Eng. Ahmed Assaf — Embedded Systems — 6.12.2018

Function like Macro

```
#define Func(x,y)  x+y
```
(without space otherwise it will be considered as normal macro)

```
main()
{
   int z=Func(3,4);  →  int z=3+4;
   int y = func(Ahmed,Ali) →  int y= Ahmed+Ali;
}
```

الفرق الوحيد بين Func like macro و object like macro

Func like macro ↓ بيكون فيه قوس بعد اسم الـ Func مباشرة

object like macro ↓ بيكون فيه space

---

*Multi Line Macro

```
#define printCV()   printf("Ahmed");\
                    printf("27");\
                    printf("Zayed")

main()
{  printCV();
   → printf("Ahmed")
     printf("27")
     printf("Zayed");
}
```

```
main()
{
  for(i=0;i<10;i++)
     printfCV()
}
```

بس فيه مشكلة كارثية لو المشكلة من غير أقواس

الحل اني أكتب أقواس

```
#define print(V()      { printf("Ahmed");   \
                          "  ("27");         \
                          "  ("Zayed");  }
```

المشكلة التانية ←

```
main()
{
    if(——)
        Print CV();
    else
    {

    }
}
```

الحل ←

```
#define printCV()     do
                      { Printf("Ahmed");   \
                        ————————;          \
                        ————————;   } while(0)
```

Note

When ever you are using Multi-line macros

you should put it in do while(0)

```
do
{

}while(0)
```

```
# define   Multiply(x,y)        x*y

main()
{
   int z = Multiply(2,3+4);  →  int z = 2*3+4   X
```

الحل :- استخدام الأقواس

```
#define Multiply(x,y)   (x)*(y)
```

Second Rule

Use Round Brackets with Arguments

VV Note

there are two functions which can't be writen without using "function like Macro". they are:

① Stringfication #

```
#define Func(x)      #x

main()
{
   Func(Ahmed);  →  "Ahmed"
}
```

Another ex

```
#define print(x)    printf(#x)
   print(Ahmed) →
```

للهو ادوات الأطفال

② Concatination

- `#define conc(x,y)    xy`

```
main
{
    int z = conc(3,4);   →   z = 34;
}
```

example (to know the importance)

SPIReg

b7	---	b2	b1	b0

SPIReg = 0b101

built in

```
#define SPIReg_b0    0
#define SPIReg_b1    1
      |              0
      |              1
      |              0
```

SPIReg = 0b ## SPIReg_b0 ## SPIReg_b1 ## ...

الـ IDE هيعمل concat. لوحده

- #include
- #define
- Conditional directives

```
#if x==3        → لازم يكون object like macro
  ===
#elif ===
#elif ===
#else
#endif
```

```
#define AVR 1
#define PIC 2
#define ARM 3
#define Microcontroller
#if Microcontroller == AVR
  ===
#elif Microcontroller == PIC
  ===
#elif Microcontroller == ARM
  ===
#else
  #error "undefine microcontroller"
#endif
```

ملاحظة هيكتب error في cmd والـ main بتاع ال... هيكون فاضي

لو استعملت

#warning "undefine Microcontroller"

printf(AVR);

هيكتب warning في cmd و هيكتب printf(AVR) في (main)

ملاحظة :-

- الـ #if لو لقيت إن أحد أطراف الـ macro هو undefine فهيعتبره بـ Zero

وبالتالي لو مفي أي

وعلشان كده

* It's forbidden to #define macro (0)

- لو استعملت

#define AVR [ ] ← فاضي

#define Micro ARM

#if Micro == AVR → Pre processor error

- في حالة الـ #if

الـ #define Macro لازم يكون ليه unique number علشان الـ Statement (comparison)

ولازم ميكونش الـ #define Macro بـ Zero لأنه هيعامل معاملة الـ undefine macro.

Conditional directives

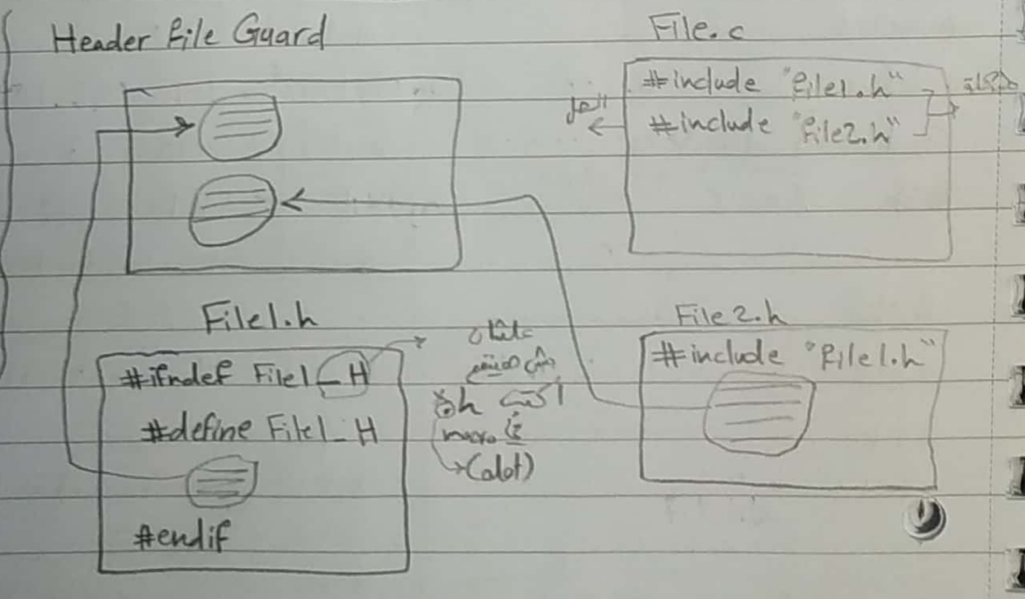
#define Ahmed

#ifdef Ahmed

#endif

#ifndef Ahmed

#endif



#undef اس کو defined کو undefine کرتا ہے

Ex:

```
#define x 10
=
=
=
#undef x
#define x 20
=
=
=
```

---

* Data types

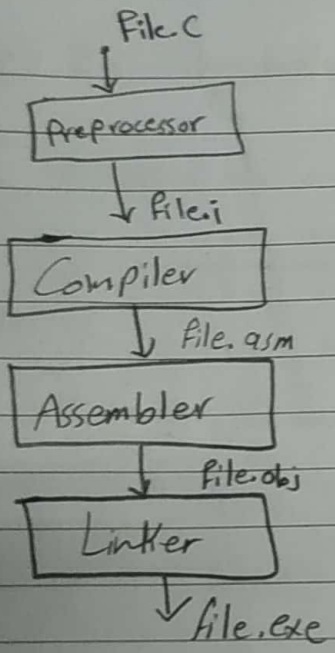
Primitive	Derived	User defined
char	Pointer	Struct
int	Array	Union
float	Function	enum
double		
void		

* Modifiers

Sign	signed	unsigned		
Size	short	long	*long long (not standard)	
Storage	auto	register		
	extern	static		
Volatility	Volatile			
Constant	const			
Pointer*	near	Far	huge	
Calling* Function*	cdecl	Pascal	stdCall	fCall
	inline			

Operators in C

Arithmetic	uniry	++	--			
	Binary	+	-	*	/	%
Relation	>	<	>=	<=		
	==	!=				
Logical	&&	||	!			
Bitwise	&	|	~	^	>>	<<
Assignment	=	+=	-=	*=	/=	
	&=	|=	^=	>>=	<<=	
Other	Ternary	? : ;	sizeof()			
	Address &	Dereference *				
	Subscriptor []	Calling ( )				
	Dot .	Arrow ->				
	ellipses ...	Camma ,				
	Colon :	Semi Colon ;				


File.C
Preprocessor
File.i
Compiler
File.asm
Assembler
File.obj
Linker
file.exe