

# タブ 1

# W7white開発環境構築

「仮称:おもしろパソコン」をデイサービスに納品した。新しいrobotを作成するために開発環境をW7whiteに構築する。W7は複数所有しているので、区別するためにwhiteを名前につける。理由は、一部白塗りしてあるから。

T8は、「仮称:おもしろパソコン」の開発環境として、塩漬けする。いつでも、おもしろパソコンを修正できるように。。。おもしろパソコンは心外である。ロボットなのだ。

## 構成

・W7white:Panasonic CF-W7              OS:Debian12

## 目次

### 基本構成の検討

テキスト

Webブラウザー

音声入出力制御

端末

環境構築

少し考える。

1.再現性

2.環境のきれいに保つ

3.結論

### Docker 環境構築

必要なパッケージをインストール

Docker の公式 GPG キーを追加

Docker のリポジトリを追加

Docker をインストール

Docker サービスを起動し、有効化

Docker の動作確認

sudo なしで Docker を実行できるようにする

### urxvtのperlによる拡張機能

便利な Perl 拡張機能

結論

## フォントのインストール

[インストール](#)

[フォントキャッシュの更新](#)

[urxvtでフォント指定](#)

## VirtualBox

[インストール](#)

[vdiファイルのコピー](#)

[VirtualBoxで新規作製](#)

## Docker 開発環境のコピー

[image作成](#)

[imageの確認](#)

[containerの作成](#)

## ホストPCのアプリケーション

[Dockerfile](#)

[ビルド](#)

[Docker 実行](#)

## ソフトインストール用のcontainer作成

[Dockerfile作成](#)

[Docker containerの作成しdocker run実行スクリプト作成・実行](#)

## 今後の予定

## 参考

[質問:Docker インストール方法](#)

[回答:ChatGPT](#)

[質問:urxvtの最大化 不採用](#)

[回答:Gemini](#)

[回答:ChatGPT](#)

[質問:perlをpythonに変更 不採用](#)

[回答:Gemini](#)

[回答:ChatGPT](#)

[質問:urxvtのperlによる拡張機能](#)

[回答:ChatGPT](#)

[回答:Gemini](#)

[質問:フォントのインストール方法](#)

[回答:Gemini](#)

[質問:DockerでOpenScadを実行](#)

[回答:Gemini](#)

[回答:ChatGPT](#)

## 基本構成の検討

普段使いのPC作成。開発環境はDockerで構築するので、基本システムは最小限とする。一般的には、開発環境はホストPC、実行環境をDocker Containerに作製することが無駄がない。しかし、Containerを移動するだけで、開発環境を含めて全てを移動できることは魅力的である。なので、私は開発と実行環境をあわせて、Docker Containerの中で構築する。

## テキスト

- ・neovim: すべての基本
- ・fbterm: cui環境で日本語の表示・入力を行う。
- ・uim-mozc: cuiで日本語入力を行う。
- ・IPAexゴシック

## Webブラウザー

- ・firefox-esr: ChatGPT用  
    sudo apt install firefox-esr
- ・google-chrome: スプレットシート・ドキュメント・スライド等のアプリ  
    firefoxでgoogle chromeダウンロードサイトからプログラムをダウンロード  
    sudo apt install ./google-chrome-stable\_current\_amd64.deb

## 音声入出力制御

- ・sudo apt install alsa-utils pulseaudio-utils
- ・sudo apt install pulseaudio
- ・sudo apt install pavucontrol

## 端末

- ・urxvt: F11で最大化:  20230102urxvtをフルスクリーンに切替える

## 環境構築

- ・docker:

- 20250312DayServiceRB-kataをT8からW7にコンテナ移動

- ・VirtualBox

少し考える。

firefoxやgoogle-chromeもコンテナ内にいれてはどうか？

起動は、コンテナへのattachとexecを使えば、ホストからの起動は可能である。

その必要があるか、それをすることのメリットはなにか、美しいか、楽しいか？

コンテナに押し込めるメリットとはなにか？

### 1.再現性

開発を行っているときに、開発しているPC-(a)で出来ていることが、PC-(b)で同じことをしているつもりでもうまく作動しない。そんな経験がある。Dockerを使っていれば、それがほとんどなくなる。

PC-(a)で作った環境は、PC-(b)にそのまま持っていくのである。

### 2.環境のきれいに保つ

Dockerを使っていない場合、開発のためにアプリケーションのインストールと削除を繰り返す。このとき、環境が汚れていく。コンテナを使ていれば、それはコンテナ内に収めることができる。

### 3.結論

今回は、Webブラウザを、ホストPCにインストールする。

理由は、すでにインストールが終了しているから。そんな、理由？、時は金なり。やり直すのは、時間がもったいない。十分な理由である。なので、次回はブラウザもコンテナに収めることを再度検討する。

## Docker 環境構築

必要なパッケージをインストール

```
yo@W7white:~$ sudo apt update
```

```
yo@W7white:~$ sudo apt install -y ca-certificates curl gnupg
```

```
yo@W7white:~$ sudo apt install -y lsb-release
```

## Docker の公式 GPG キーを追加

```
yo@W7white:~$ sudo install -m 0755 -d /etc/apt/keyrings  
yo@W7white:~$ curl -fsSL  
https://download.docker.com/linux/debian/gpg | sudo tee  
/etc/apt/keyrings/docker.asc > /dev/null  
  
yo@W7white:~$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

## Docker のリポジトリを追加

```
yo@W7white:~$ echo "deb [arch=$(dpkg --print-architecture)  
signed-by=/etc/apt/keyrings/docker.asc]  
https://download.docker.com/linux/debian ${lsb_release -cs} stable" |  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
yo@W7white:~$ sudo apt update
```

## Docker をインストール

```
yo@W7white:~$ sudo apt install -y docker-ce docker-ce-cli  
containerd.io docker-buildx-plugin docker-compose-plugin
```

## Docker サービスを起動し、有効化

```
yo@W7white:~$ sudo systemctl enable --now docker  
<出力>  
Synchronizing state of docker.service with SysV service script with  
/lib/systemd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable docker
```

## Docker の動作確認

```
yo@W7white:~$ sudo docker run hello-world  
<出力>  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
e6590344b1a5: Pull complete  
Digest:  
sha256:7e1a4e2d11e2ac7a8c3f768d4166c2defeb09d2a750b01041  
2b6ea13de1efb19
```

Status: Downloaded newer image for hello-world:latest

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:  
<https://hub.docker.com/>

For more examples and ideas, visit:

```
https://docs.docker.com/get-started/
```

```
yo@W7white:~$ docker ps -a
```

<出力>

```
permission denied while trying to connect to the Docker daemon  
socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun  
%2Fdocker.sock/v1.48/containers/json?all=1": dial unix  
/var/run/docker.sock: connect: permission denied
```

```
yo@W7white:~$ sudo docker ps -a
```

<出力>

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS		PORTS	NAMES

```
4f8f58d9b3a6 hello-world "/hello" 24 seconds ago Exited (0) 20  
seconds ago agitated_montalcini
```

sudo なしで Docker を実行できるようにする  
sudo usermod -aG docker \$USER

## urxvtのperlによる拡張機能

便利なものが見つかったので、新規で追加してみる。

### 便利な Perl 拡張機能

いくつかの便利な拡張機能を追加する。

- `matcher` 端末内の URL をクリックして開く
- `clipboard` クリップボードのコピー・ペースト
- `keyboard-select` マウスなしでテキスト選択

```
yo@W7hite:~$ vi .Xdefault
```

以下を追記

```
URxvt.perl-ext: 1  
URxvt.perl-ext-common: default,matcher,clipboard,keyboard-select  
URxvt.urlLauncher: firefox  
URxvt.matcher.button: 1  
URxvt.keysym.M-c: perl:clipboard:copy  
URxvt.keysym.M-v: perl:clipboard:paste  
URxvt.keysym.M-s: perl:keyboard-select:activate
```

‘M-’ は `Alt` キーを意味する。  
いまいち、使い方が分からぬ。

### 結論

正常に動かないね。

## フォントのインストール

T8でどうしても引き継ぎたいのが、urxvtで使用しているフォントである。これがなかなか、難しくようやく探し当てた。

## インストール

```
yo@W7hite:~$ sudo apt install fonts-ipaexfont
```

## フォントキヤッショの更新

```
yo@W7hite:~$ fc-cache -fv
```

## urxvtでフォント指定

```
yo@W7hite:~$ vi .Xdefaults
```

### 下記を追記

```
URxvt.font: xft:DejaVu Sans Mono-10.5, xft:IPAGothic
```

## VirtualBox

将来、この記録が役に立つとは思えないがうろ覚えながら、記録を残す。

## インストール

当初、T8(Debian10)と同じ6. 1をインストール。VirtualBoxの起動は問題ないが、T8から持ってきたvdiが起動の途中でハンギングアップ。以前は、できたのだが、出来なくなってしまったのかと、諦めた。

新規で、インストールするならばVer7. 1を利用すべきだと考えた。

インストール方法は、Oracle公式HPよりダウンロード、以下のコマンドで実行した。

```
yo@W7white:~/Downloads$ sudo apt install  
./virtualbox-7.1_7.1.6-167084~Debian~bookworm_amd64.deb
```

## vdiファイルのコピー

T8のvdiファイルをW7whiteにコピー

```
yo@T8:~$ scp t8disk.vdi yo@192.168.11.7:/home/yo/Downloads
```

## VirtualBoxで新規作製

ここからが、苦労の連続。しかし、結果だけ記す。

新規にvdiを指定し、実行する。エラーとなる。エラーなのだがver6.1とは違い、ゲストOSの起動画面まで表示される。Logを確認すると、

Displayに問題ありそう。ディスプレイのグラフィックコントローラの設定を変更

VBoxVGA→VMSVGA:起動OK

しかし、ディスプレイの解像度、色数を変更できない。ここで、メニューウ→デバイス→GuestAddition CDの挿入し、インストール。途中、error表示ができるが、「無視」を選択し、インストール完了。

ゲストOSの電源を切り、VirtualBoxの設定からディスプレイの

グラフィックコントローラの設定:VMSVGA→VBoxVGA変更。

ゲストOS起動後、自動的に解像度・色数が正常に戻っていた。きつねに化かされた気分であるが、上記の内容で起動した。

以下の点に注意されたし。

・マウスの統合環境を切らなければ、クリックが反応しない。

・CPUの割当数を2以上にすると、起動しない。(T8では2だったが)

以上が、VirtualBoxの設定プロセスである。うろ覚えである。もっと、いろいろなことをやっているが、うまくいったのはこれだと思う。もしかしたら、前にやっていたことを忘れているが、それが問題を解決しているかもしれない。後日、この記録が、役に立つと良いのだが。

過去に、こんな方法もやっていたようだ。

■ 20240326Debian12にVirtualBoxインスト

インストールの後の作業については、記述がまったくない。そのときは、問題がなかったのか？思い出せない。

## Docker 開発環境のコピー

これまで、W7whiteで開発を行ってきたので、その環境をそのままコピーする。

参考: ■ 20250312DayServiceRB-kataをT8からW7にコンテナ移動

### image作成

前項で作成した.tarファイルからimageを作成する。

yo@W7white:~\$ docker load -i noetic\_audio\_rev4\_image.tar

### imageの確認

yo@W7white:~\$ docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
------------	-----	----------	---------	------

```
noetic_audio_rev4_image latest 337ffef92d26 2 weeks ago  
5.54GB
```

## containerの作成

docker runコマンドに付随するオプションが多く存在するのでシェルスクリプトで管理する。

```
yo@W7white:~$ cat docker-run-noetic-audio-rev4.sh  
#!/bin/bash  
docker run -it \  
--device /dev/dri:/dev/dri \  
--device /dev/snd \  
--device /dev/ttyUSB0:/dev/ttyUSB0 \  
-e DISPLAY=$DISPLAY \  
-e PULSE_SERVER=unix:${XDG_RUNTIME_DIR}/pulse/native \  
-v /home/yo/workspace:/home/yo/workspace \  
-v /etc/localtime:/etc/localtime \  
-v /tmp/.X11-unix:/tmp/.X11-unix \  
-v /dev/shm:/dev/shm \  
-v  
${XDG_RUNTIME_DIR}/pulse/native:${XDG_RUNTIME_DIR}/pulse/  
native \  
-v ~/.config/pulse/cookie:/root/.config/pulse/cookie \  
--net host \  
--group-add audio \  
--name noetic_audio_rev4 \  
noetic_audio_rev4_image
```

## 上記スクリプトの実行

```
yo@W7white:~$ ./docker-run-noetic-audio-rev4.sh
```

## ホストPCのアプリケーション

現在(20250406)、ホストPCにインストールしているアプリケーションは、firefoxとgoogle-chromeだけである。jpilotは、VirtualBox-win2kにplam-desktopを代替としてインストールした。あと、必要なアプリケー

ションは、OpenSCADとLibreCADである。これは、Docker内に収める。さて、ベースのOSに何を選択するか？ホストPCがDebian12なので、同じものを選択する。最新のDebianとするだけである。

## Dockerfile

```
yo@W7white:~/command/11Docker_cad$ vi Dockerfile
# ベースイメージ
FROM debian:12

# 環境変数(日本語ユーザー向け)
ENV DEBIAN_FRONTEND=noninteractive \
    TZ=Asia/Tokyo \
    LANG=ja_JP.UTF-8 \
    LC_ALL=ja_JP.UTF-8

# 必要なパッケージをインストール
RUN apt-get update && \
    apt-get install -y \
    locales \
    tzdata \
    x11-apps \
    openscad \
    libreCAD \
    && rm -rf /var/lib/apt/lists/*

# 日本語ロケールの設定
RUN sed -i '/ja_JP.UTF-8/s/^# //g' /etc/locale.gen && \
    locale-gen

# ユーザー作成(rootでなく)
RUN useradd -ms /bin/bash yo
USER yo
WORKDIR /home/yo

# 起動時のデフォルトコマンド(bash)
CMD ["bash"]
```

## ビルド

```
yo@W7white:~/command/11Docker_cad$ docker build -t
cad-gui-debian .
[+] Building 21.7s (9/9) FINISHED
docker:default
=> [internal] load build definition from Dockerfile
1.4s
=> => transferring dockerfile: 701B
0.4s
=> [internal] load metadata for docker.io/library/debian:12
3.9s
=> [internal] load .dockerignore
0.3s
=> => transferring context: 2B
0.0s
=> [1/5] FROM
docker.io/library/debian:12@sha256:18023f131f52fc3ea21973cabffe
0b216c60b417fd2478e94d9d59981ebba6af    0.1s
=> CACHED [2/5] RUN apt-get update && apt-get install -y
locales tzdata x11-apps openscad libr 0.0s
=> CACHED [3/5] RUN sed -i '/ja_JP.UTF-8/s/^# //g' /etc/locale.gen
&& locale-gen 0.0s
=> [4/5] RUN useradd -ms /bin/bash yo
8.2s
=> [5/5] WORKDIR /home/yo
2.0s
=> exporting to image
2.1s
=> => exporting layers
1.4s
=> => writing image
sha256:063c2b16af47127b45b7f8469dfee87b40e5d6aa4bc6d07ab3
a897d0ed79031c 0.1s
=> => naming to docker.io/library/cad-gui-debian
0.1s
```

## Docker 実行

```
yo@W7white:~/command/11Docker_cad$ vi docker_run_cad.sh
#!/usr/bin/bash
docker run -it \
-e DISPLAY=$DISPLAY \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v /home/yo/workspace:/home/yo/workspace \
-v /home/yo/00RB:/home/yo/00RB \
-v /etc/localtime:/etc/localtime \
cad-gui-debian
```

```
yo@W7white:~/command/11Docker_cad$ chmod +x
docker_run_cad.sh
yo@W7white:~/command/11Docker_cad$ ./docker_run_cad.sh
```

## ソフトインストール用のcontainer作成

その他、自由にソフトをインストールできるcontainerを作成する。

### Dockerfile作成

```
yo@W7white:~/command/12Docker_etc$ vi Dockerfile
FROM debian:12
```

```
# ロケールとタイムゾーンを日本に設定
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update && \
    apt-get install -y locales tzdata && \
    echo "ja_JP.UTF-8 UTF-8" > /etc/locale.gen && \
    locale-gen && \
    update-locale LANG=ja_JP.UTF-8 && \
    ln -fs /usr/share/zoneinfo/Asia/Tokyo /etc/localtime && \
    dpkg-reconfigure -f noninteractive tzdata
```

```
ENV LANG=ja_JP.UTF-8 \
LANGUAGE=ja_JP:ja \
LC_ALL=ja_JP.UTF-8 \
TZ=Asia/Tokyo
```

```
# 必要なパッケージをインストール
RUN apt-get update && apt-get install -y \
    sudo \
    byzanz \
    bash \
    dbus-x11 \
    x11-utils \
    x11-xserver-utils \
    xauth \
    x11-apps \
    wget \
    fonts-freefont-ttf \
    neovim \
    && rm -rf /var/lib/apt/lists/*

# ユーザー yo を作成し sudo 権限を付与
RUN useradd -ms /bin/bash yo && \
    echo "yo ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers

# ユーザーに切り替え
USER yo
WORKDIR /home/yo

# デフォルトシェル
CMD [ "bash" ]

Docker containerの作成しdocker run実行スクリプト作成・実行
yo@W7white:~/command/12Docker_etc$ vi docker_build_run.sh
#!/bin/bash
docker build -t byzanz-image .
docker run -it \
    -e DISPLAY=$DISPLAY \
    -v /tmp/.X11-unix:/tmp/.X11-unix \
    -v /home/yo/workspace:/home/yo/workspace \
    --name byzanz-container \
    byzanz-image
```

```
yo@W7white:~/command/12Docker_etc$ chmod +x  
docker_build_run.sh
```

```
yo@W7white:~/command/12Docker_etc$ ./docker_build_run.sh
```

## 今後の予定

- ・W7whiteの開発環境構築。
- ・新しいrbのurdfを定義する。
- ・Ryzen9rev2の環境構築。
- ・Hitabitの実行環境構築。

## 参考

- 20231208日本語フォントインストール
- 20220824遠隔PCの音を手元PCIに転送
- 20221003bashのキーバインドをVimに変更
- 20230102urxvtをフルスクリーンに切替える
- 20250312DayServiceRB-kataをT8からW7にコンテナ移動

質問:Docker インストール方法

debian12にDockerをインストールする方法を教えてください。

回答:ChatGPT

Debian 12 に Docker をインストールするには、以下の手順を実行してください。

---

### 1. 必要なパッケージをインストール

```bash

sudo apt update

sudo apt install -y ca-certificates curl gnupg

```

---

### ### 2. Docker の公式 GPG キーを追加

```bash

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo tee
/etc/apt/keyrings/docker.asc > /dev/null
sudo chmod a+r /etc/apt/keyrings/docker.asc
````
```

---

### ### 3. Docker のリポジトリを追加

```bash

```
echo "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/debian ${lsb_release -cs} stable" |
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
````
```

---

### ### 4. Docker をインストール

```bash

```
sudo apt install -y docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
````
```

---

### ### 5. Docker サービスを起動し、有効化

```bash

```
sudo systemctl enable --now docker
````
```

---

### ### 6. Docker の動作確認

```
```bash
sudo docker run hello-world
```
---
```

### ### 7.(オプション) sudo なしで Docker を実行できるようにする

```
```bash
sudo usermod -aG docker $USER
```
---
```

その後、ログアウトして再ログインすると、「`sudo` なしで `docker`」コマンドを実行できるようになります。

これで Debian 12 に Docker をインストールできます！ 

### 質問:urxvtの最大化 不採用

debian12でopenboxを使用しています。端末として、urxvtを利用してい  
るのですが、wmctrlを用いて画面の最大化をF11 キーで切り替えた  
い。perlを用いる設定方法を教えてください。

### 回答:Gemini

Debian 12のOpenboxでurxvt端末の最大化をF11キーで切り替えるた  
めのPerlスクリプトを作成し、Openboxの設定に統合する方法を説明し  
ます。

#### \*\*1. Perlスクリプトの作成\*\*

まず、端末の最大化を切り替えるPerlスクリプトを作成します。以下のス  
クリプトを`urxvt\_maximize.pl`という名前で保存してください。

```
```perl
#!/usr/bin/perl
```

```
use strict;
use warnings;

my $window_id = `xdotool getactivewindow`;
chomp $window_id;

my $state = `wmctrl -l -G -p -b | awk '$1 == "0x$window_id" {print
\$0}' | grep maximized`;

if ($state) {
    system("wmctrl -ir $window_id -b
remove,maximized_vert,maximized_horz");
} else {
    system("wmctrl -ir $window_id -b
add,maximized_vert,maximized_horz");
}
```

```

このスクリプトは、`xdotool`でアクティブなウィンドウのIDを取得し、`wmctrl`でそのウィンドウの最大化状態を調べます。最大化されている場合は解除し、そうでない場合は最大化します。

## \*\*2. スクリプトの実行権限付与\*\*

作成したスクリプトに実行権限を与えます。

```
```bash
chmod +x urxvt_maximize.pl
```

```

## \*\*3. スクリプトの配置\*\*

スクリプトを`/usr/local/bin`などの実行パスが通ったディレクトリに移動します。

```
```bash
sudo mv urxvt_maximize.pl /usr/local/bin/
```

```

## \*\*4. Openboxの設定\*\*

Openboxの設定ファイル`~/.config/openbox/rc.xml`を編集し、F11キーにスクリプトを割り当てます。

```
```xml
<keybind key="F11">
  <action name="Execute">
    <command>urxvt_maximize.pl</command>
  </action>
</keybind>
````
```

この設定を`<keyboard>`セクションに追加してください。

## \*\*5. Openboxの再起動\*\*

設定を反映するためにOpenboxを再起動します。

```
```bash
openbox --reconfigure
````
```

これで、F11キーを押すとurxvt端末の最大化が切り替わるはずです。

## \*\*注意点\*\*

\* `xdotool`と`wmctrl`がインストールされている必要があります。インストールされていない場合は、以下のコマンドでインストールしてください。

```
```bash
sudo apt install xdotool wmctrl
````
```

\* このスクリプトはアクティブなウィンドウがurxvt端末であることを前提としています。他のウィンドウでF11キーを押した場合、意図しない動作になる可能性があります。必要に応じてスクリプトを修正してください。

\* Openboxの設定ファイルの編集には注意が必要です。誤った設定を行うとOpenboxが起動しなくなる可能性があります。編集前に設定ファイルのバックアップを取ることをお勧めします。

回答:ChatGPT

`wmctrl`を使って`F11`キーで`urxvt`の最大化をトグルするには、Openboxのキーバインド設定(`~/.config/openbox/rc.xml`など)に以下のようPerlスクリプトを登録すると実現できます。

#### ### 1. Perlスクリプトを作成

`~/local/bin/toggle\_maximize.pl`というファイルを作成し、以下の内容を書き込んでください。

```
```perl
#!/usr/bin/perl
use strict;
use warnings;

# アクティブなウィンドウIDを取得
my $win_id = `xdotool getactivewindow`;
chomp($win_id);

# 現在のウィンドウ状態を取得
my $wm_state = `xprop -id $win_id _NET_WM_STATE`;
if ($wm_state =~ /_NET_WM_STATE_MAXIMIZED_VERT/ &&
$wm_state =~ /_NET_WM_STATE_MAXIMIZED_HORZ/) {
    # 最大化されている場合、元のサイズに戻す
    system("wmctrl -i -r $win_id -b
remove,maximized_vert,maximized_horz");
} else {
    # 最大化されていない場合、最大化する
    system("wmctrl -i -r $win_id -b
add,maximized_vert,maximized_horz");
}
````
```

#### ### 2. 実行権限を付与

ターミナルで以下のコマンドを実行してスクリプトを実行可能にします。

```
```sh
chmod +x ~/.local/bin/toggle_maximize.pl
```
```

### ### 3. Openbox にキーバインドを設定

`~/.config/openbox/rc.xml` を編集し、以下の ``<keyboard>`` セクションに追加します。

```
```xml
<keybind key="F11">
  <action name="Execute">
    <command>~/.local/bin/toggle_maximize.pl</command>
  </action>
</keybind>
```
```

### ### 4. Openbox を再読み込み

設定を反映するために、以下のコマンドを実行します。

```
```sh
openbox --reconfigure
```
```

これで `F11` を押すと、現在アクティブなウィンドウの最大化がトグルされるようになります。

質問: perlをpythonに変更 不採用

前項の最大化のperlスクリプトをpython3に書き換えるとどうなりますか？

回答: Gemini

前項のPerlスクリプトをPython 3に書き換えると、以下のようにになります。

```
```python
```

```
#!/usr/bin/env python3

import subprocess

def toggle_maximize():
    """アクティブなウィンドウの最大化状態を切り替える"""
    window_id = subprocess.check_output(["xdotool",
   "getactivewindow"]).decode().strip()
    state = subprocess.check_output(["wmctrl", "-l", "-G", "-p",
                                    "-b"]).decode()
    if f"0x{window_id}" in state and "maximized" in state:
        subprocess.run(["wmctrl", "-ir", window_id, "-b",
                      "remove,maximized_vert,maximized_horz"])
    else:
        subprocess.run(["wmctrl", "-ir", window_id, "-b",
                      "add,maximized_vert,maximized_horz"])

if __name__ == "__main__":
    toggle_maximize()
```

```

## \*\*変更点:\*\*

- \* \*\*shebang:\*\* `#!/usr/bin/env python3` を使用して、Python 3インタプリタを自動的に検出するようにしました。
- \* \*\*subprocessモジュール:\*\* `subprocess`モジュールを使用して、外部コマンド(`xdotool`と`wmctrl`)を実行します。
- \* \*\*文字列処理:\*\* Pythonの文字列処理を使用して、ウィンドウIDの取得と最大化状態の判定を行います。
- \* \*\*f-strings:\*\* f-stringsを使用して、文字列のフォーマットを簡潔に記述します。

## \*\*使用方法:\*\*

1. 上記のPythonスクリプトを`urxvt\_maximize.py`などの名前で保存します。
2. スクリプトに実行権限を付与します。

```
```bash
chmod +x urxvt_maximize.py
```

```

3. スクリプトを`/usr/local/bin`などの実行パスが通ったディレクトリに移動します。

```
```bash
sudo mv urxvt_maximize.py /usr/local/bin/
```

```

4. Openboxの設定ファイル`~/.config/openbox/rc.xml`を編集し、F11キーにスクリプトを割り当てます。

```
```xml
<keybind key="F11">
  <action name="Execute">
    <command>urxvt_maximize.py</command>
  </action>
</keybind>
```

```

5. Openboxを再起動します。

```
```bash
openbox --reconfigure
```

```

#### \*\*注意点:\*\*

\* `xdotool`と`wmctrl`がインストールされている必要があります。インストールされていない場合は、以下のコマンドでインストールしてください。

```
```bash
sudo apt install xdotool wmctrl
```

```

- \* このスクリプトはアクティブなウィンドウがurxvt端末であることを前提としています。他のウィンドウでF11キーを押した場合、意図しない動作になる可能性があります。必要に応じてスクリプトを修正してください。
- \* Openboxの設定ファイルの編集には注意が必要です。誤った設定を行うとOpenboxが起動しなくなる可能性があります。編集前に設定ファイルのバックアップを取ることをお勧めします。

回答:ChatGPT

Perl スクリプトを Python 3 に書き換えると、以下のようになります。Python の `subprocess` モジュールを使用して `wmctrl` と `xprop` を実行し、ウィンドウの最大化をトグルします。

```
### **1. Python スクリプトの作成**  
`~/local/bin/toggle_maximize.py` を作成し、以下の内容を書き込んでください。
```

```
```python  
#!/usr/bin/env python3  
  
import subprocess  
  
def get_active_window():  
    """アクティブなウィンドウIDを取得"""  
    try:  
        win_id = subprocess.check_output(["xdotool",  
"getactivewindow"]).strip().decode("utf-8")  
        return win_id  
    except subprocess.CalledProcessError:  
        return None  
  
def is_maximized(win_id):  
    """ウィンドウが最大化されているか確認"""  
    try:  
        win_state = subprocess.check_output(["xprop", "-id", win_id,  
"_NET_WM_STATE"]).decode("utf-8")
```

```
    return "_NET_WM_STATE_MAXIMIZED_VERT" in wm_state  
and "_NET_WM_STATE_MAXIMIZED_HORZ" in wm_state  
except subprocess.CalledProcessError:  
    return False
```

  

```
def toggle_maximize():  
    """アクティブなウィンドウの最大化をトグル"""  
    win_id = get_active_window()  
    if not win_id:  
        return  
  
    if is_maximized(win_id):  
        # 最大化を解除  
        subprocess.run(["wmctrl", "-i", "-r", win_id, "-b",  
"remove,maximized_vert,maximized_horz"])  
    else:  
        # 最大化  
        subprocess.run(["wmctrl", "-i", "-r", win_id, "-b",  
"add,maximized_vert,maximized_horz"])  
  
    if __name__ == "__main__":  
        toggle_maximize()  
    ...
```

#### \*\*2. 実行権限を付与\*\*  
ターミナルで以下のコマンドを実行し、スクリプトを実行可能にします。

```
```sh  
chmod +x ~/.local/bin/toggle_maximize.py  
```
```

#### \*\*3. Openbox にキーバインドを設定\*\*  
`~/.config/openbox/rc.xml` を編集し、以下の `` セクションに追加します。

```
```xml  
<keybind key="F11">  
  <action name="Execute">
```

```
<command>python3  
~/local/bin/toggle_maximize.py</command>  
</action>  
</keybind>  
```
```

#### ### \*\*4. Openbox を再読み込み\*\*

設定を反映するために、以下のコマンドを実行します。

```
```sh  
openbox --reconfigure  
```
```

#### ### \*\*動作概要\*\*

- `F11` キーを押すと、現在アクティブなウィンドウが最大化される／元のサイズに戻る
- `xdotool` でアクティブウィンドウの ID を取得
- `xprop` で最大化状態を確認
- `wmctrl` で最大化をトグルする

これで Python 版の最大化トグルスクリプトが動作するはずです！

質問:urxvtのperlによる拡張機能

debian12のurxvtにおいて、perlによる拡張機能を使う方法について教えてください。

回答:ChatGPT

Debian 12 の `urxvt` で Perl による拡張機能を使うには、以下の手順を実行してください。

---

#### ## 1. 必要なパッケージのインストール

Debian 12 では `urxvt` の Perl 拡張機能を使用するには、`rxvt-unicode-256color` パッケージが必要です。

また、拡張機能によっては `perl` が必要になるため、念のため確認・インストールしておきます。

```
```sh
sudo apt update
sudo apt install rxvt-unicode-256color perl
```
---
```

## ## 2. 設定ファイルの編集

Perl 拡張機能を有効にするには `~/.Xresources` に設定を追加します。

#### 基本設定 (`~/.Xresources`)  
以下の行を追加または編集してください。

```
```plaintext
URxvt.perl:      1
URxvt.perl-ext-common: default,matcher
URxvt.urlLauncher: firefox
URxvt.matcher.button: 1
```
---
```

この設定では、以下の機能を有効にしています：

- `URxvt.perl: 1` → Perl 拡張機能を有効化
- `URxvt.perl-ext-common: default,matcher` → デフォルトの拡張機能と `matcher` を有効化
- `matcher` を使うと、端末内の URL をクリックして開ける
- `URxvt.urlLauncher: firefox` → URL をクリックした際に Firefox で開く
- `URxvt.matcher.button: 1` → 左クリック (ボタン1) で URL を開く

設定を反映するには、以下のコマンドを実行します。

```
```sh
xrdb -merge ~/.Xresources
```
---
```

## 3. Perl スクリプトを追加する  
独自の Perl スクリプトを `~/.urxvt/ext/` に配置すれば、追加の拡張機能を作成できます。  
例えば、カスタムスクリプト `example.pl` を作成する場合：

```
```sh
mkdir -p ~/.urxvt/ext
nano ~/.urxvt/ext/example.pl
```

```

`example.pl` に以下のような内容を記述：

```
```perl
#!/usr/bin/perl
# シンプルな Perl 拡張の例

package urxvt::example;

sub on_start {
    my ($self) = @_;
    $self->print("Hello from Perl extension!\n");
}

1;
```

```

そして、`~/.Xresources` に追加：

```
```plaintext
URxvt.perl-ext-common: default,matcher,example
```

```

設定を再読み込みし、urxvt を再起動すれば、ターミナル起動時に「Hello from Perl extension!」と表示されます。

---

## ## 4. その他の便利な Perl 拡張機能

いくつかの便利な拡張機能を紹介します。

| 拡張機能 | 説明 |

| -----                            | ----- |
|----------------------------------|-------|
| `matcher`   端末内の URL をクリックして開く   |       |
| `clipboard`   クリップボードのコピー・ペースト   |       |
| `keyboard-select`   マウスなしでテキスト選択 |       |

利用するには `~/.Xresources` に追加します。

```
```plaintext
URxvt.perl-ext-common: default,matcher,clipboard,keyboard-select
```

```

また、キーバインドを設定することで `keyboard-select` を使いやすくなります。

```
```plaintext
URxvt.keysym.M-c: perl:clipboard:copy
URxvt.keysym.M-v: perl:clipboard:paste
URxvt.keysym.M-s: perl:keyboard-select:activate
```

```

ここで `M-` は `Alt` キーを意味します。

## ## 5. 反映と確認

設定を反映するには以下を実行してください。

```
```sh
xrdb -merge ~/.Xresources
killall urxvt # 現在の urxvt を閉じる
urxvt &
```

```

この手順で `urxvt` の Perl 拡張機能を利用できるはずです。

---

試してみて、問題があれば教えてください！

回答:Gemini

Debian 12のurxvtでPerl拡張機能を使用する方法について説明します。urxvtはPerlスクリプトによる拡張が可能であり、さまざまな機能を追加できます。

#### \*\*1. 必要なパッケージのインストール\*\*

まず、Perl拡張機能を使用するために必要なパッケージをインストールします。以下のコマンドを実行してください。

```
```bash
sudo apt update
sudo apt install rxvt-unicode-perl
````
```

'rxvt-unicode-perl'パッケージをインストールすることで、Perl拡張機能を使用するためのライブラリやスクリプトがインストールされます。

#### \*\*2. Perl拡張スクリプトの配置\*\*

Perl拡張スクリプトは、通常、以下のディレクトリに配置します。

- \* `/usr/lib/urxvt/perl/` (システム全体で利用可能なスクリプト)
- \* `~/.urxvt/ext/` (ユーザー固有のスクリプト)

自分で作成したスクリプトや、インターネットからダウンロードしたスクリプトをこれらのディレクトリに配置します。

#### \*\*3. .Xresourcesまたは.Xdefaultsの設定\*\*

urxvtの設定ファイルである`~/.Xresources`または`~/.Xdefaults`に、使用するPerl拡張スクリプトを指定します。例えば、`url-select`拡張機能を使用する場合は、以下のように設定します。

```
```
URxvt.perl-ext-common: default,url-select
````
```

複数の拡張機能を指定する場合は、カンマ区切りで記述します。

#### \*\*4. 設定の反映\*\*

設定ファイルを変更した後、urxvtを再起動するか、以下のコマンドを実行して設定を反映させます。

```
```bash
xrdb ~/.Xresources
````
```

または

```
```bash
xrdb ~/.Xdefaults
````
```

#### \*\*5. Perl拡張スクリプトの例\*\*

以下に、簡単なPerl拡張スクリプトの例を示します。このスクリプトは、urxvtの背景色をランダムに変更するものです。

```
```perl
# ~/.urxvt/ext/randbg.pl
package URxvt::perllib::randbg;

sub on_start {
    my $self = shift;
    $self->randbg();
}
````
```

```
sub randbg {
    my $self = shift;
    my @colors = qw(red green blue yellow cyan magenta);
    my $color = $colors[rand @colors];
    $self->command("bg $color");
}

1;
...
```

このスクリプトを使用するには、`Xresources`に以下のように記述します。

```
...
URxvt.perl-ext-common: default,randbg
...
```

## \*\*6. 注意事項\*\*

- \* Perl拡張スクリプトは、urxvtの動作に影響を与える可能性があるため、信頼できるスクリプトのみを使用してください。
- \* Perl拡張スクリプトの記述方法については、urxvtのドキュメントや、インターネット上の情報を参考にしてください。

これらの手順に従うことで、Debian 12のurxvtでPerl拡張機能を使用することができます。

質問:フォントのインストール方法

ipaexg.ttf debianで使う方法を教えてください

回答:Gemini

fbterm で IPAex ゴシックフォント ('ipaexg.ttf') を使用する方法は、直接フォントファイルを指定するのではなく、fontconfig の設定を利用するのが一般的です。以下に手順を説明します。

\*\*1. IPAex ゴシックフォントのインストール:\*\*

まず、IPAex ゴシックフォントが Debian システムにインストールされている必要があります。

\* \*\*パッケージマネージャを使用する場合:\*\*

\* ターミナルを開き、以下のコマンドを実行して IPAex フォントパッケージをインストールします。

```
```bash
sudo apt install fonts-ipaexfont
```
```

\* \*\*手動でインストールする場合:\*\*

\* IPAex フォントの配布サイトから `ipaexg.ttf` ファイルをダウンロードし、適切なフォントディレクトリ (`~/.local/share/fonts` または `/usr/share/fonts`) にコピーします。

\* フォントキャッシュを更新します。

```
```bash
fc-cache -fv
```
```

\*\*2. fbterm の設定:\*\*

fbterm は、fontconfig の設定に従ってフォントを検索します。そのため、fontconfig で IPAex ゴシックフォントが適切に設定されていることを確認する必要があります。

\* \*\*fontconfig 設定ファイルの確認:\*\*

\* `~/.config/fontconfig/fonts.conf` ファイルが存在する場合は、内容を確認し、IPAex ゴシックフォントが適切に設定されているか確認します。

\* ファイルが存在しない場合は、作成する必要があります。

\* \*\*fbterm の起動:\*\*

\* fbterm を起動する際に、`-f` オプションでフォントを指定することができますが、fontconfig のエイリアス名を使用するのが推奨されます。

\* 例えば、IPAex ゴシックフォントをデフォルトのフォントとして使用したい場合は、`fbterm` コマンドをそのまま実行します。

\* または、`-f` オプションでフォントのエイリアス名を指定することもできます。例:

```
```bash
fbterm -f "IPAexGothic"
```

```

### \*\*3. fbterm の設定ファイル (`~/.fbtermrc`) の編集 (オプション):\*\*

fbterm の設定ファイル (`~/.fbtermrc`) を編集することで、デフォルトのフォントを指定できます。

\* `~/.fbtermrc` ファイルを開き、`font-names` の行を編集します。例:

```
```
font-names=IPAexGothic
```

```

\* ファイルを保存し、fbterm を再起動します。

### \*\*補足:\*\*

\* fontconfig の設定によっては、IPAex ゴシックフォントが正しく認識されない場合があります。その場合は、`~/.config/fontconfig/fonts.conf` ファイルを編集して、フォントのエイリアスやパスを明示的に指定する必要があります。

\* fbterm で使用できるフォントは、フレームバッファデバイスの解像度や文字コードの設定によって制限される場合があります。

\* もし、IPAexGothicのフォントファミリーネームが認識されない場合は、`fc-list`コマンドでfontconfigが認識しているフォントファミリーネームを確認し、`font-names`に正しく記述してください。

これらの手順に従って、fbterm で IPAex ゴシックフォントを使用してください。

質問:DockerでOpenScadを実行

Dockerを使って、openscadとlibrecadを使いたい。ホストPCは、Debian12でDockerはインストール済みである。ゲストOSとしてDebian12を使い、OpenScadとLiberCADを使うためのDockerfileの提示をお願いします。

回答:Gemini

はい、Debian 12をベースとしたOpenSCADとLibreCADが利用できるDockerfileと、それに関連する情報を提供します。

\*\*Dockerfile\*\*

```dockerfile

FROM debian:12

# 必要なパッケージをインストール

```
RUN apt-get update && apt-get install -y --no-install-recommends \
    openscad \
    librecad \
    xvfb \
    x11-utils \
    && rm -rf /var/lib/apt/lists/*
```

# 作業ディレクトリを作成

WORKDIR /home/user

# OpenSCADとLibreCADを実行するためのスクリプト

```
COPY docker-entrypoint.sh /usr/local/bin/
RUN chmod +x /usr/local/bin/docker-entrypoint.sh
```

ENTRYPOINT ["/usr/local/bin/docker-entrypoint.sh"]

CMD ["openscad"]

```

\*\*docker-entrypoint.sh (実行スクリプト)\*\*

```bash

```
#!/bin/bash
set -e

if [ "$1" = "openscad" ]; then
    xvfb-run --auto-servernum --server-args='-screen 0 1024x768x24'
    openscad "$@"
elif [ "$1" = "librecad" ]; then
    xvfb-run --auto-servernum --server-args='-screen 0 1024x768x24'
    librecad "$@"
else
    exec "$@"
fi
...
```

#### \*\*Dockerfileと実行スクリプトの説明:\*\*

\* \*\*\*`FROM debian:12`\*\*\*: ベースイメージとしてDebian 12を使用します。

\* \*\*\*`RUN apt-get update && apt-get install -y ...`\*\*\*:  
\* パッケージリストを更新し、`openscad`、`librecad`、仮想フレームバッファの`xvfb`、X11ユーティリティの`x11-utils`をインストールします。  
\* `--no-install-recommends` オプションは、推奨パッケージのインストールをスキップし、イメージサイズを小さくするのに役立ちます。

\* `&& rm -rf /var/lib/apt/lists/\*` は、パッケージリストのキャッシュを削除し、イメージサイズを削減します。

\* \*\*\*`WORKDIR /home/user`\*\*\*: コンテナ内の作業ディレクトリを設定します。

\* \*\*\*`COPY docker-entrypoint.sh /usr/local/bin/`\*\*\*: ホストPCにある`docker-entrypoint.sh` スクリプトをコンテナ内の `/usr/local/bin/` にコピーします。

\* \*\*\*`RUN chmod +x /usr/local/bin/docker-entrypoint.sh`\*\*\*: コピーしたスクリプトに実行権限を与えます。

\* \*\*\*`ENTRYPOINT [" /usr/local/bin/docker-entrypoint.sh"]`\*\*\*: コンテナ起動時に実行されるデフォルトのコマンドとして、作成したスクリプトを設定します。

\* \*\*\*`CMD ["openscad"]`\*\*\*: `ENTRYPOINT` に渡されるデフォルトの引数を設定します。これにより、`docker run` だけを実行した場合、OpenSCADが起動しようとします。

## \*\*docker-entrypoint.sh の説明:\*\*

- \* このスクリプトは、コンテナ起動時に実行され、実行するアプリケーション(`openscad` または `librecad`)を判別します。
- \* \*\*\*`xvfb-run ...`\*\*: OpenSCADやLibreCADのようなGUIアプリケーションを、グラフィック環境がないコンテナ内で実行するために `xvfb` を使用します。
  - \* `--auto-servernum`: 利用可能な最初のサーバー番号を自動的に選択します。
    - \* `--server-args='-screen 0 1024x768x24'`: 仮想スクリーンのサイズと色深度を設定します。
  - \* 引数として "openscad" が渡された場合は OpenSCAD を、"librecad" が渡された場合は LibreCAD を `xvfb-run` 経由で実行します。
  - \* それ以外の引数が渡された場合は、そのまま実行します。これにより、他のコマンドもこのエントリーポイントスクリプトを通じて実行できます。

## \*\*Dockerfileと実行スクリプトの利用手順:\*\*

### 1. \*\*ファイルの作成:\*\*

- \* 上記の内容で `Dockerfile` という名前のファイルを作成します。
- \* 上記の内容で `docker-entrypoint.sh` という名前のファイルを作成し、`Dockerfile` と同じディレクトリに保存します。

### 2. \*\*Dockerイメージのビルド:\*\*

```
```bash
docker build -t openscad-librecad .
````
```

このコマンドは、現在のディレクトリにある `Dockerfile` を基に `openscad-librecad` という名前のDockerイメージをビルドします。

### 3. \*\*Dockerコンテナの実行 (OpenSCAD):\*\*

```
```bash
docker run -it --rm -v /tmp/.X11-unix:/tmp/.X11-unix -e
DISPLAY=$DISPLAY openscad-librecad
````
```

\* `-it`: 対話型ターミナルを割り当てます。

\* `--rm`: コンテナ終了時に自動的に削除します。

- \* ``-v /tmp/.X11-unix:/tmp/.X11-unix``: ホストPCのXサーバーソケットをコンテナにマウントします。
- \* ``-e DISPLAY=\$DISPLAY``: ホストPCのDISPLAY環境変数をコンテナに渡します。
- \* ``openscad-librecad``: ビルドしたイメージの名前。
- \* デフォルトで `CMD ["openscad"]` が実行されるため、OpenSCADが起動しようとします。ただし、コンテナ内にはGUIがないため、このままで画面は表示されません。

#### 4. \*\*Dockerコンテナの実行 (LibreCAD):\*\*

```bash

```
docker run -it --rm -v /tmp/.X11-unix:/tmp/.X11-unix -e  
DISPLAY=$DISPLAY openscad-librecad libreCAD  
```
```

- \* 最後の `librecad` が `docker-entrypoint.sh` に渡され、LibreCADが `xvfb-run` 経由で実行されます。ただし、同様にGUIは表示されません。

#### \*\*GUIの表示について:\*\*

上記のDockerfileと実行方法では、コンテナ内でOpenSCADやLibreCADは起動しますが、ホストPCの画面にGUIを表示させるには追加の設定が必要です。一般的には以下のいずれかの方法が用いられます。

- \* \*\*X転送:\*\* 上記の ``-v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=\$DISPLAY`` はX転送を試みる設定ですが、セキュリティ上の理由や設定によってはうまくいかない場合があります。ホストPC側でXサーバーが適切に設定されているか確認してください。
- \* \*\*VNCやRDP:\*\* コンテナ内にVNCサーバーやRDPサーバーをインストールし、ホストPCから接続する方法です。この方法はより安定していますが、Dockerfileの修正と追加のソフトウェアのインストールが必要です。
- \* \*\*Webベースのインターフェース:\*\* OpenSCADやLibreCAD自体には標準でWebインターフェース機能はありません。Webブラウザ経由で操作するには、追加のラッパーや変換ツールが必要になる場合があります。

## \*\*X転送がうまくいかない場合の確認事項:\*\*

- \* ホストPCのXサーバーがTCP接続を許可しているか (例: `xhost +` を実行してみる)。ただし、セキュリティリスクがあるため、使用後は `xhost -` でアクセス制御を戻すことを推奨します。
- \* ホストPCのDISPLAY環境変数が正しく設定されているか (`echo \$DISPLAY` で確認)。
- \* Dockerがホストのネットワークにアクセスできる設定になっているか。

## \*\*より高度な利用:\*\*

- \* OpenSCADのスクリプトやLibreCADの図面をホストPCと共有するために、`-v` オプションを使ってディレクトリをマウントすることができます。

```bash

```
docker run -it --rm -v /path/to/your/cad_files:/home/user/cad_files  
-v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY  
openscad-librecad openscad /home/user/cad_files/your_design.scad  
...``
```

このDockerfileと実行スクリプトを基本として、ご自身の環境や目的に合わせてカスタマイズしてください。GUIの表示に関しては、上記の方法を検討し、最適なものを選択してください。

回答:ChatGPT

了解しました！

以下に、\*\*Debian 12 をベースにした Dockerfile\*\* を提示します。この Docker イメージは `OpenSCAD` と `LibreCAD` をインストールし、  
\*\*X11 を通して GUI アプリケーションをホスト側で表示できるように設定\*\*します。

---

### ### 🔧 必要な前提(ホスト側)

まず、ホスト側で以下を確認・準備してください:

1. \*\*X11が動作していること\*\*

2. DockerでGUIアプリを表示できるように設定する(例: `xhost`)

```
```bash
# X11のアクセス許可を与える(セキュリティ注意)
xhost +local:docker
```
---
```

## 🐳 Dockerfile (Debian 12 + OpenSCAD + LibreCAD)

```Dockerfile

# ベースイメージ

FROM debian:12

# 環境変数(日本語ユーザー向け)

ENV DEBIAN\_FRONTEND=noninteractive \

  TZ=Asia/Tokyo \

  LANG=ja\_JP.UTF-8 \

  LC\_ALL=ja\_JP.UTF-8

# 必要なパッケージをインストール

RUN apt-get update && \

  apt-get install -y \

  locales \

  tzdata \

  x11-apps \

  openscad \

  librecad \

  && rm -rf /var/lib/apt/lists/\*

# 日本語ロケールの設定

RUN sed -i '/ja\_JP.UTF-8/s/^# //g' /etc/locale.gen && \

  locale-gen

# ユーザー作成(rootでなく)

RUN useradd -ms /bin/bash user

USER user

```
WORKDIR /home/user
```

```
# 起動時のデフォルトコマンド(bash)
```

```
CMD ["bash"]
```

```
---
```

```
---
```

```
## 🚧 ビルド方法
```

この `Dockerfile` を保存したディレクトリで:

```
```bash
```

```
docker build -t cad-gui-debian .
```

```
---
```

```
---
```

```
## 🚶 実行方法(GUIを表示)
```

```
```bash
```

```
docker run -it \
```

```
--rm \
```

```
-e DISPLAY=$DISPLAY \
```

```
-v /tmp/.X11-unix:/tmp/.X11-unix \
```

```
cad-gui-debian
```

```
---
```

この状態で、コンテナ内のターミナルで:

```
```bash
```

```
openscad &
```

```
librecad &
```

```
---
```

を実行すれば、ホストの画面にそれぞれのGUIが表示されるはずです。

```
---
```

## ## 補足

- `x11-apps` はテスト用。`xeyes` や `xclock` で X11 転送の確認ができます。
- 必要なら `volume` オプションでホストのCADデータをマウント可能です。

---

必要であれば、このDocker環境をもっと日本語向けにしたり、他のCADツールを追加したりすることもできます。  
気になるところがあれば教えてください！