Il preprocessore

Alessandro Pellegrini
a.pellegrini@ing.uniroma2.it

Il preprocessore

- Il preprocessore C è un preprocessore di macro per i linguaggi di programmazione C, Objective-C e C ++.
- Le sue funzionalità fondamentali sono:
  - inclusione di file di intestazione
  - espansione di macro
  - compilazione condizionale
  - controllo di riga

- Il linguaggio delle direttive del preprocessore è solo debolmente correlato alla grammatica del C, e quindi a volte è usato per elaborare altri tipi di file di testo (anche codice assembly)

Inclusione di file

- Uno degli usi più comuni del preprocessore è includere un altro file.
- Il preprocessore sostituisce, ad esempio, la riga

```
#include <stdio.h>
```

- con il contenuto testuale del file 'stdio.h'.
- Se il nome del file è racchiuso tra parentesi angolari, il file viene cercato nel compilatore standard include percorsi.
- Se il nome del file è racchiuso tra virgolette doppie, il percorso di ricerca viene espanso per includere la directory del file sorgente corrente

Compilazione condizionale

- Le direttive `#if`, `#ifdef`, `#ifndef`, `#else`, `#elif` e `#endif` possono essere utilizzate per la compilazione condizionale.
- `#ifdef` e `#ifndef` sono semplici scorciatoie per:
  - `#if defined (...)` e `#if !defined (...)`.
- Esempi:

```
#if VERBOSE >= 2
  printf("trace message");
#endif
```

```
#ifdef __unix__
# include <unistd.h>
#elif defined _WIN32
# include <windows.h>
#endif
```

Definizione ed espansione di macro

- Vi sono due tipi di macro:
  - Macro oggetto:
    `#define <identifier> <replacement token list>`
  - Macro funzione:
    - `#define <identifier>(<parameter list>) <replacement token list>`
- Esempi:
  - `#define PI 3.14159`
  - `#define RADTODEG(x) ((x) * 57.29578)`
- Una macro può essere rimossa:
  - `#undef <identifier>`

Un esempio

- `#define MAX(a,b) ((a) > (b) ? (a) : (b))`
- `#define MIN(a,b) ((a) < (b) ? (a) : (b))`

- Cosa succede se scrivo del codice di questo tipo?

```
int x = 20, y = 10;
int max = MAX(x++, y);
```

- Double evaluation side effect:
  - `int max = ((x++) > (y) ? (x++) : (y));`

Utilizzando le *statement expressions*

```
#define max(a,b)             \
({                           \
    __typeof__ (a) _a = (a); \
    __typeof__ (b) _b = (b); \
    _a > _b ? _a : _b;       \
})

#define min(a,b)             \
({                           \
    __typeof__ (a) _a = (a); \
    __typeof__ (b) _b = (b); \
    _a < _b ? _a : _b;       \
})
```

Generare errori o warning nel preprocessore

- Sono presenti due direttive speciali:
  - `#warning Message`: stampa “Message” a schermo durante la compilazione, come warning
  - `#error Message`: stampa “Message” a schermo e fa fallire la compilazione

- È un supporto utile se unito, ad esempio, alla compilazione condizionale:

```
#ifdef __unix__
# include <unistd.h>
#elif defined _WIN32
# error Unsupported operating system
#endif
```

Macro variadiche

- È possibile definire una macro variadica:
  - `#define macro(M, ...)`
  - L'elenco dei parametri può essere recuperato utilizzando la direttiva di preprocessore `##__VA_ARGS__`

- Ad esempio:

```
#define print(fmt, ...) printf(fmt, ##__VA_ARGS__)
```

Macro speciali

- Sono presenti alcune macro speciali, definite a tempo di compilazione automaticamente:
  - `__FILE__`: Il nome del file in cui viene utilizzata la macro
  - `__LINE__`: La riga del file in cui viene utilizzata la macro
  - `__func__`: Il nome della funzione in cui viene utilizzata la macro
  - `__STDC_VERSION__`: La versione del C utilizzata dal compilatore
  - `__DATE__`: La data di compilazione
  - `__TIME__`: L'ora di compilazione

Esempio: macro di debug

```
#ifdef NDEBUG
#define debug(M, ...)
#else
#define debug(M, ...) fprintf(stderr, "DEBUG %s:%d: " M "\n", \
                __FILE__, __LINE__, ##__VA_ARGS__)
#endif

#define check(A, M, ...) if(!(A)) { \
                log_err(M, ##__VA_ARGS__); }

#define log_err(M, ...) fprintf(stderr,\
                "[ERROR] (%s:%d: errno: %s) " M "\n", __FILE__, __LINE__, \
                (errno == 0 ? "None" : strerror(errno)), ##__VA_ARGS__)
```