Sistemi Operativi - Tutoraggi

Laurea in Ingegneria Informatica
Università Tor Vergata
Tutor: Romolo Marotta
Docente del corso: Francesco Quaglia

Introduzione

1. Variabili puntatore in C
2. printf, scanf
3. gcc
4. Layout programma

I puntatori in C

- Il linguaggio di programmazione C permette al programmatore di accedere esplicitamente allo spazio di indirizzamento di un processo
- Alcuni operatori:
  - * dato un puntatore accede al contenuto della variabile referenziata
  - & data una variabile ne ottiene il puntatore

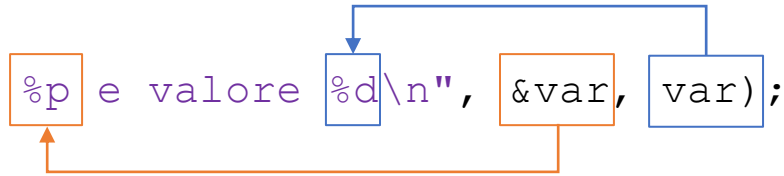

```
printf("%d\n", var);
printf("%p\n",&var);
```

```
printf("%d\n", *var_ptr);
printf("%p\n", &var_ptr);
```

printf

- int printf (const char * format, ...);
- stampa su standard output la stringa *format*
- può prendere argomenti addizionali che saranno stampati secondo la formattazione specificata dalla stringa *format*
- alcuni specificatori:
  - %c : un carattere
  - %s : sequenza di caratteri terminanti con '\0' (Stringa)
  - %p : un puntatore
  - %d : signed int decimale
  - %u : unsigned int decimale
  - %x : esadecimale unsigned
  - %o : ottale unsigned

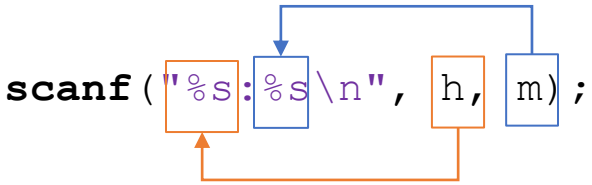
```
printf("La variabile 'var' ha indirizzo %p e valore %d\n", &var, var);
```



Esempio 1

scanf

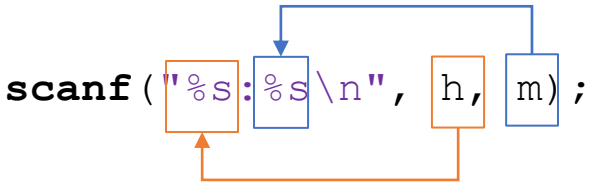
- int scanf(const char * format, ...);
- scansiona l'input in accordo alla stringa *format*
- la stringa *format* nel può prendere argomenti addizionali che saranno stampati secondo la formattazione specificata dalla stringa *format*



Esempio 2

scanf

- int scanf(const char * format, ...);
- scansiona l'input in accordo alla stringa *format*
- la stringa *format* nel può prendere argomenti addizionali che saranno stampati secondo la formattazione specificata dalla stringa *format*

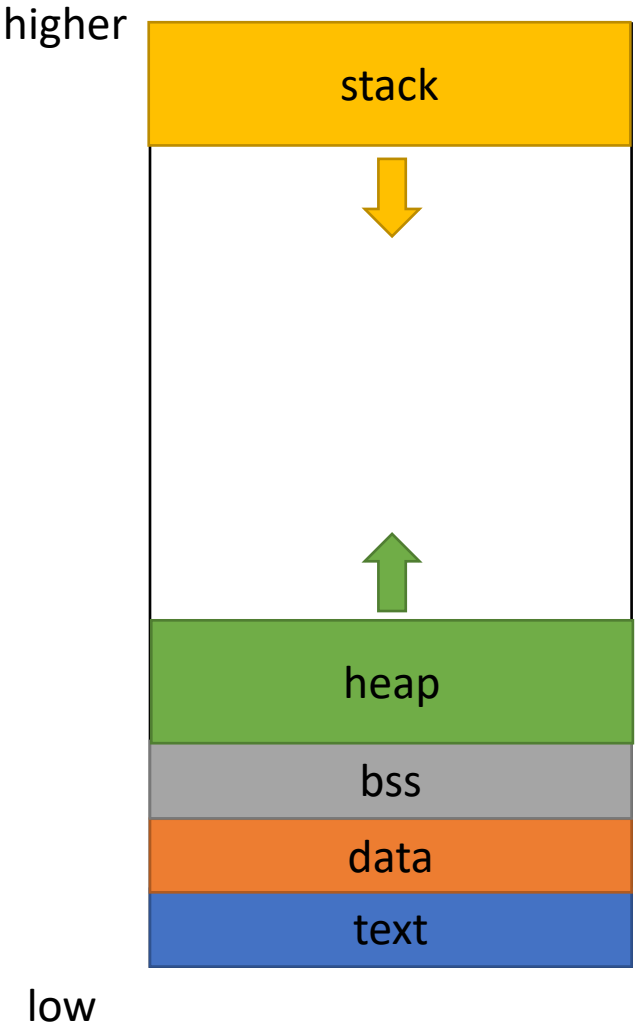


- il prototipo di uno specificatore di formato è:

```
%[*][width][length]specifier
```

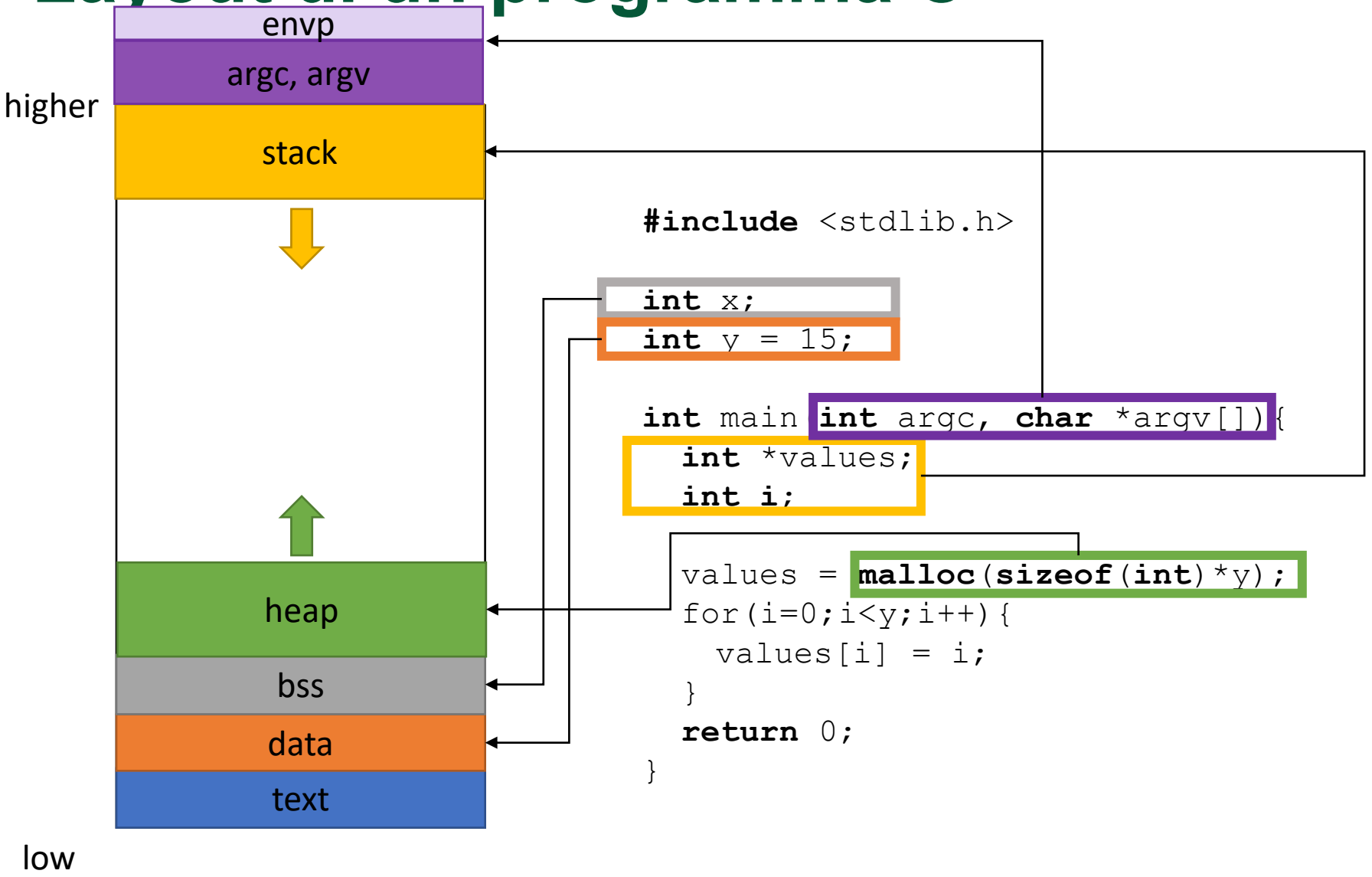
Esempio 3

Layout di un programma C



- **Text:** istruzioni eseguibili
- **Data:** dati inizializzati
- **Block started by symbol (BSS):** dati non inizializzati o initizializzati al valore zero
- **Heap:** sezione di dati allocati dinamicamente
- **Stack:** per chiamate a procedura, passaggio parametri, indirizzo di ritorno, variabili locali

Layout di un programma C



Esercizio 1

- Scrivere un programma che prende una stringa da tastiera e la inserisce all'interno di un buffer allocato dinamicamente nella heap da parte della funzione scanf().
- Copiare poi tale stringa all'interno di un secondo buffer allocato sullo stack della taglia necessaria a contenerla.
- Liberare quindi il buffer allocato nella heap utilizzando la funzione free().
- Stampare sullo schermo la stringa copiata nel buffer allocato sullo stack.

Esercizio 2

- Scrivere un programma che prende una stringa passata come primo argomento (i.e. char *argv[]) al programma stesso quando questo viene eseguito.
- Copiare tale stringa all'interno di un buffer di dimensione fissa facendo attenzione a non superare il limite imposto dalla taglia, e stamparla quindi sullo schermo.
- Rigirare la stringa (primo carattere in ultima posizione, secondo carattere in penultima posizione, ecc.) senza fare utilizzo di un ulteriore buffer per poi stampare anche questa stringa sullo schermo.