devtool

- Collection of tools for working on recipes:
  - `devtool add`
  - `devtool edit-recipe`
  - `devtool upgrade`
  - `devtool finish`
  - `etc...`
devtool

- ...and more!
  - devtool modify
  - devtool deploy-target
  - devtool undeploy-target
  - devtool build
  - devtool build-image
  - etc...
devtool – why it exists

• Our build system is great for repeatable builds from source

• Working with the source itself was hard
  ● Tempting to just edit sources under tmp/work/…
  ● But workflow is painful after that (forced builds, manual patch generation, lost work…)

• Help newer users add new software (alongside regular build and within eSDK)
devtool – past presentations

• ELC 2017
  • Using Devtool To Streamline Your Yocto Project Workflow - Tim Orling
    • https://www.youtube.com/watch?v=CiD7rB35CRE
  
• ELC 2017
  • Yocto Project Extensible SDK: Simplifying the Workflow for Application Developers - Henry Bruce
    • https://www.youtube.com/watch?v=d3xanDJuXRA&t=57s
devtool – past presentations

• ELC 2018
  - Working with the Linux Kernel in the Yocto Project - Sean Hudson
  - https://www.youtube.com/watch?v=tZACGS5nQxw
devtool – past presentations

• YPDD 2018 - ELC
  ● Session 3, Devtool 1 - Tim Orling
  ● https://www.youtube.com/watch?v=C-usM6gFVSY

• YPDD 2018 - ELC
  ● Session 7, Devtool 2 - Tim Orling & Henry Bruce
  ● https://www.youtube.com/watch?v=UYsqlP_Qt_Q
devtool – documentation

• Yocto Project Reference Manual
  • chapter 8 - *devtool* Quick Reference
    • https://www.yoctoproject.org/docs/current/ref-manual/ref-manual.html#ref-devtool-reference

• Yocto Project Application Development and the Extensible Software Development Kit (eSDK)
  • chapter 2 - Using the Extensible SDK
    • https://www.yoctoproject.org/docs/current/sdk-manual/sdk-manual.html#sdk-extensible
devtool – documentation

- Yocto Project Linux Kernel Development Manual
  - section 2.4 - Using devtool to Patch the Kernel
$ devtool --help
usage: devtool [--basepath BASEPATH] [--bbpath BBPATH] [-d] [-q]
       [--color COLOR] [-h]
       <subcommand> ...

OpenEmbedded development tool

options:
  --basepath BASEPATH  Base directory of SDK / build directory
  --bbpath BBPATH      Explicitly specify the BBPATH, rather than getting it
                      from the metadata
  -d, --debug          Enable debug output
  -q, --quiet          Print only errors
  --color COLOR        Colorize output (where COLOR is auto, always, never)
  -h, --help           show this help message and exit

subcommands:
  Beginning work on a recipe:
    add                  Add a new recipe

...
devtool – documentation

$ devtool add --help

usage: devtool add [-h] [--same-dir | --no-same-dir] [--fetch URI]
   [--fetch-dev] [--version VERSION] [--no-git]
   [--srcrev SRCREV | --autorev] [--srcbranch SRCBRANCH]
   [--binary] [--also-native] [--src-subdir SUBDIR]
   [--mirrors] [--provides PROVIDES]
   [recipename] [srctree] [fetchuri]

Adds a new recipe to the workspace to build a specified source tree. Can
optionally fetch a remote URI and unpack it to create the source tree.

arguments:
  recipename            Name for new recipe to add (just name - no version,
                        path or extension). If not specified, will attempt to
                        auto-detect it.
  srctree               Path to external source tree. If not specified, a
                        subdirectory of /z/ypdd/2018-10-devtool/my-
                        class/poky/build/workspace/sources will be used.
  fetchuri              Fetch the specified URI and extract it to create the
                        source tree

options:
  -h, --help            show this help message and exit
  ...
devtool – workspace

• a separate environment (layer) in which to work on recipes, sources, patches
devtool – workspace (bitbake mode)

- how the various devtool commands relate to your layers, your target, and your workspace
devtool – multiple targets?
devtool – multiple targets?

- yes
- specify target’s IP with un/deploy-target
Sidebar: recipetool

- Extra set of tools for working on recipes
- Contains logic for creating recipes (used by `devtool add`)
- Can also create/update bbappends, programmatically set variables in recipes, etc.
Questions?
Hands On
devtool – setup

$ nano ~/.ssh/config

```
Host qemu
  User root
  Hostname localhost
  Port 2222
  StrictHostKeyChecking no
  UserKnownHostsFile /dev/null
```

$ git clone -b hardknott git://git.yoctoproject.org/poky # (already done in devday host)
$ source ./yp-summit-may-21/poky/oe-init-build-env ~/yp-summit-may-21/poky/build-devtool
$ edit conf/local.conf
   MACHINE = "qemux86-64"
   IMAGE_INSTALL_append = " openssh"
   EXTRA_IMAGE_FEATURES ?= "debug-tweaks"
$ bitbake core-image-base
**devtool – setup**

```bash
$ bitbake-layers create-layer ../meta-foo
$ bitbake-layers add-layer ../meta-foo
$ git config --global user.name "name"
$ git config --global user.email "name@example.com"
```

- open a second ssh connection to the build machine

```bash
2$ source ./yp-summit-may-21/poky/oe-init-build-env ~/yp-summit-may-21/poky/build-devtool
2$ runqemu slirp nographic serial
```

- do the exercises in the first connection, work on the target in the second connection

- login as "root", no password (thanks to "debug-tweaks")
devtool – getting started

```
$ devtool add \\n  https://nano-editor.org/dist/v4/nano-4.2.tar.xz
```

- implicitly creates workspace (if it doesn’t already exist)
- guesses the recipe name `nano` (correctly!)
- looks at the source and determines it’s an *autotools* project (true! and *pkgconfig* and *gettext*)
- guesses at DEPENDS (correctly! *ncurses* and *zlib*)
- creates a “rough” recipe

```
$ devtool status
$ devtool find-recipe nano
$ devtool edit-recipe nano
```
devtool – getting started

• let’s see if it builds

$ devtool build nano

• it works!
devtool – what goes in a workspace?

• the things on which you are working:
  • recipes
  • patches
  • sources
  • etc...

$ tree -d workspace

• ...except sources can be, optionally, outside the workspace
devtool – let’s see nano run

- examine
tmp/deploy/images/qemux86-64/core-image-base-qemux86-64.manifest

  - verify there’s no “nano” package

- in the terminal running qemu, log in and verify there’s no nano

```bash
root@qemux86-64# nano
-sh: nano: command not found
```

- send nano to target

```bash
$ devtool deploy-target nano qemu
```

- now nano runs
sidebar – SLIRP versus TUN/TAP

- Yocto Project supports several connection technologies for QEMU

- **SLIRP**: advantage is no root access required, disadvantages are minimal documentation, requires SSH knowledge, ICMP (e.g. ping) not available by default

- **TAP**: advantage is simpler setup, disadvantage is that it requires sudo access

```bash
2$ sudo runqemu nographic serial
$ devtool deploy-target nano qemu

2$ runqemu slirp nographic serial
$ devtool deploy-target nano root@192.168.7.2
```

_qemu_ is defined in ~/.ssh/config (see earlier slide)
devtool – let’s see nano run

- build an entire image

```bash
$ devtool build-image core-image-base
...
NOTE: Building image core-image-base with the following additional packages: nano
...
```

- examine `tmp/deploy/images/qemux86-64/core-image-base-qemux86-64.manifest`
  - now there is a `nano` package
- why not just use “bitbake core-image-base”?
  - `nano` package not automatically added
devtool – upgrade

• try upgrading nano

```bash
$ devtool upgrade nano
ERROR: recipe nano is already in your workspace
```

• we need to move the `nano` recipe to `Your Layers` before we can `upgrade`
  • preferably our own (meta-foo)

• this is only an issue because nano is in the workspace already – normally `devtool upgrade` is where you start an upgrade for an existing recipe
devtool – upgrade

- we can't *upgrade* a recipe that is already in the workspace

- an *upgrade* must come from *your layers*
devtool – upgrade

• first we need to **finish**
devtool – upgrade

- then we can **upgrade**
devtool – upgrade

```
$ devtool finish nano ..META-f
ERROR: Source tree is not clean:
...
```

- this error is *not* a problem we introduced; it is a nano-specific issue – but we need to tell devtool it’s OK with `-f`

```
$ devtool finish -f nano ..META-f
INFO: No patches or files need updating
INFO: Moving recipe file to .../META-f/recipes-nano/nano
INFO: Leaving source tree .../poky/build-devtool/workspace/sources/nano as-is; if you no longer need it then please delete it manually
```

- it is worth noting that it will not remove the sources; we need to do it explicitly

```
$ rm -fr workspace/sources/nano
```
$ devtool upgrade nano

...  
ERROR: Automatic discovery of latest version/revision failed - you must provide a version using the --version/-V option, or for recipes that fetch from an SCM such as git, the --srcrev/-S option.

- in the specific case of **nano**, devtool can’t figure out how to find and upgrade tarballs (this information is not obvious from the URL)
devtool – upgrade

• we need to give devtool more help
  $ devtool upgrade -V 4.3 nano

• it works!
  $ devtool build nano

• it works!
devtool deploy-target - dive in

- is it okay to re-deploy a second time without cleaning up the first deploy?
  - yes... usually

- on the target

```
root@qemux86-64# cd /
root@qemux86-64# ls -a
...
.devtool
...
root@qemux86-64# cd .devtool
root@qemux86-64# ls -l
-rw-r--r-- 1 root root 4969 Oct 20 06:03 nano.list
```
**devtool deploy-target - dive in**

- *nano.list* is created by devtool, per package, when it deploys to the target

- examine *poky/scripts/lib/devtool/deploy.py* for all the answers
  - it creates a script that is copied to target
  - preserves any files that would be clobbered
  - generates a list of files being deployed, so they can be undeployed
  - deploying starts by undeploying (same recipe name)
devtool deploy-target - dive in

- undeploy, and verify nano is removed from target, and the plumbing is also removed

```
$ devtool undeploy-target nano qemu

root@qemux86-64# ls -a /
```

- remember to finish and cleanup

```
$ devtool finish -f nano ../meta-foo
$ rm -fr workspace/sources/nano
```
devtool - floating devtool commands

• some devtool commands don’t care whether the recipe is in the workspace or the layers

$ devtool status
NOTE: No recipes currently in your workspace

$ devtool edit-recipe ethtool
(works)

$ devtool latest-version ethtool
NOTE: Current version: 5.10
NOTE: Latest version: 5.10

$ devtool find-recipe ethtool

$ devtool search ethtool
devtool - creating a patch

- use-case? patches can be needed to
  - add/remove functionality
    - reduce size on target
    - remove dependency/dependencies
  - allow code to be (cross-)compiled
devtool - creating a patch

$ devtool add https://github.com/twoerner/autotool-devtool-example/archive/v1.0.0.tar.gz
$ devtool build autotool-devtool-example
$ devtool deploy-target autotool-devtool-example qemu

root@qemux86-64# autotool-devtool-example
Hello, world!
version: 1.0.0
Hello from the library
devtool - creating a patch

• edit the code

$ pushd workspace/sources/autotool-devtool-example
$ nano src/autotool-devtool-example.c

• change from

```c
printf("Hello, world!\n");
```  

• to

```c
printf("Hello, devtool!\n");
```
devtool - creating a patch

- build, deploy, verify

$ popd
$ devtool build autotool-devtool-example
$ devtool deploy-target autotool-devtool-example qemu

root@qemux86-64# autotool-devtool-example
Hello, devtool!
version: 1.0.0
Hello from the library
devtool - creating a patch

• cleanup

```bash
$ devtool finish autotool-devtool-example ..,/meta-foo
ERROR: Source tree is not clean:
  M src/autotool-devtool-example.example.c
```

• oops! but it’s nice it didn’t clobber or lose my work

```bash
$ pushd workspace/sources/autotool-devtool-example
$ git commit -avs
...
$ popd
$ devtool finish autotool-devtool-example ..,/meta-foo
...
INFO: Adding new patch 0001-update-salutation.patch
...
$ rm -fr workspace/sources/autotool-devtool-example
```
devtool - creating conflict

• now we’ll update to a newer release, but the newer release will conflict with our patch

```shell
$ devtool upgrade autotool-devtool-example
...
Connecting to github.com (github.com)|192.30.253.113|:443... connected.
HTTP request sent, awaiting response... 404 Not Found

ERROR: Automatic discovery of latest version/revision failed - you must provide a version using the --version/-V option, or for recipes that fetch from an SCM such as git, the --srcrev/-S option.
```

• devtool can’t figure it out, we need to help it
devtool - creating conflict

$ devtool upgrade -V 1.0.1 autotool-devtool-example

...  
WARNING: Command 'git rebase cdb5e8e1d76e5022ae754ea95dc5e4cf85af7670' failed: 
First, rewinding head to replay your work on top of it...
Applying: update salutation
Using index info to reconstruct a base tree...
M src/autotool-devtool-example.c
Falling back to patching base and 3-way merge...
Auto-merging src/autotool-devtool-example.c
CONFLICT (content): Merge conflict in src/autotool-devtool-example.c
error: Failed to merge in the changes.
Patch failed at 0001 update salutation
The copy of the patch that failed is found in: .git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue". 
If you prefer to skip this patch, run "git rebase --skip" instead. 
To check out the original branch and stop rebasing, run "git rebase --abort". 
You will need to resolve conflicts in order to complete the upgrade.
Devtool - resolving conflict

- keep the new, or keep the old?
  - keep the new

```bash
$ pushd workspace/sources/autotool-devtool-example
$ nano src/autotool-devtool-example.c
```
devtool - resolving conflict

• from

...  
13  <<<<<<<< HEAD
14    /* a meaningful comment */
15    printf("Hello, world!\n");
16    ||||||| merged common ancestors
17    printf("Hello, world!\n");
18    =======
19    printf("Hello, devtool!\n");
20    >>>>>>> update salutation
...

• to

...  
13    /* a meaningful comment */
14    printf("Hello, devtool!\n");
...
devtool - resolving conflict

$ git add src/autotool-devtool-example.c
$ git rebase --continue
Applying: update salutation
$ popd

- This time, let’s inspect recipe updates first with -N:

  $ devtool finish autotool-devtool-example ../meta-foo -N

- If we’re happy with the proposed changes, apply them:

  $ devtool finish autotool-devtool-example ../meta-foo
devtool - resolving conflict

```
$ devtool finish autotool-devtool-example ../meta-foo
$ tree ../meta-foo
../meta-foo/
...
  recipes-nano
  ├── nano
  │   └── nano_4.3.bb
  └── recipes-autotool-devtool-example
      ├── autotool-devtool-example
      │   └── autotool-devtool-example
      │       └── 0001-update-salutation.patch
      └── autotool-devtool-example_1.0.1.bb
```
devtool - modify

1) takes an existing recipe from layers
2) unpacks sources into workspace
3) edit recipe or sources
4) … (same as devtool add/devtool upgrade workflow)
devtool modify example

$ devtool modify bc
INFO: Source tree extracted to /home/ilab01/yp-summit-may-21/poky/build-devtool/workspace/sources/bc
INFO: Recipe bc now set up to build from /home/ilab01/yp-summit-may-21/poky/build-devtool/workspace/sources/bc
$ devtool edit-recipe bc

• Take a note of file://libmath.h in SRC_URI, then exit and continue

$ pushd /home/ilab01/yp-summit-may-21/poky/build-devtool/workspace/sources/bc
$ ls
aclocal.m4  compile  COPYING.LIB  FAQ  Makefile.am  README
ar-lib  config.h.in  dc  h  Makefile.in  Test
AUTHORS  configure  depcomp  INSTALL  missing  ylwrap
bc  configure.ac  doc  install-sh  NEWS
ChangeLog  COPYING  Examples  lib  oe-local-files
$ ls oe-local-files
libmath.h
devtool modify example

- Edit `bc/main.c` and make a trivial change to the help text printed in `usage()` (line 69)

$ nano bc/main.c

- Commit changes and run `devtool finish`

$ git add bc/main.c
$ git commit -s
...
$ popd
$ devtool finish bc ../meta/foo
...

NOTE: Writing append file `/home/ilab01/yp-summit-may-21/poky/meta-foo/recipes-extended/bc/bc_% bbappend`

NOTE: Copying `0001-Change-help-text.patch` to `/home/ilab01/yp-summit-may-21/poky/meta-foo/recipes-extended/bc/bc/0001-Change-help-text.patch`
devtool modify example

- devtool finish realised the bc recipe is not in meta-foo
  - Thus it created a bbappend and placed the patch next to it
  - Naturally if we had passed the path to poky/meta it would have modified the original recipe
Wrap up
devtool - eSDK Mode

- the eSDK includes many improvements over the standard SDK
- everything the standard SDK can do, plus all of the functionality we’ve been looking at that is provided by devtool
devtool – mode commands

- **bitbake mode**
  - add
  - build
  - build-image
  - configure-help
  - check-upgrade-status
  - create-workspace
  - deploy-target
  - edit-recipe
  - export
  - extract
  - find-recipe
  - finish
  - import
  - latest-version
  - menuconfig
  - modify
  - rename
  - reset
  - search
  - status
  - sync
  - undeploy-target
  - update-recipe
  - upgrade

- **eSDK mode**
  - add
  - build
  - build-image
  - build-sdk
  - configure-help
  - check-upgrade-status
  - deploy-target
  - edit-recipe
  - export
  - extract
  - find-recipe
  - finish
  - import
  - latest-version
  - menuconfig
  - modify
  - package
  - rename
  - reset
  - runqemu
  - sdk-install
  - sdk-update
  - search
  - status
  - sync
  - undeploy-target
  - update-recipe
  - upgrade
devtool – mode commands

- why does eSDK mode get extra features?
  - because an eSDK doesn’t have `bitbake` or `scripts/`
  - `devtool` is the cornerstone of the eSDK
Future

- Multiconfig support
- Recipe modification fixes
- recipetool enhancements (make `devtool add smarter`)
- Your idea here :)
  - Help very much welcome!
Conclusion

• Try it out on your own sources / recipes:
  - `devtool add` on a source tree / tarball / URL
  - `devtool modify` and work on an existing recipe
  - `devtool upgrade` existing recipe to a new upstream version

• See documentation links & other presentations (earlier slide)
Conclusion

• Please send feedback!
  • Yocto Project mailing list
    • https://lists.yoctoproject.org/g/yocto
  • IRC (#yocto on Freenode)
  • Email: paul.eggleton@linux.microsoft.com
Questions?
Thanks for your time