空気録学

空気×録学

RE:TRIED.

改花。



2016 691 DECEMBER

sooyen [tax out]

誰の為じゃなく
録るしかない。
思う侭に

クウキロクガク

平成28年12月29日発行
（コミックマーケット発行）同日頒布
第2号

いつの間にか、我を忘れ、時を忘れ、周りが見えなくなるほどの恋に、無駄な時間は1ビットもない。

✽オープンソース DTV チューナーサーバー「**Mirakurun**」取扱説明書 第二版（RC 対応）

ぴゅあぴゅあ Mirakurun 

第１章・入門編 06

1.1 Mirakurun とは？ ・・・・・・・・ 06
1.2 動作環境 ・・・・・・・・・・・・・・・・ 08
1.3 インストール ・・・・・・・・・・・・ 09
1.4 アップデート ・・・・・・・・・・・・ 10
1.5 アンインストール ・・・・・・・・・・ 10
1.6 CLI ・・・・・・・・・・・・・・・・・・・・ 11
1.7 初期設定 ・・・・・・・・・・・・・・・・ 12
1.8 プライオリティ ・・・・・・・・・・・ 14
1.9 ログファイル ・・・・・・・・・・・・・ 15

第２章・活用編 16

2.1 Rivarun ・・・・・・・・・・・・・・・・・ 16
2.2 Chinachu[beta] で使う ・・・・・・・ 17
2.3 TVTest で使う ・・・・・・・・・・・・ 20
2.4 VLC で使う ・・・・・・・・・・・・・・・ 22
2.5 VLC for iOS で使う ・・・・・・・・ 23
2.6 外出先から使う ・・・・・・・・・・・・ 24

第３章・上級編 25

3.1 高度な設定 ・・・・・・・・・・・・・・・ 25
3.2 Open API / REST API ・・・・ 28
3.3 Munin で監視する ・・・・・・・・・ 29

第４章・開発編 32

4.1 アーキテクチャ ・・・・・・・・・・・・ 32
4.2 ビルドとデバッグ ・・・・・・・・・・ 34
4.3 Change Log ・・・・・・・・・・・・・ 35

第1章

入門編

Mirakurun の概要、インストール方法から使い方まで

1.1 Mirakurun とは？



←1-1-1

Mirakurun（ミラクるん）は、Chinachu Air 向けに開発を行っているデジタル放送番組用のチューナーサーバーです。 これを使えば、LAN 環境内ないしプロセス間において任意のチューナーを柔軟に共有することができます。Chinachu Air 専用かといえばそうでもなく、インターフェースとして Open API をベースとしたシンプルな REST API を備えていますので、簡単にクライアントを実装することができます。すでに Rivarun（ライバるん）☞P.16 や TVTest(BonDriver) ☞P.20 をはじめとした対応クライアントが登場しています。Chinachu[beta] ☞P.17 でも使えますので、この機会にぜひお試しください。

❶ 公式リポジトリはこちら ☞ https://github.com/Chinachu/Mirakurun

❖ 高度なチューナープロセス管理

今日の組込製品を除いた Linux 環境において放送波を取得する場合、インターフェースとして recpt1 のような録画コマンドや DVB インターフェースを利用するのが一般的です。

しかしこれらはチューナーデバイスを制御し結果復調された MPEG-2 TS を出力するだけの非常に低レイヤーなものです。実際の視聴環境に落とし込むには、排他処理をはじめとした様々な処理を利用するソフトウェア側で実装しなければならず、複雑になりがちです。

Chinachu[beta] でもそういった処理を実装していましたが、メンテナンスコストが大変高く、多くの問題を抱えていたため、Chinachu Air を開発するにあたっては、この部分を分離して独立したサービスとして動作させるというのが第一課題でした。

そこで誕生したのが Mirakurun です。Chinachu[beta] で抱えていた問題を一つ残らず解決すべく、Chinachu Air 開発プロジェクトの一環として開発を行っていますが、他のソフトウェアでも活用できるように、なるべく汎用的に設計しています。

視聴・録画ソフトウェアに必要なチューナー排他制御や自動フェイルオーバー、マルチストリーム配信などの高度なチューナープロセス管理機能を透過的に提供しています。

❇ プライオリティ管理 ☞ P.14

Mirakurun は同時に複数クライアントへのストリーム配信をサポートしています。たとえば Chinachu で録画をしながら TVTest で番組をリアルタイム視聴するといったことが可能になる機能です。

同一のチャンネルであれば一つのチューナーのみを使用しますが、チャンネルが違えば当然チューナーの数には限りがありますので、チューナーの取り合い合戦が発生します。そのためのプライオリティ管理機能があります。大切な予約録画を優先させ、録画事故を防ぐことができるのです。

❇ MPEG-2 TS パーサー・パケットフィルター

Mirakurun は MPEG-2 TS パケットを常に読み取り、必要に応じて最低限の解析処理を行っています。これはいわば Mirakurun の心臓部であり、Service ID や Event ID 指定に対応するための実装もここにあります。

Service ID 指定によって要らないパケットを切り捨てた場合は必要に応じて PAT パケットを書き換えて、正常な MPEG-2 TS を生成します。

❇ リアルタイム EPG パーサー

Mirakurun は EPG もパースし、JSON 形式で保存しています。これは REST API で提供しているため、クライアントの EPG を取得するコストを低減できます。

チューナーに空きがある限り最低のプライオリティで EPG の取得を自動的に行っていますので、いつでもほぼ最新の EPG を簡単に取得できます。

❇ Open API / REST API ☞ P.28

Mirakurun が備えたインターフェースは Open API をベースとしたシンプルな REST API のみです。これによりサーバーもクライアントも、シンプルな設計になることを狙っています。

1.2 動作環境

Mirakurunを動作させるために、環境を整えましょう。基本的にはNode.jsがあれば動作します。

※ recpt1 等の録画コマンドやドライバは別途必要です。

❇ Linux

Mirakurun は主に Linux での動作を想定しています。通常は Linux をお使いください。

- **Distribution** … **Debian** / Ubuntu / CentOS / Gentoo
  - ⚠ サーバー版が存在する場合はそちらを推奨。
    デスクトップ環境は非推奨。
- **Architecture** … x86 / **x64** / ARMv7 / ARMv8
- **実行環境** … Node.js `>= 6.5.0`
  PM2 `>= 2.0.12`

❇ Win32

実験的ではありますが、筆者は Windows 上の VS Code で開発・デバッグを行っているため、Windows でも動作するようにしています。Bash on Ubuntu on Windows ではありません。



←1-2-1

- **Version** … Windows 10 RS1 (x64)
  - ※ 筆者により動作確認済みのもの。
- **実行環境** … Node.js `>= 6.5.0`
  winser `>= 1.0.2`

❇ Darwin

たぶん、Mac OS X でも動くと思って開発していますが、残念なことにテストする時間がありませんでした…。対応チューナーに関してもよく分かっていません。DVB デバイスで使えるんじゃないかなぁ。**だれか動かす事ができた方はぜひご一報ください。そして次号で記事を書いてください。**

1.3 インストール

それでは早速、Mirakurun をインストールしましょう。ここでは、Debian 8.5 の amd64 版を例に使用しています。

❀ 動作に必要なライブラリ等をインストール

```
root@kurumi:~# apt-get install -y curl build-essential
```

❀ Node.js のインストール

Debian / Ubuntu 以外の方はこちら ☞ https://nodejs.org/en/download/package-manager/ に手順が記載されていますのでご確認ください。 ⚠ バージョン間違いに注意

```
root@kurumi:~# curl -sL https://deb.nodesource.com/setup_6.x | bash -
root@kurumi:~# apt-get install -y nodejs
```

❶ Node.js 及び NPM のバージョンは **`node -v && npm -v`** で確認できます。

❀ PM2 のインストール

Linux における Mirakurun のデーモン化およびプロセス管理には PM2 を採用しています。

```
root@kurumi:~# npm install pm2 -g
```

❶ PM2 プロジェクトサイト ☞ http://pm2.keymetrics.io/

❀ Mirakurun のインストール

Mirakurun のインストールは七森中録画研究会史上最も簡単です ♨

```
root@kurumi:~# npm install mirakurun -g --unsafe --production
```

このコマンド一発でインストールが完了し、PM2 によって自動起動されるようになります。

❶ 起動状態は **`mirakurun status`** で確認できます。計算機を再起動しても正常に自動起動されるかテストすることをお勧めします。

▶ 次は**初期設定** ☞ P.12 へ進みましょう。

1.4 アップデート

Mirakurun は今後も機能実装やバグ修正等のためにバージョンが更新されていきます。Twitter や Slack ☞ https://chinachu.moe/ でのお知らせを確認するか、定期的にアップデートを行うことをお勧めします。

❇ Mirakurun のアップデート

アップデートを行う前に、大事な予約録画などが実行中でないか確認してください。

```
root@kurumi:~# npm install mirakurun@latest -g --unsafe --production
```

ℹ サービスは自動的に再起動されます。

❇ PM2 のアップデート

Mirakurun のアップデートを行った際に PM2 のアップデートを促されることがあります。その他必要に応じてアップデートを行ってください。

```
root@kurumi:~# npm install pm2@latest -g
root@kurumi:~# pm2 update
```

❇ Node.js のアップデート

同じく必要に応じて Node.js のアップデートを行ってください。先の手順に従ってインストールした場合は次のコマンドでアップデートできます。

```
root@kurumi:~# apt-get upgrade -y nodejs
```

1.5 アンインストール

何らかの事情により Mirakurun をアンインストールする場合は次のコマンドです。

```
root@kurumi:~# npm uninstall mirakurun -g --unsafe
```

⚠ 設定ファイル及びデータベースは下記のパスに残ります。

- **/usr/local/etc/mirakurun/**
- **/usr/local/db/mirakurun/**

1.6 CLI ⚠ Windows 非対応

Mirakurun にはサービスの管理などを行うための CLI（コマンドラインインターフェース）が用意されています。

CLI は root 権限で **`mirakurun <command> ...`** といった形式で使えます。**`<command>`** には以下のコマンドが実装されています。

- **`config server`** … サーバー設定
- **`config tuners`** … チューナー設定
- **`config channels`** … チャンネル設定

- **`log server`** … サーバーログ表示

- **`status`** … サービスステータス
- **`start`** … サービス開始
- **`stop`** … サービス停止
- **`restart`** … サービス再起動

- **`help`** … コマンド一覧とか

❄ 設定

設定ファイルを表示・編集できます。

```
root@kurumi:~# mirakurun config [server|tuners|channels]
```

❄ ログストリーム

ログをリアルタイムで確認できます。

```
root@kurumi:~# mirakurun log server
```

❄ サービス管理

サービスを簡単に管理できます。

```
root@kurumi:~# mirakurun [status|start|stop|restart]
```

1.7 初期設定

まず最初に、チューナーとチャンネルを設定する必要があります。サーバーはデフォルトで TCP 40772 番ポートで待ち受けるようになっています。詳しくは**高度な設定** ☞ P.25 を参照してください。

設定ファイルの形式は YAML です。初めて触れる方はこの機会にぜひ覚えてみてください。

設定ファイルに誤りがある場合は起動時にエラーログが出力されます。**ログファイル** ☞ P.15 をご確認ください。

❇ チューナー設定

利用するチューナーデバイスに合わせて設定します。

```
root@kurumi:~# mirakurun config tuners
```

❶ あらかじめ参考用の設定が入っています。そのまま使う場合は **isDisabled** を **false** に変更してください。 ⚠ ここで設定したチューナーは Mirakurun 専用として他のプロセスでは使用しない

次のプロパティは必須ですので、必ず指定してください。

- **name** … チューナーデバイスを識別するための名前
- **types** … 放送波タイプ **[GR|BS|CS|SKY]**
- **command** … 標準出力で MPEG-2 TS を吐く録画コマンドまたは DVB tune コマンド

また、次の任意のプロパティがあります。

- **dvbDevicePath** … DVB デバイスである場合に設定
- **decoder** … スクランブル等を処理するためのコマンドを設定
  ⚠ EPG 取得時やクライアントからの要求パラメーターによっては使用されません。
- **isDisabled** … 無効化する場合は **true** と設定

実装参考用の **decoder** は次のコマンドでインストールできます。

```
root@kurumi:~# npm install arib-b25-stream-test -g --unsafe
```

❀ チャンネル設定

地域や環境に合わせて設定します。

```
root@kurumi:~# mirakurun config channels
```

❶ あらかじめ参考用の設定が入っています。**isDisabled** が **true** になっているものがありますので注意してください。

次のプロパティは必須ですので、必ず指定してください。

- **name** … チャンネルを識別するための名前
- **type** … 放送波タイプ **[GR|BS|CS|SKY]**
- **channel** … チューナーの **command** に渡す **<channel>** 引数 ⚠ 必ず文字列型で設定

また、次の任意のプロパティがあります。

- **serviceId** … **GR** 以外の放送波タイプに設定 ⚠ 必ず整数型で設定
- **satelite** … チューナーの **command** に渡す **<satelite>** 引数
- **isDisabled** … 無効化する場合は **true** と設定

❀ Mirakurun を再起動

設定を反映させるには再起動します。

```
root@kurumi:~# mirakurun restart
```

❶ 自動的にサービス名のスキャンが行われます。そのまま 2~3 分程度お待ちください。

❀ 動作確認

ブラウザまたは **curl** コマンドを用いて動作確認ができます。これで初期設定は完了です ♨

```
http://host:port/api/tuners
```

▲ 有効なチューナー一覧が JSON 形式で表示されます。

```
http://host:port/api/channels
```

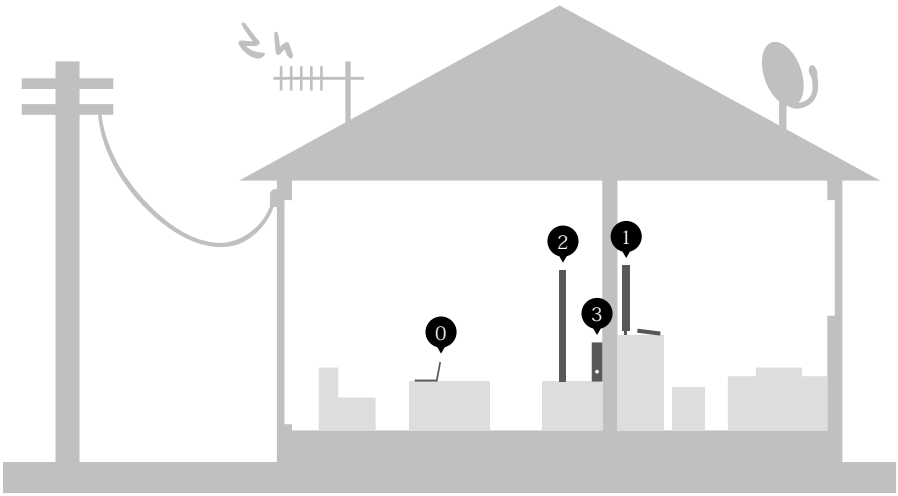
▲ 有効なチャンネル一覧および検出されたサービスが JSON 形式で表示されます。

```
http://host:port/api/programs
```

▲ 正常に EPG が取得できている場合、JSON 形式で何か表示されます。

1.8 プライオリティ

Mirakurun にはストリーム要求時にプライオリティを付けてチューナーリソースを管理する仕組みがあります。これを活用すると、チューナーの取り合い事故を未然に防げます。



←1-8-1

❊ プライオリティの指定

プライオリティは 0 が最低で、数が大きいほど高くなります。上限値は特にありませんが、各自適度に設定値を管理してください。

❶ REST API におけるプライオリティ指定方法は ☞ P.28 を参照してください。

❊ 定義済みプライオリティ

- `-1` … Mirakurun 自身によるチャンネル・サービススキャン／ EPG 取得処理
- `0` … デフォルト（選択可能な最低プライオリティ）

❊ 使用チューナー決定ルール

ストリーム要求時に使用されるチューナーは以下の優先順位で決定されます。

❶ 指定チャンネルが選局済みのチューナー（共聴）
❷ 空きチューナー（新規）
❸ プライオリティが要求よりも低い他チャンネル選局済みのチューナー（奪取）

1.9 ログファイル

何か気になることがあれば、まずはログを確認しましょう。ログストリーム ☞P.11 も活用してください。

❀ ログファイルの場所

- 通常ログ　… **/usr/local/var/log/mirakurun.stdout.log**
- エラーログ　… **/usr/local/var/log/mirakurun.stderr.log**

❀ ログローテートの設定

ログファイルは放置していると増え続ける一方です。トラブルシューティングの妨げにもなりますので、ログローテートの設定をすることをお勧めします。

設定例 : **/var/logrotate.d/mirakurun**

```
/usr/local/var/log/mirakurun.stdout.log /usr/local/var/log/mirakurun.stderr.log {
    weekly
}
```

第 2 章

活用編

Mirakurun を活用しましょう

2.1 Rivarun

Rivarun（ライバるん）は Mirakurun のシンプルなクライアント実装サンプルとして開発されました。`recpt1` のように使えるコマンドラインアプリケーションです。インストール・実行には Mirakurun と同様に Node.js がインストールされている必要があります。Windows PowerShell にも対応しています。

ℹ 公式リポジトリはこちら ☞ https://github.com/Chinachu/Rivarun

❇ インストール

```
root@kurumi:~# npm install rivarun -g
```

ℹ Rivarun のバージョンは **`rivarun --version`** で確認できます。

❇ 使い方

`rivarun` コマンドは次のように root でなくても使えます。

```
rivarun [--b25] [--mirakurun 接続先ホスト] [--priority プライオリティ] [--sid サービス ID] [--ch 放送波 / チャンネル] 録画時間 保存先
```

ℹ 便宜上 **`--b25`** としていますが、Mirakurun で設定された **`decoder`** の使用を要求するオプションです。
ℹ 接続先ホストを省略した場合、**`/var/run/mirakurun.sock`** または **`\\.\pipe\mirakurun`** に接続します。
ℹ 録画時間を `-` と指定した場合、コマンドを中断するまで録画し続けます。
ℹ 保存先を `-` と指定した場合、標準出力を利用します。

❀ 使用例

BS11 を 15 秒 bs11.ts に録画する

```
raika@kurumi:~$ rivarun --b25 --sid 400211 15 bs11.ts
```

TOKYO MX 全サービスを 15 秒 mx.ts に録画する

```
raika@kurumi:~$ rivarun --b25 --ch GR/16 15 mx.ts
```

60 秒録画しながら FFmpeg で sptv.mp4 に変換する

```
raika@kurumi:~$ rivarun --sid 700100 60 - | ffmpeg -y -i pipe:0 sptv.mp4
```

Atd を使って直近の 7 時の MX ニュース番組を録画予約する

```
raika@kurumi:~$ echo "rivarun --sid 3239123608 5400 mx1.ts" | at 07:00
```

最初期の Chinachu も atd を使っていました。ストイックな録画予約ツールとして使えるかも？

2.2 Chinachu^{beta} で使う

2.3 TVTest で使う

TVTest とはデジタル放送総合技術研究開発機構より頒布されている、デジタル放送汎用視聴プログラム実装研究資料です。これは Windows 用アプリケーションです。



➡2-3-1

Mirakurun に対応した BonDriver を用意することで使えるようになります。

❶ 公式リポジトリはこちら ☞ https://github.com/DBCTRADO/TVTest

❇ インストール

インストール方法についてはインターネット上の情報を参考にしてください ☁

❇ 対応している BonDriver 一覧

- BonDriver_HTTP
- BonDriver_Mirakurun

HTTP で MPEG-2 TS を取得する類(たぐい)の BonDriver であれば概ね使えると思いますが、今回はせっかくなので **`BonDriver_Mirakurun`** を使ってみましょう。

❋ BonDriver_Mirakurun のダウンロード

GitHub にアクセスします ☞ https://github.com/Chinachu/BonDriver_Mirakurun/releases
ZIP をダウンロードして展開します。

BonDriver_Mirakurun.dll と **BonDriver_Mirakurun.ini** の両方を TVTest と同じディレクトリにコピーします。

❋ BonDriver_Mirakurun の設定

BonDriver_Mirakurun.ini を環境に合わせて編集します。

```
[GLOBAL]
SERVER_HOST="192.168.1.101"
SERVER_PORT=40772

; B25(1=enable)
DECODE_B25=1

; Priority(0=Low Priority)
PRIORITY=0
≈
```

❶ クライアント側でデコードする場合は **DECODE_B25** を **0** にします。すると、Mirakurun の負荷が軽減されます。

❋ チャンネルスキャン

TVTest を起動して BonDriver を選択し、チャンネルスキャンを実行します。



2-3-4 ➡

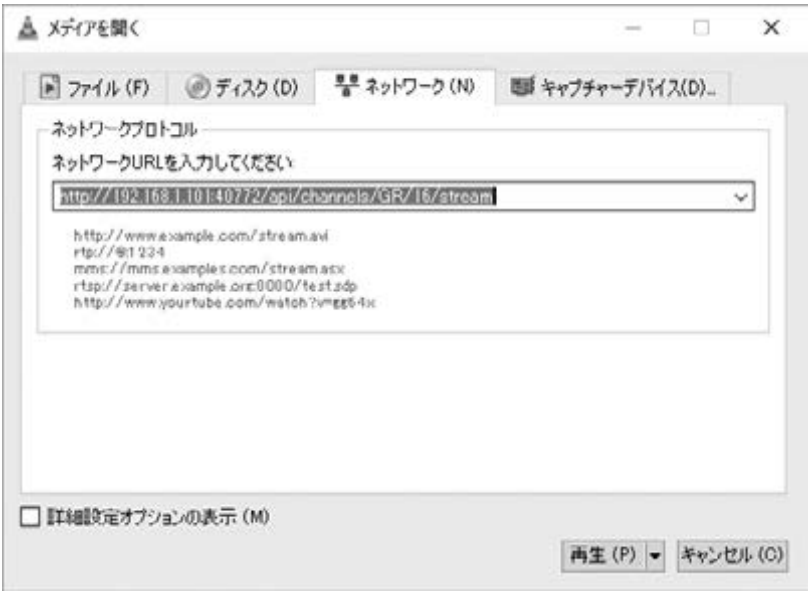
チャンネルスキャンが完了すると、サービス名が表示されるようになります。



←2-3-5

2.4 VLC で使う

VLC はネットワークストリームを開くことができます。



←2-4-1

当然 Mirakurun のストリームも再生できます。



2-4-2

Linux の場合コマンドラインから立ち上げることも可能です。

```
raika@kurumi:~$ vlc http://192.168.1.101:40772/api/channels/GR/16/stream
```

2.5 VLC for iOS で使う

VLC for iOS もまたネットワークストリームを開くことができます。



2-5-1



2-5-2

2.6 外出先から使う

⚠ Mirakurun は家庭内 LAN 専用サーバーです。したがって一切の認証機能は実装されていませんので、Mirakurun を**絶対にインターネット上に曝(さら)してはいけません**。

⚠ また、プライベート IPv4/IPv6 アドレスによる通信のみにあらかじめ制限されています。この制限を回避するために Mirakurun のソースコードを書き換えたりしないでください。

とはいえ、外出先から Mirakurun へアクセスしたくなるケースは時折ありますので、参考程度に二つ方法を紹介します。

❇ SoftEther VPN を使う



← 2-6-1

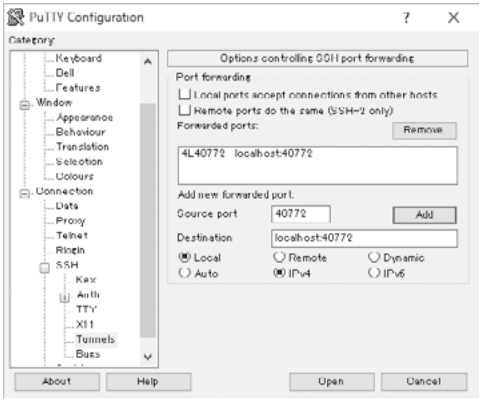
SoftEther VPN は、非常に強力で簡単な国産オープンソース VPN ソフトウェアです。

これを用いると外出先から自宅の LAN に接続することができ、自宅に居る時と変わらない感覚で Mirakurun へアクセスすることができます。

ℹ プロジェクトサイトはこちら ☞ https://ja.softether.org/

❇ SSH トンネルを掘る

SSH で Mirakurun が動作する計算機に接続できる場合、SSH ポートフォワード機能を使ってトンネルを掘ることができます。接続の度に手間がかかるのが難点ですが、安全に Mirakurun にアクセスすることができます。



← 2-6-2

第 3 章

上級編

Mirakurun に手を出す時点で上級者

3.1 高度な設定

Mirakurun のサーバー設定にはいくつか高度なオプションがありますので、必要に応じて設定できます。

```
root@kurumi:~# mirakurun config server
```

❄ ログレベル

ℹ プロパティ … **`logLevel`**
ℹ デフォルト … **`2`**

`-1` 〜 **`3`** の範囲でログレベルを変更できます。値が大きいほどログの量が多くなります。

各値が持つ意味は次の通りです。

- **`-1`** … FATAL（致命的なエラー）
- **`0`** … ERROR（エラー）
- **`1`** … WARN（警告）
- **`2`** … INFO（情報）
- **`3`** … DEBUG（デバッグ用）

❄ UNIX Domain Socket ファイルパス ⚠ Linux

ℹ プロパティ … **`path`**
ℹ デフォルト … **`/var/run/mirakurun.sock`**

❇ 名前付きパイプ名 ⚠ Windows

ℹ プロパティ … **path**

ℹ デフォルト … **\\.\pipe\mirakurun**

❇ TCP ポート

ℹ プロパティ … **port**

ℹ デフォルト … **40772**（覚え方：よぉ、ちなつ。）

❇ IPv6 無効化

ℹ プロパティ … **disableIPv6**

ℹ デフォルト … **false**

IPv6 での listen を無効化する場合に設定します。デフォルトではネットワークインターフェースに設定されているリンクローカルアドレスと、ループバックアドレスが listen されます。

❇ バッファ最大許容量（ハイウォーターマーク）

ℹ プロパティ … **highWaterMark**

ℹ デフォルト … **25165824**（Byte）※随時最適値に更新されます

⚠ この値をむやみに設定すると動作が不安定になるおそれがあります。

ストリームあたりのバッファ最大許容量により、Mirakurun プロセスによるメモリの過剰消費を防止しています。VPN 等を利用した際に接続が切れてしまう場合はこの値を増やしてみてください。ただし、増やしても接続が切れてしまう場合はスループット不足が考えられます。

ハイウォーターマークとは、川などにある高水位標の事です。

❇ バッファオーバーフロー許容時間

ℹ プロパティ … **overflowTimeLimit**

ℹ デフォルト … **30000**（ミリ秒）※随時最適値に更新されます

⚠ この値をむやみに設定すると動作が不安定になるおそれがあります。

前項のバッファ最大許容量で設定された値を超えた状態の許容時間を設定できます。この時間内で一時的なバッファ溢れが解決することを期待します。

❇ イベント情報ガベージコレクション間隔

ℹ プロパティ … **`programGCInterval`**

ℹ デフォルト … **`900000`**（ミリ秒）※随時最適値に更新されます

⚠ この値をむやみに設定すると動作が不安定になるおそれがあります。

❇ EPG 採集間隔

ℹ プロパティ … **`epgGatheringInterval`**

ℹ デフォルト … **`900000`**（ミリ秒）※随時最適値に更新されます

⚠ この値をむやみに設定すると動作が不安定になるおそれがあります。

Mirakurun は EPG データを最新に保つため常に EPG 採集を行っています。この値を変更することで全てのネットワークの EPG を更新した後の休止時間を調整することができます。

❇ EPG 最大取得時間

ℹ プロパティ … **`epgRetrievalTime`**

ℹ デフォルト … **`600000`**（ミリ秒）※随時最適値に更新されます

⚠ この値をむやみに設定すると動作が不安定になるおそれがあります。

EPG 採集時におけるネットワークあたりの EPG 最大取得時間です。EPG 取得時間はこの時間を超えることはありません。通常は 2 ～ 3 分で取得が終わるようです。

3.2 Open API / REST API

Mirakurun のインターフェースは Open API ベースの REST API です。Open API とは、REST API を設計定義するための仕様です。

ℹ Open API ☞ https://openapis.org/

API ルートは次のようになり、全 API リソースは **/api/** から始まります。

```
http://host:port/api/
```

Open API に準拠した API ドキュメントは次のような URL で取得できます。

```
http://host:port/api/docs
```

❇ ステータスコード

リクエスト成功時は **200** や **202**、エラー発生時は **404** や **503** などのように、API はリクエストの結果を HTTP ステータスコードでクライアントに示します。

❇ Open API に対応したクライアントライブラリを利用する

たとえば Chinachu[beta] の Mirakurun に対応した EPG 取得は **swagger-client** というライブラリを使って実装されています。こういった Open API に対応したクライアントライブラリを活用すると、非常に簡単に API クライアントが開発できます。

❇ プライオリティを指定する

プライオリティを指定するには、リクエストヘッダ **X-Mirakurun-Priority** を設定します。

❇ 例：番組（イベント）指定ストリーム API

Mirakurun には他に類を見ないとても便利な API があります。それが番組指定ストリームです。指定した番組だけを録画したい時、この API を使うとぴったり録画できます※。突然番組時間が延長されても問題ありません。

```
/api/programs/{id}/stream
```

Chinachu Air ではこの API を活用することによって正確な予約録画を期待します。

※放送局が正常に現在の番組情報を送出している場合に限ります。これは通常の EPG とは異なります。

3.3 Munin で監視する

Munin とはオープンソースのリソースモニタリングシステムです。リソースの使用状況の推移をグラフで直感的に捉えることができます。Mirakurun は Munin のプラグインも提供しています。これを活用すれば、トラブルシューティングの助けになるでしょう。

❶ Munin ☞ http://munin-monitoring.org/

プラグインの動作には Mirakurun に加えて Munin `>= 1.4.0` が必要です。

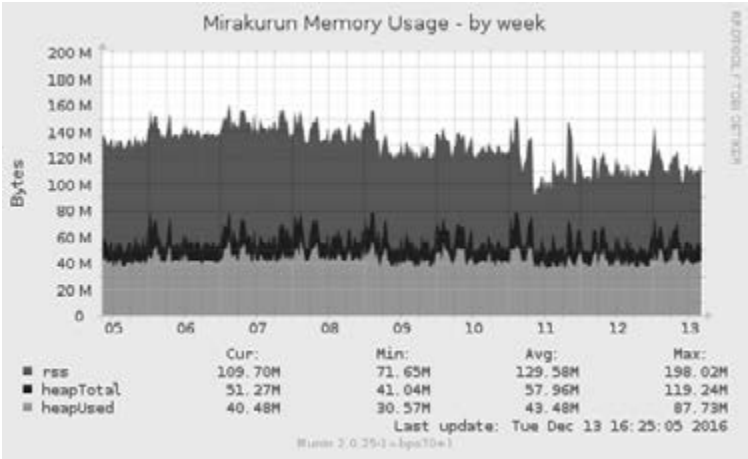
✲ インストール

```
ln -s /usr/local/lib/node_modules/mirakurun/bin/munin-plugins/mirakurun_status.js
      /usr/share/munin/plugins/mirakurun_status
ln -s /usr/share/munin/plugins/mirakurun_status /etc/munin/plugins/mirakurun_status
# check
munin-run mirakurun_status
# apply
service munin-node restart
```

シンボリックリンクを貼ったあと正常に動作するか確認して Munin を再起動すれば OK です。

✲ グラフ：Mirakurun Memory Usage

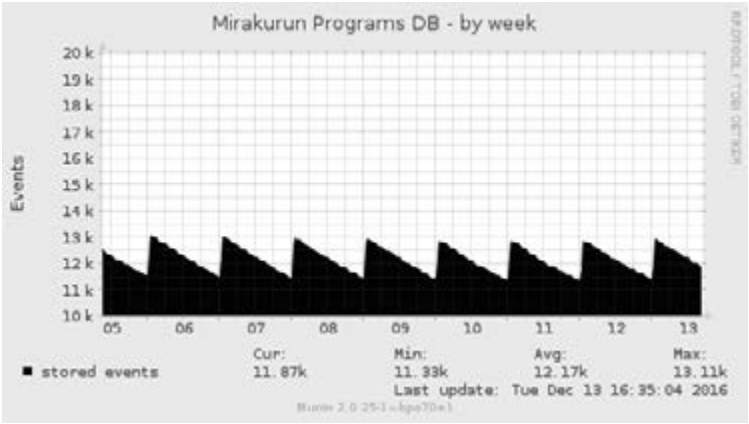
Mirakurun のメモリ使用量です。際限なく右肩上がりになっている場合は大問題ですので、ただちにフィードバックしてください。



3-3-1➡

❇ グラフ：Mirakurun Programs DB

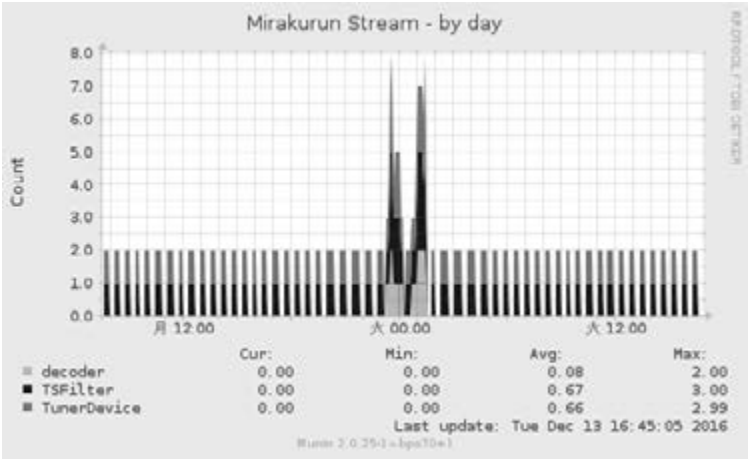
Mirakurun が保持しているイベント (EPG) の数です。下の画像のように毎日 0 時過ぎに追加されているのが分かります。



←3-3-2

❇ グラフ：Mirakurun Stream

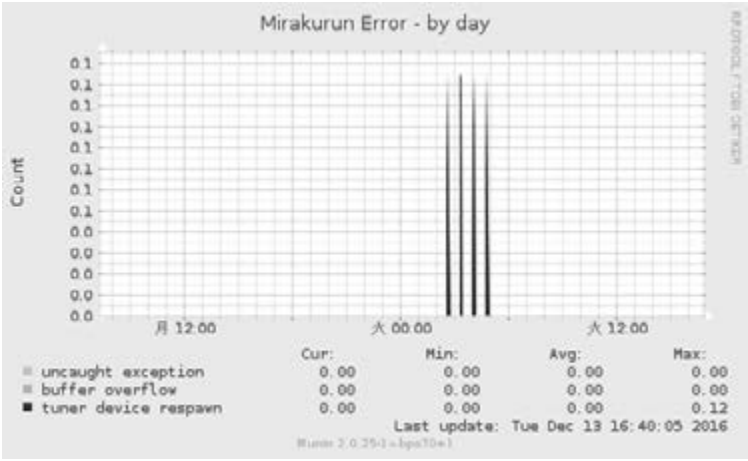
Mirakurun が処理している主なストリームの数です。



←3-3-4

✲ グラフ：Mirakurun Error

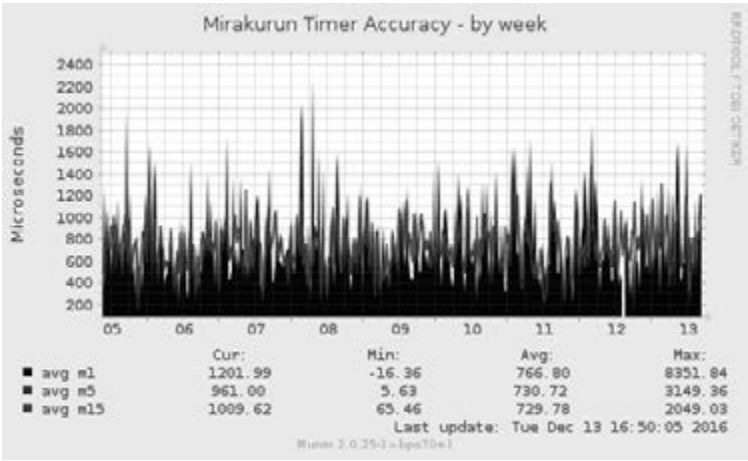
Mirakurun のエラーカウンターです。通常は下の画像のように tuner device respawn エラーが放送休止時間帯にのみ発生します。



3-3-3➡

✲ グラフ：Mirakurun Timer Accuracy

Mirakurun の内部タイマー精度です。サーバーが過負荷になっている場合などは精度が落ちる場合があります。通常は特に気にしなくてもよいでしょう。



3-3-5➡

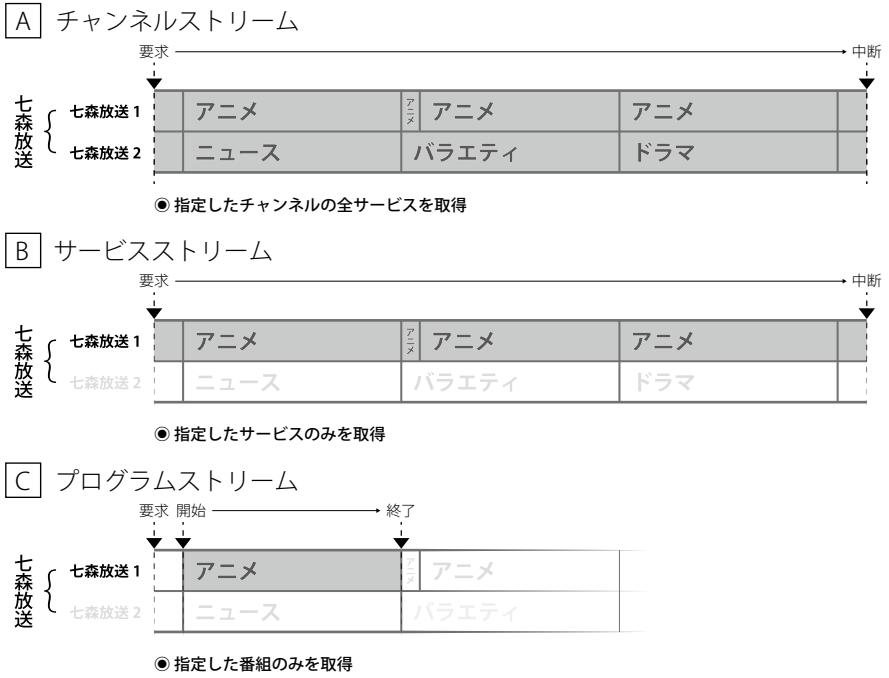
第 4 章

開発編

Mirakurun の開発に足を突っ込もう

4.1 アーキテクチャ

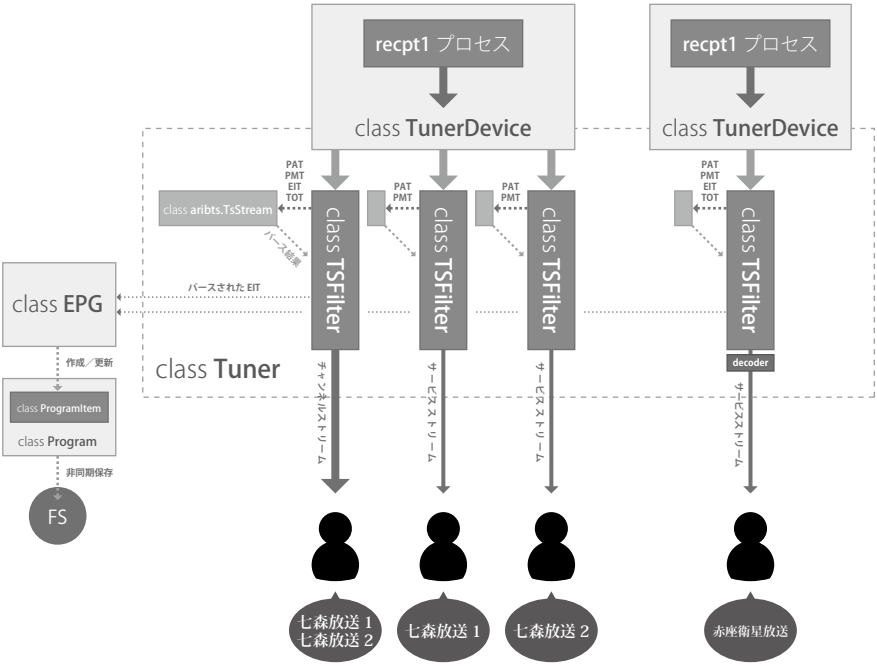
Mirakurun は Node.js 実行環境で動作します。大部分は TypeScript(ES6) で書かれています。Web アプリケーションフレームワークとして **Express**、Open API フレームワークとして **`express-openapi`** を使用しています。EPG パーサーには **`node-aribts`** を使用しています。



←4-1-1

❄ ストリームの流れ

Mirakurun はチューナー資源とそのストリームを最大限に活用するために設計されています。まだまだ改善の余地があると考え、我々は常に無駄のないストリームフローを検討しています。妙案を思いついた方は Slack の #air チャンネルへいつでもお越しください。



4-1-2 ➡

4.2 ビルドとデバッグ

Mirakurun への貢献方法についての詳細はこちら ☞ https://github.com/kanreisa/Mirakurun/blob/master/CONTRIBUTING.md をご覧ください。

❇ ソースの取得

```
git clone git@github.com:kanreisa/Mirakurun.git
cd Mirakurun
```

❇ ビルド

```
npm install
npm run typings-install
npm run build
```

❇ インストール

```
# Linux / Darwin
raika@kurumi:~$ sudo npm install pm2 -g
raika@kurumi:~$ sudo npm install . -g --unsafe --production

# Win32 (Windows PowerShell as Admin)
PS> npm install winser -g
PS> npm install . -g --production
```

❇ デバッグ w/ node-inspector

```
# Linux / Darwin
raika@kurumi:~$ sudo mirakurun stop
raika@kurumi:~$ sudo npm run debug

# Win32 (Windows PowerShell as Admin)
PS> Stop-Service mirakurun
PS> npm run debug.win32
```

4.3 Change Log

Technical Preview から Beta、RC 版までの主な変更履歴です。

❇ Beta

❶ 2016 年 9 月 14 日リリース

- Service Logo API（局ロゴ）の実装
- Events API 実装
- Version API 実装
- Version Update API 実装
- Status API 内容拡充
- munin-plugin 実装
- バッファオーバーフロー許容時間
- クライアントライブラリ実装
- 各種バグ修正
- パフォーマンス向上

❇ RC

❶ 2016 年 12 月 11 日リリース

- OpenRC サポート修正
- Status API 内容拡充
- IPv6 サポート実装
- PT2 サポート削除
- 軽微なバグ修正

❇ 貢献者一覧　2015 年 4 月 5 日 — 2016 年 12 月 14 日

- kanreisa (304 commits)
- miyukki (15 commits)
- rndomhack (3 commits) + aribts
- atrandom (1 commit)
- akimasa (1 commit)

終

PT2 サポート情報

文 ● akimasa

お急ぎの方は、**原因 (P.40)** からお読みください。

はじめまして

初めて記事を書く akimasa です。前回のちな本で h-mineta さんの BonDriver_Mirakurun を fork した自分のリポジトリが取り上げられていて、本を購入した後大変驚きました。まさか、2ch でダメ出しされていた、あの手抜きのコードが紹介されるとは…。結局、あの後、upstream の方でチャンネル情報を Mirakurun API から持ってくる機能は実装されました。自分に必要な Magic Packet を送る機能も Pull Request を送らせていただき、採用されました。Windows で TVTest を使って Mirakurun のチューナーを使いたい方は P.21 の BonDriver_Mirakurun を使うと便利です。今ならバイナリがあるので、自分でコンパイルする必要もありません。

Mirakurun サーバーマシン

うちには Mirakurun サーバーマシンがありますが、これには PT2 と PT3 が刺さっています。PT3 はホストの Arch Linux でそのまま Mirakurun や Chinachu を起動し、PT2 は Docker コンテナ内の Mirakurun で使っています。コンテナ内で Chinachu も走らせています。

なぜ PT2 は Docker コンテナ内で使っているかというと、PT2 と PT3 を使う人を分けたいからです。アニメばっかり溜まっている Chinachu を家族と共有して使っていると、以下の問題が懸念されます。

- 間違えて大事なアニメが消されてしまう恐れがある
- 録画済の大量のアニメの中から家族が録画した番組を探し出すのは大変

あと、なんとなく PT2 を信頼していないというのもあります。大事なアニメの録画に失敗したら困りますので、信頼できる PT3 を使って録画したかったのです。

それと、recpt1 の仕様を勘違いしたというのもあります。PT1/2 と PT3 ではドライバが違うので、recpt1 も PT1/2 と PT3 で互換性が無いと勘違いしていました。

ホストで Mirakurun を走らせて、PT2 と PT3 をまとめて扱えばいいでは無いかというツッコミが来そうですが、上記の理由で分けることにしています。

チューナーが一時的に死す

一時的に死にますが、サスペンドからリジュームすると勝手に復活するので、特に困ってはいません。

PT2 を信頼しないのは正解でした。Docker コンテナで PT2 を利用している Mirakurun を実行しているのですが、稀に PT2 のチューナーが全部死にます。ログを確認してみたら、週に 2、3 回は PT2 が死んでいるようでした。ただし、放送波が来ていないのか、本当に全部死んでいるのか、今となっては定かではありません。PT2 が死ぬと、大量に `Cannot tune to the specified channel: /dev/pt1video3` みたいなログが Mirakurun のログに残ります。

サスペンド

録画が行われていない等、一定条件を満たすと節電のため Mirakurun サーバーマシンをサスペンドさせています。ただ、pt3_drv がロードされている状態でサスペンドすると、PT3 が使えなくなります。http://haruo31.underthetree.jp/2013/08/29/ にあ

るスクリプトを /etc/pm/sleep.d/80pt3drv に置いて回避しています。

```
#!/bin/sh

PATH=/sbin:/usr/sbin:/bin:/usr/bin

case "${1}" in
  hibernate|suspend)
    if lsmod | grep pt3_drv > /dev/null 2>&1; then
      rmmod pt3_drv
    fi
    ;;
  resume|thaw)
    modprobe pt3_drv
    ;;
esac
```

抜粋するとこんな感じです。先ほど挙げた URL に詳しく載っていますが、`chmod a+x 80pt3drv /etc/pm/sleep.d/80pt3drv` で実行権限を付け忘れないように気をつけましょう。

特に困っていなかった

上記では pt3_drv だけアンロードしていますが、自分の環境では pt1_drv もアンロードするようにしています。実は、PT2 が一時的に死んでも、`rmmod pt1_drv; modprobe pt1_drv` で復活します。節電のためにサスペンドするたびに、pt1_drv がアンロードされるので、勝手に復活します。普段はサスペンドしていて、rtcwake で録画が始まる数分前にリジュームするようにしているので、録画が失敗することは滅多にありません。PT2 が死んでいて家族から苦情が来るのは、長時間付けっぱなしにしていた時です。長時間かかる重い処理のために自動的にサスペンドに入らないようにしない限り、長時間付けっぱなしだということはありません。限られた条件で、しかも滅多に起こらないので、それほど困っていませんでした。

▨ 調べてみた

デバイスファイルが同時のタイミングで閉じられると死ぬみたいです。また、起動と終了（デバイスファイルの open と close）がうまいことかぶっても、やはりデバイスファイルが死にます。

最初は、コマンドの起動間隔が狭いからだと思っていました。実際、Chinachu の Wiki にも PT2 の場合は operRecCmdSpan を 1000 以上に設定するように書かれています。しかし、いくら連続で recpt1 を実行しても、全く現象が再現しません。最後にわかるのですが、どうやら起動間隔というよりも終了のタイミングの方が肝心なようです。

シェルスクリプトで叩いてみる

recpt1 を普通に実行しているのでは実行間隔が広すぎるのかと思い、次のようなシェルスクリプトを書きました。

```
for i in {0..1000}
do
  recpt1 --device /dev/pt1video2 22 1 - >/dev/null &
  pid=$!
  sleep 0.05
  kill -TERM $pid
  wait $pid
done
```

上記スクリプトと、`for i in {0..100};do (recpt1 --device /dev/pt1video3 26 0 - >/dev/null) ;done` を実行してみたところ、しばらくすると見事に recpt1 がエラーを吐くようになり、PT2 が死にました。

あるいは、上記スクリプトの --device を /dev/pt1video2 にしたのと、/dev/pt1video3 にしたのを同時に実行させ、あとから recpt1 を実行してみると、PT2 が死んでいることが確認できます。

この時点では、実行間隔が相当狭くないと現象が再現しないのかと思っていましたが、後に**これは誤り**だと判明しました。

バイナリで叩いてみる

recpt1 のソースコードを眺めてみました。詳しくないので、デバイスファイル (/dev/pt1video2 とかの事) を開いて、ioctl で選局やコマンドを送っているなというくらいしかわかりませんでした。それはともかく、チャンネル合わせが成功するかどうかで、デバイスファイルが生きているかを調べられるみたいです。このことを踏まえて、10 回ほど連続で何かをして、最後にチャンネル合わせができるかどうかを確認するということを繰り返していました。シェルスクリプトから ioctl を使う方法があるかどうかよくわからなかったので、今度は C 言語のソースをコンパイルして実行していました。

いくつかパターンを確認したところ、デバイスファイルの開閉を行うパターンで一番デバイスファイルが死にやすかったので、さらに詳しく確かめることにしました。

スレッドやセマフォを使って、同じタイミングでデバイスファイルを開き、同じタイミングで閉じると、たまにデバイスファイルが死ぬことが確認できました。今度は開くタイミングと閉じるタイミングの一方か両方をずらしてみました。そしたら、次のようなことがわかりました。

- 開くタイミングはずらさなくても大丈夫
- 閉じるタイミングだけずらせば大丈夫
- 閉じるタイミングは 10ms 以上ずらす必要がある
- 1ms ずらすのでは 1000 回弱、連続で叩くとデバイスファイルが死ぬ
- 全くずらさない場合、100 ～ 500 回くらい叩けば死ぬ

デバイスファイルを開くのにも結構時間がかかるので 10ms ずらすのは 5000 回しか試していません。

usleep のどちらか両方コメントアウトして実行してみると、どういう条件でデバイスファイルが死ぬのか確認できるかと思います。 https://gist.github.com/akimasa/2f9279bd8e3d919b5b9f567d2c4ebd64 にソースコードを置いているので、実行したい方は、こちらから落としてください。

Mirakurun の対応中に他の死ぬ事例が見つかった

さて、SIGTERM 送信のタイミングを `2.0.0-beta.1` でずらすように修正していただきました。しかし、しばらくすると、やはりデバイスファイルが死にました。結局のところ、これも終了がかぶってしまったのが原因です。

夜中に放送されていない時間帯がありますが、2 つの放送されていないチャンネルに同時に合わせようとすると、やはりデバイスファイルが死にます。次のようなスクリプトを実行する事で確認ができます。18ch は放送波が来ていないチャンネルだとします。

```
for i in {0..10}
do
  echo $i
  recpt1 --device /dev/pt1video2 18 0 - >/dev/null &
  pid=$!
  recpt1 --device /dev/pt1video3 18 0 - >/dev/null
  wait $pid
done
```

Mirakurun のログを見る限りでは、多分、上記のスクリプトみたいな現象が起こっていたと思います。放送波が来ていないチャンネルの録画を 2 つ同時に起動すると、ダメみたいです。放送が行われているチャンネルだと、recpt1 の終了がばらつきます。逆に、放送が行われていないチャンネルだと実行時間のばらつきが少なく、一定しているみたいです。そのため、同時に起動すると、同時に終了する、つまり同時にデバイスファイルが閉じられるので、デバイスファイルが死にます。

閉じるのと開くのがかぶってもだめか？

`2.0.0-beta.3` で起動するタイミングも、ずらすようにしていただきました。そのため、終了のタイミングがずれるはずですが、やっぱり稀にデバイスファイルが死にました。ログを眺めてみたら、SIGTERM を送るのとほぼ同時に起動させていたみたいです。おそらく、ドライバで次のような感じで２つ同時にスリープモードに入れようとしてしまっていたと思われます。

pt1video2	pt1video3
close(){	open(){
lock	
何か色々 ...	
unlock	
	lock
	何か色々 ...
＼ **setsleepmode** ／	＼ **setsleepmode** ／
wait	

Pull Request を送った

Pull Request を送り、`2.0.0-beta.6` でリリースされました。今度は、SIGTERM 送信と起動で、ちゃんと間隔をあけて行うようにしてみました。そして、起動後、約 4 秒は SIGTERM の送信も新たに起動も行わないようにしました。これは、先ほどの放送波の来ていないチャンネルが存在して、一定時間で勝手に終了する対策です。

わかりづらいと思うので、詳しく説明します。放送波の来ていないチャンネルに合わせると、勝手に終了します。そのため、recpt1 が終了するタイミングが制御できません。通常、勝手に終了するのに 3.5 秒かかるみたいです。なので、少し多めに 4 秒は新たに起動も終了もしないようにしています。これが 1 秒とかだと、1 秒経った後は、いつでも SIGTERM が送信される可能性があります。さらに起動から約 3.5 秒後に recpt1 が勝手に終了します。勝手に終了するタイミングに偶然、チューナーが不要になって SIGTERM を送信すると、終了がかぶり、デバイスファイルが死にます。

効果があるか確認した

結論からいうと、**ダメでした**。

Mirakurun から使う限りでは、次のいずれかを満たすとデバイスファイルが死にました。

- 放送波の来ていないチャンネルが存在する
- チャンネルを切り替えまくる

そのため、Mirakurun の対策が有効であるかどうかは、次のような方法で行いました

- 放送波の来ていないダミーのチャンネルを追加する
- ランダムな録画時間、ランダムな実行間隔、ランダムなチャンネルでストリームを取得するスクリプトを走らせる

ダミーのチャンネルを足すだけで、問題が起こるときは、割とすぐに起こるようになりました。約 1 日で現象が再現したと思います。スクリプトはダメ押しです。スクリプトを実行させるようにしたのは、現象がほぼ再現しなくなった頃なので、どれくらい現象の再現の誘因になるのか不明です。

放送波の来ていないチャンネルに合わせた場合、終了するタイミングが制御できないので、どうしても終了のタイミングが被ります。Mirakurun の対策で概ねデバイスファイルが死ぬ不具合は回避できますが、**確実ではありません**。

まとめ

以下のような条件を満たすと、デバイスファイルが死にました。

- 終了がかぶる
  - o 同じタイミングで SIGTERM を送る
  - o 同じタイミングで放送波の来ていないチャンネルを録画する
- 起動と終了がかぶる
  - o 同じタイミングで起動と SIGTERM の送信が行われる

起動と終了は、正確にはデバイスファイルの open と close です。

▨ 原因

pt1_drv のバグ（※この記事は chardev 版のドライバを対象に書かれています。Linux に標準で入っている DVB 版ドライバ earth-pt1 ではどうなのか不明です。）

▨ 対策

いくつか考えられます。

・Mirakurun

・recpt1

・pt1_drv

上記のいずれかで対策すれば良さそうですが、pt1_drv のバグを潰すのだけが有効でした。他の Linux の録画ソフトを使っていて、不具合が起こる方にも有効です。

Mirakurun

Mirakurun でも対策が可能だと思い、試行錯誤していました。しかし、対策を実施したところで不具合が起こる頻度を落とす事が出来ても、確実になくすことは出来ませんでした。**（RC 版で PT2 サポートは削除されました。）**

recpt1

recpt1 でも対策が可能かもしれません。何らかの方法で、同じ PT2 を使うときに、同じタイミングで閉じないようにすれば良いと思います。ただ、試していないので本当に大丈夫かどうかは不明です。強制終了されると、勝手にデバイスファイルが閉じられるので、強制終了が原因でデバイスファイルが使い物にならなくなるかもしれません。ドライバを修正しましょう。

pt1_drv

PT2 のドライバで対策ができます。チューナースリープ時にロックされていないのが原因っぽいです。同じタイミングでスリープ処理が行われるからまずいのではないかと思います。デバイスファイルを閉じるタイミングが 10ms 以上ずれると現象が再現しなくなったので、スリープ処理は 10ms 未満で終わっているものだと思われます。

```
--- a/driver/pt1_pci.c
+++ b/driver/pt1_pci.c
@@ -396,13 +397,14 @@ static int pt1_
release(struct inode *inode, struct file *file)
                channel->req_dma = FALSE ;
                wake_up(&channel->ptr->dma_
wait_q);
        }
-       mutex_unlock(&channel->ptr->lock);

        /* send tuner to sleep */
        set_sleepmode(channel->ptr->regs,
&channel->lock,
                                  channel-
>address, channel->type, TYPE_SLEEP);
        schedule_timeout_interruptible(msecs_
to_jiffies(100));

+       mutex_unlock(&channel->ptr->lock);
+
        return 0;
 }
```

上記のパッチを適用して、あとは普通にインストールすれば良いです。インストールが終わったら、**rmmod pt1_drv; modprobe pt1_drv** を実行してパッチが適用されたモジュールを読み込ませます。Mirakurun が止まっている状態じゃないと、うまくいかないかもしれません。可能なら、システムごと再起動するのが一番確実だと思います。

http://hg.honeyplanet.jp/pt1/rev/8a2a032bfa50 のパッチを適用しても解決します。（右頁）

```
--- a/driver/pt1_pci.c  Wed Mar 03 03:54:05 2010 +0900
+++ b/driver/pt1_pci.c  Sat Mar 06 19:13:17 2010 +0900
@@ -343,13 +343,13 @@
                        mutex_unlock(&device[lp]->lock);
                        return -EIO ;
                    }
-
+#if 0
                    /* wake tuner up */
                    set_sleepmode(channel->ptr->regs, &channel->lock,
                                  channel->address, channel->type,
                                  TYPE_WAKEUP);
                    schedule_timeout_interruptible(msecs_to_jiffies(50));
-
+#endif
                    channel->drop  = 0 ;
                    channel->valid = TRUE ;
                    channel->overflow = 0 ;
@@ -389,10 +389,12 @@
     }
     mutex_unlock(&channel->ptr->lock);

+#if 0
     /* send tuner to sleep */
     set_sleepmode(channel->ptr->regs, &channel->lock,
                   channel->address, channel->type, TYPE_SLEEP);
     schedule_timeout_interruptible(msecs_to_jiffies(50));
+#endif

     return 0;
 }
@@ -755,7 +757,7 @@
     // 初期化完了
     for(lp = 0 ; lp < MAX_CHANNEL ; lp++){
         set_sleepmode(dev_conf->regs, &dev_conf->lock,
-                      i2c_address[lp], channeltype[lp], TYPE_SLEEP);
+                      i2c_address[lp], channeltype[lp], TYPE_WAKEUP);

         schedule_timeout_interruptible(msecs_to_jiffies(50));
     }
```

ただ、こちらはスリープモードに入らないので、もしかすると何か不都合があるかもしれません。 自分が試してみて気づいた事といえば、デバイスファイルの開閉が非常に速くなった事でしょうか。

謝辞

お忙しい中、Mirakurun の PT2 サポートに関してアドバイスをしていただいた kanreisa さんに感謝します。

h-mineta さん、Magic Packet 送信機能の Pull Request の採用ありがとうございます。

また、お二人とも、私の拙い Pull Request の問題点の指摘や手直しをして頂きまして本当にありがとうございます。

終

編集者コメント「PT2 をお使いの方はぜひご確認ください。」

話題の番組をコメント付きで見よう
Chinachuと作る全録視聴環境

MiyouTV with Chinachu

文 ● takaaki（未来検索ブラジル）

開発を振り返りながら、MiyouTVについて紹介していきます。

MiyouTVとは

テレビ番組を全録して、2ちゃんねるの実況コメント付きで視聴するシステムです。テレビをもっと楽しく見ることを目指して開発されました。MiyouTVは番組を再生するフロントエンドと全録をサポートするエージェントの2つのソフトウェアで構成されています。エージェント(MiyouTV Agent)がChinachuに時間単位での予約データを入力して、EPGに依存しない全録を可能とします。また、Mirakurunから番組表を収集し、古い番組の削除やスケジューラーの実行を行います。もちろん、エージェントを導入しなくても、Chinachuの番組再生フロントエンドとしての利用が可能です。全録が必要ないと思っている人も気軽に試せます。コメント付きのテレビ番組を一度観ると、コメント無しで観るのが物足りなくなります。この感覚は不思議で、きっと欠かせないものになります。ぜひ体験してみてください。



MiyouTV

MiyouTVにおける全録の意味

MiyouTVでは指定した放送局の番組をとにかく全部録画します。正確に言えば番組単位でさえなく、機械的に時間毎のTSの塊を管理しています。こうすることで特定のジャンルや出演者のみをチェックするためには番組の「予約」が必要ですが、全部録ってしまうことでその手間を省けるというメリットがあります。ただ、TS全録は大きなストレージ容量が必要になります。しかし、最近の大容量HDDの普及度、価格であれば十分に可能です。次に、全部のTSを録ってあれば、その放送局の一定期間内のあらゆる番組を参照できるようになります。あらかじめ観たい番組だけでなく、面白いハプニングがあったとか、SNSで何度も話されているエピソードなどを後追いでチェックできるというのは、全録を実際にやってみないとなかなか実感できないメリットです。三つ目はコメントを付けることでコメントがツッコミとして働き、あまり興味の無かった番組が面白く観られるようになります。ニコニコ動画と同じ、緩い共時性というものです。細かい事としては、番組表データの更新が遅れ、かつ番組時間の変更があった場合の録画漏れの心配とは無縁になる、といったメリットもあります。

大きく言えば、MiyouTVは全録によって、テレビの不自由な部分を減らし、テレビの楽しみ方が大きく拡がることを目指しています。

バックエンドに Chinachu を使った経緯

MiyouTV のバックエンドは Chinachu です。プロジェクトの初期は MythTV の利用を想定していました。MythTV は高機能で、EPG に依存しない予約や柔軟な録画番組の管理、EPG のリアルタイム取得など、全録に必要な要素をすべて備えていました。しかし、国内のデジタル放送に対応していないことが問題でした。ツールの利用とプログラムの改造で対応は可能ですが、改造は容易ではありません。また、対応したチューナーが少なく、安定性に不安が残るものでした。更に、MythTV フロントエンドのプラグイン機能ではコメント再生は実現できません。

そこで、MythTV はきっぱり諦め、専用フロントエンドアプリの開発とバックエンドシステムの変更を検討しました。日本国内の既存の録画予約システムを利用することで、プログラムに対する改造などの特別な対応の必要が根本的になくなります。その中でも、バックエンドに適した WebAPI が用意されている Chinachu が候補に挙がりました。ただし、MiyouTV のバックエンドに採用するにあたって、Chinachu は EPG による予約で、時間指定での録画に対応していないことが問題でした。これに対しては、外部プログラムから擬似的な予約データを Chinachu に対して直接書き込むことで問題が解決しました。こうして Chinachu をバックエンドにできる目処がたち、開発が始まりました。

全録を支える MiyouTV Agent

上に書いたように、Chinachu の機能だけでは全録システムは実現できません。一定周期の予約情報の登録や古い番組の削除を外部から実行する必要があります。更に番組表の収集や録画中に自動実行されないスケジューラーの実行を誰かが担う必要があります。その誰か、にあたるプログラムが MiyouTV Agent (miyoutv-agent) です。

また、MiyouTV Agent は Mirakurun を利用します。以前は録画したファイルを元にスケジューラーを実行し、Chinachu 内の番組表を収集していましたが、それによって得られる番組表は不完全なものでした。現在はその部分を Mirakurun に担わせ、MiyouTV の必須プログラムのひとつにすることによって、負荷をかけずに番組表収集が安定して行えるようになっています。そして、MiyouTV Agent は Mirakurun が導入されている環境であれば、特別なことをしなくてもインストールできます。

このように、MiyouTV の全録機能は Chinachu + Mirakurun と MiyouTV Agent の連携によって実現しています。



← 番組表

MiyouTV のフロントエンド

全録したテレビ放送と実況コメントを結び付けるものが、MiyouTV のフロントエンドアプリです。フロントエンドアプリを起動すると番組表が表示されます。コメント数が番組セルの上方に表示されるため、盛り上がっている番組が一目でわかります。番組表データは MiyouTV Agent が収集したものを利用しています。録画一覧では録画した番組が一覧できます。こちらは MiyouTV Agent を導入していない環境でも利用可能です。どちらの画面でもコメント数とサムネイルが表示されます。サムネイルはバックエンドへの負荷を考慮し、1 つずつ読み込む仕様です。プレイヤーは VLC がベースになっています。番組を再生するとニコニコ動画のように右から左へコメントが流れます。コメントのアニメーションは CSS3 の Transition によりスムーズです。サイドバーではコメント一覧や番組情報、再生設定にアクセスできます。コメントやサイドバーの非表示化も可能です。また、コメント一覧からはスレッドの絞り込みが可能です。

フロントエンドは最初 Qt[2] で開発されていましたが、今後の開発の利便性、技術トレンドから、Web ベースの技術を最大限活かすことにしました。そして、libVLC ラッパーの WebChimera.js によって JavaScript で動画をまともに扱え、また、ソースプロテクトに対応していたこともあって、HTML アプリへの移行時に NW.js を採用しました。しかし、MiyouTV のオープンソース化に伴い今後は Electron に移行する予定です。また、UI に使用しているフレームワークは AngularJS1 と Bootstrap3 です。少々バージョンが古くなっていますが、これらもアップグレードを計画しています。

2 ちゃんねるの実況コメントは専用の API を利用して取得しています。そして、コメントの取得には moritapo アカウントが必要です。アカウントは OAuth を使用して認証されます。

MiyouTV は Web ベースの技術によって、話題の番組を好きな時に見られる環境を実現しています。そのため、他の実行環境への移植が比較的容易に行え、Web の知識を生かした開発が可能になっています。

MiyouTV フロントエンドと上記 miyoutv-agent はこの URL でソースコードを参照できます。

https://github.com/search-future/miyou.tv

不具合の修正や機能追加など歓迎です。ぜひ pull-request をお願いします。



→エージェントを導入していない環境での録画一覧



←プレイヤー（サイドバー非表示）



→プレイヤー（サイドバー表示）

MiyouTV のこれから

MiyouTV プロジェクトでは導入に必要な各種情報の公開を積極的に行っています。また、導入を簡単にするツールや、必要な機材を揃えられる仕組みも検討中です。MiyouTV は誰にでも手軽に始められるシステムを目指しています。

MiyouTV はまだまだ開発途上です。新しい機能の追加や使い勝手の改善などを進めています。コメント機能では盛り上がりがすぐにわかる仕組みを考えています。コメント数のランキングや番組内の盛り上がり表示などの実装を予定しています。

MiyouTV Agent が未導入の環境でも全機能が動作するようにし、番組再生はより柔軟に行えるよう調整していきます。現在、フロントエンドがサポートしているのは Linux と Windows だけですが、他のプラットフォームへの対応も計画中です。加えて、他のコメントサービスへの対応も視野に入れています。また、MiyouTV は Chinachu Air へ対応予定です。Air では、より親和性の高いシステムを MiyouTV は目指します。

MiyouTV の最新情報は常に **http://miyou.tv/** に掲載されます。また、Twitter の **@miyou_tv** アカウントで随時様々なお知らせをしていきます。

終

UNKNOWN

所謂「聖地巡礼」と云うものについて。

世界を震撼させた一大テロ事件の発生という劇的な幕開けとなった21世紀も、早いもので16年目の節目を数える年となった。戦争の世紀と呼ばれた20世紀を生き抜いた人々がバラ色の夢と希望を抱いて迎え入れた21世紀だったはずなのに、今度はテロの世紀とも別名されるだけの、薄暗く不気味な世界が前世紀から引き続いたままの現状には世界中の誰もが落胆している事だろう。しかし、萌え産業の一大生産国であるここ日本※1においては相変わらず平和な様相を呈しており、人々はのんびりとアニメを見たり、萌え絵を愛でたり、オリンピック開催期とコミケとの関係を憂慮したりと、暴力と恐怖に支配される世界情勢から隔離されたかのような変わらぬ日常が送られているのであった。そんな中、最近ではアニメや漫画、小説などの創作作品に登場する実在の場所を訪れる「聖地巡礼」と呼ばれる余暇活動がここ日本において一大ブームとなっている。作品の中の登場人物がそこで生まれ育ち、生活し、あるいは活躍した舞台となった場所を、宗教における聖人ゆかりの場所を指し示すそれと同意の「聖地」と称し、ファンを信奉者に見立てそれらの土地を見て回ることを「巡礼」と呼び始めたのがこの言葉の由来である※2。昨今では主にアニメを中心に、モデルとなった実在の土地の風景が細部にわたって緻密に描写されていたり、熱狂的なファンの間で作中の要素から舞台が特定される間でもなく原作者や版権者が先回りして明言する場合も多くなり、より一層「作品の舞台」という要素を重視した作品が増えてきたように感じる。特に今年は「君の名は。」という劇場アニメ作品が爆発的にヒットし、舞台となった山間部の地方都市に多数のファンが殺到する様子が各メディアで取り上げられたのは記憶に新しい。さらにその作品による聖地の盛況っぷりにより、某社が主催する流行語大賞に「聖地巡礼」がノミネートされるまでにもなった。これまでマニア層にだけ通じる専門用語※3だった「聖地巡礼」という単語がだいぶ世間一般にも浸透してきた感がある。

ではなぜ人々は、自分が“ハマった”作品に描かれた光景を追い求めるのか。それは「聖地巡礼」の大きな魅力のひとつとして、ストーリーや登場人物は架空の創作作品であるにも関わらず、その物語の顛末が繰り広げられる舞台は実際に我々が生活している世界に存在し、そこに踏み入れることであたかもその作品の世界に入り込んだような不思議な感覚を味わえるという点にあるからだ。実際に「聖地」に降り立つと、作品内での場所の描写は実際に存在するものをベースとして描かれているだけに、完全にフィクションの領域内にいるはずの登場人物たちまでもがあたかもそこに実在するかのような錯覚に陥る。さらにじっくりと「聖地」を彷徨えば、そこが適当に舞台として選ばれただけのものではなく、設定された土地の光景や気候、住む人たちの人柄などの環境も、創造されたキャラクターたちの個性や行動に影響を及ぼす変数として作用している気さえしてくる。そして、彼らが見た光景、通った道、住んでた家を見るにつけ、映像や文章では読み取れない彼らの心中をより深く汲み取ることができ、一層作品への親しみが涌いてくるのだ。この作り手側による「架空世界」と「現実世界」を意図的に交錯させる構成手法により、アニメや小説などの世界が「現実世界から隔離され絶対に物理的に入り込むことのできない創造上のもの」では無くなり、現実世界からのアプローチによるフィクション世界への探訪が可能になるという驚くべき革新性がもたらされた。これまで完全に架空のもの、または（特に、あくまでフィクション作品であるが故に実在のものに触れる事が禁忌とされていた時勢では）ごく曖昧な描写のみだった登場人物たちの周辺環境を、実在の土地として物語内で包み隠さず存分にに表現することで、作品のファンたちにさらに作品に没入する楽しみを与える事にも成功したのだ。

さて、本稿が掲載される媒体からして「ゆるゆり」の舞台である富山県高岡市の話をしたいところだが……残念ながらそれはひとまず脇に置いておき、今回は私が実際に「聖地」に強い関わりを持っている、アニメを中心としたメディアミックス作品「ラブラ

※1　日本の萌え産業が及ぼす世界平和貢献についての考察は、連載※3「七森中録画研究会は日本のアニメを応援します☆」にて述べているのでご興味のある読者はぜひとも参照されたい。

※2　語源が宗教用語からの派生であることからもわかるとおり、これら聖地巡礼を行うファンの多くは作品に対する熱烈な支持者であり、その作品を絶対的に崇めるような狂信的な人々を嘲笑する意味を含む蔑称として使用される場合もあるので注意。

※3　筆者が初めて「聖地巡礼」なる言葉を聞いたのは、木崎湖が舞台の某作品のファンからだった気がする。

イブ！」と、その後継作である「ラブライブ！サンシャイン!!」の聖地について簡単に紹介したい。そもそも私は「ラブライブ！」の聖地となる秋葉原（外神田３丁目）に居住しており、自宅最寄りの神社は作中でも重要な場所として描かれる神田明神だし、歩いてほんの数分以内の場所の光景は作中で複数回に亘り登場している。まさにキャラクター達と同じ光景を毎日見て、同じ味の空気を吸って日々生活している状態である。いつもは何気なく通り過ぎていくこの風景や生活でも、たまにふとこのことを思い出し、作中内の彼らと日常を共有しているんだな、という感に浸るとたまらないものがある。さらに、今年アニメも放送されたシリーズ後継作「ラブライブ！サンシャイン!!」では、もともと魅力のある登場人物たちの中でもひときわ異彩を放つ某キャラクターにはまりすぎてしまい、舞台となる沼津にも１月のアニメ化発表前の探訪を皮切りにこれまで６～７回は訪れているし、作中の背景に描かれる旅館に宿泊までしている。さらに秋葉原から沼津までの回数券を常備し、暇を見つけては東海道線で２時間ほどかけて沼津まで移動する。ちなみにラブライブ！サンシャイン!!では、沼津と秋葉原を東海道線に乗車して往復する描写があり、住んでいるところや目指すところも聖地、移動中の東海道線の聖地と、もはや休日に沼津を往復しただけで聖地ばかり動き回っていることになる。とにかく、「聖地」を巡る事で、アニメのキャラクターたちに会えたり、彼らと同じ気持ちが通いてくるような気がするのだ。聖地に行けば、四季も、天候も、気温もさまざまだけど、作中の彼らだって暑ければ汗を流すし、寒ければ着込む、冬になれば寒い朝に憂鬱になるだろう。だから、その土地の肌感覚を共有しているんだと思うだけで嬉しくなるし、普通の観光地とは違った目で周囲を見渡すことができるのだ。

ところで最近では「聖地巡礼」に限らず、さまざまな手法で架空の２次元世界と、実際に存在する３次元世界を橋渡しする手法が多数開発されつつある。例えば先に出てきた「ラブライブ！」では、雑誌内での読者参加型企画としてスタートした点もあるからだろう、多角的なメディアミックスの一環として各キャラクターの声優陣によるライブベントが興行的にも成功している。他の作品ではライブイベントはもちろん、「2.5次元」などと銘打ったミュージカルなど、舞台や演劇という形式で興行しているものもあり、こちらも女性向け作品を中心に人気を博している。このように、これまではOculusやPSVRに代表されるヘッドマウントディスプレイを使用した仮想現実や、偶然にも「聖地巡礼」と同様に今年の流行となった「ポケモンGO」での携帯端末のカメラ映像にCGをオーバーラップさせる拡張現実などハードウェア中心の工業的手法で次元を融合させる試みばかりだったが、それ以外にも「聖地巡礼」のように非ハードウェア技術や新たな表現手法を用いた様々な発想で次元間をリンクさせようとする試みも多数見られるようになってきた。これにより、単なる創作作品だったものが読者や観覧者が実際に存在するものとしてさらに楽しむことが可能となってきた。これこそ21世紀型の二次元ヲタクの楽しみになるのではないか、そう感じる昨今である。さらに今後は、より２次元と３次元世界のシームレスな意識遷移が行える技術や表現も続々と登場するであろう。このように、今後は旧来の視覚や聴覚だけではなく、五感をフルに使って物語の世界を楽しめる仕掛けがさらに増えてくるだろうし、非常に楽しみなところである。

基本的にアニメや小説などの作品は、我々が存在する３次元空間では起きえない願望を具現化させるためのものであった。それが今後は、まるで実在の空間に創作物がぽつんと置かれたようなリアリティを持たせることがもっともっと緻密に、簡単に行えるようになる事だろう。クリエイターが頭の中で妄想していたものが現実世界にそのまま出現するのだ。

まだまだ21世紀は始まったばかり。これから80年が、妄想世界と現実世界のリンク化でバラ色に染まって行ければ…ひとりの２次元ヲタとして、切にそう願うばかりだ。

編集後記

このPDFはサンプルです。空気録学を今後ともよろしくお願いいたします。

https://chinachu.moe/



れいさ Twitter: @kanreisa

Chinachu とか Mirakurun とか Rivarun とかの開発者。UI／UX／ウェブ／アプリエンジニア。クリエイター。グラフィックデザイナー。主に FPS ゲーマー。面白いものと、可愛いものが好き。基本的に気まぐれ。最近、ポップコーン作りにハマっている。



mzyy94 Twitter: @mzyy94



らんだむ Twitter: @rndomhack



muken



akimasa Twitter: @akimasa2000

ただの Chinachu や Mirakurun のユーザー。PT2 で使っていて不具合に遭遇したため情報提供とか行ってみた。ゆるゆりは Amazon プライムビデオで見られる間に一通り見てみた。即、BD-BOX を発注した。



ろっと Twitter: @aayh



takaaki Twitter: @takaakis62

未来検索ブラジル。元サーバー屋。サーバー / ウェブが専門。MiyouTV の開発に駆り出され異分野で迷走したものの、ウェブ系への回帰に成功する。今も迷走を続ける。録画に関して、小学生の頃からビデオデッキに張り付く録画人間。中学時代、初めて買った PC も録画機能付き。ニコニコ実況を導入してからリアルタイムで見れるアニメを大体見ている雑食。録画サーバーは毎日エンコードで唸りを上げている。

sudosan Twitter: @sudosan



こすもす



あどみそ Twitter: @adomiso

私、あどみそ。アニヲタを目指して Chinachu をインストールしたけど初の録画ではい○り3話の録画に失敗してしまったの。(以下略



もずく Twitter: @_mozk_

吉川ちなつちゃんばっっかり描くストイックな地下イラストレーター。ちなみに今回の表紙絵のちなつちゃんはカラコンをしています。中の人はれ〓〓と〓じ。この間なもり先生の絵で逝きかけた。百合は世界平和に貢献している。



みりの Twitter: @millino

空気録学
平成28年12月29日発行（コミックマーケット発行）同日頒布　第2号　発行人◉菅祐貴　編集人◉れいさ　発行◉七森中録画研究会



mzyy94

こたつみかんの御供にどうですか号…みかんの汁で染まらないようにね！

秋葉原の夏きょうこの夏
あどみそ

所謂「聖地巡礼」と云うものについて。
または現実世界と交錯する２次元世界のこれからの話。
あどみそ

表紙
COVER STORY
RE:TRIED.
絵＝もずく（表紙・裏表紙）
みりの（表紙裏・裏表紙裏）

jikkyo で
アニメを
倍楽しもう
ろっと

演習問題で実力がつく！（気がする）
動画講座
らんだむ

PT2 サポート
情報
akimasa

もしかして難しい？
配信におけるエンコード
muken

MiyouTV with Chinachu
takaaki（未来検索ブラジル）

アニメの放送格差のお話
sudosan

最新版に完全追従
ぴゅあぴゅあ Mirakurun
れいさ

四方山話
こすもす

七森中録画研究会 C91

ISDN278-4-599800-05-7
C9404 ¥500E

2784599800057
2929404005009

定価 本体500円＋税