How to create an SDK for OpenSTLinux distribution

Contents

How to create an SDK for OpenSTLinux distribution

Stable: 21.02.2020 - 08:26 / Revision: 20.02.2020 - 07:31

When an OpenSTLinux distribution has been modified, it is pertinent to build a new software development package that integrates the modifications, and to redistribute this SDK to developers (see SDK development cycle model).

Contents

1 Prerequisites

The Distribution Package relative to your STM32 microprocessor Series is installed: Category:Distribution Package.

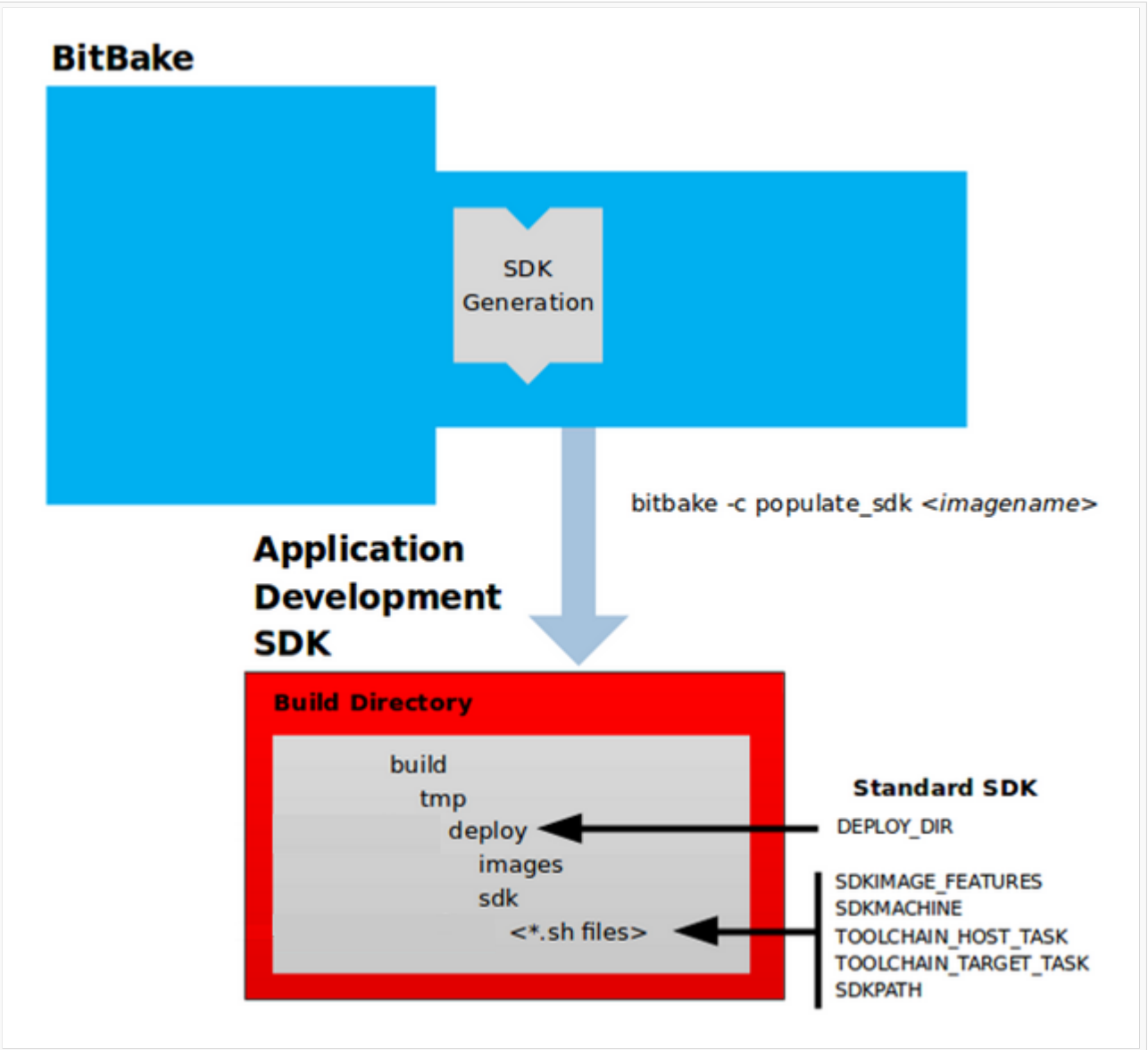
On the installation:

- some pieces of software might have been modified or integrated
- the build environment script has been executed
- the selected image has been rebuilt

2 SDK generation

The OpenEmbedded build system uses BitBake to generate the software development package (SDK) installation script.

For more information about the SDK, see the SDK for OpenSTLinux distribution article.



Standard SDK generation
Source: Yocto Project[1]

The *do_populate_sdk* task helps to create the standard SDK and handles two parts: a target part and a host part. The target part is built for the target hardware and includes libraries and headers. The host part is the part of the SDK that runs on the host machine.

- Check that the build environment script has been executed, and that the current directory is the build directory of the OpenSTLinux distribution (for example, *openstlinux-20-02-19/build-openstlinuxweston-stm32mp1')*
- Generate the SDK installation files (including the installation script) for a standard SDK with the following command :

```
PC $> bitbake -c populate_sdk <image>
```

Where:

<image>	Image name; example: • st-image-weston

Example:

PC $> bitbake -c populate_sdk st-image-weston

- The SDK installation files (*<image>-<distro>-<machine>-<host machine>-toolchain-<Yocto release>+snapshot.**) are written to the *deploy/sdk* directory inside the build directory *build-<distro>-<machine>* as shown in the figure above

Where:

<host machine>	Host machine on which the SDK is generated • x86_64 (64-bit host machine; this is the only supported value)
<Yocto release>	Release number of the Yocto Project; example: • 2.6 (aka thud)

Example

```
PC $> ls tmp-glibc/deploy/sdk/
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-2.6-snapshot.host.manifest
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-2.6-snapshot.sh
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-2.6-snapshot.target.
manifest
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-2.6-snapshot.testdata.json
```

The main final output is the cross-development toolchain installation script (*.sh* file), which includes the environment setup script.

Note that several OpenEmbedded variables exist that help configure these files. The following list shows the variables associated with a standard SDK:

```
    DEPLOY_DIR: points to the deploy directory.
    SDKMACHINE: specifies the architecture of the machine on which the cross-
development tools are run to create packages for the target hardware.
    SDKIMAGE_FEATURES: lists the features to include in the "target" part of the SDK.
    TOOLCHAIN_HOST_TASK: lists packages that make up the host part of the SDK (that
is, the part that runs on the SDKMACHINE). This variable allows packages other than
the default ones to be added.
    TOOLCHAIN_TARGET_TASK: lists packages that make up the target part of the SDK
(that is, the part built for the target hardware).
    SDKPATH: Defines the default SDK installation path offered by the installation
script.
```

3 To go further: eSDK generation

The OpenEmbedded build system uses BitBake to generate the extensible software development package (eSDK) installation script. The eSDK is a SDK in which user can execute Yocto commands, such as devtool. For more information about eSDK, please refer to Yocto SDK manual [2].

The *do_populate_sdk_ext* task helps to create the eSDK. You can customize a little bit the eSDK, to add or not the toolchain inside and to add or not all build artifacts inside (please note that the eSDK will be big in that case, more than 1GB).

- Generate the eSDK installation files (including the installation script) for a extensible SDK with the following command :

```
PC $> bitbake -c populate_sdk_ext <image>
```

Where:

<image>	Image name; example: • st-image-weston

Example:

PC $> bitbake -c populate_sdk st-image-weston

- The eSDK installation files (*<image>-<distro>-<machine>-<host machine>-toolchain-ext-<Yocto release>-snapshot.**) are written to the *deploy/sdk* directory inside the build directory *build-<distro>-<machine>* as shown in the figure above

Where:

<host machine>	Host machine on which the eSDK is generated • x86_64 (64-bit host machine; this is the only supported value)
<Yocto release>	Release number of the Yocto Project; example: • 2.6 (aka thud)

Example

```
PC $> ls tmp-glibc/deploy/sdk/
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-ext-2.6-snapshot.host.
manifest
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-ext-2.6-snapshot.sh
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-ext-2.6-snapshot.target.
manifest
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-ext-2.6-snapshot.testdata.
json
```

The main final output is the cross-development toolchain installation script (*.sh* file), which includes the environment setup script.
Note that several OpenEmbedded variables exist that help configure these files. Variables used to configure standard SDK also impact eSDK and following are specific to the eSDK:

```
    SDK_EXT_TYPE: Controls whether or not shared state artifacts are copied into the
extensible SDK.
    SDK_INCLUDE_TOOLCHAIN : Include or not the toolchain in the extensible SDK.
```

4 Reference list

- http://www.yoctoproject.org/documentation
- https://www.yoctoproject.org/docs/2.6.4/sdk-manual/sdk-manual.html

Software development kit (A programming package that enables a programmer to develop applications for a specific platform.)

also known as

Extensible Software development kit

Permission error

Stable: 21.02.2020 - 08:24 / Revision: 20.02.2020 - 07:13

You do not have permission to read this page, for the following reason:

The action "Read pages" for the draft version of this page is only available for the groups ST_editors, ST_readers, Selected_editors, sysop, reviewer

Category:Distribution Package

This category groups together all articles related to a Distribution Package (whatever the microprocessor device and the board).

The Distribution Package is specified in the Which Package better suits your needs article.

Pages in category "Distribution Package"

The following 6 pages are in this category, out of 6 total.

H

S