

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 fetched 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:

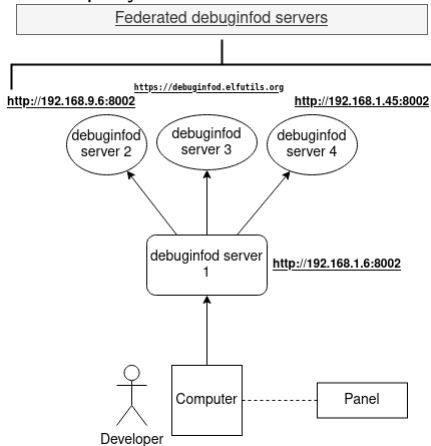
- set environment variable `DEBUGINFOD_URLS` to the ip address of the file server, prefixed by `http://` and suffixed by the port number defaulted to `:8002`, which can be changed by `-p [port num]`

```
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 environment 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 [G] [Gnu] attach -d 0
Reading symbols from /bin/nvim...

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.archlinux.org
Enable debuginfod for this session? (y or [n]) y
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
8c518c328c49/debuginfo...
(gdb) list
Downloading 0.06 MB source file /usr/src/debug/neovim-0.7.0/src/nvim/main.c
200     set_lang_var();           // set v:lang and v:ctype

201
202     init_signs();
203     ui_comp_syn_init();
204 }
205
206 #ifdef MAKE_LIB
207 int nvim_main(int argc, char **argv); // silence -Wmissing-prototypes
208 int nvim_main(int argc, char **argv)
209 #elif defined(WIN32)
(gdb) |

dylandy@super-desktop ~
└─rm -rf .cache/debuginfod_client
dylandy@super-desktop ~
└─export DEBUGINFOD_URLS="http://192.168.1.2"
dylandy@super-desktop ~
└─
```

```
[PV4: 192.168.1.6] dylandy | eRAM: 3.5 GB | CPU: 2% | Speed: 2.7 GHz | 0:02:06.06 13:27:31
[dylandy@super-computer ~]$ export DEBUGINFOD_URLS="https://debuginfod.elfutils.org"
[dylandy@super-computer ~]$
```

```
[Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174): archive s
def 0
[Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174): buildids
615
[Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174): filenames
310324
[Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174): files sca
nned (#) 308005
[Mon 06 Jun 2022 05:25:51 PM GMT] (19172/19174): files sca
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-skip
ped=0

[0] 0:gdb* "super-desktop" 13:27 06-Jun-22
[0] 0:python* "dylandy@super-compute" 13:29 06-Jun-22
```

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) only 2 changes are needed for debuginfod to work on a build. Append the following the `local.conf`:

1

```
PACKAGE_CONFIG_pn-elfutils-native="debuginfod libdebuginfod"
```

2

```
DISTROFEATURES += "debuginfod"
```

Yocto Project has built in script for debuginfod to eliminate compatibility issues.

Demo

Demo