const & volatile

mobytrick

2 settembre 2014

Nel Linguaggio C `const` e `volatile` sono considerati dei *qualificatori*. Governano le modalità di modifica di una variabile. In macanza di entrambi il contenuto di una variabile può essere modificato in varie maniere. Ma in certe circostanze è bene disporre di uno strumento software che impedisca la manomissione del contenuto.

const qualifica la variabile come *di sola lettura* ovvero il contenuto non può essere modificato. Pertanto per questo tipo di variabili il valore viene specificato assieme alla loro tipizzazione. Ad esempio, volendo memorizzare in una variabile qualificata `const` -supponiamo `pi`- il valore di π, si procede nel seguente modo:

```
const double pi = 4. * atan (1.);
```

Le due parole chiave (`const` e `double`) possono essere invertite senza ottenere effetti collaterali indesiderati.

Tuttavia non è possibile effettuare delle *furbate*, tipo tentare di carpire l'indirizzo di una variabile per manometterne il contenuto. Il compilatore gcc ha un comportamento che lascia basiti. Segnala, infatti, il tentativo con un *warning*, cosa che non impedisce la creazione del file object e poi dell'eseguibile. L'avvertimento è del tipo *guarda che elimino la qualifica di const*, laddove ci si sarebbe aspettati non un avvertimento bensì un errore bloccante. Lo standard del linguaggio però prescrive questo preciso comportamento.

La qualifica `const` è applicabile pure ai puntatori, ma ponendo la massima attenzione in fase di dichiarazione. Si osservino le seguenti dichiarazioni e le implicazioni connesse.

1. `const int *alfa;`

   La variabile `alfa` è tipizzata come *puntatore ad una variabile costante intera*. Ciò che non può essere modificato (ovvero ciò che deve restare costante) è la variabile puntata. Pertanto la variabile `alfa` può essere

dichiarata senza dover essere contestualmente inizializzata. È possibile pure alterare il suo valore per recuperare il contenuto di un'altra variabile. Ciò che invece non è possibile è la modifica della variabile puntata. Se si tenta di violare la prescrizione, ci pensa il compilatore ad emettere una segnalazione di tipo error, che impedisce la generazione del file object ed in via surrettizia dell'eseguibile. Utile nell'intestazione delle funzioni cui viene passato un indirizzo perché cosí ci si assicura che la funzione non altererà i parametri.

2. `int * const beta = valore;`

   `beta` è tipizzato come *puntatore costante ad una variabile intera*. In quanto tale la variabile `beta`, oltre che dichiarata, deve essere anche inizializzata. Poiché la qualifica di rimanere costante è riferita al puntatore ma non all'oggetto puntato, il valore di quest'ultimo può essere modificato.

3. `const int * const frozen = valore;`

   Con la dichiarazione di cui sopra tutto rimane fermo, imbalsamato. Sia il puntatore che il contenuto della variabile. Necessaria, ovviamente, l'inizializ-zazione.

volatile indica che il contenuto della variabile cui si riferisce può essere modificato da un ambiente esterno a quello del programma. Pertanto il compilatore non può sottoporre tali variabili a processi di ottimizzazione, quali ad esempio porre il loro contenuto in un registro di memoria per velocizzare l'accesso.

—===oooOooo===—

Malgrado il nome, i qualificatori `const` e `volatile` possono coesistere. La dichiarazione di una variabile che li preveda entrambi sta ad indicare che il contenuto non potrà essere modificato dall'interno del programma. Cosa che invece potrà essere effettuata dall'esterno.