C Programming
1DT301

Lecture 4

[OPNOVA]
ENGINEERED INNOVATION

Katarina Rönndahl

- 11:45 talking about studying abroad.

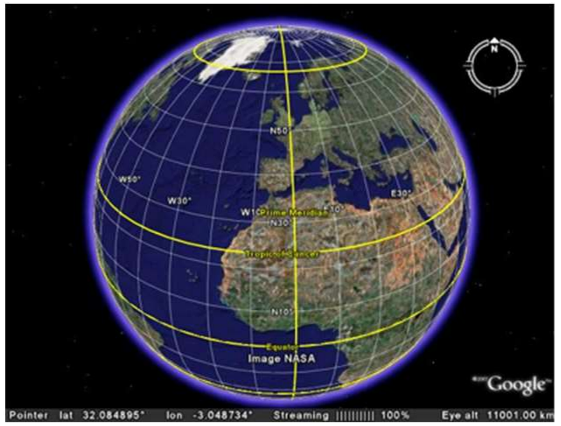
Structures

struct

- A grouped collection of variables
- Organize complicated data
- A.k.a a record

Example

- Represent a GPS coordinate (latitude, longitude)



Name of structure

```
struct GPS_coord
{
        double lat;
        double lon;
};
```

Members

Example (cont'd)

Init (lat=12.43, lon=33.55)

```
struct GPS_coord gps1;
struct GPS_coord gps2 = {12.43, 33.55};
```

access a member.

```
gps1.lat = 44.55;
gps1.lon = 67.98;

printf("Lat=%f Lon=%f", gps1.lat, gps1.lon);
```

Pointer to a struct

!

```
struct GPS_coord *p_gps = NULL;
         p_gps = &gps2;
```

```
         p_gps->lat = 88.9;
```

No dereferencing , instead!

or

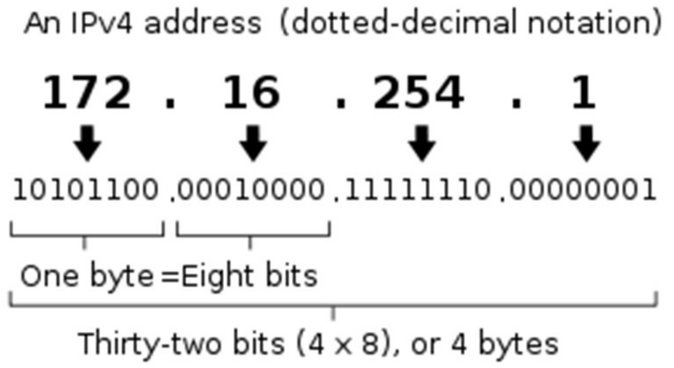
```
(*p_gps).lat = 88.9;
```

union

- A memory area that holds objects of different types and sizes
  →
  Different viewpoints of the same memory area.

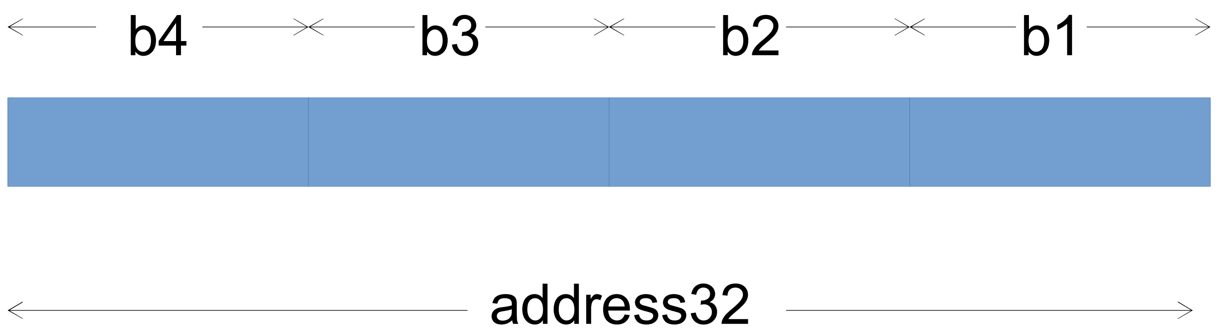
Example

- IPv4 address[1]
- Two views of the same data
  - As integer 2886794753
  - As bytes 172, 16, 254, 1



[1]http://en.wikipedia.org/wiki/IP_address

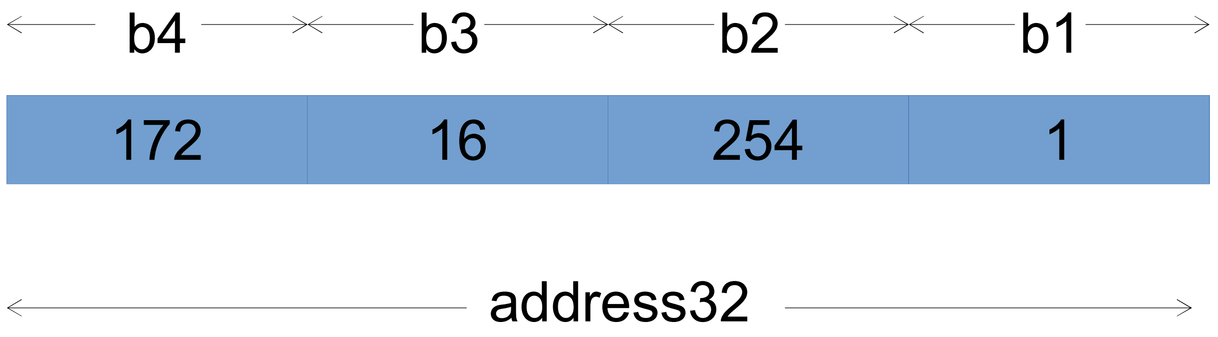
Example (cont'd)



```
struct dotted
{
        unsigned char b1;
        unsigned char b2;
        unsigned char b3;
        unsigned char b4;
};

union ip_address
{
        int address32;
        struct dotted byte;
};
```

Example (cont'd)

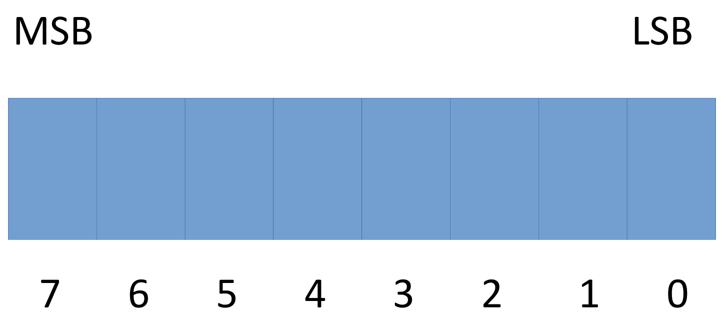
```
union ip_address ip = {0};

    ip.address32 = 2886794753U;
    printf("b1=%u,b2=%u,b3=%u,b4=%u",
                ip.byte.b1, ip.byte.b2, ip.byte.b3, ip.byte.b4);

    ip.byte.b2 = 255;
    printf("Address32=%u", ip.address32);
```



Bitfields

- Using fragments of integers
  - E.g use a single bit as a flag (boolean)
- Bit-oriented protocols
- When memory is expensive

Bits



Bit positions

Note! MSB and LSB may be reversed!
See the concept of endianness.

Example

- Represents a byte’s high and low nibble
  - A nibble has 4 bits.

Bitfield names

```
struct
{
        unsigned int lo:4;
        unsigned int hi:4;
} unsigned char;
```

size in bits

typedef

- Creating new types

```
typedef struct GPS_coord GPS;

GPS gps3;
```

`GPS` is a synonym for **`struct`** `GPS_coord`

Preprocessor

- Prior to compilation.
- Inclusion of files (#include)
- Macro
- Conditional compilation
  - E.g. Useful when compiled on multiple platforms.

#define (macro)

- Three forms
  - #define name
  - #define name replacement
  - #define name(params) replacement
- Exact textual replacement!

Example

```c
#define BOOL int
#define TRUE 1
#define FALSE 0
#define FOREVER while(1);
#define DBG_PRINT(msg) printf("[DBG]:%s\n",msg);

int main(void)
{
  BOOL myBool = TRUE;
  DBG_PRINT("Before Loop")
  FOREVER
  DBG_PRINT("After Loop");
  return EXIT_SUCCESS;
}
```

Preprocessed to?

```c
int main(void)
{
  int myBool = 1;
  printf("[DBG]:%s\n","Before Loop");
  while(1);
  printf("[DBG]:%s\n","After Loop");;
  return EXIT_SUCCESS;
}
```

#define and side effects

```c
#define DIVIDE(a,b)  a/b

int main(void)
{
  float f1 = DIVIDE(16.0, 2.0);
  float f2 = DIVIDE(16.0, 2.0 - 1.0);
  return EXIT_SUCCESS;
}
```

Preprocessed to??

```c
int main(void)
{
  float f1 = 16.0 / 2.0;       // 2.0 - OK!
  float f2 = 16.0 / 2.0 - 1.0; // 7.0 - NOT OK!
  return EXIT_SUCCESS;
}
```

#define and side effects (solved)

```c
#define DIVIDE(a,b)  (a)/(b)

int main(void)
{
  float f1 = DIVIDE(16.0, 2.0);
  float f2 = DIVIDE(16.0, 2.0 - 1.0);
  return EXIT_SUCCESS;
}
```

```c
int main(void)
{
  float f1 = (16.0) / (2.0);       // 2.0 - OK!
  float f2 = (16.0) / (2.0 - 1.0); // 16.0 - OK!
  return EXIT_SUCCESS;
}
```

Conditional *compilation*

```
#if condition
  :
 Some code here
  :
#else
  :
 Some other code here
  :
#endif
```

```
#if defined(name)
   :
  Some code here
   :
#else
   :
  Some other code here
   :
#endif
```

! operator works.

Conditional *Compilation*

```c
#define MYDEBUG 1
int main(void)
{
#if MYDEBUG
  printf("Running in MYDEBUG mode.");
#else
  printf("Some other mode.");
#endif

  return EXIT_SUCCESS;
}
```

```c
#define MYDEBUG
int main(void)
{
#if defined(MYDEBUG)
  printf("Running in MYDEBUG mode.");
#else
  printf("Some other mode.");
#endif

  return EXIT_SUCCESS;
}
```

Have a look at iom2560.h

- Some macros in there
- Never ever change these (unless you know what you’re doing).

→

You have been warned!

Headers

- Contains declarations
- So there need to be definitions somewhere!
- AKA *Include files* .
- Standard libraries (need .lib-files!)
- Modularisation[1]
- Abstract Data Types[1]

[1]Not int this course

Headers

- File (.h)
- Referred to by using #include
- #include <stdio.h> ↔ #include ”stdio.h”
- Custom made header files.
- Typically #include ”filename.h”
- Problem with circular includes.
- #ifndef + #endif

Some standard header files

File	Content
assert.h	Diagnostics.
math.h	Mathematical functions.
stdio.h	Input and output functions.
stdlib.h	Number conversions, storage allocations etc..
string.h	String handling.
time.h	Manipulating time and date.

Example

a) Create a function that calculates the square of a float value.
   a) Using normal return
   b) Using returning through params.

Example

a) In a project You have two values that both have their value range from 0-15. Find a way to store these in as little space as possible.

b) Also, it should be possible to assign and retrieve these.

Example

- Create a module (= .h and .c file) where there are two math functions for adding and subtracting two integer values.
  - Use the module.