Go 1.24

@peterhellberg

*"We expect **almost all** Go programs to continue to compile and run as before."*

Changes to the language

Go **1.24** now fully supports generic type aliases

(a type alias may be parameterized like a defined type)

Tools

GO command

Go command

Go modules can now track executable dependencies using `tool` directives in `go.mod`

The `go tool` command can now run these tools in addition to tools shipped with the Go distribution.

Go command

The new `tool` **meta-pattern** refers to all tools in the current module.

This can be used to upgrade them all with

```
go get tool
```

Cgo

Cgo

Cgo supports new annotations for C functions to improve run time performance.

`#cgo noescape cFunctionName` tells the compiler that memory passed to the C function cFunctionname does not escape.

`#cgo nocallback cFunctionName` tells the compiler that the C function cFunctionName does not call back to any Go functions.


Vet

Vet

The new `tests` analyzer reports **common mistakes** in declarations of tests, fuzzers, benchmarks, and examples in test packages, such as malformed names, incorrect signatures, or examples that document non-existent identifiers.

Runtime

Runtime

Several performance improvements to the runtime have **decreased** CPU overheads by **2–3%** on average across a suite of representative benchmarks.

Results may vary by application.

Compiler

Compiler

The compiler already disallowed defining new methods with receiver types that were cgo-generated, but it was possible to **circumvent** that restriction via an alias type.

Linker

Linker

The linker now generates a GNU build ID *(the ELF* ***`NT_GNU_BUILD_ID`*** *note)* on ELF platforms and a **`UUID`** *(the Mach-O* ***`LC_UUID`*** *load command)* on macOS by default.

The build ID or UUID is derived from the Go build ID.

It can be disabled by the **`-B none`** linker flag, or overridden by the **`-B 0xNNNN`** linker flag with a **user-specified hexadecimal value**.

Standard library

Directory-limited filesystem access

- The new **`os.Root`** type provides the ability to perform filesystem operations within a *specific* directory.
- The **`os.OpenRoot`** function opens a directory and returns an **`os.Root`**.

  Methods on **`os.Root`** operate within the directory and **do not** permit paths that refer to locations outside the directory, including ones that follow symbolic links out of the directory.

New benchmark function

Benchmarks may now use the faster and less error-prone **`testing.B.Loop`** method to perform benchmark iterations like **`for b.Loop() { ... }`** in place of the typical loop structures involving **`b.N`** like **`for range b.N`**.

New benchmark function

This offers two significant advantages:

- The benchmark function will execute **exactly once** per **`-count`**, so expensive setup and cleanup steps execute only once.
- Function call parameters and results are kept alive, preventing the compiler from fully optimizing away the loop body.

Improved finalizers

The new **`runtime.AddCleanup`** function is a finalization mechanism that is more flexible, more efficient, and less error-prone than **`runtime.SetFinalizer`**.

New weak package

The new **`weak`** package provides weak pointers.

Weak pointers are a low-level primitive provided to enable the creation of memory-efficient structures, such as weak maps for associating values, canonicalization maps for anything not covered by package **`unique`**, and various kinds of caches.

New crypto/mlkem package

The new **`crypto/mlkem`** package implements **ML-KEM-768** and **ML-KEM-1024**.

ML-KEM is a post-quantum key exchange mechanism formerly known as **Kyber** and specified in *FIPS 203*.

New crypto/hkdf, crypto/pbkdf2, and crypto/sha3 packages

- The new **`crypto/hkdf`** package implements the HMAC-based Extract-and-Expand key derivation function HKDF, as defined in RFC 5869.
- The new **`crypto/pbkdf2`** package implements the password-based key derivation function PBKDF2, as defined in RFC 8018.
- The new **`crypto/sha3`** package implements the SHA-3 hash function and SHAKE and cSHAKE extendable-output functions, as defined in FIPS 202.

FIPS 140-3 compliance

This release includes a new set of mechanisms to facilitate **FIPS 140-3 compliance**.

New experimental testing/synctest package

The new *experimental* **`testing/synctest`** package provides support for testing concurrent code.

Minor changes to the library

As usual, there has been **minor** changes to a number of standard library packages, those packages are:

archive bytes crypto/aes crypto/cipher
crypto/ecdsa crypto/md5 crypto/rand crypto/rsa
crypto/sha1 crypto/sha256 crypto/sha512
crypto/subtle crypto/tls crypto/x509 debug/elf
encoding encoding/json go/types hash/adler32
hash/crc32 hash/crc64 hash/fnv hash/maphash
log/slog math/big math/rand math/rand/v2 net
net/http net/netip net/url os/user regexp runtime
strings sync testing text/template time

Ports

Ports

- **Linux**: Go 1.24 requires Linux kernel version 3.2 or later.
- **Darwin**: Go 1.24 is the last release that will run on macOS 11 Big Sur.

  Go 1.25 will require macOS 12 Monterey or later.
- **WASM**: The **`go:wasmexport`** compiler directive is added for Go programs to **export** functions to the WebAssembly host.

  On WebAssembly System Interface Preview 1 (**`GOOS=wasip1 GOARCH=wasm`**),

  Go 1.24 supports building a Go program as a **`reactor/library`**, by specifying the **`-buildmode=c-shared`** build flag.
- **Windows**: The 32-bit windows/arm port (**`GOOS=windows GOARCH=arm`**) has been marked broken.

Learn more

Please read the Go 1.24 Release Notes

https://go.dev/doc/go1.24