

Sistemi Operativi

Laurea in Ingegneria Informatica

Università Roma Tre

Docente: Romolo Marotta

Richiami di programmazione C

1. Compilazione
2. Puntatori
3. printf
4. scanf
5. Layout

Dal sorgente all'eseguibile

PREPROCESSORE: tutte le direttive vengono sostituite con il contenuto del file. Ad esempio: la direttiva `#include<stdio.h>s`

`a.out`: per compilare in file `c`

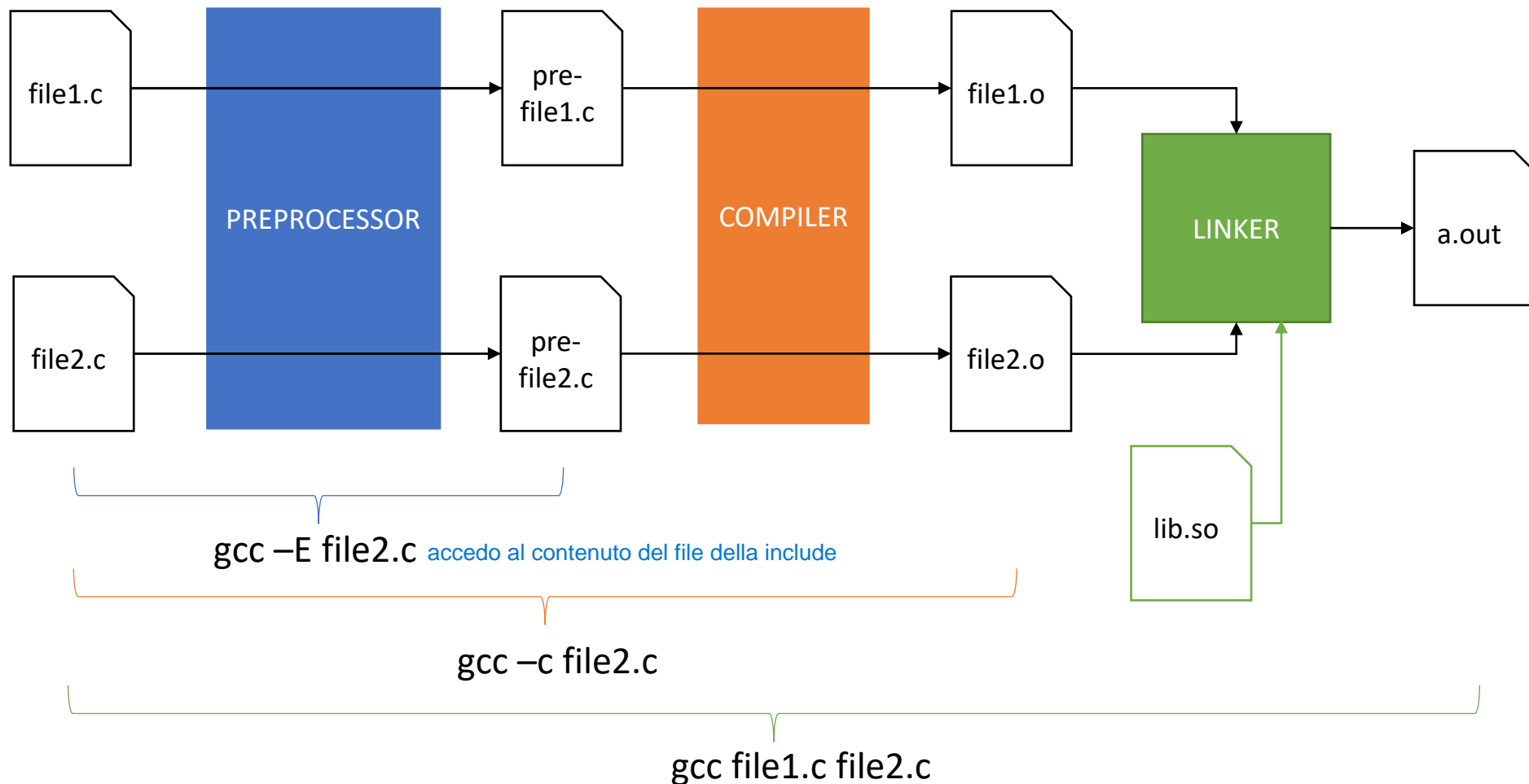
`gcc -DITERATIONS=10 simple.c`: con questo comando stiamo definendo una variabile

i file oggetti formano l'eseguibile

`!.:esegue l'ultima riga`

`-o` : ci dice qual'è l'output

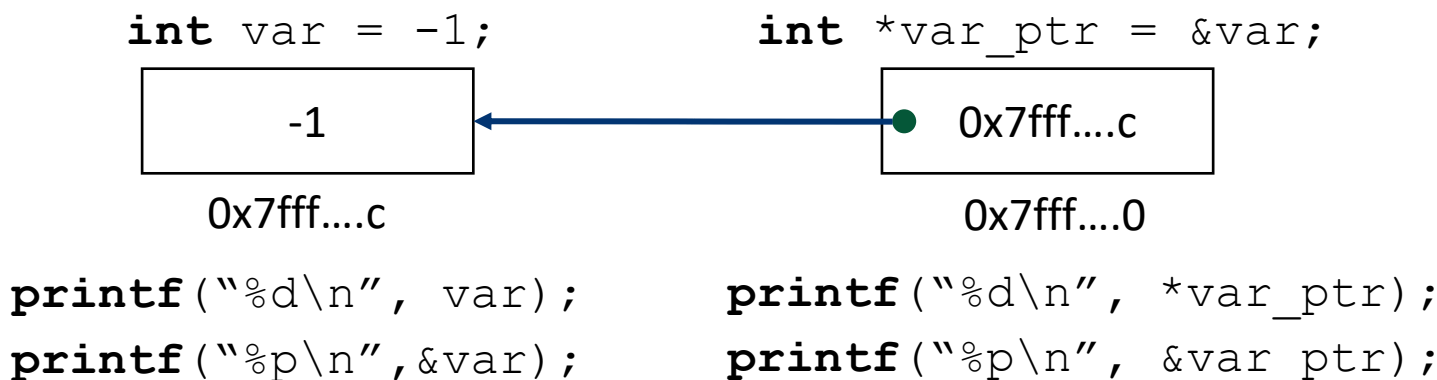
COMPILER: compila ogni file e genera ogni file oggetto



Esempi 1&2

I puntatori

- 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

- `int printf (const char * format, ...);`
- stampa su standard output la stringa *format*
- può prendere argomenti aggizionali 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);
```

The diagram illustrates the argument passing in the `printf` function call. It shows the format string `"La variabile 'var' ha indirizzo %p e valore %d\n"` followed by two arguments: `&var` and `var`. An orange box highlights the `%p` specifier, and a blue box highlights the `%d` specifier. An orange arrow points from the `&var` argument to the `%p` specifier. A blue arrow points from the `var` argument to the `%d` specifier. A blue line connects the two boxes above the comma, indicating the sequence of arguments.

Esempio 3

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*

The diagram illustrates the mapping of format specifiers in the `scanf` function to variables. The code `scanf ("%s %* \n", h, m);` is shown. The format string is enclosed in a box, and its components are further detailed: `%s` is in an orange box, `%* \n` is in a blue box, `h` is in an orange box, and `m` is in a blue box. An orange arrow points from the `h` variable to the `%s` specifier, and a blue arrow points from the `m` variable to the `%* \n` specifier.

```
scanf ( "%s %* \n", h, m );
```

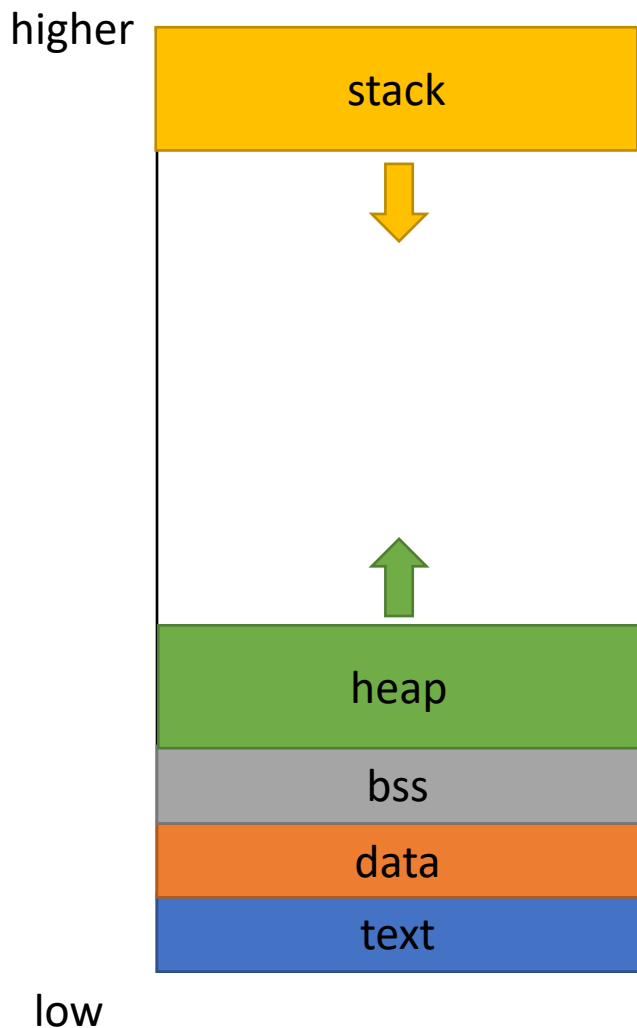
- il prototipo di uno specificatore di formato è:

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

`%*` :quel specificatore non deve essere memorizzato in nessuno variabile

Esempio 4

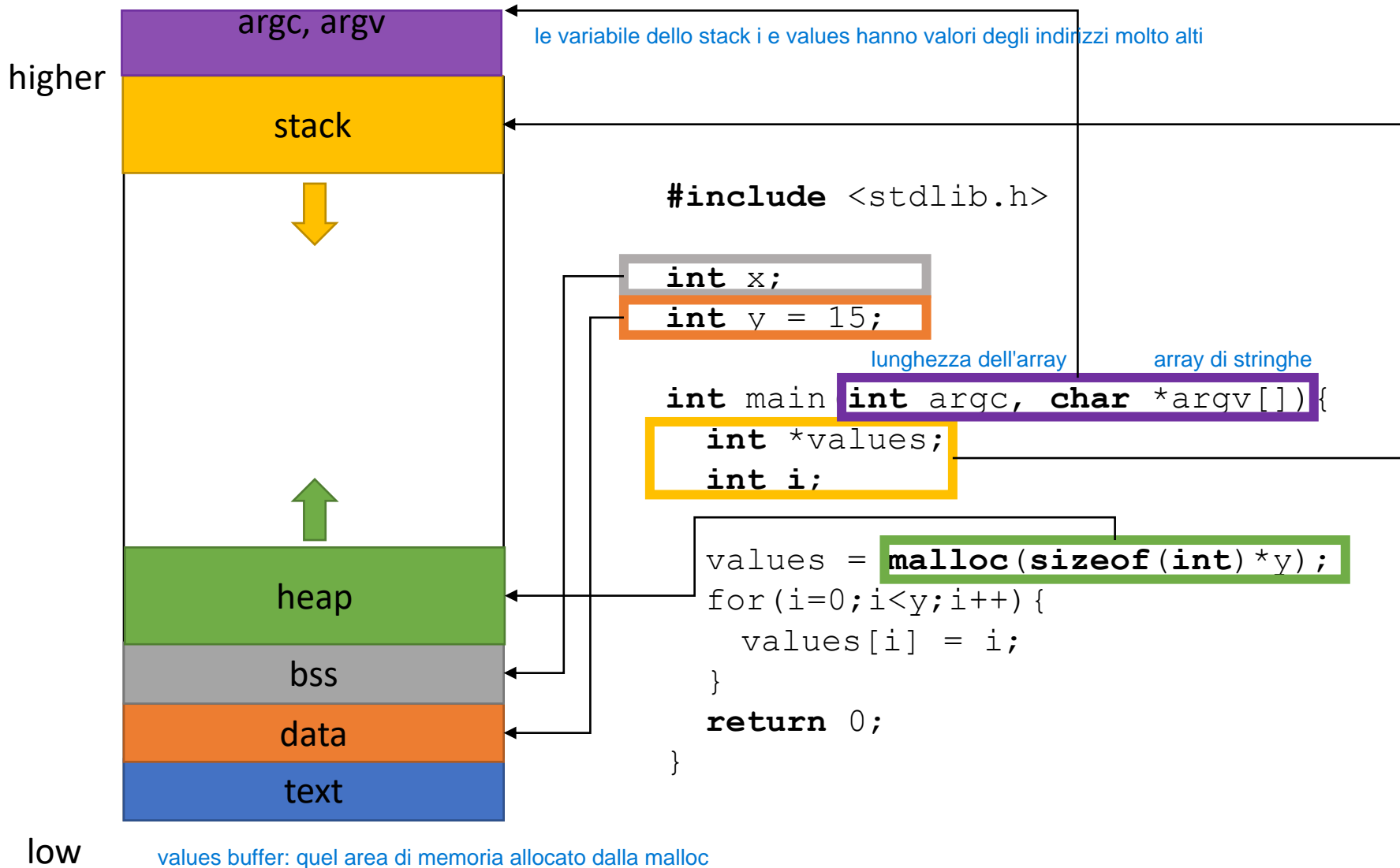
Layout di un programma C



- **Text:** istruzioni eseguibili
- **Data:** dati inizializzati
- **Block started by symbol (BSS):** dati non inizializzati o inizializzati 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

low e higher ci l'ordine degli indirizzi in ordine crescentes



Esempio 5