

Informática  
Ingeniería en Electrónica y Automática Industrial  
El preprocesador de lenguaje C

V1.2 © Autores

## El preprocesador de lenguaje C

- Definición
- Directivas de preprocesado
- Constantes simbólicas y macros
- Directivas de compilación condicional

V1.2 © Autores 2

## Definición

- El **preprocesador** del lenguaje C
  - Procesador de texto que manipula el fichero fuente
  - Actúa en una fase previa a la compilación
  - Facilita la programación
  - Las *instrucciones* en este caso se llaman **directivas o directrices de preprocesado**

V1.2 © Autores 3

## Directivas de preprocesado (I)

- Las *directivas o directrices de preprocesado* son instrucciones especiales que condicionan una parte preliminar en el proceso de compilación completo
  - Formalmente no son parte del lenguaje C
  - Hay un conjunto de ellas establecidas por el estándar ANSI. Los compiladores suelen incluir algunas particulares
  - Siempre comienzan con el símbolo «#»
  - Al no ser sentencias de lenguaje C no finalizan con el «;»
  - Sólo pueden ocupar una línea. Para extender una directiva más de una línea, se utiliza el símbolo «\» antes del cambio de línea
  - Pueden colocarse en cualquier parte del fichero fuente, pero sólo actúan a partir de la línea en la que se encuentran

V1.2 © Autores 4

## Directivas de preprocesado (II)

- Las directivas incluidas en el ANSI C son
  - #include            #define
  - #error             #if
  - #elif                #else
  - #ifdef              #ifndef
  - #endif              #undef
  - #line                #pragma

V1.2 © Autores 5

## Directivas de preprocesado (III)

- **#include**
  - Hace que el compilador incluya en el fichero fuente el archivo que se indica, llamado *fichero cabecera* (header file)
    - #include <nombrefichero>
    - #include "nombrefichero"
  - **nombrefichero** Representa el nombre del archivo, con extensión .h
    - Si va entre los símbolos <> el compilador busca el archivo en el directorio del sistema
    - Si va entre comillas dobles "", el compilador lo busca en el directorio actual o directorio de trabajo

V1.2 © Autores 6

## Directivas de preprocesado (IV)

- **#define**
  - Permite definir *constantes simbólicas* y *macros*  
`#define NOMBREMACRO Contenido a sustituir`
- **#undef**
  - Elimina la definición de una constante simbólica o macro previamente definida  
`#undef NOMBREMACRO`
- **#error**
  - Fuerza una parada del compilador en la línea de la directiva presentando el mensaje que sigue a la directiva  
`#error Mensaje a mostrar en pantalla`
- **#line**
  - Fuerza un cambio de valor en las constantes simbólicas `_LINE_` y `_FILE_` que representan el número de línea y el archivo que está siendo objeto de compilación  
`#line númerolínea "nuevonombreachivo"`

V1.2

© Autores

7

## Directivas de preprocesado (V)

- **#pragma**
  - Sirve para que cada compilador pueda producir compilaciones particularizadas, de acuerdo con la sintaxis que defina el propio compilador  
`#pragma opcióndecompilación`
- **#if**      **#elif**      **#else**  
**#endif**    **#ifdef**      **#ifndef**
  - Son directivas que permiten la compilación condicional de diferentes bloques de código

V1.2

© Autores

8

## Constantes simbólicas y macros. Directiva #define (I)

- Permite definir *constantes simbólicas* y *macros*
    - Una **constante simbólica** es un identificador que se asocia a una cadena o constante.
      - Durante el proceso de compilación se sustituye la constante simbólica por la cadena asociada a ésta antes de generar el código
- ```
#define IDENTIFICADOR cadena o constante
```
- **IDENTIFICADOR** Es la etiqueta y se denomina *nombre de macro*. Normalmente se escribe con mayúsculas
  - **cadena** es un conjunto de caracteres o símbolos que sustituirán a **IDENTIFICADOR** siempre que este aparezca en el programa

V1.2

© Autores

9

## Constantes simbólicas y macros. Directiva #define (II)

- Una **macro** es un identificador que se asocia a una expresión en la que puede haber elementos reemplazables  

```
#define NOMBREMACRO (parámetros) expresión
```

  - **NOMBREMACRO** Es el identificador o nombre (en mayúsculas)
  - **parámetros** Representa una lista de parámetros reemplazables, separados por comas, que forman parte de la macro
  - **expresión** Es cualquier expresión válida en C que opere con los parámetros incluidos en la macro
- El compilador sustituye el nombre de la macro por la expresión o sentencia asociada, cambiando los parámetros de la macro por los argumentos que acompañan al nombre de la macro en su llamada

V1.2

© Autores

10

## Constantes simbólicas y macros. Directiva #define (III)

- Ejemplo de macro:

```
#define MAYOR(a,b) ((a)>(b)) ? (a) : (b)
...
max = MAYOR(dato1, dato2);
/* max es igual al mayor de dato1 o dato2 */
```

  - Es similar a una función en su apariencia
    - Genera mayor cantidad de código (ocupa más memoria)
    - En ejecución, es más rápido que una llamada a función (no hay accesos a la pila)
  - Algunas de las funciones de los compiladores son, en realidad, macros (`getc()` y `getchar()`, por ejemplo)
  - Los parámetros no son variables de ningún tipo. En la sentencia de la macro deben ponerse entre paréntesis para evitar efectos indeseados en la sustitución

V1.2

© Autores

11

## Constantes simbólicas y macros. Directiva #define (IV)

- En ANSI C hay cinco macros predefinidas
  - **\_LINE\_** Representa el número de línea que se está compilando en cada momento
  - **\_FILE\_** Contiene una cadena con el nombre del fichero fuente que se está compilando
  - **\_DATE\_** Representa la fecha de traducción del código fuente a código objeto en formato "mes día año"
  - **\_TIME\_** Contiene la hora de traducción del código en el formato "hora:minuto:segundo"
  - **\_STDC\_** Contiene la constante decimal 1 si la implementación se ajusta al estándar ANSI

V1.2

© Autores

12

### Directivas de compilación condicional (I)

- Las **directivas de compilación condicional** permiten la compilación selectiva de partes del fichero fuente
  - Facilitan la depuración
  - Permiten personalizar los programas
- Tipos:
  - Compilación condicionada al valor de una expresión:
 

```
#if      #elif      #else      #endif
```
  - Compilación condicionada a la definición de constantes simbólicas
 

```
#ifdef      #ifndef      #endif
```

V1.2

© Autores

13

### Directivas de compilación condicional (II)

- Directivas de compilación condicionada al valor de una expresión (I)
 

```
#if expresiónconstante1
    secuencia de sentencias 1;
#elif expresiónconstante2
    secuencia de sentencias 2;
#elif expresiónconstante3
    .....
#elif expresiónconstanteN
    secuencia de sentencias N;
#else
    secuencia de sentencias M;
#endif
```

V1.2

© Autores

14

### Directivas de compilación condicional (III)

- Directivas de compilación condicionada al valor de una expresión (II)
  - Los términos `expresiónconstante` se evalúan en tiempo de compilación
    - Pueden incluir operaciones lógicas y relacionales
    - No pueden incluir variables del programa
  - Mediante secuencia de sentencias X; se representan múltiples líneas de código en lenguaje C
  - La directiva `#elif` es equivalente a `#else #if`
  - Las directivas `#else` y `#elif` van asociadas a la `#if` más próxima hacia arriba y son opcionales

V1.2

© Autores

15

### Directivas de compilación condicional (IV)

- Directivas de compilación condicionada a la definición de una macro
 

```
#ifdef NOMBREMACRO1
    secuencia de sentencias 1;
#endif
#ifdef NOMBREMACRO2
    secuencia de sentencias2;
#endif
```

  - La secuencia de sentencias 1; se procesa (compila) si `NOMBREMACRO1` está definida previamente
  - La secuencia de sentencias 2; se procesa si `NOMBREMACRO2` NO está definida previamente
  - Puede incluirse la directiva `#else` como alternativa a las directivas `#ifdef` y `#ifndef`, pero no se permite `#elif`

V1.2

© Autores

16