Using debuginfod with The Yocto Project®

Dorinda Bassey

Yocto Project Summit, May 2021
About me

• Past Outreachy Intern at the Yocto project
  https://dorindabassey.github.io/
• B.eng Electrical Electronics Engineering, University of Uyo Nigeria.
• Interest
  Embedded Linux systems
  workflow around building and packaging OS Image
  FOSS contribution

  dorindabassey@gmail.com
Introduction
• Bugs are inevitable

• Identifying and fixing these defects is part of the development process

• Hence the need for flexible tools to improve program performance
  • Many different techniques exist – tracers, profilers, interactive debugging. (perf, gdb, systemtap...). They all depend on “debugging information” but won’t talk about them except for accessing debuginfo using GDB tool.
What is Debuginfod?

• Debuginfod from elfutils is a web fileserver for debugging artifacts
  • Makes debugging information available on a server for easy debug and distribution of "debuginfo" files.
  • Debuginfod serves that content over http to debuggers & similar tools currently in the Yocto Project - (gdb, elfutils, binutils)
  • Debugging becomes easier
Why Debuginfod

- Debuginfo containing debug files and source files is usually not packaged with it’s distro / deployed binaries due to:
  - Size of debug packages (15+ times the size of stripped binaries)
  - Memory or disk space constraints of the target device

- Although debug and source packages can be installed by adding this to the image:
  
  ```
  EXTRA_IMAGE_FEATURES = "dbg-pkgs src-pkgs"
  ```

  - But note → dramatic increase in size of image
• enable build history in local.conf to see difference
Hence elfutils debuginfod server

• **Setup server:**
  • Enable debuginfod in elfutils-native pkgconfig and distro via local.conf
    ```
    PACKAGECONFIG_pn-elfutils-native = "debuginfod libdebuginfod"
    DISTRO_FEATURES_append = " debuginfod"
    ```
  • run the script for the packages deploy dir
    ```
    user@user:~/Poky/build$ oe-debuginfod
    ```
  • or aim it at Build directories, RPMs, DEBs, IPKs, etc. By passing the variables directly specifying dir e.g
    ```
    user@user:~/Poky/build$ oe-run-native elfutils-native debuginfod --verbose -U /home/dorinda/Poky/build/tmp/deploy/ipk/core2-64/
    user@user:~/Poky/build$ oe-run-native elfutils-native debuginfod --verbose -R /home/dorinda/Poky/build/tmp/deploy/rpm/
    ```

all depending on what binary package directories you want to scan
Client support?

- Find the port debuginfod is listening to on the host (default port is 8002)

- Export address variable to the environment:
  - On the target e.g qemu
    ```bash
    root@qemux86-64:~# export DEBUGINFOD_URLS="http://192.168.7.1:8002/"
    ```
  - If debugging on the host
    ```bash
    user@user:~/Poky/build$ export DEBUGINFOD_URLS="http://localhost:8002/"
    ```

- Load debugging symbols of the binary
  ```bash
  $ gdb /bin/bash
  ```
Client support?

- You could also use `debuginfod-find` command to query the server.
- Remember to add “elfutils” to the image to use `debuginfod-find` on Target.
  
  ```
  IMAGE_INSTALL_append = " elfutils"
  ```
- On Target
  
  ```
  root@qemu-x86-64:~# debuginfod-find debuginfo /executable/Path
  ```
- On Host
  
  ```
  user@user:~/Poky/build$ oe-run-native elfutils-native debuginfod-find debuginfo BuildId[SHA]
  ```
What is a Build-ID

- **Unique Identification of binaries**
  - Each executable or shared library is assigned a unique identification of 160-bit
- **Display the build-id of a binary with the following command:**
  - `readelf -n /bin/ls` or
  - `file /bin/ls`
- **Useful in analysing core files such as:**
  - The core file itself
  - Executable binaries which has crashed
  - The shared libraries loaded in the binary when it crashed
Demo Time:
See Also:

https://sourceware.org/elfutils/Debuginfod.html

#elfutils on irc.freenode.net