Warning Peaks – perchè i warning sono importanti in C (e C++) – pt.2

Dale Cooper: *Harry, ti dirò un piccolo segreto. Ogni giorno, una volta al giorno, fatti un regalo. Non pianificarlo, non aspettarlo... fallo solo succedere..*

Ok, è già ora di tornare sul pezzo. Dove eravamo rimasti? Ah, si: **nell'ultimo articolo** avevo parlato dell'importanza, spesso sottovalutata, dei **warning**, e avevo anche mostrato qualche (spero) interessante esempio. Avevo poi concluso dicendo che, nella seconda parte avremmo visto una panoramica dei *flag* consigliati per scrivere una applicazione impeccabile (*"a prova di bomba"*). E allora, come diceva (qui sopra) il buon **Dale Cooper** nel mitico **Twin Peaks**, oggi tocca auto-regalarsi qualcosa di molto utile… come avrete già intuito, anche in questo articolo parleremo di **warning**!



…Harry, regalati un warning e risolvilo. La tua applicazione ti ringrazierà…

Avevo già anticipato nello scorso articolo che alcuni **warning** appaiono di base, mente altri appaiono usando il *flag* di compilazione **-Wall**. È quindi venuto il momento di mostrare una lista degli altri *flag* che si possono usare, anche se vedremo solo i più significativi per lo scopo che ci siamo prefissi. I *flag* disponibili con **GCC** sono tantissimi, e alcuni sono dei "*flag di gruppo*" (come il **-Wall**) che raggruppano molte opzioni in una botta sola. Vedremo, quindi, i *flag di gruppo* più utili e anche alcuni *flag semplici* da usare separatamente. Vai con la lista!

Flag di gruppo

- **-Wall**: abilita i **warning** più comunemente utili, tra cui variabili non inizializzate, parametri inutilizzati e costrutti sospetti. Include ben 81 *flag semplici* che si possono usare separatamente (*-Waddress, -Warray-compare -Warray-parameter=2*, ecc.), ed è il *flag di gruppo "principe"*, quello assolutamente da usare.
- **-Wextra**: aggiunge **warning** più completi rispetto a *-Wall*, come parametri di funzione inutilizzati, variabili oscurate e dichiarazioni di funzione implicite. Include ben 29 f*lag semplici* che si possono usare separatamente (*-Walloc-size, -Wcalloc-transposed-args, -Wcast-function-type*, ecc.), ed è il *flag di gruppo "principe"* n.2, da usare sempre in coppia con *-Wall* (o anche da solo, ma in coppia è meglio… *scusate il doppio senso, ah ah ah*).
- **-Wformat=2**: combina i *flag semplici -Wformat, -Wformat-nonliteral, -Wformat-security* e *-Wformat-y2k* per rilevare l'uso non sicuro delle stringhe di formato. È un *flag* utile ma solo per usi avanzati e restrittivi, normalmente è sufficiente usare il *-Wformat=1* che è incluso nel *–Wall*.

Flag semplici

- **–pedantic** (oppure **-Wpedantic**): impone la stretta

conformità agli standard ISO C/C++, segnalando le estensioni non standard e il codice non conforme. È un *flag* altamente raccomandabile.
- **-Werror**: tratta tutti i **warning** come errori, impedendo la compilazione fino alla risoluzione dei problemi, ideale per il codice di produzione. È un *flag* utile ma non strettamente indispensabile; comunque è importante sapere che è disponibile.
- **-Wshadow**: avvisa quando una variabile locale oscura una variabile da uno scope esterno. È un *flag* altamente raccomandabile.
- **-Wnull-dereference**: avvisa in merito a percorsi che potrebbero portare a referenze di puntatori nulli. È un *flag* utile ma solo per usi avanzati e restrittivi.
- **-Wcast-align=strict**: rileva conversioni di *pointer* non sicure che potrebbero causare problemi di disallineamento su determinate architetture. È un *flag* utile ma solo per usi avanzati e restrittivi.

Che vi sembra la lista? È, ovviamente, una selezione molto ponderata, ma abbastanza completa; e comunque, se volete conoscere tutti il *flag* disponibili, **potete consultarli qui** (*ma mettetevi comodi, è una lista lunghissima!*).

A questo punto, magari, vi può anche interessare cosa usa, normalmente, *il vostro C-blogger preferito (io, no? ah ah ah)* quando scrive codice di produzione… ecco la linea!

```
gcc -Wall -Wextra -pedantic -Wshadow
```

a volte aggiungo, poi, anche alcuni *flag semplici* in base alle esigenze del momento, ma normalmente la linea di compilazione che uso è questa, e permette di produrre applicazioni molto affidabili, ve lo garantisco! *Provare per credere…*

E direi di finire con una altro esempietto, perché un articolo di programmazione senza un esempio reale è un articolo incompiuto… Vediamo un po': il codice seguente ha un problema,

ma compilando senza **warning**, o anche usando **-Wall** non segnala niente (!). Ci vorrà mica il **-Wextra**? Vai col codice!

```
// typelimits.c
#include <stdio.h> // prototipi locali
int isGreaterThan200(char c); // funzione main
int main() { int i = 250; printf("arg = %d --> %s\n", i,
isGreaterThan200(i) ? "è maggiore di 200" : "non è maggiore di
200"); return 0;
} // isGreaterThan200 - valuta se l'argomento è più grande di
200
int isGreaterThan200(char arg) { if (arg > 200) // 200 è
troppo grande per un char: la comparazione è sempre falsa
return 1; else return 0;
}
```

compiliamo con **-Wall** ed eseguiamo:

```
aldo@Linux $ gcc -Wall typelimits.c -o typelimits
aldo@Linux $ ./typelimits arg = 250 --> non è maggiore di 200
```

Ma che strano, è un programma molto semplice ed eppure funziona così male! E il compilatore non ci segnala nulla! Proviamo allora a compilare con **-Wextra**:

```
aldo@Linux $ gcc -Wextra typelimits.c -o typelimits
typelimits.c: In function ‘isGreaterThan200’:
typelimits.c:20:13: warning: comparison is always false due to
limited range of data type [-Wtype-limits] 20 | if (arg > 200)
// 200 è troppo grande per un char: la comparazione è sempre
falsa | ^
aldo@Linux $ ./typelimits arg = 250 --> non è maggiore di 200
```

Ok, l’errore di esecuzione è rimasto (*e ci mancherebbe solo!*), ma almeno sappiamo a cosa è dovuto e possiamo correggerlo facilmente. Questo è solo un esempio dei **warning** che può segnalare il **-Wextra**: spesso è veramente utile!

E, prima di chiudere, vorrei ricordarvi che tutti i *flag semplici* hanno un *flag* parallelo che serve a negare (sopprimere) il **warning** in oggetto, ed è lo stesso flag però con un *no-* davanti. Quindi, se vogliamo sopprimere uno di

quelli inclusi nel **-Wall** (ad esempio il *-Waddress*) possiamo cambiare la linea di compilazione (che ho passato sopra) in questa maniera:

gcc -Wall -Wextra -pedantic -Wshadow -Wno-address

Questa linea modificata aggiungendo un *-Wno-address* ci permette di compilare segnalando tutti i **warning** utili ma senza mostrare quelli di tipo *-Waddress*; e a cosa serve questo? Beh, il compilatore **GCC** è un gran strumento, ma a volte può sbagliare anche lui offrendoci dei falsi negativi, oppure scambiando per errore cose che abbiamo fatto volutamente, oppure mostrandoci delle segnalazioni che riteniamo completamente ininfluenti: in questo caso possiamo *"pulire"* l'uscita della compilazione dai messaggi che non ci interessano.

E vabbè, penso che è giunto il momento di chiudere questa larga parentesi sui **warning.** Non so cosa ci riserverà il futuro… ma sarà sicuramente molto interessante! Restate sintonizzati!

Ciao, e al prossimo post!

https://italiancoders.it/warning-peaks-perche-i-warning-sono-importanti-in-c-e-c-pt-2/