Forth am PC (und Android) — mit Gforth

Bernd Paysan

Forth–Tagung 2016, Augsburg

Inhalt

Forth–Merkmale mit Beispiel

Interactiv (Antwort in blau)

```
.( Hello World!) Hello World! ok
```

Für den Benutzer sichtbarer Stack (UPN) und sehr einfache Syntax (durch Leerzeichen abgetrennte Wörter)

```
3 4 + 6 * . 42  ok
```

Keine Trennung von Compile– und Runtime (inkrementeller Compiler)

```
: hi ( -- ) ." Hello World!" ; ok
hi Hello World! ok
```

Keine Typenprüfung

```
'a' 5 * . 485  ok
```

Forth–Merkmale mit Beispiel

Interactiv (Antwort in blau)

```
.( Hello World!) Hello World! ok
```

Für den Benutzer sichtbarer Stack (UPN) und sehr einfache Syntax (durch Leerzeichen abgetrennte Wörter)

```
3 4 + 6 * . 42  ok
```

Keine Trennung von Compile– und Runtime (inkrementeller Compiler)

```
: hi ( -- ) ." Hello World!" ; ok
hi Hello World! ok
```

Keine Typenprüfung

```
'a' 5 * . 485  ok
```

Forth–Merkmale mit Beispiel

Interactiv (Antwort in blau)

```
.( Hello World!) Hello World! ok
```

Für den Benutzer sichtbarer Stack (UPN) und sehr einfache Syntax (durch Leerzeichen abgetrennte Wörter)

```
3 4 + 6 * . 42  ok
```

Keine Trennung von Compile– und Runtime (inkrementeller Compiler)

```
: hi ( -- ) ." Hello World!" ; ok
hi Hello World! ok
```

Keine Typenprüfung

```
'a' 5 * . 485  ok
```

Forth–Merkmale mit Beispiel

Interactiv (Antwort in blau)

```
.( Hello World!) Hello World! ok
```

Für den Benutzer sichtbarer Stack (UPN) und sehr einfache Syntax (durch Leerzeichen abgetrennte Wörter)

```
3 4 + 6 * . 42  ok
```

Keine Trennung von Compile– und Runtime (inkrementeller Compiler)

```
: hi ( -- ) ." Hello World!" ; ok
hi Hello World! ok
```

Keine Typenprüfung

```
'a' 5 * . 485  ok
```

Die Forth–Prinzipien

Einfachheit Forth ist so einfach, dass man alle Aspekte verstehen kann

Abstraktion Forth ist low–level, man kann aber auch high level oder alles dazwischen implementieren: Unnötige Abstraktionen vermeiden, nötige selber machen

Ressourcensparend Forth ist kompakt, passt auch in ein paar kB Speicher

Erweiterbarkeit Forth liefert die Werkzeuge, man kann alles einbauen, was man braucht — auch Sprachkonzepte und Debug–Mittel.

Die Forth–Prinzipien

Einfachheit Forth ist so einfach, dass man alle Aspekte verstehen kann

Abstraktion Forth ist low–level, man kann aber auch high level oder alles dazwischen implementieren: Unnötige Abstraktionen vermeiden, nötige selber machen

Ressourcensparend Forth ist kompakt, passt auch in ein paar kB Speicher

Erweiterbarkeit Forth liefert die Werkzeuge, man kann alles einbauen, was man braucht — auch Sprachkonzepte und Debug–Mittel.

Die Forth–Prinzipien

Einfachheit Forth ist so einfach, dass man alle Aspekte verstehen kann

Abstraktion Forth ist low–level, man kann aber auch high level oder alles dazwischen implementieren: Unnötige Abstraktionen vermeiden, nötige selber machen

Ressourcensparend Forth ist kompakt, passt auch in ein paar kB Speicher

Erweiterbarkeit Forth liefert die Werkzeuge, man kann alles einbauen, was man braucht — auch Sprachkonzepte und Debug–Mittel.

Die Forth–Prinzipien

Einfachheit Forth ist so einfach, dass man alle Aspekte verstehen kann

Abstraktion Forth ist low–level, man kann aber auch high level oder alles dazwischen implementieren: Unnötige Abstraktionen vermeiden, nötige selber machen

Ressourcensparend Forth ist kompakt, passt auch in ein paar kB Speicher

Erweiterbarkeit Forth liefert die Werkzeuge, man kann alles einbauen, was man braucht — auch Sprachkonzepte und Debug–Mittel.


GNU

Beispiel für Lowlevel

Lowlevel

```
\ code snipped from embedded ethernet driver
$400EC000 constant EMACCFG
$3000480C constant MACMODE
: ether-on ( -- ) MACMODE EMACCFG ! ;
: emac-leds ( -- )
  \ Set Ethernet LEDs on Port F:
     $11 PORTF_AFSEL !
  $50005 PORTF_PCTL ! ;
```

Beispiel für Makros

Makros

```
: <map ]] cells bounds ?DO  i @ [[ ; immediate
: map> ]] cell +LOOP [[ ; immediate
: sumof ( addr cells -- ) 0 -rot <map + map> ;
see sumof
: sumof
   0 -rot cells bounds
   ?DO    i @ + 8
   +LOOP
   ; ok
Create testcase 1 , 2 , 3 , 4 ,
testcase 4 sumof . 10  ok
```

Erweiterbar: Minimalistisches OOP[2]


```
object class
  cell var text
  cell var len
  cell var x
  cell var y
  method init
  method draw
end-class button

:noname ( o -- ) >r
 r@ x @ r@ y @ at-xy  r@ text @ r> len @ type ;
 button defines draw
:noname ( addr u o -- ) >r
 0 r@ x ! 0 r@ y ! r@ len ! r> text ! ;
 button defines init
```

Mini–OOF: Implementierung


```
: method ( m v "name" -- m' v ) Create  over , swap cell+ swap
  DOES> ( ... o -- ... ) @ over @ + @ execute ;
: var ( m v size "name" -- m v' ) Create  over , +
  DOES> ( o -- addr ) @ + ;
: class ( class -- class methods vars ) dup 2@ ;
: end-class  ( class methods vars "name" -- )
  Create  here >r , dup , 2 cells ?DO ['] noop , 1 cells +LOOP
  cell+ dup cell+ r> rot @ 2 cells /string move ;
: >vt ( class "name" -- addr )  ' >body @ + ;
: bind ( class "name" -- xt )    >vt @ ;
: defines ( xt class "name" -- ) >vt ! ;
: new ( class -- o )  here over @ allot swap over ! ;
: :: ( class "name" -- ) bind compile, ;
Create object  1 cells , 2 cells ,
```

Ziele von Gforth[3]

Die Entwicklung von Gforth begann 1992, zwei Jahre, bevor der ANS Forth Standard offiziell wurde, hauptsächlich, weil die Forth–Gemeinschaft der Auffassung war, dass der Standard zu waage ist, und durch eine Referenzimplementierung konkretisiert werden sollte.

Das Ziel des Gforth–Projekts ist also eine Modell–Implementierung vom Forth–Standard. Das teilt sich in folgende Unterziele auf:

- Gforth sollte dem ANSI/2012–Standard von Forth entsprechen
- Es sollte ein Modell sein, also alle implementierungsabhängigen Details mit hoher Qualität festlegen
- Es sollte ein Standard–Modell sein, also breit akzeptiert und benutzt werden. Das ist der schwierigste Punkt.

Ziele von Gforth[3]

Die Entwicklung von Gforth begann 1992, zwei Jahre, bevor der ANS Forth Standard offiziell wurde, hauptsächlich, weil die Forth–Gemeinschaft der Auffassung war, dass der Standard zu waage ist, und durch eine Referenzimplementierung konkretisiert werden sollte.
Das Ziel des Gforth–Projekts ist also eine Modell–Implementierung vom Forth–Standard. Das teilt sich in folgende Unterziele auf:

- Gforth sollte dem ANSI/2012–Standard von Forth entsprechen
- Es sollte ein Modell sein, also alle implementierungsabhängigen Details mit hoher Qualität festlegen
- Es sollte ein Standard–Modell sein, also breit akzeptiert und benutzt werden. Das ist der schwierigste Punkt.

Ziele von Gforth[3]

Die Entwicklung von Gforth begann 1992, zwei Jahre, bevor der ANS Forth Standard offiziell wurde, hauptsächlich, weil die Forth–Gemeinschaft der Auffassung war, dass der Standard zu waage ist, und durch eine Referenzimplementierung konkretisiert werden sollte.
Das Ziel des Gforth–Projekts ist also eine Modell–Implementierung vom Forth–Standard. Das teilt sich in folgende Unterziele auf:

- Gforth sollte dem ANSI/2012–Standard von Forth entsprechen
- Es sollte ein Modell sein, also alle implementierungsabhängigen Details mit hoher Qualität festlegen
- Es sollte ein Standard–Modell sein, also breit akzeptiert und benutzt werden. Das ist der schwierigste Punkt.

Ziele von Gforth[3]

Die Entwicklung von Gforth begann 1992, zwei Jahre, bevor der ANS Forth Standard offiziell wurde, hauptsächlich, weil die Forth–Gemeinschaft der Auffassung war, dass der Standard zu waage ist, und durch eine Referenzimplementierung konkretisiert werden sollte.
Das Ziel des Gforth–Projekts ist also eine Modell–Implementierung vom Forth–Standard. Das teilt sich in folgende Unterziele auf:

- Gforth sollte dem ANSI/2012–Standard von Forth entsprechen
- Es sollte ein Modell sein, also alle implementierungsabhängigen Details mit hoher Qualität festlegen
- Es sollte ein Standard–Modell sein, also breit akzeptiert und benutzt werden. Das ist der schwierigste Punkt.

Ziele von Gforth[3]

Die Entwicklung von Gforth begann 1992, zwei Jahre, bevor der ANS Forth Standard offiziell wurde, hauptsächlich, weil die Forth–Gemeinschaft der Auffassung war, dass der Standard zu waage ist, und durch eine Referenzimplementierung konkretisiert werden sollte.
Das Ziel des Gforth–Projekts ist also eine Modell–Implementierung vom Forth–Standard. Das teilt sich in folgende Unterziele auf:

- Gforth sollte dem ANSI/2012–Standard von Forth entsprechen
- Es sollte ein Modell sein, also alle implementierungsabhängigen Details mit hoher Qualität festlegen
- Es sollte ein Standard–Modell sein, also breit akzeptiert und benutzt werden. Das ist der schwierigste Punkt.

Wo läuft Gforth

- Gforth läuft auf vielen Prozessoren: Alpha, ARM, x86(_64), IA64, PowerPC(64), SPARC(64), MIPS(64), HP–PA, M68K, und sogar S390. Überall, wo ein funktionierender GCC vorhanden ist, kann man Gforth laufen lassen.
- Betriebssysteme: GNU/Linux, *BSD, Mac OS X und andere proprietäre Unixe, Android, Windows mit Cygwin
- Gforth EC auf embedded Systemen: 6502, R8C, 8086, C165, 4stack, MISC, braucht 16kB Flash, und einige hundert Bytes RAM
- Gforth ist ein sehr populäres Forth–System (vielleicht sogar das populärste), aber nicht so dominant, dass Gforth–Erweiterungen de–facto–Standard sind

Wo läuft Gforth

- Gforth läuft auf vielen Prozessoren: Alpha, ARM, x86(_64), IA64, PowerPC(64), SPARC(64), MIPS(64), HP–PA, M68K, und sogar S390. Überall, wo ein funktionierender GCC vorhanden ist, kann man Gforth laufen lassen.
- Betriebssysteme: GNU/Linux, *BSD, Mac OS X und andere proprietäre Unixe, Android, Windows mit Cygwin
- Gforth EC auf embedded Systemen: 6502, R8C, 8086, C165, 4stack, MISC, braucht 16kB Flash, und einige hundert Bytes RAM
- Gforth ist ein sehr populäres Forth–System (vielleicht sogar das populärste), aber nicht so dominant, dass Gforth–Erweiterungen de–facto–Standard sind

Wo läuft Gforth

- Gforth läuft auf vielen Prozessoren: Alpha, ARM, x86(_64), IA64, PowerPC(64), SPARC(64), MIPS(64), HP–PA, M68K, und sogar S390. Überall, wo ein funktionierender GCC vorhanden ist, kann man Gforth laufen lassen.
- Betriebssysteme: GNU/Linux, *BSD, Mac OS X und andere proprietäre Unixe, Android, Windows mit Cygwin
- Gforth EC auf embedded Systemen: 6502, R8C, 8086, C165, 4stack, MISC, braucht 16kB Flash, und einige hundert Bytes RAM
- Gforth ist ein sehr populäres Forth–System (vielleicht sogar das populärste), aber nicht so dominant, dass Gforth–Erweiterungen de–facto–Standard sind

Wo läuft Gforth

- Gforth läuft auf vielen Prozessoren: Alpha, ARM, x86(_64), IA64, PowerPC(64), SPARC(64), MIPS(64), HP–PA, M68K, und sogar S390. Überall, wo ein funktionierender GCC vorhanden ist, kann man Gforth laufen lassen.
- Betriebssysteme: GNU/Linux, *BSD, Mac OS X und andere proprietäre Unixe, Android, Windows mit Cygwin
- Gforth EC auf embedded Systemen: 6502, R8C, 8086, C165, 4stack, MISC, braucht 16kB Flash, und einige hundert Bytes RAM
- Gforth ist ein sehr populäres Forth–System (vielleicht sogar das populärste), aber nicht so dominant, dass Gforth–Erweiterungen de–facto–Standard sind

Neuigkeiten für das nächste Gforth release

- Neue Plattform: Gforth auf Android (mit Java Native–Interface)
- Neue Wortheader, mit denen man den Compiler eleganter erweitern kann
- Recognizer ermöglichen, den äußeren Interpreter mit einfachen Plugins zu erweitern, also z.B. „echte“ String–Literale
- Ein SWIG–Module, um C Header automatisch in C–Bindings zu konvertieren

Neuigkeiten für das nächste Gforth release

- Neue Plattform: Gforth auf Android (mit Java Native–Interface)
- Neue Wortheader, mit denen man den Compiler eleganter erweitern kann
- Recognizer ermöglichen, den äußeren Interpreter mit einfachen Plugins zu erweitern, also z.B. „echte“ String–Literale
- Ein SWIG–Module, um C Header automatisch in C–Bindings zu konvertieren

Neuigkeiten für das nächste Gforth release

- Neue Plattform: Gforth auf Android (mit Java Native–Interface)
- Neue Wortheader, mit denen man den Compiler eleganter erweitern kann
- Recognizer ermöglichen, den äußeren Interpreter mit einfachen Plugins zu erweitern, also z.B. „echte“ String–Literale
- Ein SWIG–Module, um C Header automatisch in C–Bindings zu konvertieren

Neuigkeiten für das nächste Gforth release

- Neue Plattform: Gforth auf Android (mit Java Native–Interface)
- Neue Wortheader, mit denen man den Compiler eleganter erweitern kann
- Recognizer ermöglichen, den äußeren Interpreter mit einfachen Plugins zu erweitern, also z.B. „echte“ String–Literale
- Ein SWIG–Module, um C Header automatisch in C–Bindings zu konvertieren

Gforth auf Android

- Im App–Store nach Gforth googeln und einfach installieren
- Wer keinen App–Store haben möchte: `https://net2o.de/android/Gforth.apk` (auch alte Versionen).
- Das Terminal ist ein einfaches farbiges Text–Terminal, in OpenGL geschrieben.
- Es gibt noch ein paar andere kleine OpenGL– und JNI–Bespiele, etwa zum Auslesen der Sensoren
- Im Gforth ist Net2o drin, zum Ausprobieren, und auch als Beispiel, wie man eigenen Quelltext zur App hinzufügt

Gforth auf Android

- Im App–Store nach Gforth googeln und einfach installieren
- Wer keinen App–Store haben möchte: `https://net2o.de/android/Gforth.apk` (auch alte Versionen).
- Das Terminal ist ein einfaches farbiges Text–Terminal, in OpenGL geschrieben.
- Es gibt noch ein paar andere kleine OpenGL– und JNI–Bespiele, etwa zum Auslesen der Sensoren
- Im Gforth ist Net2o drin, zum Ausprobieren, und auch als Beispiel, wie man eigenen Quelltext zur App hinzufügt

Gforth auf Android

- Im App–Store nach Gforth googeln und einfach installieren
- Wer keinen App–Store haben möchte: `https://net2o.de/android/Gforth.apk` (auch alte Versionen).
- Das Terminal ist ein einfaches farbiges Text–Terminal, in OpenGL geschrieben.
- Es gibt noch ein paar andere kleine OpenGL– und JNI–Bespiele, etwa zum Auslesen der Sensoren
- Im Gforth ist Net2o drin, zum Ausprobieren, und auch als Beispiel, wie man eigenen Quelltext zur App hinzufügt

Gforth auf Android

- Im App–Store nach Gforth googeln und einfach installieren
- Wer keinen App–Store haben möchte: `https://net2o.de/android/Gforth.apk` (auch alte Versionen).
- Das Terminal ist ein einfaches farbiges Text–Terminal, in OpenGL geschrieben.
- Es gibt noch ein paar andere kleine OpenGL– und JNI–Bespiele, etwa zum Auslesen der Sensoren
- Im Gforth ist Net2o drin, zum Ausprobieren, und auch als Beispiel, wie man eigenen Quelltext zur App hinzufügt

Gforth auf Android

- Im App–Store nach Gforth googeln und einfach installieren
- Wer keinen App–Store haben möchte: `https://net2o.de/android/Gforth.apk` (auch alte Versionen).
- Das Terminal ist ein einfaches farbiges Text–Terminal, in OpenGL geschrieben.
- Es gibt noch ein paar andere kleine OpenGL– und JNI–Bespiele, etwa zum Auslesen der Sensoren
- Im Gforth ist Net2o drin, zum Ausprobieren, und auch als Beispiel, wie man eigenen Quelltext zur App hinzufügt

Neue Wortheader

- Individuelle Aktionen für COMPILE,: Gedacht für Optimierungen
- Individuelle Aktionen für TO, damit man verschiedenartige Values (FVALUE, 2VALUE aus Forth–2012, User–Values, Methoden in Mini–OOF2) implementieren kann
- Individuelle Aktionen für DEFER@, ebenfalls für Methoden in Mini–OOF2
- Der Name ist direkt vom Xt aus erreichbar
- Noch zu implementieren: Sourcecode–Informationen für LOCATE

Neue Wortheader

- Individuelle Aktionen für COMPILE,: Gedacht für Optimierungen
- Individuelle Aktionen für TO, damit man verschiedenartige Values (FVALUE, 2VALUE aus Forth–2012, User–Values, Methoden in Mini–OOF2) implementieren kann
- Individuelle Aktionen für DEFER@, ebenfalls für Methoden in Mini–OOF2
- Der Name ist direkt vom Xt aus erreichbar
- Noch zu implementieren: Sourcecode–Informationen für LOCATE

Neue Wortheader

- Individuelle Aktionen für COMPILE,: Gedacht für Optimierungen
- Individuelle Aktionen für TO, damit man verschiedenartige Values (FVALUE, 2VALUE aus Forth–2012, User–Values, Methoden in Mini–OOF2) implementieren kann
- Individuelle Aktionen für DEFER@, ebenfalls für Methoden in Mini–OOF2
- Der Name ist direkt vom Xt aus erreichbar
- Noch zu implementieren: Sourcecode–Informationen für LOCATE

Neue Wortheader

- Individuelle Aktionen für COMPILE,: Gedacht für Optimierungen
- Individuelle Aktionen für TO, damit man verschiedenartige Values (FVALUE, 2VALUE aus Forth–2012, User–Values, Methoden in Mini–OOF2) implementieren kann
- Individuelle Aktionen für DEFER@, ebenfalls für Methoden in Mini–OOF2
- Der Name ist direkt vom Xt aus erreichbar
- Noch zu implementieren: Sourcecode–Informationen für LOCATE

Neue Wortheader

- Individuelle Aktionen für COMPILE,: Gedacht für Optimierungen
- Individuelle Aktionen für TO, damit man verschiedenartige Values (FVALUE, 2VALUE aus Forth–2012, User–Values, Methoden in Mini–OOF2) implementieren kann
- Individuelle Aktionen für DEFER@, ebenfalls für Methoden in Mini–OOF2
- Der Name ist direkt vom Xt aus erreichbar
- Noch zu implementieren: Sourcecode–Informationen für LOCATE

Strings&Co.

- String–Erzeugung mit Higher–Order–Funktionen, die die Terminal–Worte umbiegen, also >STRING–EXECUTE, $EXEC und $TMP.
- Beispiel: String umdrehen

```
: .reverse ( addr u -- )
  over + \ start end
  BEGIN  2dup u<  WHILE
         dup xchar- tuck - \ start cur n
         over swap type    \ start cur
  REPEAT  2drop ;
: reverse ( addr1 u1 -- addr2 u2 )
  ['] .reverse $tmp ;
```

- Dynamische String–Arrays mit $[]@ und $[]!

Strings&Co.

- String–Erzeugung mit Higher–Order–Funktionen, die die Terminal–Worte umbiegen, also >STRING–EXECUTE, $EXEC und $TMP.
- Beispiel: String umdrehen

```
: .reverse ( addr u -- )
  over + \ start end
  BEGIN  2dup u<  WHILE
         dup xchar- tuck - \ start cur n
         over swap type    \ start cur
  REPEAT  2drop ;
: reverse ( addr1 u1 -- addr2 u2 )
  ['] .reverse $tmp ;
```

- Dynamische String–Arrays mit $[]@ und $[]!

Strings&Co.

- String–Erzeugung mit Higher–Order–Funktionen, die die Terminal–Worte umbiegen, also >STRING–EXECUTE, $EXEC und $TMP.
- Beispiel: String umdrehen

```
: .reverse ( addr u -- )
  over + \ start end
  BEGIN  2dup u<  WHILE
         dup xchar- tuck - \ start cur n
         over swap type    \ start cur
  REPEAT  2drop ;
: reverse ( addr1 u1 -- addr2 u2 )
  ['] .reverse $tmp ;
```

- Dynamische String–Arrays mit $[]@ und $[]!

Recognizer

- Wörter (alles durch Leerzeichen abgetrennte) kann man interpretieren, compilieren, oder diese Aktionen später durchführen (POSTPONE).
- Der Recognizer erkennt zur Parse–Zeit, ob er für das Wort zuständig ist
- Er gibt neben den aus dem Wort extrahierten Daten (Zahl, String oder Xt) noch eine Tabelle mit den drei Aktionen INT, COMP und POST zurück
- Für Dictionary–Suche, Integer und Floats ersetzt der Recognizer–Stack den starren Aufbau des äußeren Interpreters, man kann z.B. R:INT und R:FLOAT vertauschen, wenn man keine Doubles braucht
- Recognizer gibt es als Ersatz für „böse“ Parsing–Wörter, also ->VALUE als Ersatz für TO, und ”String” als Ersatz für S”
- Recognizer für VOKABULAR:WORD und für .METHOD in Mini–OOF2

Recognizer

- Wörter (alles durch Leerzeichen abgetrennte) kann man interpretieren, compilieren, oder diese Aktionen später durchführen (POSTPONE).
- Der Recognizer erkennt zur Parse–Zeit, ob er für das Wort zuständig ist
- Er gibt neben den aus dem Wort extrahierten Daten (Zahl, String oder Xt) noch eine Tabelle mit den drei Aktionen INT, COMP und POST zurück
- Für Dictionary–Suche, Integer und Floats ersetzt der Recognizer–Stack den starren Aufbau des äußeren Interpreters, man kann z.B. R:INT und R:FLOAT vertauschen, wenn man keine Doubles braucht
- Recognizer gibt es als Ersatz für „böse“ Parsing–Wörter, also ->VALUE als Ersatz für TO, und ”String” als Ersatz für S”
- Recognizer für VOKABULAR:WORD und für .METHOD in Mini–OOF2

Recognizer

- Wörter (alles durch Leerzeichen abgetrennte) kann man interpretieren, compilieren, oder diese Aktionen später durchführen (POSTPONE).
- Der Recognizer erkennt zur Parse–Zeit, ob er für das Wort zuständig ist
- Er gibt neben den aus dem Wort extrahierten Daten (Zahl, String oder Xt) noch eine Tabelle mit den drei Aktionen INT, COMP und POST zurück
- Für Dictionary–Suche, Integer und Floats ersetzt der Recognizer–Stack den starren Aufbau des äußeren Interpreters, man kann z.B. R:INT und R:FLOAT vertauschen, wenn man keine Doubles braucht
- Recognizer gibt es als Ersatz für „böse“ Parsing–Wörter, also ->VALUE als Ersatz für TO, und ”String” als Ersatz für S”
- Recognizer für VOKABULAR:WORD und für .METHOD in Mini–OOF2

Recognizer

- Wörter (alles durch Leerzeichen abgetrennte) kann man interpretieren, compilieren, oder diese Aktionen später durchführen (POSTPONE).
- Der Recognizer erkennt zur Parse–Zeit, ob er für das Wort zuständig ist
- Er gibt neben den aus dem Wort extrahierten Daten (Zahl, String oder Xt) noch eine Tabelle mit den drei Aktionen INT, COMP und POST zurück
- Für Dictionary–Suche, Integer und Floats ersetzt der Recognizer–Stack den starren Aufbau des äußeren Interpreters, man kann z.B. R:INT und R:FLOAT vertauschen, wenn man keine Doubles braucht
- Recognizer gibt es als Ersatz für „böse“ Parsing–Wörter, also ->VALUE als Ersatz für TO, und ”String” als Ersatz für S”
- Recognizer für VOKABULAR:WORD und für .METHOD in Mini–OOF2

Recognizer

- Wörter (alles durch Leerzeichen abgetrennte) kann man interpretieren, compilieren, oder diese Aktionen später durchführen (POSTPONE).
- Der Recognizer erkennt zur Parse–Zeit, ob er für das Wort zuständig ist
- Er gibt neben den aus dem Wort extrahierten Daten (Zahl, String oder Xt) noch eine Tabelle mit den drei Aktionen INT, COMP und POST zurück
- Für Dictionary–Suche, Integer und Floats ersetzt der Recognizer–Stack den starren Aufbau des äußeren Interpreters, man kann z.B. R:INT und R:FLOAT vertauschen, wenn man keine Doubles braucht
- Recognizer gibt es als Ersatz für „böse“ Parsing–Wörter, also ->VALUE als Ersatz für TO, und ”String” als Ersatz für S”
- Recognizer für VOKABULAR:WORD und für .METHOD in Mini–OOF2

Recognizer

- Wörter (alles durch Leerzeichen abgetrennte) kann man interpretieren, compilieren, oder diese Aktionen später durchführen (POSTPONE).
- Der Recognizer erkennt zur Parse–Zeit, ob er für das Wort zuständig ist
- Er gibt neben den aus dem Wort extrahierten Daten (Zahl, String oder Xt) noch eine Tabelle mit den drei Aktionen INT, COMP und POST zurück
- Für Dictionary–Suche, Integer und Floats ersetzt der Recognizer–Stack den starren Aufbau des äußeren Interpreters, man kann z.B. R:INT und R:FLOAT vertauschen, wenn man keine Doubles braucht
- Recognizer gibt es als Ersatz für „böse“ Parsing–Wörter, also ->VALUE als Ersatz für TO, und ”String” als Ersatz für S”
- Recognizer für VOKABULAR:WORD und für .METHOD in Mini–OOF2

Längerfristige Pläne

- Ersetze vmgen/GCC durch einen eigenen inkrementellen Compiler (JIT), zumindest für populäre Prozessoren wie amd64 und ARM
- Forth als portabler Assembler (PAF), damit auch andere Sprachen von dem Compiler profitieren können
- MINOS2 GUI library — OpenGL based, cross–platform, modernes look&feel, schlank und schnell
- Mehr Interfaces zu externen Libraries (C++? Objective–C für iOS&Mac?)

Längerfristige Pläne

- Ersetze vmgen/GCC durch einen eigenen inkrementellen Compiler (JIT), zumindest für populäre Prozessoren wie amd64 und ARM
- Forth als portabler Assembler (PAF), damit auch andere Sprachen von dem Compiler profitieren können
- MINOS2 GUI library — OpenGL based, cross–platform, modernes look&feel, schlank und schnell
- Mehr Interfaces zu externen Libraries (C++? Objective–C für iOS&Mac?)

Längerfristige Pläne

- Ersetze vmgen/GCC durch einen eigenen inkrementellen Compiler (JIT), zumindest für populäre Prozessoren wie amd64 und ARM
- Forth als portabler Assembler (PAF), damit auch andere Sprachen von dem Compiler profitieren können
- MINOS2 GUI library — OpenGL based, cross–platform, modernes look&feel, schlank und schnell
- Mehr Interfaces zu externen Libraries (C++? Objective–C für iOS&Mac?)

Längerfristige Pläne

- Ersetze vmgen/GCC durch einen eigenen inkrementellen Compiler (JIT), zumindest für populäre Prozessoren wie amd64 und ARM
- Forth als portabler Assembler (PAF), damit auch andere Sprachen von dem Compiler profitieren können
- MINOS2 GUI library — OpenGL based, cross–platform, modernes look&feel, schlank und schnell
- Mehr Interfaces zu externen Libraries (C++? Objective–C für iOS&Mac?)

Literatur I

Usenet

comp.lang.forth FAQ

http://www.complang.tuwien.ac.at/forth/faq/faq-general.html

Bernd Paysan

Detailed Description of Mini–OOF

https://bernd-paysan.de/mini-oof.html

Anton Ertl, Bernd Paysan et al

Gforth Homepage

https://www.gnu.org/software/gforth/