

C Common Defects

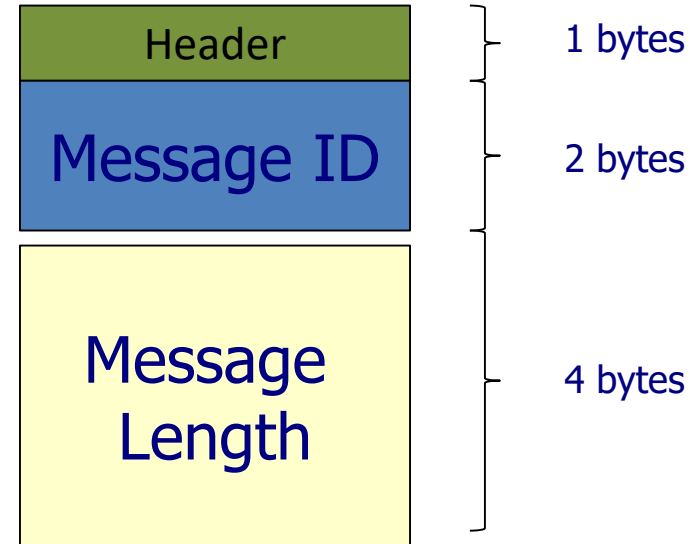
<Training Topic /Lesson Name>



```
typedef struct message {  
    char          head_byte;  
    short         message_id;  
    unsigned int  length;  
} message_header;  
void receive_data ( void *buffer)  
{  
    message_header m;  
    if(7 == sizeof(m))  
        m = &((message_header)buffer);  
}
```

- Are there any problems? If yes, how to fix them???

The format of message:



```
struct struct1{  
    double d1; /* 8 bytes */  
    char c; /* 1 byte */  
    double d2; /* 8 bytes */  
};
```

padding



```
struct struct1{  
    double d1; /* 8 bytes */  
    char c; /* 1 byte */  
    char padding[7]; /* 7 bytes */  
    double d2; /* 8 bytes */  
};
```

- You might think that `sizeof(struct my_struct)` should be 17 bytes, but it's actually 24 bytes. This is because of self-alignment. Compiler inserted 7 bytes of padding between `c` and `x` to keep the structure aligned.

- `#define SQUARE(x) x*x`
`int x = 2;`
what will be `SQUARE(x + 1)` ?
- `#define DOUBLE(x) x+x`
`int x = 3;`
what will be `DOUBLE(++x)` ??
- How to fix it???

- `#define max(a,b) ((a) > (b) ? (a) : (b))`
is this always run true???
- ```
biggest = x[0]
int i = 1;
while(i<n)
 biggest = max(biggest, x[i++]);
```

  
biggest will be the biggest number in x array???
- Never pass an expression that has side effects as a macro argument

- `#define MAX 10;`

```
int arr[MAX];
```

What's problem?

- `int x = 1, y = 2;`

```
if (x > y);
```

```
 printf("Impossible\n");
```

```
if (x < y);
```

```
 printf("Possible\n");
```

What it will print???

```
int test(int i)
{
 if(i > 0)
 return
 i = -1;
 if(i < 0)
 return
 i = 2;
}
int a = -1;
printf("%d", test(a));
```

What it will print???

- `int x;`  
Consider to `x*3/5`. What problem in this expression???
- Can be fix that: `(x/5) *3` ???  
`int x;`  
`(float) (3/5) *x; ???`
- `char buf[8];`  
`printf("What is your name?\n");`  
`scanf("%s", buf);`  
What's problem???
- `float x;`  
Consider to `(x/1e20) *1e30;`. What problem in this expression???



- `const char *p;`  
`char * const p;`  
what is difference???
- `const char *p;`  
`const char p[];`  
what is difference???

```
■ char x;
 for (x=0; x<200; x++)
 printf("%d ", x);
 putchar('\n');
```

What's problem??

- `unsigned char c;`  
`c = '\xff';`  
`if ( c != '\xff' ) printf( "Impossible!\n" )`  
`else printf( "Possible!\n" )`  
**what it will print???**
- `char *p = "ab";`  
`char p1[2] = { 'a', 'b' };`  
**are they identical???**

```
■ float f = 0.1;
 if (f != 0.1)
 printf("Impossible");
 else
 printf("Possible");
```

What it will print???

- `if (-5 <= x <= 5) {...}`

is it wrong???

what it mean???

- ```
if ( x < 0 ) {  
    printf( "Invalid value.\n" );  
    exit;  
}
```

is it exit if x is a negative number???

- ```
int x[10][10];
int y = x[++i, ++j];
```

C doesn't actually have true multi-dimensional arrays

- ```
x[ ++i, ++j ] ~ * ( x + ( ++j ) )
```

Which is an address, not an integer.

In C, always use one pair of [] for each level of array subscripting

- `char c = '\n';`
- `char *p = "\n";`
`printf("%s", &c);`
`printf("%s", p);`

Is it the same???

How to fix???

- *r* to an 8-bit value whose low-order bits are those of *l* and whose high-order bits are those of *h*:
`r = h<<4 + l;`
- *but the real mean:* `r = h << (4 + l);`
- *How to fix???*
- `r = (h << 4) + l;`
- `r = h << 4 | l;`
- `*p++` *is ???*

- *Arithmetic operators* ($++$, $--$, $+$, $-$, $...$)
- *Shift Operators* ($<<$, $>>$)
- *Relation Operator* ($==$, $!=$, $<$, $<=$, $>$, $>=$)
- *Logical Operators* ($\&\&$, $||$)
- *Assignment Operators*

- ```
void b(char **p) {
 char * str="print this string";
 *p = str;
}

int main(void) {
 char * s;
 b(&s);
 s[0]='j';
 return 0;
}
```
- What's problem???

- ```
int main(void) {  
    char *line = NULL;  
    size_t size = 0;  
    getline(&line, &size, stdin);  
    return 0;  
}
```

- What's problem???

- `if (xcnt < 2)`
 `return`

- `date = x[0];`

- `time = x[1];`

What it mean???

- `int x = 3;`

- `int *p = &x;`

- `int y = x/*p /* p point to x */;`

What is value of `y`???

- `int *g(), (*h)();`
are these the same???
- ```
struct foo{
 int x;
}
f()
{
 ...
}
```

**what's problem???**

- ```
char * curstr;  
char * prvstr;  
curstr = (char *) malloc( 10 );  
prvstr = (char *) malloc( 10 );  
strcpy( curstr, "abc" );  
prvstr = curstr;  
strcpy( curstr, "xyz" );  
what is prvstr value???
```
- ```
*prvstr = *curstr; ???
```

- `char *p;`  
`if(p == (char *) 0) ...`  
`if(strcmp(p, (char *) 0) == 0) ...`

**are they the same???**

- `char c;`  
`while ((c=getchar()) != EOF)`  
 `putchar(c);`

**How this code works???**

- Definition: Function pointer is a pointer that points to functions

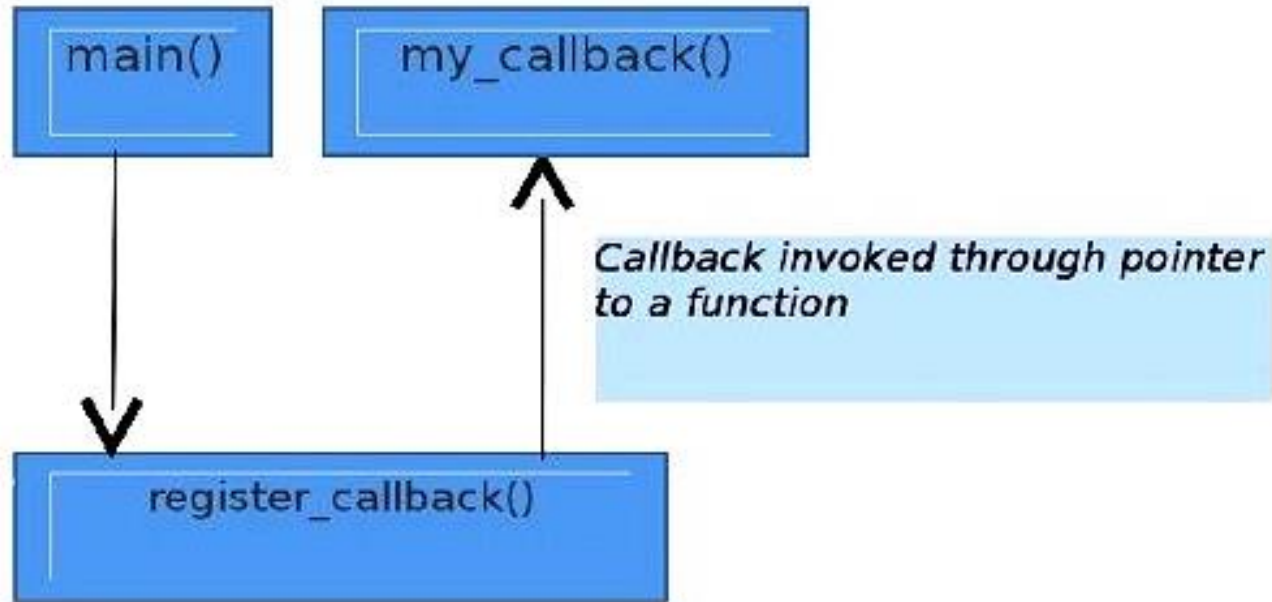
- Declaration:

```
<return_type> (* pfunc) (arg1, arg2);
```

- Purpose
  - ✓ Menu implementation
  - ✓ Callback function



- Callback function



# Thank you

