




Lezione 4

stringhe, argc/argv



Una stringa è un array di caratteri, in questo caso speciale non si memorizza la lunghezza ma si usa la convenzione che la stringa termina quando si incontra il byte 0. Esempio:

```
char *s = "the answer is 42";
```

s =

?

Una stringa è un array di caratteri, in questo caso speciale non si memorizza la lunghezza ma si usa la convenzione che la stringa termina quando si incontra il byte 0. Esempio:

```
char *s = "the answer is 42";
```

s =

10000

10000				10005					10010						10015	
t	h	e	_	a	n	s	w	e	r	_	i	s	_	'4'	'2'	\0

fa capire a **printf**
puts etc. che la
stringa finisce qui



Una stringa è un array di caratteri, in questo caso speciale non si memorizza la lunghezza ma si usa la convenzione che la stringa termina quando si incontra il byte 0. Esempio:

```
char *s = "the answer is 42";
```

s =

10000

10000				10005				10010				10015				
t	h	e	_	a	n	s	w	e	r	_	i	s	_	'4'	'2'	\0

\0 rappresenta
il carattere il cui
codice ASCII è
il byte 0

116	104	101	32	97	32	52	50	0
-----	-----	-----	----	----	-----	-----	-----	----	----	----	---

Bytes in RAM a partire dalla posizione 10000 (codici ASCII)

I parametri della funzione `main()`

- Il prototipo della funzione `main` è

```
int main(int argc, char *argv[])
```

- Questi parametri servono per passare al programma quello che scriviamo sulla riga di comando insieme al nome dell'eseguibile
- Esempio: supponiamo di scrivere sulla riga di comando:

```
a.out ciao 52
```

(nota: **a.out** è il nome del file eseguibile creato da **gcc** se non usiamo l'opzione **-o** per scegliere il nome)

Una possibile fotografia della memoria all'inizio del programma:

10000	10004	10008	10012	10016	10020	10024
3	11000					

argc

argv

11000	11008	11016
11040	11046	11051

argv[0]

argv[1]

argv[2]

11040					11046					11051				
a	.	o	u	t	\0	c	i	a	o	\0	'5'	'2'	\0	