Armadillo-600 シリーズ Armadillo Base OS 移行ガイド

A6400-U00Z
A6400-D00Z
A6400-B00Z
A6400-N00Z
A6100-U00Z
A6100-D00Z

Version 1.0.0
2023/06/29

株式会社アットマークテクノ [https://www.atmark-techno.com]

Armadillo サイト [https://armadillo.atmark-techno.com]

Armadillo-600 シリーズ Armadillo Base OS 移行ガイド

株式会社アットマークテクノ

Version 1.0.0
2023/06/29

目次

Armadillo-600 シリーズ Armadillo Base OS 移行ガイド

表目次

1. はじめに

1.1. 本書の対象読者と目的

本文書は、 Armadillo-600 シリーズで Debian GNU/Linux 10(コードネーム buster)もしくは 9(コードネーム stretch)(以下、Debian)をご使用頂いている方が、新しく Armadillo Base OS を採用いただく場合に、スムーズに Armadillo Base OS での開発に移行していただけるようにガイドすることを目的としています。

OP-A600-AWLMOD-20 または OP-A600-BTTHMOD-21 を使用する場合は、 Armadillo Base OS への移行が必須です。

各オプションモジュールと OS の対応については「3.2. オプションモジュールの対応」を参照してください。

1.1.1. 対象読者

- Armadillo-600 シリーズで、以前から Debian を利用していた方

1.1.2. 記載内容

- Armadillo Base OS の特徴
- Armadillo Base OS に移行する際の注意点
- Armadillo Base OS のインストール方法
- Armadillo Base OS の基本操作
- Armadillo Base OS への開発済みシステムの移行方法

1.2. 対応ドキュメント

Armadillo Base OS に対応した Armadillo-600 シリーズのドキュメントは「表 1.1. Armadillo Base OS 対応ドキュメント」に示す通りです。本文書と合わせてご参照ください。

表 1.1 Armadillo Base OS 対応ドキュメント

製品	対象ドキュメント
Armadillo-640	Armadillo-640 製品マニュアル Armadillo Base OS 対応
Armadillo-610	Armadillo-610 製品マニュアル Armadillo Base OS 対応

2. Armadillo Base OS の特徴

Armadillo Base OS はアットマークテクノが提供する、軽量な Alpine Linux をベースとした Armadillo 専用ディストリビューションです。

Armadillo Base OS に搭載されている機能の一部として、以下が挙げられます。

- 安全なローカル/リモートアップデート機能
- アプリケーションをコンテナベースで開発することによる高いセキュリティ機能
- 堅牢性の高いファイルシステム
- Visual Studio Code にインストールできる専用の拡張機能による開発の簡易化

Armadillo Base OS の詳細は以下のページも合わせてご参照ください。

Armadillo サイト - Armadillo Base OS

https://armadillo.atmark-techno.com/guide/armadillo-base-os

3. Armadillo Base OS に移行する際の注意点

3.1. eMMC のパーティション構成の違い

Armadillo Base OS と Debian では eMMC のパーティション構成に違いがあります。

Armadillo Base OS のパーティション構成を「表 3.1. Armadillo Base OS インストール時の eMMC パーティション構成」に、 Debian のパーティション構成を「表 3.2. Debian インストール時の eMMC パーティション構成」に示します。

表 3.1 Armadillo Base OS インストール時の eMMC パーティション構成

パーティション	サイズ	ラベル	説明
1	300MiB	rootfs_0	A/B アップデートの A 面パーティション(Linux カーネルイメージ, Device Tree Blob, Alpine Linux rootfs を含む)
2	300MiB	rootfs_1	A/B アップデートの B 面パーティション(Linux カーネルイメージ, Device Tree Blob, Alpine Linux rootfs を含む)
3	50MiB	logs	ログ書き込み用パーティション
4	200MiB	firm	ファームウェア用パーティション
5	2.7GiB	app	アプリケーション用パーティション

表 3.2 Debian インストール時の eMMC パーティション構成

パーティション	サイズ	ラベル	説明
1	30.6MB	-	予約領域
2	3.4GB	-	Linux カーネルイメージ, Device Tree Blob, Debian
3	122.1MB	-	予約領域

3.2. オプションモジュールの対応

Armadillo Base OS と Debian では対応するオプションモジュールに違いがあります。

それぞれが対応しているオプションモジュールを「表 3.3. Armadillo Base OS と Debian がサポートするオプションモジュール」に示します。

表 3.3 Armadillo Base OS と Debian がサポートするオプションモジュール

オプションモジュール	型番	Armadillo Base OS	Debian
LCD オプションセット(7 インチタッチパネル WVGA 液晶)	OP-LCD70EXT-L00	○	○
RTC オプションモジュール	OP-A600-RTCMOD-00	○	○
WLAN オプションモジュール	OP-A600-AWLMOD-00	×	○
WLAN コンボオプションモジュール	OP-A600-AWLMOD-20	○	×
BT/TH オプションモジュール	OP-A600-BTTHMOD-00, OP-A600-BTTHMOD-20	○	○
BT/TH オプションモジュール(WLAN 対応)	OP-A600-BTTHMOD-01	×	○
BT/TH オプションモジュール(WLAN コンボ対応)	OP-A600-BTTHMOD-21	○	×

3.3. ブートローダのバージョンの違い

Armadillo Base OS と Debian では ブートローダのバージョンに違いがあります。

それぞれのブートローダのバージョンを「表 3.4. Armadillo Base OS と Debian のブートローダのバージョンの違い」に示します。

表 3.4 Armadillo Base OS と Debian のブートローダのバージョンの違い

Armadillo の OS	対応するブートローダバージョン
Armadillo Base OS	2020.04
Debian 9 及び 10	2018.03

3.4. Linux カーネルのバージョンの違い

Armadillo Base OS と Debian では Linux カーネルのバージョンに違いがあります。

それぞれの Linux カーネルのバージョンを「表 3.5. Armadillo Base OS と Debian の Linux カーネルのバージョンの違い」に示します。

表 3.5 Armadillo Base OS と Debian の Linux カーネルのバージョンの違い

Armadillo の OS	対応するカーネルバージョン
Armadillo Base OS	5.10
Debian 9 及び 10	4.14

3.5. ATDE のバージョンの違い

Armadillo Base OS と Debian では ATDE のバージョンに違いがあります。ATDE についての詳細は「ATDE について [https://armadillo.atmark-techno.com/guide/atde]」を参照してください。

それぞれの ATDE のバージョンを「表 3.6. Armadillo Base OS と Debian の ATDE のバージョンの違い」に示します。

表 3.6 Armadillo Base OS と Debian の ATDE のバージョンの違い

Armadillo の OS	対応する ATDE バージョン
Armadillo Base OS	ATDE9(Debian 11 bullseye ベース)
Debian 10	ATDE8(Debian 10 buster ベース)
Debian 9	ATDE7(Debian 9 stretch ベース)

4. Armadillo Base OS のインストール方法

インストールディスクを使用して Armadillo-600 シリーズに Armadillo Base OS をインストールします。

4.1. 必要なもの

- Armadillo Base OS をインストールする Armadillo-600 シリーズ本体
- 作業用 PC
  - Linux もしくは Windows OS
  - カードリーダなどを介して microSD カードを読み込めるもの
  - インターネットに接続できるもの
- microSD カード
  - 4GB 以上のもの
- (必要ならば)microSD カードリーダ

4.2. インストールディスクイメージを microSD カードに書き込む

作業用 PC の任意の場所に Armadillo-600 シリーズ対応 Armadillo Base OS のインストールディスクイメージをダウンロードしてください。インストールディスクイメージの最新版は以下からダウンロードできます。

表 4.1 Armadillo Base OS インストールディスクダウンロード URL

製品	対象ドキュメント
Armadillo-640	https://download.atmark-techno.com/armadillo-640/image/baseos-600-installer-latest.zip
Armadillo-610	https://download.atmark-techno.com/armadillo-640/image/baseos-600-installer-latest.zip

Armadillo-600 シリーズ対応 Armadillo Base OS のインストールディスクイメージは、 Armadillo-640 と Armadillo-610 のそれぞれのダウンロードページからダウンロード出来ますが、ファイルの中身は共通で同一のものです。

microSD カードを作業用 PC に接続し、ダウンロードしたインストールディスクイメージを microSD カードに書き込みます。作業用 PC の OS によって手順が異なるので注意してください。

4.2.1. Linux の場合

以下、作業用 PC に接続した microSD カードが /dev/sda として認識されている前提で説明します。お使いの環境によって変化するので適宜読み替えてください。

```
[PC ~]$ sudo fdisk -l /dev/sda
ディスク /dev/sda: 29.72 GiB, 31914983424 バイト, 62333952 セクタ
Disk model: SD/microSD
単位: セクタ (1 * 512 = 512 バイト)
セクタサイズ (論理 / 物理): 512 バイト / 512 バイト
I/O サイズ (最小 / 推奨): 512 バイト / 512 バイト
ディスクラベルのタイプ: gpt
ディスク識別子: 72908E7E-2AF1-4289-A9AE-E1B0048AB5C6

デバイス    開始位置 最後から    セクタ サイズ タイプ
/dev/sda1       2048 62333918 62331871  29.7G Linux ファイルシステム
```

ダウンロードしたインストールディスクイメージは、zip 形式で圧縮されています。以下のコマンドを実行し、 圧縮されたインストールディスクイメージを展開し、microSD カードに書き込みます。以下の実行例は Armadillo-640 の場合です。VERSION は適宜読み替えてください。

```
[PC ~]$ unzip baseos-600-installer-latest.zip
[PC ~]$ sudo dd if=./baseos-600-installer-[VERSION].img of=/dev/sda bs=1M oflag=direct
status=progress
```

以上で、インストールディスクイメージの microSD カードへの書き込みは完了です。

4.2.2. Windows の場合

以下のページを参考に、「Win32 Disk Imager」を用いて microSD カードにインストールディスクイメージを書き込んでください。

Windows 上でのインストールディスクの作成方法 [https://armadillo.atmark-techno.com/blog/1913/2400]

4.3. Armadillo のバックアップを取る

インストールを実行すると、 Debian が書き込まれている Armadillo の eMMC 内のデータは消去されます。インストール前に必要なファイルについてはバックアップを取っておいてください。

dump_rootfs を使用すると開発済みのユーザーランドをまとめて .tar.gz アーカイブファイルにまとめることができますので、ご活用ください。

dump_rootfs については以下を参照してください。「Armadillo 標準ガイド Armadillo 入門編 dump_rootfs による方法 [https://manual.atmark-techno.com/armadillo-guide-std/armadillo-guide-std-getting-started_ja-1.2.1/ch08.html#sec_make_debian_rootfs_by_dump_rootfs]」

4.4. インストールを実行する

各製品マニュアルの 「インストールディスクを使用する」を参照し、インストールを実行してください。

正しくインストールが完了した場合、JP1 をオープンにして再度電源を投入すると、 Armadillo Base OS に書き換わっています。

正しくインストールされない場合は、インストールディスクイメージのダウンロードからやり直してみてください。

やり直しても上手く行かない場合は、「Armadillo フォーラム [https://armadillo.atmark-techno.com/forum/armadillo/]」にてインストール時のログを記載の上ご質問ください。

5. Armadillo Base OS の基本操作

Armadillo Base OS に移行が完了したので、実際に動かしてみながら従来の Debian がインストールされた Armadillo-600 シリーズと使い方を比べてみましょう。

5.1. ログインする

電源投入後にしばらくすると、ログインプロンプトが出てきます。

```
Welcome to Alpine Linux 3.17
Kernel 5.10.180-0-at on an armv7l (/dev/ttymxc0)

armadillo login:
```

デフォルトでは Debian と同じくユーザー名、パスワードともに `root` でログインできます。正しくログインが完了するとコマンドプロンプトが表示されます。

```
armadillo:~#
```

5.2. Armadillo Base OS 上でファイルを保存(永続化)する

Armadillo Base OS 上のファイルは変更しただけでは次回再起動時に失われていまいます。これは、Armadillo Base OS の rootfs が overlayfs であるためです。

変更したファイルを永続化するには、変更したファイルに対して `persist_file` コマンドを実行する必要があります。

```
armadillo:~# persist_file -v hoge
'/root/hoge' -> '/mnt/root/hoge'
```

これにより `hoge` は、 Armadillo 再起動後も保持されます。

5.3. コンテナを立ち上げる

このままの状態でも Debian と同様に任意のファイルを配置したり、コマンドを実行したりできますが、 Armadillo Base OS ではユーザーの開発する対象はコンテナ内の環境です。

コンテナを管理するためのツールとして有名なものといえば Docker が挙げられますが、 Armadillo Base OS では Docker と同じく OCI 規格に準拠した Podman というソフトウェアを使用します。Podman では、 `docker` コマンドと基本的に互換性のある `podman` コマンドを使用してコンテナを操作します。

ここでは、 `podman` コマンドを使用して Debian のコンテナを作成し、その中で開発を行います。

5.3.1. Podman のデータの保存先を eMMC に変更する

コンテナ内は仮想的な Linux マシンとして利用できますが、不意の電源の遮断等でデータの破損を防いだり、 eMMC の寿命消費をおさえるため、基本的にコンテナ内での変更はメモリ上に書き込まれ、電

源を切ると消えてしまいます。あえて永続的に保存したいデータがある場合は特定のディレクトリを指定しておきます。この仕様については製品マニュアルの「Podman のデータを eMMC に保存する」及び、「コンテナの変更を保存する」を参照してください。

一方で、この機構は長期運用時には役に立ちますが、開発の初期にはソフトウェアの追加やツールの配置を試行錯誤することが多くあるため不便だと思います。ここでは、一旦メモリ上に書き込む機能を停止して直接 eMMC に変更を書き込む状態に変更します。

本書の後に、製品マニュアルを参照することで最終的にメモリ上に書き込む状態へ再び移行することで、長期間安定して動作するシステムを構築することが可能です。

```
armadillo:~# abos-ctrl podman-storage --disk ❶
Creating configuration for persistent container storage
Create a snapshot of '/var/lib/containers/storage_readonly' in '/mnt/containers_storage'
Successfully switched podman-storage to disk

armadillo:~# podman pull docker.io/debian:bullseye ❷
Trying to pull docker.io/library/debian:bullseye...
Getting image source signatures
Writing manifest to image destination
Storing signatures
d352b92ddb5df3919bb910c3e7f46b1cbea5a6072f96eb837d7147a3c3f84d39
```

❶ Podman のデータの保存先を tmpfs から eMMC に変更する

❷ Debian 11 (bullseye)のコンテナイメージを取得する

5.3.2. コンテナ起動用 conf ファイルの作成

vi コマンドなどで/etc/atmark/containers/sample_container.conf ファイルを作成し、再起動後も消えないように persist_file します。

```
armadillo:~# cat /etc/atmark/containers/sample_container.conf
set_image docker.io/debian:bullseye ❶
set_command sleep infinity ❷
armadillo:~# persist_file -v /etc/atmark/containers/sample_container.conf
'/etc/atmark/containers/sample_container.conf' -> '/mnt/etc/atmark/containers/
sample_container.conf'
```

❶ ベースとなるコンテナイメージとして、先程 podman pull した Debian 11 のイメージを指定します

❷ コンテナ起動時に実行するコマンドを指定します。ここでは sleep infinity です

5.3.3. コンテナを起動する

podman_start コマンドで conf ファイルの設定どおりにコンテナが起動します。

```
armadillo:~# podman_start sample_container
Starting 'sample_container'
dfc9660d3ff974bd9c4e45d0cf512b96c9bff34975f21809da199b300343dfeb
```

```
armadillo:~# podman ps
CONTAINER ID  IMAGE                                 COMMAND         CREATED         STATUS                  ⏎
PORTS       NAMES
dfc9660d3ff9  docker.io/library/debian:bullseye  sleep infinity  12 seconds ago  Up 13 seconds           ⏎
ago             sample_container
```

5.3.4. コンテナ内に入る

以下のコマンドを実行して、作成したコンテナ内に入ります。

```
armadillo:~# podman exec -it sample_container bash

root@dfc9660d3ff9:/# cat /etc/debian_version ❶
11.7
```

❶ cat コマンドで Debian のバージョンを表示しています

Armadillo Base OS では、このように作成したコンテナの中で開発を行っていただきます。

上記の手順で作成したコンテナは Debian 環境ですので、ほとんど従来の Debian がインストールされた Armadillo-600 シリーズと同じように開発することができます。

もし既に Debian がインストールされた Armadillo-600 シリーズでソフトウェアを開発されていた場合でも、 このコンテナ内や ATDE9 などの bullseye 環境でコンパイルし直してこのコンテナ内に配置すると実行できるはずです。必要なライブラリがある場合は今までどおり apt install コマンドでインストールできます。

5.4. コンテナを保存する

コンテナの中で exit コマンドを実行することで、コンテナから抜けて Armadillo Base OS に戻ることができます。

```
root@dfc9660d3ff9:/# exit
exit
armadillo:~#
```

この状態で、作成したコンテナを podman commit コマンドでコンテナイメージとして保存できます。

```
armadillo:~# podman commit sample_container sample_container_image:latest
```

これで再起動後もコンテナイメージを保持し続けます。

5.5. Armadillo の起動時にコンテナを自動起動する

「5.3.2. コンテナ起動用 conf ファイルの作成」のように /etc/atmark/containers/ の下に.conf ファイルを配置することで、 Armadillo の起動時に自動的にコンテナを起動できます。これにより、従来の Debian での /etc/rc.local や systemd でのアプリケーションの自動実行と同等のことを実現できます。

sample_container.conf を以下のように編集し、 persist_file コマンドで永続化することで、Armadillo 起動時に「5.4. コンテナを保存する」で保存したコンテナイメージからコンテナを起動し、コンテナ内の sample.sh を実行します。

```
armadillo:~# cat /etc/atmark/containers/sample_container.conf
set_image localhost/sample_container_image:latest
set_command sh sample.sh
```

5.6. Armadillo Base OS で開発を進める

ここまでは Armadillo Base OS での開発のスタートラインに立つまでの内容です。これ以降の本格的な開発の手順については、 Armadillo Base OS に対応した製品マニュアルを参照してください。

Armadillo Base OS で開発を進めていくにあたって不明な点がございましたら、「Armadillo フォーラム [https://armadillo.atmark-techno.com/forum/armadillo/]」にてご質問ください。

6. Debian で開発済みのシステムを Armadillo Base OS に移行する

既に Debian 搭載の Armadillo でシステムを開発しており、そのシステムを Armadillo Base OS 環境に移行する方向けに手順を紹介します。

本書では移行時によくあるケースについて説明します。説明しきれない部分もありますので、製品マニュアルの対応する箇所も合わせて参照してください。

以下の手順は、「5. Armadillo Base OS の基本操作」の手順を一度実行している前提で進めます。一度も実行していない場合は、「5. Armadillo Base OS の基本操作」の手順を実行してからこの先に進んでください。

ここでは Debian で開発済である以下のようなシステムを Armadillo Base OS に移行することを例として説明します。

- アプリケーションを Armadillo 起動時に自動的に app.py という名前の Python スクリプトを実行する
- app.py は USB メモリに読み書きアクセスする

6.1. Debian コンテナを起動する

任意の名前で Debian コンテナを作成します。以下の手順では、 user_app という名前の Debian コンテナを作成します。

/etc/atmark/containers/に user_app.conf というファイルを作成し、中身を以下のように編集します。

```
set_image docker.io/debian:bullseye
add_args --privileged
set_command sleep infinity
```

上記の例では add_args --privileged を指定しています。これを指定することで、コンテナに全権限と全てのデバイスへのアクセス許可を与えます。開発したアプリケーションが使用する最低限の権限を洗い出す必要がないため、初めの動作確認には有用です。

ただし、本来コンテナには必要最低限の権限のみを与えるべきです。この add_args --privileged は、開発中のみのご利用に留めることを強く推奨します。

その後、 podman_start コマンドでコンテナを起動し、 podman ps コマンドで起動していることを確認します。

```
armadillo:~# podman_start user_app
Starting 'user_app'
9aaf31b7275798fc0c6d53bec3b47bbe9548b631b6c0845c482028a830c3148c

armadillo:~# podman ps
```

```
CONTAINER ID  IMAGE                                 COMMAND         CREATED         STATUS                  ⏎
PORTS       NAMES
9aaf31b72757  docker.io/library/debian:bullseye  sleep infinity  12 seconds ago  Up 13 seconds           ⏎
ago             user_app
```

6.2. コンテナ内にアプリケーションの実行環境を整える

「6.1. Debian コンテナを起動する」で作成・起動したコンテナ内で、開発したアプリケーションの実行環境を作成します。以下のコマンドを実行して、コンテナ内に入ります。

```
armadillo:~# podman exec -it user_app bash
root@dfc9660d3ff9:/# ❶
```

❶　ここはコンテナのルートディレクトリです

6.2.1. コンテナ内に必要なファイルを配置する

まず、アプリケーション本体やアプリケーションの設定ファイルなどをコンテナ内の任意の場所に配置しましょう。

/root/ に、 app.py を配置します。このコンテナは add_args --privileged が設定されているので、USB メモリを Armadillo に挿入していつも通り mount して使用することもできますし、ネットワーク経由でダウンロードすることも可能です。お好きな手順でコンテナ内にアプリケーションを配置してください。

```
root@dfc9660d3ff9:/# ls /root/app.py
/root/app.py ❶
```

❶　/root/app.py が存在することを確認しています

6.2.2. コンテナ内に必要なパッケージをインストールする

アプリケーションの実行のために特定のパッケージが必要な場合は. Debian コンテナなので apt install コマンドでインストールできます。

app.py は Python スクリプトなので、 python3 パッケージをインストールします。

```
root@dfc9660d3ff9:/# apt update && apt upgrade -y
root@dfc9660d3ff9:/# apt install python3
```

6.2.3. アプリケーションの動作確認をする

実行環境が整ったので、実際にアプリケーションを実行して期待したとおりの動作をすることを確認してください。

アプリケーションによってはネットワーク設定などが必要で、この段階では期待した動作をしない場合もあると思います。それらの必要な設定はこのあと行うので、ここでは最低でも必要なパッケージ類が足りていて、アプリケーションが実行できることを確認してください。

```
root@dfc9660d3ff9:/# python3 /root/app.py ❶
```

❶　パッケージの不足等なく、動作することを確認してください

6.2.4. コンテナの変更を保存する

コンテナ内で exit コマンドを実行することでコンテナから抜けて Armadillo Base OS に戻ります。

```
root@dfc9660d3ff9:/# exit
exit
armadillo:~#
```

ここで、コンテナの変更をコンテナイメージとして保存するために、 podman commit コマンドを実行します。以下の実行例では、 user_app_image という名前、 1.0.0 というタグ名で、 user_app コンテナを保存します。

```
armadillo:~# podman commit user_app user_app_container:1.0.0
```

保存したコンテナイメージは、 podman images コマンドを実行すると確認できます。

```
armadillo:~# podman images
REPOSITORY                    TAG         IMAGE ID      CREATED        SIZE
localhost/user_app_container  1.0.0       9f8fa6515340  3 seconds ago  123 MB
docker.io/library/debian      bullseye    d352b92ddb5d  3 weeks ago    123 MB
```

6.3. コンテナ外の設定を行う

コンテナ内にアプリケーションを配置しただけでは Debian で開発したシステムを再現できないケースが多いです。

アプリケーションの自動実行やネットワーク設定など、アプリケーション以外の設定項目も必要になりますが、これらはコンテナ内ではなく Armadillo Base OS 上で設定します。

6.3.1. コンテナに与える権限を変更する

この段階では user_app コンテナには add_args --privileged が指定されており、全てのリソースにアクセスできる状態になっているため、セキュリティ的に運用に向きません。必要最低限の権限に設定することを強く推奨します。

コンテナの権限は、/etc/atmark/containers 以下の conf ファイルによって制御できます。今回の例では、 user_app.conf を以下のように編集します。

```
set_image docker.io/debian:bullseye
add_hotplug usb ❶
set_command sleep infinity
```

❶　コンテナに、USB メモリをホットプラグで認識し、読み書きできる権限を与えます

conf ファイルに記載できる文言については、対応する製品マニュアルの「コンテナの運用」の箇所を参照してください。

6.3.2. アプリケーションを自動実行する

Debian の頃は systemd や rc.local によって自動実行していましたが、 Armadillo Base OS ではコンテナを Armadillo の起動時に自動的に生成、そのコンテナ中でコマンドを自動実行する仕様になっています。

具体的には、「6.1. Debian コンテナを起動する」で /etc/atmark/containers の下に、user_app.conf ファイルを作成して配置しましたが、このファイルがあることによってコンテナが自動的に起動されます。

今回の例では、 user_app.conf の以下の 2 点を変更します。

- 使用するコンテナイメージ(`set_image`)
- 実行するコマンド(`set_command`)

修正後の user_app.conf は以下の通りです。

```
set_image localhost/user_app_container:1.0.0 ❶
add_hotplug usb
set_command python3 /root/app.py ❷
```

❶　「6.2.4. コンテナの変更を保存する」で保存したコンテナイメージから起動するようにします

❷　コンテナ起動時に、 `python3 /root/app.py` が実行されるようにします

ここまで設定できたら、 user_app.conf 永続化するために、 `persist_file` コマンドを実行します。

```
armadillo:~# persist_file -v /etc/atmark/containers/user_app.conf
'/etc/atmark/containers/user_app.conf' -> '/mnt/etc/atmark/containers/user_app.conf'
```

これで Armadillo 起動時に user_app_container 内で app.py が自動実行されるようになりました。

6.3.3. Armadillo 本体の設定をする

今回の例では扱いませんが、場合によっては固定 IP アドレスを割り当てるためのネットワーク設定など、 Armadillo 本体の設定が必要な場合があります。

ユーザーアプリケーション以外の Armadillo 本体の設定については、基本的にコンテナの外で設定する場合がほとんどです。

例えばネットワークの設定は、 Debian 搭載 Armadillo-600 シリーズにおいては `ifupdown` を使用していましたが、 Armadillo Base OS では NetworkManager を使用しています。このように、 Debian と Armadillo Base OS では各種本体設定のやり方についても差があるものがあるので、詳細は対応する製品マニュアルを参照してください。

6.4. 動作確認をする

開発済みシステムの移行が完了した後には、動作確認をしてください。

正しく設定できているならば、 Armadillo に電源を投入すると自動的にコンテナが起動し、その中でユーザーアプリケーションが実行されるはずです。それが確認できたならば移行完了です。

6.5. Podman のデータの保存先を tmpfs に戻す

「5.3.1. Podman のデータの保存先を eMMC に変更する」で、開発前に Podman のデータの保存先を一時的に eMMC にしました。

開発が完了したタイミングで、必ず Podman のデータの保存先を tmpfs に戻しておいてください。

Podman のデータの保存先を tmpfs に戻す前に、不要なコンテナイメージを削除します。以下の例では、必要なコンテナイメージは user_app_container のみで、 debian のコンテナイメージは必要ありません。

```
armadillo:~# podman images
REPOSITORY                    TAG         IMAGE ID      CREATED        SIZE
localhost/user_app_container  1.0.0       9f8fa6515340  3 seconds ago  123 MB
docker.io/library/debian      bullseye    d352b92ddb5d  3 weeks ago    123 MB
```

podman rmi コマンドを実行して、 debian コンテナイメージを削除します。

```
armadillo:~# podman rmi -f docker.io/library/debian:bullseye ❶
Untagged: docker.io/library/debian:bullseye
```

❶　debian コンテナイメージを削除します

不要なコンテナイメージを削除できたら、 Podman のデータの保存先を tmpfs に戻します。

```
armadillo:~# abos-ctrl podman-storage --tmpfs
List of images configured on development storage:
REPOSITORY                    TAG         IMAGE ID      CREATED      SIZE
localhost/user_app_container  1.0.0       9f8fa6515340  12 days ago  123 MB

What should we do? ([C]opy (default), [N]othing, [D]elete)
Copy ❶
Delete subvolume (no-commit): '/mnt/containers_storage'
Replacing development images to readonly storage succeeded
Switching back to tmpfs container storage.
Successfully reverted podman storage to tmpfs
```

❶　eMMC に保存済みのコンテナイメージをどう扱うかを聞かれるので、 Copy と入力して Enter キーを押下してください

これで、 Podman のデータの保存先が tmpfs に戻りました。

6.6. Armadillo Base OS への移行後

Armadillo Base OS は、クローン用のインストールディスクの作成や、OTA 機能など、開発だけでなく量産及び運用時に便利な機能を多く備えています。詳細は対応する製品マニュアルを参照の上、ぜひご活用ください。

ここまでの内容および、 Armadillo Base OS で開発を進めていくにあたって不明な点がございましたら、「Armadillo フォーラム [https://armadillo.atmark-techno.com/forum/armadillo/]」にてご質問ください。

Armadillo-600 シリーズ Armadillo Base OS 移行ガイド

改訂履歴

バージョン	年月日	改訂内容
1.0.0	2023/6/29	・ 1.0.0 発行

Armadillo-600 シリーズ Armadillo Base OS 移行ガイド
Version 1.0.0
2023/06/29