Go 1.5 Release Features Blog

stack.ch

Die stack.ch Plattform wurde problemlos mit der neuen Go 1.5 Version kompiliert und deployed.

Übersicht Go 1.5

 Die neueste Go-Version, Version 1.5, ist eine bedeutende Version, einschliesslich wichtiger architektonischer Änderungen an der Implementierung. Trotzdem erwarten wir, dass fast alle Go-Programme weiterhin wie zuvor kompiliert und ausgeführt werden, da die Version weiterhin das Kompatibilitätsversprechen von Go 1 einhält.
Die grössten Entwicklungen bei der Implementierung sind:

Der Compiler und die Laufzeit sind jetzt vollständig in Go geschrieben (mit einem kleinen Assembler). C ist nicht mehr an der Implementierung beteiligt, sodass der C-Compiler, der einst zum Erstellen der Distribution erforderlich war, nicht mehr vorhanden ist.
Der Garbage Collector ist jetzt gleichzeitig verfügbar und bietet erheblich kürzere Pausenzeiten, indem er, wenn möglich, parallel zu anderen Goroutinen ausgeführt wird.
Standardmässig werden Go-Programme ausgeführt, wobei GOMAXPROCS die Anzahl der verfügbaren Kerne festgelegt ist. In früheren Versionen war der Standardwert 1.
Die Unterstützung für interne Pakete wird jetzt für alle Repositorys bereitgestellt, nicht nur für den Go-Kern.
Der go Befehl bietet jetzt experimentelle Unterstützung für das "Vendoring" externer Abhängigkeiten.
Ein neuer go tool trace Befehl unterstützt die detaillierte Verfolgung der Programmausführung.
Ein neuer go doc Befehl (im Unterschied zu godoc) wird für die Verwendung in der Befehlszeile angepasst.

Diese und eine Reihe anderer Änderungen an der Implementierung und den Tools werden nachstehend erläutert.
Die Version enthält auch eine kleine Sprachänderung mit Map literals.

Sprachänderungen

Aufgrund eines Versehens wurde die Regel, nach der der Elementtyp aus Slice-Literalen entfernt werden konnte, nicht auf Zuordnungsschlüssel angewendet. Dies wurde in Go 1.5 korrigiert. Ein Beispiel wird dies verdeutlichen. Ab Go 1.5 ist dieses Map Literal:m := map[Point]string {
 Point{29.935523, 52.891566}: "Persepolis",
  Point{-25.352594, 131.034361}: "Uluru",
  Point{37.422455, -122.084306}: "Googleplex",
}kann wie folgt geschrieben werden, ohne den explizit aufgeführten Typ Point: m := map[Point]string {
 {29.935523, 52.891566}: "Persepolis",
  {-25.352594, 131.034361}: "Uluru",
  {37.422455, -122.084306}: "Googleplex",
}

Implementation

No more CDer Compiler und die Laufzeit sind jetzt in Go und Assembler ohne C implementiert. Die einzige im Tree verbleibende C-Quelle bezieht sich auf das Testen oder auf cgo. In Go 1.4 und früheren Versionen befand sich ein C-Compiler im Tree. Er wurde verwendet, um die Laufzeit zu erstellen; Zum Teil war ein benutzerdefinierter Compiler erforderlich, um sicherzustellen, dass der C-Code mit der Stapelverwaltung von Goroutinen funktioniert. Da die Laufzeit jetzt in Go ist, wird dieser C-Compiler nicht benötigt und ist weg. Die Konvertierung von C wurde mit Hilfe von benutzerdefinierten Tools durchgeführt, die für den Job erstellt wurden. Am wichtigsten ist, dass der Compiler tatsächlich durch automatische Übersetzung des C-Codes in Go verschoben wurde. Es ist praktisch das gleiche Programm in einer anderen Sprache. Es handelt sich nicht um eine neue Implementierung des Compilers, daher erwarten wir, dass der Prozess keine neuen Compiler-Fehler verursacht hat. Compiler und ToolsUnabhängig von der Umstellung auf Go haben sich die Namen der Tools geändert. Die alten Namen 6g, 8g ... sind weg; Stattdessen gibt es nur eine Binärdatei, auf die zugegriffen werden go tool compile kann, um die Go-Quelle in Binärdateien zu kompilieren, die für die durch $GOARCH und $GOOS angegebene Architektur und Betriebssystem geeignet sind. Ebenso gibt es jetzt einen Linker (go tool link) und einen Assembler (go tool asm). Der Linker wurde automatisch aus der alten C-Implementierung übersetzt, aber der Assembler ist eine neue native Go-Implementierung, die weiter unten ausführlicher beschrieben wird.Ähnlich wie die Entfernung der Namen 6g, 8g ..., ist die Ausgabe des Compilers und Assembler nun alle mit dem .o Suffix statt .8, .6 etc. gegeben. Garbage CollectorDer Garbage Collector wurde für die Version 1.5 als Teil der Entwicklung des skizzierte Re-engineered Designdokument. Erwartete Latenzen sind viel niedriger als mit dem Collector in früheren Versionen durch eine Kombination von

fortschrittlichen Algorithmen, besserem Scheduling und durch die parallele Ausführung. Die "Stop the World" -Phase des Collectors liegt fast immer unter 10 Millisekunden oder deutlich darunter.Bei Systemen, die von einer geringen Latenz profitieren, z. B. Websites, die auf Benutzer reagieren, kann der Rückgang der erwarteten Latenz mit dem neuen Collector wichtig sein.RuntimeIn Go 1.5 wurde die Reihenfolge geändert, in der Goroutinen ausgeführt werden. Die Eigenschaften des Schedulers wurden nie durch die Sprache selbst definiert, aber Programme, die von der geplanten Ausführungsreihenfolge abhängen, können durch diese Änderung beeinträchtigt werden. Wir haben einige (fehlerhafte) Programme gesehen, die von dieser Änderung betroffen sind. Wenn Sie Programme haben, die implizit von der Ausführungsreihenfolge abhängen, müssen Sie diese anpassen.Eine weitere wichtige Änderung besteht darin, dass die Laufzeit jetzt die Standardanzahl der gleichzeitig ausgeführten Threads GOMAXPROCS auf die Anzahl der auf der CPU verfügbaren Kerne festlegt. In früheren Versionen war der Standardwert 1. Programme, die nicht mit mehreren CPU Kernen ausgeführt werden sollen, können versehentlich unterbrochen werden. Sie können die Einschränkung aufgeheben oder GOMAXPROCS explizit festgelegen. BuildNachdem der Go-Compiler und die Runtime in Go implementiert sind, muss ein Go-Compiler verfügbar sein, um die Distribution aus dem Quellcode zu kompilieren. Um den Go-Kern aufzubauen, muss daher bereits eine funktionierende Go-Distribution vorhanden sein. (Go-Programmierer, die nicht am Core arbeiten, sind von dieser Änderung nicht betroffen.) Jede Go 1.4- oder spätere Distribution (einschliesslich gccgo) wird bereitgestellt.

Ports

Aufgrund der in der Praxis reduzierten Verwendung der 32-Bit-x86-Architektur wird die Anzahl der bereitgestellten Binärdownloads mit Go Version 1.5 reduziert. Eine Distribution für das OS X-Betriebssystem wird nur für die amd64 Architektur bereitgestellt. Ebenso funktionieren die Ports für Snow Leopard (Apple OS X 10.6) weiterhin, werden jedoch nicht mehr als Download freigegeben oder verwaltet, da Apple diese Version des Betriebssystems nicht mehr verwaltet. Ausserdem wird der dragonfly/386 Port überhaupt nicht mehr unterstützt, da DragonflyBSD selbst die 32-Bit-386-Architektur nicht mehr unterstützt.Es stehen jedoch mehrere neue Ports zur Verfügung, die aus dem Quellcode erstellt werden können. Dazu gehören darwin/armund und darwin/arm64. Der neue Port linux/arm64 ist grösstenteils vorhanden, wird jedoch cgo nur über external Linking unterstützten.Ebenfalls als Experimente erhältlich sind ppc64 und ppc64le (64-Bit-PowerPC, Big- und Little-Endian). Beide Ports unterstützen cgo jedoch nur mit internal Linking.Unter FreeBSD benötigt Go 1.5 FreeBSD 8-STABLE+, da die SYSCALL Anweisung neu verwendet wird.Für NaCl benötigt Go 1.5 die SDK-Version Pepper-41. Spätere Pfefferversionen sind nicht kompatibel, da das sRPC-Subsystem aus der NaCl-Laufzeit entfernt wurde.Unter Darwin kann die Verwendung der System X.509-Zertifikatschnittstelle mit dem ios Build-Tag deaktiviert werden .Der Solaris-Port hat jetzt volle Unterstützung für CGO und die Pakete net und crypto/x509 sowie eine Reihe weiterer Korrekturen und Verbesserungen.

Tools

TranslatingAls Teil des Prozesses zum Entfernen von C aus dem Tree wurden der Compiler und der Linker von C nach Go übersetzt. Es war eine echte (maschinengestützte) Übersetzung, daher sind die neuen Programme im Wesentlichen die alten Programme und keine neuen mit neuen Fehlern. Wir sind zuversichtlich, dass der Übersetzungsprozess nur wenige oder gar keine neuen Fehler verursacht und tatsächlich eine Reihe bisher unbekannter Fehler aufgedeckt hat, die jetzt behoben wurden. Der Assembler ist jedoch ein neues Programm; es wird unten beschrieben.RenamingDie Compiler (6g, 8g usw.), die Assembler (6a, 8a usw.), und die Linker (6l, 8l usw.) wurden im gleichen Werkzeug kombiniert, die durch die Umgebungsvariablen konfiguriert ist, GOOS und GOARCH. Die alten Namen sind weg; die neuen Werkzeuge sind durch die verfügbaren go tool Funktionen wie go tool compile, go tool asm und go tool link. Die Dateiendungen .6, .8 usw. wurden entfernt; Jetzt sind es nur noch einfache .o Dateien. Wenn Sie beispielsweise ein Programm auf amd64 für Darwin erstellen und linken möchten ohne go build sondern durch die direkte Anwendung der Tools so führen Sie das folgende Script aus: $ export GOOS=darwin GOARCH=amd64
$ go tool compile program.go
$ go tool link program.oMovingDa das go/types Paket nun in das Main-Repository verschoben wurde, wurden die vet und cover Werkzeuge auch verschoben. Sie werden nicht mehr im externen golang.org/x/tools Repository verwaltet, obwohl sich die (veraltete) Quelle aus Gründen der Kompatibilität mit alten Versionen noch dort befindet. CompilerWie oben beschrieben, ist der Compiler in Go 1.5 ein einzelnes Go-Programm, von der alten C-Quelle übersetzt, solches ersetzt 6g, 8g und so weiter. Das Ziel wird durch die Umgebungsvariablen GOOS

und konfiguriert GOARCH.Der 1.5-Compiler entspricht grösstenteils dem alten, aber einige interne Details haben sich geändert. Eine wesentliche Änderung besteht darin, dass bei der Auswertung von Konstanten jetzt das math/big Paket anstelle einer benutzerdefinierten (und weniger gut getesteten) Implementierung hochpräziser Arithmetik verwendet wird. Wir erwarten nicht, dass dies die Ergebnisse beeinflusst.Nur für die amd64-Architektur verfügt der Compiler über eine neue Option -dynlink, welche die dynamische Verknüpfung unterstützt, indem Verweise auf Go-Symbole unterstützt werden, die in externen gemeinsam genutzten Bibliotheken definiert sind.
AssemblerAnalog dem Compiler und Linker ist der Assembler in Go 1.5 ein einzelnes Programm, welches die Assembler 6a, 8a usw. ersetzt. Die Umgebungsvariablen GOARCH und GOOS konfigurieren die Ziel-Architektur und das -Betriebssystem. Im Gegensatz zu den anderen Programmen ist der Assembler ein völlig neues Programm, das in Go geschrieben wurde.Der neue Assembler ist nahezu kompatibel mit den vorherigen, es gibt jedoch einige Änderungen, die sich auf einige Assembler-Quelldateien auswirken können. Weitere Informationen zu diesen Änderungen finden Sie im aktualisierten Assembler-Handbuch. Zusammenfassend:Erstens ist die für Konstanten verwendete Ausdrucksbewertung etwas anders. Es nutzt nun die unsigned 64-Bit-Arithmetik und die Priorisierung von Operatoren (+, -, <<, etc.) kommt von Go, nicht C.Vielleicht wichtiger ist, dass auf Computern, auf denen SP oder PC nur ein Alias ..sind wie z. B. R13 für den Stapelzeiger und R15 für den Hardwareprogrammzähler auf ARM, ein Verweis auf ein solches Register, das kein Symbol enthält, jetzt unzulässig ist. Zum Beispiel SP und 4(SP) sind illegal, aber sym+4(SP) in Ordnung. Verwenden Sie auf solchen Computern den echten RNamen, um auf das Hardwareregister zu verweisen.Eine kleine Änderung ist, dass einige der alten Assembler die folgende Notation zugelassen haben constant=valueum eine benannte Konstante zu definieren. Da dies immer mit der traditionellen C-ähnlichen #define Notation möglich ist, die weiterhin unterstützt wird (der Assembler enthält eine Implementierung eines vereinfachten C-Präprozessors), wurde die Funktion entfernt.LinkerDer Linker in Go 1.5 ist jetzt ein Go-Programm, das ersetzt 6l, 8l etc. Das Betriebssystem und der Befehlssatz sind durch die Umgebungsvariablen GOOS und GOARCH festgelegt. Es gibt mehrere andere Änderungen. Das wichtigste ist das Hinzufügen einer -buildmode Option, die den Verknüpfungsstil erweitert. Es unterstützt jetzt Situationen wie das Erstellen gemeinsam genutzter Bibliotheken und das Aufrufen anderer Sprachen in Go-Bibliotheken. Einige davon wurden in einem Designdokument beschrieben . Eine Liste der verfügbaren Build Modes finden Sie via $ go help buildmodeEine weitere kleine Änderung besteht darin, dass der Linker keine Build-Zeitstempel mehr im Header der ausführbaren Windows-Dateien aufzeichnet. Obwohl dies möglicherweise behoben ist, fehlen in den ausführbaren Dateien von Windows cgo einige DWARF-Informationen.Schließlich das -X Flag, das zwei Argumente akzeptiert, wie in -X Wert für importpath.nameAkzeptiert jetzt auch einen allgemeineren Go-Flag-Stil mit einem einzelnen Argument, das selbst ein name=valuePaar ist:-X importpath.name = valueObwohl die alte Syntax weiterhin funktioniert, wird empfohlen, die Verwendung dieses Flags in Skripten und dergleichen auf das neue Format anzupassen. Go CommandDer go Command bleibt unverändert, es gibt jedoch eine Reihe bemerkenswerter Änderungen.In der vorherigen Version wurde die Idee eingeführt, dass ein internes Verzeichnis eines Pakets über den go Befehl nicht importierbar ist. In 1.4 wurde es mit der Einführung einiger interner Elemente im Kern-Repository getestet. Wie im Entwurfsdokument vorgeschlagen, wird diese Änderung nun allen Repositorys zur Verfügung gestellt. Die Regeln werden im Entwurfsdokument erläutert. Zusammenfassend kann jedoch jedes Paket in oder unter einem Verzeichnis mit dem Namen internalvon Paketen importiert werden, die im selben Teilbaum verwurzelt sind. Bestehende Pakete mit den genannten Verzeichniselementen können intern durch diese Änderung versehentlich beschädigt werden, weshalb sie in der letzten Version angekündigt wurden.Eine weitere Änderung im Umgang mit Paketen ist die experimentelle Hinzufügung der Unterstützung für "Vendoring". Es gab auch einige weitere kleine Änderungen:SWIG Unterstützung wurde so aktualisiert, dass .swig und .swigcxx SWIG 3.0.6+ benötigen.Der install SubCommand entfernt jetzt die vom build SubCommand im Quellverzeichnis erstellte Binärdatei, falls vorhanden, um Probleme mit zwei Binärdateien im Baum zu vermeiden.Die std (Standard Library) Wildcard schliesst jetzt Befehle aus. Eine neue cmd Wildcard deckt die Command Befehle ab.Eine neue -asmflags Build-Option setzt Flags, die an den Assembler übergeben werden sollen. Die -ccflagsBuild-Option wurde jedoch entfernt. Solche war spezifisch für den alten, jetzt gelöschten C-Compiler.Eine neue -buildmode Build-Option legt den oben beschriebenen Build-Modus fest.Eine neue -pkgdir Build-Option legt den Speicherort der installierten Paketarchive fest, um benutzerdefinierte Builds zu isolieren.Eine neue -toolexec Build-Option ermöglicht das Ersetzen eines anderen Befehls zum Aufrufen des Compilers usw. Dies dient als benutzerdefinierter Ersatz für go tool.Der test SubCommand verfügt jetzt über ein -count Flag, mit dem angegeben wird, wie oft die einzelnen Tests und Benchmarks ausgeführt werden sollen. Das testing Paket erledigt die Arbeit hier durch das -test.count Flag.Der generate SubCommand verfügt über einige neue

Funktionen. Die -run Option gibt einen regulären Ausdruck an, um auszuwählen, welche Anweisungen ausgeführt werden sollen.Dies wurde vorgeschlagen, aber in 1.4 nie umgesetzt. Das Pattern hat jetzt Zugriff auf zwei neue Umgebungsvariablen: $GOLINE gibt die Quellzeilennummer der Direktive zurück und $DOLLAR repräsentiert ein Dollarzeichen.Der get SubCommand verfügt jetzt über ein -insecure Flag, das beim Abrufen aus einem unsicheren Repository aktiviert werden muss, das die Verbindung nicht verschlüsselt.go vet CommandDer go tool vet Befehl führt jetzt eine gründlichere Überprüfung der struct Tags durch.Trace CommandFür die dynamische Ablaufverfolgung von Go-Programmen steht ein neues Tool zur Verfügung. Die Verwendung entspricht der Funktionsweise des Testabdeckungstools. Die Generierung von Traces ist im go test Befehl integriert, und eine separate Ausführung des Tracing-Tools selbst analysiert die Ergebnisse:$ go test -trace = trace.out Pfad / zu / Paket
$ go tool trace [flags] pkg.test trace.outMit den Flags kann die Ausgabe in einem Browserfenster angezeigt werden. Für Details führen Sie go tool trace -help aus. go doc CommandVor einigen Releases wurde der go doc Befehl als unnötig gelöscht. Man könnte den Befehl via "godoc ." laufen lassen. Die Version 1.5 führt einen neuen go doc Befehl mit einer bequemeren Befehlszeilenschnittstelle als godoc's ein. Es wurde speziell für die Verwendung in der Befehlszeile entwickelt und bietet eine kompaktere und fokussiertere Darstellung der Dokumentation für ein Paket oder seine Elemente, je nach Aufruf. Es bietet auch ein Case Insensitives Matching und Unterstützung für die Anzeige nicht exportierter Symbole. Für Details führen Sie den Befehl "go help doc" aus. CgoBeim Parsen von #cgo Zeilen wird der Aufruf ${SRCDIR} jetzt in den Pfad zum Quellverzeichnis erweitert. Auf diese Weise können Optionen an den Compiler und den Linker übergeben werden, die Dateipfade relativ zum Quellcodeverzeichnis enthalten. Ohne die Erweiterung wären die Pfade ungültig, wenn sich das aktuelle Arbeitsverzeichnis ändert.Solaris bietet jetzt volle CGO-Unterstützung.Unter Windows verwendet cgo jetzt standardmäßig externe Links.Wenn eine C-struct mit einem zero-sized Feld endet, das struct selbst jedoch nicht zero-sized ist, kann sich der Go-Code nicht mehr auf das zero-sized Feld beziehen. Solche Referenzen müssen neu geschrieben werden.

Performance

Wie immer sind die Änderungen so allgemein und vielfältig, dass es schwierig ist, genaue Aussagen über die Leistung zu treffen. Die Änderungen sind noch umfassender als üblich in dieser Version, die einen neuen Garbage Collector und eine Konvertierung der Laufzeit in Go umfasst. Einige Programme werden möglicherweise schneller ausgeführt, andere langsamer. Im Durchschnitt laufen die Programme in der Go 1-Benchmark-Suite in Go 1.5 einige Prozent schneller als in Go 1.4, während die Pausen des Garbage Collectors, wie oben erwähnt, dramatisch kürzer sind und fast immer unter 10 Millisekunden liegen.Builds in Go 1.5 sind um den Faktor zwei langsamer. Die automatische Übersetzung des Compilers und Linkers von C nach Go führte zu einem unidiomatischen Go-Code, der im Vergleich zu gut geschriebenem Go eine schlechte Leistung erbringt. Analysetools und Refactoring haben zur Verbesserung des Codes beigetragen, aber es bleibt noch viel zu tun. Weitere Optimierungen werden in Go 1.6 und zukünftigen Versionen fortgesetzt.

Core Library

FlagDie Flag Package Funktion PrintDefaults und FlagSet Methoden wurden geändert und damit benutzerfreundlichere Messages erstellt. Das Format wurde geändert um die Benutzerfreundlichkeit zu verbessern. In den Messages wird ein mit "Backquotes" zitiertes Wort als Name des Operanden des Flags verwendet, der in der Message angezeigt werden soll. Zum Beispiel ein Flag, das mit dem Aufruf erstellt wurde:cpuFlag = flag.Int("cpu", 1, "run `N` processes in parallel")zeigt die Message:-cpu N. run N processes in parallel (default 1)Ausserdem wird der Standard jetzt nur aufgelistet, wenn es sich nicht um einen Nullwert handelt. Floats in math/bigDas math/big Package verfügt über einen neuen Basis Datentyp Float, der Gleitkommazahlen mit beliebiger Genauigkeit implementiert. Ein Float Wert wird durch ein Boolesches Vorzeichen, eine Mantisse variabler Länge und einen vorzeichenbehafteten 32-Bit-Exponenten fester Grösse dargestellt. Die Genauigkeit eines Float Wertes (Mantissengrösse in Bit) kann explizit angegeben werden oder wird auf andere Weise durch die erste Operation bestimmt, die den Wert erstellt. Einmal erstellt, kann die Präzision mit der SetPrec Methode geändert werden. Floats unterstützen das Konzept von Unendlichkeiten, wie sie durch Überlauf entstehen. Werte, die zum Äquivalent von IEEE 754 NaNs führen würden, lösen jedoch Panic aus. Float Operationen unterstützen alle IEEE-754-Rundungsmodi. Wenn die Genauigkeit auf 24 (53) Bit eingestellt ist, werden Operationen ausgeführt, die im Bereich von float32 (float64) bleiben, führen zu denselben Ergebnissen wie die

entsprechende IEEE-754-Arithmetik für diese Werte. Go TypesDas go/types Package wurde im golang.org/x Repository verwaltet. Ab Go 1.5 wurde es in das Haupt-Repository verschoben. Der Code am alten Speicherort ist jetzt veraltet. Es gibt auch eine kleine API-Änderung im Package, die unten erläutert wird.Im Zusammenhang mit dieser Verschiebung wurde das go/constant Package vom golang.org/x/tools/exact Package in das Hauptrepository verschoben. Das go/importer Package wurde auch in das Haupt-Repository sowie in einige der oben beschriebenen Tools verschoben. NetDer DNS-Resolver im net Package wurde fast immer für cgo den Zugriff auf die Systemschnittstelle verwendet. Eine Änderung in Go 1.5 bedeutet, dass auf den meisten Unix-Systemen keine DNS-Auflösung mit cgo mehr erforderlich is, was die Ausführung auf diesen Plattformen vereinfacht. Wenn die Netzwerkkonfiguration des Systems dies zulässt, reicht der native Go-Resolver aus. Der wichtige Effekt dieser Änderung besteht darin, dass jede DNS-Auflösung eher eine Goroutine als einen Thread belegt, sodass ein Programm mit mehreren ausstehenden DNS-Anforderungen weniger Betriebssystemressourcen verbraucht.Die Entscheidung, wie der Resolver ausgeführt werden soll, gilt zur Laufzeit und nicht zur Erstellungszeit. Das netgo Build-Tag, mit dem die Verwendung des Go-Resolvers erzwungen wurde, ist nicht mehr erforderlich, obwohl es weiterhin funktioniert. Ein neues netcgo Build-Tag erzwingt die Verwendung des cgo Resolvers zur Build-Zeit. Die Forcierung der cgo Auflösung kann mit der Umgebungsvariablen GODEBUG=netdns=cgo in der Umgebung festgelegten werden. Diese Änderung gilt nur für Unix-Systeme. Windows-, Mac OS X- und Plan 9-Systeme verhalten sich wie zuvor. ReflectDas reflect Package hat zwei neue Funktionen: ArrayOf und FuncOf. Diese Funktionen analog zu SliceOf erstellen zur Laufzeit neue Typen zur Beschreibung von Arrays und Funktionen. HardeningDurch randomisierte Tests mit dem go-fuzz Tool wurden mehrere Dutzend Fehler in der Standardbibliothek gefunden. Fehler wurden in archive/tar, archive/zip, compress/flate, encoding/gob, fmt, html/template, image/gif, vimage/jpeg, image/png, und text/template gefunden. Die Korrekturen schützen die Implementierung vor falschen und böswilligen Eingaben. Minor changes to the libraryDie archive/zip Package Writer verfügen nun über eine SetOffset Methode zum Angeben des Speicherorts innerhalb des Ausgabestreams, an dem das Archiv geschrieben werden soll.Der Reader im bufio Package verfügen nun über eine Methode zum Verwerfen von Daten aus der Eingabe.Im bytes Package verfügt der BufferTyp jetzt über eine Cap Methode, die die Anzahl der im Puffer zugewiesenen Bytes angibt. In ähnlicher Weise verfügt der Reader jetzt sowohl im bytes als auch im strings Package eine Size Methode, die die ursprüngliche Länge des zugrunde liegenden Slice oder Strings angibt.Sowohl das bytes als auch das strings Package haben jetzt eine LastIndexByte Funktion, die das Byte ganz rechts mit diesem Wert im Argument findet.Das crypto Package verfügt über das neue Interface Decrypter, die das Verhalten eines privaten Schlüssels abstrahiert, der bei der asymmetrischen Entschlüsselung verwendet wird.In dem crypto/cipher Package wurde die Dokumentation für die Stream Schnittstelle hinsichtlich des Verhaltens bei unterschiedlichen Längen von Quelle und Ziel präzisiert. Wenn das Ziel kürzer als die Quelle ist, wird panic ausgelöst. Dies ist keine Änderung in der Implementierung, sondern nur in der Dokumentation.Ausserdem enthält das crypto/cipher Package jetzt Unterstützung für andere Nonce-Längen als 96 Byte im Galois / Counter-Modus (GCM) von AES, die einige Protokolle erfordern.Im crypto/elliptic Package befindet sich jetzt ein Name Feld in der CurveParams Struktur, und den im Package implementierten Kurven wurden Namen gegeben. Diese Namen bieten eine sicherere Möglichkeit zur Auswahl einer Kurve als die Auswahl ihrer Bitgrösse für kryptografische Systeme, die kurvenabhängig sind.Ebenfalls im crypto/elliptic Package überprüft die Unmarshal Funktion jetzt, ob der Punkt tatsächlich auf der Kurve liegt. (Ist dies nicht der Fall, gibt die Funktion nils zurück.) Diese Änderung schützt vor bestimmten Angriffen.Das crypto/sha512 Package unterstützt jetzt die beiden abgeschnittenen Versionen des SHA-512-Hash-Algorithmus SHA-512/224 und SHA-512/256.Die crypto/tls Mindestversion des Paketprotokolls ist jetzt standardmässig TLS 1.0. Die alte Standardeinstellung SSLv3 ist Config bei Bedarf weiterhin verfügbar.Das crypto/tls Package unterstützt jetzt SCTs (Signed Certificate Timestamps) gemäss RFC 6962. Der Server stellt sie bereit, wenn sie in der Certificate Struktur aufgeführt sind, und der Client fordert sie an und macht sie, falls vorhanden, in seiner ConnectionState Struktur verfügbar.Die gestapelte OCSP-Antwort auf eine crypto/tls Clientverbindung, die bisher nur über die OCSPResponse Methode verfügbar war, wird jetzt in der ConnectionState Struktur verfügbar gemacht.Die crypto/tls Serverimplementierung ruft jetzt immer die GetCertificate Funktion in der ConfigStruktur auf, um ein Zertifikat für die Verbindung auszuwählen, wenn keines angegeben ist.Schliesslich können die Session Ticket Keys im crypto/tls Package jetzt geändert werden, während der Server ausgeführt wird. Dies erfolgt durch die neue SetSessionTicketKeys Methode des Config Typs.In der crypto/x509 Verpackung werden Platzhalter nur noch ganz links akzeptiert.Auch im crypto/x509 Package wurde die Behandlung unbekannter kritischer Erweiterungen geändert. Früher verursachten sie Analysefehler, jetzt werden sie analysiert und

verursachen nur ein VerifyDie DB Type des database/sql Packages hat nun eine Stats Methode um Datenbankstatistiken abzurufen.Das debug/dwarf Package enthält umfangreiche Ergänzungen, um DWARF Version 4 besser zu unterstützen. Siehe zum Beispiel die Definition des neuen Typs Class.Das debug/dwarf Package unterstützt jetzt auch die Dekodierung von DWARF-Zeilentabellen.Das debug/elf Package unterstützt jetzt die 64-Bit-PowerPC-Architektur.Das encoding/base64 Package unterstützt jetzt Unpadded Codierungen über zwei neue Codierungsvariablen RawStdEncoding und RawURLEncoding.Das encoding/json Package gibt jetzt einen UnmarshalTypeError zurück wenn ein JSON-Wert für die Zielvariable oder -komponente, für die er nicht gemarshallt wird, nicht geeignet ist.Der encoding/jsons Decoder Typ hat eine neue Methode, die zum Decodieren eines JSON Dokuments ein Streaming-Interface liefert: Token. Es arbeitet auch mit der vorhandenen Decode Funktionalität zusammen, wodurch eine bereits gestartete Dekodierungsoperation fortgesetzt wird Decoder.Token.Das flag Package verfügt über eine neue Funktion UnquoteUsage, die bei der Erstellung von Nutzungsnachrichten unter Verwendung der oben beschriebenen neuen Konvention hilft.In dem fmt Package gibt ein Wert vom Typ Value jetzt aus, was er enthält, anstatt die reflect.Value' Stringer Methode zu verwenden, die Dinge wie erzeugt.Der EmptyStmt Typ im go/ast Package verfügt jetzt über ein boolesches Implicit Feld, das aufzeichnet, ob das Semikolon implizit hinzugefügt wurde oder in der Quelle vorhanden war.Aus Gründen der Vorwärtskompatibilität reserviert das go/build Package GOARCH Werte für eine Reihe von Architekturen, die Go möglicherweise eines Tages unterstützt. Ob dies mal effektiv umgesetzt wird bleibt offen. Ausserdem verfügt die Package Struktur jetzt über ein PkgTargetRoot Feld, in dem das architekturabhängige Stammverzeichnis gespeichert wird, in dem es installiert werden soll, sofern bekannt.Im image Package unterstützt Rectangle jetzt das Image Interface, sodass Rectangle beim Zeichnen als Maske dienen kann.Um die Verarbeitung einiger JPEG-Bilder zu erleichtern, enthält das image Package jetzt Unterstützung für 4:1:1 und 4:1:0 YCbCr-Subsampling und grundlegende CMYK-Unterstützung, dargestellt durch die neue image.CMYK Struktur.Das image/color Package bietet grundlegende CMYK-Unterstützung durch die neue CMYK Struktur, das CMYKModel Farbmodell und die CMYKToRGB Funktion, wie sie von einigen JPEG-Bildern benötigt werden.Auch im image/color Package ist die Konvertierung eines YCbCr Wertes in RGBA präziser geworden. Zuvor waren die niedrigen 8 Bits nur ein Echo der hohen 8 Bits; Jetzt enthalten sie genauere Informationen. Aufgrund der Echo-Eigenschaft des alten Codes funktionierte die Operation uint8(r) zum Extrahieren eines 8-Bit-Rotwerts jedoch falsch. In Go 1.5 kann diese Operation einen anderen Wert ergeben. Der richtige Code ist und war immer die Auswahl der hohen 8 Bits : uint8(r>>8) Wertes. Im Übrigen bietet das image/draw Paket eine bessere Unterstützung für solche Konvertierungen.Ab Go 1.5 wird Alpha-Kanal Index beim nächsten Check-in berücksichtigt.Das image/gif Package enthält einige Verallgemeinerungen. Eine GIF-Datei mit mehreren Frames kann jetzt Gesamtgrenzen haben, die sich von allen Grenzen der enthaltenen Einzelbilder unterscheiden. Ausserdem verfügt die GIF Struktur jetzt über ein Disposal Feld, das die Destroy Methode für jeden Frame angibt.Das io Package fügt eine CopyBuffer analog der Copy Funktion hinzu, die einem vom Aufrufer bereitgestellten Puffer ähnelt, jedoch die Steuerung der Zuordnung und der Puffergrösse ermöglicht.Das log Package verfügt über ein neues LUTC Flag, mit dem Zeitstempel in der UTC-Zeitzone gedruckt werden. Ausserdem wird eine SetOutput Methode für vom Benutzer erstellte Logger hinzugefügt.In Go 1.4 Max wurden nicht alle möglichen NaN-Bitmuster erkannt. Dies ist in Go 1.5 behoben, sodass sich Programme, die math.Max Daten einschliesslich NaNs verwenden, möglicherweise anders verhalten, jetzt jedoch gemäss der IEEE754-Definition von NaNs korrekt.Das math/big Package fügt eine neue Jacobi Funktion für Ganzzahlen und eine neue ModSqrt Methode für den IntTyp hinzu.Das WordDecoderMIME Package fügt einen neuen Typ zum Dekodieren von MIME-Headern hinzu, die RFC 204-codierte Wörter enthalten. Es bietet auch BEncoding und QEncoding als Implementierungen der Codierungsschemata von RFC 2045 und RFC 2047.Das mime Package fügt ausserdem eine ExtensionsByType Funktion hinzu, die die MIME-Erweiterungen zurück gibt, von denen bekannt ist, dass sie einem bestimmten MIME-Typ zugeordnet sind.Es gibt ein neues mime/quotedprintable Package, das die in RFC 2045 definierte, in Anführungszeichen druckbare Codierung implementiert.Das net Package wird nun Dial Hostnamen erstellen, indem jede IP-Adresse der Reihe nach ausprobiert wird, bis eine erfolgreich ist. Der Dialer.DualStack Modus implementiert jetzt Happy Eyeballs (RFC 6555), indem der ersten Adressfamilie ein Vorsprung von 300 ms eingeräumt wird. Dieser Wert kann vom neuen Dialer.FallbackDelay überschrieben werden.Eine Reihe von Inkonsistenzen in den Typen, die durch Fehler im net Paket zurückgegeben wurden, sind behoben. Die meisten geben jetzt einen OpError Wert mit mehr Informationen als zuvor zurück. Ausserdem enthält der OpError Typ jetzt ein Source Feld, das die lokale Netzwerkadresse enthält.Das net/http Package unterstützt jetzt das Festlegen von Trailern von einem Server Handler. Einzelheiten finden Sie in der Dokumentation zu ResponseWriter.Ebenfalls

im net/http Package ist Code enthalten, um den Nullwert Time in der ServeContent Funktion zu ignorieren. Ab Go 1.5 wird jetzt auch ein Zeitwert ignoriert, der der Unix-Epoche entspricht.Das net/http/fcgi Package exportiert zwei neue Fehler ErrConnClosed und ErrRequestAborted, um die entsprechenden Fehlerbedingungen zu melden.Das net/http/cgi Package hatte einen Fehler, der die Werte der Umgebungsvariablen REMOTE_ADDR und REMOTE_HOST falsch behandelte. Dies wurde behoben. Ab Go 1.5 legt das Paket die REMOTE_PORT Variable fest.Das net/mail Package fügt einen AddressParser Typ hinzu, der E-Mail-Adressen analysieren kann.Das net/smtp Package verfügt jetzt über einen TLSConnectionState Accessor für den Client Typ, der den TLS-Status des Clients zurückgibt.Das os Package verfügt über eine neue LookupEnv Funktion, die Getenv einer leeren und einer fehlenden Umgebungsvariablen ähnelt, diese jedoch unterscheiden kann.Das os/signal Package fügt neue Ignore und Reset Funktionen hinzu.Die runtime, runtime/trace und net/http/pprof Packages enthalten neue Funktionen fürs Tracing wie: ReadTrace, StartTrace, StopTrace, Start, Stop und Trace.Das runtime/pprof Package enthält jetzt standardmässig die gesamte Speicherstatistik in allen Speicherprofilen.Das strings Package hat eine neue Compare Funktion. Solche ist als Pendant zum bytes Package vorhanden, ist jedoch ansonsten nicht erforderlich, da Zeichenfolgen den nativen Vergleich unterstützen.Die WaitGroup Implementierung im sync Package diagnostiziert jetzt Code bezogen auf einen Add Call mit Wait Return. Wenn dieser Zustand erkannt wird, gerät die Implementierung in Panik.Im syscall Package enthält die Linux-SysProcAttr Struktur jetzt ein GidMappingsEnableSetgroups Feld, das durch Sicherheitsänderungen in Linux 3.19 erforderlich wird. Auf allen Unix-Systemen verfügt die Struktur auch über neue Felder Foreground und Pgid Felder, um mehr Kontrolle bei der Ausführung zu bieten. Auf Darwin gibt es jetzt eine Syscall9 Funktion, um Aufrufe mit zu vielen Argumenten zu unterstützen.Das testing/quick generiert nun nil Werte für Zeigertypen und ermöglicht die Verwendung mit rekursiven Datenstrukturen. Ausserdem unterstützt das Package jetzt die Generierung von Array-Typen.In den text/template und html/template Packages, welche zu gross für die Darstellung sind wird nun ein Parser Fehler ausgelöst. Zuvor wurden sie stillschweigend in Gleitkomma umgewandelt, wodurch die Präzision verloren ging.Auch in den Paketen text/template und html/template ermöglicht eine neue Option Methode die Anpassung des Verhaltens des Templates während der Ausführung. Die implementierte Option ermöglicht die Kontrolle darüber, wie ein fehlender Schlüssel beim Indizieren der Map behandelt wird. Die Standardeinstellung, die jetzt überschrieben werden kann, lautet wie folgt: to continue with an invalid value.Der time Package Typ Time verfügt über eine neue Methode AppendFormat, mit der die Zuordnung beim Drucken eines Zeitwerts vermieden werden kann.Das unicode Package und die damit verbundene Unterstützung im gesamten System wurden von Version 7.0 auf Unicode 8.0 aktualisiert.

Feedback

War dieser Blog für Sie wertvoll. Wir danken für jede Anregung und Feedback

Kontakt

Simtech AG
Finkenweg 23
3110 Münsingen
Schweiz

Impressum

Das Copyright für sämtliche Inhalte dieser Website liegt bei Simtech AG, Schweiz.
Beachten Sie auch unsere Hinweise zum Urheberrecht, Datenschutz und Haftungsauschluss.
Jeder Hinweis auf Fehler nehmen wir gerne entgegen.

Copyright

2024 Simtech AG, All rights reserved, Powered by stack.ch written in Golang by Daniel Schmutz

https://www.simtech-ag.ch/repositorys