Funzioni VARIADICHE

venerdì 23 dicembre 2022 10:45

FUNZIONI VARIADICHE

È una funzione di "arietà variabile", ossia una funzione che può accettare un numero arbitrario di argomenti!

ALL'ATTO DELLA DICHIARAZIONE, NON AVENDO UN NUMERO PREFISSATO DI ARGOMENTI,

Come facciamo a specificare che da un certo punto in poi c'è un numero qualsiasi di argomenti?

UTILIZZIAMO 3 PUNTINI DI SOSPENSIONE ... !

- In C, nella definizione di tale funzione, si utilizzano tre puntini di sospensione come ultimo parametro
- In questo caso, le variabili vengono passate tramite stack
- La funzione variadica deve esplicitamente recuperare le variabili dallo stack una ad una, specificandone correttamente il tipo, sfruttando le definizioni in stdarg.h
- Un esempio è la funzione printf(), che determina in funzione della stringa di formato il tipo di ciascun parametro
- L'utilizzo delle funzioni variadiche può dare adito a problemi di sicurezza

Non sappiamo quanti parametri verranno passati, in generale i parametri vengono passati tramite STACK!

Devo anche dire la dimensione in byte di questi parametri!

Per questo motivo si utilizza un header STDARG.h che permette di utilizzare delle MACRO che comunicano, parametro per parametro, la dimensione!

ESEMPIO:

```c
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <stdarg.h>

int somma(int n, ...)
{
    int nSum = 0;
    va_list int_ptr;
    va_start(int_ptr, n);
    for(int i=0;i<n;i++)
    {
        nSum += va_arg(int_ptr, int);
    }
    va_end(int_ptr);

    return nSum;
}

int main()
{
    printf("10 + 20 = %d\n", somma(2, 10, 20));
    printf("10 + 20 + 30 = %d\n", somma(3, 10, 20, 30));
    printf("10 + 20 + 30 + 40 = %d\n", somma(4, 10, 20, 30, 40));

    return 0;
}
```

→ N numero di argomenti

→ Lista argomenti di una funzione variadica! → Mi tiene traccia sullo stack dell'ultimo argomento

→ dimensione argomento

→ I-esimo argomento

OUTPUT

```
10 + 20 = 30
10 + 20 + 30 = 60
10 + 20 + 30 + 40 = 100
```

I primi 6 argomenti verranno passati tramite registro e gli altri tramite stack;

I primi 6 argomenti vengono passati tramite registro e gli altri tramite stack;

- Nel caso di funzioni variadiche, valgono le convenzioni di chiamata standard
  - I primi sei parametri vengono passati per registro
  - I successivi vengono passati tramite stack

VA_ARG, che deve andare a leggere l'i-esimo argomento, diventa complessa da implementare!

↳ PERCHÈ I PRIMI 6 se li deve prendere da registro, i restanti su stack!

L'implementazione moderna che si adotta, non rende complessa VA_ARG, ma utilizza una call in convention che passa tutti gli argomenti tramite stack!

- Soluzione: copiare i parametri passati per registro su stack (`va_start`)
- Successivamente, ogni parametro può essere recuperato da stack

Ogni volta che viene chiamato va_start su x86 a 64 bit, i primi 6 parametri vengono scritti sulla cima dello stack!
E così ora, va_arg, può recuperare tutti gli argomenti tramite stack!

Se io dichiaro una funzione variadica e non utilizzo va_end prima dell'istruzione di "return", i parametri che va_start ha scritto sullo stack non vengono rimossi!
Corrompo lo stack ed entro in Undefined Behaviour;
Quasi sicuramente il mio programma andrà in CRASH!