C++ e puntatori a funzione

evilsocket
http://evilsocket.altervista.org/

Tutti coloro che conoscono il c++ sanno che questo linguaggio usa delle variabili di diverso tipo e (si spera) sanno anche come usare queste
variabili come argomenti di una funzione .
Ciò che non tutti sanno è che esiste un tipo particolare, che permette di usare delle funzioni come fossero variabili .
Questi tipi sono i cosiddetti "function pointers" o puntatori a funzione .
Come ogni puntatore, quelli a funzione letteralmente "puntano" all indirizzo di memoria dove è mappata la prima istruzione di una determinata
routine .
Facciamo un esempio di uno di questi puntatori .

```
#include <stdio.h>

typedef void (*puntatore_a_funzione_tipo)(char *);


void funzione( char * msg )
{
    printf( "%s\n", msg );
}


int main()
{
    puntatore_a_funzione_tipo p_funzione_variabile = funzione;
    p_funzione_variabile("ciao");


    return 0;
}
```

Come potete vedere nella seconda riga di codice, definiamo un tipo con la direttiva "typedef" :

```
typedef void (*puntatore_a_funzione_tipo)(char *);
```

Il che significa letteralmente "definici un nuovo tipo, chiamato puntatore_a_funzione_tipo, che punta ad una funzione generica la quale accetta
come argomento un 'char *'" .

Successivamente creiamo la funzione e nell blocco main definiamo

una variabile del tipo appena definito che punta alla funzione stessa .
Nella riga successiva vediamo come questa variabile può essere richiamata come una funzione .

Nel secondo esempio, simile al primo, vediamo come usare questi function pointers passandoli come variabili ad un altra funzione .

```
#include <stdio.h>

typedef void (*puntatore_a_funzione_tipo)(char *);


void funzione( char * msg )
{
    printf( "%s\n", msg );
}


void esegui_funzione( puntatore_a_funzione_tipo p_funzione_variabile )
{
    p_funzione_variabile("ciao");
}


int main()
{
    esegui_funzione( funzione );


    return 0;
}
```

In questo codice vediamo che la funzione "esegui_funzione" accetta come argomento un function pointer per poi richiamarlo .

Tutto questo avviene perchè, a livello del processore, la chiamata ad una funzione altro non è che un jump (letteralmente "salto") all indirizzo della funzione stessa il quale, essendo un semplice numero, può essere trattato come una variabile .

Nonostante questi piccoli pezzi di codice sono limitati, le potenzialità dei puntatori a funzione sono infinite, vi basti pensare
che tutta la gestione dei driver di qualunque sistema operativo si basa proprio sulla definizione di puntatori a funzione generici che vengono poi impostati in base alle specifiche della periferica e del rispettivo driver .

Spero che questo piccolo articolo sia stato interessante per qualcuno, come al solito vi auguro buono studio e vi saluto ...

ciauz !

evilsocket