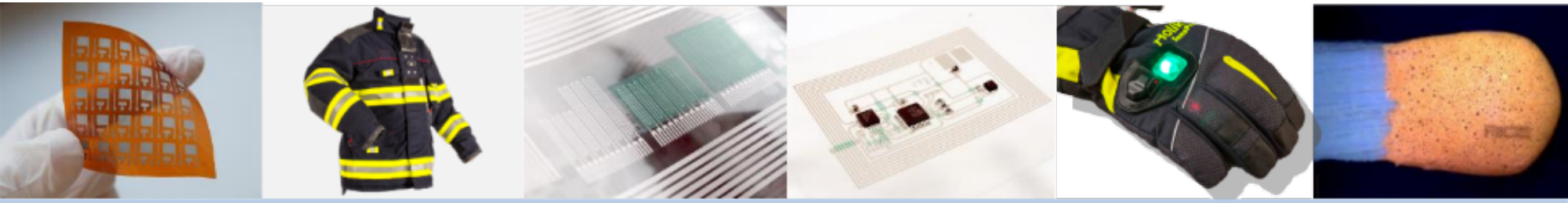

KET/KTL 2019

Introduction to the Safe, Readable, Maintainable and Secure Code for Embedded C

Tipy a příklady

Jan Bělohoubek

FACULTY OF ELECTRICAL ENGINEERING
UNIVERSITY OF WEST BOHEMIA

About C
I'm Free to (do anything) . . . realy?

- C is imperative procedural language
  - describes algorithms
  - uses functions (procedures)
- C is everywhere
  - C compiler is the must for any platform (from MCU to supercomputer)
- C has simple design
  - weak types, pointers & pointer arithmetic, pre-processor, inline assembler, . . .
  - Basic types: int, float, enum
  - Derived types: array, pointer, struct, union

- Data types:
  - char, short, long, int
  - signed, unsigned – signed is default and may be implicit
  - float, double

- Data types:
  - char, short, long, int
  - signed, unsigned – signed is default and may be implicit
  - float, double

```
sizeof(char) == 1
sizeof(short int) <= sizeof(int) <= sizeof(long int)
sizeof(unsigned int) = sizeof(signed int)
sizeof(float) <= sizeof(double) <= sizeof(long double)

...

typedef unsigned char uint8_t;
typedef signed char uint8_t;
typedef unsigned short int uint16_t;
```

- Data types:
  - char, short, long, int
  - signed, unsigned – signed is default and may be implicit
  - float, double

```
// File: /usr/arm-none-eabi/include/machine/_default_types.h
#ifdef __INT16_TYPE__
  typedef __INT16_TYPE__ __int16_t;
  #ifdef __UINT16_TYPE__
    typedef __UINT16_TYPE__ __uint16_t;
  #else
    typedef unsigned __INT16_TYPE__ __uint16_t;
  #endif
  #define ___int16_t_defined 1
#elif __EXP(INT_MAX) == 0x7fff
  typedef signed int __int16_t;
  typedef unsigned int __uint16_t;
  #define ___int16_t_defined 1
```

- Data types:
  - char, short, long, int
  - signed, unsigned – signed is default and may be implicit
  - float, double

```
// File: /usr/arm-none-eabi/include/sys/_stdint.h
#ifdef ___int16_t_defined
  #ifndef _INT16_T_DECLARED
    typedef __int16_t int16_t ;
    #define _INT16_T_DECLARED
  #endif
  #ifndef _UINT16_T_DECLARED
    typedef __uint16_t uint16_t ;
    #define _UINT16_T_DECLARED
  #endif
  #define __int16_t_defined 1
#endif /* ___int16_t_defined */
```

- Data types:
  - char, short, long, int
  - signed, unsigned – signed is default and may be implicit
  - float, double

```
sizeof(char) == 1
sizeof(uint8_t) == sizeof(int8_t) == 1
sizeof(uint16_t) == sizeof(int16_t) == 2
sizeof(uint32_t) == sizeof(int32_t) == 3
```


```
void main() {
  int a, i;
  int *x = &a;

  for(a = 10, i = 0; i < 10; i++)
    if (x = 10)
      goto LOOP;
    else
      printf("X␣!=␣10␣\n");
      x += 1;
      a += 2;
      printf("X␣+=␣1;␣a␣+=2␣\n");

  LOOP:
    printf("Start␣Loop␣...␣%d␣%d\n", a, *x);

    while(1)
      asm("nop");
}
```


```
void main() {
  int a, i;                                  // bad names - too short
  int *x = &a;                               // and undocumented

  for(a = 10, i = 0; i < 10; i++)            // missing brackets
    if (x = 10)                              // this is always TRUE!
      goto LOOP;                             // goto in C ...
    else
      printf("X␣!=␣10␣\n");                  // END-of-for is here!
      x += 1;
      a += 2;
      printf("X␣+=␣1;␣a␣+=2␣\n");

  LOOP:
    printf("Start␣Loop␣...␣%d␣%d\n", a, *x);
    // *x is at address 10 ... that's not our memory ...
    while(1)
      asm("nop");
}
```

- use comments
- write self-documented code
- decomposition – use functions, translation units
- ...

Preprocessor

- preprocessor operates on strings
- may do dangerous and unexpected (!) things . . .

Compliant solution:

```
#ifndef HEADER_H
#define HEADER_H

/* ... Contents of <header.h> ... */

#endif /* HEADER_H */
```


```
#define SWAP(x, y) \
  tmp = x; \
  x = y; \
  y = tmp

int x, y, z, tmp;
if (z == 0)
  SWAP(x, y);
```

Expands to:

```
int x, y, z, tmp;
if (z == 0)
  tmp = x;
x = y;
y = tmp;
```

Compliant solution:

```
#define SWAP(x, y) \
  do { \
    tmp = (x); \
    (x) = (y); \
    (y) = tmp; } \
  while (0)
```


```
#define FOR_LOOP(n)  for(i=0; i<(n); i++);

int i;
FOR_LOOP(3)
{
  puts("Inside␣for␣loop\n");
}
```

Expands to:

```
for(i=0; i<(3); i++);    // empty loop

{
  puts("Inside␣for␣loop\n");
}
```

Compliant solution:

```
#define FOR_LOOP(n)  for(i=0; i<(n); i++)

int i;
FOR_LOOP(3)
{
  puts("Inside␣for␣loop\n");
}
```

```c
#define CUBE(X) ((X) * (X) * (X))

void func(void) {
  int i = 2;
  int a = 81 / CUBE(++i);
  /* ... */
}
```

Expands to:

```c
int a = 81 / ((++i) * (++i) * (++i));
```

- brackets are fine, but
- multiple increment was not intended!

Compliant solution:

```
inline int cube(int i) {
  return i * i * i;
}

void func(void) {
  int i = 2;
  int a = 81 / cube(++i);
  /* ... */
}
```


```
#define STATE_A       0
#define STATE_B       1
#define STATE_C       2
#define STATE_D       3
#define STATE_LAST    4

uint8_t getNextState(uint8_t currState) {
  return (currState + 1) % STATE_LAST;
}
```

Better solution:

```
typedef enum {
  STATE_A = 0,
  STATE_B,
  STATE_C,
  STATE_D,
  STATE_LAST
} states_t;

states_t getNextState(states_t currState) {
  return (currState + 1) % (STATE_LAST + 1);
}
```

Note: do not use constants declared inside enumerate in preprocessor constructs – preprocessor does not know compiler defines!

- provides (a little bit of) type checking
- debugger may be able to display ENUM names instead of values (NICE!)
- compiler numbers enum items automatically
- compiler warning (limited, but present !)
  - switch statement missing case
  - mixing types – some compilers and analyzers can warn you (clang -Wenum-conversion)

Code Analyzers

- Static code analysis: splint, cppcheck, compilers (incl. warnings), ...
- Preffer C-language over preprocessor – (future) tools may catch possible errors

- Featured Reading and Resources:
  - Boswell, Dustin, and Trevor Foucher. The Art of Readable Code: Simple and Practical Techniques for Writing Better Code. " O'Reilly Media, Inc.", 2011.
  - Seacord, Robert C. The CERT C secure coding standard. Pearson Education, 2008.
  - Barnes, John Gilbert Presslie. Safe and secure software: An invitation to Ada 2005. AdaCore, 2008.

Thank you for your attention!

Jan Bělohoubek
UWB, Czech Republic
belohoub@ket.zcu.cz
+420 377 634 514

FACULTY OF ELECTRICAL ENGINEERING
UNIVERSITY OF WEST BOHEMIA