Uso di `scanset`

mobytrick

5 settembre 2014

Nel linguaggio C *scanset* è una particolare notazione del formato di lettura delle stringhe oppure dei singoli caratteri che permettere di accettare solo caratteri conformi ad un modello mutuando alcune forme sintattiche delle *Regular expressions*.

Supponiamo di voler leggere 1 carattere; tale carattere deve inoltre essere una vocale dell'alfabeto italiano. Senza ricorrere a scanset la sequenza delle fasi è la seguente:

1. lettura di un carattere con il formato `%c`
2. controllo se è una vocale

Codificando nel linguaggio C le due fasi abbiamo il seguente spezzone:

```
#include <string.h>
...
char *vocale = "aeiou", carattere;
int risultato;
...
(void) printf ("immetti una vocale ");
risultato = scanf ("%c", &carattere);
if (risultato == 0 || strchr (vocale, carattere) == NULL))
  (void) fprintf (stderr, "dovevi inserire una vocale\n");
else
  (void) printf ("vocale immessa: %c\n", carattere);
```

È possibile fondere le due fasi in un'unica operazione usando appunto scanset. L'originario formato viene cambiato in `%[aeiou]`. Ovvero la specifica `c` (per caratteri) viene sostituita con i caratteri ammessi. Il carattere letto viene memorizzato solo se è conforme al modello e contemporaneamente viene aumentato di 1 il numero delle variabili inizializzate (comportamento standard di scanf). Passando alla codifica in linguaggio C:

```
char carattere;
int risultato;
...
(void) printf ("immetti una vocale ");
risultato = scanf ("%1[aeiou]", &carattere);
if (risultato == 0)
  (void) fprintf (stderr, "dovevi inserire una vocale\n");
else
  (void) printf ("vocale immessa: %c\n", carattere);
```

Un altro esempio, per fissare le idee. Il formato `%3[02468]` indica che si intende leggere 3 caratteri numerici, tutti e 3 pari. Attenzione, però. I caratteri numerici vengono collocati in una stringa. Poi si tratta di convertirli in formato numerico. Per inciso, la sintassi `%[02468]{3}` che fa uso del meccanismo di ripetizione proprio delle *Regular Expression* non è ammessa.
Un ulteriore esempio. [a-z] rende valide le lettere dell'alfabeto internazionale purché minuscole, mentre `[^A-Z]` rende invalide quelle maiuscole perché l'accento circonflesso ha la caratteristica di negare l'intervallo.

Alla luce di tutto ciò un uso abbastanza utile di scanset permette la lettura di stringhe che contengono il carattere space. Come è noto, il formato `%s` del linguaggio C interrompe la lettura all'incontro di un carattere spazio. Basta sostituire al formato `%s` l'espressione `%[^\n]`. L'accento circonflesso (`^`) nega il carattere newline. In altre parole vengono accettati tutti i caratteri, spazio compreso, sino all'incontro di `\n` (generato dall'uso del carattere Enter/Invio della tastiera) che segnala la fine della lettura. Attenzione a rimanere entro i limiti. Ovvero, si tratta non eccedere il dimensionamento della stringa che memorizza i caratteri letti. Si può autocostruirsi il formato di lettura col numero esatto di caratteri desiderati e poi usarlo, così come indicato nel sottostante programma.

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>

int main (void)
{
  char fmt [10], stringa [11];
  int  len;

  len = sizeof (stringa) - 1;
  bzero (fmt, sizeof (fmt));
```

```
  bzero (stringa, sizeof (stringa));
  (void) printf ("immettere max. %d caratteri ", len);
    /* costruzione formato di lettura */
  (void) sprintf (fmt, "%%%d[^\n]", len);
    /* lettura tramite scanset */
  (void) scanf (fmt, stringa);
  (void) printf ("stringa immessa: %s\n", stringa);
  exit (EXIT_SUCCESS);
}
```