Programming in the Weird

This is a talk about computer programming languages.
Not your normal programming languages like C or python.
Instead what happens when you look at the stranger corners of programming.
Esoteric Programming Languages as they are known

Why not use a sensible language

- https://en.wikipedia.org/wiki/List_of_programming_languages
- Many of those can do useful things and have domain specific benefits, why would you want to avoid them and look at something completely different
- C, Python, .Net, C#, Lisp, Haskell, Prolog, Shell (Bourne and C http://www.faqs.org/faqs/unix-faq/shell/csh-whynot/) and many others.
- These make sense, they can be useful and solve problems, future programmers can read and understand the source.

Obfuscating normal languages

- There are competitons to make some of these sensible languages look horrible
- C - http://www.ioccc.org/
- Python - http://p-nand-q.com/programming/obfuscation/python/more.html and http://wiki.c2.com/?ObfuscatedPython
- Perl (of course) - https://en.wikipedia.org/wiki/Obfuscated_Perl_Contest or the Just Another Perl Hacker thing http://perl.plover.com/obfuscated/ (2500 words to explain what it is doing)

Prolog – Useful domain specific language

Prolog is a language used for AI and searching problems with nicely defined data structures, naturally recursive and data is declared as part of the code. So sure if you define a huge database of information and then make some qquestions it works brilliantly. The Airline industry uses it for airfare searches and it matches that problem really well. However if you want to use it for something like a syslog daemon it may not be the perfect match.

- http://www.drdobbs.com/parallel/the-practical-application-of-prolog/184405220
- http://www.academia.edu/16789218/If_Prolog_is_the_Answer_What_is_the_Question_or_What_it_Takes_to_Support_AI_Programming_Paradigms
  (great paper talking about using many domain speccific tools to solve a problem)

Wrong tool for the job?

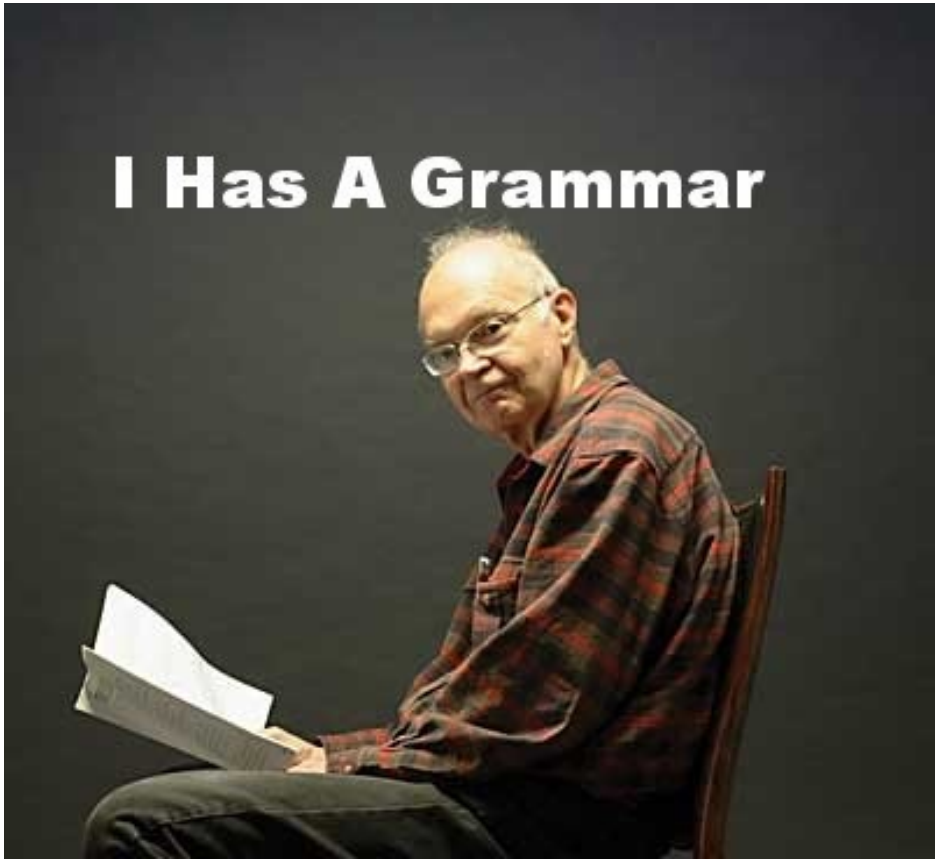
- A prime example of this is SHDNS
- Michael Still gave a talk at linux.conf.au in 2004 about his implemntation of DNS in Shell
- https://linux.org.au/conf/2004/abstracts.html#1
- Source available at http://www.stillhq.com/shdns/
-

So why Esoteric languages?

- They help you think outside the box
- Though many of them are like thinking outside a box in a foreign country
- With instructions in Latin
- Likely containing a wild animal that may try to bite you
- haec continet magna felis sit mordebunt vos

Creating new languages

- Lexx and Yacc (or the better GNU tools, flex and bison) allow easy creation of fast output from some arbitrary grammar



- LEX analyses input streams (text files) using regular expressions to tokenise them into something that makes more sense
- YACC (Yet Another Compiler Compiler) analyses the structure of the tokens, makes sure they are syntactically correct and applies grouping for a bigger picture look
- Using these tools you are able to turn text into some other form of text or into some actions to do something in c code

Obviously some of the creators of Esoteric languages decided using text as an input really made things too easy. Hence we have Velato with midi files as source code and Piet which uses bitmaps that appear to be abstract art as source code.

Intercal – the original

- From 1973 - http://www.muppetlabs.com/~breadbox/intercal-man/
- For a long time a print out of that was all that people had available to talk about it
- In the 90s C-Intercal was implemented and released and the language became more popular
- A paper worth reading, Why you should program in intercal http://catb.org/esr/intercal/stross.html

- Getting rid of GOTO, https://en.wikipedia.org/wiki/Considered_harmful
- Polite programmers
- Percentage chance execution
-

```
      PLEASE NOTE THIS PROGRAM RECOGNIZES "HELLO, WORLD" USING COME FROM
      DON'T TYPE IN ANYTHING ELSE, OR YOU'LL GET AN ERROR!

      PLEASE NOTE: COMPILE WITH ick -m FOR THIS TO WORK.

      DO ,1 <- #12
(1)   DO WRITE IN ,1
      DO GIVE UP

      PLEASE NOTE THIS CHECKS EACH CHARACTER IN TURN
      DO COME FROM '#255~'"?',1SUB#1'$#72"~"#0$#255""~#1
      PLEASE START WITH AN H NEXT TIME

      DO COME FROM '#255~'"?',1SUB#2'$#253"~"#0$#255""~#1
      DO (2) NEXT
      DO REMEMBER THAT E COMES SECOND

      DO COME FROM '#255~'"?',1SUB#3'$#7"~"#0$#255""~#1
      DO (4) NEXT
      PLEASE USE L THIRD NEXT TIME

      DO COME FROM '#255~'"?',1SUB#4'$#0"~"#0$#255""~#1
      PLEASE DO (2) NEXT DO (4) NEXT
      DO USE TWO LS, NOT A SINGLE L

      DO COME FROM '#255~'"?',1SUB#5'$#3"~"#0$#255""~#1
      DO (8) NEXT
      PLEASE END 'HELLO' WITH 'O'

      DO COME FROM '#255~'"?',1SUB#6'$#221"~"#0$#255""~#1
      DO (8) NEXT PLEASE DO (2) NEXT
      DO USE COMMAS TO SEPARATE WORDS

      DO COME FROM '#255~'"?',1SUB#7'$#244"~"#0$#255""~#1
      DO (8) NEXT PLEASE DO (4) NEXT
      PLEASE USE SPACES AFTER PUNCTUATION

      DO COME FROM '#255~'"?',1SUB#8'$#55"~"#0$#255""~#1
      DO (8) NEXT DO (4) NEXT PLEASE DO (2) NEXT
      DO START 'WORLD' WITH A 'W'

      DO COME FROM '#255~'"?',1SUB#9'$#248"~"#0$#255""~#1
      DO (16) NEXT
      PLEASE PLACE AN O IN THE NINTH POSITION

      DO COME FROM '#255~'"?',1SUB#10'$#3"~"#0$#255""~#1
      DO (16) NEXT DO (2) NEXT
      DO USE AN R IN THE MIDDLE OF WORLD

      DO COME FROM '#255~'"?',1SUB#11'$#250"~"#0$#255""~#1
      DO (16) NEXT DO (4) NEXT
      PLEASE LET AN L BE PENULTIMATE

      DO COME FROM '#255~'"?',1SUB#12'$#248"~"#0$#255""~#1
      DO (16) NEXT PLEASE DO (4) NEXT DO (2) NEXT
      DO END WITH A D

(2)   PLEASE RESUME #1
(4)   DO (2) NEXT DO (2) NEXT DO RESUME #1
(8)   DO (4) NEXT DO (4) NEXT PLEASE RESUME #1
(16)  DO (8) NEXT DO (8) NEXT PLEASE RESUME #1
```

Reigniting the passion in the 90s

The paper about why it is the next best thing from ESR, his implementation C-Intercal etc

Brainfuck

- Interesting origin, has exploded to be the best known Esoteric language
- Brainfuck was invented by Urban Müller in 1993, in an attempt to make a language for which he could write the smallest possible compiler for the Amiga OS, version 2.0. He managed to write a 240-byte compiler. The language was inspired by False, which had a 1024-byte compiler. Müller chose to name the language brainfuck
- Thus it is an interesting idea and implementation, though any attempt to use it becomes painful

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.++++
+++..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.
```

Taking the idea of what BF was doing to another level, Binary Lambda Calculus

Whitespace

Meme Central, LOLCODE

- This is almost a shame to talk about, if you were ever caught up in the lolcats memes online
- http://lolcode.org/
- The spec is online linked from there https://github.com/justinmeza/lolcode-spec/blob/master/v1.2/lolcode-spec-v1.2.md
- Though the spec is somewhat incomplete this is mostly just a substitution for normal operation language

- Comments -

```
I HAS A VAR ITZ 12,         BTW VAR = 12
I HAS A VAR ITZ 12
        OBTW this is a long comment block
            see, i have more comments here
            and here
        TLDR
I HAS A FISH ITZ BOB
```

- If-Then-Else

```
BOTH SAEM ANIMAL AN "CAT"
O RLY?
  YA RLY, VISIBLE "J00 HAV A CAT"
  MEBBE BOTH SAEM ANIMAL AN "MAUS"
    VISIBLE "NOM NOM NOM. I EATED IT."
OIC
```

- Loops

```
IM IN YR <label> <operation> YR <variable> [TIL|WILE <expression>]
  <code block>
IM OUTTA YR <label>
```

However there is a LOLPython that allows you to write effectively in LOLCode but has access to the python libraries, best of all worlds?
http://www.dalkescientific.com/writings/diary/archive/2007/06/01/lolpython.html

On that page you can see an example of generating the fibbonacci sequence in LOLCODE

Arnold Scharzenegger

- ArnoldC is a language that is similar to LOLCODE with pretty basic token repalcement and is another Meme style language
- Hello World is

```
IT'S SHOWTIME
TALK TO THE HAND "hello world"
YOU HAVE BEEN TERMINATED
```

False I LIED

True NO PROBLEMO

If BECAUSE I'M GOING TO SAY PLEASE

Else BULLSHIT

EndIf YOU HAVE NO RESPECT FOR LOGIC

While STICK AROUND

EndWhile CHILL

PlusOperator GET UP

MinusOperator GET DOWN

MultiplicationOperator YOU'RE FIRED

DivisionOperator HE HAD TO SPLIT

ModuloOperator I LET HIM GO

EqualTo YOU ARE NOT YOU YOU ARE ME

GreaterThan LET OFF SOME STEAM BENNET

Or CONSIDER THAT A DIVORCE

And KNOCK KNOCK

DeclareMethod LISTEN TO ME VERY CAREFULLY

NonVoidMethod GIVE THESE PEOPLE AIR

MethodArguments I NEED YOUR CLOTHES YOUR BOOTS AND YOUR MOTORCYCLE

Return I'LL BE BACK

EndMethodDeclaration HASTA LA VISTA, BABY

CallMethod DO IT NOW

AssignVariableFromMethodCall GET YOUR ASS TO MARS

DeclareInt HEY CHRISTMAS TREE

SetInitialValue YOU SET US UP

BeginMain IT'S SHOWTIME

EndMain YOU HAVE BEEN TERMINATED

Print TALK TO THE HAND

ReadInteger I WANT TO ASK YOU A BUNCH OF QUESTIONS AND I WANT TO HAVE THEM ANSWERED IMMEDIATELY

AssignVariable GET TO THE CHOPPER

SetValue HERE IS MY INVITATION

EndAssignVariable ENOUGH TALK

ParseError WHAT THE FUCK DID I DO WRONG

Similar but different - Omgrofl

- Similar tokens to LOLCODE, however operates differently
- Stack/Queue based language (thus turing complete)
- The behaviour is more similar to BF, though obviously far more verbose
- Examples

Example- Addition of two numbers

- In Omgrofl

```
loool iz lol
looooool iz lool
rtfm
 wtf looooool iz liek 0
  tldr
 brb
 lmao loool
 roflmao looooool
brb
```

- In C

```
uint8_t loool = lol;
uint8_t looooool = lool;
while (true)
{
 if (looooool == 0)
  break;
 loool++;
 looooool--;
}
```

Doing something different (CHEF, Shakespeare)

Hello World Cake with Chocolate sauce.

This prints hello world, while being tastier than Hello World Souffle. The main chef makes a " world!" cake, which he puts in the baking dish. When he gets the sous chef to make the "Hello" chocolate sauce, it gets put into the baking dish and then the whole thing is printed when he refrigerates the sauce. When actually cooking, I'm interpreting the chocolate sauce baking dish to be separate from the cake one and Liquify to mean either melt or blend depending on context.

Ingredients.

33 g chocolate chips

100 g butter

54 ml double cream

2 pinches baking powder

114 g sugar

111 ml beaten eggs

119 g flour

32 g cocoa powder

0 g cake mixture

Cooking time: 25 minutes.

Pre-heat oven to 180 degrees Celsius.

Method.

Put chocolate chips into the mixing bowl.

Put butter into the mixing bowl.

Put sugar into the mixing bowl.

Put beaten eggs into the mixing bowl.

Put flour into the mixing bowl.

Put baking powder into the mixing bowl.

Put cocoa powder into the mixing bowl.

Stir the mixing bowl for 1 minute.

Combine double cream into the mixing bowl.

Stir the mixing bowl for 4 minutes.

Liquify the contents of the mixing bowl.

Pour contents of the mixing bowl into the baking dish.

bake the cake mixture.

Wait until baked.

Serve with chocolate sauce.

chocolate sauce.

Ingredients.

111 g sugar

108 ml hot water

108 ml heated double cream

101 g dark chocolate

72 g milk chocolate

Method.

Clean the mixing bowl.

Put sugar into the mixing bowl.

Put hot water into the mixing bowl.

Put heated double cream into the mixing bowl.

dissolve the sugar.

agitate the sugar until dissolved.

Liquify the dark chocolate.

Put dark chocolate into the mixing bowl.

Liquify the milk chocolate.

Put milk chocolate into the mixing bowl.

Liquify contents of the mixing bowl.

Pour contents of the mixing bowl into the baking dish.

Refrigerate for 1 hour.

The really challenging

- Befunge - really weird

  Hello World

0"!dlroW ,olleH">:#,_@

- Malbolge – Dante, hell

(=<`#9]~6ZY32Vw/.R,+Op(L,+k#Gh&}Cdz@aw=;zyKw
%ut4Uqp0/mlejihtfrHcbaC2^W\>Z,XW)UTSL53\HGFjW

Thoughts and interesting bits

- Greenspun's Tenth Rule
  https://en.wikipedia.org/wiki/Greenspun's_tenth_rule

  Any sufficiently complicated C or Fortran program contains an ad hoc, informally-specified, bug-ridden, slow implementation of half of Common Lisp
- JWZ - Every program attempts to expand until it can read mail. Those programs which cannot so expand are replaced by ones which can.
- Software Peter Principle (called that on wikipedia), the idea of how software can become too complex for anyone to understand. Most languages and methodologies attempt to avoid that
- Esoteric languages go in the other direction