

# Linux DTM Guidebook

# **Linux デスクトップ DTM ガイド**

**atsushieno 著**

**2022-09-25 版 オーディオプラグイン研究所 発行**

# 序文

## Linux で DTM は難しい？

本書は、読者が Linux デスクトップ環境でいわゆる DTM( デスクトップミュージック、和製英語)を始められるようにしたいという思いから作られた本です。

筆者の日頃からの夢として、Linux でもなるべく Windows 環境や Mac 環境と比べて難しくないレベルで DTM ができるようになってほしいと思っています。それは Linux で発展してきた難易度がやや高い DTM 環境に習熟してから楽曲の打ち込みを始めたとか、Web アプリケーションとして動く範囲のものだけ作曲するといった、特殊な環境に身を置いて「Linux でも( がんばれば) DTM はできる!」というものから、もう少し一般向けに歩み寄ってみたい、ということです。

現実問題として、Linux は Windows や Mac ではありません。そのため、Windows 用ソフトや Mac 用ソフトは一般的には動きません。しかし、各 DAW で LV2 のサポートが安定し、Linux でも VST3 が使えるようになり、プラグイン開発フレームワーク JUCE が Linux サポートを拡充してきたりなど、さまざまな事情で以前より改善されてきた状況もあります。

「Windows でも Mac でもこの DAW は使えるし、同じものを Linux で使えばいいだけなんだ」とか「このプラグインはどの環境でも動かせる」あるいは「このプラグインは Windows 版／Mac 版しか無いんだけど、Linux でも使えるこれはそんなに難しくない」といったことを、主に Linux 環境で使える DTM 用ソフトを紹介していくことである程度示していかなければと思います。

## 本書の主なトピック

本書では DTM に関するツールを紹介します。音楽用ソフトウェアといつても、MP3 などのメディアプレイヤーなどを紹介することにはあまり意味がないので、本書の話題としては扱いません。

本書は( 筆者がほぼ常に読者として想定している) プログラマーを対象とするプログラミングに関するものではありません。しかし一方で、Linux プラットフォーム用に「インストールして実行するだけ」のバイナリプログラムが用意されていないものも多く、それらを紹介しないとなると、少なからぬソフトウェアを紹介できなくなってしまいます。Linux デスクトップで生きていくためには、どんなユーザーでもある程度「ビルドしてインストールして使う」ことに慣れておく必要がある／あつたはず、と考えて、ソースコードしか入手できないプログラムも紹介することにしました。一般的な音楽ソフトのビルド方法については、ページを割いて説明します。

本書の各章の内容は概ね次のようになっています。

- 第1 章では、Linux デスクトップ環境でしか直面しない、オーディオデバイスと MIDI デバイ

---

スの扱いの面倒なところについて、あまり技術的な詳細に踏み込まずに解説します（詳細で正確な情報は常に外部資料を読むほうがよいでしょう）。

- 第2章では、Linuxデスクトップ環境で使えるオーディオプラグインの種類について解説します。
- 第3章では、DTMに必要になるソフトの入手方法について解説します。上記の通り、パッケージが用意されていないこともあるので、ソースコードからビルドする手順についても可能な範囲で概説します。
- 第4章では、音楽制作に使用できるDAWを紹介します。各DAWの使い方の説明は長大になるのであまり踏み込みません。
- 第5章では、DAW上でロードして使うオーディオプラグインを紹介します。各プラグインの使い方もあまり踏み込みません。
- 第6章では、DAW・プラグインの分類に馴染まないその他のソフトウェアを紹介します。

## 本書に書かれていないこと

筆者は各種のプラグインがLinux環境で動作することを知っていますが、それぞれのプラグインの使い方に詳しいわけでもなければ、時としてそもそもそのプラグインが「何なのか」すら説明できないことがあります。これは特にシンセサイザーについて顕著です。ただ言い訳をさせていただくと、多分これは大抵のDTMerにとっては難しいことです。本書では最低限それらのプラグインをどう使えば良いのかを何とか示して、あとは読者に掘り下げてもらうスタイルを基本とします。

本書はLinuxデスクトップ上で動作するDAWやプラグインをそれなりに数多く紹介しますが、万遍なく紹介するものではありません。著者の知識の範囲でのみ書かれていますし、著者が紹介を絞り込んでいることもあります。商用製品は試すにも限度があるため、紹介はしていますが多くはありません。

# 目次

<b>序文</b>	<b>2</b>
Linux で DTM は難しい? . . . . .	2
本書の主なトピック . . . . .	2
本書に書かれていないこと . . . . .	3
<b>第 1 章 Linux とオーディオデバイス</b>	<b>6</b>
1.1 DAW とオーディオ API . . . . .	6
1.1.1 Jack と向き合う（あるいは回避する） . . . . .	6
1.2 MIDI: ALSA rawmidi と ALSA sequencer . . . . .	7
1.2.1 Jack MIDI: MIDI アクセスに影響するオーディオ API の選択 . . . . .	9
<b>第 2 章 Linux で使えるオーディオプラグインの種類</b>	<b>10</b>
2.1 概要 . . . . .	10
2.2 VST2 / VeSTige . . . . .	11
2.3 VST3 . . . . .	11
2.4 LV2 . . . . .	12
2.5 CLAP . . . . .	12
<b>第 3 章 音楽ソフトの入手方法</b>	<b>13</b>
3.1 パッケージからインストールする . . . . .	13
3.2 Windows 用のオーディオプラグインを使用する . . . . .	15
3.3 オープンソースのプラグインをソースコードからビルドする . . . . .	18
3.4 ソースからビルドする場合の手順 . . . . .	18
3.4.1 オーディオプラグイン以外でも使われるビルドシステム . . . . .	19
3.4.2 JUCE: Projucer . . . . .	19
3.4.3 waf . . . . .	20
<b>第 4 章 Linux で使える DAW</b>	<b>21</b>
4.1 Bitwig Studio . . . . .	22
4.2 Waveform Free/Pro . . . . .	23
4.3 Reaper . . . . .	24
4.4 Renoise . . . . .	25

---

4.5	Ardour / MixBus . . . . .	26
4.6	Zrythm . . . . .	27
4.7	QTractor . . . . .	28
4.8	helio-workstation . . . . .	29
<b>第 5 章</b>	<b>Linux で使えるオーディオプラグイン</b>	<b>30</b>
5.1	サンプラー . . . . .	30
5.1.1	サウンドフォント (SF2) . . . . .	31
5.1.2	SFZ . . . . .	31
5.1.3	その他の製品 . . . . .	32
5.2	シンセサイザー . . . . .	33
5.2.1	ウェーブテーブル／ハイブリッドシンセサイザー . . . . .	34
5.2.2	バーチャルアナログシンセサイザーなど . . . . .	37
5.2.3	チップチューン音源・エミュレーター . . . . .	40
5.3	エフェクトプラグイン . . . . .	43
5.3.1	リバーブ . . . . .	43
5.3.2	イコライザー . . . . .	44
5.3.3	コンプレッサー . . . . .	45
5.3.4	ギターアンプ、オーバードライブ、ディストーション . . . . .	46
5.3.5	フェイザー . . . . .	47
5.3.6	エフェクターコレクション . . . . .	47
<b>第 6 章</b>	<b>補遺: Linux で使える歌唱合成ソフト</b>	<b>49</b>
6.0.1	Synthesizer V . . . . .	49
6.0.2	NEUTRINO . . . . .	50

# 第 1 章

## Linux とオーディオデバイス

### 1.1 DAW とオーディオ API

DAW とは digital audio workstation の略であり、DAW を使うにあたってオーディオデバイスへのアクセスはほぼ必須でしょう。

オーディオ制作環境の高度なセットアップについては、Windows には Windows なりの、Mac には Mac なりの煩雑さがありますが、カジュアルに DAW を起動して音楽制作したいだけであれば、そのようなセットアップは通常は不要です。しかし Linux の場合は、使用する DAW の選択にもよりますが、デフォルトで面倒事がついて回ると考えたほうがいいかもしれません。

詳しくは後述しますが、デフォルトで面倒事を回避したい筆者のいつものやり方は「オーディオ API を選択できる DAW では PulseAudio を選ぶ」「オーディオ API が PulseAudio でも MIDI 入力デバイスが使える DAW を選ぶ」です。これはリアルタイム性を犠牲にする（状況によっては演奏中に音が飛ぶかもしれない）選択肢ですが、それさえ許容できれば問題ありません。この節の残りの長い説明も読み飛ばして大丈夫かもしれません。

#### 1.1.1 Jack と向き合う（あるいは回避する）

オーディオをリアルタイムで（音が途切れたりしないように）処理するために、Windows では ASIO というドライバーのサポート技術がありますが、これは他のオーディオデバイスの利用を制限することになります。Linux では Jack というオーディオアクセス機構が ASIO に近いものです。Linux の汎用的なオーディオアクセス API は ALSA (Advanced Linux Sound Architecture) と呼ばれ、Jack は ALSA デバイスを占有してリアルタイム性を保証します。Jack はそれ自体はオーディオデバイスへのアクセスを自ら実装する存在ではなく、オーディオデバイスや MIDI デバイスへの読み書きをリアルタイムで処理するための API です。

この Jack が環境によっては割と曲者で、他のアプリケーションがオーディオデバイスを利用しているときはそれらを止めてしまったり、逆に Jack の初期化に失敗してそれ以上 DAW が起動してくれない、といった問題が起こったりします。たとえば Web ブラウザが起動していてオーディオデバイスが使われていたりすると、Jack が使えません。Jack を正常に起動できるようにするには、`/etc/security/limits.conf` など Linux のシステム設定を正しく調整する必要があります。Jack には Jack1 と Jack2 があり、互換性がありません。通常は Jack2 を使うことになると考えられますが、

Jack1 のパッケージが依存関係で引き込まれるようになることもあります（注意して回避すれば大丈夫です）。

それらのことが理解できいても、筆者の場合は、jack を制御するツール qjackctl などで手動で起動しようとしても期待通りに起動してくれないような Ubuntu デスクトップ環境を何度も経験してきました。

そもそも音楽の打ち込みの際には他のアプリケーションでオーディオファイルを再生したりする場面が頻繁にあるはずで、オーディオデバイスを複数セットアップしているのでなければ、Jack を使わないほうが作業環境としては妥当でしょう。オーディオアクセスを排他的に行いたいのであれば、Jack を正しく理解して使う必要があるでしょう。

ちなみに、ALSA を直接使用する場合でも、DAW がデバイスに排他的にアクセスするような使い方をしていれば（Jack はあくまでそのような使い方を肩代わりしてくれる API であって、誰でも独自に Jack 相当の機能を実装できるのです）、仮にデバイスを占有できたとしても、作業中に遭遇する問題は Jack と大きく変わりません。

ALSA と Jack 以外の選択肢としては、ネットワークオーディオなどもサポートする PulseAudio が有り得ます。古い Linux 用 DAW には PulseAudio のようにリアルタイム性を放棄した「プロフェッショナルではない」作業環境に対するリスクが無く PulseAudio をサポートしなかったものもありますが、現在では Ardour も含め PulseAudio がサポートされているのが一般的といってよいでしょう。それでも PulseAudio デフォルトのオーディオ設定にならないことが多いので、その場合は手動で設定を変更する必要があります。また、Ardour では Jack 以外の選択肢では MIDI デバイスが利用できなかったりしますし、QTractor のように現在でも Jack が必須になっているものもあります。

オーディオアクセス API のトレンドは、今後 PulseAudio から PipeWire に切り替わっていくことも考えられますが、2022 年の本書執筆時点では直接 PipeWire をサポートする DAW は見当たません。PipeWire には Jack API のサポートや PulseAudio サーバーの代替があるので、DAW がオーディオデバイスへの排他的なアクセスを前提としているなら従来の API でアクセスすれば十分であると考えられます。

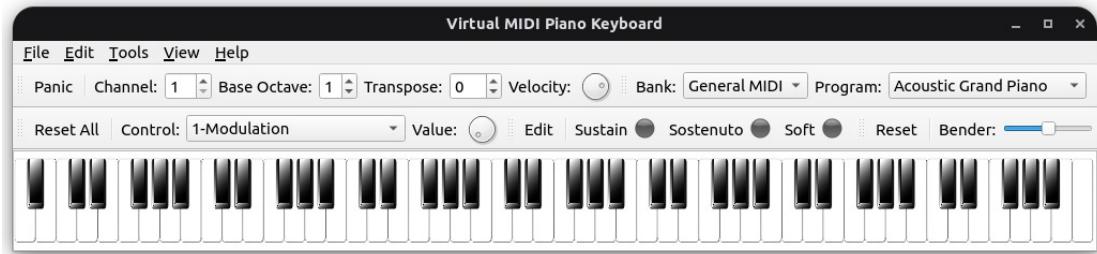
## 1.2 MIDI: ALSA rawmidi と ALSA sequencer

MIDI デバイスの利用もさまざまな困難がつきまといます。

いったん DAW から考えを切り離して、MIDI デバイス単体での利用可否を論じますが、Linux では USB-MIDI デバイスの接続はほぼ問題なく行えるでしょう。筆者の環境にはあまり USB-MIDI デバイスが存在しませんが、USB-MIDI デバイスの接続で問題になった経験はありません。USB とは、USB ドライバーさえあれば個別のデバイスドライバーが無くてもそのデバイスを Universal に利用できるようにするための規格であるはずです。

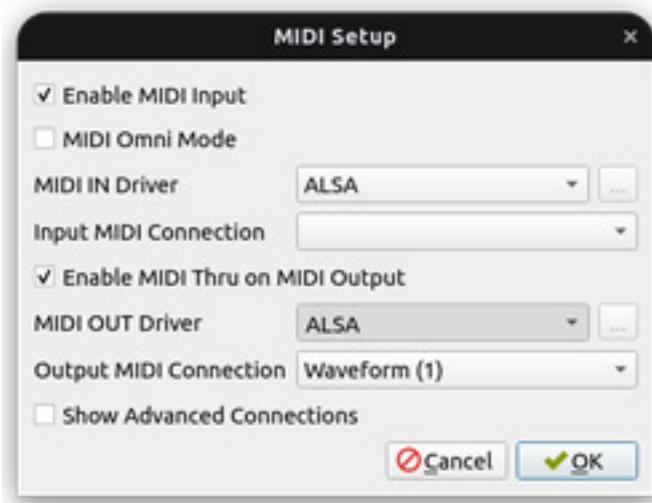
一般的に Linux の MIDI 接続は ALSA であり、API としては ALSA sequencer がサポートされていて USB-MIDI だけでなく仮想 MIDI デバイスなども利用できるというのが一般的です。仮想 MIDI デバイスというのはつまりソフトウェアで仮想的に作り出す MIDI デバイスのことです。MacOS や Linux ではこれが自由にできます。Windows ではできません。Windows 用ソフト loopMIDI のような仮想 MIDI デバイスドライバーをインストールする必要もありません。Linux デ

スクロールには VMPK という仮想ピアノアプリがあって、PC 上のキーボードから MIDI 演奏が可能です。



▲図 1.1 vmpk

デフォルトでは Fluidsynth を利用してアプリケーション単体で音を出しますが、ALSA 仮想 MIDI デバイスとして使えるように設定することも可能です。( ALSA デバイス名が”out”というわかりにくい名前になるので、利用時には気に留めておきましょう。)



▲図 1.2 vmpk を ALSA 仮想 MIDI デバイスとして設定する

ALSA の仮想 MIDI デバイスを作るのは( プログラマー視点では) 比較的簡単で、筆者も Kotlin 言語で自作した MIDI キーボードアプリで実現しています。

ところで、一般的に DAW では ALSA sequencer がサポートされていると書きましたが、この例外が alsoundrawmidi しかサポートしていない Oracle Java の javax.sound.midi の API を使っているソフトウェアで、具体的には Bitwig Studio です。alsoundrawmidi では仮想 MIDI がサポートされていないので、VMPK のようなソフトウェアが MIDI デバイスとして認識されないことになります。

BLE MIDI デバイスのサポートの状況は厳し目です。Linux で BLE をサポートするのは Bluez というソフトウェアスタックで、BLE MIDI のサポートも実装されているのですが、Ubuntu などで

は(なぜか)デフォルトで無効になっています。有効になっているかどうかは、`ldd /usr/sbin/blueoothd`などを実行して、`libasound.so.*`が含まれているかどうかで判断できます。無効になっている BLE MIDI を有効にするには、bluez をソースから「BLE MIDI を有効にして」ビルドしてインストールする必要があります(やや高度な作業になるので諦めたほうが良いかもしれません)。

なお、Linux アプリケーションが動作する ChromeOS では、USB-MIDI デバイスが認識されないレベルで使えません。その割に Web MIDI API では認識されたりするのですが、これは ChromeOS が USB デバイスへのアクセスをストレージなど限定された種類に限定しているためです。2021 年時点でのこの問題を改善する意思がないので( chromium issue #1050136)、今後も少なくとも数年間は期待できなそうです。

### 1.2.1 Jack MIDI: MIDI アクセスに影響するオーディオ API の選択

さて、MIDI デバイスが単独で利用可能かどうかは、オーディオ API の選択とは無関係であるべきですが、残念ながら現実の Linux 用 DAW では密接に関連していることが多いです。

典型的な問題パターンは「MIDI デバイス接続サポートの実装が Jack 経由でしか実装されていない」、すなわち MIDI サポートが Jack の MIDI アクセス API(プログラマー向けに説明するなら `jack/midiport.h`)を使用して実装されている場合です。伝統的に Jack 経由でのオーディオアクセスしか実装してこなかった DAW では一般的な実装だったとも考えられます。その場合、DAW でオーディオ API で Jack を選択できるようになった後も、「MIDI デバイスを使いたいなら Jack を使え」という状態になっているわけです。

オーディオ API の選択が ALSA であれば MIDI デバイスアクセスの部分には ALSA sequencer API を使って実装すべきところです。DAW によってはオーディオ API と MIDI API を別々に選択できるようにもなっていますので、「何を選択したら何が利用可能になるのか」が整理される必要がありますし、これは開発者だけでなくユーザーのレベルでも混乱の原因となります(これが Linux の DAW を不必要に難しくしているといえます)。

類似の問題として、「Jack と ALSA はサポートしているが PulseAudio では MIDI デバイスが選択できない」というケースもあります( Ardour6)。

筆者の私見としては、「DTM はノート PC1 台だけでできるべき」であり、そのためには「複数のオーディオデバイスが無くてもできるべき」であり、かつ「DAW で打ち込み作業をしながら Web ブラウザやオーディオファイルプレイヤーで音を出せるべき」なので、「Jack で排他的にデバイスを利用せずに DAW を操作できるべき」と考えています。おそらくそのように考える DTMer は少なくないのではないでしょうか。この考えだと通常利用できない DAW がいくつか出てきますが( Ardour, QTractor など)、これは仕方ないと考えています。

最後に、Jack や Jack MIDI に関する筆者のお気に入りの参考情報を(英語ですが)いくつか Web 検索して探せる範囲で列挙します。

- "ALSA and JACK MIDI explained (by a dummy for dummies)" - Jack MIDI とは何なのかを説明してくれています
- "How do I configure my linux system to allow JACK to use realtime scheduling?" - Jack がリアルタイムスケジューリングに失敗する原因と対処方法がいくつかまとめられています

## 第 2 章

# Linux で使えるオーディオプラグイン の種類

### 2.1 概要

オーディオプラグインには「プラグインフォーマット」と呼ばれる概念があります。プログラマー向けに説明するなら、これは API の違いです。あるプラグインフォーマットのプラグインをロードできるのは、そのプラグインフォーマットをサポートする DAW のみです。iPhone アプリが iPhone でしか動かないように、あるいは Android アプリが Android でしか動かないように、VST2 プラグインは VST2 をサポートする DAW でしか使えず、VST3 プラグインは VST3 をサポートする DAW でしか使えません。VST2 と VST3 は互換性がありません。

実際には、DAW とプラグインでフォーマットが同一ならというわけではなく、さらに動作プラットフォームとして Linux がサポートされていないものを Linux で動作させることはできません（Wine を使うブリッジの仕組みなどが必要になります）。

「プラットフォームの数 × プラグインフォーマットの数」だけ組み合わせがあって同一でなければならないとなると、絶望的な確率であるように聞こえますが、実際には状況はもう少し楽観的です。DAW の多くは複数のプラグインフォーマットをサポートしています。Linux の場合、2022 年の本書執筆時点では商用製品の DAW はほぼ VST2 と VST3 をサポートしています（筆者の記憶では Renoise は未対応です）。オープンソースの DAW ではほぼ全てが VST2 と LV2 をサポートしていて、Reaper も LV2 をサポートしています。JUCE の 2022 年時点での最新メジャーバージョンである JUCE7 が LV2 をサポートしたので、JUCE で作られている Tracktion Waveform の次世代でも多分 LV2 をサポートするでしょう。

そして重要なのは、大抵は複数のフォーマットのプラグインを自由に組み合わせることが可能であるということです。Linux ネイティブの Dexed で MIDI から演奏して、その出力を第 3 章で紹介する方法で Xfer Records OTT に送ってエフェクトをかける、といった処理が可能です。

オーディオプラグイン開発においては、どのプラットフォームでも実行できるようなソースプログラムを書いて個別のプラットフォーム向け、個別のプラグイン向けのコードは最小限に抑えるやり方がトレンドとなっています。もともと音声データの処理はクロスプラットフォーム開発に向いていて、GUI もオーディオプラグインでできることは限られているので、クロスプラットフォームの技術を採用するデメリットが小さいです。ひとつの製品で複数のプラグインフォーマットをサポートす

る JUCE や iPlug2 といったプラグイン開発フレームワークが流行する理由のひとつでもあります。

最後になりますが、この章では「汎用プラグインフォーマット」についてのみ言及しています。各 DAW には固有のビルトインプラグインが用意されており、また Reaper の ReaPlug のようにユーザーが独自に開発できるものもあります。これらは他の DAW に持ち出すことができませんが、汎用フォーマットのプラグイン同様、自由に組み合わせて利用できるのが一般的です。

## 2.2 VST2 / VeSTige

VST2 は Steinberg が開発した 20 世紀からあるプラグインフォーマットで、すでにサポートが終了しており、VST2 SDK はオープンソースで公開されていなかったので、開発用の SDK を入手することすらできません。一方で商用製品の世界では、VST2 対応製品しか持っていないミュージシャンのために、未だに製品がリリースされ続けているのが現状です。

Linux での利用については、特に VST3 とは異なり Steinberg のプロプラエタリなライセンスでしか存在しなかったので、VST2 SDK を利用したプラグインは特にオープンソースの世界ではありませんでした。一方で、VST2 の API さえ互換性があれば SDK が同一の実装である必要はないので、Linux の MIDI シーケンサー LMMS プロジェクトでは VeSTige と呼ばれる VST2 互換 API の実装が開発されました。この VeSTige にはいくつかの派生バージョンがあつて現在ではどこにあるものを使うのが良いかはわかりませんが、DISTRHO/DPF などで VST2 ビルドを実現するために使われています。VeSTige の利用は Steinberg のライセンス違反であるという俗説も散見されますが、Steinberg からも Steinberg が VeSTige の利用を禁止することはできないと Steinberg のフォーラム上で明言されています（API には著作権が無く実装が別物なので当然です）。

DTMer としては、同じ製品の VST2 版と VST3 版がある場合は、今後は VST3 版を使うようにしていったほうがよいでしょう。VST2 版はいつサポートされなくなるかわかりません。

## 2.3 VST3

VST3 は VST2 の後継フォーマットですが、VST2 とは完全に互換性が無いどころか設計思想も根本的に異なるフォーマットになっています。やや複雑な仕様で、プラグインでも DAW でもなかなかサポートされなかつたこと也有って、最終的には Steinberg が VST2 SDK の公開を終了してからようやく普及してきた、というのが一般的な VST 開発者の印象でしょう。

VST3 は Steinberg が GPLv3 ライセンスで公開していることもあって、少なくともフリーソフトウェアの世界では躊躇なく使われています。商用製品をリリースしたい開発者は別途 Steinberg から商用ライセンスを得ることになります。

Linux 上の VST3 サポートは、VST3 SDK の公開当初から十分にあったわけではありませんでした。VST3 SDK が最初に公開されたのは 2008 年ですが、筆者の感覚でいえば、VST3 SDK が Linux 上でも使用に耐えるプラグインをビルドできるところまで成長してきたのは、2010 年代後半の話です。その後 JUCE でも VST3 がサポートされるようになり、Linux 上でも VST3 プラグインが使われるようになってきました。オープンソースのビルドに関連して JUCE と Projucer について言及したことがありました。古い JUCE プラグインでは VST3 に対応していないことがあります。DAW のほうも、2020 年以降になって Linux VST3 サポートが実装されたものが多いです（Ardour,

QTractor, Tracktion Waveformなど)。

### 2.4 LV2

2022年の本書執筆時点ではオーディオプラグインフレームワークはクロスプラットフォームで利用できるのが一般的ですが、従来は「WindowsではVST、MacではAudioUnit」という棲み分けが一般的でした。LV2は「Linux用のプラグインフォーマット」として20世紀末頃に作られたLADSPA (Linux Audio Developer's Simple Plugin API) のバージョン2として策定され発展してきたものです。LV2は実際にはクロスプラットフォームで利用できるように設計されているのですが、現実的にWindowsやMacでLV2をサポートするDAWは一部のオープンソースのものに限られています。

LV2は実際には個人開発者が1人で全て策定しており、かなり独自の技術選択の上に成り立っており、特にセマンティックWebで使われるRDFに基づく仕様書やそのメタデータ文法であるTurtle Syntaxを理解しないとプラグインが開発できないなど、カジュアルな開発者を遠ざける要素が多くあります。それを乗り越えてLV2プラグインを開発する猛者も何人かいますが、限られた世界の話であり、LV2はあまり普及してきませんでした。しかしDISTRHO/DPFやLV2サポートを実現したJUCEのfork(独自版)が使えるようになってきて、それらを使う手法がだんだん共有されてきて、2022年の本書執筆時点ではLinuxデスクトップで最も普及しているプラグインフォーマットと捉えても良いといえます(VST3が拮抗してきたかもしれません)。

LV2をDTMで利用する際の注意点は2つあります。ひとつは商用製品DAWでのサポートがまだ充実していないことです。Reaperと(これを商用製品として評価するなら)ArdourをベースにしたHarrison MixBusくらいです。もうひとつは前述の通りLinux以外でのサポートが絶望的に小さいことです。LV2しかサポートしていないプラグインを使うと、Linuxから他のプラットフォームに楽曲データを移行できないということになります(VST3やVST2の打ち込みで保存された状態データを共有できるとは限りません)。

(なおLV2の出発点であるLADSPAについては、今でもサポートしているDAWもありますが、基本的に過去の存在として本書では扱わないことにしました。LADSPAだけをプラグインとしてサポートするLMMSなども言及しません。DSSIも同様とします。)

### 2.5 CLAP

CLAPは2022年6月にバージョン1.0としてオーディオプラグインフォーマットの世界で新たな挑戦者として出現したもので、まだ採用している製品は多くはないものの、商用製品でもオープンソースでもVST3やLV2が受容されるようになるまでの時間に比べると格段に早いスピードで受容されつつあります。

現状でCLAP「のみ」をサポートする製品はおそらくひとつもないでしょうし、CLAPを積極的に売り出そうとしているBitwig以外の環境でDTMerがCLAPプラグインに依存して打ち込むのは時期尚早と言わざるを得ませんが、QTractorでもサポートされるようになっており、今後は利用する場面が出てくるかもしれません。Bitwigと同じくCLAPをプッシュしているu-heのプラグイン製品(Hiveなど)では、CLAP版がVST3版と同様に手厚くサポートしていくでしょう。

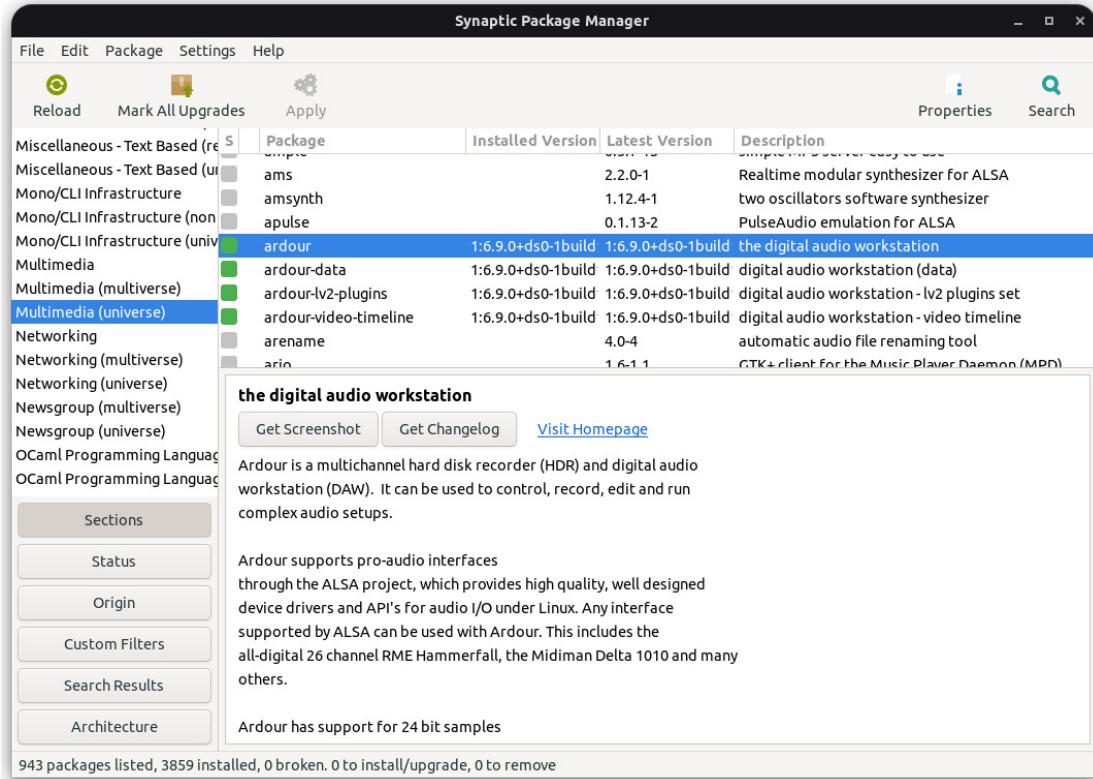
## 第3章

# 音楽ソフトの入手方法

### 3.1 パッケージからインストールする

Linuxディストリビューションはパッケージソフトウェアの集合体です。DAWやオーディオオブジェクトのようなソフトウェアもその一部となり得ます。ポピュラーなDAWは、公式パッケージとしてインストールすることも可能かもしれません（Linuxディストリビューションによります）。もし公式パッケージでインストールできて、かつそのバージョンに（古すぎる等の）不満が無ければ、それが一番楽なセットアップ方法でしょう。

synapticやpacmanのようなパッケージマネージャーがインストールされていれば、その中に"Multimedia"のようなカテゴリーが含まれていて、その中にオーディオ関連のアプリケーションが集められている可能性が高く、その中から面白そうなアプリケーションを探すことも可能です（パッケージマネージャーはLinuxディストリビューションによって異なります）。



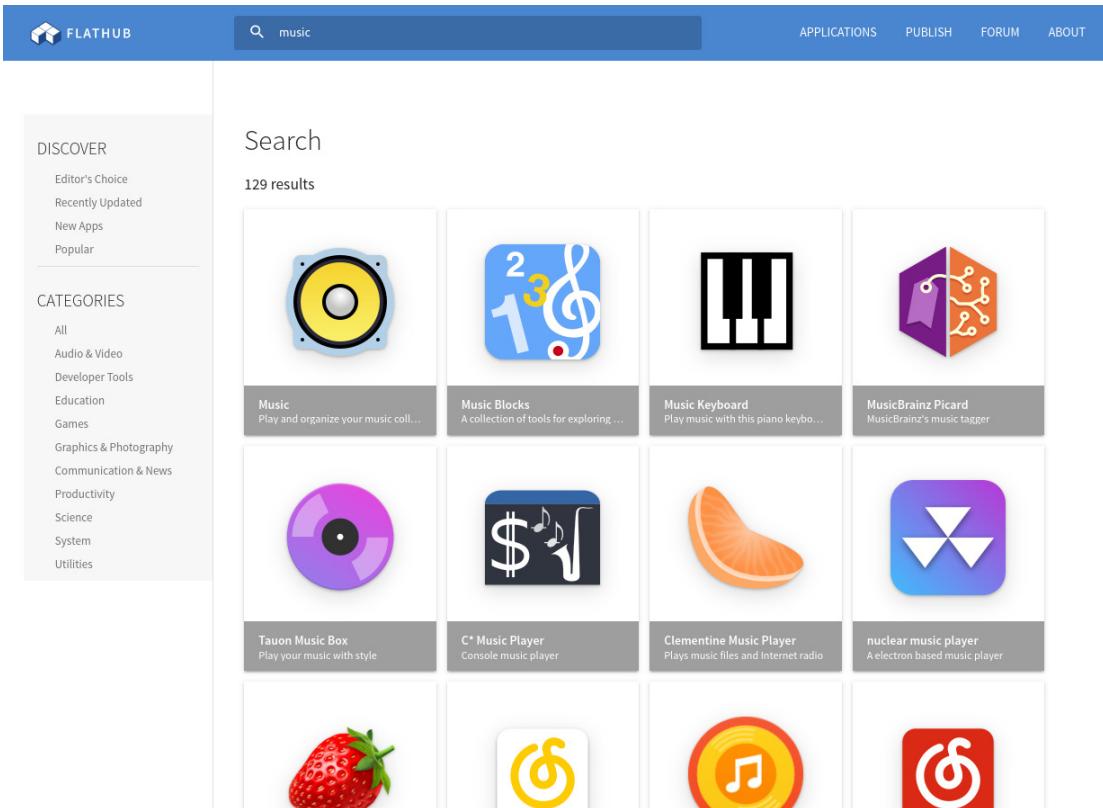
▲図3.1 synaptic の Multimedia カテゴリー

ディストリビューション公式パッケージは、アプリケーションの開発元が公開しているパッケージと比べると古いのが一般的です。DAW やプラグインの開発元がバイナリパッケージをリリースしていれば、それらを使うのが最も満足度が高いでしょう。ただし自分から積極的にパッケージをダウンロードしてインストールしたり、パッケージの提供 URL をパッケージマネージャーに追加しないといけなかつたりはします。

開発者が公式にパッケージを提供していないディストリビューションであっても（あるいは最初から Linux 版をサポートしていない場合）、アプリケーションのパッケージングに長けた外部の開発者が独自にパッケージを作り公開している場合があります。場合によっては開発者が公式に README などで言及していることもあります。それらが提供されていれば、公式パッケージをインストールするのと同程度には簡単にインストールできます。ただ、開発者が自ら提供しているパッケージと比べると信用度が下がる（悪意ある改変が加えられたパッケージが配布されている可能性がある）ことには注意しましょう。

パッケージのインストール方法としては、ディストリビューション固有のパッケージマネージャーの他に、Flatpak や snap を使うことも考えられます。注意点として、flatpak のようなアプリケーションのコンテナー環境では、Jack の API を使ったオーディオアクセスには PipeWire が必要です。

### 3.2 Windows 用のオーディオプラグインを使用する



▲図 3.2 flatpak で"music"を検索した結果(抜粋)

### StudioRack プロジェクト

筆者が理想としている音楽制作環境は、どのプラットフォームで制作した音楽データをどこにでも以降できることであり、そのために Linux 環境でも利用できるソフトウェアの充実が課題だと思っていますが、類似の目標意識をもってオープンソースで利用可能なオーディオプラグインを収集し、時にはビルドスクリプトを開発者に提供しつつ、Web サイトにまとめてパッケージを提供している StudioRack というプロジェクトがあります。オーディオプラグインベンダー Waves が出している Studio Rack という製品と名前がかぶっているので検索性が悪いのですが、StudioRack github などで検索するとこのプロジェクトの情報が出てきます。

インストール方法はやや独特ですが、よく知られたプラグインのインストールを楽にするためのプロジェクトなので、セットアップに苦労したら活用してみるとよいでしょう。

### 3.2 Windows 用のオーディオプラグインを使用する

現実的な問題として、Linux 用オーディオプラグインのラインアップは、Windows 用のそれには遠く及びません。一方で、Windows 用ソフトウェアは、Wine を使うと動かせるものが少なくありません。STEAM では、Windows 版のゲームを Wine でそのまま動かせるとしていくつかのパッケー

ジが販売されています。オーディオプラグインの場合は特に VST2/VST3 の API が実装されていればよく、プラグイン本体に Wine でも実装されていないような機能を実装する場面は少なく、特に基本的に.NET などを使わず C++ で作られているはずで、Wineとの相性はだいぶ良いはずです。

オーディオプラグインの実装テクニックのひとつとして、自身をオーディオプラグインとしてそのプラグインフォーマットの API を実装しつつ、その中では他のプラグインをホストして呼び出す、いわゆるプラグインラッパーあるいはブリッジと呼ばれる手法があります。オーディオプラグインとして、Wine の機能を使って Windows のプラグインをブリッジするソフトウェアが、Linux デスクトップでは古くから開発されてきました。

これらは Wine が成熟する以前からさまざまな開発者によって作られてきたもので、2022 年の本書執筆時点では [robbert-vdh/yabridge](#) がお薦めです。少し古い情報であれば、Airwave、LinVST といったツールが紹介されていると思います。Melodyne や Serum、OTT が動作している様子が見られます。筆者は Linux ネイティブプラグインを使うので、yabridge を使ったことはほとんどありませんが、OTT、Varhalla Supermassive、Plogue などは動作確認しています。



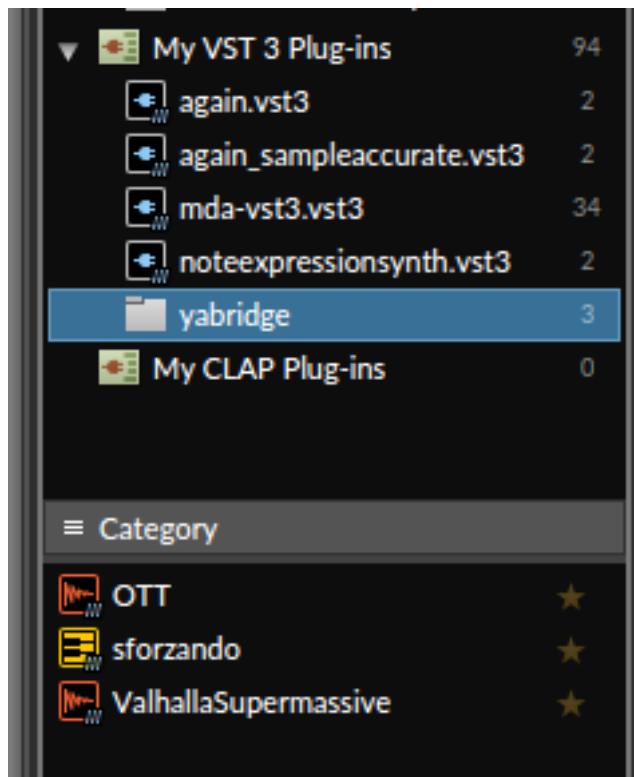
▲図 3.3 yabridge のスクリーンショット ( README より )

yabridge の基本的な使い方を少しだけ説明します。うまく行かない場合は GitHub プロジェクトの README.md を参照してください。

- Wine を使用して、Windows 用のオーディオプラグインをインストールします。インストラーが無い場合は、一般的には`~/.wine/drive_c/Program Files/Common Files/VST2` ( VST2 の場合) や `~/.wine/drive_c/Program Files/Common Files/VST3` ( VST3 の場合) にファイルをコピーします ( VST3 の場合は`*.vst3` の形式のディレクトリ名になるはずです)。

- yabridge のバイナリパッケージは公式 GitHub リポジトリの Release ページ、あるいは Open(SUSE) Build Service (OBS) 上にあります。\*.tar.gz アーカイブの場合は、ダウンロードして内容を ~/.local/share/ ディレクトリ以下に展開します。~./local/share/yabridge というディレクトリができるはずです。このディレクトリの中に yabridgectl というツールがあるので、これを次に使います。OBS パッケージの場合は、このツールが PATH の通った場所にインストールされていることでしょう。
- yabridgectl を使って、Wine 上の VST2/VST3 のパスを追加します。例: ~./local/share/yabridge/yabridgectl add ~/wine/drive\_c/Program Files/Common\ Files/VST3( VST3 の場合)
- yabridge の sync コマンドを実行すると、Linux 環境上の~/.vst や~/.vst3 に yabridge のプラグインが追加されます: ./local/share/yabridge/yabridgectl sync

これで DAW を開いて、プラグインリストを最新の状態に更新してみると、yabridge が VST プラグインとしてリストアップされるでしょう。全ての Wine 上のプラグインが 1 つの yabridge というプラグインコンテナーに含まれた状態として認識されるはずです。



▲図 3.4 Bitwig Studio で yabridge が検出された状態

### 3.3 オープンソースのプラグインをソースコードからビルドする

序文にも書きましたが、本書で紹介するソフトウェアには、バイナリパッケージが用意されていないものがあります。それらを使うためには、読者各位のPC上でソースコードからビルドしてインストールする必要があります。これはプログラマーでない読者にとっては難しく見える作業でしょうが、一般的なプログラマーにとっても（慣れた作業に聞こえるとしても）本書で紹介するソフトは特殊で、一般的な経験は通用しません。「./configure; make; sudo make installでいける」ものは多くありません。逆に、「インストール」する作業も、特にオーディオプラグインの場合は単にホームディレクトリ以下の所定のディレクトリにコピーするだけのことが多く、比較的安全に「システムを壊す」心配なく済ませることができます。

日本ではソフトウェアのビルドはプログラマーの仕事とみなされることが多いと思いますが、筆者の周辺にはたまに「プログラミングはできないがオープンソースのツールをインストールして使うくらいはできる」という職業ソフトウェアユーザーが（日本人でも）少なからずいます。そもそもプログラミングに類する作業もプログラマー以外の従業員が着手していたりします。義務教育でプログラミングが普及すれば、このような傾向はさらに拡大するでしょう。

ソースコードからのビルドとインストールは、「コマンドラインで指定されたコマンド／実行すべきコマンドを必要な時に実行するだけ」であって、プログラミングの知識が無くても何とかなります。出自不明で自分でもよくわからないコマンドの実行（特に得体の知れないサイトに載っているようなもの）には警戒すべきですが、読者にとって筆者が何者であるかが明らかで、いつでも公開の場で糾弾できる本書に、危険なコマンドを掲載するのは筆者にとってはデメリットだらけで、読者が恐れる理由はほぼ無いと思います。とはいえ、筆者が無知／無頓着で危険なスクリプトを提示する可能性は常に警戒したほうが良いでしょう。心配な場合は周囲の詳しい知人などに質問しつつ、不必要に恐れることなく実行することをお勧めします。

### 3.4 ソースからビルドする場合の手順

オーディオプラグインのソースには特殊なものもありますが、一般的なビルドシステムを使っているものもあります。プロジェクトによってはひとつのビルドスクリプトの中で複数のビルドシステムを部分的に利用していることもあります。一方で複数のビルドシステムをサポートしているだけ（どれを使っても良い）ということもあります。正確なことは、それぞれのプロジェクトの READMEなどのドキュメントを見てください。

もっとも、オーディオプラグインのソースによっては、READMEが不親切な場合も少なからずあります。典型的なパターンとしては「これは JUCE のプロジェクトなんだから JUCE アプリのビルド方法なんて説明しない」という感じのものです。「Windows の Visual Studio 用のソリューションファイル\*.sln がリポジトリに入っているれば、それを Visual Studio で開くだけ」「Mac の Xcode 用の\*.xcodeproj が入っているれば Xcode で開くだけ」などが自明であるのと同様です。とはいえ、これは知識がないと分からぬ話なので、この節では JUCE を含む、典型的なビルドスクリプトファイルとビルド方法について、説明しておこうと思います。

### 3.4.1 オーディオプラグイン以外でも使われるビルドシステム

次のリストは「一般的なほう」のビルドシステムです。

- **CMake:** ソースツリー（ディレクトリツリー）に `CMakeLists.txt` というファイルがあれば、そのプロジェクトでは CMake というビルドシステムが使われています。この場合、典型的なビルド手順は、（`cmake` をシステムにインストールした上で）`cmake -Bbuild -GNinja` からの `cmake --build build` となります。CMake はソースツリーに影響を及ぼさずにビルドディレクトリ（この手順例の場合は `build`）以下にビルド中間ファイルとビルド結果を全て生成します。`sudo cmake --install build` などもサポートされているかもしれません、VST2 や VST3 の場合は、ビルド結果を `~/.vst` や `~/.vst3` にコピーするだけのほうが安全です。ビルドディレクトリの中から `find build -name *.vst3`などを実行して、コピーすべきファイル／ディレクトリを探してください。CMake は数多くのプラグインの開発に使われている JUCE の比較的新しいバージョンである JUCE6 以降でもサポートされており、`CMakeLists.txt` があれば JUCE のプロジェクトのものである可能性がそこそこ高いです。
- **Make:** ソースツリーに `Makefile` というファイルがあれば、そのプロジェクトは `make` でビルドするものである可能性が高いです。Linux で特に LV2 をサポートするプラグインの場合は、`DISTRHO/DPF` というオーディオプラグインフレームワークが使われていることが少なからずあります。ビルド結果がどこに作られるかはプロジェクト次第ですが（CMake のように洗練されたビルド出力ディレクトリは存在しないでしょう）、`find . -name *.vst3`などを探せるでしょう。
- **meson:** ソースツリーに `meson.build` というファイルがあれば、そのプロジェクトでは Meson という比較的新しいビルドシステムが使われています。この場合の典型的なビルド手順は、`ninja` と `meson` をシステムにインストールするか、`meson` のソースツリーをチェックアウトして、`meson setup build` からの `ninja -C build` を実行することとなるでしょう（`meson` ソースツリーから実行する場合はトップディレクトリの `meson.py` を `meson` の代わりに実行します）。
- **autotools:** ソースツリーに `configure` や `autogen.sh` といったファイルが含まれていれば、それは GNU Autotools をビルドシステムとして使用している可能性が高いです。典型的なビルドは `./configure; make; sudo make install` のようになるでしょうが、Autotools の使い方は Linux の一般的な話になるので、これ以上は割愛します。

### 3.4.2 JUCE: Projucer

JUCE はかつて Projucer という独自のビルドシステムのみをサポートしていました。もしソースツリーに `*.jucer` というファイルがあれば、それは Projucer 用のプロジェクトファイルです。Projucer は CMake や meson と同様、それ自体はビルドツールではなく、プラットフォームに合わせて Visual Studio のプロジェクトや Xcode のプロジェクト、Linux 用には Makefile を生成したりするものです。JUCE の Projucer は GUI ツールであり、これでロードすることもできます。

GUIでビルドファイルを生成するというのは、必ずしも作業体験が良いわけではなく、特にビルドの自動化とは相容れません。Projucer は`--resave`という引数を指定すると、コマンドラインだけで`.jucer`から各種プロジェクトファイルをエクスポートできます。

Projucer からビルドファイルをエクスポートすると、Linux の場合は( デフォルトでは) `Builds/LinuxMakefile` というディレクトリが作られ、その中に `Makefile` が生成されます。`make -C Builds/LinuxMakefile` でこの `Makefile` に記述されたビルドを実行できます。ビルド結果は `Builds/LinuxMakefile/builds/` 以下に作られるので、その中から( VST3 の場合は) `*.vst3` などのディレクトリを`~/.vst3` 以下にコピーすれば使えるようになります。

Linux サポートに関しては、Projucer のプロジェクトにはひとつ問題があります。開発者が Linux 環境の存在を意識していない場合、Linux の `Makefile` をエクスポートする設定が含まれていない可能性があります。その場合、`.jucer` ファイルを編集して Linux 用ビルド 設定を追加する必要があります。

また、Projucer は JUCE のバージョンにも密接に結びついているところがあり、生成されたプロジェクトファイルが古いままで JUCE 本体をアップデートしたりすると、ビルドに失敗します。JUCE を使用したプラグインは JUCE を git submodule で指定していること多く、その JUCE のソースツリーに含まれている Projucer を使用してビルドするほうが確実です。

なお、JUCE で Linux 環境での VST3 がサポートされるようになったのは JUCE6 以降のみなので注意が必要です。もし VST3 版を使いたいプラグインで JUCE5.4.7 以前のバージョンが使われていたら、まず JUCE6 にアップグレードしてみるとよいでしょう。ただしこれはプログラムの修正が必要になるかもしれない、難しそうなら諦めて Windows 版の VST3 を yabridge で使うのも手です。

JUCE プロジェクトによっては、ビルドするユーザーが VST2 SDK をダウンロードして持っていることを前提としていることがあります。現在すでに入手不可能になっているので、すでに持っているユーザーでなければビルドが失敗することになります。Projucer を使っているプロジェクトであれば、そのファイルの `buildVST` とある部分を消すか `buildVST3` と書き換えて上記`--resave` コマンドで `Makefile` を生成し直すと、うまく行くことがあります。( この辺りまで来るとかなりプログラマー寄りの知識が必要になってきて、本書のスコープからは外れすぎてしまうので、これ以上のトラブルシューティングは本書の範囲外とします。)

### 3.4.3 waf

waf は Python で作られたビルドシステムのひとつですが、LV2 本家が waf を使っていることもあってか、LV2 プラグインのビルドによく使われています。近年では CMake や meson など waf 以外のビルドシステムも使われているようです。

waf が使われている場合は、`waf` というスクリプトファイルが含まれている場合がほとんどで、別途ビルドツールをダウンロードする必要は通常はありません。たまに LV2 の作者が作った `autowaf` という waf の補助ライブラリが git の submodule で参照されていることがあるので、( `autowaf` を使っている) LV2 プラグインのソースを git リポジトリからチェックアウトした場合は、`git submodule update --init recursive` を忘れずに実行しましょう。

## 第 4 章

# Linux で使える DAW

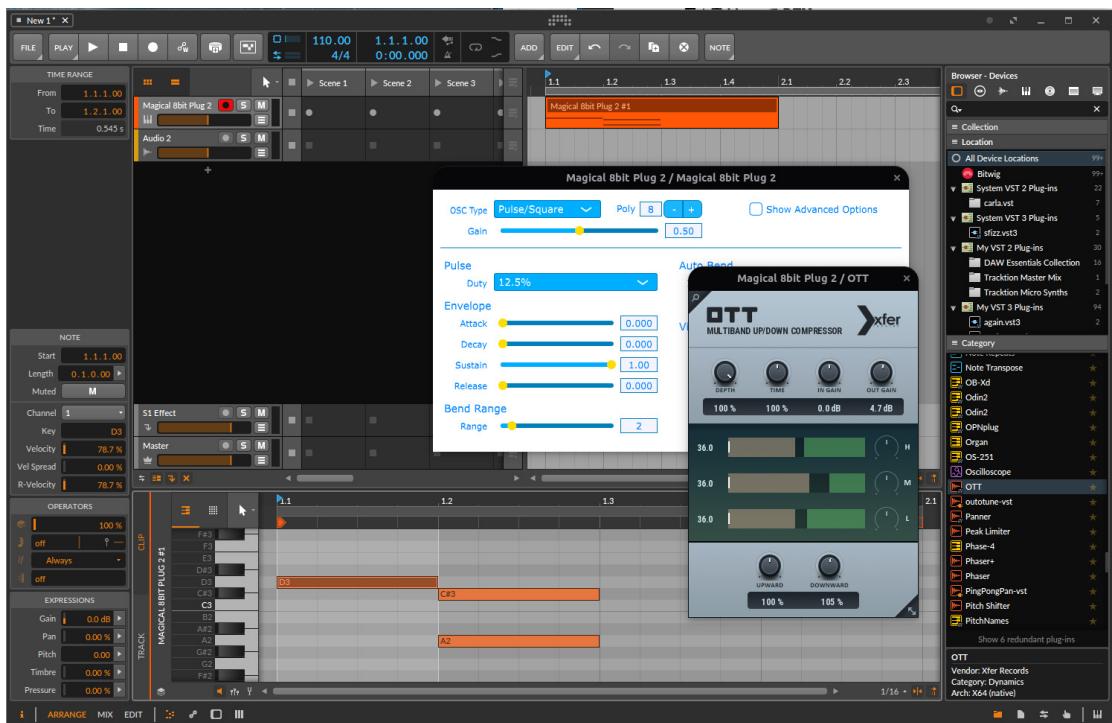
この章では Linux で使える DAW を商用製品・オープンソースとともに紹介していきます。商用 DAW は Windows 版・Mac 版があるもの（そしてそれらが主要なプラットフォームであること）がほとんどです。オープンソース DAW は Linux 版がメインのものが多いですが、やはり他のプラットフォームでも動作するものがほとんどです。いずれの場合も、利用できるプラグインの顔ぶれが（まだまだ）だいぶ異なるはずなので、ユーザーエクスペリエンスもそれなりに違うと想定しておいたほうがよいかもしれません。

本書では各 DAW の使い方全般を説明することはありません（それぞれ 1 冊の独立した本が書けるレベルになってしまっててしまうでしょう）。「このソフトが／こんなソフトが Linux でも使えるんだ」という情報を広めるのが本書の主な目的です。この章の残りの部分は、各 DAW の簡単な説明とスクリーンショットの掲載が中心となります。

## 4.1 Bitwig Studio

ドイツにある Bitwig 社の Bitwig Studio は、2009 年に Ableton 社の従業員らが独立して作り上げた DAW で、本書で取り上げる DAW の中では比較的一般層向けの直感的な GUI 構成をもち、その実 Java で作られている（ただし GUI は OpenGL 系の独自ライブラリを使っている）DAW です。2022 年には CLAP プラグインの開発を後押しする主要な企業としても知られるようになっています。

第1章でも言及しましたが、Bitwig が ALSA 仮想 MIDI デバイスを認識できない数少ない DAW のひとつです。MIDI 入力として仮想 MIDI デバイスを使用できません。さらに、Jack が ALSA MIDI をサポートするように設定されると Bitwig Studio から使えなくなるという罠があるようです。これは javax.sound.midi による MIDI アクセスが ALSA 経由でしか行えず、Jack が MIDI デバイスを占有すると ALSA から見えなくなってしまうためだと筆者は推測します。Bitwig Studio の Linux 版には実のところさまざまな落とし穴があり、KVR Forum では "Bitwig on Linux -- How to avoid common issues" というスレッドに情報がいろいろまとまっています。



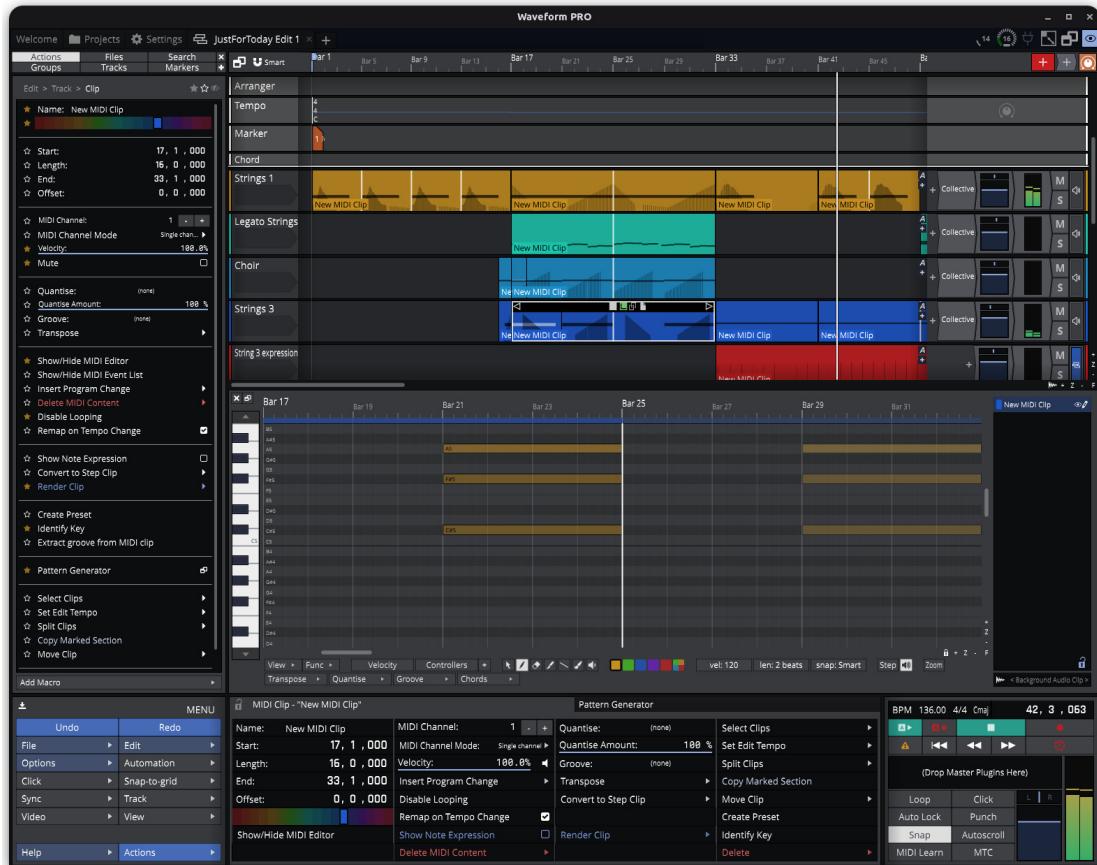
▲図 4.1 Bitwig Studio 4.2.5

## 4.2 Waveform Free/Pro

Tracktion 社の Waveform は、もともとは本書で何度も登場している JUCE を使用して作られた DAW であり、JUCE の開発者も元々は Tracktion のために開発していた側面があります。そのため、JUCE の最新バージョンの適用も比較的早く、オーディオプラグイン技術の最新機能も取り込まれやすくなっています。その上、DAW の根幹となる、演奏と編集を担う機能の非 GUI 部分が tracktion\_engine というオープンソースプロジェクトで GitHub に公開されており、誰でもこの DAW と同じ編集機能をもつ DAW を独自に構築でき、あるいは Waveform で打ち込んだ楽曲を独自のツールで再生できます。

Waveform はクローズドソースの有償製品で、かつては古いバージョンをフリー版としていたのですが、Waveform 11 か 12 の頃から最新版 Waveform の機能制限版をフリー版として配布するようになったので、Waveform11 以降でサポートしている Linux 用 VST3 もフリー版で使えます。

Waveform は、オーディオドライバーの選択が PulseAudio でも問題なく MIDI デバイスに接続できる希少な DAW のひとつで、筆者はこれを常用しています。

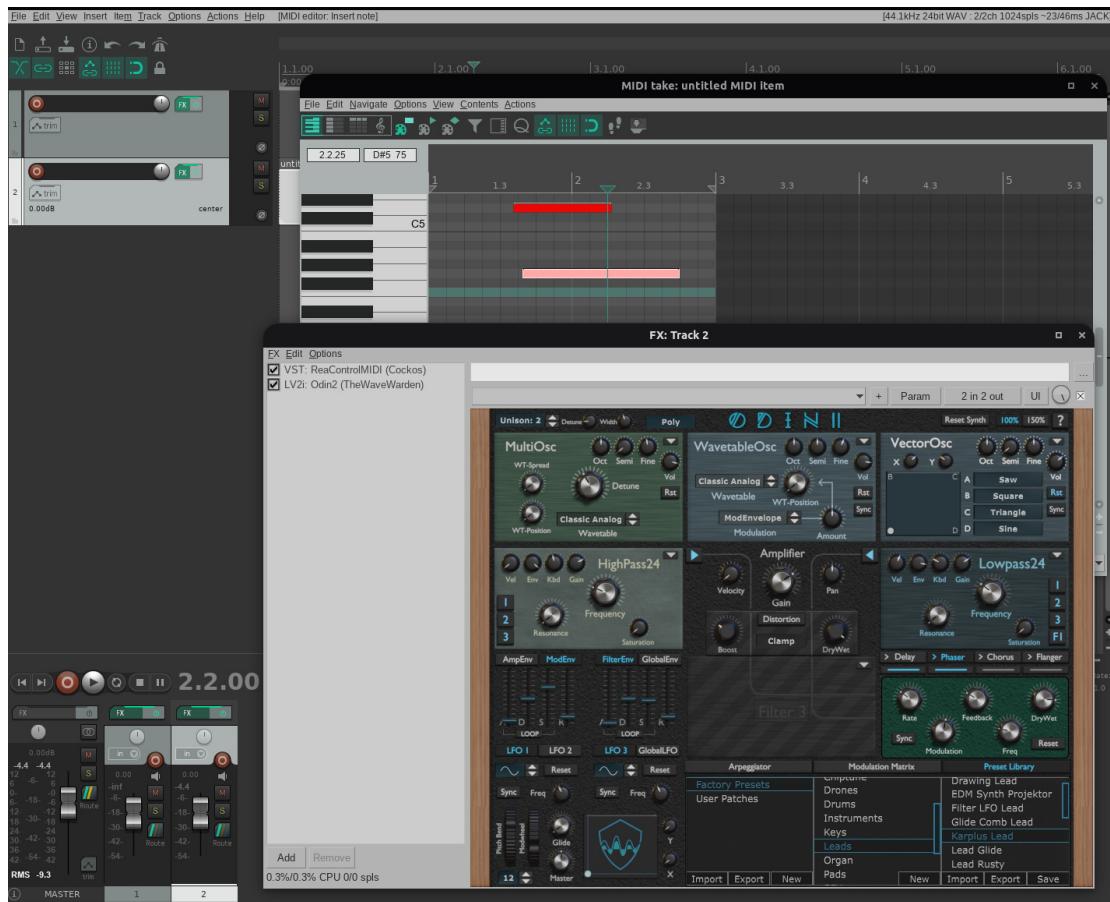


▲図 4.2 Tracktion Waveform 12

## 4.3 Reaper

Reaperはかなりギーク寄りの、プログラミング指向の強いユーザーが好んで使う傾向のあるDAWです。一方で、筆者が複数のDTMユーザーアンケート情報のサイトを参照した限り、この章で説明するDAWの中では、なんと日本人には一番使われているらしいDAWでもあります。最も人気があるとしても難解なDAWであることに変わりはないので、とっつきやすいものを好む読者には向いていないかもしれません。そういう読者は、もっと機能が少なくてシンプルなDAWを使うよいでしよう。

ReaperもオーディオAPIをALSA/JACK/PulseAudioから選択できますが、Reaper for Linuxでは仮想MIDIデバイスすなわちALSA rawmidiに出現しないALSA sequencerデバイスを接続できないようです。USB-MIDIデバイスはどのオーディオAPIを選択していても利用できます。



▲図4.3 Reaper 6.67

## 4.4 Renoise

Renoise は他の DAW とは毛色が大きく異なります。音楽制作ツールのカテゴリーとしてはトラッカーということになり、ピアノロールエディタの類は含まれていません。サンプラーとしての利用が中心となると思われますが、オーディオプラグインの類もロードできます。Linux 環境では VST2 と VST3、あと古いですが LADSPA や DSSI がサポートされています。

Renoise では ALSA と JACK からのみオーディオ API を選択できますが、筆者が非リアルタイム環境で試した限りでは ALSA デバイスへのアクセスは非排他的で、またその状態で ALSA sequencer デバイスも正しく認識されるので、ストレスの無いオーディオ作業のワークフローが確保できるでしょう。

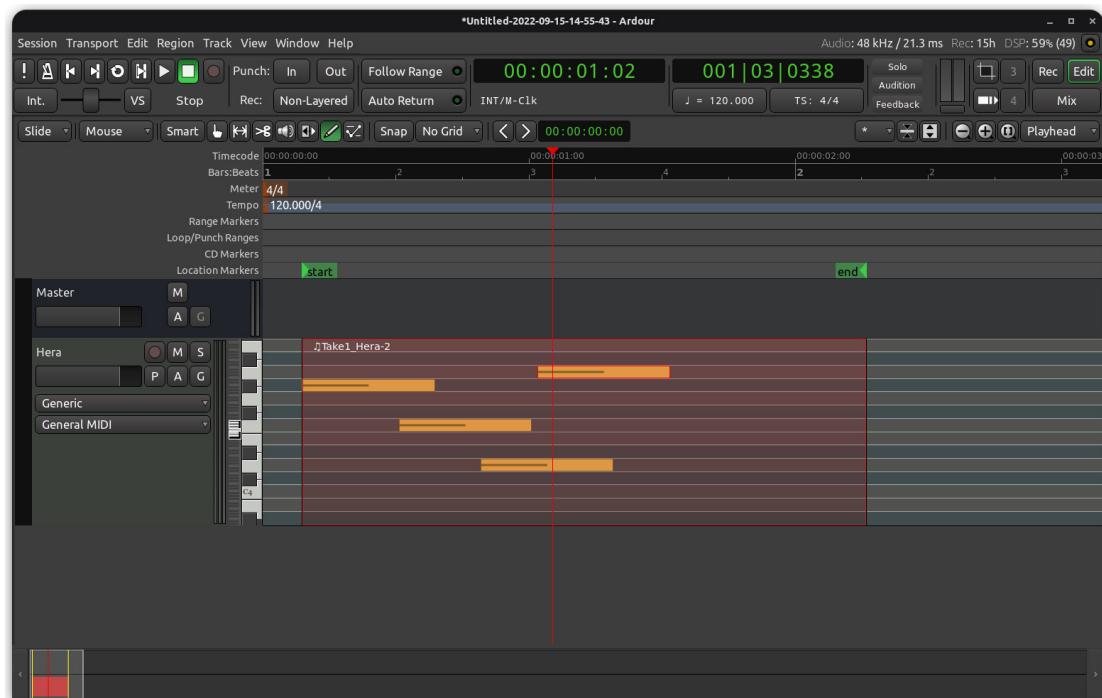


▲図 4.4 Renoise 3.4.2

## 4.5 Ardour / MixBus

ArdourはLinux音楽フリーソフトウェア界隈で最もポピュラーと思われる伝統的なDAWで、今でもアクティブに開発が進められています。ArdourはGPLv2で公開されており、ArdourをHarrison製品に合わせてカスタマイズして販売されているのがMixBusです。低レベルオーディオサポートでは幅広いAPIをカバーしていて、たとえばJackのバックエンドにALSAではなくOSSやNetJackなどさまざまなドライバーを利用できます。一方でPulseAudioを使っているとやはりMIDIデバイスを利用できません。筆者はqjackctlで事前に立ち上げていたJackサーバ、およびALSAで利用したときに、MIDIデバイスを接続できました。ALSAの場合もALSA sequencerに対応しているので、仮想MIDIデバイスも利用可能です。

GUIの作り方がとにかく特殊で、特にピアノロールエディターは一般的なDAWユーザーには最初のノート1つ打ち込むまでの操作方法が全くわからないまま使うのを諦めるかもしれません。筆者はトラックリストがピアノロールエディタを包含していてトラックの高さを買えないと全く表示されないということに気づくまでかなり時間がかかりました。MIDI入力をトラックに結びつけるUIもかなり特殊です。



▲図4.5 Ardour 6.9

## 4.6 Zrythm

Zrythm は新進気鋭のフリーソフトウェア DAW です。バイナリパッケージは有償販売されていますが、ソースコードは AGPL で公開されています。実装技術としては、Carla というプラグインホストのエンジン部分を利用して作られています。Carla がサポートするものは全てサポートされていて、LV2、VST、VST3、LADSPA、DSSI のほか、Carla が Fluidsynth を組み込んでいる SF2 サウンドフォントや SFZero を組み込んでいる SFZ サンプラーなどもカバーされています。

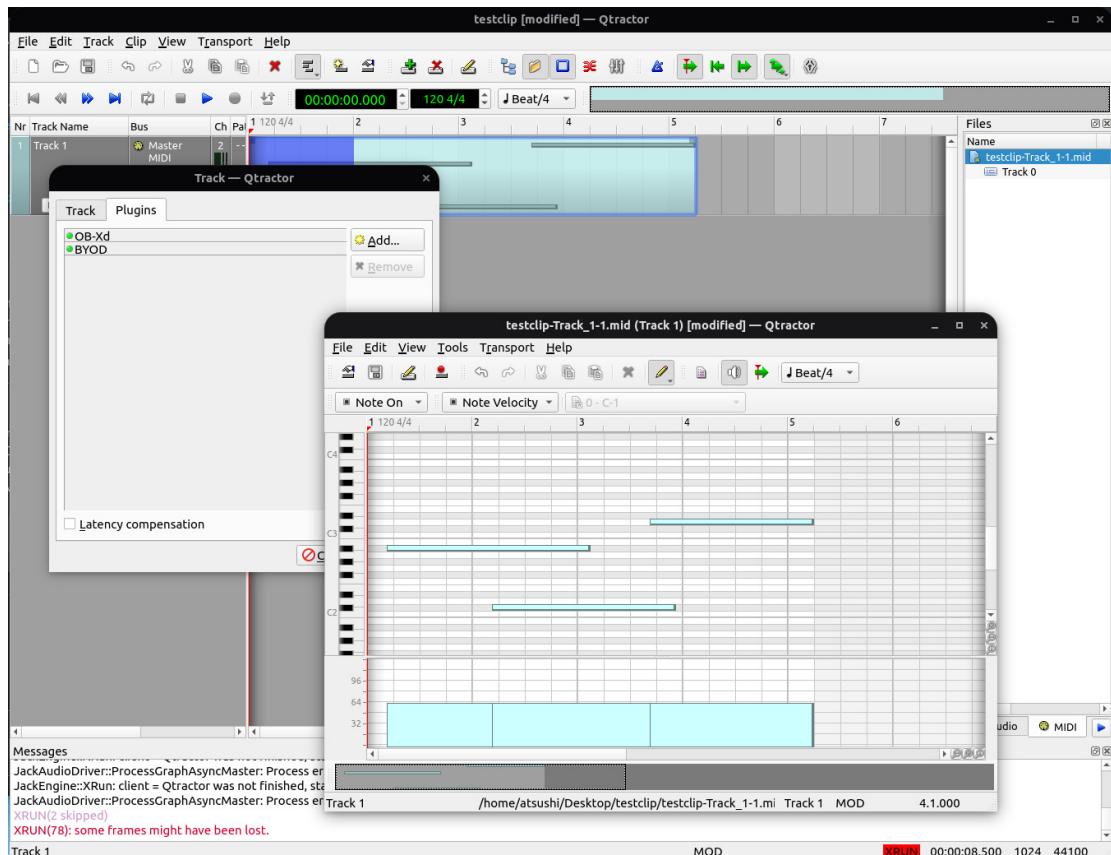
Zrythm はまだバージョン 1.0 に向けて開発が進んでいる状態で、2022 年の本書執筆時点では、Jack と PulseAudio が選択できるものの、ALSA サポートが "not working" とされている状態です。MIDI デバイスのサポートも Jack MIDI のみ実装されています。



▲図 4.6 Zrythm v1.0.0-beta.3.0.1

## 4.7 QTractor

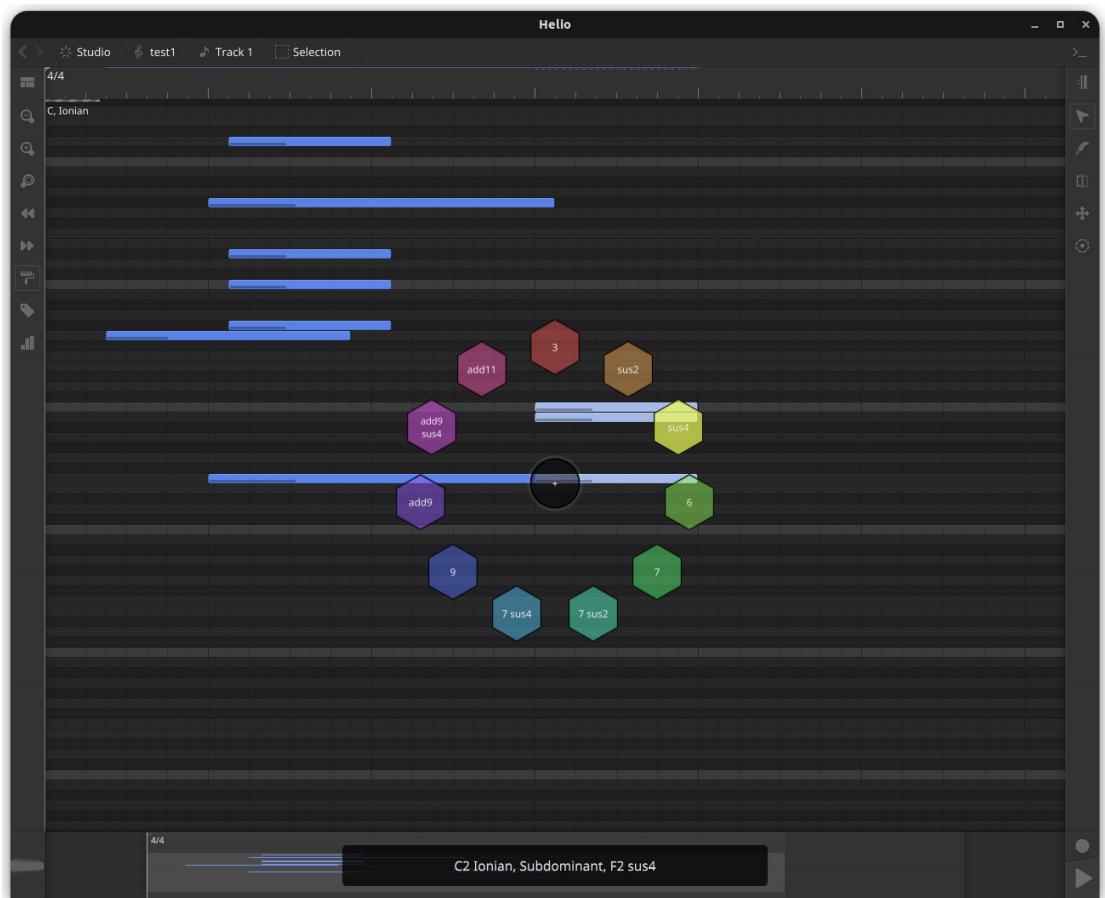
QTractorはQtで作られたDAWです。機能的にはシンプルなUIでMIDI1.0の範囲だけ作られているように見えますが、プラグインパラメーターのオートメーションも実装されていますし、CLAPプラグインのサポートなど先進的な機能にも取り組んだりしています。オーディオAPIサポートはJack一択です。



▲図4.7 QTractor 0.9.28

## 4.8 helio-workstation

helio-workstation は、ここまで紹介してきたものとはまたいぶ毛色の違う UI をもつ DAW です。作者は「現代の DAW は複雑過ぎる」として新しく直感的な設計を志向して作っているようです。helio は distraction free であるとも主張していて、いわゆる zen mode の雰囲気があります。helio は JUCE アプリケーションであり、オーディオレイヤーでは ALSA、Jack、PulseAudio を、MIDI 接続が壊れる心配なく利用できます。サポートするオーディオプラグインフォーマットは、バイナリパッケージでは VST2 と VST3 ですが、ソースからビルドすれば LV2 も簡単に追加できるでしょう。



▲図 4.8 helio-workstation 3.9

## 第 5 章

# Linux で使えるオーディオプラグイン

この章では Linux で利用可能なオーディオプラグインを紹介していきます。2022 年現在、Linux でも利用可能なオーディオプラグインもだいぶ増えてきて、「知っているもの」「動作するもの」をあるだけ紹介するようなフェーズではないので、「クロスプラットフォームで利用できるもの」「Windows や Mac で使うものとあまり変わらないもの」「シンプルで使いやすいもの」を中心に紹介していきます。筆者は DTM のプロフェッショナルではないので、豊富な経験に基づく推薦ができるほどの知見がありません。

本書ではプラグインを「サンプラー」「シンセサイザー」「エフェクト」に大分類して紹介します。ギタープラグイン、ピアノプラグイン、ドラムプラグイン…のような紹介にはなりません。これら生楽器の類は基本的にサンプラーで音源を探してください。リズムトラックの入力を補助できるプラグインのようなものも考えられますが、筆者が知るものには無いため、今回紹介するリストには含まれていません。

### 5.1 サンプラー

サンプラーはさまざまな音源から録音された PCM データに対して、音量や音程を自由に加工して楽器として利用するものです。サンプラーの技術は 20 世紀からありますが、PC 上で特定のシンセや処理系に依存しない汎用データとして利用できる可能性を示したのはサウンドフォントが起源であると考えてよいでしょう。

サウンドフォントは、SoundBlaster という PC 用の音源基盤を販売していた Creative Technology 社が、PCM 音源を活用して MIDI メッセージ入力をもとにあたかも MIDI 楽器であるかのようにオーディオデータを生成する仕組みであり、またその仕組みのために使われる PCM データをバンドルしたファイルのフォーマットもあります。

もともとは SoundBlaster というサウンドボード上で処理することを前提に作られたものでしたが、PC の性能が向上してソフトウェアのみでも十分に合成できるようになって、Creative Technology 社とは無関係に Timidity や Fluidsynth といったソフトウェアが誕生しました。

サンプラーには、そのソフトウェアで使用されるプロプライエタリなフォーマットを使うものと、オープンなデータフォーマットを使用するものがあります。また、独自フォーマットであってもポピュラーなものはさまざまな楽器サンプラーデータの作成者が有償・無償でデータを公開しています。オープンなデータフォーマットでは SF2 や SFZ がポピュラーです。クローズドなフォーマット

としては Native Instruments 社の Kontakt( NKI) も広く使われています。

この章ではサンプラーに限らずさまざまなプラグインを紹介するのが主目的です。筆者としてはお薦めのサンプラーデータやその収集方法についても詳しく解説したいところですが、内容のバランスが大きく変わってしまうため、本書の少なくとも初版では守備範囲外とします。

### 5.1.1 サウンドフォント (SF2)

SF2 はサウンドフォントのバージョン 2.0 として規定されたもので、一般的には .sf2 という拡張子のファイルで配布されています。バイナリフォーマットは公開されているので、誰でも作ることができます。Swami や Polyphone など、サンプラーソフトによっては SF2 をサポートしているものがあります。

#### Fluidsynth、JuicySFPlugin、Fluida.lv2

Fluidsynth は SF2「サウンドフォント」を利用するソフトウェア MIDI 音源です。ソフトウェア MIDI 音源はサンプラーとして機能させることができ、実際 Fluidsynth を JUCE に組み込んでオーディオプラグインとして利用できる Juicy SF Plugin などが存在しています。LGPL v2.1 で公開されており、20 世紀から存在している非常に息の長いソフトウェアです（ CVS リポジトリから移行した sourceforge.net のリポジトリの起源が 2003 年であり、筆者にはそれ以前の正確な起源を辿ることができませんでした）。

Fluidsynth 自体はオーディオプラグインではありませんが、Fluidsynth をオーディオプラグインとして使用できるプラグインとして JuicySFPlugin という JUCE プラグインと Fluida.lv2 という LV2 プラグインが開発されています。どちらも Linux 用バイナリは用意されていませんが、ソースからビルドが可能ですが…と言いたいところですが、実のところ JuicySFPlugin の Linux ビルドのサポートは流動的で、2022 年の本書執筆時点では古いバージョンである 1.0.8 のビルドのみが通ることになっています。

Fluida.lv2 は Fluidsynth を LV2 プラグインとして利用できるようにしたものです。Fluidsynth を利用したオーディオプラグインは FluidPlug や avldrums など他にもいくつかあるのですが、Fluidsynth 本家のように任意のサウンドフォントファイルを指定してロードできるものは、筆者は Fluida しか発見できていません。

### 5.1.2 SFZ

SFZ は SF2 とは全く異なるサンプラーフォーマットです。SFZ ファイルはテキストで書かれたスクリプトで、標準として規定された OpCode の命令を使用して記述します。テキストとしての SFZ ファイルにはサンプリングデータは含まれないので、SFZ ファイルを利用するには複数のファイルをまとめてコピーしたり配布したりする必要があります。SFZ サンプラーのデータは 1 つでギガバイト単位になるものもあります。

## sfizz

SFZ フォーマットを規定してきたのは Sforzando という製品を開発している Plogue という企業で、Sforzando は SFZ サンプラーの代表的な製品といえるでしょう。しかしこの製品はプロプラエタリで、Linux 版はありません。オープンソースで SFZ サンプラーを実装するソフトウェアはいくつか存在しますが、SFZ バージョン 1.0 の OpCode を全て実装して、2022 年の本書執筆時点でも継続的に開発されているのは sfizz 一択でしょう。



▲図 5.1 Sfizz

### 5.1.3 その他の製品

#### Hive

Hive は u-he 社の商用サンプラー製品で、独自のフォーマットを規定しています。そのデータフォーマットは製品をダウンロードするとそのアーカイブに含まれているので、誰でも参照できます（作成できるのかは筆者は確認していません）。Hive を含む u-he 社の製品は基本的に Linux 版も VST2 としてリリースされており、また u-he 社は CLAP プラグインフォーマットの開発を先導しているので、いずれ CLAP 版もリリースされることになるでしょう。



▲図 5.2 Hive

## 5.2 シンセサイザー

序文でも言及しましたが、筆者は各種のプラグインが、特にシンセサイザーが、それぞれ「これは一体何なのか」を説明できないことが多いです。理論的なカテゴリーが説明されていればそれを紹介できますが、公式サイトの説明が「これは神々の悦びである」みたいになっていることもあります（TheWaveWarden/odin2）、それ以上掘り下げられるのはひとえに筆者のシンセ経験の不足によるものです。幸いなことに、多くの場合、使い方が難しくてよくわからないシンセには「プリセット」でそのプラグインの開発者やユーザーが蓄積してきた音色のパラメーターセットがいくつも用意されているので、この節では読者がそれらに辿り着けるところまでは説明するつもりです。

## 5.2.1 ウエーブテーブル／ハイブリッドシンセサイザー

### Vital

Vitalは2020年末に登場したSpectral Warping wavetable synthesizerと称するシンセサイザーです。このジャンルでは商用製品であるXfer RecordsのSerumが有名ですが、そのSerumがもつ機能の多くを実現しています。Vitalのソフトウェア部分は完全にGPLv3で公開されていますが、パッケージとして配布されているのはプロプラエタリなデータを含んでいて（有償版も無償版もあります）、データ部分は単なるプリセット音色以外でも実のところさまざまな部分で設定可能なので（たとえば波形の選択肢などもありません）、オープンソース版をビルドして使用すると全く使い勝手が違うものになります。Linux版はLV2も含めバイナリパッケージを使用できれば一般的には十分でしょう。



▲図5.3 Vital

## Surge XT

Surge XT はウェーブテーブルを含むさまざまな波形ソースをオシレーターとして利用できるハイブリッドシンセサイザーです( ウェーブテーブルを含むのでここに分類しています)。開発チームが CLAP に注力していて、オープンソースのシンセで CLAP を試すときは第一候補と考えてもよいでしょう。長い歴史を持つプロジェクトで、プリセットも豊富に用意されています。



▲図 5.4 Surge

## Odin2

神々の悦びシンセサイザーです。3つのオシレーターそれぞれに、ウェーブテーブル他いくつかの種類のソースを設定できるハイブリッドシンセサイザーです。各種オシレーターソースにもフィルター要素にもクールなUIが作り込まれています。プリセット音色も豊富に用意されているのですが、右下のほうに"Arpeggiator", "Modulation Matrix", "Preset Library"と並んでいるボタンのうち最後のものがプリセット選択ビューを出すようになっています(この3つがタブページコントローラーになっています)。



▲図 5.5 Odin2

## 5.2.2 バーチャルアナログシンセサイザーなど

### OB-Xd

OB-Xd は Oberheim の OB-X をシミュレートする音源です。2017 年からある JUCE プロジェクトですが、現在でも最新の JUCE を使うようにアップデートされています。プリセットの変更は一見 GUI 上には見当たりませんが、右クリックのコンテキストメニューから選択できます。



▲図 5.6 OB-Xd

## Hera

HeraはRoland JUNO-60のエミュレーターをプラグイン化した音源です。JUNO-60やJUNO-106のエミュレーターは、Roland自身によるものも含めて、さまざまな製品が作られています。Heraはプラグインとして細かく作り込まれた音源というわけではありませんが、MPE(MIDI Polyphonic Expression)をサポートしているのが特徴です。



▲図 5.7 Hera

## OS-251

OS-251 はここまで紹介したシンセとは異なりバーチャルアナログエミュレーターではありませんが、JUNO-106 に近いパラメーターをもつシンセサイザーです。プリセットなども選択できるようになっています。Linux 版も公開されており、VST3 と LV2 がサポートされています。



▲図 5.8 OS-251

### 5.2.3 チップチューン音源・エミュレーター

#### Daxed

Daxed は YAMAHA の伝説的な FM 音源シンセ DX7 のエミュレーターとして機能するプラグインです。シンセサイザー部分は実は Google の従業員が Android 用に開発した music-synthesizer-for-Android (MSFA) というプロジェクトのコードですが、DX7 の音源プリセット集となる Cart のロード機能などプラグインとしてさまざまな機能が追加されて便利なものになっています。DX7 はさまざまな開発者がクローンを実装しているものですが、MSFA の実装は実機にさまざまなパラメーターを与えてオーディオデータを生成したものと機械的に比較しながら調整していくたという話が MSFA のソースコードのリポジトリに記録されています。



▲図 5.9 Daxed

## ADLplug/OPNplug

ADLplug は 20 世紀の PC に搭載されていた YAMAHA の FM 音源 OPL、OPM、OPN、OPN2などをエミュレートする音源のプラグインです。ADLplug が OPL の、OPNplug が OPM～OPN2 のエミュレーターになります(これら 2つのプラグインが1つのソースプロジェクトとして GitHub に存在しているので、本書では1つにまとめています)。これらの FM 音源では、Nuked、DOSBox、Opal、Neko、MAME などさまざまな実装が試みられてきましたが、それぞれにエミュレーションがうまく行くパラメーターのパターンとそうでないパターンがあります。ADLplug/OPNplug の面白い点は、それら複数のエミュレーター実装をスイッチひとつで切り替えられるところです。



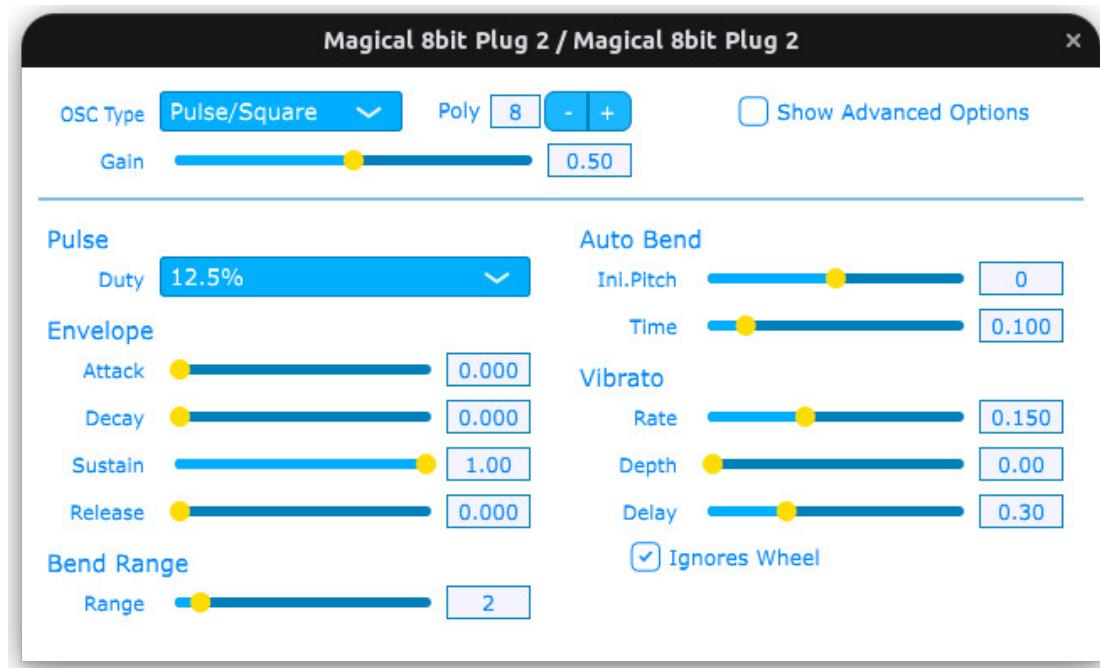
▲図 5.10 OPNplug

## Magical8bitPlug2

Magical8bitPlug2はYMCKというユニットから誕生した、いわゆる8bit musicあるいはチップチューンと呼ばれる類の、ファミリーコンピューターのようなサウンドを作るためのプラグインです。GitHubで公開されているオリジナルのソースコードにはLinuxサポートが含まれておらず、ビルド済みバイナリもありませんが、Linuxでビルトできるスクリプトを含むforkを公開している人がいます（本家にも変更取り込み要望が送られていますが本書執筆時点では取り込まれていません）。筆者もCLAP版をビルトできるCMakeLists.txtを作成しています。

（類似の音源にSANA\_8BIT\_VSTというものがありますが、こちらは古いProjucerを使って作られたもので、これを修正してビルトする手順を説明するのは煩雑になるので、本書では掲載しません。）

音色のプリセットの選択肢はほぼありませんが、OSC Typeで矩形波、三角波、ノイズを選択でき、その選択次第でさらにデューティー比やノイズタイプなどを選択できます。



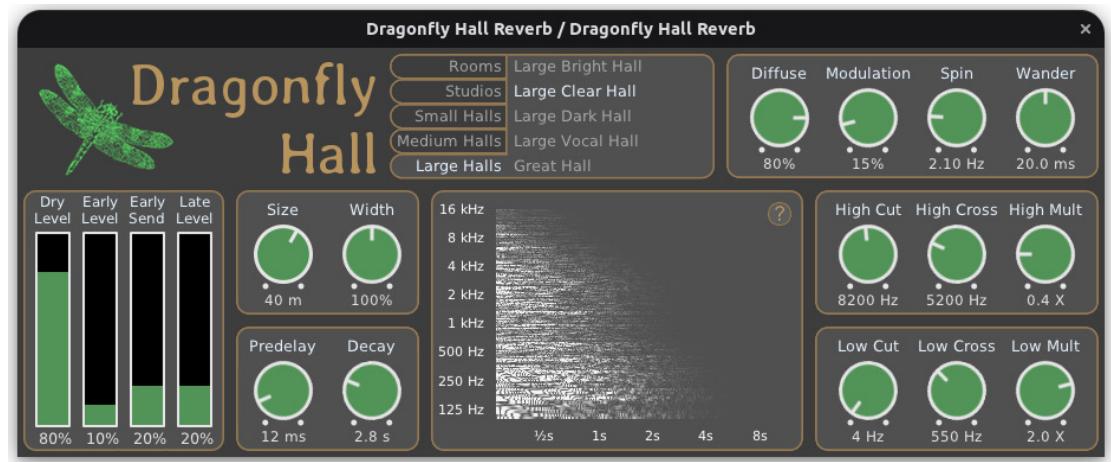
▲図 5.11 Magical8bitPlug2

## 5.3 エフェクトプラグイン

### 5.3.1 リバーブ

#### Dragonfly Reverb

Dragonfly-Reverb は DPF で作られた、インパクトの強いリバーブを適用できる使いやすいプラグインで、ホール／ルーム／プレート／アーリーリフレクションの 4 種類のリバーブが実装されています（別々のプラグインになっています）。リリースされているバイナリパッケージは VST2 と LV2 のみですが、GitHub にある最新のソースからは VST3 版もビルドできます。

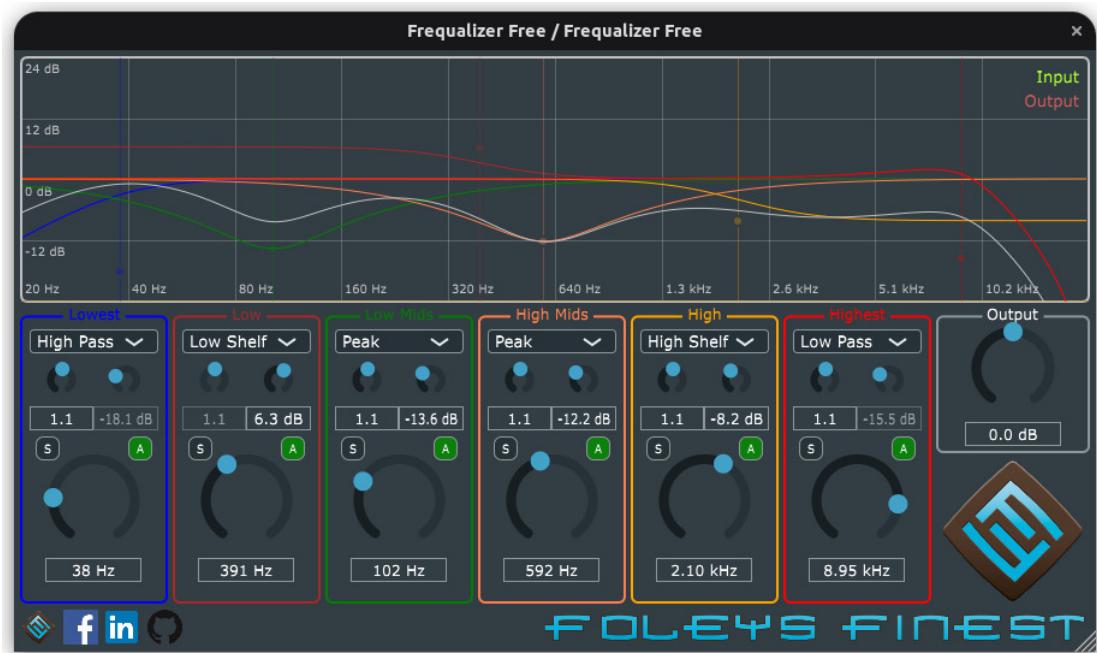


▲図 5.12 Dragonfly-Hall-Reverb

### 5.3.2 イコライザー

#### Frequalizer

Frequalizerは周波数アナライザーを表示できる6点パラメトリックイコライザーで、それぞれの制御点で11種類のフィルターを選択できます。制御点を探すのがやや難しいところがありますが、処理そのものは典型的なEQです。Linux版のバイナリパッケージはありませんが、GitHubにあるソースからビルドできます(筆者の環境では `CC=clang CXX=clang++ cmake ...` でビルドする必要がありました)。



▲図 5.13 Frequalizer Free

### 5.3.3 コンプレッサー

#### Squeezer

Squeezer は汎用コンプレッサーであると README では説明されています。開発者がコンプレッサーの何たるかを理解するためにさまざまな論文を読んで半年くらいで作ったようです。maximizer、transient shaper としての機能も実装されています。ソースからの Linux 用ビルトは用意されているようですが筆者の環境ではビルトが成功しなかったので、VST2 のバイナリパッケージをダウンロードして利用しています。

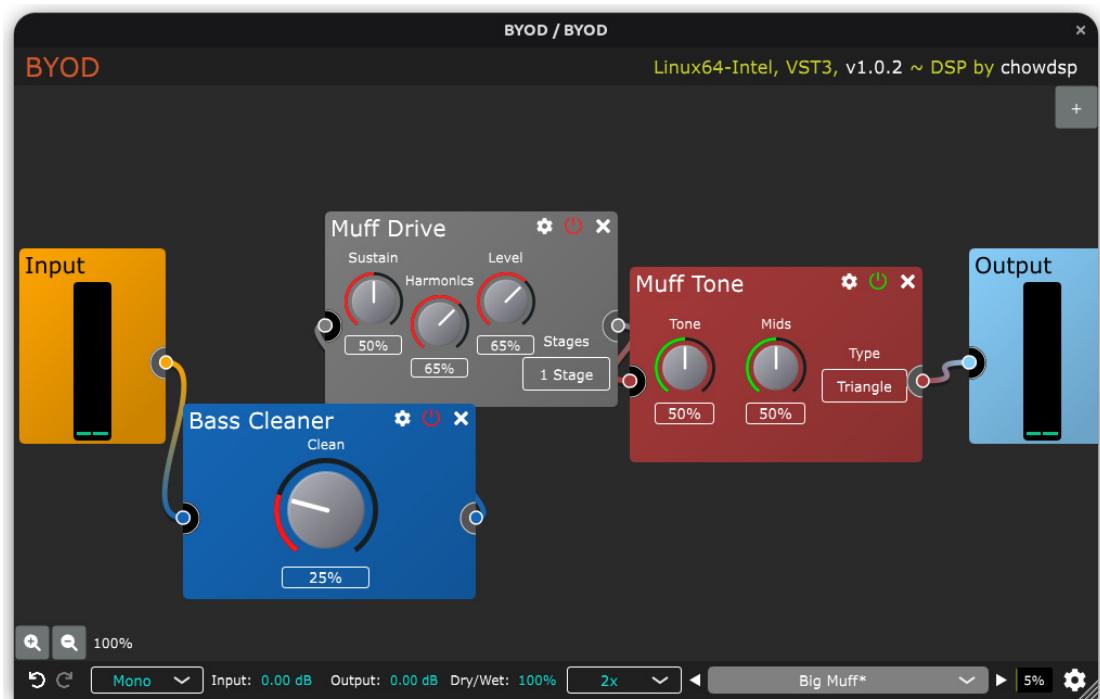


▲図 5.14 Squeezer

### 5.3.4 ギターアンプ、オーバードライブ、ディストーション

#### BYOD

BYODはBuild Your Own Distortion = ディストーションを自作・カスタマイズできるプラグインです。といっても、一般ユーザーにとっては "build" できるのは、エフェクターのユニット(フィルター)をオーディオグラフで選んで繋ぐ回路設計くらいでしょう。ユニットには機械学習でギター エフェクターのモデルを構築する GuitarMLも含まれているので、GuitarMLプロジェクトの成果も取り込みます。著名な製品やギタリストのエフェクター設定などがプリセットで用意されているので、それで遊ぶだけでも楽しめます。



▲図 5.15 BYOD

#### Guitarix

Guitarixは大抵のLinuxディストリビューションで公式パッケージとして入手できるギター エフェクターのコンポーネントが大量に入っているコレクションです。LV2プラグインで、LinuxとMacで使えるとされていますが、Mac版バイナリは公式配布されていないので「自前でビルトした人が過去にいた」程度かもしれません。

Guitarixプラグインは実際には数多くのgx\_\* .lv2というLV2パッケージに別れており、これをGuitarix単体で「積み上げて使う」(フィルターチェインにする)のは、スタンドアローンアプリであるguitarixが前提となっていて、そうでない環境ではやや使いにくいです。

### 5.3.5 フェイサー

#### ChowPhaser

Shulte Compact Phasing "A"というハードウェアのフェイサーに似せて作られているとされているフェイサー効果のプラグインです。



▲図 5.16 ChowPhaser

### 5.3.6 エフェクターコレクション

エフェクトプラグインは細かいものが多く、それぞれがさまざまな効能をもつので、ひとつのパッケージに小さなプラグインをたくさんまとめて配布されていることが少なくありません。ここではLinuxでも利用できるオープンソースのコレクションをいくつか紹介します。

#### mda-vst, mda-vst3, mda-lv2

MDAは(正確にはエフェクターだけではなくインストゥルメントも含むのですが)古くからあるプラグインのコレクションです。VST2版が作られ、VST3やLV2に移植されています。GUIは実装されていません。新しいプラグインフォーマットが開発された時にサンプルプロジェクトのように移植されがちですが、それぞれのプラグインはシンプルながら機能的です。

## slPlugins

slPlugins は SocaLabs というサイトで公開されている JUCE プラグイン集です。スクリーンショットで示すような小物のプラグインがいくつか含まれています。このサイトでは他にも、mverb や setbfree organ の移植などいくつか実用性のあるオープンソースのプラグインが JUCE に移植されて公開されています。



▲図 5.17 slPlugins

## 第 6 章

# 補遺: Linux で使える歌唱合成ソフト

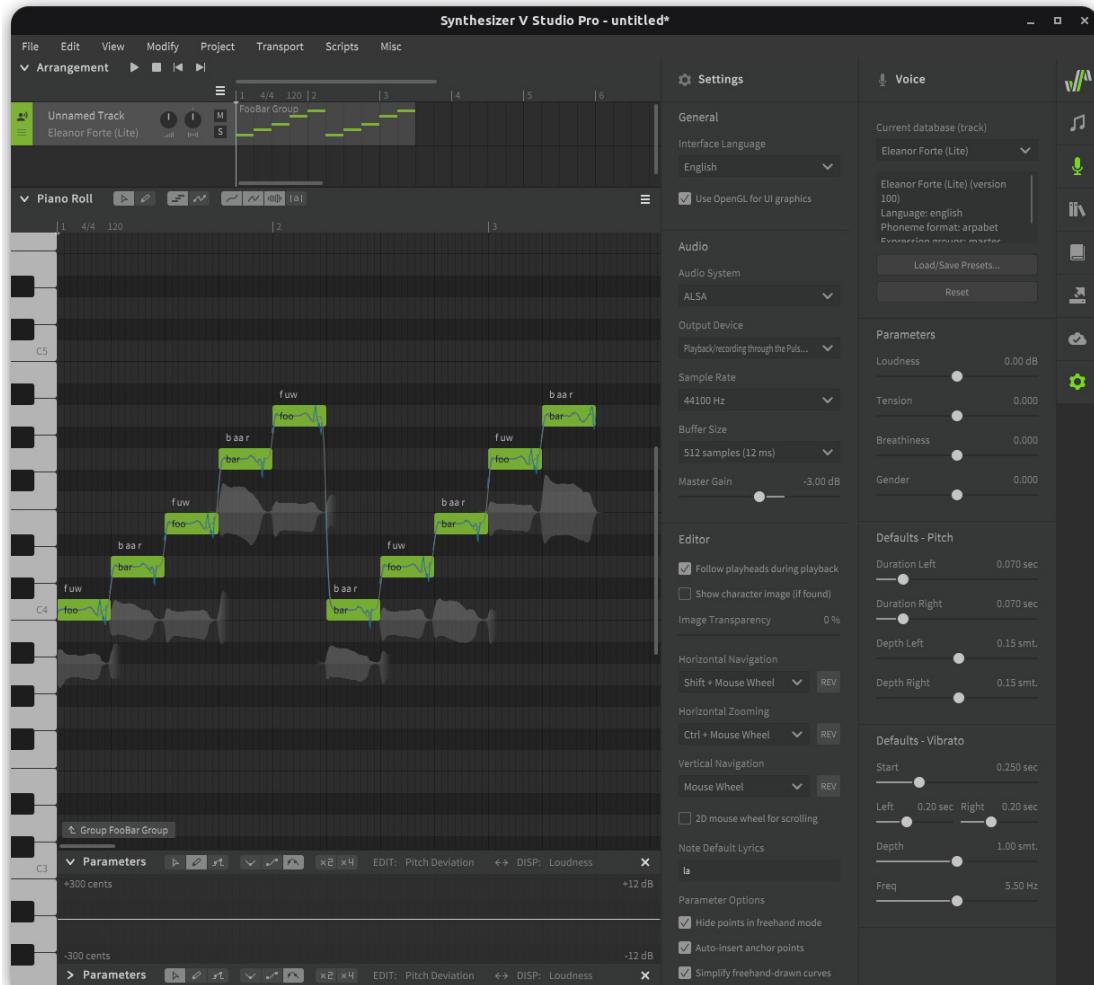
歌唱合成は DAW・オーディオプラグインとは独立したセクションにまとめます。オーディオプラグインは一般的にはリアルタイムで DAW から流れてくるオーディオ・MIDI 入力を処理できるような機能が作り込まれているもので、一般的には歌唱合成はリアルタイム処理には馴染まないものです。そのため、ヴォーカルパートを歌唱合成で作り込む場合は、いったん専用のツールから合成結果をオーディオデータとして出力しておいてそれを DAW のオーディオトラックに渡す、あるいは DAW からの入力とは無関係な出力を生成するようなプラグインが用意されていればそれを使うくらいでしょう。

Vocaloid や CEVIO シリーズのような伝統的な商用製品は Windows 専用あるいは Windows / Mac 専用のものが多いですが、モダンな歌唱合成ソフトウェアは、商用製品でも GUI まで含めてクロスプラットフォームで開発されているものもあり、Linux でも動作するものがいくつかあります。

### 6.0.1 Synthesizer V

Dreamtonics 社の Synthesizer V は AI 歌唱合成の先駆者の存在で、現在でも同市場の最先端を走る商用製品のひとつですが、同時に（Vocaloid や CeVIO シリーズとは異なり）初代から Linux をサポートし続けてきた製品もあります。歌唱合成技術については Vocaloid、UTAU などでいくらでも経験を積んだユーザーがいて関連情報が蓄積されているはずなので、本書で筆者が表層的な話を書くまでもないでしょう。AI 歌唱合成技術に関しては本節以降で取り上げるソフトウェアのユーザーが同様に情報を蓄積していると考えます。

Synthesizer V は JUCE を利用していて、オーディオ基盤は DAW の章で見てきたとおり最もトラブルの少ない構成になっていると考えてよいです。



▲図 6.1 Synthesizer V

## 6.0.2 NEUTRINO

NEUTRINOは「東北きりたん」などのエンジンとして使われているニューラルネットワーク歌唱合成のソフトウェアです。オープンソースではありませんが無料でダウンロードできる、いわゆるフリーソフトウェアではないフリーウェアです。NEUTRINO自体はコンソールツールでモデルデータにもGUI要素はありません。NEUTRINOの公式サイトでLinux環境での使い方も説明されています。

調整ツールというものが別の開発者から公開されていますが、これはWindows専用です(WPFが使われているのでwine-monoでも動作しないでしょう)。Linux上で使用するなら、NEUTRINOがデータフォーマットとして採用しているMusicXMLをMuseScore等の譜面作成ツールで編集して、コマンドラインツールで渡すアプローチで使うことになるでしょう。



# **Linux デスクトップ DTM ガイド**

---

2022 年 9 月 25 日 技術書典 13 版 v0.9.0

著 者 atsushieno

編 集 atsushieno

発行所 オーディオプラグイン研究所

---

(C) 2022 atsushieno