT	Yes	Yes	No	Yes	No	No
HFS+	No	No	Yes	Yes	Yes	No
APFS	No	No	No	>10.13	No	No
EXT 2, 3, 4	No	No	No	No	Yes	No

Though it may seem like FAT32 is the common denominator, it has the limitation of not being able to support files larger than 4GB - a great downfall in today's age of ever-expanding data.[5]

Second to FAT32, the FS with the best cross-platform performance is exFAT. ExFAT has a significant downfall when it comes to Linux. It typically (varying by flavor) requires a third party driver to be accessed. Though a driver is a simple thing to install when online, there are a multitude of scenarios (from establishments that cannot trust foreign code, even open source code, to offline users) where downloading a driver is impossible, or against the rules.

---

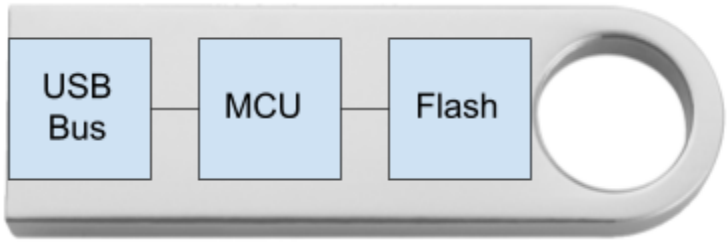
[1] Universal Serial Bus
[2] File System
[3] Table adapted from https://www.howtogeek.com/73178/what-file-system-should-i-use-for-my-usb-drive/; Only plug and play, read & write considered to be 'compatible'.
[4] https://www.androidcentral.com/how-connect-usb-flash-storage-device-your-android-phone
[5] https://en.wikipedia.org/wiki/File_Allocation_Table#FAT32

Can't we just ignore Linux users, though? No. Not only has Linux has been gaining traction in recent years, but the OS[6] with the largest market share, android[7], is Linux- based. Further, given Linux's prevalence in personal computers[8] and servers[9], compatibility with linux can no longer be ignored.

Solutions

There are a couple different ways to solve this problem. The ideal scenario would be if the manufacturers combined their technical skill with the open source community to create the ideal File System, then deployed FS drivers for this collective FS in their operating systems respectively, there would be no compatibility. This will never happen because the companies that control market share (Microsoft, Apple, specifically) are historically closed-source companies.

The other solution is CFSD. A chameleon USB drive compatible with all FS. Below is a simplified diagram of how CFSD would be designed:



Internally, the CFSD would consist of a Microcontroller (MCU) that handles requests from the host computer. The MCU pulls or modifies the data in the flash array, responding with the appropriate message/data that the host asked for, in the format the host expects. Therefore, this USB stick is able to effortlessly masquerade as most common File Systems at once.

More Details & Software Diagram

The CFSD drive will work in the following way:

1. Plug into computer. This powers on the MCU.
2. MCU detects the host OS using tactics similar to a Rubber Ducky[10,11]
3. MCU selects most appropriate FS to use given detected OS, as it responds to the host's requests

---

[6] Operating System
[7] https://www.computerhope.com/issues/ch001777.htm, android phones claim 37.89% of all computers
[8] https://beav.es/ZyJ, 1.93% OS Market Share for Linux
[9] https://hostingtribunal.com/blog/linux-statistics/, "96.3% of the world's top 1 million servers run on Linux"
[10] https://shop.hak5.org/products/usb-rubber-ducky-deluxe, https://stackoverflow.com/questions/30208215/what-data-can-a-hid-device-receive/30362736#30362736
[11] https://patents.google.com/patent/US20120054372, patent only marginally related.

To achieve this chameleon-like behavior, the MCU will need to be programmed in a way such that it can achieve the aforementioned steps, as well as access and modify data stored on a potentially large flash array. Further this MCU will have to be user-updatable, as OS's and FS's change over time, and the behavior of the MCU must update to keep the chip functioning.

Therefore, the following overall design displays how the CFSD will work: