Subject: Re: mIRC Scripting
Posted by jnz on Thu, 03 May 2007 21:10:10 GMT
View Forum Message <> Reply to Message

If you still don't get it, you should give up.

I'm using a spoiler because this is a lot of infomation.
Toggle Spoiler
quote from "/help on TEXT"
on TEXT

The on TEXT event triggers when you receive private and/or channel messages.

Format:	on <level>:TEXT:<matchtext>:<*><?><#[,#]>:<commands>

Example:	on 1:TEXT:*help*:#mirc,#irchelp:/msg $nick what's the problem?

The on ACTION and on NOTICE events use exactly the same format as on TEXT, and trigger on an action and on a notice event respectively.

The match text can be a wildcard string, where:

*	matches any text
&	matches any word
text	matches if text contains only this word
text*	matches if text starts with this word
*text	matches if text ends with this word
text	matches if text contains this word anywhere

The match text can also be a regular expression. See the $ prefix section in Access Levels.

The location where this event occurs can be specified using:

?	for any private message
#	for any channel message
#mirc	for any messages on channel #mirc
*	for any private or channel messages

Examples

```
on 1:TEXT:hello*:#:/msg $chan Welcome to $chan $nick!
```

This listens on any channel for any line beginning with the word hello and welcomes the user who said it to the channel.

```
on 1:TEXT:*cookie*:#food:/describe $chan gives $nick a cookie
```

This listens on channel #food for any message containing the word cookie and gives the user who said it a cookie.

```
on 1:ACTION:moo:#:/msg $chan Aha, I see we have a cow among us.
```

This listens on any channel for an action that contains the word moo and responds accordingly.

```
on 1:NOTICE:*:?:/msg $nick I'm AFK, back in a moment!
```

This listens for any private notice and responds with the message that you're away from the keyboard.

For more flexibility, you can also use Variables in place of both the matchtext and the channel parameters.

```
on 1:TEXT:%matchtext:%channel:/msg $nick You just said $1- on channel %channel
```

The value of %matchtext will be matched against whatever text the user sends, and the value of %channel will be matched against the channel to which the message was sent.

Note: You can't test out these events by typing text to yourself. They can only be initiated by someone else saying something in a channel or in a private message.

quote from "/help If-then-else statements"
If-then-else statements

The If-then-else statement allows you to compare values and execute different parts of a script based on that comparison.

Basic format

```
if (v1 operator v2) { commands }

elseif (v1 operator v2) { commands }

else { commands }
```

The ( ) brackets enclose comparisons, whereas the { } brackets enclose the commands you want to be performed if a comparison is true. You must make sure that the number of ( ) and { } brackets match to make sure that the correct comparisons are made, and that the correct commands are executed.

Using brackets speeds up processing. If an alias uses too few brackets then the statement might be ambiguous and the alias will take longer to parse, might be parsed incorrectly, or might not be parsed at all.

You can nest as many if-then-else statements as you want inside each other.

The Operators

==	equal to
===	equal to (case-sensitive)
!=	not equal to
<	less than
>	larger than
>=	larger than or equal to
<=	smaller than or equal to
//	v2 is a multiple of v1
\\	v2 is not a multiple of v1
&	bitwise comparison

isin	string v1 is in string v2
isincs	string v1 is in string v2 (case sensitive)
iswm	wildcard string v1 matches string v2

iswmcs	wildcard string v1 matches string v2 (case sensitive)
isnum	number v1 is a number in the range v2 which is in the form n1-n2 (v2 optional)
isletter	letter v1 is a letter in the list of letters in v2 (v2 optional)
isalnum	text contains only letters and numbers
isalpha	text contains only letters
islower	text contains only lower case letters
isupper	text contains only upper case letters

ison	nickname v1 is on channel v2
isop	nickname v1 is an op on channel v2
ishop	nickname v1 is a halfop on channel v2
isvoice	nickname v1 has a voice on channel v2
isreg	nickname v1 is a normal nick on channel v2
ischan	if v1 is a channel which you are on.
isban	if v1 is a banned address in internal ban list on channel v2

isaop	if v1 is a user in your auto-op list for channel v2 (v2 optional)
isavoice	if v1 is a user in your auto-voice list for channel v2 (v2 optional)
isignore	if v1 is a user in your ignore list with the ignore switch v2 (v2 optional)
isprotect	if v1 is a user in your protect list for channel v2 (v2 optional)
isnotify	if v1 is a user in your notify list.

To negate an operator you can prefix it with an ! exclamation mark.

$v1 & $v2

Returns the first and second parameters of an if-then-else comparison. So, in the case of this comparison:

```
if (text isin sometext) { ... }
```

$v1 will return "text" and $v2 will return "sometext".

Combining comparisons

You can combine comparisons by using the && for AND and || for OR characters.

```
number {

if (($1 > 0) && ($1 < 10)) {

   if ($1 < 5) echo Number is less than five

   else echo Number is greater than five

}

else echo Number is out of bounds

}
```

This alias checks if the number you specify, when you type /number <value>, lies within the required range.

The ! not prefix

You can use the ! prefix in front of variables and identifiers in an if-then-else statement to negate a value. The following statements are equivalent.

```
if (%x == $null) echo x has no value

if (!%x) echo x has no value
```

Examples

```
listops {
echo 4 * Listing Ops on #
set %i 1
:next
set %nick $nick(#,%i)
if %nick == $null goto done
if %nick isop # echo 3 %nick is an Op!
inc %i
goto next
:done
echo 4 * End of Ops list
}
```

This alias lists the Ops on the current channel. It does this the hard way since we could just use $opnick() instead but using $nick() serves as an example of how isop can be used and how $null is returned once we reach the end of the list.

```
GiveOps {
```

```
%i = 0

%nicks = ""

:nextnick

inc %i

if ($snick(#,%i) == $null) { if ($len(%nicks) > 0) mode # +oooo %nicks | halt }

%nicks = %nicks $snick(#,%i)

if (4 // %i) { mode # +oooo %nicks | %nicks = "" }

goto nextnick

}
```

This is a popup definition which Ops the nicknames which are selected in the current channel nicknames listbox.

```
on 1:ctcpreply:PING* {

if ($2 == $null) halt

else {

   %pt = $ctime - $2

   if (%pt < 0) set %pt 0

   if (%pt < 5) echo 4 [PING reply] $nick is too close for comfort

   elseif (%pt < 20) echo 4 [PING reply] $nick is at just about the right distance

   else echo 4 [PING reply] Earth to $nick earth to $nick

}

halt

}
```

This intercepts a ping reply and prints out a silly message based on how far away the person is.

quote from "/help Remote Identifiers"
Remote Identifiers

You can use the following identifiers in scripts to refer to values relating specifically to events. There are also quite a few other identifiers which relate only to specific events, these are described with the events themselves in other parts of the help file. There are also other identifiers which can be used in both remote and non-remote scripts.

$1-

You can use the $1 $2 ... $N identifiers to refer to individual parameters in a line. You can also use $N- to refer to parameters N and onwards, and $N-M to refer to parameters $N through to $M. So to refer to a whole line, you would use $1-.

$0

Returns the number of space-delimited tokens in the $1- line.

$(...)

This identifier can only be used in the text match section of an event definition. It allows you to create a match parameter dynamically. You can use $1- to reference the incoming line.

```
on 1:TEXT:$(*hello $me $+ *):?:echo hello back!
```

$address

Returns the address of the user associated with an event in the form user@host.

$chan

Returns the name of the channel for a specific event. For all non-channel events $chan will be $null.

$clevel

Returns the matching event level for a triggered event.

$dlevel

Returns the current default user level.

$event

Returns the name of the event that was triggered.

$fulladdress

Returns the full address of the user triggering an event in the form nick!user@host.

$group(N/#)

Returns the name or status of a #group in a script.

Properties: status, name, fname

$group(0)	returns the number of groups
$group(1)	returns the name of the first group
$group(1).status	returns on or off for the first group
$group(#test)	returns on or off for group #test

$group(#test).fname	returns the script filename in which the group exists
$group(3).name	returns the name of the 3rd group

$maddress

Returns the address that was matched for the triggered event.

$matchkey

Returns wildcard matchtext that was used in the matching remote event.

$mode(N)

Returns the Nth nick affected by a channel mode change.

Properties: op, deop, ban, unban, voice, devoice, help, dehelp

The properties can be used to specify which type of mode change you want to check.

$mode(0).op returns the number of opped nicks

$mode(1).op returns the first opped nick

Note: This identifier is only used in conjunction with the on OP/DEOP etc. events.

$nick

Returns the nickname of the user associated with an event.

$numeric

Returns the numeric for the matching numeric event.

$rawmsg

Returns raw server line for server events.

$script

Returns the filename of the currently executing remote script.

$script(N/filename)

Returns the filename for the Nth loaded script file. If you specify a filename, it returns $null if the file isn't loaded.

$script(0)	return the number of script files loaded
$script(2)	returns the filename of the 2nd loaded script file
$script(moo.txt)	returns $null if the file isn't loaded, or moo.txt if it is.

Note: This can't be used to reference the users or variables files.

$scriptdir

Returns the directory of the currently executing remote script.

$scriptline

Returns line number in current script.

$site

Returns the portion of $address after the @ for the user associated with an event in the form user@host.

$target

Returns the target of an event.

$ulevel

Returns the user level that was matched for the currently triggered event.

$ulist(nick!userid@address,L,N)

Returns the Nth address in the Users list that matches the specified address and level.

Properties: info

You can specify a wildcard address or a * to match any address in the user list. If you don't specify a full address, it completes the address with wildcards. If you don't specify N, the first matching address is returned.

If you specify L, only matching addresses that contain the specified level are returned.

Note: L and N are optional, but if you specify L, you must specify N.

$wildsite

Returns the address of the user who triggered an event in the form *!*@host.