



### Cicli, Condizioni e Alias

Start:

A cosa serve fare uno script? La risposta è semplice, serve a facilitarci la vita in chat, a trasformare il nostro client IRC nell'ambiente che vogliamo sia perfetto per noi, con le nostre comodità, i nostri pulsantini... insomma tipo la nostra camera per intenderci.

Premesso questo, andiamo subito al dunque, poche parole e molto codice, ovviamente opportunamente spiegato.

Suppongo abbiate il vostro client KVIrc lì vicino, scaricate l'ultima versione, se siete utenti linux compilate la versione cvs o prendete una snapshot già impacchettata, se siete utenti Windows scaricate l'ultima SNAPSHOT (non la stabile).

Qui troverete la snapshot più aggiornata

<ftp://ftp.kvirc.net/pub/kvirc/snapshots/>

Adesso per iniziare andiamo ad attivare la barra che ci permetterà di poter creare i nostri script.

Dal KVIrc:

*Impostazioni-->Barre degli strumenti-->Mostra scripting.*

Vedrete apparire la barra con i pulsanti che permetteranno di rendere il KVIrc ancora più accogliente e soprattutto personalizzarlo come più piace.

In questo volume parlerò degli **alias** e di come usarli.

Gli alias sono dei comandi, in poche parole possiamo aggiungere comandi nostri, personalizzati, a quelli che già esistono sul client.

Per farlo dobbiamo, per prima cosa, aprire l'editor degli alias usando il pulsantino con lo [/a] verde che appare nella barra dello scripting, e crearci un nuovo comando (o alias come preferite):

fate tasto destro sulla lista di comandi che vedrete alla vostra sinistra e createne uno, apparirà col nome "alias", nome che provvederemo a cambiare portandoci sulla riga bianca che appare sopra la finestra nera di destra e cambiando la scritta alias con, ad esempio, "ciao".

Adesso metteremo del codice dentro l'alias appena creato, il codice dell'esempio farà in modo di salutare tutti quelli che ci sono su un canale fermandosi comunque a 6 utenti (magari può sembrare inutile, ma ci serve per capire alcune cose, e poi... niente è veramente completamente inutile).

Il codice lo spiegherò molto approfonditamente quindi niente paura =).

```
# code n1
%idx=6;
foreach(%i,$chan.users){
if(%idx==0) {
say Azzo siete moltissimi =D ciaO a Tutti AlloRA!;
break;
}
say ciao %i;
%idx--;
}
# end code
```

Cominciamo subito col dire che non vi spiegherò la sintassi di tutti i comandi del KVIrc, dato che dovrete imparare a convivere con il manuale del KVIrc: infatti basta fare *Aiuto-->Browser help..* e vi apparirà il manuale che sarà il vostro fido compagno. Esaminiamo il codice sopra incollato:

#### **# code n1**

Questo è un semplice commento: i commenti in kvs (kvs = KVIrc scripting) si fanno facendo precedere la riga dal simbolo del cancelletto "#"

#### **%idx=6;**

Qui invece sto inizializzando la variabile *%idx*, in kvs le variabili si fanno precedere dal simbolo di percentuale. Inoltre è differente scrivere *%Idx* e *%idx*, poiché le variabili che cominciano per una lettera maiuscola, sono variabili che il KVIrc considererà globali, ciò comporta che non si distruggono da sole ma è necessario che l'utente le cancelli appositamente assegnandogli un valore nullo (nel nostro esempio potremmo fare *%Idx=""*), e possono essere, inoltre, viste anche da altri script del nostro client; mentre quelle che iniziano per lettera minuscola avranno la stessa durata dell'esecuzione del comando o della funzione in cui esse si trovano dopodiché si autodistruggeranno.

Per capire meglio, andiamo nella riga di input di una qualsiasi finestra e digitiamo:

```
/%prova =3
```

e dopo

```
/echo %prova
```

come vedrete il risultato è ..niente, non c'è risultato, poiché la vita di *%prova* è durata solo per l'esecuzione del comando */%prova =3* dopodiché è andata distrutta.

Se, invece, scriviamo:

```
/%Prova =3
```

```
/echo %Prova
```

Possiamo vedere che il risultato è 3, quindi con la lettera maiuscola la variabile ha continuato a vivere anche dopo l'esecuzione del primo comando.

Nel nostro codice inizializziamo la variabile a 6 (il numero massimo di persone che voglio salutare, per evitare di fare troppi "ciao nick" in un canale con moltissimi utenti).

```
foreach(%i,$chan.users) {  
codice  
}
```

Qui uso il comando *foreach*; la sintassi del comando potrete vederla nella guida del KVIrc mentre il *\$chan.users* è una funzione del KVIrc, che restituisce, in un array (considerate un array come una lista) tutti i nick delle persone presenti sul canale (per vedere le altre funzioni vi rimando sempre al solito help).

Per rendercene conto in modo pratico andiamo sul un canale qualsiasi e proviamo a scrivere:

```
/echo $chan.users
```

Da me il risultato è:

```
[09:31:00]
```

```
Grifisx_away,morfeux,AM1C0H4CK3R,Cif,franzi_,MysteryBETA,nonno_rik,oh  
i[gentoo],ohiahiohi,redsend,Scorp[Zzz],zerymo.
```

Quel *%i*, che troviamo nel *foreach* invece indica, secondo la sintassi del comando, un elemento (quello attuale) dell' array.

Pertanto il codice che abbiamo scritto significa, "per ciascun elemento(*foreach*) presente nell'array *\$chan.users* esegui questo codice { codice da eseguire }" ed adesso andiamo a vedere il codice che gli faremo eseguire.

```
if(%idx==0) {  
say Azzo siete moltissimi =D ciaO a Tutti AlloRA!;  
break;  
}
```

Qui facciamo un piccolo controllo che ci eviterà di salutare tutti sul canale, e limiterà il saluto a solo 6 persone.

Se (la variabile *%idx* è = a zero) esegui questo codice { codice }

*say* è un comando di KVIrc che serve per scrivere del testo in una particolare finestra.

*break* invece serve per interrompere un ciclo, in questo caso interromperà il ciclo *foreach*.

Quindi qui viene detta(*say*) la frase "siete moltissimi =D ciaO a Tutti AlloRA!" dopodiché il ciclo si interrompe.

Ma vediamo cosa succede se la condizione sopra (*if(%idx==0)*) non è vera:

```
say ciao %i;  
%idx--;
```

Questo significa: dici ciao *%i* (*%i* abbiamo detto è l'elemento attuale dell' array in cui stiamo ciclando quindi il nick attuale) e diminuisci *%idx*.

Quindi al primo "giro" *%idx* è = a 6, la condizione (*%idx=0*) non si verifica, dice "ciao nick" diminuisce *%idx*, che diventa 5 e ricicla, la condizione non si verifica, dice "ciao nick" etc etc, fino a quando *%idx* sarà = 0, allora la condizione si verificherà, e uscirà dal ciclo *foreach*.

Ok, andiamo a vedere un po' prima di cominciare a rimettere mani al nostro codicillo alcuni comandi particolari di KVIrc:

Proviamo a portarci sulla linea di input del nostro canale preferito e premere:

**-CTRL+B** ci apparirà un simbolo, tutto quello che scriveremo dopo, gli

utenti del canale lo leggeranno in grassetto (b sta per Bold);

**-CTRL+U** sottolineato;

**-CTRL+K** colorato;

gli altri comandi poi li andiamo a vedere sul manuale per adesso ci bastano questi=P.

In scripting per scrivere bold si usa `$b()`, per scrivere sottolineato `$u()`, per scrivere colorato `$k(<numero colore testo>,[ numero colore sfondo])`

Proviamo subito portandoci sulla nostra linea di input e digitando:

```
/echo $b()Ciao a tutti
```

Dopo di che andiamo a mettere mani al nostro codice e modifichiamolo come segue:

```
# code n1
%idx=6;
foreach(%i,$chan.users){
if(%idx==0) {
say Azzo siete moltissimi =D ciaO a Tutti AlloRA!;
break;
}
say Ciao $k($rand(12))%i;
%idx--;
}
# end code
```

La cosa che dobbiamo esaminare è solo il comando che qui appare in più, cioè `$k($rand(12))`, e allora qui è semplice perché `$k()` abbiamo visto che serve per indicare un colore, mentre `$rand(<intervallo valore>)` se andiamo a guardare nella nostra guida (il famoso help del kvirc) vedremo che serve per avere un numero casuale nell'intervallo che noi gli andiamo a dare tra le parentesi ad esempio provate uno:

```
/echo $rand(12)
```

vedrete che il risultato sarà un valore numerico tra 0 e 12.

A questo punto qualcuno potrebbe anche volere che non vengano scelti 2 colori uguali di seguito, perché nella casualità questo potrebbe accadere, allora dobbiamo inventarci un controllo che vada a modificare il numero di colore attuale se esso risulterà, casualmente, uguale a quello precedente.

Quindi, prima cosa dobbiamo conservarci il vecchio valore, il che significa che dobbiamo creare una variabile che lo memorizzi, dopo di che dobbiamo andare a confrontare il valore nuovo con quello vecchio e se sono uguali modificare quello nuovo (ad esempio aggiungendogli 1 così non è uguale a quello di prima).

Trasformiamo in codice:

```
Creiamo la variabile %oldColor e la inizializziamo a 0;
Creiamo la variabile %newColor e la inizializziamo con $rand(12);
Creiamo il controllo, se %oldColor == %newColor allora %newColor++;
Immagazziniamo in %oldColor il valore attuale di %newColor;
In codice reale uscirà fuori:
```

```
%idx=6;
%oldColor =0;
foreach(%i,$chan.users){
%newColor=$rand(12)
if(%idx==0) {
say Azzo siete moltissimi =D ciaO a Tutti AlloRA!;
break;
}
}
```

```

if (%newColor==%oldColor){ %newColor++;};
say Ciao $k(%newColor)%i;
%oldColor=%newColor;
%idx--;
}

```

Qui le cose da notare non sono molte, per prima cosa, ovviamente, che l'inizializzazione della variabile %oldColor è al di fuori del ciclo, altrimenti si reinizializzerebbe ogni volta, il resto credo sia abbastanza chiaro.

Però... io sono poco accontentabile quindi mi piace complicarmi la vita e voglio anche che i nick me li scriva alternando una lettera maiuscola ad una lettera minuscola =D.

Prima di andare a vedere il codice seguente, vi consiglio di aprire il manuale del KVIrc alla voce *funzioni* e poi alla lettera *s*, qui vedrete che ci sono tutte le funzioni che ci permettono di modificare le stringhe; sono quelle con \$str.xxxxxx, siccome a noi serve una trasformazione maiuscolo/minuscolo e viceversa, quelle più adatte come vedrete sono:

```

$str.upcase(<stringa da convertire>)
$str.lowercase(<stringa da convertire>)

```

Come al solito, per provare, andiamo alla linea di input e spariamo il comando:

```

/echo $str.upcase("provaaaa")

```

Dopo di che siccome non vogliamo andare a convertire tutti i nick (sarebbe stato troppo facile) ma una lettera per volta, dobbiamo andare a trovare una funzione che ci divida la stringa, funzione che io ho trovato nella

```

$str.section ( < stringa_da_dividere >,
<elemento_da_usare_come_separatore>, <posizione_da_dove_dividere>,
<posizione_fino_a_cui_dividere>).

```

Proviamo subito con l'echo

```

/echo $str.section( "Nome**Cognome**Nick**Telefono","**", 2, 2 );

```

il risultato sarà "Nick".

Inoltre mi servirò della funzione \$str.len(<stringa>), che mi restituisce la lunghezza della stringa.

ok... adesso vediamo il codice:

```

%idxN=6;
%oldColor =0;
foreach(%i,$chan.users){
%szNick=%i
%idx=1
%itmp=1
while(%idx!=($str.len(%szNick)+1))
{
    if(%itmp==1)
    {
        %sztmp =$str.upcase($str.section(%szNick,"",%idx,%idx));
        %itmp=0
    }
    else
    {
        %sztmp =$str.lowercase($str.section(%szNick,"",%idx,%idx));
        %itmp=1
    }
    %szNickColorato=%szNickColorato%sztmp
    %idx++;
}

```

```

}
%newColor=$rand(12)
if(%idxN==0) {
echo Azzo siete moltissimi =D ciaO a Tutti AlloRA!;
break;
}
if (%newColor==%oldColor){ %newColor++;};
say Ciao $k(%newColor)%szNickColorato;
%szNickColorato=""
%oldColor=%newColor;
%idxN--;
}

```

Bene, cominciamo ad esaminarlo un po', in fondo non è così complicato.

Ho cambiato `%idx` in `%idxN` solo per comodità, perché poi uso un'altra variabile come indice e siccome gli indici mi piace chiamarli `%idx`, allora per non confonderli qui ho messo la N finale.

Inizializzo le variabili necessarie e poi comincio il mio ciclo per la trasformazione.

Controlliamo prima i passaggi salienti della trasformazione e poi il motore di essa:

```
while(%idx!=$(str.len(%szNick)+1))
```

Fino a quando la variabile `%idx` è diversa dalla lunghezza del nick + 1 (fino a quando non raggiungiamo la fine del nick;

```
%sztmp=$(str.upcase($str.section(%szNick,"",%idx,%idx))
```

`%sztmp` è uguale alla lettera trasformata in Maiuscola, in pratica qui io seziono il nick(`%szNick`) in base al carattere "" (cioè a nessun carattere, significa che viene divisa lettera per lettera) a partire dall'indice attuale fino ad arrivare all'indice attuale stesso, mettiamo il caso che il nick è Grifisx

```
/echo $str.section("Grifisx","",1,1)
```

vedrete che vi darà "G".

Bene quindi fare ad esempio

```
/echo $str.lowercase($str.section("Grifisx","",1,1))
```

ci porterà come risultato "g", perché è la stessa cosa di

```
/echo $str.lowercase("G").
```

Dopo come vedete nel codice

setto `%itmp=0` questo è un modo per far sì che al prossimo giro, non entri più in questo `if` ma vada nell'`else`, `%idx` inoltre crescerà ad ogni giro quindi sarebbe come fare:

```
$str.upcase($str.section("Grifisx","",1,1))
```

```
$str.lowercase($str.section("Grifisx","",2,2))
```

```
$str.upcase($str.section("Grifisx","",3,3))
```

```
$str.lowercase($str.section("Grifisx","",4,4))
```

Poi come vedete faccio un `%szNickColorato=%szNickColorato%sztmp` questo perché `%sztmp` contiene la lettera attuale trasformata, nel caso di Grifisx mettiamo "g", quindi il ragionamento è questo:

Dato che `%szNickColorato` è vuoto all'inizio avremo :

```
%szNickColorato=%szNickColorato%sztmp --> ""=""g
```

A questo punto `%szNickColorato` sarà uguale a "g", quindi al secondo giro avremo:

```
%szNickColorato=%szNickColorato%sztmp --> g=gR (perché %sztmp conterrà la seconda lettera del mio nick in maiuscolo)
```

A questo punto `%szNickColorato` sarà uguale a "gR", quindi al terzo giro avremo:

```
%szNickColorato=%szNickColorato%sztmp --> gR=gRi
```

e così via fino a quando

```
while (%idx != ($str.len(%szNick)+1))
```

quindi fino a quando non si supera la lunghezza del mio nick di 1, (in poche parole il mio nick è finito).

Benissimo siamo a sei pagine di tutorial però, è ancora presto per dire basta, adesso vogliamo che la scritta "Azzo siete moltissimi =D ciaO a Tutti AlloRA!" sia tipo arcobaleno!

Quindi, mettiamo mano al nostro codice e inventiamoci un modo per trasformare una stringa in un arcobaleno.

```
%idxN=6;
%oldColor =0;
foreach(%i,$chan.users){
%szNick=%i
%idx=1
%itmp=1
while(%idx != ($str.len(%szNick)+1))
{
    if(%itmp==1)
    {
        %sztmp = $str.upcase($str.section(%szNick,"",%idx,%idx));
        %itmp=0
    }
    else
    {
        %sztmp = $str.lowercase($str.section(%szNick,"",%idx,%idx));
        %itmp=1
    }
    %szNickColorato=%szNickColorato%sztmp
    %idx++;
}
%newColor=$rand(12)
if(%idxN==0) {
%szFrase= "Azzo siete moltissimi =D ciaO a Tutti AlloRA!";
%idxW=1
while(%idxW != ($str.len(%szFrase)+1))
{
    %sztmp2 = $str.section(%szFrase,"",%idxW,%idxW)
    %szFraseColorata=%szFraseColorata$k($rand(15))%sztmp2
    %idxW++;
}
say %szFraseColorata;
break;
}
if (%newColor==%oldColor){ %newColor++;};
say Ciao $k(%newColor)%szNickColorato;
%szNickColorato=""
%oldColor=%newColor;
%idxN--;
}
```

Non credo ci sia molto da spiegare, perché alla fine avete già visto un po' tutti i costrutti. Quindi credo che sia meglio finire qui.

Una raccomandazione, questo scriptino che abbiamo fatto, può risultare fastidioso in molti canali, l'unico scopo per cui l'abbiamo creato è che, come vedete, ci è servito per cominciare a capire i costrutti e i ragionamenti che, più avanti, ci porteranno alla creazione di cose molto più complesse.

Prima di salutarci però voglio chiudere il discorso sugli alias,

dicendovi come si fa a passare un "argomento" ad un alias, vi faccio subito un esempio:

Mettiamo il caso che io voglia scrivere

```
/colora "ciao ragazzi"
```

e che l'alias *colora*, che ho creato in precedenza, mi debba colorare la scritta "ciao ragazzi", cioè a dire voglio dare io la scritta da colorare di volta in volta, benissimo passiamo al codice da inserire nell'alias:

```
echo $k($rand(12)) $0-
```

ecco.... semplicemente questo.

In pratica *\$0-* indica tutta la frase che noi passiamo all'alias partendo dall'inizio

invece *\$0* (senza il *-*) indica il primo elemento che passiamo, *\$1* il secondo, *\$2* il terzo e così via (ovviamente *\$1-* indica tutta la frase a partire dal secondo elemento, *\$2-* tutto a partire dal 3zo elemento e così via)

Ultimo esempio:

nel nostro alias *colora* mettiamo questo

```
echo $k($0) $1-
```

e dalla nostra riga di input digitiamo:

```
/colora 5 scrivo a colori
```

come vedrete abbiamo usato il primo valore (*\$0*) per dare il colore alla frase, e poi abbiamo usato *\$1-* per prendere il resto della frase a partire da dopo il primo elemento (*\$0* è per noi il numero del colore).

Per concludere aggiungiamo una particolarità: se volessimo creare un'alias al volo, senza dover aprire l'editor, basterebbe fare:

```
/alias(nomealias){codice alias;}
```

es:

```
/alias(colora){echo $k($0) $1-;}
```

In pratica creerà l'alias "colora" e gli darà come codice `echo $k($0) $1-`;

Per eliminarlo, sempre direttamente dalla nostra riga di input, basterà digitare `/alias(colora){}`, cioè senza mettere codice tra le `{}`.

Inoltre considerate pure che gli alias, possono essere utilizzati, come se fossero delle funzioni, per farsi restituire dei valori tramite il comando **return**, ad esempio andiamo nella riga di input e digitiamo:

```
/alias(sum3){ return $($0 + $1 + $2); }; [INVIO]
```

```
/%Somma = $sum3(3,4,5) [INVIO]
```

```
/echo %Somma [INVIO]
```

Come vedete abbiamo creato l'alias *sum3* (che contiene il codice `$(($0 + $1 + $2)` considerando che `$(espressione_aritmetica)` serve a calcolare il valore della espressione aritmetica che contiene tra le parentesi), trattato l'alias come se fosse una funzione (notate `$sum3` invece del semplice `sum3`) passandogli 3 numeri (3,4,5) e l'abbiamo usato per farci restituire il valore della somma dei tre numeri, che gli abbiamo passato memorizzandola nella variabile `%Somma`.

Con l'ultimo comando, ovviamente, ci restituisce a schermo il contenuto della variabile `%Somma`.

Adesso digitiamo pure..

**/ECHO STOP.**

-----  
" Tu vedi cose e ne spieghi il perché, io invece immagino cose che non sono mai esistite e mi chiedo perché no." (George Bernad Shaw)  
-----

Grifisx