debuginfod

and its interaction with the Yocto Project

Dylan Garza

June 6, 2022

debuginfod

- What is debuginfod? debuginfod is a daemon turns a machine that holds debug artifacts into file server for easier debugging.
 Tools such as gdb, valgrind, and systemtap utilize debuginfod.
- Why use it? Debug info is too big to be stored locally on the target devices. Different versions of binaries exist, so corresponding debug info is fethced automatically with gdb.

Installing debuginfod

There are two ways to install on a Ubuntu machine:

- Upgrade to Ubuntu 22.04
- append the "impish" package to /etc/apt/sources.list by appending the following line:

deb http://archive.ubuntu.com/ubuntu impish main restricted universe multiverse

Setting up and using debuginfod

Starting the debuginfod server:

- To start the file server, simply run debuginfod -F. Different arguments will search for different debug artifacts (-F -R -U)
 - -F sets file scanning for ELF/DWARF artifacts.
 - -R sets file scanning for RPMs
 - -U sets file scanning for .deb
- Providing a path to a directory will point debuginfod where to scan for the debug artifacts.
- debuginfod must initially scan the pointed directory for debug artifacts. A rescan is done every 5 minutes but this can be changed with the -t option

debuginfod -t 30 -F /home/dylan/debug_info



Setting up and using debuginfod (cont.)

On the developers machine:

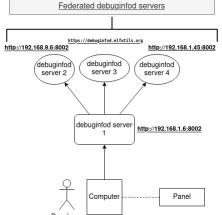
 set environment variable DEBUGINFOD_URLSto the ip address of the file server, prefixed by http:// and suffixed by the port number defaulted to :8002, which can be changed by -p [port num]

```
export DEBUGINFOD_URLS="http://ip_addr:8002"
```

• gdb will now query the given debuginfod server for the debug info via Build-ID and download it to the machines ~/.cache/debuginfod_client/directory, where it will be sorted by Build-ID

Federating debuginfod servers

Adding multiple URLs to the environement variable can become cumbersome. Federating debuginfod servers can simplify the process by having one debuginfod server query other servers if it does not contain the requested artifacts.



Federeating debuginfod servers (cont.)

```
1 2 3 4 5 6 7 8 9 [2]
                                                                                                  Pv4: 192.168.1.6II (dylandy II (RAM: 3.5 Glb. I (CPU: 2% I (Speed: 2.7 GHz I) (2022-06-06
Reading symbols from /bin/nvim...
                                                                                            [dylandy@super-computer ~]$ export DEBUGINFOD URLS="https:
                                                                                            //debuginfod.elfutils.org"
This GDB supports auto-downloading debuginfo from the following URLs:
                                                                                            [dylandy@super-computer ~]$ [
https://debuginfod.archlinux.org
Enable debuginfod for this session? (v or [n]) v
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading 14.73 MB separate debug info for /bin/nvim
Reading symbols from /home/dylandy/.cache/debuginfod client/2e49f4960123ce62457f5ad3fc66
(gdb) list
Downloading 0.06 MB source file /usr/src/debug/neovim-0.7.0/src/nvim/main.c
          set lang var():
          init signs();
203
          ui comp svn init():
284
206
        #ifdef MAKE LIB
        int nvim_main(int argc, char **argv);
                                                                                            [Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174); archive :
208
        int nvim main(int argc, char **argv)
        #elif defined(WIN32)
                                                                                            [Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174): buildids
(gdb) |
                                                                                            [Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174): filenames
                                                                                             310324
rdvlandv@super-desktop ~
 Arm -rf .cache/debuginfod_client
                                                                                            [Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174): files sca
 —dylandy@super-desktop ~
                                                                                            nned (#) 308005
 -A export DEBUGINFOD URLS="http://192.168.1.2"
                                                                                            [Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174): files sca
 —dylandy@super-desktop ~
                                                                                            nned (mb) 82493
                                                                                            [Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174): index db
                                                                                            size (mb) 80
                                                                                            [Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174); groomed d
                                                                                            atabase in 1.15333s
                                                                                            [Mon 06 Jun 2022 05:25:54 PM GMT] (19172/19175): fts trave
                                                                                            rsed source paths in 4.13616s, scanned=364305, regex-skipp
                                                                                            ed=0
                                                           "super-desktop" 13:27 06-Jun-22 [0] 0:python*
                                                                                                                "dylandy@super-compute" 13:29 06-Jun-2
```

debuginfod-find

 The debuginfod-find command allows the retrieval of debug artifacts, binary executables, or sources without the use of gdb. The command simply calls for the type of file to query and retrieve and the path to the binary or Build-ID. Additionally if a source is requested the path to the source also must be provided.

debuginfod-find [debuginfo/executable/source] [path/to/file or Build-ID] [path/to/source]

- The requested file will be stored in the same directory as a normal successful debuginfod query.
- Source files will be named by their path, with "##" in place of "/".

~/.cache/debuginfod_client/[Build-ID]/path##to##source.c



debuginfod with the Yocto Project

For versions after 3.4(honister) debuginfod is built into the Yocto Project. Only 2 changes need to be made to local.conf in order for debuginfod to

- PACKAGECONFIG_pn-elfutils-native="debuginfod libdebuginfod"
 - From the elfutils package that is for the target device, enable the recipe feature of debuginfod and libdebuginfod.
- DISTROF_EATURES += "debuginfod"

debuginfod with the Yocto Project (cont.)

- Yocto has a built-in wrapper script around debuginfod that ensures the same versions are used so that the target device is able to query the server properly.
- Source paths is the concatination of usr/src/debug/.. (which
 is set by standard) and the binary's \$WORKDIR/[package] or
 simply \$PKGD
- An easy way to find the source path is to use gdb to retrieve the binary with debug symbols. Then run the list in gdb in which the source path will be displayed.

Demo

Demo