# IT doupě

# Jak na… Media Player (v terminálu)

Jestli si ještě pamatujete na úplně první díl elektronky (já vím že ne), vydal jsem v něm článek o tzv. *Bad Apple Challenge*, ve které se vývojáři snaží přehrát hudební klip stejnojmenné písničky na čemkoliv, co alespoň zdánlivě připomíná displej. Tento nápad se mi tak zalíbil, že jsem se rozhodl také zúčastnit, a protože používám linux btw, k tomuto účelu jsem použil terminál. Dnes vám ukážu, jak to můžete udělat taky a pokud pochopíte princip, i vy budete moc Bad Apple přehrát opravdu všude.

Na projekt jsem použil Rust a krátil si ním letní prázdniny. Práci s terminálem zajišťuje knihovna *crossterm*, zvuk *rodio*, *image* pracuje s pixely, *eventual* má časovač a *dirs* ukládá soubory. Na konverzi mezi videem a sekvencí skvěle posloužil program ffmpeg, který si můžete stáhnout zde: <https://ffmpeg.org/>.

Kódu tu mám dnes opravdu hodně, proto začneme pěkně od začátku. Program se otevírá v terminálu, takže má bash podporu a pomocí **env::args()** z něj můžeme vytáhnout zadané argumenty, v našem případě cestu k videu, které chceme přehrát. Rovnou můžeme prověřit, zda se jedná o video soubor. Je taky celkem důležité zkontrolovat pomocí **Command::new().output()** přítomnost ffmpegu.

A můžeme konvertovat - z videa uděláme kvůli extrakci snímků gif (dá se použít i mp4, má čisté barvy, ale vyžaduje mnohem více kódu) o velikosti kolem 80x80 pixelů (čím větší, tím horší výkon) a 20 FPS a uložíme ho do nějaké složky, která nejlépe není moc na očích. To samé uděláme s audiem, to má formát .mp3. Teď přijde na řadu knihovna crossterm a její metoda stdout(), která nám umožní upravit úplně celý terminál. Zatím přepneme na „alternativní obrazovku“, vypneme kurzor a zapneme raw mode. Nahrajeme překonvertovaný gif soubor do paměti, zapneme zvuk a dostáváme se k té nejzajímavější části.

Ve funkci generate\_frame() opět použijeme stdout(), což společně s enable\_raw\_mode() o pár řádků výš mnohonásobně zefektivní renderování pixelů a umožní to předchozí snímek místo posunutí nahoru přepsat. Když jsme u těch pixelů, každý z nich má 4 data (rgb a průhlednost), která jsou všechna uložena v proměnné frame. Abysme se k nim dostali, pomocí šířky snímku \* 4 definujeme řádek a ten po 4 rozdělíme na jednotlivé pixely. Jejich data jsou ovšem jenom čísla, aby se vykreslily barevně, použijeme escape sekvenci \x1b[38;2;***a***;***b***;***c***m██, kde abc jsou hodnoty pixelů a ██ je unicode znak, který bude náš pixel reprezentovat. Všechno to uložíme do stringu a jakmile bude hotový celý řádek, posune se kurzor terminálu o bod níž a celý se vyrenderuje. Nezapomeňte někam na konec programu umístit end\_process(), jinak bude vaše aktuální terminálová relace nepoužitelná.

A jako vždy, tohle je jen začátek – je opravdu hodně věcí, co přidat, například caching videí, celobarevnou podporu (mp4 místo gif), podporu klávesových zkratek (teď nefunguje ani ctrl+c), paralelní načítání do paměti, text o aktuálním času videa, škálovatelnost podle velikosti terminálu, změnu hlasitosti a mnohem víc. Protože má kód přes sto řádků a já mám rád lesy, místo pod článkem ho najdete zde: <https://github.com/dlabaja/elektronka_media_player>. Mám tam také repozitář s názvem TerminalMediaPlayer, kde jsem vše, co jsem výše vypsal, implementoval. To je pro tento měsíc vše a já se loučím s lego animací naší školy, jak jinak než v terminálu.

//img1