Fuchsia OS

dahlia OS

Operating System Maniacs

(2nd Seasons)

Version 4.0

Google「Fuchsia OS」を実機にインストールする

はじめに

「Fuchsia」は、数年前に開始され、最近になって明示的にサイト開設がされたことなどで若干の話題になっている Google の「第 3 の」OS と呼ばれているものです。

直近でついに搭載したデバイスが発売されるとかなんとかといった記事も出てきているようです。

https://cloud.watch.impress.co.jp/docs/column/infostand/1328053.html

本稿では前号でも少しコメントした通り、実機への Fuchsia のインストールをやっていきます。

また、前半では前号で実施した Fuchsia のビルド環境の構築についても簡単に記載します。

メインサイト

https://fuchsia.dev/

前号でも記載しましたが、基本的に今回やっていることは上記サイトを見ればすべて実施可能です。

気を付けるところは実機を購入する場合に明示的に対応しているとされているものを購入するように、ということくらいでしょうか。

ハードウェア

今回導入するのはオフィシャルサイト内の各種資料内でインストール可能なデバイスとしてあげらている、Intel NUC 6 にインストールを行います。SSD はありものの流用、メモリは新規に購入しました。

Fuchsia のビルドと実行

それでは、Fuchsia のビルドと実行を行っていきます。

なお、前半で簡単に Fuchsia のビルド環境構築も振り返ります。

ビルド環境の構築

Virtual Box 上の Ubuntu 20.04 LTS を用います。

CPU：4
メモリ：16GB
ディスク：200GB

```
VBoxManage modifyvm "【仮想マシン名】" --nested-hw-virt on
```

で、仮想化のネスト設定も有効化しています。

最新版へのアップデートを忘れないようにしましょう。

```
$ sudo apt update
$ sudo apt upgrade
```

必須ツールのインストール

オフィシャルサイトの記載通りに以下ソフトウェアをインストールします。

```
$ sudo apt-get install build-essential curl git python unzip
```

FUSHSIA のソースをダウンロード

改行されてますが curl 以降は 1 行で入力します

```
$ cd ~
$ curl -s "https://fuchsia.googlesource.com/fuchsia/+/HEAD/scripts/bootstrap?format=TEXT" |
base64 --decode | bash
```

ダウンロード後、一部が認証エラーでダウンロードできなかった旨が表示されるので、指示されたコマンドを実行します。

【User 名】は、実際は実行したユーザー名が表示されます。

```
WARN: Some packages are skipped by cipd due to lack of access, you might want to run "/home/
【User 名】/fuchsia/.jiri_root/bin/cipd auth-login" and try again

Visit the following URL to get the authorization code and copy-paste it below.

https://accounts.google.com/o/oauth2/【非常に長い認証用の URL 文字列が表示されます】

Authorization code:
```

表示された認証用 URL をブラウザにコピペして、Google アカウントでのログインを行うと、Authorization code が表示されるので、それを上記の「Authorization code：」のところにコピペします。

```
  Scopes:

    https://www.googleapis.com/auth/userinfo.email

    openid
```

認証に成功したらダウンロードスクリプトを再実行します。

```
$ curl -s "https://fuchsia.googlesource.com/fuchsia/+/HEAD/scripts/bootstrap?format=TEXT" |
base64 --decode | bash
```

```
Updating all projects
Done creating a Platform Source Tree at "/home/【User 名】/fuchsia".
Recommended: export PATH="/home/【User 名】/fuchsia/.jiri_root/bin:$PATH"
```

パスの追加を指示されるので実行します。

また、この際に、fuchsia の各種操作を行うコマンドを実行できるように「fx-env.sh」というスクリプトも有効になるようにしておきます。

```
$ vi .bashrc
～以下追記～
export PATH=~/fuchsia/.jiri_root/bin:$PATH
source ~/fuchsia/scripts/fx-env.sh

$ source ~/.bashrc
```

パスが通ったことを確認するために、パッケージの取得ツールである「jiri」と Fuchsia のビルドのためのツールである「fx」のヘルプを表示してみます。

```
$ jiri help
Command jiri is a multi-purpose tool for multi-repo development.
Usage:
   jiri [flags] <command>
The jiri commands are:
～以下省略～
```

以下 jiri コマンドについての詳細な情報が多数表示されます。

```
$ fx help
usage: fx [--dir BUILD_DIR] [-d DEVICE_NAME] [-i] [-x] COMMAND [...]
Run Fuchsia development commands. Must be run either from a directory
that is contained in a Platform Source Tree or with the FUCHSIA_DIR
environment variable set to the root of a Platform Source Tree.
host shell extensions: (requires "source scripts/fx-env.sh")
  fx-update-path          add useful tools to the PATH
  fx-set-prompt           display the current configuration in the shell prompt
～以下省略～
```

以下 fx コマンドについての詳細な情報が多数表示されます。

Path が通ったのを確認したところで、いよいよ Fuchsia をビルドしていきます。

FUCHSIA のビルド

基本的なビルド手順は

```
$ fx set
```

コマンドで、ビルド対象のプロダクトとターゲットアーキテクチャ、モード等の設定を行い、

```
$ fx build
```

コマンドで fuchsia のビルドを行います。

fx set の後に必ず指定する必要のあるオプションは

```
$ fx set 【プロダクト名】.【ターゲットアーキテクチャ名】
```

になります。

【プロダクト名】

プロダクト名には ./products ディレクトリ以下の .gni ファイル名を指定可能です。

コマンド

```
$ fx list-products
```

で確認することができます。

デフォルトで用意されているものは以下のものがあります。

・**bringup**　Zircon カーネルと最低限のドライバ

・**core**　　bringup にさらにネットワークの機能やシステムサービスを追加したもの。

・**terminal**　core に基づきさらにコマンドライン端末を備えたシンプルなグラフィカルユーザーインターフェイスを備えたシステム。

・**workstation**　GUI 付きのパッケージを追加したもの。

【ターゲットアーキテクチャ名】

ターゲットとするアーキテクチャの指定です。

```
$ fx list-boards
```

で詳細な情報を得ることができます。なお、アップデート時に削られたのか、前号での表示からずいぶんと減っています。

```
arm64
chromebook-x64
*
qemu-arm64
qemu-x64
toulouse
vim2
vim3
x64
x64-reduced-perf-variation
```

今回は Intel NUC で動かしますので「workstasion.x64」を使ってビルドします。

最初の--dir オプションは、セットアップの際に出力する際のディレクトリを個別に指定するためのものです。これで同じセットアップ情報でも、別々にビルドすることも可能です。

```
$ fx --dir out/NUC set core.workstation.x64

$ fx build
```

セットはそれなりにすぐ完了しますが、セットアップ後の build はかなり時間がかかりますのでしばし待ちます。

その他オプション等：

--release

オプションを付けると、リリースビルドになります。つけていない場合はデバッグビルドです。

fx set-device

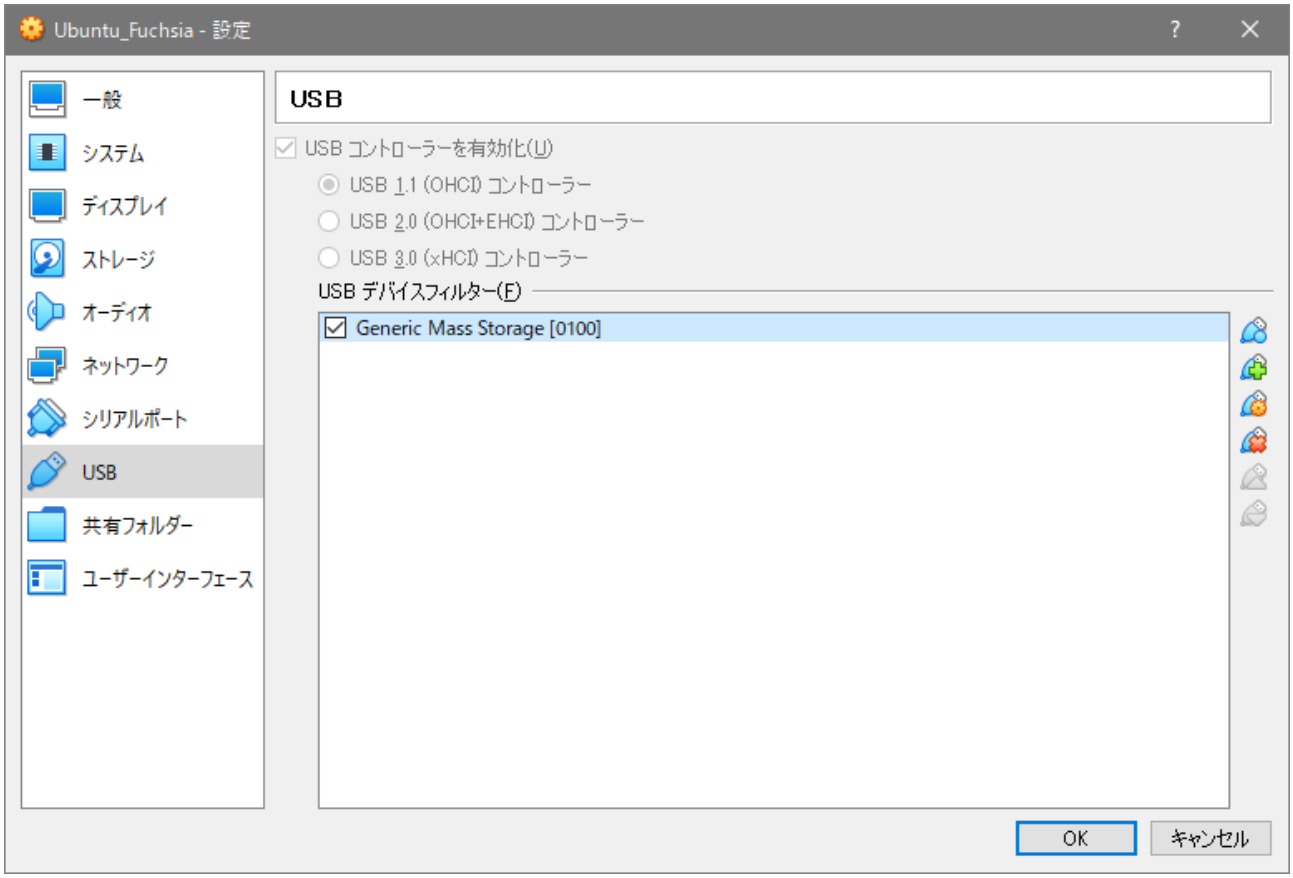
を用いて設定することで、同じディレクトリから別々のデバイス向けの build を出力することもできるようです。

インストール用 USB メモリの作成

ビルドが完了したら、開発側マシンに USB メモリを認識させて、そこにブートローダーを構成します。

今回は VirtualBox で開発環境を構築していますので、仮想マシンで USB メモリが認識されていることを確認します。

通常は「USB」メニューのデバイスフィルターに表示されていれば問題ないはずです。ただ仮想マシン起動後に USB メモリを母艦に接続した場合、一度仮想マシンの再起動が必要かもしれません。



起動可能な USB メモリを作成するため、まずは USB メモリのパスを特定します。

```
$ fx list-usb-disks
$ /dev/sdb
```

ビルドしたイメージ用の Zedboot USB ドライブを作成します。

```
$ fx mkzedboot /dev/sdb
```

以下のようなログが表示されます。

```
Creating disk image...
/dev/sdb - Flash Disk
INFO: Changing ownership of /dev/sdb to makoyuki
[sudo] password for makoyuki:
INFO: Opening device...
INFO: Create new GPT partition table...
WARNING: Primary GPT header is invalid
WARNING: Secondary GPT header is invalid
20676E69-7973-7473-656D-000000637B9A
INFO: done
INFO: Create new partitions...
INFO: done
INFO: Writing zedboot for EFI
56720+0 records in
56720+0 records out
29040640 bytes (29 MB, 28 MiB) copied, 133.559 s, 217 kB/s
INFO: done
INFO: Closing device.
done
```

「done」と表示されて正常終了が確認できたら、Intel NUC での Boot を行います。

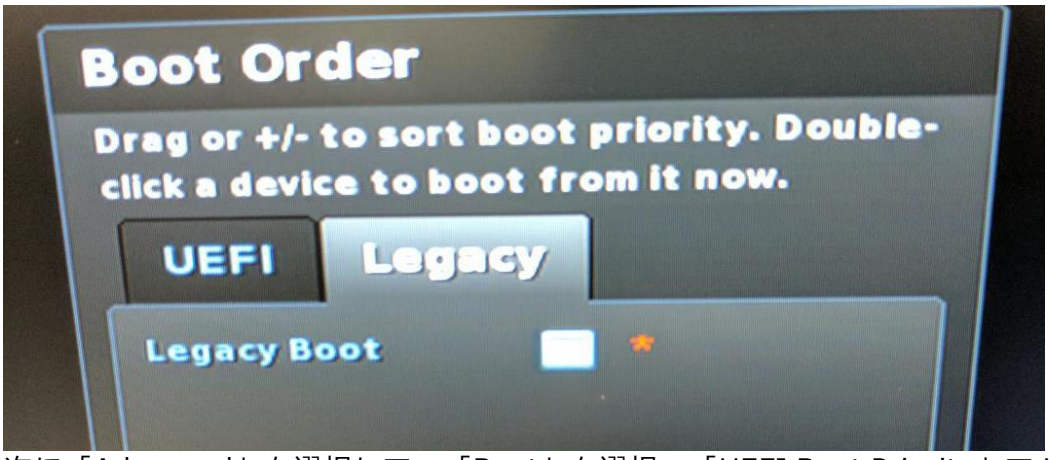
INTEL NUC を USB メモリで起動

Intel NUC を先ほど作成した USB メモリで起動します。

Intel NUC に USB メモリを接続して電源を入れます。

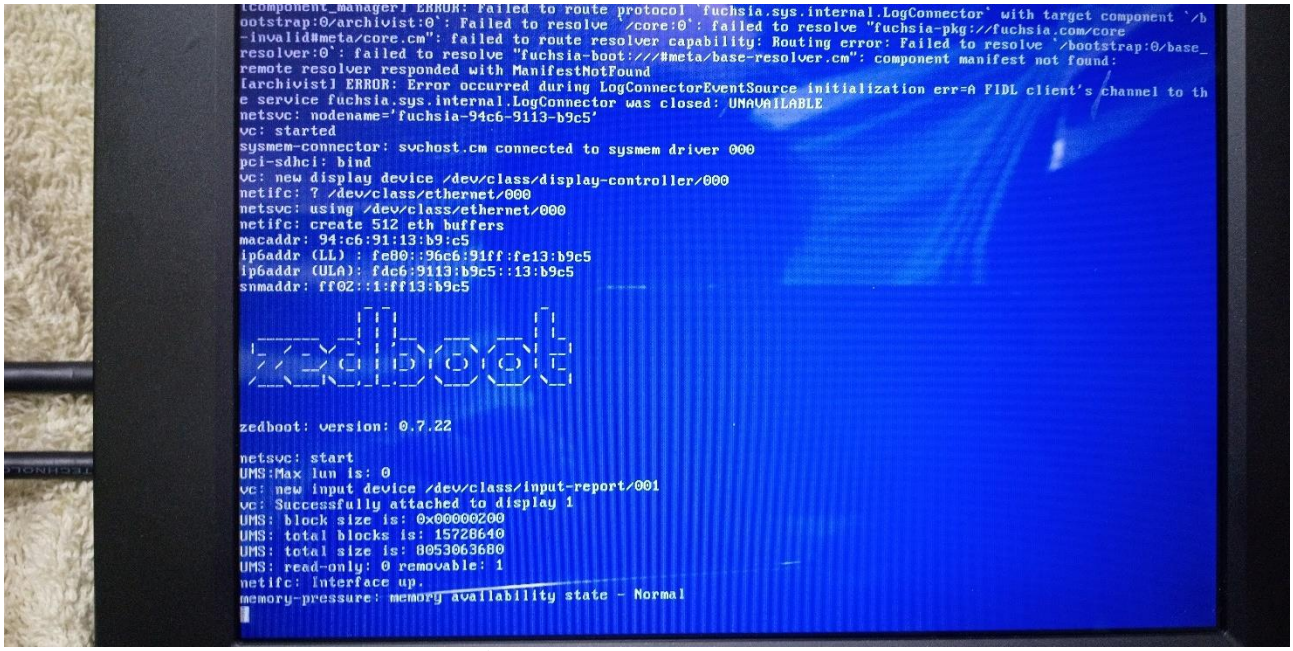
起動作業の前に、INTEL NUC 側の設定をします。起動時に F2 を押下して「Visual BIOS」画面を開きます。

「Legacy」タブをクリックして「Legacy Boot」のチェックを外します。

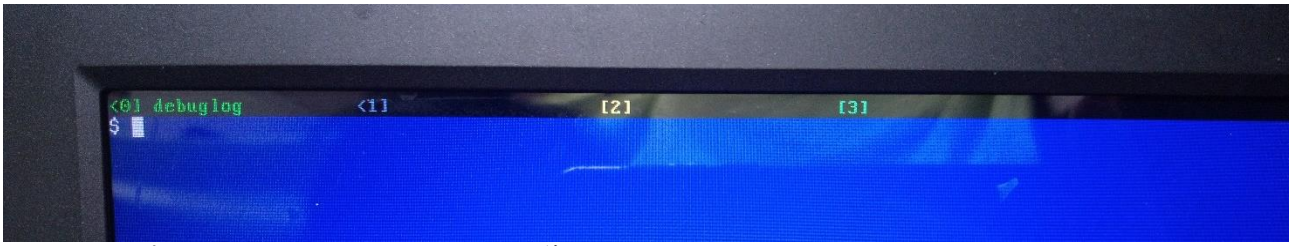


次に「Advanced」を選択して、「Boot」を選択、「UEFI Boot Priority」で USB メモリの起動順序を最優先に変更します。

この状態で、設定を保存し、再起動します。

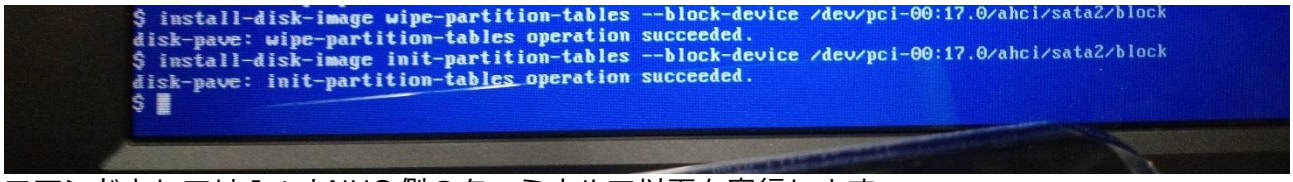


USB メモリから起動し。zedboot が正常に起動し、上記のような画面になります。



Alt+F3 を押下し、ターミナル画面に遷移します。

zedboot から見た SSD のブロックデバイス名を「lsblk」コマンドで表示させて、パーティションをいったん削除/新規にパーティションを作成し、そこにイメージをインストールします。



コマンドとしては Intel NUC 側のターミナルで以下を実行します。

```
$ lsblk
$ install-disk-image wipe-partition-tables --block-device /dev/pci-00:17.0/ahci/sata2/block
$ install-disk-image init-partition-tables --block-device /dev/pci-00:17.0/ahci/sata2/block
```

これでパーティションが作成されましたので、インストールを行います。

今度は、fuchsia の開発環境を構築したマシンに戻り、ターゲットマシンに Fuchsia をインストールします。

```
$ fx pave
```

それなりにインストールの進捗が表示されて、以下のように表示されればインストールは完了です。USB メモリは抜いてしまって問題ありません。

```
 [bootserver] Transfer ends successfully.
[bootserver] Issued boot command to [fe80::96c6:91ff:fe13:b9c5]:33330
```

ちょっとわかりづらいので簡単にまとめると

・開発環境でイメージ作成

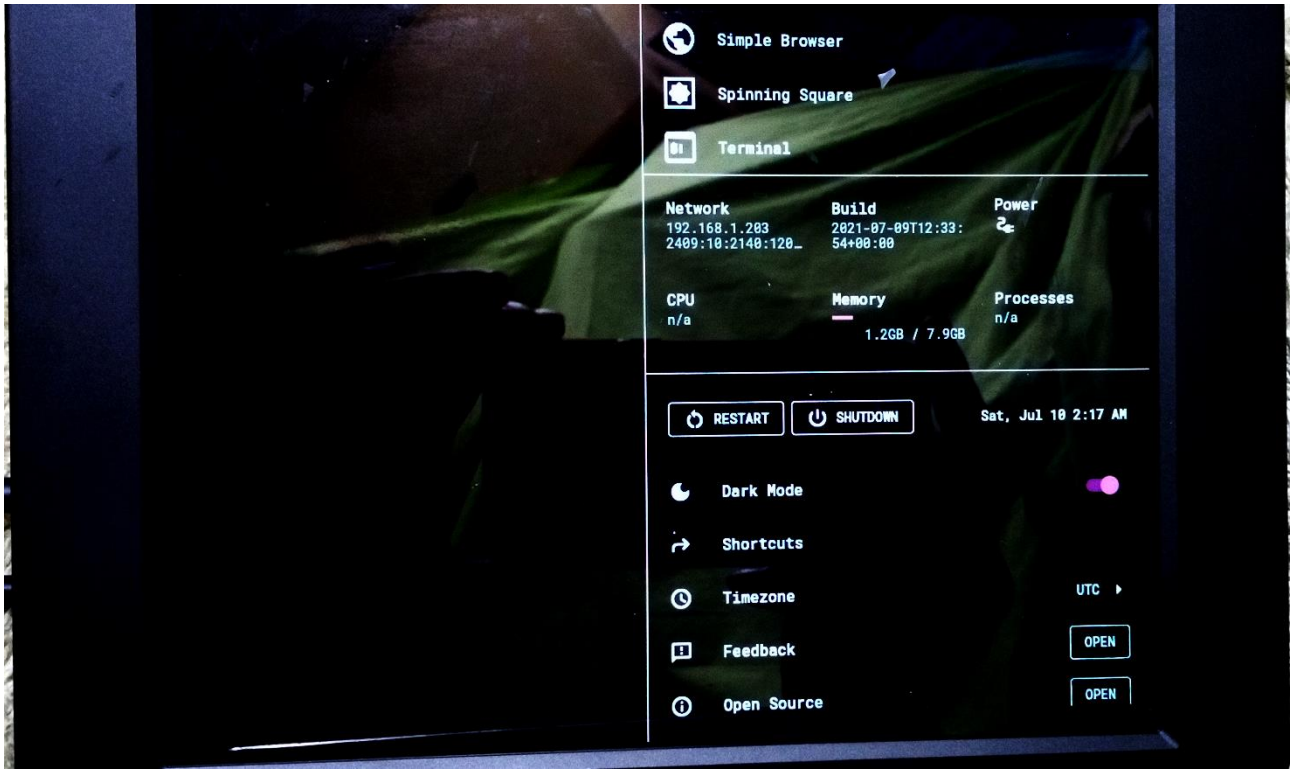
・起動用 USB メモリを作成

・デバイスを USB メモリで起動

・起動したデバイスに、開発環境からイメージを転送

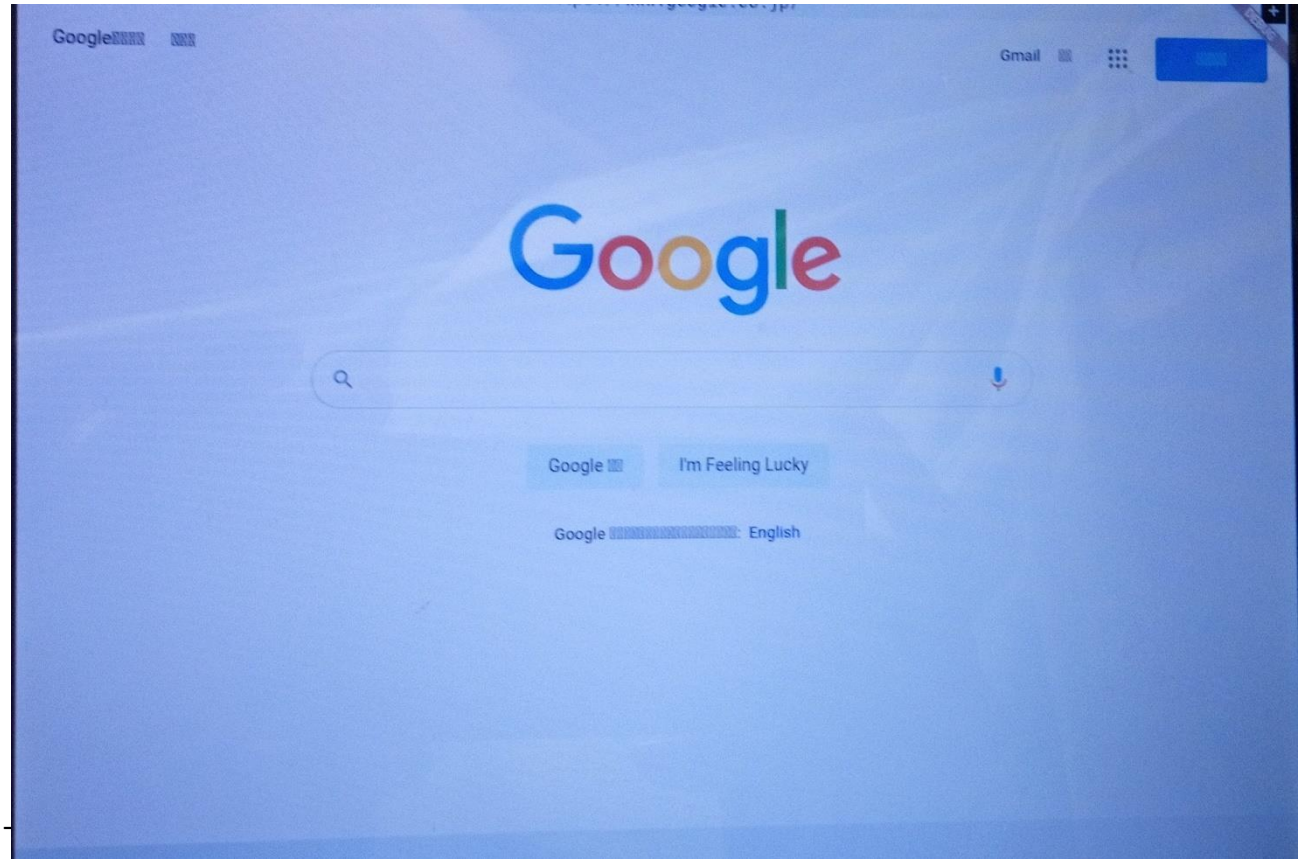
ということをやっています。

起動画面は以下のような感じです。ちょっと映り込みが見えるかもしれませんがご容赦



「Timezone」が UTC になっていますが、これはクリック、もしくはキーボードで選択して「Asia/Tokyo」を選択することで変更できます。

アプリケーションの類があまり無いようですが、一応「Simple Browser」を選択することでインターネットに接続できます。



ああフォントが無い、、、、

というわけで日本語フォントが今のところ準備できておらず文字化けしております。フォント自体は準備されているはずなので、将来的には「日本語フォント追加⇒Google 日本語入力」

でなんとかなったりしないですかね。。。

おわりに

というわけで、FuchsiaOS を実機に導入する記事でした。

キーボード、マウスの認識は問題なく、操作もできましたが、マウスに関しては操作が効かなくなることがしばしばありました。

また、現状は日本語フォントが入らずブラウザで文字化けが見られたりともう一歩なところもあります。

そろそろ実際のデバイスにも搭載され始めているので、どこまで素人の遊びが通用するかわかりませんが、注視はしていきたいと思います。

なお、一応オフィシャルサイトにある Hello World のテストとかもやったんですが、結果が超絶地味なので掲載してません。もう少しビジュアル的にイケてるテストやデモがありそうであれば、またやってみたいと思います。

マイナーOS駆け足レビュー

オマケです。

今回はレビューというほどではないんですが、Fuchsia と Linux のハイブリッドみたいな OS がリリースされているとのことなので、それだけちょっと試してみます。なので毎回書いてる前置きは今回は省略。

dahliaOS

この OS は？

基本的には Linux の 1 ディストリビューションです。
Pangolin デスクトップという独自の GUI を備えています。Flutter を用いてアプリケーションが書かれており、dahliaOS 上の仮想マシンとコンテナーを管理するツールがあり、これにより既存の OS の古いアプリケーションをそのまま使ったり、Flutter アプリケーションを Fuchsia 用にポーティングするための環境もあったりするようです。

メインサイト

https://dahliaos.io/

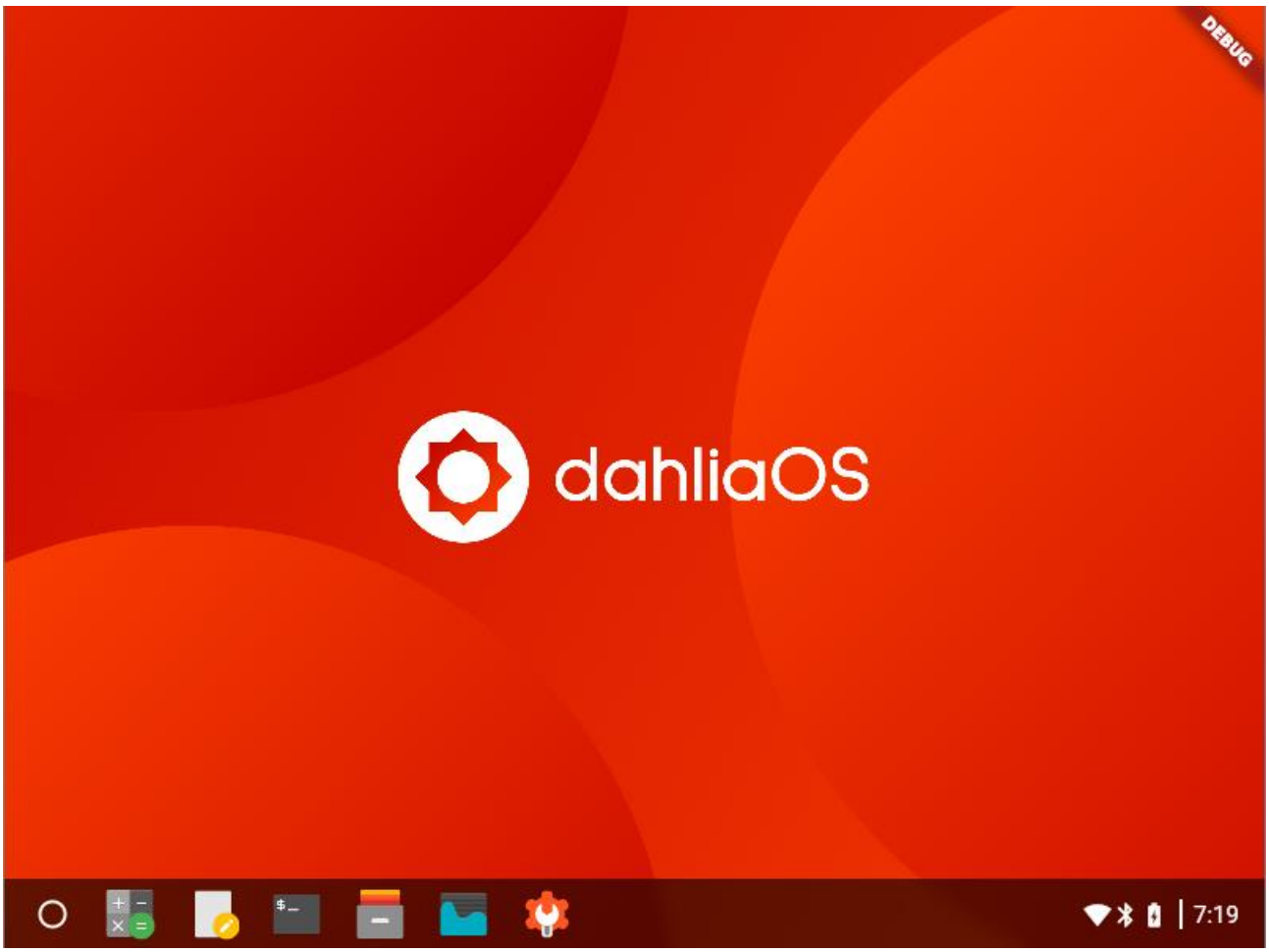
ダウンロードとインストール

https://dahliaos.io/downloads

ISO イメージが提供されているので、こちらをダウンロードしてインストールします。

仮想マシンの場合、最低 2GB のメモリで良いようです。

一応仮想ディスクも設定しますが、実際にはライブ動作しているようで、HDD へのインストール等はありません。
以下は Linux 部分が起動した直後のスプラッシュスクリーンです。



起動後の画面です。

右上の「DEBUG」は Fuchsia OS をデバッグモードでビルドした際に表示されるものと同じですね。

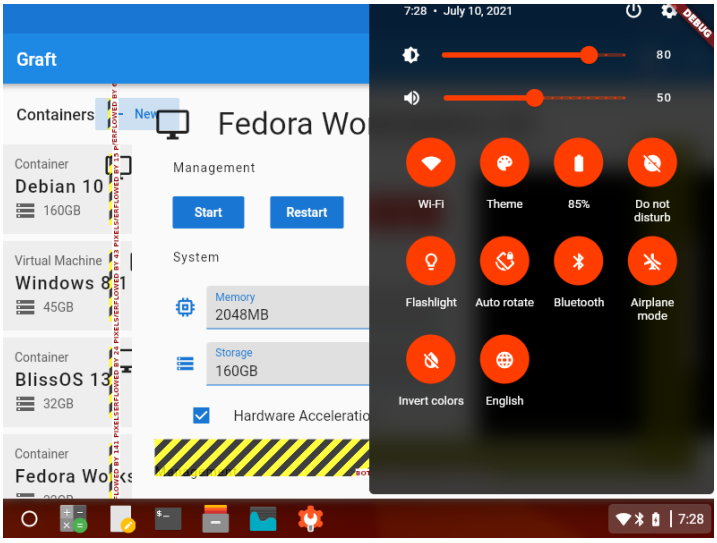
見た目も良く動作も軽快ですが、アプリケーションの類はまだ十分に(デモ版のようなリリースなので)実装されていないようで、電卓やテキストエディタ、コンソール等が使えるのみです。

ウリである Gtaft という、仮想マシン、コンテナ管理のアプリケーションも現状では動作せずでも版のような扱いのようです。

正直に言うと現状で Fuchsia OS らしさがよくわからないのですが、Flutter で実装しているので、GUI(Pangolin Desktop)はディストリビューションを選ばず使用可能という話なので、まだまだこれからの OS なのかな、と思います。

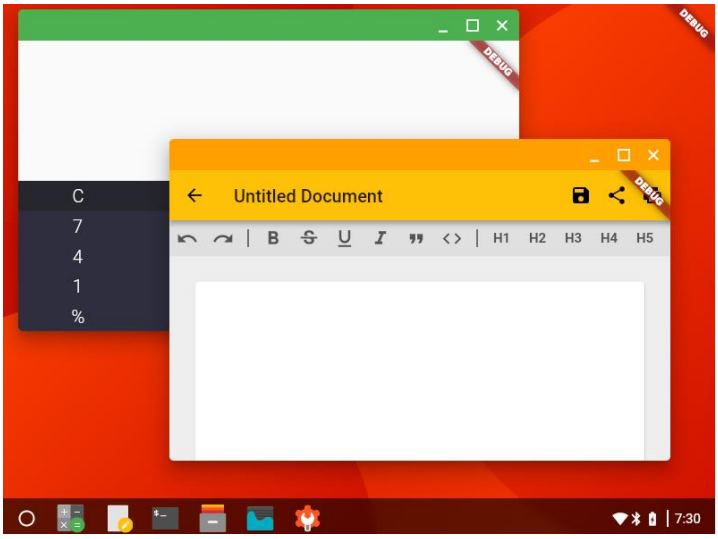
あとは当方が開発畑でないので、FImage を使っての Fuchsia への Fluttre アプリのポーティングというのがどのくらいの有用性なのかがわかっていないというのもありますが。

次ページ以降に簡単にスクリーンショットを載せておきます。



Graft は起動させてもデモ画面的に表示されているだけで実際に仮想マシンが動いているわけではありません。

画面右は右下をクリックした際に表示される設定パネルですが、こちらもすべてが動作するというわけではありません。



よくある電卓とテキストエディタです。

細かなアプリケーションに見える部分はほかにもありますが

「pre-Build リリースなんで実装されてないもんもいっぱいあるでよ」

のようなコメントが多く記載されているので、本当にこれからの OS ですね。

Fuchsia とのハイブリッド、あるいは既存 OS、レガシーアプリケーションの実行が同時に可能な環境となれば非常に興味深いですが、売りの Gtaft、Fimage あたりを触れたらまたトライしてみたいと思います。

おわりに

正直、こんなに薄い本になってしまうとは想定外でした。

相当早い段階で Intel NUC まで準備していたんですが、その後仕事が忙しくなってしまいなかなか時間が原稿の作成に割けなくなり、4 月と 6 月に体調を崩すなどして余計に進捗が出ませんでした。

Fuchsia を実機で試す、という本当に最低限度のことしかできていませんが、テストやデモ的なアプリケーションなどももう少しあるようですし、もうちょっと何かできたんではないかと言う気もします。

また、本来は OSS ネタももう少し進捗があるはずだったんですが、せいぜい子供のための Minecraft サーバを構築したくらいで、こちらも技術的な記事は薄目なので、今回はこれが精いっぱいでした。(もう少し凝ったことをしたら原稿にするかもしれませんが。)

ただ、これまでマイナーOS を多数レビューするという方向性でしたが、少し前あたりから書いていますように、いくつか主要なマイナーOS(というのも微妙な表現ですが)について継続的にウォッチするような方向と、OSS 関連の各種構築経験やインフラ寄りのものになっていくのかなぁというところです。

Fnow　マイナーOS関連　活動紹介

活動コミュニティ

いまだコンテンツが豊富とは言い難いですが当方のマイナーOSネタサイトは以下になります。

Far Northern Other World　マイナーOS コンテンツ
https://www.fnow.org/

引き続き忙殺の中、サイト更新までなかなか手が回らない状況です。

なお､各 SNS 等にもコミュニティを作っていましたが最近活動が有名無実なので整理する予定です。

なお、過去の Operating System Maniacs(2nd シーズンより前)のものはすべて上記サイトよりダウンロード可能になっておりますので、是非見てみていただければと思います。

参加イベント等

コロナ禍の中、開催が延期されていますが、基本的には「コミックマーケット」および「技術書典」へ参加しています。

今後も基本的にはこれらのイベントへ参加予定です。

今後の目標

(どこまでできるかとか、以前に書いたことと重複してるとかはありますがまぁ置いといて)

■エミュレータ関連

PDP-11、IRIX など、異種アーキテクチャや Old アーキテクチャなどのエミュレータです。

当然ながらその上で各種古い OS を動かしてみたりしたいなぁと以前から考えています。

■異種アーキテクチャ系の何か

以前も各種商用 Unix 特集でも取り扱いましたが、せっかく Alpha マシン、PA-RISC マシン、Sparc マシン等を所持しているので、それらで何かできないか考えています。

■es Operating System

以前からずっと試したいとおもってはや何年。。。
任天堂からリリースされた OS です。最終リリースはかなり古いですが開発者の方が任天堂⇒Google⇒会社設立し引き続きいろいろやられているようなので、是非触ってみたいですが、とてもハードルが高いので躊躇したままここまで来てしまいました。
次回技術書典開催の際には是非取り扱いたいと思います。

これら以外にもマイナーOS についての情報や気になる OS、持っている OS、古い異種アーキテクチャマシンなどの情報をお待ちしております。

編集後記

というわけで「おわりに」にだいたい書いてしまいましたが、一応原稿作成時に思ったことを。

以前、MorphOS を取り扱った時にも思ったのですが、やはり HDMI－USB3 で取り込めるビデオキャプチャが欲しいなぁ、というのが 1 つですね。

ハードウェアに直接インストールした場合の画面をスマホやデジカメで撮影して原稿に使うというのもクォリティもあまり高くないですし、夏ボーで買うか。。。。

あとは PC の新調についてはここ数年悩んでいます。

10 年くらい前のモデルではあるのですが、16CPU、192GB のマシンなので、マシンパワー的には現状でもそれほど不満は無かったりするのです、、とはいえ(CPU やボードのアーキテクチャなんかは新しいものになるとは思いますが)CPU 数やメモリを搭載したものを購入しようとなるとまた高額になるので悩ましいです。

またここから数年ヤフオク眺めながら悩むことにしますか。。。

それではまた。

————————————————————————————

奥付

Operating System Maniacs 2nd Version4.0(Free)

発行：Fnow「Far Northern Other World」

〒107-0052

東京都港区赤坂 1-4-4　冨士野ビル 501

佐藤誠之(辻瀬蒼伊)

2021/07/10 発行　(技術書典 11)

————————————————————————————

Operating System Maniacs (2nd Seasons)Version 4.0

Far Northern Other World