Compiler Update

Applications Subcommittee

Dan Gritter

2024 TPF Users Group Conference
May 05-08, New Orleans, LA

IBM Z

IBM

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

As-Is
Compiler Support

GCC 7

- PJ45408 – Mar 2019
- Full support for C11/C++11/C++14 standards

Systems/C and Systems/C++ (Dignus) 2.25

- PJ46531 – Feb 2022
- Mirrors GCC 7 in terms of libstdc++ run time support

Note: GCC 4.6 no longer supported as of Aug 2023

- Deprecated and withdrawn support
- Fully removed from product (PJ47248 – Feb 2024)

Recent Compiler Updates

PJ46962 – Dignus 2.25.32 (Jan 2023)

- Compiler issue with DWARF anonymous unions, no z/TPF updates

PJ46997 – GCC 7 tpf-17r1-7 (Feb 2023)

- Compiler issue with skip trace exception handling

PJ46990 – Dignus 2.25.41 (Jul 2023)

- Compiler update in support of OpenSSL 3.0

Compiler requirements documented by APAR level

- Compiler update always coupled with a z/TPF APAR for maintainability

Recent Deliverables

PJ47102 – iostream compiler warning (Jul 2023)

- Optionally warn when including the <iostream> header for any z/TPF online program (GCC 7 and Dignus 2.25)

PJ47248 – Discontinue GCC 4.6 support (Feb 2024)

- Use of GCC 4.6 fully removed from product

Recent Deliverables
iostream compiler warning

Background

- Use of <iostream> on z/TPF has always been discouraged
- C++ chooses to statically initialize the global stream objects for stdin and stdout upfront in each application
- Incurs unnecessary performance overhead when an application does not use any <iostream> functionality
- Upstream GCC recently addressed this by moving object construction inside of libstdc++, instead of in every application source including <iostream>

Recent Deliverables
iostream compiler warning

PJ47102 provides the ability to optionally warn when including <iostream> for any z/TPF online program

- Compile time detection of runtime performance impact
- Controlled via _TPF_WARN_IOSTREAM macro definition
- Disabled by default to mirror prior compiler experience
- Recommend overriding default to globally enable detection, and subsequently suppress warning as needed, on an application-by-application basis
- Further details: https://community.ibm.com/community/user/ibmz-and-linuxone/blogs/jt-plotzke/2023/08/16/iostream-compiler-warning

Recent Deliverables
Discontinue GCC 4.6 support

PJ47248 fully removes the ability to use GCC 4.6 from z/TPF

- Already no longer supported as of Aug 2023
- MakeTPF tools updated to no longer allow GCC 4.6
- All OCO libraries rebuilt with GCC 7 now and going forward
- Some .mak file updates to remove any GCC 4.6 guards
- No additional rebuild requirements beyond OCO updates
- Necessary prior to supporting glibc 2.37

You must migrate to GCC 7 before applying PJ47248

Future Deliverables
glibc 2.37 support

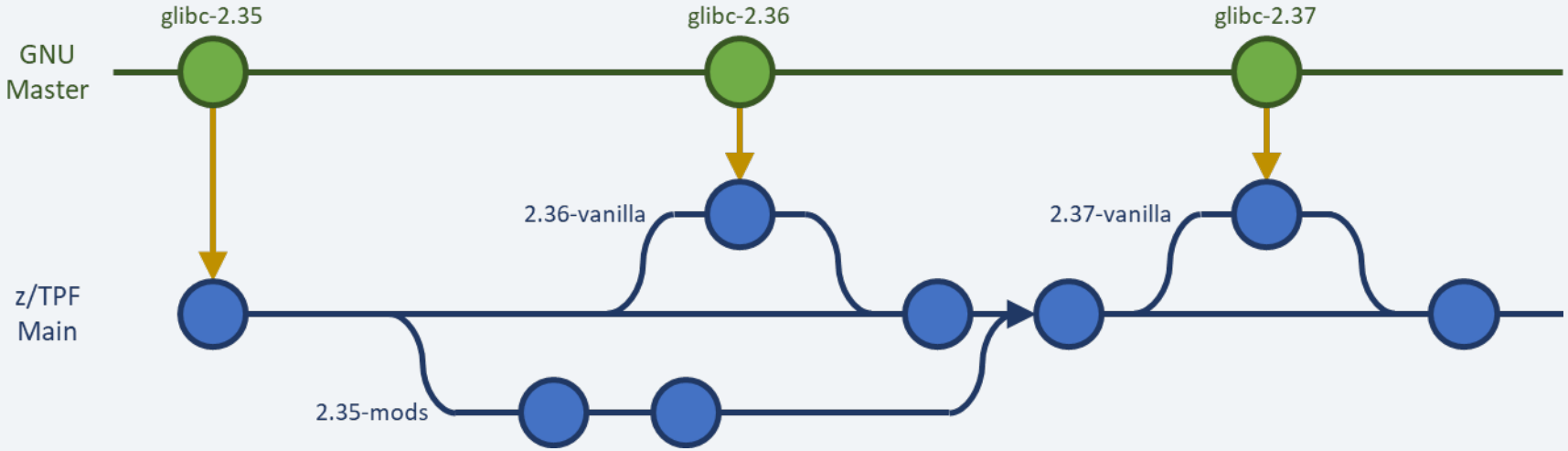
PJ46900 will provide support for glibc 2.37 (2024)

- Upgraded existing CISO and CLBM functionality
- New thread library support will be incorporated into CISO
- C11 standard support
- Favor open-source headers over z/TPF copies previously forked under base/include and base/filesys/include
- Better functionality and maintainability
- Dignus 2.25.*xx* minor version update to be required

Future Deliverables
glibc 2.37 support

Will revamp our internal process for maintaining glibc

- Internally using a Git source repository to incorporate each new glibc version



Future Deliverables

Add support for new GCC *v.next* (e.g. GCC 14)

- Multiple deliverables (compiler pre-release, then GA support)
- Further details in compiler release strategy

C++17 standard support (all compilers)

- Requires new glibc 2.37 support first
- Will only support new C++ standards once **full** standard support is available in our latest supported version of GCC

Update Dignus compiler support as needed to maintain compatibility with latest GCC

- Must always follow support for new GCC *v.next* due to shared C++ library

Compiler Release Strategy

Compiler upgrades to happen more frequently

- Every 3(?) years
- Aim to minimize effort (amount needed to recompile and test) to migrate to newer compiler
- Better incentivize picking up new compiler once available vs. when old compiler is no longer available

Investigation underway to identify next GCC version and features

- New architecture level support
- New compiler features (e.g. <iostream> performance)
- New compiler optimizations and bug fixes

Pain Points
Compiler Upgrades

Overall effort to perform mass recompiles with new compiler

New warnings in existing, untouched applications

Addressing all new errors/warnings in non-IBM applications

Maintaining clean compile at both current and new compiler versions before migration completes

Compiler Release Strategy

Continue pattern of:

1. (Announce) Deprecate oldest supported GCC version (e.g. 4.6)
2. (APAR) Discontinue support for GCC *v.old*
3. (APAR) Pre-release GCC *v.next* compiler with any necessary code compatibility updates for compilation
4. (APAR) Add support for GCC *v.next* (e.g. 13)

GCC *v.cur* (e.g. 7) continues to receive full support throughout entire cycle above

Compiler Release Strategy

3. Pre-release GCC *v.next* with compatibility updates (APAR)

- APAR would focus on z/TPF header updates require for clean compiles
- Post a beta version of the GCC *v.next* cross compiler
- Introduce new **`TPFGCC_VERSION := NEXT`** maketpf.cfg option
- When using the **`NEXT`** version, only perform compilation not linkage
- Allows same compile time testing and warning resolution to be done for non-IBM applications while the TPF Lab concurrently does so for the z/TPF product
- Better overlaps migration timelines with compiler release timelines

Compiler Release Strategy

4. Add support for GCC *v.next* (APAR)

- APAR itself already standard business approach
- Options for **`TPFGCC_VERSION`** are now: **`7 13 COMPAT`**
- When using the **`COMPAT`** version:
  - GCC 14 would be used but with as many new warnings as possible disabled
  - Also revert any default compile options back to their GCC 7 defaults
- Can adopt the new compiler before all new application warnings are addressed

To-Be Compiler Support

GCC 7

- Full support for C11/C++11/C++14 standards
- Experimental support for C++17 standards

GCC 14

- 2025
- Full support for C11/C++17 standards

Systems/C and Systems/C++ (Dignus) 2.xx

- Mirrors GCC 14 in terms of libstdc++ run time support

Be a sponsor user

Sponsor users assist in design and implementation, and your feedback drives our development cycle.

Target personas

- Application developer
- System programmer
- Build manager

Interested? Contact

Dan Gritter (dgritter@us.ibm.com)
JT Plotzke (jtplotzk@us.ibm.com)

Thank you