Tutoraggio di Sistemi Operativi

Matteo Federico

Lezione #2

TOR VERGATA

UNIVERSITÀ DEGLI STUDI DI ROMA

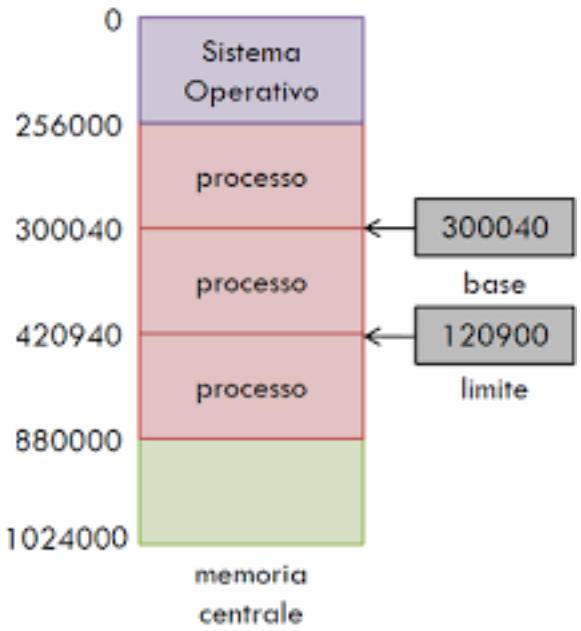
Roadmap

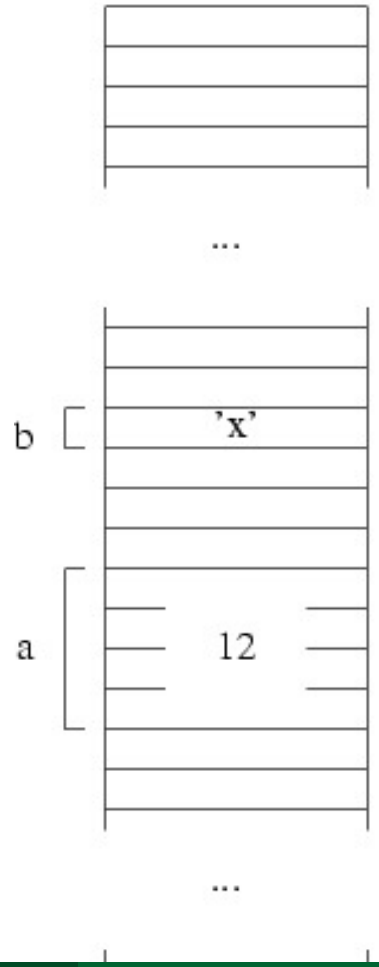
1. Puntatori in C
2. Aritmetica dei puntatori
3. Layout della memoria di un programma
4. Buffer Overflow
5. Challenge

Come funziona la memoria (idea generale)

memoria

Possiamo astrarre la memoria come un vettore dove ogni byte viene referenziato tramite un indice, questo indice viene comunemente detto indirizzo di memoria.
Il processore deve usare questi indirizzi per poter leggere e scrivere dati dalla memoria.





Cos'è un puntatore

Puntatori

- Un **puntatore** è una variabile che contiene un indirizzo di memoria.
- Un `int`, `double`, `char` sono semplicemente modi differenti di indicare sequenze contigue di byte e permettere di asserire quel'area di memoria come può venire usata.
- I **puntatori** sono variabili che permettono di indicare queste aree di memoria.

```
1 int x = 10;
2 int *p = &x;
```

Operazioni

- & $\rightarrow$ Operazione che permette di recuperare l'indirizzo di un area di memoria.
- * $\rightarrow$ Permette di accedere al valore dell'area puntata da un puntatore, viene anche usato per dichiarare una variabile puntatore.

TOR VERGATA

Dereferenziazione

```c
int x = 10;
int *p1 = &x;
int **p2 = &p1;
printf("%d %p",x,&x);
printf("%d %p %p",*p1,p1,&p1);
printf("%d %p %p %p",**p2,*p2,p2)
    ;
```

- p1 contiene l'indirizzo di x
- p2 contiene l'indirizzo di p
- Sia x che p1 che p2 possono accedere al valore di x
- Ognuno ha un numero diverso di operazioni di deferenziazione per accederci.

TOR VERGATA

Puntatori e array

```
1 int v[4] = {10,20,30,40};
2 int *p = v;
3 printf("%d",v[2]);
4 printf("%d", *(p+2));
5 printf("%d", *(v+2));
6 printf("%d", p[2]);
```

- Gli array occupano memoria contigua.
- Tramite l'aritmetica dei puntatori possiamo spostarci tra un elemento e l'altro dell'array.
- l'opearzione v[val] è eguale a dire *(v+val)

v quindi è un puntatore?

TOR VERGATA

Domanda: v quindi è un puntatore?

```
1 int v[4] = {1,2,3,4};
2 int * p= v;
3 printf("%d %ld\n",v[0],sizeof(v));
4 printf("%d %ld\n",*p,sizeof(p));
```

Risposta: In generale Si, ma dipende

Aritmetica dei puntatori

Regola fondamentale:

$$p + n = p + n \times sizeof\,(tipo)$$

Esempio:

- char $\rightarrow$ 1 byte
- int $\rightarrow$ 4 byte
- double $\rightarrow$ 8 byte

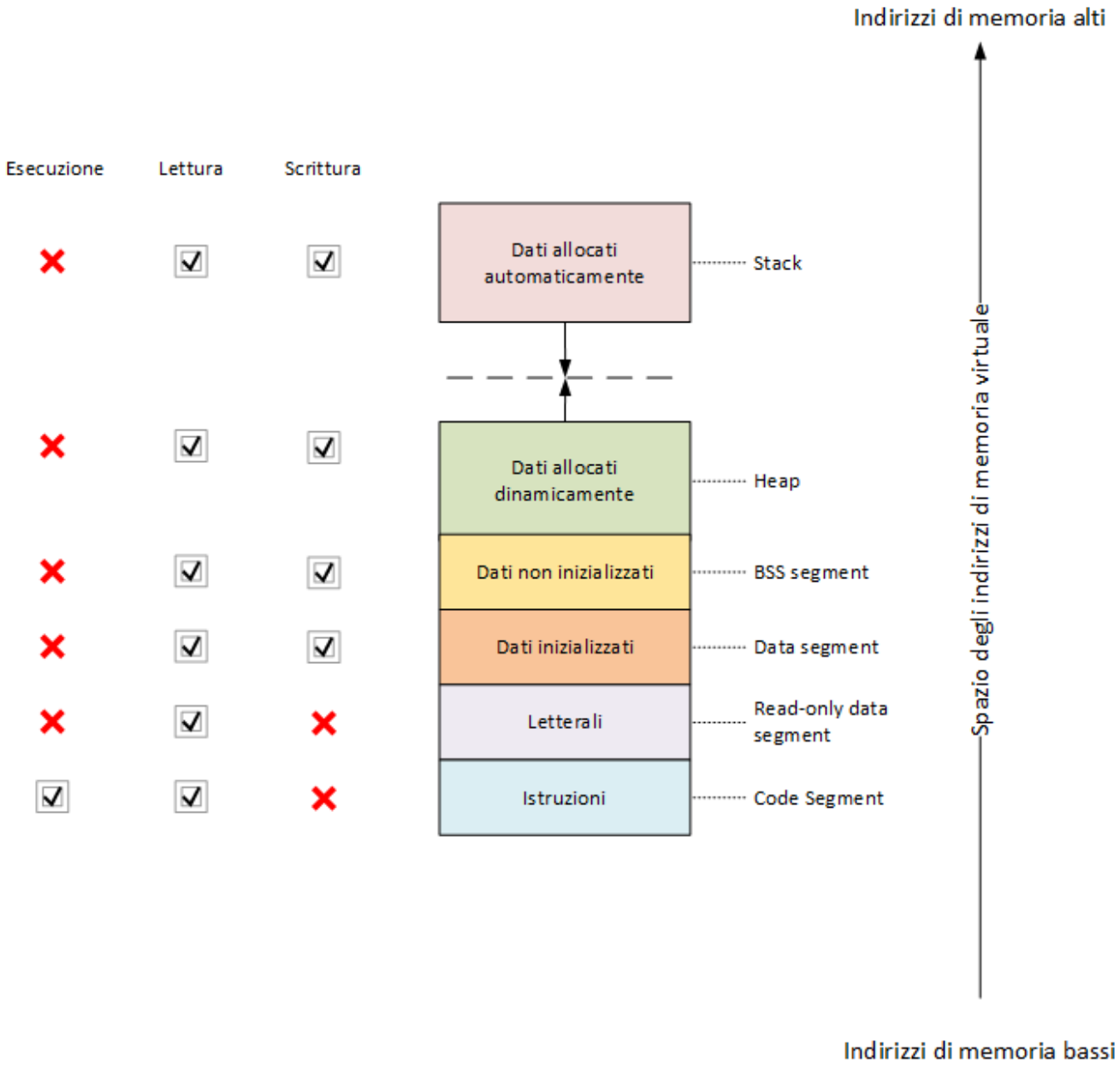
Esempio di memoria

Array:

Indirizzo	Valore
1000	10
1004	20
1008	30
1012	40

- p = 1000
- p+1 = 1004

Memoria di un programma



- Ogni sezione ha uno scopo diverso
- Il sistema operativo carica queste sezioni quando il programma viene eseguito
- Gli indirizzi effettivi dipendono da architettura e sistema operativo

Frame Title

Stack

variabili locali, return address, parametri
cresce verso indirizzi più bassi

Heap

variabili locali, return address, parametri
cresce verso indirizzi più bassi

BSS

variabili globali non inizializzate

Data

variabili globali inizializzate
può essere diviso in due sezioni con le variabili comuni e le const ovvero variabili in sola lettura

Text

istruzioni del programma

Esempio

```
1 int global;
2
3 int main() {
4     int x;
5     int *p = malloc(10);
6 }
```

- x → stack
- p → stack
- malloc → heap
- global → area globale

Vediamo l'esempio

Challenge

Scrivere un programma C che mantiene in memoria dati inseriti dall'utente, il programma dovrà mantenere questi dati in un unico array di char di grandezza 30.
Il programma dovrà supportare l'inserimento e il salvataggio dei seguenti tipi di dato: char, int, long, double.
L'utente deve poter inserire questi dati in qualsiasi ordine e quantità che desidera, ogni dato inserito dovrà essere salvato all'interno dell'array immediatamente dopo il precedente.
Quando l'utente inserisce un dato che che non entra nello spazio rimanente sull'array, il programma deve uscire e stampare tutti i byte nell'array in esadecimale.

Side Quest

Side Quest 1

Stampare i dati nel formato inserito!

Side Quest 2

Non usare funzione “memcpy” o simili

Side Quest 3

Invece di terminare il programma quando arriviamo alla fine dell’array, mettere i byte extra all’inizio dello stesso e continuare a salvare da lì

Potete inviare le vostre soluzioni a
matteo.federico@uniroma2.it.
Scrivete nell oggetto della mail [SO] challenge 1.
Vedremo le più interessanti (in positivo e negativo) la prossima lezione.
(Sarò anonimo nel mostrare le soluzioni)

TOR VERGATA