Yocto project Tricks & Hacks

- MOORTHY
  (moorthy.baskaravenkatraman-sambamoorthy@linaro.org)
Agenda

❖ Yocto project (OE-Core) and bitbake usages
❖ Things to be handled for faster build and development
❖ Extensible SDK Installer
❖ Things to be handled for yocto version migration
Introduction

✓ Python based build framework system
✓ Uses bitbake for task execution

Open Embedded Core (OE-Core)
- is a collaborative effort of yocto project and the recognized openembedded model
- Advantages: major architectures supported, distro less, one version for a recipe

Bitbake
➢ Engine to execute shell and python based tasks
➢ Parses meta-data and split up tasks
➢ Includes a fetch library to download source code

Meta-openembedded
• Collection of multiple layers and a supplement for OE-core with additional packages
Bitbake options

● To run individual task
   $ bitbake <recipe> -c <task>

● Force task to run (mainly for forceful configure and compile)
   $ bitbake <recipe> -c <task> -f

● To continue even on error
   $ bitbake <image> -k

● To create dependency tree file
   $ bitbake <image/recipe> -g

● To run without dependency
   $ bitbake -b ../meta-xxx/recipes-yyy/zzz/recipe-z.bb

● To show more logs
   $ bitbake -D <recipe>- enables more debug prints
   $ bitbake -v <recipe>- verbose log
Override Source Revision

Why?

- Queries to fetch a desired revision to build
- Helps to test new change on HEAD of source tree if SRCREV is pinned

Options

★ SRCREV\_pn-xxx
  - Picks the mentioned revision
  - ${AUTOREV}$ gets the HEAD revision from the source control
  - buildhistory-collect-srcrevs- tool to list out revisions built for all recipes

★ BB\_SRCREV\_POLICY
  - Cache - retains the revision
  - Clear - queries HEAD revision every time
  - Option applies for all over the recipes
Conserving disk space

**rm_work** class
- Lowers the amount of cache data
- Removes the work directory once all tasks completed for a recipe
- Usually to be inherit from `local.conf`

**RM_WORK_EXCLUDE**
- Excludes a list of recipes having their work directories deleted

**INHIBIT_PACKAGE_DEBUG_SPLIT**
- Prevents build system from splitting debug info during packaging

**DL_DIR**
- Overriding existing `downloads` dir for replicating build workspace
- Fasten builds by avoiding `do_fetch()` and conserves disk space
Task’s Script files

- Yocto project generates shell or python based scripts for each task
- Available under `temp` sub directory of `WORKDIR` work directory
- Scripts named as `run.do_<task>.<pid>` and logs store as `log.do_<task>.<pid>`

**OE-core** provides predefined classes for autotools, cmake, meson and other configuration manager system

```bash
$ bitbake <recipe> -c <task> -f
```

- Runs the same script but involves prior parsing recipes from all layers
- This script file can be launched manually to run the task faster as it avoid bitbake parsing
- Also can be modify but will not reflect on bb meta data

```bash
do_compile() {
    ...
    autotools_do_compile
}
autotools_do_compile() {
    oe_runmake
}

oe_runmake() {
    oe_runmake_call "@$" die "failed"
}

oe_runmake_call() {
    make -j 8 ...
}

do_configure() {
    ...
    cmake_do_configure
}
cmake_do_configure() {
    ...
    cmake ...
}
cd `$(WORKDIR)/build`
do_configure
```
Parallel builds

BB_NUMBER_THREADS
- Maximum number of threads bitbake can execute simultaneously
- OE-core build system configures automatically to equal the max. number of cores

BB_NUMBER_PARSE_THREADS
- Max. threads bitbake uses for parsing recipes

PARALLEL_MAKE
- `-j` option argument to `make` command
- By default OE-core gives the maximum core count using oe.utils.cpu_count()

PARALLEL_MAKEINST
- `-j` option argument to `make install` command
Extensible SDK

- Toolchain installer provided by OE-core environment with devtool support
- Generates Installer shell script files for host and target machines
- Integrated yocto build system with SDK
  
  $ bitbake <image_name> -c populate_sdk_ext

- Uses devtool to add, build and package recipes once installed the eSDK

SDK_EXT_TYPE

- “full” - bundles also the sstate artifacts, leads to bigger size of installer file
- “minimal” - no libraries or tools other than SDK

oe-publish-sdk

- Publishes the built SDK installer to your specified URL. (Release usage)
- For every installer generation, this has to be run to publish
Extensible SDK ...

devtool functions

➢ Add - adds a recipe with a fetch url. Url may be local or a source control
➢ Upgrade - upgrades the current version of a recipe
➢ Modify - changes the current source of an recipe to a new one
➢ Edit-recipe - opens the recipe file in an editor
➢ Build - configures & compiles the source
➢ Build-image - build the recipes mentioned in a image recipe
➢ Deploy-target - deploys the recipe to a specified target
➢ Finish - completes by creating patches if any local change and puts to the recipe path
Version Migration

Basic things to be handled on migrating meta layers to higher yocto version

➔ Layer configuration
   ✓ LAYERDEPENDS - dependency layers to be satisfied
   ✓ LAYERSERIES_COMPAT - adding new version for compatibility

➔ Replace deprecated functions & bbclass
   ✓ base_contains to bb.utils.contains, base_conditional to oe.utils.conditional
   ✓ distro_features_check to features_check

➔ Rebasing patches with latest OE version
   ✓ Open source patches to be rebased for current version from oe-core

➔ Version specific bbappends
   ✓ Best practice to maintain patches in version specific bbappends
   ✓ Generic changes can be in <recipe>_%.bbappend

➔ Toolchain error fixes
   ✓ Higher version toolchain comes with added flags that cause build error
Conclusion

The techniques mentioned in this presentation are based from the layer combination of openembedded-core and meta-openembedded. The tips and hacks mentioned are mainly focused for a faster build and development.

References

Thank you

Accelerating deployment in the Arm Ecosystem