FORTH

A slightly different Programming System

Carsten Strotmann, Forth Gesellschaft e.V.

21st Chaos Communication Congress

Outline

1. Forth in Time
   - Birth of Forth and the roaring 80s
   - Forth Today
2. An Overview on Forth
   - Forth Basics
   - Examples (Forth vs. C)
   - Forth is easily extensible
3. Why learning/using Forth
   - Broaden your horizon
   - Create lean and fast Programs
   - Forth, the good and the bad
4. 21C3 - Forth Crossover

Outline

1. Forth in Time
   - Birth of Forth and the roaring 80s
   - Forth Today
2. An Overview on Forth
   - Forth Basics
   - Examples (Forth vs. C)
   - Forth is easily extensible
3. Why learning/using Forth
   - Broaden your horizon
   - Create lean and fast Programs
   - Forth, the good and the bad
4. 21C3 - Forth Crossover

BIRTH OF FORTH

The early years

- the first program to be called Forth was written in about 1970 by Charles Moore.
- An Open Source Forth called FIG-FORTH made Forth known on many computer enthusiasts on may architectures.
- In the late 70s and during the 80s, Forth was often used on Mirco- and Home-Computer Systems.

BIRTH OF FORTH

The early years

- the first program to be called Forth was written in about 1970 by Charles Moore.
- An Open Source Forth called FIG-FORTH made Forth known on many computer enthusiasts on may architectures.
- In the late 70s and during the 80s, Forth was often used on Mirco- and Home-Computer Systems.

BIRTH OF FORTH

The early years

- the first program to be called Forth was written in about 1970 by Charles Moore.
- An Open Source Forth called FIG-FORTH made Forth known on many computer enthusiasts on may architectures.
- In the late 70s and during the 80s, Forth was often used on Mirco- and Home-Computer Systems.

The Name

How did Forth get its name?

Charles Moore:

The first time I combined the ideas I had been developing into a single entity, I was working on an IBM 1130, a "third-generation" computer. The result seemed so powerful that I considered it a "fourth generation computer language." I would have called it Fourth, except that the 1130 permitted only five-character identifiers. So Fourth became Forth, a nicer play on words anyway.

Outline

1. Forth in Time
   - Birth of Forth and the roaring 80s
   - Forth Today
2. An Overview on Forth
   - Forth Basics
   - Examples (Forth vs. C)
   - Forth is easily extensible
3. Why learning/using Forth
   - Broaden your horizon
   - Create lean and fast Programs
   - Forth, the good and the bad
4. 21C3 - Forth Crossover

Forth today

- Forth is available for a wide range of architectures
  - native on bare hardware (i386-ia64, amd64, ppc, sparc ...)
  - hosted on Operating Systems (Linux/Unix, Dos, embedded OS, even Windows ...)
  - or embedded in other Systems (Java, .NET, JavaScript, BASH ...)
  - or as a Machine Language in Hardware (μCore, b16 ...)

Forth today

- Forth is available for a wide range of architectures
  - native on bare hardware (i386-ia64, amd64, ppc, sparc ...)
  - hosted on Operating Systems (Linux/Unix, Dos, embedded OS, even Windows ...)
  - or embedded in other Systems (Java, .NET, JavaScript, BASH ...)
  - or as a Machine Language in Hardware (μCore, b16 ...)

Forth today

- Forth is available for a wide range of architectures
  - native on bare hardware (i386-ia64, amd64, ppc, sparc ...)
  - hosted on Operating Systems (Linux/Unix, Dos, embedded OS, even Windows ...)
  - or embedded in other Systems (Java, .NET, JavaScript, BASH ...)
  - or as a Machine Language in Hardware (μCore, b16 ...)

Forth today

- Forth is available for a wide range of architectures
  - native on bare hardware (i386-ia64, amd64, ppc, sparc ...)
  - hosted on Operating Systems (Linux/Unix, Dos, embedded OS, even Windows ...)
  - or embedded in other Systems (Java, .NET, JavaScript, BASH ...)
  - or as a Machine Language in Hardware (μCore, b16 ...)

Forth today (2nd)

- Forth is often used “invisible”
  - as the Firmware (or BIOS) of all Apple and SUN Sparc Machines
  - as a Bootloader in FreeBSD
  - as embedded Software, e.g. in Vending Machines
  - or F1 Race Cars
  - or Satellites

Forth today (2nd)

- Forth is often used “invisible”
  - as the Firmware (or BIOS) of all Apple and SUN Sparc Machines
  - as a Bootloader in FreeBSD
  - as embedded Software, e.g. in Vending Machines
  - or F1 Race Cars
  - or Satellites

Forth today (2nd)

- Forth is often used “invisible”
  - as the Firmware (or BIOS) of all Apple and SUN Sparc Machines
  - as a Bootloader in FreeBSD
  - as embedded Software, e.g. in Vending Machines
  - or F1 Race Cars
  - or Satellites

Forth today (2nd)

- Forth is often used “invisible”
  - as the Firmware (or BIOS) of all Apple and SUN Sparc Machines
  - as a Bootloader in FreeBSD
  - as embedded Software, e.g. in Vending Machines
  - or F1 Race Cars
  - or Satellites

Forth today (2nd)

- Forth is often used “invisible”
  - as the Firmware (or BIOS) of all Apple and SUN Sparc Machines
  - as a Bootloader in FreeBSD
  - as embedded Software, e.g. in Vending Machines
  - or F1 Race Cars
  - or Satellites

Outline

1. Forth in Time
   - Birth of Forth and the roaring 80s
   - Forth Today
2. An Overview on Forth
   - Forth Basics
   - Examples (Forth vs. C)
   - Forth is easily extensible
3. Why learning/using Forth
   - Broaden your horizon
   - Create lean and fast Programs
   - Forth, the good and the bad
4. 21C3 - Forth Crossover

Forth Simplicity

- Forth has just two rules:
  1. You have NUMBERS (in any numeric base you could imagine). They are pushed on to the DATA STACK or compiled on to the DICTIONARY.
  2. You have SYMBOLS. They are searched through the DICTIONARY and, if found, EXECUTED or compiled on to the DICTIONARY too.
- Forth has two main modes: EXECUTE and COMPILE. Depending what mode you are in, the behavior of numbers and symbols changes accordingly.
- That's it. Everything else is about WORDS, that means, what one of the SYMBOLS does when EXECUTED.

Forth Simplicity

- Forth has just two rules:
  1. You have NUMBERS (in any numeric base you could imagine). They are pushed on to the DATA STACK or compiled on to the DICTIONARY.
  2. You have SYMBOLS. They are searched through the DICTIONARY and, if found, EXECUTED or compiled on to the DICTIONARY too.
- Forth has two main modes: EXECUTE and COMPILE. Depending what mode you are in, the behavior of numbers and symbols changes accordingly.
- That's it. Everything else is about WORDS, that means, what one of the SYMBOLS does when EXECUTED.

Forth Simplicity

- Forth has just two rules:
  1. You have NUMBERS (in any numeric base you could imagine). They are pushed on to the DATA STACK or compiled on to the DICTIONARY.
  2. You have SYMBOLS. They are searched through the DICTIONARY and, if found, EXECUTED or compiled on to the DICTIONARY too.
- Forth has two main modes: EXECUTE and COMPILE. Depending what mode you are in, the behavior of numbers and symbols changes accordingly.
- That's it. Everything else is about WORDS, that means, what one of the SYMBOLS does when EXECUTED.

Forth Simplicity

- Forth has just two rules:
  1. You have NUMBERS (in any numeric base you could imagine). They are pushed on to the DATA STACK or compiled on to the DICTIONARY.
  2. You have SYMBOLS. They are searched through the DICTIONARY and, if found, EXECUTED or compiled on to the DICTIONARY too.
- Forth has two main modes: EXECUTE and COMPILE. Depending what mode you are in, the behavior of numbers and symbols changes accordingly.
- That's it. Everything else is about WORDS, that means, what one of the SYMBOLS does when EXECUTED.

How it works

- Forth is a stack-based language based on PostfixNotation
  - all processing is done in “WORDS”, the Forth name for “Subroutines”
  - Operations and function calls (Words) are placed after their arguments
  - arguments are pushed on the data stack by the user or other “Words”, and then popped off the data stack by the “Words”, performing their operation,
  - and then pushing the results back on the data stack.
  - Forth Systems can be Interpreter or Native-Compiler, or both, or Cross-Compiler.

How it works

- Forth is a stack-based language based on PostfixNotation
  - all processing is done in “WORDS”, the Forth name for “Subroutines”
  - Operations and function calls (Words) are placed after their arguments
  - arguments are pushed on the data stack by the user or other “Words”, and then popped off the data stack by the “Words”, performing their operation,
  - and then pushing the results back on the data stack.
  - Forth Systems can be Interpreter or Native-Compiler, or both, or Cross-Compiler.

How it works

- Forth is a stack-based language based on PostfixNotation
  - all processing is done in “WORDS”, the Forth name for “Subroutines”
  - Operations and function calls (Words) are placed after their arguments
  - arguments are pushed on the data stack by the user or other “Words”, and then popped off the data stack by the “Words”, performing their operation,
  - and then pushing the results back on the data stack.
  - Forth Systems can be Interpreter or Native-Compiler, or both, or Cross-Compiler.

How it works

- Forth is a stack-based language based on PostfixNotation
  - all processing is done in “WORDS”, the Forth name for “Subroutines”
  - Operations and function calls (Words) are placed after their arguments
  - arguments are pushed on the data stack by the user or other “Words”, and then popped off the data stack by the “Words”, performing their operation,
  - and then pushing the results back on the data stack.
  - Forth Systems can be Interpreter or Native-Compiler, or both, or Cross-Compiler.

How it works

- Forth is a stack-based language based on PostfixNotation
  - all processing is done in “WORDS”, the Forth name for “Subroutines”
  - Operations and function calls (Words) are placed after their arguments
  - arguments are pushed on the data stack by the user or other “Words”, and then popped off the data stack by the “Words”, performing their operation,
  - and then pushing the results back on the data stack.
  - Forth Systems can be Interpreter or Native-Compiler, or both, or Cross-Compiler.

Extending Forth

- Forth Programmers create Applications by extending Forth
  - creating (compiling) new Words (there is no difference between supplied Forth Words and User-defined Words)
  - Programmers can choose to hide the Forth System in their application...
  - or choose to expose the System, e.g. as a powerful Macro Language

Extending Forth

- Forth Programmers create Applications by extending Forth
  - creating (compiling) new Words (there is no difference between supplied Forth Words and User-defined Words)
  - Programmers can choose to hide the Forth System in their application...
  - or choose to expose the System, e.g. as a powerful Macro Language

Extending Forth

- Forth Programmers create Applications by extending Forth
  - creating (compiling) new Words (there is no difference between supplied Forth Words and User-defined Words)
  - Programmers can choose to hide the Forth System in their application...
  - or choose to expose the System, e.g. as a powerful Macro Language

Forth is not (only) the tool

- Programming in Forth is somewhat different to “mainstream” languages
  - in Pascal, Perl, C/C++, Basic, there is a clear distinction between
    - the Tool (the Programming Language Compiler/Interpreter)
    - and the Application (Binary or Sourcecode)
  - in Forth, a Basic Forth System will be extended and changed to become the Application
  - = Tool and Application are not distinct

Forth is not (only) the tool

- Programming in Forth is somewhat different to “mainstream” languages
  - in Pascal, Perl, C/C++, Basic, there is a clear distinction between
    - the Tool (the Programming Language Compiler/Interpreter)
    - and the Application (Binary or Sourcecode)
  - in Forth, a Basic Forth System will be extended and changed to become the Application
  - = Tool and Application are not distinct

Forth is not (only) the tool

- Programming in Forth is somewhat different to “mainstream” languages
  - in Pascal, Perl, C/C++, Basic, there is a clear distinction between
    - the Tool (the Programming Language Compiler/Interpreter)
    - and the Application (Binary or Sourcecode)
  - in Forth, a Basic Forth System will be extended and changed to become the Application
  - = Tool and Application are not distinct

Forth is not (only) the tool

- Programming in Forth is somewhat different to “mainstream” languages
  - in Pascal, Perl, C/C++, Basic, there is a clear distinction between
    - the Tool (the Programming Language Compiler/Interpreter)
    - and the Application (Binary or Sourcecode)
  - in Forth, a Basic Forth System will be extended and changed to become the Application
  - = Tool and Application are not distinct

Forth is not (only) the tool

- Programming in Forth is somewhat different to “mainstream” languages
  - in Pascal, Perl, C/C++, Basic, there is a clear distinction between
    - the Tool (the Programming Language Compiler/Interpreter)
    - and the Application (Binary or Sourcecode)
  - in Forth, a Basic Forth System will be extended and changed to become the Application
  - = Tool and Application are not distinct

Forth is not (only) the tool

- Programming in Forth is somewhat different to “mainstream” languages
  - in Pascal, Perl, C/C++, Basic, there is a clear distinction between
    - the Tool (the Programming Language Compiler/Interpreter)
    - and the Application (Binary or Sourcecode)
  - in Forth, a Basic Forth System will be extended and changed to become the Application
  - = Tool and Application are not distinct

Outline

1. Forth in Time
   - Birth of Forth and the roaring 80s
   - Forth Today
2. An Overview on Forth
   - Forth Basics
   - Examples (Forth vs. C)
   - Forth is easily extensible
3. Why learning/using Forth
   - Broaden your horizon
   - Create lean and fast Programs
   - Forth, the good and the bad
4. 21C3 - Forth Crossover

What does it look like

C[1]

The function main

```
main() /* test power function */
{
  int i;
  for (i=0; i<10; ++i)
    printf("%d %d %d\n", i, power(2,i), power(-3,i));
}
```

Forth

A word to test “power”

```
: main ( – ) \ test power function
  10 0 do
    i . 2 i power . -3 i power . cr
  loop
;
```

[1]

More Forth vs. C

C[2]

Here is the K&R version of the function power

```
power(x,n) /* raise x to the n-th power; n > 0 */
int x,n;
{
  int i,p;
  p = 1;
  for (i=1; i<=n; ++i)
    p = p * x;
  return(p);
}
```

[2] Again: Example taken from Holon Forth Website

More Forth vs. C

In Forth it can be written as

Definition of Word “power”

```
: power ( x n – p ) \ raise x to the n-th power; n > 0
1 swap 0 do over * loop nip ;
```

More Forth vs. C

or in more detail

Definition of Word “power”

```
: power ( x n – p ) \ raise x to the n-th power; n > 0
  1 swap     \ – x 1 n
  0          \ – x 1 n 0
  do         \ – x 1
    over     \ – x 1 x
    *        \ – x 1*x, finally: – x x^n
  loop
  nip        \ – x^n
;
```

Outline

1. Forth in Time
   - Birth of Forth and the roaring 80s
   - Forth Today
2. An Overview on Forth
   - Forth Basics
   - Examples (Forth vs. C)
   - Forth is easily extensible
3. Why learning/using Forth
   - Broaden your horizon
   - Create lean and fast Programs
   - Forth, the good and the bad
4. 21C3 - Forth Crossover

Extending Forth

- Forth can be easily extended
- because the programmer has full access to the underlying system
- extending Forth is changing the Forth System (including the Forth Compiler and/or Interpreter)
- Porting Forth to a new Processor or OS can be done in 1-2 Weeks (or less)
- New Ideas can be added to the language easily (like OOP, Object Oriented Programming)

Extending Forth

- Forth can be easily extended
- because the programmer has full access to the underlying system
- extending Forth is changing the Forth System (including the Forth Compiler and/or Interpreter)
- Porting Forth to a new Processor or OS can be done in 1-2 Weeks (or less)
- New Ideas can be added to the language easily (like OOP, Object Oriented Programming)

Extending Forth

- Forth can be easily extended
- because the programmer has full access to the underlying system
- extending Forth is changing the Forth System (including the Forth Compiler and/or Interpreter)
- Porting Forth to a new Processor or OS can be done in 1-2 Weeks (or less)
- New Ideas can be added to the language easily (like OOP, Object Oriented Programming)

Extending Forth

- Forth can be easily extended
- because the programmer has full access to the underlying system
- extending Forth is changing the Forth System (including the Forth Compiler and/or Interpreter)
- Porting Forth to a new Processor or OS can be done in 1-2 Weeks (or less)
- New Ideas can be added to the language easily (like OOP, Object Oriented Programming)

Extending Forth

- Forth can be easily extended
- because the programmer has full access to the underlying system
- extending Forth is changing the Forth System (including the Forth Compiler and/or Interpreter)
- Porting Forth to a new Processor or OS can be done in 1-2 Weeks (or less)
- New Ideas can be added to the language easily (like OOP, Object Oriented Programming)

OOP Extension in 655 Bytes

OOP Extension to ANSI Forth

```
\ Mini-OOF 12apr98py
:  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 ;
:  defines ( xt class "name" -- ) ' >body @ + !  ;
:  new ( class -- o ) here over @ allot swap over !  ;
:  ::  ( class "name" -- ) ' >body @ + @ compile, ;
Create object 1 cells , 2 cells ,
```

Outline

1. Forth in Time
   - Birth of Forth and the roaring 80s
   - Forth Today
2. An Overview on Forth
   - Forth Basics
   - Examples (Forth vs. C)
   - Forth is easily extensible
3. Why learning/using Forth
   - Broaden your horizon
   - Create lean and fast Programs
   - Forth, the good and the bad
4. 21C3 - Forth Crossover

Learn more on Programming

- Forth naturally teached you to use techniques rediscovered today in “Extreme Programming”[3]
  - create small parts (Forth Words), test them, make them work
  - “fun-down approach" of software development - start where you will get the most immediate gratification
  - Refactor - make your code work well
- read “Thinking Forth” (published 1984), now available under Creative Commons License[4]

[3] See DDJ Article “Extreme Forth”

[4] see http://thinking-forth.sourceforge.net/

Learn more on Programming

- Forth naturally teached you to use techniques rediscovered today in “Extreme Programming”[3]
  - create small parts (Forth Words), test them, make them work
  - “fun-down approach" of software development - start where you will get the most immediate gratification
  - Refactor - make your code work well
- read “Thinking Forth” (published 1984), now available under Creative Commons License[4]

[3] See DDJ Article “Extreme Forth”

[4] see http://thinking-forth.sourceforge.net/

Learn more on Programming

- Forth naturally teached you to use techniques rediscovered today in “Extreme Programming”[3]
  - create small parts (Forth Words), test them, make them work
  - “fun-down approach" of software development - start where you will get the most immediate gratification
  - Refactor - make your code work well
- read “Thinking Forth” (published 1984), now available under Creative Commons License[4]

[3] See DDJ Article “Extreme Forth”

[4] see http://thinking-forth.sourceforge.net/

Learn more on Programming

- Forth naturally teached you to use techniques rediscovered today in “Extreme Programming”[3]
  - create small parts (Forth Words), test them, make them work
  - “fun-down approach" of software development - start where you will get the most immediate gratification
  - Refactor - make your code work well
- read “Thinking Forth” (published 1984), now available under Creative Commons License[4]

[3] See DDJ Article “Extreme Forth”

[4] see http://thinking-forth.sourceforge.net/

Learn more...

Forth ideas can be used in any programming language

DDJ Electronic Review of Computer Books:

Even if you never use Forth, however, you might benefit from studying some of its features. For example, all of the internal code used by the Forth interpreter and compiler is available to application developers. This simple idea has many benefits - it exposes a complete standard set of tools, it lets you extend the environment and language in deep ways, and it enables some very advanced development paradigms.

Outline

1. Forth in Time
   - Birth of Forth and the roaring 80s
   - Forth Today
2. An Overview on Forth
   - Forth Basics
   - Examples (Forth vs. C)
   - Forth is easily extensible
3. Why learning/using Forth
   - Broaden your horizon
   - Create lean and fast Programs
   - Forth, the good and the bad
4. 21C3 - Forth Crossover

Save Time, Memory and CPU Cycles

- Forth allows for lots of functionality into limited memory
  (small Forth System ~ 4 KB, large Forth System ~ 100-200 KB)
- Code (and much data) is ROM-able (good for embedded Systems)
- Is a "HighLevelLanguage" that encourages highly modular code.

Save Time, Memory and CPU Cycles

- Forth allows for lots of functionality into limited memory
  (small Forth System ~ 4 KB, large Forth System ~ 100-200 KB)
- Code (and much data) is ROM-able (good for embedded Systems)
- Is a "HighLevelLanguage" that encourages highly modular code.

Save Time, Memory and CPU Cycles

- Forth allows for lots of functionality into limited memory
  (small Forth System ~ 4 KB, large Forth System ~ 100-200 KB)
- Code (and much data) is ROM-able (good for embedded Systems)
- Is a "HighLevelLanguage" that encourages highly modular code.

Save Time, Memory and CPU Cycles

- Can be interpreted at high speed (approaching that of machine code). Think "interactive hardware debugging".
- Easily compiled to machine code.
- Integrated access to assembly language.
- Is "fully and easily extensible." (Only a few low-level routines of the runtime are not written in FORTH.)

Save Time, Memory and CPU Cycles

- Can be interpreted at high speed (approaching that of machine code). Think "interactive hardware debugging".
- Easily compiled to machine code.
- Integrated access to assembly language.
- Is "fully and easily extensible." (Only a few low-level routines of the runtime are not written in FORTH.)

Save Time, Memory and CPU Cycles

- Can be interpreted at high speed (approaching that of machine code). Think "interactive hardware debugging".
- Easily compiled to machine code.
- Integrated access to assembly language.
- Is "fully and easily extensible." (Only a few low-level routines of the runtime are not written in FORTH.)

Save Time, Memory and CPU Cycles

- Can be interpreted at high speed (approaching that of machine code). Think "interactive hardware debugging".
- Easily compiled to machine code.
- Integrated access to assembly language.
- Is "fully and easily extensible." (Only a few low-level routines of the runtime are not written in FORTH.)

Outline

1. Forth in Time
   - Birth of Forth and the roaring 80s
   - Forth Today
2. An Overview on Forth
   - Forth Basics
   - Examples (Forth vs. C)
   - Forth is easily extensible
3. Why learning/using Forth
   - Broaden your horizon
   - Create lean and fast Programs
   - Forth, the good and the bad
4. 21C3 - Forth Crossover

What is good at Forth

- It is simple to build from the bottom up.
- You can get an *application* to run in a miniscule amount of RAM.
- You can try things out in real time as you build your system.
- Compared to any other interpreted language, it is fast.

What is good at Forth

- It is simple to build from the bottom up.
- You can get an *application* to run in a miniscule amount of RAM.
- You can try things out in real time as you build your system.
- Compared to any other interpreted language, it is fast.

What is good at Forth

- It is simple to build from the bottom up.
- You can get an *application* to run in a miniscule amount of RAM.
- You can try things out in real time as you build your system.
- Compared to any other interpreted language, it is fast.

What is good at Forth

- It is simple to build from the bottom up.
- You can get an *application* to run in a miniscule amount of RAM.
- You can try things out in real time as you build your system.
- Compared to any other interpreted language, it is fast.

What is (maybe) bad on Forth

- If you know one Forth System...
  - ... you know one Forth System
  - ... at one point of time, because Forth Systems are totally comprehensible by one person, ...
  - ... Forth Programmer start do create their own Forth, their own OOP etc etc etc
- but this can also be seen as a good thing ;)

What is (maybe) bad on Forth

- If you know one Forth System...
  - ... you know one Forth System
  - ... at one point of time, because Forth Systems are totally comprehensible by one person, ...
  - ... Forth Programmer start do create their own Forth, their own OOP etc etc etc
- but this can also be seen as a good thing ;)

What is (maybe) bad on Forth

- If you know one Forth System...
  - ... you know one Forth System
  - ... at one point of time, because Forth Systems are totally comprehensible by one person, ...
  - ... Forth Programmer start do create their own Forth, their own OOP etc etc etc
- but this can also be seen as a good thing ;)

What is (maybe) bad on Forth

- If you know one Forth System...
  - ... you know one Forth System
  - ... at one point of time, because Forth Systems are totally comprehensible by one person, ...
  - ... Forth Programmer start do create their own Forth, their own OOP etc etc etc
- but this can also be seen as a good thing ;)

What is (maybe) bad on Forth

- If you know one Forth System...
  - ... you know one Forth System
  - ... at one point of time, because Forth Systems are totally comprehensible by one person, ...
  - ... Forth Programmer start do create their own Forth, their own OOP etc etc etc
- but this can also be seen as a good thing ;)

Should I start using (only) Forth now?

- No, but you should give it a try
  - If you like it, keep it
  - If you don’t like it, there is always a lesson learned
  - You might need a Forth-compatible brain to enjoy Forth

Should I start using (only) Forth now?

- No, but you should give it a try
  - If you like it, keep it
  - If you don't like it, there is always a lesson learned
  - You might need a Forth-compatible brain to enjoy Forth

Should I start using (only) Forth now?

- No, but you should give it a try
  - If you like it, keep it
  - If you don’t like it, there is always a lesson learned
  - You might need a Forth-compatible brain to enjoy Forth

Should I start using (only) Forth now?

- No, but you should give it a try
  - If you like it, keep it
  - If you don’t like it, there is always a lesson learned
  - You might need a Forth-compatible brain to enjoy Forth

Should I start using (only) Forth now?

- Keep in mind:
  - Forth might not be the best solution for every problem (Universal truth)
  - There is no such thing as the “best programming language”
  - Build your *skills* for a *bag of tools* for the *bunch of problems* out there

Should I start using (only) Forth now?

- Keep in mind:
  - Forth might not be the best solution for every problem (Universal truth)
  - There is no such thing as the “best programming language”
  - Build your *skills* for a *bag of tools* for the *bunch of problems* out there

Should I start using (only) Forth now?

- Keep in mind:
  - Forth might not be the best solution for every problem (Universal truth)
  - There is no such thing as the “best programming language”
  - Build your *skills* for a *bag of tools* for the *bunch of problems* out there

Should I start using (only) Forth now?

- Keep in mind:
  - Forth might not be the best solution for every problem (Universal truth)
  - There is no such thing as the “best programming language”
  - Build your *skills* for a *bag of tools* for the *bunch of problems* out there

Summary

- Forth is different enough to be worth learning.
- Forth is fun. Forth is flexible. Forth is fast and small.

Summary

- Forth is different enough to be worth learning.
- Forth is fun. Forth is flexible. Forth is fast and small.

Summary

- Forth is different enough to be worth learning.
- Forth is fun. Forth is flexible. Forth is fast and small.

21C3 - Forth Crossover

Brainf*ck in Forth, see Talk #65 Clifford Wolf

```
( Brainf*ck in FORTH, 2003, Carsten Strotmann )
CR ." BrainF*ck "
0 VARIABLE IP
:  IP+ IP @ 1+ IP ! ; :  IP- IP @ 1 - IP ! ;
:  IP@ IP @ C@ ; :  2DUP DUP DUP ;
:  BF+ 2DUP C@ 1+ SWAP C! ;
:  BF- 2DUP C@ 1 - SWAP C! ;
:  BF> 1+ ; :  BF< 1 - ;
:  BF. DUP C@ EMIT ; :  BF, DUP KEY SWAP C! ;
:  BF[ DUP C@ 0= IF BEGIN IP+ IP@ 93 = UNTIL THEN ;
:  BF] BEGIN IP- IP@ 91 = UNTIL ;
:  BFI ( addr -- )
IP ! BEGIN
IP@ 93 = IF BF] THEN IP@ 91 = IF BF[ THEN IP@ 62 = IF BF> THEN
IP@ 60 = IF BF< THEN IP@ 46 = IF BF. THEN IP@ 44 = IF BF, THEN
IP@ 45 = IF BF- THEN IP@ 43 = IF BF+ THEN
IP+ IP@ 0 = UNTIL ;
:  BF HERE 3000 ERASE HERE BL WORD 1+ BFI ;
```

21C3 - Forth Crossover

- Forth in Hardware
  - See Talk #42
  - Cored Programming - Building systems on your own soft core, Klaus Schleisiek

For Further Reading I


Leo Brodie, Starting Forth.
http://www.forthfreak.net/wiki/index.cgi/?StartingForth


Leo Brodie, Thinking Forth.
http://thinking-forth.sourceforge.net/


ForthFreak Wiki
http://www.forthfreak.net/wiki/index.cgi?ForthFreak


Forth Gesellschaft e.V.
http://www.forth-ev.de/


Forth Interest Group (FIG) US
http://www.forth.org/

For Further Reading II

Forth Inc.
http://www.forth.com/index.html

c2 Wiki (Ward Cunningham) on Forth
http://c2.com/cgi/wiki?ForthLanguage

DDJ Re-Review of Thinking Forth, 2003
http://www.ercb.com/ddj/2003/ddj.0308.html