GCC – #items

Petr Novák / novakpe@fel.cvut.cz / 2021-04-29

Obsah

1 Úvod

Klíčová slova uvozená znakem „#“ představují direktivy pro kompilátor. Pomocí nich je v podstatě řízen překlad.

2 #define

Slouží i definici nějaké (textové) konstanty. Jde (v naprosté většině případů) v podstatě pouze o náhradu jednoho textu jiný textem. Pokud je v libovolném zdrojovém souboru nalezen text obsažený na levé straně (přímo za **#define**), je nahrazen textem obraženým na pravé straně (dále od **#define**). Mezi těmito dvěma texty není rovnítko, první text nesmí obsahovat mezeru, mezera značí konec prvního textu a začátek druhého. Tato náhrada se vykoná samozřejmě ještě před vlastní kompilaci.
Pro přehlednost je vhodné pro konstanty prát text velkým písmem a pro oddělení slov používat podtržítko („_“).
Nejjednodušší možnost definice je (za touto definicí není středník, jinak by byl součástí definice):

#define PREVODNI_KONSTANTA 123

Pokud se někde v textu vyskytuje například:

int vysledek = hodnota * PREVODNI_KONSTANTA;

tak před kompilací bude tento text nahrazen textem:

int vysledek = hodnota * 123;

#define se zásadně liší od **const** (definované konstanty). **#define** nezabírá žádnou paměť, jde tedy pouze o náhradu jednoho textu za jiný před vlastní kompilací, při kompilaci v podstatě původní text neexistuje, neexistuje v podstatě ani **#define**. **const** je hodnota (běžně) uložená v paměti, tedy je pro ni v paměti vyhrazeno místo a chová se jako běžná hodnota označena jako „neměnná“, tedy konstantní (pouze pro čtení). Samozřejmě záleží na optimalizaci kompilátoru, zda pro hodnotu typu **const** nevytvoří paměťové místo a přímo ji vloží do programového kódu na místo jejího použití, pak je toto v podstatě obdoba typu **#define**).
Hodnotu definovanou pomocí **#define** lze použít nejčastěji pro podmíněnou kompilaci nějakého kusu zdrojového kódu, například:

// definice verze aplikace

```
#define VERZE_2

#ifdef VERZE_2 // to samé jako #if defined(VERZE_2)
// programový kód pro verzi 2
#else
// programový kód pro verzi 1
#endif
```

Pokud některá část podmínky není platná, tak se ani nekompiluje. V jejím těle tedy může být cokoli, i chyby a kompilátorem není nic hlášeno. V podstatě, tako by text v nesplněné části podmínky, pro kompilátor vůbec neexistoval. Podmínky lze samozřejmě i řetězit (v řetězených podmínkách je potřeba používat raději **#if defined(…)** místo **#ifdef**):

```
#if defined(VERZE_5)
  // programový kód pro verzi 5
#elif defined(VERZE_4)
  // programový kód pro verzi 4
#elif defined(VERZE_3)
  // programový kód pro verzi 3
#elif defined(VERZE_2)
  // programový kód pro verzi 2
#else
  // programový kód pro verzi 1
#endif
```

Pro podmíněnou kompilaci podle různých konstant lze využít i jinou možnost, a to následující ():

```
#define VERZE 2

#if VERZE == 3
  // programový kód pro verzi 3
#elif VERZE == 2
  // programový kód pro verzi 2
#else
  // programový kód pro verzi 1
#endif
```

Jednou již definovanou konstantu je možno samozřejmě později i zrušit to pomocí (konstanta již dále neexistuje):

```
#undef VERZE_5
```

Rovněž je možno již existující konstantu předefinovat na jinou, například takto:

```
#ifdef PREVODNI_KONSTANTA 123          // pokud je definováno
   #undef PREVODNI_KONSTANTA           // tak definici zrušit
#endif
#define PREVODNI_KONSTANTA 124         // a vytvořit novou
```

Konstantu lze v podstatě i re-definovat, tedy definovat znova obsahující novou hodnotu, toto však způsobí varování při kompilaci a není to tedy dobré.
Někdy je vhodné poskytnout nějakou výchozí hodnotu (nejčastěji v případě nějakého bufferu / velikosti paměti), když uživatel žádnou vlastní nedefinuje, ale pokud ji definuje, tak použít jeho. Například následovně:

```
#ifndef PREVODNI_KONSTANTA                // pokud není definováno
  #define PREVODNI_KONSTANTA 123          // použije se tato hodnota
#endif
```

Podmínky lze rovněž spojovat pro vytvoření složitějších podmínek, například:

```
#if defined(VERZE_5) || defined(VERZE_4)
#if defined(UART_SEND) && defined(UART_RECEIVE)
```

Poznámky:

- defined(…) testuje, zda zadaná konstanta / text (vůbec) existuje, tedy bez ohledu zda jde pouze o textovou definici (a nic více) nebo text s nějakou přiřazenou konstantu.
- == porovnává, zda definovaná konstanta obsahuje požadovanou hodnotu. Pokud definice vůbec neexistuje, tak samozřejmě požadovanou hodnotu neobsahuje.

3 #warning a #error

Tato dvě klíčová slova způsobují hlášení kompilátoru. **#warning** způsobí pouze varovné hlášení a kompilace pokračuje dále. **#error** způsobí chybové hlášení a kompilace za po nastaveném / zadaném počtu celkem detekovaných chyb zastaví. Způsob použití může být následující (zadaný text se zobrazí ve výpisu kompilátoru):

```
#if defined(VERZE_2)
  // programový kód pro verzi 2
#elif defined(VERZE_1)
  // programový kód pro verzi 2
#else
  // není zadána verze, program nelze kompilovat
  // (použít pouze #warning nebo #error)
  #warning !!! Varování – není zadána verze aplikace !!!
  #error !!! ERROR – není zadána verze aplikace !!!
#endif
```