FreeBSD 12.0 continues the trend in recent FreeBSD releases of transitioning away from obsolete GPLv2-licensed toolchain components to modern ones. During the 12.0 development cycle this work was primarily focused in two areas: using more of the LLVM-based toolchain where possible and improving support for a modern GNU toolchain. As a result of these changes, developers began to use features exclusive to modern toolchains near the end of the 12.0 development cycle.

Expanding LLVM Toolchain Use
FreeBSD 10.0 and 11.0 used the clang C and C++ compiler as the default system compiler for the x86 and little-endian ARM architectures. However, the object files generated by clang were linked into binaries and executables by the GNU BFD linker. For x86 and 32-bit ARM, the legacy GPLv2 linker in the base system was used. For 64-bit ARM, a GPLv3 linker had to be installed from ports.

Over the past few years the LLVM developers have made substantial improvements to the LLD linker. FreeBSD developers have contributed to this effort both with patches and also by using FreeBSD’s base system and ports as a large testing base to flesh out bugs and missing features. As a result of this work, FreeBSD 12.0 now ships LLD as the linker for 64-bit x86, 64-bit ARM, and ARMv7 architectures replacing the use of the GNU BFD linker. For 32-bit x86 and 32-bit ARM, the legacy GPLv2 linker in the base system was used. For 64-bit ARM, a GPLv3 linker had to be installed from ports.

External GNU Toolchain
Architectures not currently supported by the LLVM toolchain also need to transition to a more modern toolchain. Newer architectures such as RISC-V are not supported by the GPLv2 toolchain in the FreeBSD tree. In addition, building the base system with a modern GNU toolchain for architectures supported by LLVM provides users with a choice in toolchains. Rather than maintaining a GPLv3-licensed toolchain in the base source tree, modern GNU toolchains are built as separate packages using the ports framework.

 GNU toolchain packages come in two varieties. The first set of packages installs GCC and binutils as an additional toolchain in /usr/local and can be used for either native or cross builds. The second set of packages builds a base system compiler that installs GCC and binutils into /usr as the default toolchain.

The additional toolchain packages consist of three separate packages for each architecture: arch-binutils, arch-gcc, and arch-xtoolchain-gcc. The last package depends on the other two packages, and all of these packages are built from ports in the devel category. Once an external toolchain is installed, it can be used to build kernels and the base system via the CROSS_TOOLCHAIN make variable. The value passed to CROSS_TOOLCHAIN is “arch-gcc”.

For example, to build a 32-bit MIPS world, one would perform the steps in the following example.
The base system packages consist of two packages: *freebsd-binutils* and *freebsd-gcc*. These packages are built from the *base/binutils* and *base/gcc* ports. Unlike the additional toolchain packages, these packages replace components in the base system toolchain such as `/usr/bin/cc` and `/usr/bin/ld`. The ports for these packages (along with `pkg(8)` itself) can be cross-built from a non-native host. This will permit the Project to provide toolchain packages even on architectures for which the Project does not provide full package repositories.

Even when using a GNU toolchain, many toolchain components are still provided from other sources. For example, all FreeBSD architectures with a modern toolchain use `libc++` from LLVM as the C++ runtime library. Utilities such as `strip(8)` and `objcopy(8)` are provided by the ELF Tool Chain project.

FreeBSD 11 included support for additional toolchain packages and *CROSS_TOOLCHAIN*. During the FreeBSD 12 development cycle, work has focused on further refining this support. For example, the support for the `--sysroot` flag has been improved by both patches and configuration changes to the toolchain packages. In addition, the build system was updated to be more friendly to external toolchains with changes such as using the compiler driver to link binaries whenever possible and supporting different MIPS ABIs such as N32.

The base system toolchain packages have also been under active development over the past two years. Support has been added for the MIPS and x86 architectures. The same fixes for `--sysroot` support applicable to the additional toolchain packages also fixed similar issues with the base system packages. While they are not yet in a state to replace the legacy GPLv2 toolchain for any architectures in FreeBSD 12.0, developers have been able to build and boot a self-hosted world and kernel on 32-bit MIPS.

### Using Modern Toolchain Features

One of the benefits of moving to modern toolchains is the ability to use new toolchain features in the base system. Much of the work on toolchains prior to FreeBSD 12 focused on bringing on supporting a permissively-licensed toolchain on x86 architectures as well as supporting new architectures such as 64-bit ARM. However, FreeBSD was still treating the legacy GPLv2 toolchain as the lowest-common-denominator for deciding which toolchain features the base system used.

Toward the end of the FreeBSD 12 development cycle, this focus has shifted. As LLDB has matured, FreeBSD has achieved the goal of a permissively-licensed toolchain on the ARM and x86 architectures. The base system toolchain packages have also been under active development over the past two years. Support has been added for the MIPS and x86 architectures. The same fixes for `--sysroot` support applicable to the additional toolchain packages also fixed similar issues with the base system packages. While they are not yet in a state to replace the legacy GPLv2 toolchain for any architectures in FreeBSD 12.0, developers have been able to build and boot a self-hosted world and kernel on 32-bit MIPS.

### Conclusion

FreeBSD 12.0 marks another milestone in toolchain development. The ARM and x86 architectures now use modern, permissively licensed compilers and linkers. Support for external GCC toolchains is maturing. FreeBSD 13 will no longer use GPLv2 bits in the base system toolchain on any architectures. As a result, FreeBSD developers will accelerate the adoption of new toolchain features in the future. This will range from expanding the use of indirect functions, to enabling new features such as link-time optimization (LTO), build identifiers, compressed debug information, and more.

---

**JOHN BALDWIN** is a systems software developer. He has directly committed changes to the FreeBSD operating system for 19 years across various parts of the kernel (including x86 platform support, SMP, various device drivers, and the virtual memory subsystem) and userspace programs. In addition to writing code, John has served on the FreeBSD core and release engineering teams. He has also contributed to the GDB debugger and LLVM. John lives in Concord, California, with his wife, Kimberly, and three children: Janelle, Evan, and Bella.

**ED MASTE** manages project development for the FreeBSD Foundation and works in an engineering support role with the University of Cambridge Computer Laboratory. He is also a member of the elected FreeBSD core team. Aside from FreeBSD and LLVM projects, he is a contributor to several other open-source projects. He lives in Kitchener, Canada, with his wife, Anna, and sons, Pieter and Daniel.