C: an introduction

Programming style guide

Any fool can write code that a computer can understand.

Good programmers write code that humans can understand.

Martin Fowler

What's this?

- **Calculate pi to 800 digits** in 160 characters of code. Written by Dik T. Winter at CWI.

```
int a=10000,b,c=2800,d,e,f[2801],g;main(){for(;b-c;)f[b++]=a/5; for(;d=0,g=c*2;c-
=14,printf("%.4d",e+d/a),e=d%a)for(b=c;d+=f[b]*a, f[b]=d%--g,d/=g--,--b;d*=b);}
```

- **Calculate the day of the week** in 45 characters of code. Written by Mike Keith.

```
(d+=m<3?y--:y-2,23*m/9+d+4+y/4-y/100+y/400)%7
```

- **Diffie-Helman in 10 lines of code** posted anonymously to sci.crypt and publicised by Adam Back. This actually carries out multiple precision modular exponentation using 8-bit digits. Set **S** to the number of 8-bit digits required plus 1. This example is for 1024 bits.
-
```
#include <stdio.h> /* Usage: dh base exponent modulus */ typedef unsigned char u;u
m[1024],g[1024],e[1024],b[1024];int n,v,d,z,S=129;a( u *x,u *y,int o){d=0;for(v=S;v--
;){d+=x[v]+y[v]*o;x[v]=d;d=d>>8;}}s(u *x){for( v=0;(v<S-
1)&&(x[v]==m[v]);)v++;if(x[v]>=m[v])a(x,m,-1);}r(u *x){d=0;for(v=0;v<
S;){d|=x[v];x[v++]=d/2;d=(d&1)<<8;}}M(u *x,u *y){u X[1024],Y[1024];bcopy(x,X,S
);bcopy(y,Y,S);bzero(x,S);for(z=S*8;z--;){if(X[S-1]&1){a(x,Y,1);s(x);}r(X);a(Y
,Y,1);s(Y);}}h(char *x,u *y){bzero(y,S);for(n=0;x[n]>0;n++){for(z=4;z--;)a(y,y
,1);x[n]|=32;y[S-1]|=x[n]-48-(x[n]>96)*39;}}p(u *x){for(n=0;!x[n];)n++;for(;n<
S;n++)printf("%c%c",48+x[n]/16+(x[n]>159)*7,48+(x[n]&15)+7*((x[n]&15)>9));
printf("\n");}main(int c,char **v){h(v[1],g);h(v[2],e);h(v[3],m);bzero(b,S);b[ S-
1]=1;for(n=S*8;n--;){if(e[S-1]&1)M(b,g);M(g,g);r(e);}p(b);}
```

KU LEUVEN

Why a programming style?

- Give your code a uniform look:
  - Develop clean and readable code
  - When working in team, it's best that everybody uses the same style
- Basic Rules
  - All should be as understandable as possible.
  - All should be as readable as possible, except when it would conflict with the previous rule.
  - All should be as simple as possible, except when it would conflict with the previous rules.

KU LEUVEN

Gnome project guidelines

- *Programmers should strive to write good code so that it is easy to understand and modify by others*
- Important qualities of good code
  - Clarity
  - Consistency
  - Extensibility
  - Correctness
- Taken from **Tyler Bletsch** NCSU course notes - https://courses.ncsu.edu/csc230/

KU LEUVEN

Programming style guide

- includes recommendations for:
  - lexical conventions
  - conventions for writing comments
- with focus on:
  - Design/Coding
  - Expressions
  - Control Flow
  - Functions
  - I/O
  - Avoiding Common Errors
  - …

KU LEUVEN

Programming by Kernighan

- Write clearly - don't be too clever
- Say what you mean, simply and directly
- Code must speak for itself; comments should add information. Do not simply echo code with comments. Do not comment bad code; rewrite it.
- Use the "telephone test" for readability - someone should be able to understand clear code over the telephone
- Every time you make a test, do something. (No long strings of if's.)
- If you do all the smart things while writing the code, by definition you are not smart enough to debug it

Coding style: comments

- A lot of time is spent on upgrading, maintaining, debugging and adapting code sometimes even more than on actually writing new code from scratch...
- Comments in modern flavors of C come in 2 forms:
  - // Single Line Comments
    (added by C99 standard, known as c++ style of comments)
  - /*Multi-Line Comments*/
    (only form of comments supported by C89 standard)
  - Comments can be placed everywhere, except in another comment (nesting of comments does not work)
    `/*  /*  */  */` is illegal

Comment example

```
#include <stdio.h>
int main(void)
{
int i=0; // loop variable.
printf("Hello, World!");
/*
   For Loop (int i) Loops the following procedure i times (for number of lines). Performs 'for' loop j on each
   loop, and prints a new line at end of each loop.
*/
for (i=0; i<1; i++)
   {
   printf("\n");
   break; //Exits 'for' loop.
   }
return 0;
}
```

KU LEUVEN

Coding style: comments

Extra relevant information, at the beginning of each file:

- name of the program
- what it does
- author (how to reach him/her, ...)
- usage:
  - how do you call it,
  - what are the options
- revision history: who edited the file when and why
- file formats, input/output files
- references
- restrictions: what the program does not do

KU LEUVEN

example

```
/*********************************************************
* hello -- program to print out "Hello World". *
*
* Ralf Kaiser, September 2003 *
* *
* Reference: Steve Oualline, Practical C Programming, *
* O'Reilly *
* *
* Purpose: Demonstration of comments *
* *
*********************************************************/
#include <stdio.h>
int main()
{
```

KU LEUVEN

Coding style: indentation

- make programs easier to read, to understand
- indent the code according to the level of the statement.
- Use `indent` programming tool (linux)
  - some styles (http://en.wikipedia.org/wiki/Indent_style)

```
int main(){
    if (morning) {
        printf("Hello World\n");
    } else {
        printf("Good Night\n");
    }
    return (0);
}
```

```
int main()
{
    if (morning)
    {
        printf("Hello World\n");
    }
    else
    {
        printf("Good Night\n");
    }
    return (0);
}
```

KU LEUVEN

Coding style

- Each variable has to be declared
  - comment it also at the same time
  - good practice to mention the units!
- Keep it simple
  - rule of thumb: a function may not be longer than 2-3 pages
  - avoid long statements
  - avoid *deep* nesting
  - ...

KU LEUVEN

C: programming style guides

- generate your own style guide
  www.rosvall.ie/CSG/
- NASA programming style guide:
  https://ntrs.nasa.gov/search.jsp?R=19950022400
- Gnome:
  https://developer.gnome.org/documentation/guidelines/programming/coding-style.html
- GNU
  https://www.gnu.org/prep/standards/html_node/Writing-C.html

KU LEUVEN