Alcuni caratteri (*in particolare i backslash-escaped*) non sono normalmente stampabili tramite il comando `echo`; si può ovviare a questo problema tramite il quoting `$'str'`, il quale:

- Espande la stringa `str` in una stringa *single-quoted*, ossia una stringa quotata tra singoli apici (*in pratica rimuovendo il* `$`);
- Se presenti, converte i *backslash-escaped characters* nel loro corrispettivo valore *ANSI C*.

La variabile d'ambiente **`IFS`** contiene i caratteri che **fungono da separatori di parole negli elenchi**, ossia quei caratteri che delimitano dove finisce una stringa e ne inizia un'altra.
Di default, `IFS=' \t\n'` - ossia i delimitatori sono **spazi bianchi, andate a capo e tabulazioni**. Ciò rende *impossibile* operare, ad esempio con file che hanno spazi bianchi nel loro nome; per fare ciò, si dovrebbe modificare `IFS` temporaneamente, rimuovendo lo spazio bianco al suo interno, effettuare l'operazione voluta ed infine ripristinare `IFS` originale.
`IFS` non determina i **separatori di parole chiave di `bash`**, ma solo dei separatori **in liste di stringhe/nomi**.

Lettura da stdin con `read`

Il comando `read` permette la lettura di una stringa dallo *standard input*, la quale verrà letta solo una volta premuto il pulsante invio.
Se lo standard input è stato rediretto da un file (vedremo poi come), verrà letta **una sola riga del file**; le successive andranno lette da altri `read`.

La `read` ha un *exit value* sulla base della corretta avvenuta della lettura: 0 se avvenuta correttamente, >0 se si è arrivati a fine file.

Può avvenire che, durante la lettura di un file, l'ultima riga non possiede il carattere di andata a capo `\n`. In tal caso la read pone nella variabile passata tutti i caratteri posti prima della fine del file e restituirà un *exit value* >0 per segnalare la fine file.
Ciò è importante, perché significa che la variabile passata a `read` potrebbe risultare vuota - l'*exit code* non riporta nulla su questa informazione in quanto si limita a controllare se si è arrivati a fine file.

L'espansione `${#VAR}` viene interpretata dalla `bash` sostituendola con **il numero di caratteri che formano il valore della variabile `VAR`**.
È necessario che, quando `read` effettua una lettura, venga controllato il numero di caratteri all'interno della variabile - se è 0, non è stato letto nulla; ciò è possibile tramite vari metodi.

È possibile passare più variabili a `read`: in questo caso verranno lette una serie di parole sulla base del contenuto della variabile `IFS`, assegnando alla prima la prima parola, alla seconda la seconda e così via; l'ultima variabile riceverà **TUTTE** le parole rimanenti.
Come per la manipolazione di stringhe, è necessario fare attenzione a ciò che è contenuto all'interno di `IFS`.

`read` può ammettere anche alcuni parametri:

- `-n` e `-N` seguiti da un numero permettono alla read di salvare all'interno della variabile solo un certo numero di caratteri specificato dopo il parametro.
  - `-n` considera `\n` (le andate a capo) come carattere terminatore, terminando la lettura dei caratteri prematuramente
  - `-N` invece non considera le andate a capo, procedendo a oltranza anche nelle righe successive, se necessario.
- `-u` serve a specificare da quale stream di dati va effettuata la lettura. Spesso si indica **la variabile file descriptor di uno specifico file**.