Lezione 33 C e assembly 11 + Memoria dinamica 2

lunedì 4 marzo 2024 11:54

Attenzione quando passo il puntatore a free : il puntatore contiene indirizzo . **Quindi attenzione ai possibili bugs** : nonostante l'area di memoria logicamente invalidata ,se non uso il free sul puntatore, quella stessa area di memoria potrebbe essere riusata. Andiamo ora a vedere tutti i possibili casi si bugs : undefined behaviour : lo evito annullando il puntatore:

```
int *ptr = malloc(sizeof(int));
free(ptr);
*ptr = 0; /* undefined behavior */
printf("%p", ptr); /* undefined behavior */
```

Oppure lo evito usando lo scope delle variabili : alloca una nuova variabile in una diversa finestra di stack (diversa locazione di memoria), quindi non sovrascrivo il valore del puntatore!!
Ma attenzione al tipo di puntatore restituito !! (**freeing wrong pointers**): puntatore è di tipo sbagliato (puntatore restituito da malloc) -> metadati corrotti (bitmap).

```
char *msg = "Una stringa globale";
int tbl[100];
free(msg); /* undefined behavior */
free(tbl); /* undefined behavior */
```

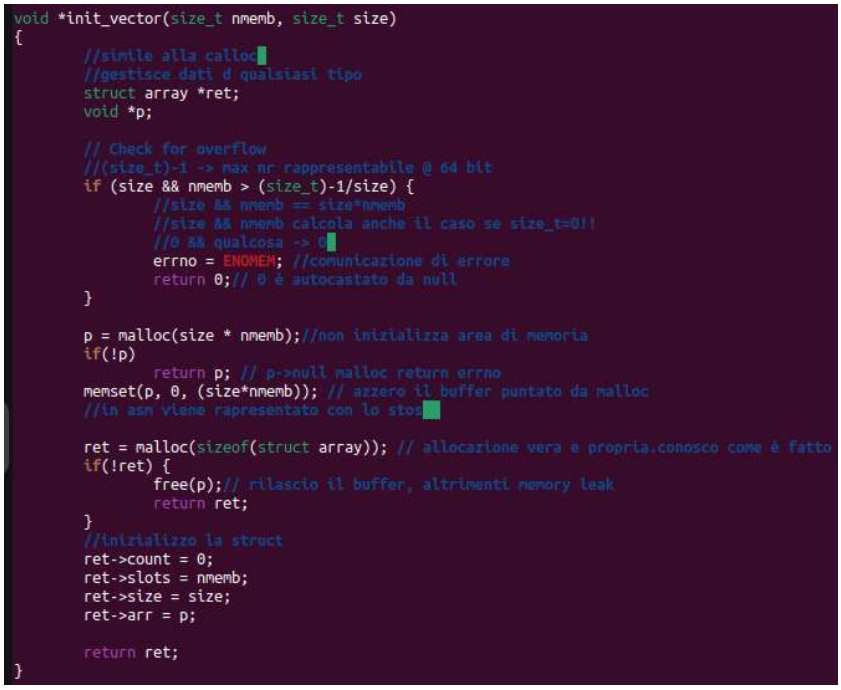
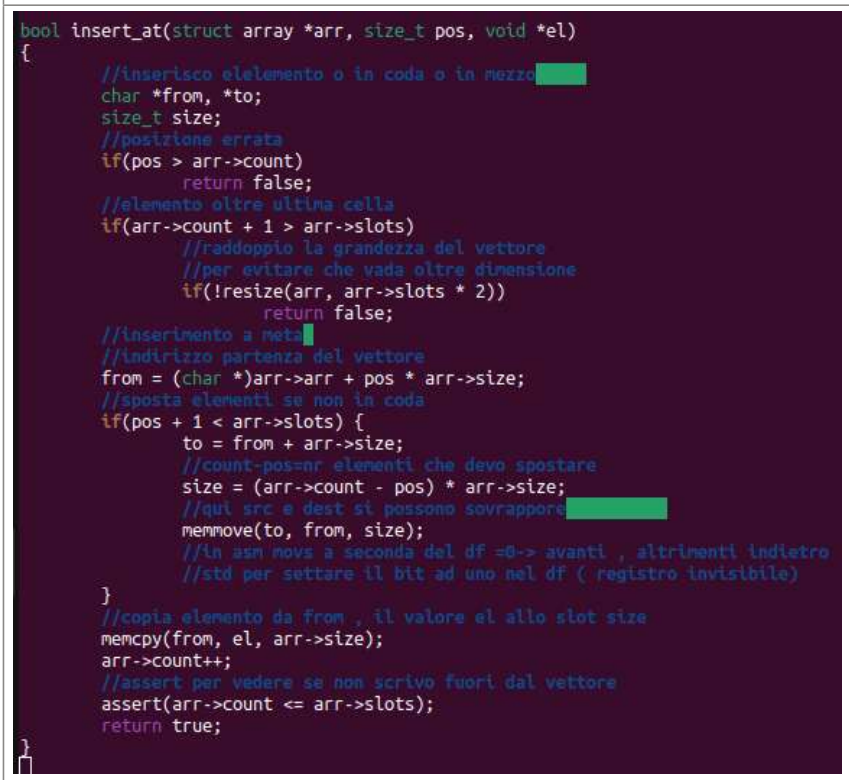
Vediamo ora la **corruzione da doppio free**(double free corruption) : dato che il free libera solo un buffer di memoria alla volta , se lo faccio due volte sullo stesso puntatore si ha un undefined behaviour :

```
int *ptr = malloc(sizeof(int));
free(ptr);
free(ptr);
```

Andiamo ora a vedere come implementare un vettore dinamico , arrivando cosi' ad esempio di matrice dinamica : andiamo a vederlo in dettaglio : il file vector.c implementa il nostro array dinamico, mentre il main lo esegue ed infine il file vector.h fa da ponte tra tutti questi file .
Andiamo a vedere il codice :

`#pragma once`
`#include <stdlib.h>`
`#include <stdbool.h>`
 	
`//pragma once compilato solo una volta`
 	
`struct array; //dichiarazione del tipo di dato incompleto`
 	
`extern void *init_vector(size_t nmemb, size_t size);//allocazione del vettore dinamico`
`extern bool insert_at(struct array *arr, size_t pos, void *el);//non ha contesto dei dati`
`extern bool push_back(struct array *arr, void *el);//insert in coda`
`extern bool read_from(struct array *arr, size_t pos, void *buffer);//estrae elemento e lo copia nel buffer`
`extern bool extract_from(struct array *arr, size_t pos, void *buffer);//idem a sopra ed in più lo rimuove`
`extern void free_vector(struct array *arr);//rilascio la memoria dinamica allocata precedentemente`
	File vector.h

`#include <stdio.h>`
`#include <stdlib.h>`
`#include <errno.h>`
`#include <string.h>`
`#include <stdbool.h>`
`#include <assert.h>`
 	
`#include "vector.h"`
`//header incluso per implementazione compatibile`
 	
`//specifico la nostra struttura "astratta"`
`struct array {`
`size_t count; // currently used slots`
`size_t slots; // total slots`
`size_t size; // size of a slot`
`void *arr; // the actual array`
`};`
	Vector.c headers

	Inizializzazione del vettore
	Insert in una poszione passata
	Estrae elemento e "aggiusta"

```c
//collegamento solamento interno
//inline aumento la leggibilità. Non si ha call a questa funzione
//il codice di questa funzione viene messa a posto della call
static inline bool resize(struct array *arr, size_t new_size)
{
        assert(arr->count < new_size);
        //alloca nuovo buffer, copia il contenuto ed aggiorna l'indirizzo
        //chiama internamente la malloc
        arr->arr = realloc(arr->arr, new_size * arr->size);
        if(!arr->arr)
                return false;
        arr->slots = new_size;
        return true;
}
```

Funzione che "mette bene"

```c
void free_vector(struct array *arr)
{
        free(arr->arr);
        free(arr);
}

bool push_back(struct array *arr, void *el)
{
        return insert_at(arr, arr->count, el);
}
```

Libera memoria

Insert in posizione data

Vediamo ora il main :

```c
#include <stdbool.h>
#include <string.h>
#include <assert.h>
#include <stdio.h>
#include "vector.h"

#define MAX_STRING 64

typedef struct person {
        char name[MAX_STRING];
        char surname[MAX_STRING];
        unsigned int age;
        unsigned char employed: 1;
        unsigned char married: 1;
        unsigned char has_children: 1;
        unsigned char student: 1;
} person_t;

struct array *people;
```

Header del main

```c
static void init_guy(person_t *guy, char *name, char *surname, unsigned int age, bool employed, bool married, bool has_children, bool student)
{
        strncpy(guy->name, name, 64);
        strncpy(guy->surname, surname, 64);
        guy->age = age;
        guy->employed = employed;
        guy->married = married;
        guy->has_children = has_children;
        guy->student = student;
}

static void print_guy(person_t *guy)
{
        assert(guy != NULL);
        printf("%s %s\n", guy->name, guy->surname);
        printf("\tage: %u\n", guy->age);
        printf("\temployed: %s\n", guy->employed ? "yes" : "no");
        printf("\tmarried: %s\n", guy->married ? "yes" : "no");
        printf("\thas children: %s\n", guy->has_children ? "yes" : "no");
        printf("\tstudent: %s\n", guy->student ? "yes" : "no");
}
```

Iniziallizzazione stampa

```c
int main(void)
{
        //inizializzo la stessa variabile molte volte
        int i;
        person_t guy;

        people = init_vector(2, sizeof(person_t));

        init_guy(&guy, "Mario", "Rossi", 48, false, true, true, false);
        push_back(people, &guy);
        init_guy(&guy, "Luigi", "Esposito", 12, false, false, false, true);
        push_back(people, &guy);
        init_guy(&guy, "Maria", "Bianchi", 34, true, true, true, false);
        insert_at(people, 0, &guy);
        init_guy(&guy, "Giuseppina", "Romano", 27, false, true, true, false);
        insert_at(people, 1, &guy);
        init_guy(&guy, "Antonio", "Ricci", 76, true, true, false, false);
        push_back(people, &guy);

        for(i = 0; i < 5; i++) {
                if(!extract_from(people, 0, &guy)) {
                        printf("Error retrieving person\n");
                        continue;
                }
                print_guy(&guy);
        }

        printf("Have records: %s\n", read_from(people, 0, &guy) ? "yes": "no");

        free_vector(people);

        return 0;
}
```

Main ed esecuione con vari esempi

```
Maria Bianchi
        age: 34
        employed: yes
        married: yes
        has children: yes
        student: no
Giuseppina Romano
        age: 27
        employed: no
        married: yes
        has children: yes
        student: no
Mario Rossi
        age: 48
        employed: no
        married: yes
        has children: yes
        student: no
Luigi Esposito
        age: 12
        employed: no
        married: no
        has children: no
        student: yes
Antonio Ricci
        age: 76
        employed: yes
        married: yes
        has children: no
        student: no
Have records: no
```

Stampa a schermo