Obsah

[Uživatelský manuál 3](#_Toc510509934)

[1 Úvod 4](#_Toc510509935)

[2 Instalace 4](#_Toc510509936)

[2.1 Instalace potřebných knihoven 4](#_Toc510509937)

[2.2 Instalace SoundiX 4](#_Toc510509938)

[2.2.1 Kompilace programu 4](#_Toc510509939)

[2.2.2 Před spuštěním 5](#_Toc510509940)

[3 Spuštění a používání aplikace SoundiX 5](#_Toc510509941)

[4 Licence 9](#_Toc510509942)

[5 Knihovny 9](#_Toc510509943)

[Programátorská dokumentace 10](#_Toc510509944)

[6 Úvod 11](#_Toc510509945)

[7 Struktura 11](#_Toc510509946)

[7.1 FFT2.cpp 11](#_Toc510509947)

[7.1.1 main() 11](#_Toc510509948)

[7.1.2 hann() 12](#_Toc510509949)

[7.1.3 FFT() 12](#_Toc510509950)

[7.1.4 transform() 12](#_Toc510509951)

[7.1.5 order() 12](#_Toc510509952)

[7.1.6 reverse() 12](#_Toc510509953)

[7.1.7 decomlex() 12](#_Toc510509954)

[7.1.8 filter() 12](#_Toc510509955)

[7.1.9 compare() 12](#_Toc510509956)

[7.1.10 vminmax() 12](#_Toc510509957)

[7.1.11 compress() 13](#_Toc510509958)

[7.1.12 freq() 13](#_Toc510509959)

[7.1.13 volume() 13](#_Toc510509960)

[7.2 OPEN2.cpp 13](#_Toc510509961)

[7.2.1 info() 13](#_Toc510509962)

[7.2.2 decode() 13](#_Toc510509963)

[7.3 FREQ2MIDI.cpp 14](#_Toc510509964)

[7.3.1 d2b() 14](#_Toc510509965)

[7.3.2 start() 14](#_Toc510509966)

[7.3.3 track() 14](#_Toc510509967)

[7.3.4 deltaTime() 14](#_Toc510509968)

[7.3.5 noteOn() 14](#_Toc510509969)

[7.3.6 noteOff() 14](#_Toc510509970)

[7.3.7 getFileSize() 14](#_Toc510509971)

[8 Poznámky autora 15](#_Toc510509972)

Uživatelský manuál

# Úvod

SoundiX je konzolová aplikace určená pro operační systém Linux na zpracování digitálního zvukového záznamu. Účelem této aplikace je překládat digitální audio soubory do smysluplného jednoduchého MIDI formátu.

# Instalace

Prvním krokem před samotnou instalací SoundiX je získání potřebných souborů a nainstalování potřebných knihoven.

## Instalace potřebných knihoven

V případě, že nemáte na svém počítači nainstalované knihovny *libsndfile1* a *libsndfile1-dev*, zadejte do Terminálu následující příkaz. (Pro spuštění Terminálu vyberte z menu Aplikace → Příslušenství → Terminál. Výchozí klávesová zkratka pro spuštění Terminálu je Ctrl+Alt+T, případně napište *gnome-terminal* do okna Spustit aplikaci (Alt+F2).)

**sudo apt-get install libsndfile1 libsndfile1-dev**

Pokud budete při instalaci dotázáni, zda chcete tuto knihovnu instalovat, napište do Terminálu **y** nebo **yes**. Můžete také použít příkaz **sudo apt-get install -y libsndfile1 libsndfile1-dev** (bude se chovat, jako by bylo na všechny otázky odpovídáno kladně). V tom případě sice nebudete dotazováni, jestli chcete tyto knihovny instalovat, ale můžete takto minout důležitou výstražnou správu.

## Instalace SoundiX

V případě, že jsou všechny potřebné knihovny nainstalovány, můžeme přejít k samotné instalaci SoundiX. Soubory nutné k instalaci najdete na přiloženém CD nebo na stránce <https://github.com/Radian-Fi/SoundiX>.

Tyto soubory si stáhněte, naklonujte (pomocí příkazu **git clone https://github.com/Radian-Fi/SoundiX.git "folder",** kde folder je jméno složky), nebo zkopírujte z přiloženého CD do vámi vybrané složky.

### Kompilace programu

Dalším krokem je kompilace programu. Do terminálu zadejte příkaz:

**cd "folder"**

a zmáčkněte Enter. "folder" je v tomto případě cesta ke složce, kam jste zkopírovali potřebné soubory (z předchozího kroku). Dále zadejte příkaz:

**sudo make all**

Pokud příkaz proběhne bez žádné chybové hlášky, je SoundiX úspěšně zkompilován. Pokud dojde k chybě, zkuste zjisti, zda máte nainstalovány všechny potřebné knihovny a proceduru opakujte.

### Před spuštěním

Spustitelný soubor SoundiX se po provedení kompilace nachází v složce bin (v instalační složce). Pokud ho chcete přemístit do jiné složky, je doporučováno udělat tak teď. Dobré místo pro uložení SoundiX je */usr/local/bin* nebo */usr/bin*. Samozřejmě je také možné SoundiX uložit do složky */bin*.

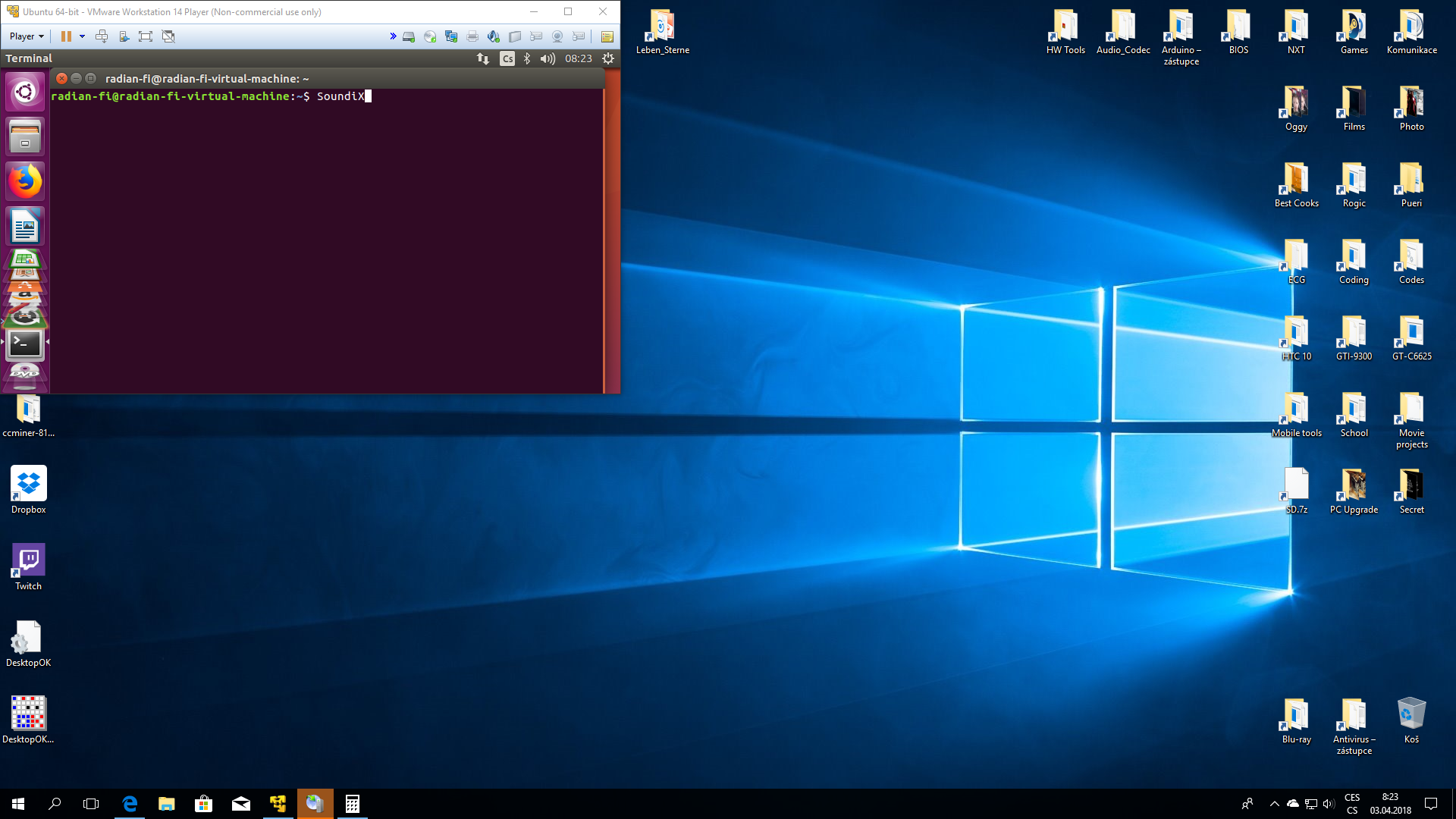
Pokud chcete, aby byla aplikace SoundiX spustitelná odkudkoliv, zadejte do Terminálu následující příkaz.

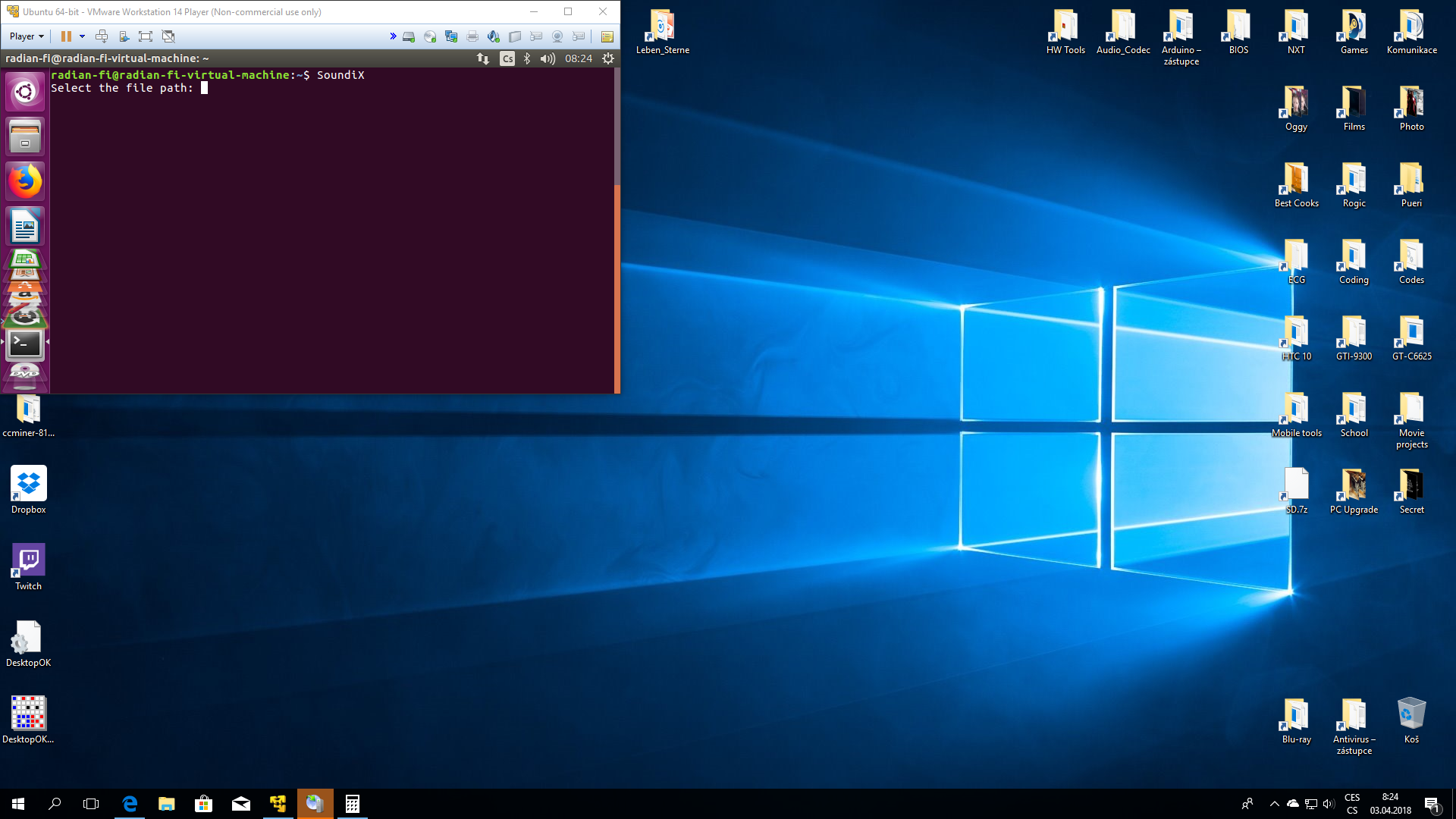
**export PATH=$PATH: "folder"**

"folder" je v tomto případě cesta ke složce, kde je uložen spustitelný soubor SoundiX. V následujících příkladech spuštění SoundiX se předpokládá, že tento příkaz byl použit, pokud tomu tak ve vašem případě není zaměňte vždy příkaz **SoundiX** za **./SoundiX**.

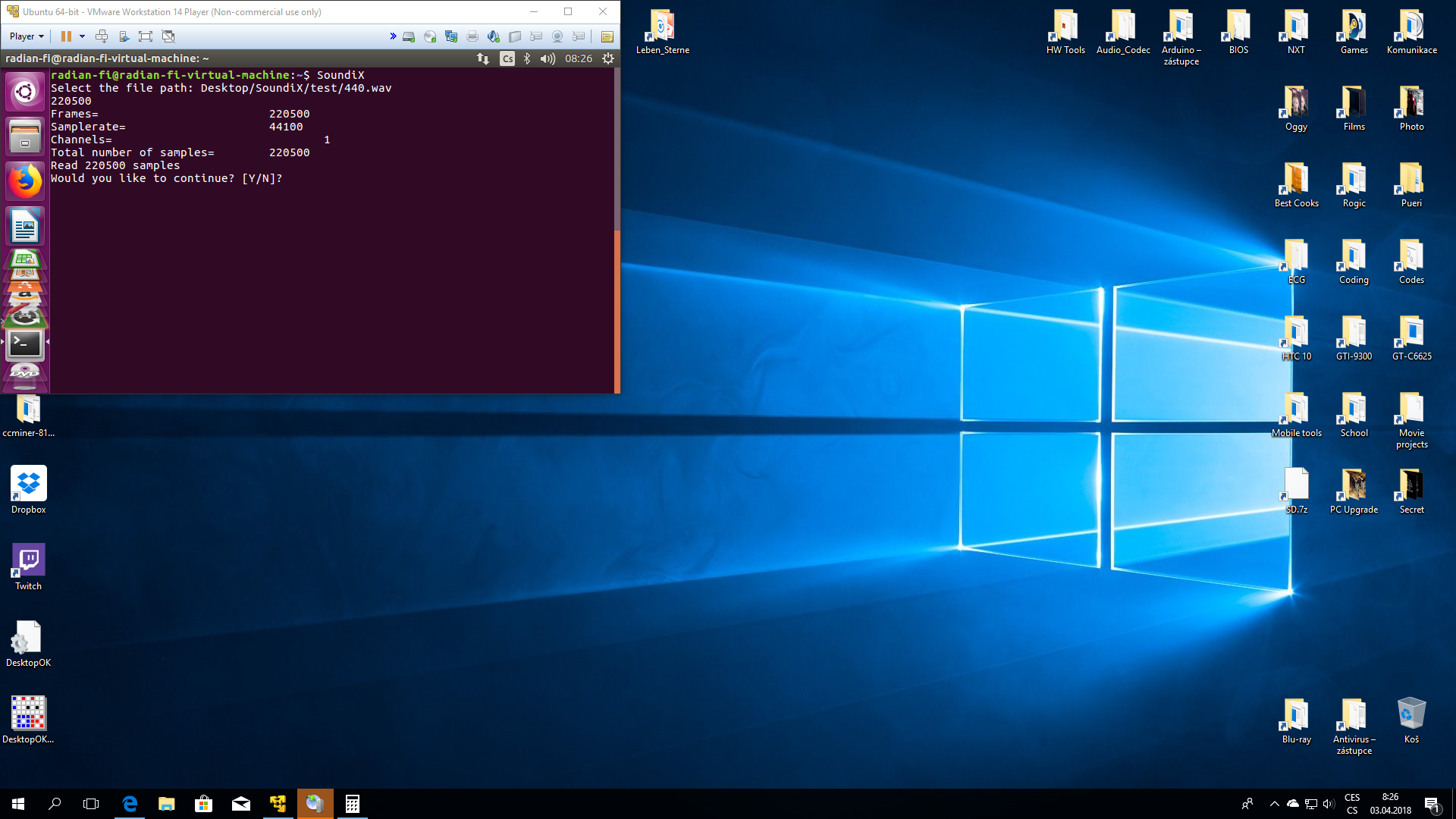
# Spuštění a používání aplikace SoundiX

Pro spuštění aplikace SoundiX zadejte do terminálu příkaz **SoundiX**. Následně budete dotázáni na zadání cesty k audio souboru, který chcete převést do MIDI formátu. Udělejte tak a klikněte na tlačítko Enter.



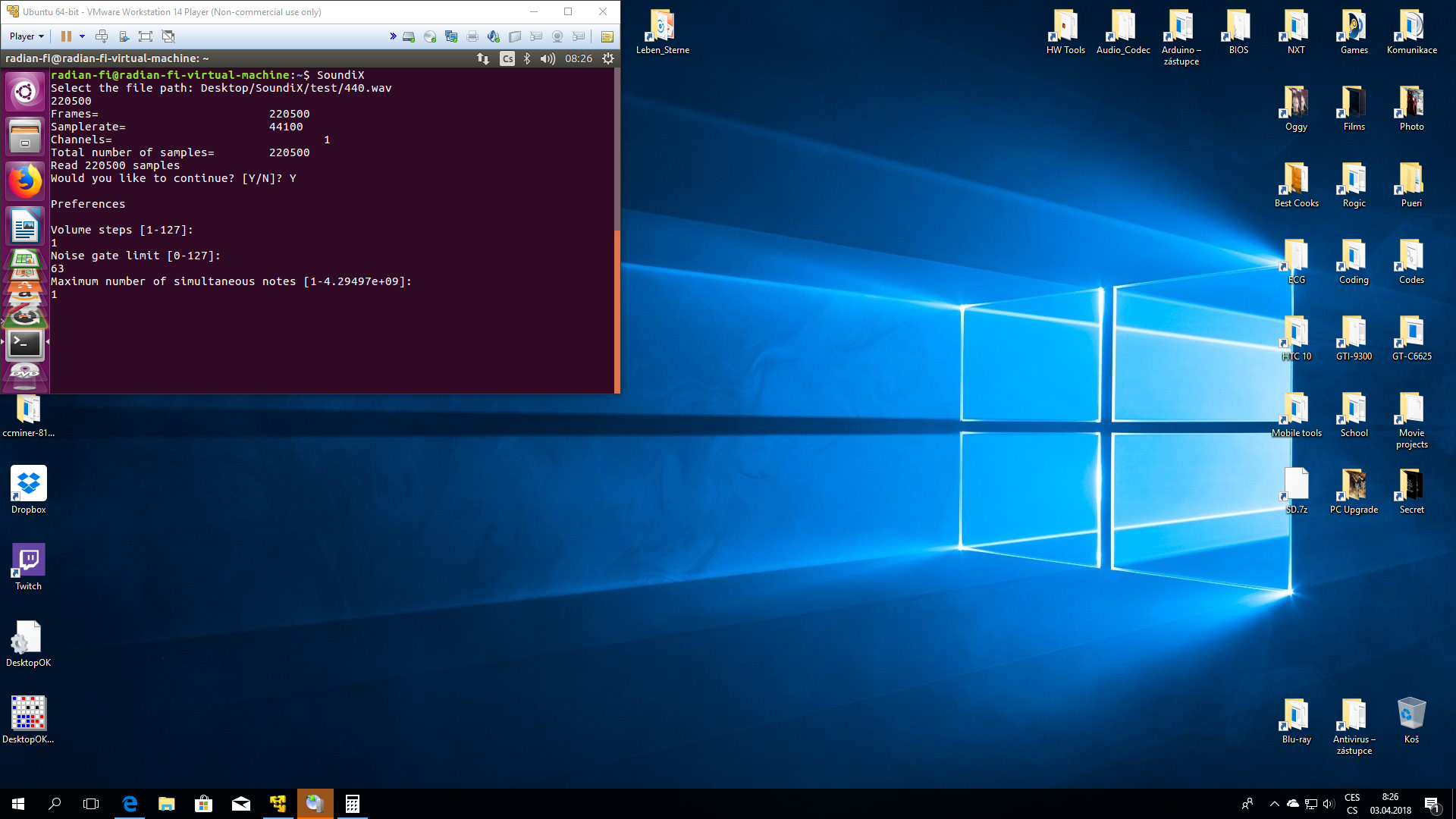


V Terminálu se zobrazí základní informace o daném audio souboru, pokud vše souhlasí, zadejte **yes**, **y** a klikněte na tlačítko Enter. Pokud data nesouhlasí a chcete program ukončit, zadejte **no**, **n**, nebo zmáčkněte **Ctrl+C**.



Dále budete dotázáni na 3 parametry, počet hlasitostních kroků (**Volume steps**, kolik má být v MIDI souboru rozdílných možností hlasitosti), limit šumu (**Noise gate limit**, hlasitost, pod kterou jsou zvuková data ignorována) a maximální počet současně hrajících not (**Maximum number of simultaneous notes**). **Volume steps** a **Noise gate limit** mají rozsah hodnot od 0 po 127. **Maximum number of simultaneous notes** má spodní limit 0, horní limit nemá (systémově stanovený na velikost integeru **int**, většinou 65 536).

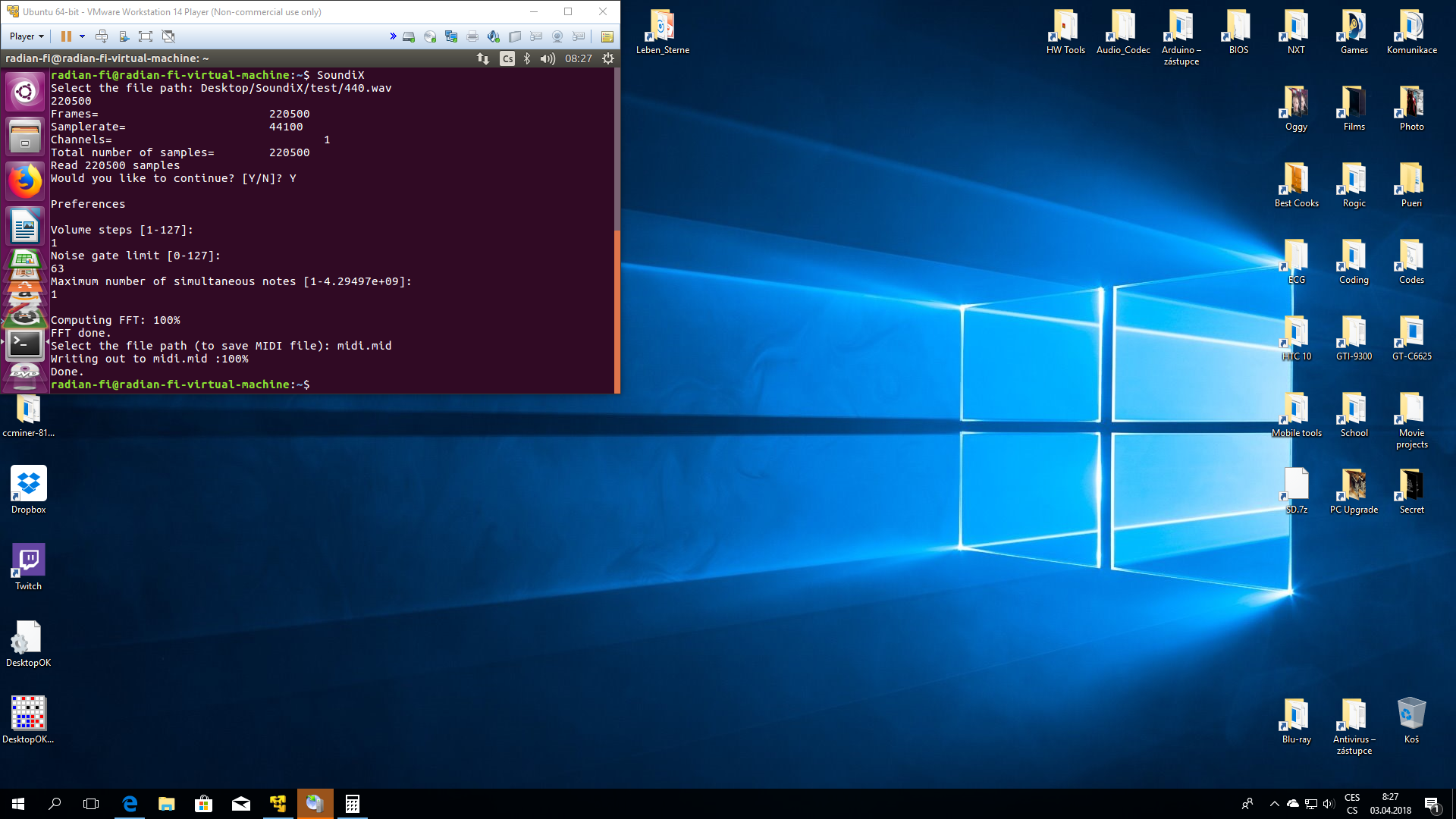
Doporučené hodnoty jsou **Volume steps: 1**, **Noise gate limit: 63**, **Maximum number of simultaneous notes: 1**. Výsledkem je poté jednoduchá melodie a srozumitelný notový zápis.



Proběhne matematická analýza zvukových dat (Rychlá Fourierova transformace).



Dále budete dotázáni, kam byste chtěli výstupní MIDI soubor uložit. Cestu i s názvem souboru a příponou .mid (nebo .midi) napište do Terminálu a stiskněte tlačítko Enter.



Poslední hláškou v Terminálu je **Done.** To znamená, že vše proběhlo v pořádku a MIDI soubor se uložil na předem zadané místo. Poté se aplikace ukončí.

# Licence

Aplikace SoundiX spadá pod standardní licenci GNU/GPL. Její plné znění můžete najít na stránce <https://github.com/Radian-Fi/SoundiX/blob/master/LICENSE> nebo v instalační složce v souboru LICENSE.md .

# Knihovny

Byly použity knihovny libsndfile, libsndfile-dev a pstreams. Všechny jsou volně dostupné pod otevřenými licencemi.

Programátorská dokumentace

# Úvod

SoundiX je konzolová aplikace na zpracování digitálního zvukového záznamu sepsaná v jazyce C++. Účelem této aplikace je překládat digitální audio soubory do smysluplného jednoduchého MIDI formátu.

# Struktura

Projekt je rozdělen na několik souborů (soubory obsahující zdrojový kód C++, hlavičkový soubor).

## FFT2.cpp

Tento soubor obsahuje jádro aplikace, hlavní funkci main() a samotné funkce pro analýzu zvuku.

### main()

Tato funkce slouží jako hlavní funkce aplikace. Tato funkce provádí následující úkony v chronologickém pořadí.

Získává od uživatele cestu k audio souboru.

Získává informace o audio souboru, jako počet vzorků, vzorkovací frekvenci, počet kanálů, celkový počet vzorků (počet vzorků vynásobený počtem kanálů)[[1]](#footnote-1). Poslední údaj, neboli celkový počet přečtených vzorků vypisuje do konzole a dále vytváří pole komplexních čísel o dané velikosti, dále volá funkci *decode()* a pole zaplňuje vzorky.

Vypočítává nejbližší menší číslo než je vzorkovací frekvence , a které je zároveň mocninou dvou. Na základě toho vytváří pole **vec** (jako vector) o dané velikosti. Toto pole slouží jako "okno" pro Rychlou Fourierovu transformaci. Také vytváří 2D pole **result**, jehož první dimenze je 2× menší než velikost okna a druhá dimenze slouží pro ukládání frekvence a amplitudy (hlasitosti; výstup Rychlé Fourierovy transformace). Nakonec zakládá 2D pole **notes** s první dimenzí o velikosti odpovídající počtu překrývajících se průchodů (Rychlou Fourierovou transformací) s frekvencí 60 průchodů za velikost okna (zhruba 60 průchodů za sekundu) a s druhou dimenzí o velikosti rozsahu frekvencí tónů MIDI (128 tónů, od 0 po 127)[[2]](#footnote-2). Daná velikost okna a frekvence průchodů je důležitá pro zachycení nízkých frekvencí (velikost okna) a rychlé změny (tónu, hlasitosti; frekvence průchodů).

Provádí samotnou frekvenční analýzu. Nahrává postupně vzorky do okna **vec**, provádí na něm Hannovu funkci[[3]](#footnote-3) (Hannovo okno). Pak provádí samotnou Rychlou Fourierovu transformaci. Vypočítává absolutní hodnotu výstupu z Rychlé Fourierovy transformace (amplitudu) a přiřazuje ji k frekvencím. Řadí hodnoty v každém okně podle amplitudy a následně je převádí do formátu kompatibilního s formátem MIDI (tyto data ukládá do **notes**). V tuto chvíli zvládá

Uživatel je zpravován o průběhu výpočtu Rychlé Fourierovy transformace a po dokončení je vyzván k zadání cesty k uložení výsledného MIDI souboru, který je následně vygenerován.

Výstupem je jednoduchý MIDI soubor odpovídající formátu MIDI 1.0.

### hann()

Tato funkce aplikuje na vstupní pole komplexních čísel (o velikosti **N**) Hannovu funkci.

### FFT()

*FFT()* je hlavní funkcí Rychlé Fourierovy transformace. Samotná transformace ale probíhá ve funkci *transform()*, kterou volá (a které předává parametry; ukazatel na pole komplexních čísel (okno) a jeho velikost (**N**)). Parametr d určuje vzorkovací krok, určuje směr (prohození dimenzí) Rychlé Fourierovy transformace.

### transform()

Tato funkce provádí samotnou Rychlou Fourierovu transformaci. Má parametry ukazatel na pole komplexních čísel a jeho velikost (**N**). Před výpočtem Rychlé Fourierovy transformace volá funkci *order()* a předává jí parametry ukazatel na pole komplexních čísel a jeho velikost (**N**).

### order()

Tato funkce provádí reverzní bitové třídění. Pomocí funkce *reverse()* obrací pořadí prvků v poli (0 → 0, 1 → 8 atd.). Jako parametry bere ukazatel na pole komplexních čísel a jeho velikost (**N**).

### reverse()

Tato funkce provádí bitové obrácení. Parametry funkce jsou **N**, celkový počet prvků pole a **n**, aktuální iterace (z funkce *order()*, pořadí v poli). Funkce vrací **p** (**n** po bitovém obrácení).

### decomlex()

Tato funkce převádí prvky pole komplexních čísel (výstup z **FFT()**) na 2D pole **results**, kde první dimenze je číslo aktuálního okna (počet průchodů = **i**; **results[i][]**) a druhá dimenze má vždy pouze 2 prvky buď amplitudu (0. je amplituda; **results[i][0]**), nebo frekvenci (1. je frekvence; **results[i][1]**).

### filter()

Funkce *filter()* fungovala původně jako filtr šumu (spodní hranice hlasitosti) a řazení hodnot 2D pole (frekvence a amplituda) na základě amplitudy (0. prvek 2. dimenze). Nyní funguje pouze jako třídící funkce. Ke třídění používá funkci *qsort(),* ta volá funkci *compare()* na porovnávání prvků. Funkce bere parametry 2D pole a jeho velikost (**N**).

### compare()

Tato funkce porovnává hodnoty (double) dvou prvků 2D pole. Pokud je první prvek větší, vrací 1, pokud druhý, vrací -1. Pokud se prvky rovnají, vrací 0.

### vminmax()

Tato funkce zjišťuje maximální a minimální amplitudu (hlasitost) v celém poli (2D poli; **res**, **result** v *main()*). Funkce bere parametry 2D pole a jeho velikost (**N**).

### compress()

Tato funkce provádí převod frekvencí a amplitud na decimální hodnoty kompatibilní s formátem MIDI, k tomu slouží funkce *freq()* a *volume()*. Také umožňuje nastavit počet hlasitostních kroků (**steps**, 127 je maximum), spodní hranici hlasitosti (spodní práh; **limit**, slouží k odstranění šumu) a maximální počet naráz znějících not (**maxSimultaneousNotes**, doporučuje se 1 pro jednoduchou melodii). Jako parametry bere 2D pole (**res[][2]**, **result** v *main()*),jeho velikost (**N**), 2D pole obsahující výstupní hodnoty funkce (**notes[][128]**, první dimenze je okno (průchod) a druhá dimenze jsou amplitudy), iteraci (**m**, **i** v *main()*) a **steps**, **limit** a **maxSimultaneousNotes**, které už byly vysvětleny.

### freq()

Tato funkce převádí frekvenci na hodnotu odpovídající notě v MIDI formátu (0 až 127 v decimální soustavě).

### volume()

Tato funkce převádí amplitudu na hodnotu odpovídající "rychlosti" (velocity v originále, spíše se jedná o hlasitost) v MIDI formátu (0 až 127 v decimální soustavě).

## OPEN2.cpp

Tento soubor obsahuje funkce pro otevírání audio souborů, zjišťování META dat (těchto audio souborů) a převádění zvukové informace z těchto audio souborů na čisté PCM. K těmto operacím je používána knihovna libsndfile[[4]](#footnote-4) (na ní jsou tedy závislé i podporované formáty).

### info()

Tato funkce zjišťuje informace o daném audio souboru. Její parametry jsou cesta k audio souboru (**fname**), počet vzorků (**f**), vzorkovací frekvenci (**sr**), počet kanálů (**c**), celkový počet vzorků (počet vzorků vynásobený počtem kanálů; **num\_items**).

### decode()

Tato funkce převádí zvukovou informaci z daného audio souboru na čisté PCM a vypisuje ji do pole komplexních čísel **a** (pouze do reálné části). Jako parametry bere cestu k audio souboru (**fname**) a pole komplexních čísel **a**.

## FREQ2MIDI.cpp

Tento soubor obsahuje funkce pro zápis do MIDI souborů.

### d2b()

Tato funkce převádí decimální čísla na znaky (char), která jsou potom zapsány do daného souboru (MIDI). Parametry této funkce jsou adresa objektu výstupního souboru (**myfile**), decimální číslo (**data**), délka požadovaného výstupu v bitech (**bit**, kdy 8 bitů je jeden znak (char)).

### start()

Tato funkce složí k vypisování hlavičky souboru MIDI. Jako parametry bere adresu objektu výstupního souboru (**myfile**), MIDI formát[[5]](#footnote-5) (**format**), počet hudebních stop (**tracks**) a definice základní časové délky "delta-time" (**division**).

### track()

Tato funkce vypisuje hlavičku hudební stopy. Jako parametry bere adresu objektu výstupního souboru (**myfile**) a celkovou délku souboru od této hlavičky (**length**).

### deltaTime()

Tato funkce vypisuje časové rozdíly mezi událostmi (v MIDI souboru) v jednotkách základní časové délky "delta-time." Parametry této funkce jsou adresa objektu výstupního souboru (**myfile**) a časový rozdíl (**time**).

### noteOn()

Tato funkce vypisuje noty (začátek noty) do souboru MIDI. Jako parametry bere adresu objektu výstupního souboru (**myfile**), notu (**note**, číslo v rozsahu od 0 do 127) a hlasitost (**velocity**)[[6]](#footnote-6).

### noteOff()

Tato funkce vypisuje konec noty (ukončení) do souboru MIDI. Jako parametry bere adresu objektu výstupního souboru (**myfile**), MIDI kanál (**channel**), notu (**note**, číslo v rozsahu od 0 do 127) a hlasitost (**velocity**). Tato funkce může být zastoupena zavoláním funkce *noteOn()* s hlasitostí 0 (*noteOn(***myfile**, **note**, **0***)*, kde myfile je fstream objekt (výstupní soubor) a note je libovolná nota)[[7]](#footnote-7).

### getFileSize()

Tato funkce zjišťuje velikost souboru (v tomto případě notes.out; ve funkci main() v souboru FFT.cpp). Jako parametry bere cestu k souboru (**filename**).

# Poznámky autora

Do původního projektu byl také zamýšlen spektrogram, z toho sešlo z toho důvodu, že na C++ neexistuje mnoho jednoduchých nebo snadno sestavitelných API pro GUI. Zkoušel jsem Qt a Tcl/Tk. První bylo až moc složité (nestíhal bych pořádnou implementaci) a druhé se mi nepodařilo zkompilovat (nebylo aktualizováno dlouhou dobu, nenašel jsem všechny nutné knihovny).

Dále byla zamýšlena také podpora mp3 souborů, tam byl zase problém, že tolik knihoven zatím není (vypršel patent, ale zatím je jich stále málo). Koukal jsem se na mpg123 a měl jsem i první prototyp algoritmu na převod mp3 do PCM, samotnou implementaci jsem ale už nestihl.

Při psaní kódu jsem také narazil na problém zvaný "stack overflow." Zapisoval jem až zbytečnš velké množství dat do polí. Bohužel jsem na to přišel moc pozdě (používal jse menší soubory) a nestihl jsem to upravit elegantně. Do budoucna bych spíše použil vector, nebo malloc (stačilo by i new a delete), abych používal dynamickou paměť a ne stack.

Při psaní této aplikace jsem narazil na zajímavý problém. Ukázalo se, že prakticky neexistuje knihovna pro C++ na vytváření souborů MIDI, takže jsem si ji vyrobil sám. Dalo by se říct, že to byla práce výzkumná. Taková práce zabrala mnoho času, ale řekl bych, že to stálo za to. Odkazy na stránky, z kterých jsem čerpal najdete pod čarou[[8]](#footnote-8).

1. funkce info() [↑](#footnote-ref-1)
2. Tabulka tónů MIDI: <http://subsynth.sourceforge.net/midinote2freq.html> [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/Hann_function> [↑](#footnote-ref-3)
4. <http://www.mega-nerd.com/libsndfile/> [↑](#footnote-ref-4)
5. <https://www.csie.ntu.edu.tw/~r92092/ref/midi/> (Pod MIDI File Formats) [↑](#footnote-ref-5)
6. První řádek je zakomentovaný, protože původně sloužil k určení kanálu a typu události. To se nyní provádí ve funkci main() v souboru FFT.cpp [↑](#footnote-ref-6)
7. Proto tato funkce není používána v FFT.cpp a je místo ní používána pouze funkce *noteOn()*. [↑](#footnote-ref-7)
8. <https://www.csie.ntu.edu.tw/~r92092/ref/midi/>

   <http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html>

   <https://github.com/colxi/midi-parser-js/wiki/MIDI-File-Format-Specifications>

   <http://www.deluge.co/?q=midi-tempo-bpm> [↑](#footnote-ref-8)