Contents

Introduction

Welcome to **mIRC**, an Internet Relay Chat Client.

mIRC attempts to provide a user-friendly interface for use with the **Internet Relay Chat** network. The IRC network is a **virtual meeting place** where people from all over the world can meet and talk.

To IRC all you need to do is Connect to a server, Join a channel, and Chat!

mIRC will guide you through these initial stages and hopefully you'll be chatting in no time. If you get stuck or want to find out more about a certain feature, just click on a Help button eg. in the Setup dialog, and you should find some hints there to help you.

As you become more **experienced** you can also start **configuring** mIRC's features to suit your own needs and tastes, features such as colours, fonts, function keys, aliases, popup menus, scripts, sounds and many others.

Remember to visit the **mIRC homepage** or one of it's mirror sites regularly for the latest version information, as well as for links to other mIRC and IRC related information and websites.

United Kingdom	http://www.mirc.co.uk

This website is owned by **Khaled Mardam-Bey**, the author of mIRC, and is maintained by **Tjerk Vonck**, the author of the mIRC homepage and FAQ.

Argentina	http://www.mirc.com.ar
Australia	http://mirc.eon.net.au
Brazil	http://www.conesul.com.br/mirc
Italy	http://www.mirc.queen.it
Kuwait	http://mirc.kems.net
The Netherlands	http://www.nip.nl/mirc
South Africa	http://www.mirc.co.za
California, USA	http://www.geocities.com/~mirc

The above are **mirror sites** which are updated regularly with the same information as the main mIRC website.

You can also find a **selection** of additional mIRC and IRC related websites which are maintained by **IRC enthusiasts** at:

http://www.mirc.co.uk/links.html

Keep in mind that websites come and go, so some of these addresses **might** become outdated at some time.

Setting Options

You can change the various options in mIRC through the Setup and Options dialogs. The Setup dialog has options related to connecting to IRC Servers, whereas the Options dialog has general settings for various features in mIRC.

Setup

IRC Servers
Local Info
Options
Identd
Firewall

Options

IRC Switches
Control
Double-Click
Event Beeps
Drag-Drop
Sound Requests
URL Catcher
Perform
Flood
Logging
DDE Control
Finger Server
Extras

Options

Connect on startup
If this is turned on, mIRC tries to connect to the default IRC server automatically when it is run.

Reconnect on disconnection
If an IRC server disconnects you without you having typed /quit then checking this command will make mIRC reconnect automatically.

Popup setup dialog on startup
If enabled, this pops up the setup dialog when you run mIRC.

Move server to top of list on connect
If a single server is selected in the servers listbox, and you double-click on it, or you click the connect button, the server is moved to the top of the list.

On connection failure...
If you attempt to connect to a server and the connection fails, mIRC will retry the connection the specified number of times and will pause the specified delay inbetween each retry. If you selected multiple servers, mIRC will cycle through the list until a connection is made.

Note: Because of the way winsock works, this might cause problems. Winsock has it's own built-in timeout delay, so even if mIRC cancels a connection, winsock might still need to pause for 10 seconds.

Default Port
This allows you to specify a default server port that will be used in case you don't specify a port for a server.

DCC

DCC allows you to connect **directly** to another irc client, **bypassing** the IRC Network, to **Send** and **Get** files, and to **Chat** privately over a more secure connection.

DCC Send

DCC Send allows you to send a file to another user. Enter the nickname of user, select the file you want to send, and click on the Send button. mIRC will tell the user that you want to send them a file. The user then has to accept your send request, at which point the file transfer will begin.

Packet Size

The packet size is the number of bytes that mIRC will send to another client in one packet. The minimum is 512, the maximum is 4096.
If you check minimize then the dcc send window(s) will be minimized automatically.

Fill Spaces

The fill spaces option is only available in the 32bit version and only under operating systems that allow spaces in filenames. It is highly recommended that you leave this option turned on. For more information read this.

Fast Send

Turning on this option should speed up your DCC sessions. You can also change this setting with the **/fsend [on|off]** command.

Note: DCC Send needs to use your IP address to initiate a connection with another client. If you are having trouble initiating a connection then your IP address might be wrong. See the Local Info section for more information.

DCC Get

When someone tries to DCC Send you a file, the DCC Get dialog pops up and asks you if you want to accept the file.

If you choose to **accept** the file, mIRC will ask the sender to begin the file transfer, at which point you should begin receiving the file.

DCC Resume

This feature allows you to resume DCC transfers that failed to complete.

If a user tries to send you a file that already exists in your get directory then you will be shown a warning that the file exists. You then have the option to either overwrite, resume, or rename the file.

If you select overwrite then the whole file will be downloaded from the beginning and any existing file of the same name will be erased.

If you select resume then mIRC will attempt to negotiate a transfer resume to get the remaining part of the file. It will append this to the portion of the file you already have.

The mIRC DCC Resume Protocol is described here.

DCC Chat

DCC Chat allows you to talk privately with another user. Enter the nickname of the user and click on the chat button. mIRC will send information to the user telling them that you want to DCC Chat. The user then has to accept your chat request, at which point you can talk with them normally as if you were in a query window on IRC.

If a user sends you a chat request, a chat dialog will pop up asking you whether you want to talk accept their dcc chat. You can then accept or decline.

Note: DCC Chat needs to use your IP address to initiate a connection with another client. If you are having trouble initiating a connection then your IP address might be wrong. See the Local Info section for more information.

DCC Options

The DCC Options dialog allows you to change various DCC related settings.

Show Get Dialog/Auto-get file/Ignore all

By default, a send request must be accepted by you before a transfer begins. If you select the **Auto-get file** option then mIRC will automatically accept a send request and begin receiving a file. If you select **Ignore All** then all incoming dcc send requests are ignored.

If Auto-get and file exists

If someone tries to send you a file that already exists in your dcc get directory, by default mIRC pops up a dialog asking you if you want to overwrite/resume/rename the file. If you check the **Resume** option, mIRC will resume the file automatically, and if you check the **Overwrite** option, mIRC will overwrite the file automatically .

Max. DCC Sends

This limits the number of simultaneous remote DCC Sends.

Note: This doesn't apply to you manually initiating a DCC Send but it does apply to users who request a DCC Send remotely ie. via a script file or via the DCC Server.

Show Chat Dialog/Auto-accept chat/Ignore all

By default, a chat request must first be accepted by you before chatting begins. If you select the **Auto-accept** option then mIRC will automatically open a DCC chat window and accept the chat request. If you select **Ignore All** then all incoming dcc chats are ignored.

Max. Fileservers

This limits the number of fileserver sessions that can be open simultaneously.

Max. DCC Gets

This limits the number of simultaneous DCC Gets a user can request.

Root Directory

This specifies the root directory that a user will see when they first enter a fileserver initiated via the DCC Server. The user will be able to access all files and directories within this root directory.

Welcome text file

This specifies the welcome text file that will be sent to a user when they first connect to a fileserver.

Display fileserver warning

If this option is checked, the fileserver warning is displayed whenver the /fserve command is used to initiate a fileserver session. To learn more about fileservers see the File Server section.

On DCC Completion...

mIRC will perform the selected options once a transfer has been completed. mIRC will indicate in the status window whether the transfer was a success or a failure. The beep options depend on the settings in the Event Beeps dialog.

Get/Chat Dialog Time Out

When a user sends you a Send or Chat request, a dialog pops up and waits for you to accept or decline. The dialog will wait the specified number of seconds before disappearing.

Send/Get Transfer Time Out
During a transfer mIRC will wait the specified number of seconds for a response from the other client before closing the connection.

Fileserver Time Out
mIRC will close the fileserver window if a user has been idle for the specified number of seconds.

DCC Ports
This allows you to specify the range of ports that mIRC will use when making DCC connections. The usual range is 1024 to 5000, though it is possible to specify 64000 as the highest port (however there is no guarantee it will work).

DCC Get Directories
This allows you to specify dcc get directories where received files will be placed according to their extension. Files which don't match any of the extensions you specify are placed in the default get directory.

If you specify a **command** to be performed when a file is downloaded, the **$1-** identifier refers to the filename.

DCC Server
The mIRC DCC Server listens for special types of connections from other **mIRC** clients.

Enable DCC Server
This turns the DCC Server on or off.

Listen on Port
The DCC Server listens on port 59 by default, however you can this to another port number.

Listen for...
You can have the DCC Server listen for only certain types of connections, such as DCC Sends, Chats, or Fileserver requests. For example, if you turn off the DCC Chat listen option, the DCC Server will ignore any chat requests.

Perform DNS lookup
When someone connects to your DCC Server only their IP address is available for identification. If you check the DNS lookup switch, mIRC will perform a /dns on the IP address to try to resolve it to a named address.

Note: It can take anything from a second to more than minute to resolve an address depending on network conditions, and sometimes it may not work at all.

Commands
You can change the settings of the DCC Server from the command line using:

```
/dccserver [+|-scf] [on|off] [port]
```

You can Send/Chat to the DCC Server using the DCC Send/Chat dialogs and specifying an **IP address** instead of a nickname.

From the command line, you can use /dcc [send|chat|fserve] with an IP Address instead of a nickname to initiate a connection to the DCC Server.

Note: /dcc fserve only works for IP address connections and does not work via IRC.

For a technical specification of the **DCC Server protocol**, see here.

Window Menu

The Tile, Cascade, and Arrange Icons menu items work the same way they usually do. If you want a **vertical** tiling of windows hold down the **Shift** key when you select tile.

The **MDI** window dialog allows you to specify how you want different types of windows to be opened. If a window is opened as an **MDI window** it will be contained **within** the main mIRC application window. If it is opened as a **Desktop window**, it can be placed **outside** and is independent of the main mIRC application window.

If the window is opened as a **Desktop window**, you can also choose to have it **stay on top** of all other windows by selecting its System Menu.

The **auto-cascade** and **auto-tile** options reposition/resize windows every time a new window is created or destroyed so as to make all of the windows more accessible.

The **auto-arrange** option re-arranges minimized icons (when the switchbar is turned off) whenever an icon window is closed.

A list of currently open windows is shown beneath these items.

Help Menu

The mIRC help menu is **dynamic** which means that mIRC looks in its **current** directory for files ending in **.hlp** and **.txt** and adds them to the **help** menu for easy access. It also makes **aliases** for each of the files listed in the help menu, the aliases being the **name** of the file **excluding** the extension.

For example, if you put a help file by the name of **winsock.hlp** in your mIRC directory, mIRC would add the **winsock.hlp** item to your help menu, and would make an alias **/winsock**. You can then type **/winsock sometext** and mIRC would do a **context-sensitive** search in that help file for the text you specified.

You can **add/remove** files whenever you want from the mIRC directory, mIRC always **updates** the help menu whenever you open it.

If you download the mIRC FAQ in the **.hlp** format into the mIRC directory, it will appear in this menu.

Miscellaneous stuff

Saving options
mIRC saves the main window position and all other settings when you exit the program. Note that the sizes (and not positions) of the finger, dcc get/chat/send, channel list, etc. windows are saved. The next time one of these windows opens up, it will be the same size as the last time you used it.

Toolbar
If you click your right mouse button on the Setup dialog Toolbar button, a popup menu of the top 15 IRC Servers in your servers list is displayed.

Multiple Instances
You can only run multiple instances with the 32-bit mIRC. With the 16-bit mIRC, you cannot run multiple instances from the same EXE. However, you can have two copies of mirc, one mirc.exe and the other mirc2.exe, and you will then be able to connect to two different servers at the same time.

The Cookies
A squeaky sound, a multitude of silly comments, a faithful pet, one bouncing dot, and a smiley face. Where oh where can they be? :)

The Author (and part-time Human Bean)

Greetings! :)

I am **Khaled Mardam-Bey**, I can be reached via email at khaled@mardam.demon.co.uk, and you can visit my website at http://www.mirc.co.uk/khaled.

One day, three years ago, in a small, quiet room by myself, I stood in front of a canvas, picked up a brush, and began to work on a painting, dabbling here and there, not really knowing **what** I was going to create, or **how**.

Three years later, I put my brush down for a moment to pause for **breath**, look around, and find that I'm now in a **different** room, a much bigger one, filled with all kinds of people, with colours and sounds, and with **other** paintings, bigger and smaller than mine; some of the people are standing in front of me, close to the canvas, touching and **smudging** it, others are behind me looking at my painting from a distance; some are even standing right **next** to me, telling me **which** colours to use or **how** to hold my brush, and criticising me whenever I make a mistake; there are a few who are working on parts of the painting all by themselves.

I look around today and see all of these wonderful faces; well, some of them are sticking their tongues out at me, but **overall** they look quite friendly ;) and I ask myself, how did all of this happen?

mIRC was, and still is, a **hobby** for me, a **personal** work that I enjoy; even after three years of working on it and with it being quite popular, I've pretty much kept that view. I **still** work on mIRC from an appartment, as I've always done, except now I use a notebook as my main computer which allows me to travel more easily with my work when I need to.

It's been difficult, reconciling mIRC as a personal work with it's public use; I've had to learn to treat **some** aspects of it impersonally, which I'm not happy about, eg. I now use a pre-written reply to emails that ask the most common questions; if I tried to answer every email, I would have little time for anything else.

I've also had to learn to accept **criticism**, which is hard for something that's personal; it's been an odd balancing act, treating mIRC as something that I do for myself and yet for others as well; but I know that mIRC wouldn't be where it is today without help from many people over the years and I am thankful.

These days I still enjoy working on mIRC, though I'm getting a **bit** more tired now, but I think there may be a year or two left in me yet :) and I still get nervous before releasing a new version, which is probably a good sign ;) I hope you enjoy it.

If you like mIRC, please try to register as that will support my work on it. Thank you :)

I hope that mIRC has had, and will continue to have, a part to play in the making of new **friendships**, in the **keeping** of old ones, in the fostering of peaceful **communication**, and in the increased **understanding** and **respect** of other people and cultures, and that it has had a **positive** effect on people's lives.

I hope you enjoy using mIRC as much as I enjoyed creating it :)

Khaled

License Agreement

mIRC® v5.41 Internet Relay Chat Client
Copyright © 1995-1998 Khaled Mardam-Bey and mIRC Co. Ltd.
All rights reserved.

mIRC is **Shareware** which means that you can use it legally for **30 days** free of charge to evaluate it. If during, or at the end of, that period you decide that you would like to continue using it, please **register** your copy. Your single-user registration will license you to use your copy of mIRC, will support work on future versions, new features, and bug fixes, and will provide you with technical support via email.

Please see the Registration section to find out how you can register.

mIRC may be freely distributed subject to, but not limited to, the following terms: mIRC may not be sold or resold, distributed as a part of any commercial package, used in a commercial environment, used or distributed in support of a commercial service, or used or distributed to support any kind of profit-generating activity, even if it is being distributed freely.

If you would like to **distribute** mIRC as part of a shareware distribution, magazine, internet book, CD ROM, etc. contact me for written permission.

All **commercial** use interests in mIRC eg. site licensing, should be directed to **khaled@mardam.demon.co.uk**.

The integrity of the original mIRC distribution file as distributed by Khaled Mardam-Bey is essential. mIRC and all of its related files must be distributed together in the original format. The mIRC distribution file may not have files added to it or removed from it, and none of its contents may be modified, decompiled, or reverse engineered.

mIRC is provided **AS IS** without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. In no event shall Khaled Mardam-Bey or mIRC Co. Ltd. be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Khaled Mardam-Bey or mIRC Co. Ltd. have been advised of the possibility of such damages.

mIRC is a registered trademark of mIRC Co. Ltd.
The mIRC Logo is a trademark of mIRC Co. Ltd.

Double-click

This feature allows you to enter a set of commands that will be executed whenever you double-click in the specified window type.

For example, for the **channel nickname listbox** you could enter:

/query $1

which means that when you **double-click** on a nickname, a query window opens up and you can start talking with that nickname privately.

Aliases

mIRC allows you to create aliases and scripts to speed up your IRC session or to perform repetitive functions more easily. To create aliases you must know some IRC commands.

Aliases can be called from the **command line**, from **other aliases**, and from **popup** and **remote** scripts. An alias **cannot** call itself recursively mainly because this seems to cause more problems for users than it solves.

Examples

The following examples show you how to create aliases that perform simple functions.

```
/gb /join #gb
```

If you now type **/gb** this is the same as typing **/join #gb**.

```
/j /join $1
```

We have now added a **parameter** string. If we type **/j #gb** this is the same as typing **/join #gb**. The **$1** refers to the first parameter in the line that you supply.

```
/yell /me $2 $1
```

If you now type **/yell There! Hello** the action command will be **/me Hello There!** The number after **$** specifies the number of the parameter in the string that you entered.

```
/jj /join $?
```

The **question mark** indicates that you should be asked to fill in this parameter. The parameter you supply will be inserted in the line at that point. So if you type **/jj** a dialog will pop up asking you for the channel you want to join. If you enter **#gb** then the final command will be **/join #gb**.

```
/jj /join #$1
```

the **# sign** indicates that the parameter you specify should be **prefixed** with a hash indicating that it is a **channel**.

```
/jj /join $?="Enter channel to join:"
```

This does the same thing but now the dialog will have the "Enter channel to join:" line displayed inside it.

```
/aw /away $?="Enter away message:" | /say $!
```

This is similar to the line above except for the addition of the **$!** parameter. This refers to the text you **just** typed into the parameter box. ie. the away message. This saves you having to type the same message twice.

```
/give /me gives $$1 a $$2
```

The **double $$** means that this command will **only** be executed if a parameter is specified. If you specify only one parameter in the above command it will **not** be executed. You can also do **$$?1** or **$?1** which means try to fill this value with parameter one if it exists. If parameter one doesnt exist, ask for it. In the first case the parameter is necessary for the command to be executed, in the second case it isn't.

```
/slap /me slaps $1 around with $2-
```

The **$2-** indicates that everything following and including **parameter 2** should be appended to the command line. if you type **/slap Sheepy a large trout** the final line will be **/me slaps Sheepy around with a large trout**.

You can also specify **$2-5** which means use only parameters 2 to 5.

```
/laugh /me laughs at $1's joke
```

Anything appended to a **$** parameter is appended to the final parameter. So if in the above example we type **/laugh mimi** the final command would be **/me laughs at mimi's joke**.

```
/silly /say Hel $+ lo th $+ ere $+ !
```

Parameters are normally separated by a space. To make mIRC **combine** parameters you can use the **$+** identifier. The above line will say **Hello there!**.

```
/p /part #
```

The **# sign** refers to the **channel** you are currently on. So if you are on channel #blah, and you type **/p** then the client replaces the # sign with #blah, and the final command is **/part #blah**.

```
/op /mode # +o $1
```

To op someone you can now just type **/op goat** instead of the whole /mode command.

```
/dop /mode # -ooo $1 $2 $3
```

You can now deop three users by typing **/dop goat mike bongo**.

For **multiple commands** you should use a **|** character (the shifted character usually under the \ key). So to write an alias that kicks and bans someone:

```
/dkb /kick # $1 | /mode # +b $1
```

The [ ] evaluation brackets

If you want greater control over the order of evaluation of identifiers, you can use the **[ ] brackets**. Identifiers within these brackets will be evaluated first, from left to right. You can **nest** brackets.

```
/say % [ $+ [ $1 ] ]
```

You can also **force** a previously evaluated identifier to be re-evaluated by using extra [ ] brackets.

```
/set %x %y
/set %y Hiya!
/echo [ [ %x ] ]
```

The { } brackets

You can create **multi-line** scripts by using the **{ } brackets**. This allows you to create an alias which performs several commands.

```
/poem {
  /msg $1 A horse is a horse..
  /msg $1 A duck is a duck..
}
```

The If-then-else statement

You can use if-then-else statements to decide which parts of your script executes based on the evaluation of a comparison.

```
/number {
  if ($1 == 1) echo The number ONE
  elseif ($1 == 2) echo The number TWO
  else echo Unknown number!
}
```

This creates an alias which tests if the parameter you supplied is the number 1 or the number 2.

For more information, see the if-then-else section.

The Goto command

The goto command allows you to **jump** from one point in a script to another point.

```
/number {
  if ($1 == 1) goto one
  elseif ($1 == 2) goto two
  else goto unknown
  :one
  echo The number ONE
  halt
  :two
  echo The number TWO
  halt
  :unknown
  echo Unknown number!
  halt
}
```

Using a **goto** incorrectly could lead to an **infinite loop**. You can **break** out of a currently running script by pressing **Control-Break**.

Note: I didn't prefix the above commands with the **/** command prefix. This is because the command prefix is really only needed when entering a command on the command line. In scripts, all lines are assumed to start with a command, so you don't need to use the / command prefix.

Identifiers and Variables

An **Identifier** returns the value of a built-in mIRC variable. For example, $time would return the current time. Whenever mIRC finds an identifier in your command, it replaces it with the **current** value of that identifier.

For a list of identifiers, see the Identifiers section.

Variables are identifiers whose values you can create and change yourself and use later in your scripts.

For more information on variables, see the Variables section.

Custom Identifiers

A custom identifier is just an **alias** which returns a value, and you can use that aliases name with an identifier prefix.

For example, create an /add alias such as:

```
add {
  %x = $1 + $2
  return %x
}
```

And then use it in a command:

```
//echo Total is: $add(1,2)
```

You can supply as many parameters as you want to your identifier eg. $add(1,2,...,N).

Built-in identifiers of the same name have priority.

Remote Scripts

You can add aliases to remote scripts by using the **alias** prefix and then entering your alias as usual.

```
alias add {
  %x = $1 + $2
  return %x
}
```

This is the same custom identifier as above, except it uses the **alias** prefix.

If you specify the **-l** switch in the alias definition, the alias becomes accessible only by commands in the same script and invisible to the command line and other scripts.

```
alias -l add {
  %x = $1 + $2
  return %x
}
```

Function Key support

You can **redefine** function keys to perform certain commands, just like aliases. For example:

```
/F1 /say Hello!
/sF2 /query $1
/cF3 /ctcp $1 version
```

The **s** and **c** prefixes for **Shift** key and **Control** key respectively.

Note: A function key will behave **differently** depending on the window in which it is used. For example, when using it in a **query window** the $1 parameter refers to the selected users nickname. If you're on a **channel** and the **nickname listbox** is active then the function key will work on the selected nicknames. If the listbox is **not** active, the function key will just work on the channel.

Command prefixes

If you are executing a command from the command line ie. by typing it into an editbox, you can **force** mIRC to **evaluate** identifiers in that command by prefixing it with two **//** instead of one /. For example:

```
/echo My nickname is $me
```

Would print out "My nickname is $me" and would not evaluate the $me.

```
//echo My nickname is $me
```

Would print out "My nickname is Pengy" if your nickname was Pengy.

If you want to force a command to peform **quietly** ie. without printing out any information, then you can prefix it with a "." fullstop. For example:

```
/ignore somenick
```

Would print out information telling you that you are now igoring "somenick". If you don't want this information to be displayed, then you can used:

```
/.ignore somenick
```

Comments

You can add **comments** to your scripts by using the **; semi-colon** at the start of a line.

```
;This is a comment
```

You can place comments anywhere in a script, they are ignored during processing.

Popups

mIRC allows you to create custom popup menus for the status window, query/chat windows, channel windows, the channel nickname listbox, and main menubar. To create these you must know how to use Basic IRC commands, how to create Aliases, and how to use Identifiers and Variables.

If you click the right mouse-button in a window, the popup menu for that window will appear and you can select menu-items which you have defined to perform certain tasks, such as opping a user or joining a channel.

Examples

Popup menu definitions use the format:

```
<menuitem>:<commands>
```

Get Help:/join #irchelp

The words before the ":" colon are the name of the menuitem. The words after the ":" colon are the commands that are to be performed. In this case, the menuitem you would see is "Get Help". The command that would be performed if you select this menuitem would be "/join #irchelp", as if you had typed it.

The format of the commands follows precisely the same as those in normal aliases. See the aliases help section to understand how to write an alias.

To create a **Submenu**, use a "." fullstop.

```
Join a Channel
.Get IRC help!:/join #irchelp
.Visit the folks at #friendly:/join #friendly
.Who shall we join?:/join $?
```

In this case, the name of the submenu is "Join a Channel". All the commands following it beginning with a "." are part of this submenu.

To create **menus within a submenu**, you just add more fullstops:

```
Channels
.Help
..irchelp:/join #irchelp
..mIRC:/join #mirc
..newbies:/join #newbies
.Other Channels
..Visit #friendly:/join #friendly
..Wibble Wobble:/join #wibble
.Join?:/join #$$?="Enter a channel name:"
```

To **separate** menuitems, you can use a single "-" dash on a line by itself.

```
whois ?:/whois $?
-
Misc
.Edit Temp:/run notepad.exe temp.txt
.say?: /say $?
```

```
.action?:/me $?
Names
.#irchelp: /names #irchelp
.#friendly: /names #friendly
.names ?:/names $?
-
channel list:/list
-
Join a Channel
.Get IRC help!:/join #irchelp
.Visit the folks at #friendly:/join #friendly
.Who shall we join?:/join $?
```

To use the popup menu for a channel nickname listbox, you need to select a nickname before the menu will pop up. Here is a simple nickname listbox popup menu:

```
Who Is?:/whois $1
-
Modes
.Op:/mode # +o $1
.Deop:/mode # -o $1
.Kick, Ban:/kick # $1 | /ban $1
-
DCC Send:/dcc send $1
DCC Chat:/dcc chat $1
-
Slap!:/me slaps $1 around a bit with a large trout
Query:/query $1 Hey you! hello? are you there...?
```

If you want to create a popup menu item which performs **several commands**, you can use the { } brackets. See the Aliases section for more information on how to use these.

```
Cookie {
  if ($1 == $me) echo I give myself a cookie!
  else echo I give $1 a cookie!
}
```

The above menitem can be used in the channel nicknames listbox. The $1 refers to the nickname of the user you have selected in the listbox. In this case, it checks to see if I've selected my own nickname, and if so displays the first message, otherwise it displays the second message.

The popup menus for the **Query/Chat** and the **MenuBar** work the same way as the channel listbox popup menu.

Identifiers and Variables

Variables or identifiers that are a part of a the title of a menu definition are evaluated each time the popup appears. This allows you to create popup menus that vary in appearance. If an entire menu item evaluates to $null, it is not displayed.

Remote Scripts

You can place **menu** definitions in your remote scripts by using the **menu** prefix.

```
menu status {
  Server
  .Links:/links
```

```
   .Lusers:/lusers
   .Motd:/motd
   .Time:/time
}
```

This definition would add a submenu to your status window popup menu.

You can also specify channel, query, nicklist, and menubar as the menu name, and this will add menuitems to your current popup menus for each of these windows.

```
menu nicklist {
   Slap
   .Haddock:/me prods $1 with a haddock
}
```

This definition would add a submenu to your channel nickname listbox popup menu.

You can also specify popup menus for custom windows by specifying the custom window name.

```
menu @test {
   dclick:/echo double-click!
   <menuitems>
}
```

The **dclick** definition allows you to specify a command that will be performed when you double-click in a custom window listbox. You can also refer to $1 which holds the line number of the line that was double-clicked.

You can also specify **multiple** window names for a menu, eg.:

```
menu @dogs,@cats,@goats {
   <menuitems>
}
```

Extras

The options in this dialog are extra bits and pieces that don't fit anywhere else.

Fast screen update
For users with slower video cards, this updates the screen in batches instead of individually line by line. This should speed up screen updating for most users by quite a bit.

Multi-line editbox
In single line mode, the text scrolls to the right when you reach the end of the edit box. In multi-line mode, the text starts again at the left and scrolls downwards. It's probably easier to use multi-line mode since you can see what you've typed if the line is long. This mode also allows you to paste several lines of text into the editbox.

Note: The status window permanently has a single line edit box.

ESCape key minimizes window
To have a window minimize whenever you press the ESCape key select this option.

Pop up colour index
If this is turned on, mIRC will pop up a colour index dialog whenever you press Control-K in an editbox to insert a colour code.

Toolbar and tooltips
You can choose to have the toolbar and it's tooltips hidden or shown.

Minimize to tray
This option only appears in the 32bit mIRC under operating systems that have a tray.

If this option is turned on, when you minimize mIRC the mIRC window is hidden and a small mIRC icon appears in the tray.

If this option is turned off, you can force mIRC to minimize to the tray by holding down the **Shift** key when you minimize mIRC.

If you **right-click** on the mIRC icon in the tray, a popup menu listing your current channels/queries appears; those that have a checkmark next to them have had their icons highlighted due to incoming messages. You can click a menu item to open its respective window.

If you hold down the Shift key when you quit mIRC, the next time you run it, it will be minimized.

Switchbar...
You can choose to have the switchbar shown or hidden. The switchbar shows all of your window's as buttons to allow quick access and switching between them.

If the **fixed width** option is on, the buttons will not span the entire length of the switchbar.

If the **always highlight** option is turned on, mIRC will highlight a window's button when there is activity in the window and the window is open and non-active. If this option is turned off, mIRC will **only** highlight the window's button if there is activity in it and the window is iconized.

The **Show DCCs** switch allows you to show **DCC sends/gets** in the switchbar, otherwise they will appear as normal icons.

You can **move** the switchbar to different positions in your mIRC window by **clicking** on it and **dragging** it with the mouse.

Line separator
You can specify a line separator to be used in the status window. You can use a space to have a blank line. If you leave the box empty, lines in the status window will not be separated.

Command Prefix
The default standard prefix to a command is a ***/*** character, however you can specify another character if the ***/*** key is hard to access on your keyboard.

Note: Regardless of the character you choose here, mIRC still recognizes the ***/*** character and uses it internally.

Scrollback buffer size
This limits the scrollback buffer to the specified number of lines. Note that if a scrollback buffer already contains more than the specified number of lines it will be shortened. You can always use the **/clear** command in a window to clear the scrollback buffer completely. If you are scrolling back through the buffer, lines will not be removed until you return to the bottom of the buffer.

Append this text to the application title bar
This allows you to specify text that will be shown in the mIRC application title bar.

Joining A Channel

Once you've connected to an IRC Server, you can join a channel to talk to other people. There are several ways to join a channel, each is explained below.

The Channels Folder

The easiest way to join a channel is to use the channels folder which holds a list of your favourite channels. mIRC **automatically** pops up this folder the moment you connect to an IRC Server. You can join one of the listed channels by selecting it and clicking the **Join** button.

You can also view the channels folder by clicking on the Channels Folder button in the toolbar.

The Channels List

Another way to join a channel is to retrieve the list of currently active channels by using the List Channels dialog. You can view the List Channels dialog by clicking on the List Channels button in the toolbar.

To retrieve the **entire** channels list, you can click on the **Get List** button. The list can be quite long, sometimes over 3000 channels, so it can take several minutes to retrieve it. mIRC will save this list once you have retrieved it, so if you wish to view it again later you can just click on the **Apply** button. However, if you want an updated list you will need to retrieve it again.

If you click your right mouse button in the channels list window, a popup menu with various options will appear.

Note: You can also specify a filename for the channels list which can be useful if you regularly visit different IRC networks.

The /join Command

The format of the /join command, which is a Basic IRC Command, is **/join #channel** where #channel is the name of the channel you want to join. So if you wanted to join channel #mIRC, you would type **/join #mIRC** and press enter, and a moment later the #mIRC window will open indicating that you have joined it.

Creating a new channel

You can create a new channel if it doesn't already exist just by joining it. So, let's say you want to create a channel called #bubbles, you would just type **/join #bubbles**, and if it doesn't exist it will be created for you. If it does exist, you will just join it as usual.

Talking on a channel

You can talk to other people by typing in a message and pressing the enter key. Your message will be sent to the channel and everyone on the channel will see it. A good first message is just to say hello with a smiley face :)

The **listbox** on the side of the channel window lists all of the people who are currently on that channel. If you click your right mouse button in the listbox, a popup menu with various options will appear.

Popup menus are actually used everywhere in mIRC, you can even click your right mouse button in the status window, or in the channel window itself, and a different popup menu will appear. These popup menus are configurable, you can change them in the popups dialog to perform whatever functions you require.

Leaving a channel

You can leave a channel by clicking it's window's close button, or you can use the **/part** command, which is another basic IRC command similar to /join. The format of the /part command is **/part #channel** where #channel is the name of the channel you want to leave. If you type **/part** without a channel name and press enter, you will part the current channel.

Acknowledgements - Thank you, thank you, and thank you...

After three years of working on mIRC, it's hard to remember all of the people who've been involved with some aspect or other of its development; some I haven't heard from in a very long time, while others I still talk to today. They've all contributed in some way, whether through bug reports, suggestions, answering questions, beta-testing, helping out folks on IRC channels, or creating helpful scripts, documentation, and web pages.

I get emails about mIRC every day from people all over the world, emails which are often very touching and heartfelt, and I know that their kind words are as much a tribute to all of you who have helped to bring so many people together.

Thanks to...

Tjerk Vonck who created the first #mIRC channel, the mIRC homepage, and mIRC FAQ, and has contributed to mIRC in too many ways to mention.

The **mIRC Beta-testers**, past and present, for their hard work in catching bugs and suggesting features.

The folks who help out new users day after day on channels, and who create helpful webpages, documentation, and scripts.

Jarkko Oikarinen, creator of IRC, and all of the **ircd coders** after him who have worked hard to improve IRC for all of us.

Nicolas Pioch for his short IRC primer, **Ramji**, **Bibbly**, **Stimps**, **Mike**, **James G.**, **Edward**, **Bunster** for the queen-size futons, **Peeg** for the frequent *bonk*s over the head, and **Viv** :)

M.Hunnibell, **M.Overtoom**, **Mark "Too Slick" Hanson**, **Ron R.**, and **Dan Lawrence** for their technical help.

Richard "Budman" Jones, who designed and contributed the new logo.

All of the other kind folks who directed me at different people, places, source codes, and documents, discussed ideas with me, and in general both **supported** and **laughed** at my humble efforts :)

Basic IRC Commands

IRC commands allow you to perform functions such as maintaining control of a channel and the users on it. The following list of basic IRC commands should help you get started. There are also mIRC Commands you can look at.

General Commands

/JOIN #channel
Join the specified channel.

example: /join #irchelp

This will make you join the #irchelp channel. Once on a channel, anything you type will be seen by all the users on this channel. The #irchelp channel is very useful, so say hello and then ask any questions you want. If the channel you specified doesn't exist, a channel with that name will be created for you.

Some channels may also have keys ie. a password, which you need to specify when using the /join command.
example: /join #irchelp trout

This will make you join the #irchelp channel using "trout" as the password.

/PART #channel
Leave a channel.

example: /part #irchelp

/LIST [#channel] [-MIN #] [-MAX #]
Lists currently available channels. You can also tell mIRC to show only channels with a minimum and a maximum number of people. If you specify a #channel then mIRC will only list information for that channel. If you specify wildcards, eg. *love* then mIRC will list all channels that contain the word **love** in them.

example:	/list
example:	/list -min 5 -max 20
example:	/list #mirc
example:	/list *love*

/ME message
Tells the current channel or query about what you are doing.

/MSG nickname message
Send a private message to this user without opening a query window.

/QUERY nickname message
Open a query window to this user and send them a private message.

/WHOIS nickname
Shows information about someone.

/NICK nickname
Changes your nickname to a new nickname.

/QUIT [reason]
This will disconnect you from IRC and will give the optional message as the reason for your departure. (this message only appears to people who are on the same channels as you).

example: /quit That's all folks!

/AWAY [away message]
Leave a message explaining that you are not currently paying attention to IRC. Whenever someone sends you a MSG or does a WHOIS on you, they automatically see whatever message you set. Using AWAY with no parameters marks you as no longer being away.

example: /away off to get something to eat, back in a moment!

/TOPIC #channel newtopic
Changes the topic for the channel.

example: /topic #friendly Oh what a beautiful day!

/INVITE nickname #channel
Invites another user to a channel.

Channel and User Commands

If you have Op status, the following commands give you control over both a channel and the users on it.

/KICK #channel nickname
Kicks named user off a given channel.

example: /kick #gb Ed

/MODE #channel|nickname [[+|-]modechars [parameters]]
This is a powerful command that gives channel operators control of a channel and the users on it.

Channel modes

ModeChar	Effects on channels
b <person>	ban somebody, <person> in "nick!user@host" form
i	channel is invite-only
l <number>	channel is limited, <number> users allowed max
m	channel is moderated, (only chanops can talk)
n	external /MSGs to channel are not allowed
o <nickname>	makes <nickname> a channel operator
p	channel is private
s	channel is secret
t	topic limited, only chanops may change it
k <key>	set secret key for a channel

User modes

ModeChar	Effects on nicknames
i	makes you invisible to anybody that does not know the exact spelling of your nickname
o	IRC-operator status, can only be set by IRC-ops with OPER

s	receive server notices
v	gives a user a voice on a moderated channel

Here a few examples of the MODE command:

To give someone Op status: /mode #channelname +o nickname

Giving someone Op status means giving them control over the channel and the users on it. Give this out sparingly and to people you trust.

To op several people: /mode #channelname +ooo nick1 nick2 nick3

To de-op someone: /mode #channelname -o nickname

To ban someone: /mode #channelname +b nickname (or user address)

example: /mode #animals +b Jiminy
example: /mode #tree +b joe@bloggs.edu

To Unban someone: /mode #channelname -b nickname (or user address)

example: /mode #gb -b Ed

To Make a channel invite only: /mode #channelname +i

You must now **invite** a user for them to be able to join your channel.

There many more commands but this list should help you get started. To learn more about IRC commands you should download an IRC FAQ.

DDE Control

mIRC currently supports a limited **DDE** capability with enough DDE topics to make it useful.

All DDE transactions are **synchronous**, mIRC waits at most one second for a reply or acknowledgement to any dde poke or query it makes.

The /ddeserver command

As a server, mIRC's **default** service name is **mIRC**. The mIRC DDE Server defaults to **on** at startup unless it finds another mIRC or another application using it's current DDE Service name. You can use:

/ddeserver [[on [service name]] | off]

to manually turn on/off the dde server mode. If you have more than one mIRC acting as a DDE server then you should specify a different **service name** for each mIRC since if they all have the same service name you won't be able to predict which dde message will go to which server. For example:

/ddeserver on mIRCBot

If you try to use a service name which is **already** in use, mIRC will **warn** you but will still turn on the ddeserver.

For each of the following DDE topics, an example is given using mIRC's /dde command and $dde identifier (both are explained after this section) which should help you understand how they work.

DDE Topics

Topic: **CHANNELS**
Transaction Type: XTYP_REQUEST
Item: None
Data (Arguments): None
Returns: A single line of text listing the channels which you are currently on separated by spaces. eg. "#mirc #mircremote #irchelp"
Example: This should only be called by an application, not from within an mIRC alias.
Description: The line of text that this returns could end up being quite long, maybe several k's worth of text, so an application should be prepared to handle this.

Topic: **COMMAND**
Transaction Type: XTYP_POKE
Item: None
Data (Arguments): One line of text containing the command to be performed.
Returns:Nothing
Example: /dde mirc command "" /server irc.demon.co.uk
Description: This tells mIRC to execute whatever command you give it.

Topic: **CONNECT**
Transaction Type: XTYP_POKE
Item: None
Data (Arguments): one line of text in the form: address,port,channel,number
Returns:Nothing
Example: /dde mirc connect "" stork.doc.ic.ac.uk,6667,#mIRC,1
Description: This will tell mIRC to connect to the server stork.doc.ic.ac.uk, port 6667 and after it has connected it will automatically join channel #mIRC. If the number at the end is a 1 then the mIRC window will be activated, if it is a 0 it will not.

Topic: **CONNECTED**
Transaction Type: XTYP_REQUEST
Item: None
Data (Arguments): None
Returns:"connected" if you're connected to a server, "connecting" if you're in the process of connecting to a server, and "not connected" if you're not currently connected to a server.
Example: /say mIRC is currently $dde mirc connected "" to a server

Topic: **EXENAME**
Transaction Type: XTYP_REQUEST
Item: None
Data (Arguments): None
Returns: The full path and name of the mIRC EXE file. eg. "c:\mirc\mirc.exe"
Example: /say The mIRC exe path and name is $dde mirc exename ""
Description: This might be useful for applications that need to know the path and name of the exe file.

Topic: **INIFILE**
Transaction Type: XTYP_REQUEST
Item: None
Data (Arguments): None
Returns: The full path and name of the main INI file that is being used. eg. "c:\mirc\mirc.ini"
Example: /say mIRC's main ini file is $dde mirc inifile ""
Description: This might be useful for applications that need to take a look at the INI file for various bits of information.

Topic: **NICKNAME**
Transaction Type: XTYP_REQUEST
Item: None
Data (Arguments): None
Returns: The nickname currently being used.
Example: /say My mIRCbot is using the nickname $dde mirc nickname ""

Topic: **PORT**
Transaction Type: XTYP_REQUEST
Item: None
Data (Arguments): None
Returns: The port currently being used to connect to the irc server
Example: /say My mIRCbot is using port $dde mirc port ""

Topic: **SERVER**
Transaction Type: XTYP_REQUEST
Item: None
Data (Arguments): None
Returns: The server to which you were last or are currently connected. eg. "irc.server.co.uk"
Example: /say My mIRCbot's irc server is $dde mirc server ""

Topic: **SERVERPORT**
Transaction Type: XTYP_REQUEST
Item: None
Data (Arguments): None
Returns: The server and port to which you were last or are currently connected. eg. "irc.server.co.uk:7000"
Example: /say My mIRCbot's irc server is $dde mirc server ""

Topic: **USERS**

Transaction Type: XTYP_REQUEST
Item: Channel name, in the form #channel
Data (Arguments): None
Returns: A single line of text listing the users on the specified channel separated by spaces.
Example: This should definitely only be called by an application, not from within an mIRC alias.
Description: You can only request the names of users on a channel which mIRC has joined. The line of text it returns could be very, very long, so any application that uses this must be prepared to handle a single line of several k's worth of text.

Topic: **VERSION**
Transaction Type: XTYP_REQUEST
Item: None
Data (Arguments): None
Returns: mIRC version number eg. "mIRC16 5.41" or "mIRC32 5.41"
Example: /say mIRC's version number is $dde mirc version ""

To use the following /dde command and $dde identifer you must know the DDE specifications of the application with which you want to communicate.

The /dde command

/dde [-re] <service> <topic> <item> [data]

The /dde command defaults to sending an **XTYP_POKE** unless you specify the **-r** switch in which case an **XTYP_REQUEST** is sent, or if you specify the **-e** switch in which case an **XTYP_EXECUTE** is sent.

If you are sending an **XTYP_POKE** then all four arguments are mandatory.

If you are sending an **XTYP_REQUEST** then the first three arguments are mandatory. This is why you can see a **""** in the examples above, it acts as a filler and isn't actually used for anything. This filler is important because you might have the /dde or $dde mixed with other commands, and mIRC has to know the exact number of parameters when parsing.

If you are sending an **XTYP_EXECUTE** then only the first three arguments are required.

/dde monolog talk "" Greetings Earthling!

The $dde identifier

$dde <service> <topic> <item>

This sends an XTYP_REQUEST and retrieves any data returned by the specified service. All three arguments are mandatory (as explained in the above paragraph). Any retrieved data is inserted in the current position in an alias.

/say My other mIRC is $dde mirc connected "" to $dde mirc server ""

Note: If the value returned to $dde is too long for mIRC to handle, $dde now returns a value of $error.

The $isdde(name) identifier
Returns $true if the specified dde name is in use, $false otherwise.

IRC Servers

Once you've entered a few **basic** pieces of information about yourself in the **Setup** dialog, you can **select an IRC Server** from the servers list and **click on the Connect button**.

You'll know you've connected when the irc server shows you it's **Message Of The Day**, which contains information about that server, and you'll then be able to join a channel to start talking to people.

Basic information

The following information is required before you can connect to a server.

Real Name

You can enter your real name here, however most people usually enter a witty one-liner or comment.

Email Address

You must enter a full email address eg. khaled@mardam.demon.co.uk.

Nickname and Alternative

Your nickname is the name people will know you by when you join IRC. Remember that there are many thousands of people on IRC, so it's possible that someone might already be using the nickname you've chosen, so you should try to pick something unique. You can enter an alternative nickname as well in case someone is using your first nickname. If both nicknames are in use, mIRC inserts "/nick" into the edit box so that all you have to do is enter a new nickname and press enter.

Invisible Mode

If you turn on the invisible mode switch, people will not be able to find you on IRC unless they already know your nickname, or if you join a channel or talk to them privately.

IRC Servers

You can manage your list of IRC servers by using the add, edit, and delete buttons. You can enter the following information for each IRC Server:

Description

This can be any text you want and serves only as a description.

Address

This is the IRC server address eg. irc.dal.net

Port Number

This is usually 6667. If the server allows connections on different ports, you can enter them all separated by commas eg. 6667,6668,6669 and mIRC will pick one randomly each time it connects to the server.

Group Name

This allows you to group servers together when they are sorted with the **Sort** button.

Password

This is **rarely** required, so you should not have to enter anything here unless you have been specifically told to.

Note: If you're having problems connecting to an IRC Server, see the Connection Problems section.

Identd

mIRC can act as an **ident server** and sends the specified **User ID** and **System** as identification. This server will be more useful to some people than others. In general it is better to leave it active as some systems might refuse a connection if there is no reply to an ident request.

Enable Ident Server
Check/uncheck this to turn the identd server on or off

User ID
This can be your account or user name on your system. For most people this will be the User ID portion of their email address (the text before the @ sign). Valid characters for a User ID are: **. 0-9 A-Z _ - a-z**

System
This identifies your operating system. For all intents and purposes, replying with a value other than UNIX would not be very useful for most people.

Port
This should usually be 113.

Show Identd requests
This displays any identd requests sent to your identd server if it is turned on.

Enable only when connecting
This turns the identd server on **only** when you're connecting to an IRC server. The moment an identd request is received and replied to, or the moment you connect to the IRC server and see the MOTD, the identd server is turned off.

Note: This server will reply to **all** ident queries sent to the specified port ie. it is not limited to replying to an IRC server for mIRC. If you have turned on the identd server and it isn't replying to identd queries then it is probable that the type of internet account you have is preventing mIRC from replying, or that another program is running which has control of the identd port.

Finger

This allows you to finger a persons address to find out more information about them.

If a user has a personal email/machine address, this will only work if they are running a finger server. Most other addresses eg. school, government, etc. can usually be fingered sucessfully, though they will not necessarily give you any useful information. For some users this can be a way of checking if they have mail.

You can also type **/finger <nick/address>** at the command line.

If there's a **.** in the parameter you provide then it assumed to be an address and is immediately fingered, otherwise it is assumed to be a nickname, so mIRC looks up the users address with a /userhost and then fingers that address.

On a related note, check out the Finger Server.

Online Timer

The **Online timer** is displayed in the titlebar of the status window next to your connection information.

You can choose to have the timer **reset to zero** every time you connect, so that you can tell how long you are on for that session, or if you select the second option, the timer will be **cumulative** and will show the **total** amount of time you have used IRC since the last reset.

Remote

The **remote** allows you to create scripts that react to IRC Server events, such as when a user joins a channel or sends you a message. This tool is the most complex part of mIRC and to use it you must already know how to use IRC Commands, how to create Aliases, and how to use Variables and Identifiers.

The remote consists of **three** distinct sections:

The **Users** section, where user addresses with assigned access levels are listed. Each User in your Users section can be assigned one or more levels. These access levels dictate which events a user will be able to access.

The **Variables** section, where the currently active variables are listed.

The **Scripts** section, where the scripts that you create are listed. You can load **multiple** scripts which work **independently** of each other. This means that a single IRC Server event can trigger events in **one or more** scripts. Scripts consist of **events** which can only be triggered by **users** who have the **required** access levels. You can also place aliases in your scripts by using the **alias** prefix, and menus in your scripts by using the **menu** prefix.

Since Access Levels play an important part in the way scripts work, you should read about them carefully before proceeding. You should also take a look at remote commands, identifiers, and the internal address list, and the section on how to Halt default text being displayed if you want to display your own custom messages for events.

All of the following **Events** use the same general format except for the **ctcp** and **raw** events. The sections below provide you with **information** on each event as well as **examples** and **tips** on how to use them.

Action	Error	Nick	ServerOp
Ban	FileRcvd	NoSound	SNotice
Chat	FileSent	Notice	Start
Close	GetFail	Notify	Text
Connect	Help	Op	Topic
Ctcp	Input	Open	UnBan
CtcpReply	Invite	Part	Unotify
DccServer	Join	Quit	UserMode
DeHelp	Kick	Raw	Voice
DeOp	Load	SendFail	Wallops
DeVoice	MidiEnd	Serv	WaveEnd
Dns	Mode	ServerMode	

Here is an example script that shows you how you can place aliases, popups, and events in a single file making it easier to distribute your scripts to other people.

Note: You should never load or use a script that you don't understand.

Remote Commands

The following commands are used to modify settings in scripts and in the remote section.

General Commands

/ctcps [on|off]
This switches processing of ctcp events on/off.

/events [on|off]
This switches processing of named events on/off.

/dlevel <level>
This changes the default user level to the specified level.

/raw [on|off]
This switches processing of numeric events on/off.

/remote [on|off]
This switches processing of all scripts on/off.

Group Commands

These commands allow to turn groups on and off in remote scripts. You can find out about groups in the Access Levels section.

/enable <group1 group2 ... groupN>
This enables the specified groups in all scripts.

```
/enable #one #two #three
```

You can also specify a wildcard for to enable all matching groups:

```
/enable #help*
```

/disable <group1 group2 ... groupN>
This disables the specified groups in all scripts.

```
/disable #one #two #three
```

You can also specify a wildcard for to disable all matching groups:

```
/disable #help*
```

/groups [-e|d]
This displays a list of either all, enabled, or disabled groups in your scripts.

User and Level Commands

You can use the following three commands to add and remove users to the users list, as well as to add and remove levels from existing users.

/auser [-a] <levels> <nick|address>
This adds the specified nick/address exactly as it is given to the users list with the specified levels. If you specify [-a], then if the user already exists, the specified levels are added to the current levels the user

has. Remember, if the first level is not preceeded by an equal sign then it is a general access level.

/auser 1,2,3 Nick

This adds this nick with these access levels to the user list (replacing an existing user of the same name).

/auser -a 1,2,3 Nick

This adds the specified levels to this user. If the user doesnt exist, it is created.

/auser -a =1,2,3 Nick

This looks like the above command, however the =1 is very important. The =1 means that the initial general access level is **not** replaced. If you had used 1 then the initial access level would be replaced.

/flush [levels]
This clears the remote user list of nickname definitions that are no longer valid.

/flush 1,2,3

For each nick in the remote user list that matches the specified levels mIRC checks to see if that nick is on any of the channels that you are currently on. If not, the nick definition is removed from the remote user list. If you do not specify levels then mIRC clears all nicks from the remote user list that don't exist on channels you are on.

/guser [-a] <levels> <nick> [type]
This works the same as the /auser command except that it looks up address of the specified nick and adds it to the user list. It does this by doing a /userhost on the given nickname, and returning an address in the format specified by type. If no type is specified then a default address format is selected.

/ruser [levels] <nick | address> [type]
If used without specifying levels, this removes the specified user from the user list. If you specify levels then these levels are removed from the current access levels of this user. If all a user's levels have been removed, the user is removed. If you specify a type then the users address is looked up with a /userhost and any users in the users list matching this address are removed.

/ruser Nick
/ruser 1,2,3 Nick
/ruser 1,2,3 Nick 1

If you use /ruser Nick! (with an exclamation mark at the end), it removes all users with an address beginning with Nick!.

/rlevel [-r] <levels>
This removes all users from the remote users list with the specified general access level.

/rlevel 20
/rlevel =10

If the -r option is specified, then this applies to all the access levels for a user (not just the first general access level). Any matching levels are removed. If a user has no more levels left then the user is also removed.

/rlevel -r 1,5,7,8

/ulist [<|>] <level>
This lists users which have the specified access levels.

/ulist <10	lists users with access levels less than or equal to 10
/ulist >5	lists users with access levels larger than or equal to 5
/ulist 4	lists users with access level 4

Note: The /guser and /ruser commands do a /userhost on a nick to find the nick's address, thus they are delayed commands since they need to wait for a reply from the server. mIRC tries to get around this delay by maintaining it's own Internal Address List which will speed things up in certain situations.

Access Levels

Access levels are assigned both to a user and to an event and serve to limit a user's access to only certain events.

The **default access level** is 1 for users that are not listed in the Users list. All users can access level 1 events. The higher a user's access level is, the more events that user can access. You can change the default user level to allow unlisted users to access more commands.

Users

In the Users section you can specify a **list of users** and their access levels using the format:

```
<level1,level2,...,levelN>:<useraddress>
```

```
3,5,6:goat!khaled@mardam.demon.co.uk
```

The **first** level is a general access level, which means that the user can access all levels equal to or less than 3. All the **other** levels are levels that an event must specifically have to allow a user to access it.

If you want to **force** the first access level to be a specific level instead of a general access level, you can prefix it with an equal sign.

```
=3,5,6:goat!khaled@mardam.demon.co.uk
```

Now this user has access specifically to level 3, 5, and 6 event and to no other events.

Events

In general the format of an event is:

```
<prefix> <level>:<event>:<commands>
```

```
ctcp 1:HELP:/msg $nick No help is available for level 1 users
```

The above ctcp command can be accessed by all users because it is a level 1 command. So if a user with nickname goat sends you a /ctcp yournick HELP, your script will send them the above reply.

Only the **highest level** matching event is triggered for a user.

Named Levels

You can also used **named levels** which work the same way as a specific level but are easier to understand and read than a number.

```
friend:goat!khaled@mardam.demon.co.uk
```

```
on @friend:JOIN:#mIRC:/mode $chan +o $nick
```

This treats the word **friend** as a specific access level and matches the user with the event, and because the user is your friend, you give him ops.

Limiting Access

You can limit access to an event by specifying a special prefix which determines how an event is processed or triggered by users.

The + prefix

You can limit an event to users with a specific access level by using the + prefix.

10:goat!khaled@mardam.demon.co.uk

ctcp +5:HELP:/msg $nick You have accessed a level +5 event

The above user can't access this ctcp event even though he has an access level higher than 5 because the event is limited only to level 5 users.

The * prefix
You can allow any user to trigger an event regardless of their access level by using the * prefix.

on *:TEXT:help:#:/msg $nick you have accesed a * level event

The ! prefix
You can prevent an event from being triggered if it was initiated by you by using the ! prefix.

ctcp !2:HELP:/msg $nick You have accessed a level 2 event

You would be unable to access the above event regardless of your access level.

The @ prefix
You can limit events to being executed only when you have Ops on a channel by using the @ prefix.

10:goat!khaled@mardam.demon.co.uk

on @2:JOIN:#mIRC:/mode $chan +o $nick

When the above user joins channel #mIRC and you have Ops on #mIRC, the associated /mode command will be executed, in this case giving the user Ops. If you don't have Ops, the event will not trigger.

The & prefix
You can prevent an event from being triggered if a previous script used **/halt** or **/haltdef** to halt the display of default text for an event by using the & prefix.

on &1:text:*:?:/echo this event won't trigger if $halted is true

The = suffix
You can prevent users with higher access levels from accessing **all** lower access level events by using the = suffix.

10:goat!khaled@mardam.demon.co.uk

ctcp 2:HELP:/msg $nick You have accessed a level 2 event
ctcp 5:HELP:=

The above user can't access any of these events because the level 5 event prevents him from accessing all HELP events with access levels lower than 5.

The ! suffix
You can prevent commands for a certain event level from being processed by using the ! suffix.

ctcp 5:PING:echo PING!
ctcp 5:*:!

The ! at the end of the line tells the remote to halt any further processing of level 5 commands.

Creating Groups

You can create separate groups in scripts by using the # hash prefix.

```
#group1 on
...
[ list of events ]
...
#group1 end
```

You can use the /enable and /disable commands to enable or disable groups. A group that is disabled will be ignored when processing events. A disabled group looks like this:

```
#group1 off
...
[ list of events ]
...
#group1 end
```

You cannot have groups within groups.

Ordering of definitions

Many of the prefixes and controls are sensitive to numerical order of the definitions. The safest thing is to order your definitions starting with the lowest access levels first and increasing numerically down the list, this makes it easier to keep track of which events should trigger first.

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-.

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

$chan
Returns the name of the channel where an event took place. Only certain events occurr specifically on a channel, for all other non-channel events $chan will have no value.

$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!userid@host.domain.

$group(N/#)
Returns the 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

$ial
Returns $true if the Interal Address List is turned on, otherwise returns $false.

$maddress
Returns the address that was matched for the triggered event.

$nick
Returns the nickname of the user associated with an event.

$numeric
Returns the numeric for the matching numeric event.

$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.

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

$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.

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.domain.

DCC Resume Protocol

The current protocol is very simple.

User1 is sending the file.
User2 is receiving the file.

To initiate a DCC Send, User1 sends:

PRIVMSG User2 :DCC SEND filename ipaddress port filesize

Normally, if User2 accepts the DCC Send request, User2 connects to the address and port number given by User1 and the file transfer begins.

If User2 chooses to resume a file transfer of an existing file, the following negotiation takes place:

User2 sends:

PRIVMSG User1 :DCC RESUME filename port position

filename = the filename sent by User1.
port = the port number sent by User1.
position = the current size of the file that User2 has.

User1 then responds:

PRIVMSG User2 :DCC ACCEPT filename port position

This is simply replying with the same information that User2 sent as acknowledgement.

At this point User2 connects to User1 address and port and the transfer begins from the specified position.

Note: the newer versions of mIRC actually ignore the filename as it is redundant since the port uniquely identifies the connection. However, to remain compatible with older mIRC's, mIRC still sends a filename as **file.ext** in both RESUME and ACCEPT.

on CHAT/SERV

The **on CHAT** and **on SERV** events trigger when a message is sent to a dcc chat or dcc fserve window.

Format:	on <level>:CHAT:<matchtext>:<commands>
Example:	on 1:CHAT:*help*:/msg $nick what's the problem?

For an explanation of **matchtext** see the on TEXT event.

Examples

on 1:CHAT:boo!:/msg =$nick Boo back at ya!

This triggers when someone in a dcc chat window says boo! The equal sign in **=$nick** is required to send the reply as a dcc chat message. If no equal sign is used, the message is sent as a private irc server message.

on 1:SERV:bye:/msg =$nick Thanks for using my fileserver, bye!

This triggers when a user in a dcc fileserver session says the word bye to quit the fileserver. You can also refer to the **$cd** identifier to reference the current directory a fileserve is in.

Note: These events react to **all** users level 1 and above because of the way dcc chat works.

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 **location** where this event occurrs 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 %matchtext 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.

on JOIN/PART

The **on JOIN** and **on PART** events trigger when a user joins or parts a channel.

Format:	on <level>:JOIN:<#[,#]>:<commands>
Example:	on 1:JOIN:#mirc,#irchelp:/msg $nick hiya!

Examples

on 1:JOIN:#:/msg $chan Welcome $nick!

This triggers when any user joins any channel which you are on.

on 5:PART:#mIRC,#newbies:/describe $chan waves bye-bye to $nick *sniff*

This triggers when a user with access level 5 leaves channels #mIRC or #newbies.

on KICK

The **on KICK** event triggers when a user is kicked from a channel.

Format:	on <level>:KICK:<#[,#]>:<commands>
Example:	on 1:KICK:#mirc,#irchelp:/msg $nick Oops! ;)

Examples

```
on 5:KICK:#:/invite $knick $chan | /msg $nick Hey, $knick is my friend!
```

This triggers when when a user who has access level 5 is kicked out of any channel. **$knick** refers to the nickname of the user who was kicked.

Comparing levels

You can **compare the levels** of the kicker and the kicked users by prefixing the line with <,>,<=,=>,<>, or =, in the following way:

```
on >=2:KICK:#mIRC:/msg $chan $nick kicked $knick (legal)
on 1:KICK:#mIRC:/msg $chan $nick kicked $knick (illegal)
```

In this situation, if the kickers level is larger than or equal to the kicked users level, then it is a legal kick. Otherwise, it defaults to the second on KICK event which indicates that it is an illegal kick. Remember, this is comparing the **kickers and kicked users levels** and has nothing to do with with the level "2" in the definition.

Note: This event only works on a nickname because the IRC server only sends the nickname of the user being kicked and not an address.

on OP/DEOP

The **on OP** and **on DEOP** events trigger when a user on a channel is opped or deopped.

Format:	on <level>:OP:<#[,#]>:<commands>
Example:	on 1:OP:#mirc,#irchelp:/msg $nick Please don't abuse your Op status

The **on VOICE/DEVOICE** and **on HELP/DEHELP** events use the same format and trigger when a user is voided/devoiced or helped/dehelped respectively.

The **on SERVEROP** event also uses the same format and triggers when a user is opped by a **server**.

Examples

on 9:OP:#:/mode $chan -o $opnick
on 9:VOICE:#:/mode $chan -v $vnick
on 9:HELP:#:/mode $chan -h $hnick

This triggers when a user with access level 9 is opped/voiced/helped on any channel. **$opnick** refers to the nickname of the person being opped/deopped, **$vnick** the person being voiced/devoiced, and **$hnick** the person being helped/dehelped.

on 1:DEOP:#beginner:/mode $chan +o $opnick

This triggers when any Op is deopped on channel #beginner.

on 1:SERVEROP:#:/mode $chan -o $opnick

This triggers when an irc server ops a user on any channel. You immediately deop them.

Comparing levels

You can **compare the levels** of the opper and the opped by prefixing the line with <,>,<=,=>,<>, or =, in the following way:

on >=1:DEOP:#mIRC:/msg $chan $nick deopped $opnick (legal)
on 1:DEOP:#mIRC:/msg $chan $nick deopped $opnick (illegal)

In this situation, if the deoppers level is larger than or equal to the deopped users level, then it is a legal deop. Otherwise, it defaults to the second line which indicates that it is an illegal deop. Remember, this is comparing the oppers and opped users levels and has nothing to do with with the level 2 in the definition.

Note: These events only work on nicknames because the IRC server only sends the nickname of the user being affected and not their address.

on CONNECT

The **on CONNECT** event triggers when mIRC connects to an IRC Server right after the MOTD is displayed.

Format: on <level>:CONNECT:<commands>
Example: on 1:CONNECT:/join #new2irc

The **on DISCONNECT** event uses exactly the same format as above and triggers when you quit or are disconnected from an IRC Server.

Examples

on 1:CONNECT:/echo Connected to $server at $time with nickname $nick

This triggers after mIRC connects to a server.

on 1:DISCONNECT:/echo Disconnected from $server at $time with nickname $nick

This triggers when mIRC is disconnected from a server.

Note: The CONNECT event is essentially the same as the Perform section.

mIRC Commands

The following commands are mostly unique to mIRC, though some are only modifications or extensions of standard IRC commands.

/abook [nickname]
Pops up the address book and shows information for the specified nickname if it exists.

/alias [filename] <aliasname> <commands>
Adds, removes, replaces aliases; it is limited to single line aliases and will not affect mutiple line definitions. To add a new alias, you can use:

/alias /moo /me moos!

This will add the /moo alias to the top of the aliases list. To remove an existing aliases:

/alias /moo

To add an alias to a specific alias file, you would use:

/alias moo.txt /moo /me moos!

If you don't specify a filename, it defaults to using the first filename in which the alias exists, or if it doesn't exist then it uses the first loaded aliases file.

/amsg <message>
This and the **/ame** command send the specifed message or action to all channels which you are currently on.

/auser
This and remote-related commands are listed in the remote section.

/auto [-r] <nickname/address> [#channel1,#channel2,...] [type]
Adds someone to the auto-op list, see the Control section for a full description.

/background [-amsgdluhcfnrtpx] [window] [filename]
Changes the background picture setting for a window. This can also be changed via a windows System Menu.

-a = active window
-m = main mIRC window
-s = status window
-g = finger window
-d = dedicated query window

-cfnrtp = center, fill, normal, stretch, tile, photo

-l = toolbar
-u = toolbar buttons
-h = switchbar

You can right-click in the toolbar/switchbar to pop up a menu for changing these settings. Toolbar buttons can use RGB Colour **192,192,192** for transparency, the BMP must be of the same form as that in mIRC resources. It should be a 16 or 256 colour BMP.

-x = no background picture

Note: The **window** name should only be specified if none of the window switches are specified. The **filename** does not need to be specified if you are only changing the display method.

/ban [-ruN] [#channel] <nickname|address> [type]
Bans someone from the current channel using their address. To do this, it first does a /userhost on the user, which gives it the user's address, and then it does a /mode # +b <user address>.

If you specify the **-uN** option then mIRC pauses N seconds before removing the ban.

If you specify the **-r** switch then /ban **removes** the ban of the specified type for that nickname, eg. /ban -r nick 2

If you do not specify a ban type, then mIRC uses the whole nick!*user@host.domain to do the ban. If you are banning an IP address then a wild card replaces the last number of the IP address. If you are on the channel then the #channel specification is not necessary.

If you specify a wildcard address it is used as-is, if you specify a full address then the type mask is applied to it.

For a list of ban types see the **$mask** identifier in the Identifiers section.

Note: This command uses the Internal Address List maintained by mIRC.

/beep <number> <delay>
Beeps a number of times with a delay.

/bset, /bread, and /bwrite
These binary commands allow you to handle binary information.

/channel
Pops up the channel central window (only works in a channel)

/clear [-sghlc] [windowname]
Clears the entire scrollback buffer of the current window. If you specify a window name, that window's buffer will be cleared.

The **-s** switch clears the status window.
The **-g** switch clears the finger window.
The **-l** switch clears the side-listbox in a custom window.
The **-c** switch clears the click history in a picture window.
The **-h** switch clears the editbox command history for a window.

/clipboard <text>
Copies the specified text to the clipboard.

/close [-icfgms@] [nick1] ... [nickN]
Closes **all** windows of the specified type and nicknames. If **no** nicknames are given, **all** windows of the specified type are closed. The type of window is denoted by c for chat, f for fserve, g for get, i for inactive dcc windows, m for message (query), s for send, and @ for custom windows.

If you wanted to close all **chat and fserve** windows for user **Nerp**:

/close -cf Nerp

/closemsg [windowname]
Closes the specified message window.

/copy -o <filename> <filename>
Copies a file to another filename or directory. You can also use wildcards for the source filename, and a directory name for the destination.

The **-o** switch overwrites a file if it exists.

/creq [+m|-m] [ask | auto | ignore]
This is the command line equivalent of setting the DCC Chat request radio buttons in the dcc options dialog (see /sreq below). The +m|-m switch turns the minimize setting on|off.

/ctcpreply <nick> <ctcp> [message]
Sends a reply to a ctcp query.

```
/ctcpreply goat HELP no help available.
```

/dcc send <nickname> <filename>
Initiates a sending a file to another user. It can take multiple file names, the format is:

```
/dcc send <nickname> <file1> <file2> <file3> ... <fileN>
```

This will initiate multiple dcc send sessions to the specified user. If you specify a wildcard filename, then the DCC Send dialog will display files matching the wildcard, eg.:

```
/dcc send moogoat *.txt
```

/dde [-re] <service> <topic> <item> [data]
Sends information to other applications. See the DDE Control section for a full explanation.

/disconnect
Forces a disconnect from a server. This is different from the /quit command which sends a quit message to the server and waits for the server to disconnect you.

/dns [-c] [nick|address]
Resolves an address. If mIRC sees a "." in the name you specify it assumes it's an address and tries to resolve it. Otherwise it assumes it's a nickname and does a /userhost to find the users address and then resolves it. If you give it an IP address, it looks up the host name.

You can queue multiple /dns requests, and you can view the current queue by using /dns with no parameters.

The **-c** switch clears all currently queued DNS requests, except for the one currently in progress.

Note: because of the way the DNS lookup works, any DNS related functions currently in progress eg. connecting to a server, must be resolved before subsequent requests. This means that if a prior DNS is having problems resolving, subsequent DNSs have to wait until it times out before they can be resolved.

/dqwindow [on|off|show|hide|min]
Manipulates the dedicated query window.

/echo [colour] [-dehiNtsa] [#channel|[=]nick] <text>
Prints text in the specified window using the specified colour (0 to 15).

/echo 3 #mIRC Testing

would print "Testing" in the colour green in channel window #mIRC, assuming it's already open.

If a channel/nickname isn't specified, the **-s** switch echos to the status window, the **-d** switch echos to the dedicated message window, and the **-a** switch echos to the currently active window.

The **-e** switch encloses the line in line separators.
The **-iN** switch indents the wrapped line by N characters.
The **-h** switch forces lines to hard-wrap so resizing the window doesn't change the line.
The **-t** switch prefixes the line with a timestamp if global time stamping is on or timestamping is on for that window.

Note: This text is only displayed in your own window, it isn't sent to the server so no one else can see it.

/editbox [-sap|[=]window] <text>
Fills the editbox of the current window with the specified text.

The **-s** switch specifies the Status window.
The **-a** switch specified the Active window.
The **-p** switch indicates that a space should be appended to text.

To specify a dcc chat window, prefix the nickname with an **=** equal sign.

/exit
Close down mIRC and exit.

/filter [-sgdfwxnp] <infile> <outfile> <matchtext>
This command scans lines of text in a window or file and if any of them contain matchtext, they are written out to another window or file which you can then use.

The **infile** can be a filename or a window name (custom or normal). The **outfile** can be a filename or a custom window name. You should specify the **-fw** switches if the names are ambiguous eg.

/filter -ff in.txt out.txt *mirc*

This indicates that both are filenames, and:

/filter -wf #in.txt #out.txt *help*

indicates that the first is actually a window name, and the second is a filename.

The **-x** switch excludes matching lines.
The **-n** switch prefixes lines with a line number.
The **-s** switch makes the status window the infile.
The **-g** switch makes the finger window the infile.
The **-d** switch makes the dedicated message window the infile.
The **-p** switch wraps the text output in a custom window.

This command also fills the **$filtered** identifier with the number of matches found, if any.

/finger <nick/address>
If you specify an address then the address is immediately fingered. If you specify a nickname then the users address is looked up using a /userhost and then it is fingered.

/flash [-wb] <text>

This flashes the mIRC window/icon with the specified text in the titlebar but **only** if mIRC is not the active application.

The **-b** switch makes mIRC beep every second.
The **-w** switch makes mIRC play the Flash sound specified in the Event Beeps section.

/flood [on|off|clear] <bytes> <maxlines> <maxmessages> <ignoretime>
Turns flood protection on and off. See the Flood section for a full description.

/flushini <filename>
Flushes the specified INI file to the hard disk. INI files are cached in memory, so you may want to do this to make sure that your INI is updated properly.

/font [-asgb|window] <fontsize> <fontname>
This allows you to change the font for the current window. If no parameters are specified, the font dialog pops up, otherwise the specified parameters are used. You can make the font bold by using the **-b** switch.

/fsend [on | off]
Turns DCC fast send on or off.

/fserve
Initiates a fileserver session to another user. See the File Server section.

/goto <name>
This is used to jump to different points in a script. See the aliases section.

/halt
Halts a script and prevents any further processing. You can use this in remote commands to prevent mIRC from replying to normal ctcp messages, or in aliases to halt an alias, and any calling aliases, completely.

/help [keyword]
Brings up the Basic IRC Commands section in the mIRC help file with the specified keyword.

/ial [on|off]
Turns the Internal Address List on and off.

/identd [on|off] [userid]
Turns identd server on and off, and changes to a new userid if it is specified.

/ignore [-rpcntixu#] <nickname/address> [type]
Allows you to ignore messages from the specified nick, see the Control section for a full description.

/join [-inx] <#channel>
This is a standard IRC command for joining a channel.

The **-i** switch makes you join the channel to which you were last invited.
The **-n** and **-x** switches minimize/maximize the channel window when you join it.

/linesep [-s|window]
Prints the line separator selected in the Options dialog Extras section in the specified window.

/links
Retrieves the servers to which your current server is linked.

/load <-a|-pscqnm|-ruvs> <filename>
Loads the specified alias, popup, or script.

/load -a aliases.ini	loads an aliases file
/load -pc status.ini	loads a channel popup
/load -pn status.ini	loads a nickname list popup
/load -ru users.ini	loads a users file
/load -rv vars.ini	loads a variables file
/load -rs script.ini	loads a scripts file

If you try to load a file that is already loaded, it's contents are updated and it's position in the alias/script processing order is maintained.

You can also use the **/reload** command with the same parameters to reload a file without triggering the on start/load events in the script being loaded.

Note: You can only load one section at a time.

/loadbuf [lines] [-psglecN] <window> <filename>
Loads the specified number of lines from the end of the file of filename into the specified window.

/loadbuf 20 @test info.txt

This loads the last 20 lines of info.txt into custom window @test.

/loadbuf 10-40 @test info.txt

This loads lines 10 to 40 of info.txt into custom window @test.

The **-p** switch forces lines of text to wrap when added to the window.
The **-s** and **-g** switches apply the command to the status and finger windows respectively.
The **-l** switch applies the command to the side-listbox in a custom window.
The **-e** switch evaluates variables and identifiers in the line being read.
The **-cN** switch specifies the default background colour for lines.

/log <on|off> <window> [-f filename]
Turns logging on and off for a window, if you specify a filename the logs file dialog is not popped up.

/mkdir <dirname>
Creates the specified directory.

/nick <nickname> [alternate]
Changes your current nicknames.

/notify <nickname> [note]
Add a nickname with a note to your notify list. If you don't specify a nickname, a notify request is sent to the irc server to update your notify list.

/omsg [#channel] <message>
This and the **/onotice** command sends the specified message to all channel ops on a channel. You must be a channel operator to use these commands. If the #channel isn't specified, then the current channel is used.

/partall [message]

Parts all of the channels you are currently on. On certain IRC Servers, you can also specify a message.

/perform [on|off]
Turns the "Perform these commands" section of the Perform dialog on and off.

/play [-scp q# m# rl# t#] [channel/nick/stop] <filename> [delay]
This is a powerful command that allows you to send text files, or parts of them, to a user or a channel.

The **delay** is in milliseconds. If you play files too quickly to a server you will probably be disconnected for flooding. The default setting is 1000 ie. 1 second. Empty lines between text are treated as a delay.

/play c:\text\mypoem.txt 1500

The **-s** switch allows you to play commands to the status window while offline. If you do not specify the -s switch then you must be connected to a server to use the /play command.

The **-c** switch forces mIRC to interpret lines as actual commands instead of plain text.

The /play command **queues** requests by users; it does this because if it tried to play all requests at the same time you would probably be disconnected from a server for flooding.

The **-p** switch indicates that this is a priority play request and should be placed at the head of the queue for immediate playing. The current play request will be paused and will resume once this play request is finished.

The **-q#** switch specifies the maximum number of requests that can be queued. If the queue length is already larger than or equal to the specified number then the play request is ignored.

/play -q5 c:\text\info.txt 1000

The **-m#** switch limits the number of requests that can be queued by a specific user/channel. If the user/channel already has or exceeds the specified number of requests queued then the play request is ignored.

/play -m1 info.txt 1000

The above line limits each user to a maximum of one request at a time and ignores all of their other requests.

Note: The **-q#** and **-m#** switches only apply to a /play initiated via a remote definition, not by you.

To combine the above switches you would do:

/play -cpq5m1 info.txt 1000

The **-r** switch forces a single line to be chosen randomly from a file and played.

/play -r action.txt 1500

The **-l#** switch forces the specified line-number to be read from a file and played.

/play -l25 witty.txt 1500

For both **-r** and **-l#** the first line in the file can be a single number specifying the number of lines in the file, this speeds up the process of reading the file.

The **-t** switch forces mIRC to look up the specified topic in the file and play all lines under that topic. For example:

```
/play -thelp1 c:\help.txt
```

In the help.txt file you would have:

```
[help1]
line1
line2
line3

[help2]
...
```

mIRC will play everything after [help1] and stop when it reaches the next topic header or the end of the file.

You can also use the **$pnick** identifier in commands which identifies the nick/channel to which you are playing.

To **stop** the playing of a text file and clear the queue you can use /play stop.

/pop <delay> [#channel] <nickname>
Performs a delayed Op on a nickname. The purpose of this command is to prevent a channel window filling up with Op mode changes whenever several users have the same nickname in their auto-op section.

mIRC will pause around <delay> seconds before performing the Op. If <delay> is zero, it does an immediate Op. Before performing the Op it checks if the user is already Opped. If you do not specify the #channel, the current channel is assumed.

/raw [-q] <command>
Sends any parameters you supply directly to the server. You **must** know the correct RAW format of the command you are sending. Useful for sending commands which mIRC hasn't implemented yet. The -q switch makes the raw work quietly without printing what it's sending. This command does the same thing as **/quote** in other IRC clients.

```
/raw PRIVMSG nickname :Hellooo there!
```

/remini <inifile> <section> [item]
Deletes whole sections or single items in an INI file.

```
/remini my.ini DDE ServerStatus
```

This would delete the ServerStatus item, and:

```
/remini my.ini DDE
```

Would delete the DDE section.

See the /writeini command below for a related example.

Warning: Do not use this command to modify any of the INI files currently being used by mIRC.

/remove <filename>

Deletes the specified file.

/rename <filename> <newfilename>
Renames a file, can also be used to move a file from one directory to another.

/resetidle [seconds]
This resets the $idle identifer to zero or to the number of seconds you specify.

/return [value]
Halts a currently executing script and allows the calling routine to continue processing. You can also optionally return a **value** which will be stored in the **$result** identifier. The $result can then be used in the calling routine.

/rmdir <dirname>
Deletes the specified directory.

Note: If the directory contains files, it cannot be deleted.

/run [-n] <filename> [parameters]
Runs the specified program with parameters.

/run c:\net\ftp.exe sunsite.unc.edu

This runs the ftp program with the parameter sunsite.unc.edu.

/run notepad.exe $?

This asks you for a parameter and runs notepad using the parameter as the filename.

If you specify a **non-executable** file, mIRC tries to find the program associated with that file and then runs it.

/run info.txt

You can specify the **-n** switch to minimize the window of the program being run.

/save <-pscqnm|-ruv> <filename>
Saves the specified popup or remote users/variables file.

/save -ps status.ini	saves the status popup to status.ini
/save -pn nick.ini	saves the nickname list popup to nick.ini
/save -ru users.ini	saves the user list to users.ini
/save -rv vars.ini	saves the variables list to vars.ini

Note: You can only save one section at a time.

/savebuf [-sga] [lines] <window> <filename>
Saves the specified number of lines from the end of the buffer of the specified window into the specified filename.

/savebuf 20 @test info.txt

This saves the last 20 lines in custom window @test to info.txt.

/savebuf 10-40 @test info.txt

This saves lines 10 to 40 in custom window @test to info.txt.

The **-s** switch saves the status window buffer, the **-g** switch saves the finger window buffer, and the **-a** switch makes it append the text to the end of a file instead of overwriting it.

/saveini
Updates all mIRC-related INI files with the current settings.

/say <message>
This lets you define an alias that writes directly to a channel as if you were saying something. So "/say Hello there" would be the same as just typing "Hello there". This is useful in an alias when you want to ask the same question (or send the same information) again and again.

/info /say Please note that the games server is currently down and will be offline for a few hours...

Note: You can't use this command in the remote section. Use /msg #channel <message> instead.

/server <server/groupname> [port] [password]
Connects you to a server, first disconnecting you from the current server.

/server irc.server.co.uk 6667 mypassword

If you type /server with no parameters, mIRC will connect to the last server you used. If you use the server command while still connected, you will be disconnected with your normal quit message and will then connect to the specified server.

You can also use /server N which connects to the Nth server in the server list in the setup dialog.

You can also use /server groupname which will cycle through all the servers in the server list which have that group name until it connects to one of them.

/set and /unset
Defines and undefines your own Variables.

/showmirc -nrstx
Manipulates the display of the main mIRC window, where -n = minimize, -r = restore, -s = show, -t = tray, and -x = maximize.

/sline [-a|r] <#channel> <N|nick>
Selects or deselects lines in a channel nickname listbox. It can select either the Nth nickname in a listbox, or a specified nickname.

If you do not specify any switches, any existing selections in the listbox are cleared. If you specify the **-a** switch then the specified is selected without affecting the selection states of other lines. If you specify the **-r** switch then the specified item is deselected.

/socket commands
Socket commands and identifiers are described in the Sockets section.

/sound
Sends a sound request to another user. See the Sound Requests dialog for more information.

/speak <text>
Sends the specified text to Monologue (or Text Assist) which is a program that speaks whatever text is sent to it.

/splay [-qwm] <filename>
Plays the specified .wav or .mid file. If you do not specify a directory, the sounds directory from the Sound Requests options dialog is used. You can also use **/splay stop** to stop a file that is playing.

The **-q** switch allows you to queue .wav and .mid sounds for playing.

The **-wm** switches are used in conjunction with /splay -wm stop, and allow you to specify whether you want to stop .wavs, .mids, or both playing.

/sreq [+m|-m] [ask | auto | ignore]
This is the command line equivalent of setting the DCC Send request radio buttons in the dcc options dialog (see /creq above). The +m|-m switch turns the minimize setting on|off.

/strip [+-burc]
Turns control code stripping options in Options dialog on/off.

/strip +bur-c

would turn bold, underline, reverse stripping **on**, and turn colour stripping **off**.

/timer[N/name] [-eom] [time] <repetitions> <interval> <command>
Activates the specified timer to perform the specified command at a specified interval, and optionally at a specified time.

If you are not connected to a server and you start a timer, it defaults to being an offline timer which means it will continue to run whether you are connected to a server or not.

If you are connected to a server and you start a timer, it defaults to being an online timer, which means that if you disconnect from the server, it will be turned off. You can specify the **-o** switch to force it to be an offline timer.

/timer1 0 20 /ame is AWAY!

Timer1 will repeat an all channel action every 20 seconds until you stop the timer.

If you specify a delay of 0 seconds, the timer will trigger immediately after the calling script ends.

/timer5 10 60 /msg #games For more info on the latest games do /msg GaMeBoT info

Timer5 will repeat this message to channel #games every sixty seconds and stop after 10 times.

/timer9 14:30 1 1 /say It's now 2:30pm

This will wait until 2:30pm and will then announce the time once and stop.

To see a list of active timers type /timers. To see the setting for timer1 type /timer1. To deactivate timer1 type /timer1 off. To deactivate all timers type /timers off. If you are activating a new timer you do not need to specify the timer number, just use:

/timer 10 20 /ame I'm not here!

And mIRC will allocate the first free timer it finds to this command.

If you specify the **-m** switch, this indicates that the interval delay is in milliseconds.

If you specify the **-e** switch, this executes the command associated with the specified timer name.

Note: millisecond timers can slow mIRC down significantly because each timer can trigger many times per second, so they should not be used unless they are necessary.

The **$ltimer** identifier returns the number of the timer that was just started by the /timer command.

Instead of using a number you can also specify a **name** for a timer.

/timershow 0 10 echo -a $nick $server $time

You can force identifiers to be re-evaluated when used in a /timer command by using the format $!me or $!time.

If you wish to turn off a range of timers, you can use a wildcard for the number, for example:

/timer3? off

Will turn off all timers from 30 to 39.

/timestamp [-s|a|e] [on|off] [windowname]
Turns time-stamping of events on/off, and attempts to timestamp **most** events that occur.

-s = for status window
-a = for active window
-e = for every window

If a windowname is not specified, then the global timstamp switch is turned on or off.

/titlebar [@window] <text>
Sets the main application titlebar. If you specify a custom @window name, then the titlebar for that custom window is changed.

/unload <-a|-rs> <filename>
Unloads the specified alias or remote script file.

/unload -a aliases.ini	unloads the alias.ini file
/unload -rs script.ini	unloads the script.ini file

Note: You can only unload one section at a time.

/updatenl
Updates the channel nicknames lists and IAL in a kick/part/quit script event.

/url [-d] [on|off|hide]
Pops up the URL list window. If you specify the [-d] then all "?" marked items will be deleted before showing the window.

You can also use **on|off** to turn URL catching on or off and **hide** to hide the URL window if it is currently showing.

See URL Catcher options for other settings.

/uwho [nickname] [nickname]
Pops up the address book and shows the server information for the specified user. It is the same information you would get if you did a "/whois nickname".

/window [-abcdelnorsx] <@name> [x y [w h]] [/command] [popup.txt] [font [size]]
Creates and manipulates custom windows.

/winhelp <filename> [key]
Opens a help file with the specified search key.

/write [-cida l# s#] <filename> [text]
Writes lines to a text file. For example:

/write store.txt This line will be appended to the file **store.txt**

The **-c** switch clears the file completely before writing to it, so it allows you to start with a clean slate.

/write -c c:\info.txt This file will be erased and have this line written to it

The **-l#** switch specifies the line number where the text is to be written. If you do not specify a line number then the line is added to the end of the file.

/write -l5 c:\info.txt This line will overwrite the 5th line in the file

The **-i** switch indicates that the text should be inserted at the specified line instead of overwriting it. If you do not specify any text then a blank line is inserted. If you do not specify a line number then a blank line is added to the end of the file.

/write -il5 c:\info.txt This line will be inserted at the 5th line in the file

The **-d** switch deletes a line in the file. If you don't specify a line number then the last line in the file is deleted.

/write -dl5 c:\info.txt

The above command will delete the 5th line in the file.

The **-s#** switch scans a file for the line beginning with the specified text and performs the operation on that line.

/write -dstest c:\info.txt

This will scan file info.txt for a line beginning with the word "test" and if found, deletes it.

If you do not specify any switches then the text is just added to the end of the file.

The **-a** switch indicates that mIRC should append the line of text you specified to the existing text of the specified line.

/writeini <inifile> <section> [item] [value]
Writes to files in the standard INI file format.

A part of the mirc.ini file looks like this:

```
[DDE]
ServerStatus=on
ServiceName=mirc
```

You could achieve this with /writeini by using:

```
/writeini my.ini DDE ServerStatus on
/writeini my.ini DDE ServiceName mirc
```

You can delete whole sections or items by using the /remini command.

Warning: Do not use this command to modify any of the INI files currently being used by mIRC.

Other features

The File Server

The mIRC fileserver allows other users to access files on your hard disk and is therefore **dangerous** since if used **improperly** it will allow them to access private/confidential information.

The /fserve command

A fileserver is initiated by using the **/fserve** command which initiates a DCC Chat to the specified user. You must specify a homedirectory. The user will be limited to accessing only files and directories within this homedirectory. The format is:

/fserve <nickname> <maxgets> <homedirectory> <welcomefile>

The **maxgets** is the maximum number of **simultaneous** dcc gets that a user can have during a fileserver session.

The **welcome file** is a text file that is sent to the user when they first connect. For example:

```
/fserve goat 5 c:\users\level1 level1.txt
```

This will initiate a filserver session to user goat with his homedirectory as c:\users\level1 and will send goat the text in the level1.txt file (presumably informing him that he is a level1 user and what files he can access etc.). The user can only have 5 simultaneous gets.

In each directory, you can place a **dirinfo.srv** file which describes that directory. Everytime the user does a CD to change into a directory, mIRC will look for this file and if it finds it, the text in it will be sent to the user.

Fileserver commands

The commands availabe to a user connected to your fileserver are:

cd <directory> - change to the specified directory.

dir [-b|k] [-#] [/w] - lists the name and size of each file in the current directory. The /w switch forces a wide listing. The [-b|k] selects bytes or k's. The [-#] specifies the number of files on each line in a horizontal listing.

ls [-b|k] [-#] - lists the name of each file in the currenty directory using a wide listing.

get <filename> - asks the fileserver to DCC Send the specified file.

read [-numlines] <filename.txt> - reads the specified text file. The user will be sent a default of 20 lines and then prompted whether to continue listing. The -numlines option changes the default number of lines to a value between 5 and 50.

help - lists the available commands.

exit or **bye** - terminates the connection.

Note:
1.If a directory has a large number of files try to split them up into subdirectories, this will improve performance.
2.If a user is idle for too long the fileserver will automatically close the connection. You can set the idle time out in the DCC Options dialog.
3.A user is limited to opening a **single** fileserver session at any one time. If mIRC initiates a fileserver

session to a user and that user doesnt respond then the fileserver session will have to time-out and close before that user can ask for another session.

System menu

If you click the **system menu** button in the top left hand corner of a window ie. the button you usually double-click to close a window, it will popup the usual system menu but with a few added functions (these vary depending on the type of window).

Position
You can tell mIRC to **remember** or **forget** the position/size of a window. If you select remember, the next time that window opens up it will do so in the saved position. If you choose **reset**, the window will be moved back to it's previously saved position.

Note: if a window's saved position lies outside the size of the main window then it will open in a default position and size. To force a window to open up in default positions assigned by windows, select **forget**.

Buffer
You can **clear** the text in the current window buffer or you can **save** the text to a file.

Background
This allows you to select a background .BMP picture for the a window. You can choose to have the picture displayed in various ways, eg. tiled, centered, etc.

Note: Displaying a background picture slows down the display of text in a window significantly.

Font
This allows you to change the current or default font for a window. The font settings for each window will be remembered across sessions. The only fonts shown in the font dialog are those which mIRC can properly display.

Logging
If logging is turned on, any text displayed in a window will be logged to a file. This setting stays on across sessions until you switch it off. The filename is automatically taken from the name of the window.

Beeping
If beeping is turned on, mIRC will beep any time a message is sent to the window if it **isn't** active. This setting is remembered across sessions for each window.

Flashing
If flashing is turned on, the mIRC window/icon is flashed if there is a new message in the window while mIRC is not the active application. The flash sound specified in the Event Beeps section is also played.

Desktop
This allows you to position a window outside of the main mIRC window and onto the desktop.

Stay On Top
This appears only when a window is opened in Desktop mode and forces the window to stay above all other windows.

Timestamp
This turns timestamping of events on/off for a window.

Text Copy and Paste

To copy text
You mark the text as usual with the mouse by pressing the left mouse-button and dragging it. The moment you release the left mouse-button, the text will be copied into the clipboard.

If you want to copy text **exactly** as it appears in the window, you can hold down the **Shift** key while you do the copy. Otherwise, text is copied in it's original format.

To paste text
You can then do the usual Shift-Insert key combination to paste the text any where you want.

The limitation
You can only copy the currently displayed text. To copy text from another page, you must scroll up/down to it and then copy it. If you want to store most of the text you see on a channel, you might want to use the logfile/buffer options in the System Menu.

The explanation
The use of colour in mIRC means that a simple text box cannot be used since text boxes can display only plain text (and they also have other limitations). However, text boxes also have built in cut/copy/paste routines which unfortunately are unavailable to a graphic window. This means that I had to code the mark/copy routine myself. I'm not sure which ran out first, my patience or my programming ability :-)

List Channels

The List Channels dialog allows you to retrieve the list of currently active channels. You can view the List Channels dialog by clicking on the List Channels button in the toolbar.

Apply
This allows you to respecify the list parameters without having to retrieve the whole list again from the IRC server. Just change the parameters and then click on apply to have them relisted according to your new criteria.

Get List
This retrieves a list of all of the active channels from the IRC server. This list can be quite long and depending on your connection it might take several minutes to download. The IRC Server actually sends the whole list, regardless of the filters you specify. You will not be able to do anything on IRC until this retrieval has been completed.

Match text
You can enter several words (separated by spaces) which mIRC will look for in channel names. Only those channels which match any of the words you specify will be listed. If you leave this **empty** then all channels will be listed.

Match topic
If this is turned on then mIRC will apply the Match Text procedure to channel topics as well. So only channel topics that match any of the words in the Match text editbox will be listed.

Do not sort
This prevents the channels list from being sorted.

Number of people
This allows you to limit the channels list to those channels which contain a number of people ranging between the specified minimum and maximum.

Lock/Unlock
This allows you lock the Hide parameters with a password thus preventing anyone from changing the Hide settings. The same password must be used to unlock this.

Hide channels which match...
You can enter several words (separated by spaces) which mIRC will look for in both channel names and topics. Any channels which match any of these words will be excluded from the channels list.

Hide non-alphanumeric channels
This will filter out any channels that begin with characters that aren't numbers or letters.

Note: Remember that you can right-click in the Channels List window to pop up the channels list menu.

URL Catcher

mIRC's URL feature catches any text that looks like a URL and saves it for future reference.

Enable URL Catcher
If this option is turned on mIRC will catch references to URLs and store them in the URL listbox. Turning on this option slows down mIRC a bit since it has to scan each incoming line before printing it, so if you don't need this option it will speed things up if you turn it off.

mIRC looks for URLs beginning with "http://", "ftp://", "gopher://", "www.", and "ftp.". mIRC also checks to make sure addresses are not added to a list if they already exist. Addresses longer than 256 characters are ignored.

mIRC will also catch any text that looks like an email as long as the word "mail" is mentioned in the same line of text as the email address.

On View...
When you select View from the popup menu to view a URL, mIRC can ask your browser to open a new separate URL window and it will also activate it if required.

On Send...
When sending URLs to a channel, query, etc. mIRC can send only the URL or both the URL and it's description.

Place ? marked URLs at top of list
If this is checked then mIRC will place ? marked URLs at the top of the URL list, otherwise they will be placed at the bottom of the list.

Delete All ? marked URLs on exit
To prevent your URL list getting too long you can choose to have all ? marked items deleted when you exit mIRC. Any items whose marker has been changed to something other than ? will remain in your list for future reference.

Location and name of WWW browser
This specifies your WWW browser so that mIRC can run it (if it isn't already running) when you select view from the popup menu.

Note: You can click your right mouse-button in the URL window for a popup menu which provides various URL functions.

Sound Requests

Sound requests allow users to share the experience of playing sounds together. When someone uses the **/sound** command on a channel, all users on the channel who have that same sound will hear it play.

Accept sound requests
If this option is turned on, mIRC will listen for **/sound** requests from other users.

On Sound Request...
If a sound is already playing and a new sound request is received, you can either have mIRC halt the currently playing sound and play the new sound, or you can choose to have mIRC ignore the new sound.

mIRC can also **warn** you if a user has requested a sound that you don't have so that you can then ask the user for that sound.

Listen for !nick file get requests
If someone sends you a message starting with an exclamation mark prefixing your nickname mIRC will assume that they are requesting a sound file from you and will search your sounds directory for the file. If it finds it, it dcc sends it to the user.

Send !nick file as private message
If this option is turned off and you send the message !nick file to a channel, everyone on the channel will see it. if you prefer not to crowd the channel with these messages, you can turn this option on and the user will receive the message privately.

Location of wave/midi files
Whenever a sound is requested, mIRC will look in these directories and all of their **subdirectories** for it. These directories are also searched when you use the /splay command.

The /sound command
You can send a sound request to another user using the /sound command.

/sound [on|off|nick/channel] <file.wav|file.mid> <message>

If a **nick/channel** is not specified, the message is sent to the current channel or query window.

The sound file must end in **.wav or .mid**, and may be located in the default sounds directory or in any of the subdirectories in the sounds directory. You do not need to specify a directory for the filename unless the file is not in the sounds directory.

The **message** is optional, if you do provide one it appears exactly like an action command.

The Internal Address List

mIRC maintains an **internal address list** of all users who are currently on the same **channels** as you.

This address list is used by the /guser, /ruser, /ban, /ignore, /finger, and /dns commands to quickly find a user's address without resorting to a **/userhost** server lookup.

A user's address is **added** to the list either when they join the channel, send a message to a channel, or make a mode change.

A user's address is **removed** from the list when they are no longer on any of the channels which you are currently on.

The reason why only addresses for users on the **same channels** as you are stored is because this guarantees the **integrity** of the list. ie. that a specific nickname is associated with a specific address.

Note: Maintaining the integrity of the internal address list requires quite a bit of processing. For people with faster computers this won't make a difference but for others this might slow down IRCing quite a bit if you are on several channels with lots of people with many messages, modes, joins, parts, etc. mIRC has to keep checking it's internal list to make sure it is updated correctly.

You can turn the Internal Address List off in the remote dialog if you find that it is slowing down your IRC session too much.

Key Combinations

F1
Shows the help file and is context sensitive, so you can press it in a dialog and it will bring up the help file section describing that dialog. Remember that if this key is redefined as an alias it will no longer work as a help key.

Shift-F1
Displays the Keyword search dialog for the help file. If this is redefined as an alias it will no longer work as a help key.

Control-Tab
Switches quickly between windows.

Shift-Tab
Switches between the editbox and the nickname listbox in a channel window.

Alt-Enter
In a multi-line editbox this moves the cursor to the next line allowing you to enter in several separate lines.

Control-F
In status/channel/query/etc. windows, this opens up a text search dialog allowing to you search the text buffer of that window for matching text.

Control-N
Cycles through channel windows.

Tab
If an editbox is empty and you press the Tab key, mIRC inserts a "/msg nickname" into the editbox of the window you are currently on, where **nickname** is the last person that sent you a message.

Note: The editbox needs to be empty in order for the Tab key to work as described above since the Tab key also performs the functions below...

If you are in a channel window editbox and it contains some text, the Tab key performs nickname completion on the word the cursor is on, this allows you type in only the first character or two of a nickname on that channel and then press Tab and mIRC will expand it to the full nickname.

If you press the Tab key while the cursor is positioned over a %variable or $identifier, it is evaluated.

Cursor Up/Down
Browses the command line history buffer for a single-line editbox.

Control-Cursor Up/Down
Browses the command line history buffer for a multi-line editbox.

Page Up/Down
Browses the scrollback buffer of a window a page at a time.

Control-Page Up/Down
Browses the scrollback buffer of a window a line at a time.

ESCape key

Quickly minimizes the active window, it must be turned on in the Extras dialog.

Shift key
If you want to copy text with line-breaks as it appears in a window, you can hold down the **Shift** key while you do the copy. Otherwise, text is copied without line-breaks.

Control-Enter
If you want to send information beginning with the ***/*** command prefix and you want it to be sent as normal text instead of **interpreted** as a command, just hold down the **Control** key when you press enter.

Control-B/U/R/K
Inserts control characters for bold, underline, reverse and colour in text.

Alt-1...9
If you press Alt and a number, mIRC will display the Nth window listed in your Window menu. If you press Alt-0 (zero) and you are in an mIRC desktop window, it will jump to the main mIRC window.

Command Line Parameters

You can specify the following parameters on the command line:

-s<server:port> forces mIRC to connect to the specified server and port on startup.

-j<#chan1,...,#chanN> forces mIRC to join the specified channels on connect.

-p<password> specifies password required to join channel.

-n<nick1,nick2> sets your nickname and alternate nickname to these nicks.

-i<filename.ini> forces mIRC to use the specified INI file.

IRC Switches

Prefix own messages
If this is selected, your nickname will prefix any messages you type in a channel/query/chat.

Iconify query window
If someone sends you a query, the default is for the query window to open, ready for input. You can select this option to force mIRC to iconify the window preventing it from taking the focus from the window you are currently in.

Dedicated query window
This directs all private messages or notices from other users to one single message window. You will need to use the /msg command to reply. If you want to open a query window to a user, use the /query window.

Whois on query
Select this to have mIRC do a /whois nickname on any person that sends you a private message.
The /whois will be done the first time the query window is opened.

Auto-join on invite
This will make you automatically join a channel when you are invited to it. mIRC will also try to minimize the window, however this might not always work.

Rejoin channel on connect
If this switch is turned on, mIRC will automatically rejoin channel windows which are open when you reconnect to an IRC server after being disconnected.

Rejoin channel when kicked
If you are kicked from a channel, mIRC will immediately try to rejoin the same channel. It won't close the channel window unless it finds that you can't rejoin the channel.

Cancel away on keypress
If you set yourself as away (using /away <message>) then selecting this option cancels the away option automatically if you type a message to a channel or a query/chat window.

Timestamp events
This will prefix most events with the time of their occurrence.

Skip MOTD on connect
This makes mIRC hide any MOTD information which the server sends you when you first connect to it.

Show Alternate Join/Part/Quit
Checking this option makes mIRC display the join, part, and quit messages in a different, more compact format.

Show user addresses
Whenever a user joins/parts/quits/is kicked/etc. from a channel, you can choose to see their address by selecting this option.

Show quits in channel
Normally if a user on the same channel as you quits IRC, this message is only printed in the status window. If you want the quit message to be printed in the channel window as well then select this option.

Show joins/parts in channel

You can turn off this option if you are on channels that are very crowded and your channel window is filling up with joins and parts. The joins and parts will instead be sent to the status window.

Show invites in active window
If this setting is turned off then invite messages will appear in the status window.

Show modes in channel
Normally all mode changes are shown in a channel window but you can turn this option off to dump these to the status window.

Show queries in active window
Shows all queries in the active channel window instead of opening up a query window. However, if you are not in a channel window, a query window will be opened.

Show topics in channel
Shows the channel topic in a channel when you first join it if it is turned on, otherwise the channel topic is displayed in the status window.

Show /whois in active window
Shows /whois results in the channel, query/chat, or custom windows if it is turned on and if a /whois is issued inside one of these windows, otherwise all /whois results are shown in the status window.

Strip Codes
This allows you to strip out the Bold, Underline, Reverse, and Colour codes from incoming private messages or channel messages.

Perform

Highlight lines with these words

This option will highlight incoming messages if they contain any of the words in this list. Lines which contain any of these words will be highlighted in the highlight colour you've chosen in the Colour dialog. This eases locating messages directed at you or about you eg. you could place your nickname in this box. The line you enter must consist of words separated by commas.

If **include nicks** is turned on, mIRC will also check the nickname of the user who sent the message to see if it matches any of the highlight words.

Note: You can also use %variables or $identifiers as hilight word.

On highlight...

This works in conjunction with the highlight option above.

If a line is highlighted and the beep option is turned on, mIRC will play the sound specified for highlight in the Event Beeps section.

If the **flash window** option is turned on and mIRC isn't the active application, mIRC will play the sound specified for flashing in the Event Beeps section.

On connect, perform these commands

You can enter a set of commands in this box which will be performed every time you connect to an IRC Server. To understand how commands work, see the Aliases help section. A quick example:

```
/join #mIRC
```

The above command would make you join the #mIRC channel when you connect to the server.

```
/nick $$?="Enter Nick:"
```

This command would ask you for a new nickname each time you connect to a server.

Ctcp finger reply

The message a user receives when they /ctcp finger you.

Quit message

The message displayed to other users when you quit IRC.

Event Beeps

On event, beep...

Certain events cause a number of beeps to be made to alert you of their occurence. It is possible to alter the number of beeps as well as how quickly they are sounded. The seconds value is in milliseconds. To turn off notification all together, specify zero beeps.

On event...

This allows you to turn on/off sounds for specific events. eg. for the disconnect event, if you are **disconnected** by the server without having typed /quit or selected the disconnect menu item, then mIRC will beep or play the specified sound to indicate that you are no longer connected to IRC.

Beep on channel message

Whenever a message is sent to a channel window that isn't currently the active window, mIRC will beep. Note that all this option does is automatically turn on the **Beep** setting in a channel window's System Menu when it first opens. So changing this setting doesn't affect windows which are already open.

Beep on query message

Whenever a message is sent to a query window that isn't currently the active window, mIRC will beep. Note that all this option does is automatically turn on the **Beep** setting in a query window's System Menu when it first opens. So changing this setting doesn't affect windows which are already open.

Beep on message while in buffer

Selecting this option will make mIRC beep if someone speaks on a channel while you are scrolling back through lines in the scrollback buffer.

Logging

Automatic logging
This allows you to automatically log **channel** and **chat** automatically whenever they open.

Strip control codes
This makes mIRC strip any Bold, Underline, Reverse, or Colour control codes from text that is being logger to a file.

Lock log files
If this switch is turned off and your log files are being saved properly then you should **leave** it turned off, otherwise turn it on.

Log files
You can view and delete any log files which are listed in your Logs and Buffers directory.

Log and Buffers directory
The directory in which all log files and buffer saves are stored.

Finger Server

The finger server listens for **finger requests** on port 79.

Enable finger server
This turns on the finger server.

Show finger requests
If this is turned on, finger requests are displayed in the status window.

Finger file
You can specify a finger file, which must be a plain text file, and this must be set up with **named sections** which will be used to reply to any finger requests. Each section begins in the following way:

```
[name]
line1
   .
   .
   .
lineN
```

The section name corresponds to the **userid** that is being fingered. eg. if someone fingers khaled@mardam.demon.co.uk, then I will have a section named [khaled].

There should be at least one section named **[default]** which will be used to reply to a finger request which does not specify a user, or specifies a user that doesn't exist.

Therefore, for my own system I would have two sections:

```
[default]
etc.

[khaled]
etc.
```

This can, for example, allow you to setup a menu system, with various sections that can answer user finger queries, the default section being the main menu.

If you want identifiers or variables in a line in the finger text file to be evaluated, you must prefix the text line with a $ character.

On a related note, check out the /finger command.

Local Info

Your **Local Host** and **IP Address** are needed if you want to use the DCC capabilities of mIRC.

mIRC will try to look up these values by itself and display them in the **Setup** dialog. However, if you see the message **Unable to resolve Local host** or you are unable to **initiate DCC sessions**, then changing the settings below might help.

If you see the message **Unable to resolve Local host** with the **32bit version of mIRC**, the problem might be related to using a **16bit winsock**, so you should try out the **16bit version of mIRC** to see if it works for you.

Local Host
This is used to register with the server and may be the part of your email address after the @ sign, eg. if my email address is khaled@mardam.demon.co.uk, then I would enter mardam.demon.co.uk here.

If you leave this box empty then mIRC will try to get your local hostname by itself. However, if mIRC replies with the message **Unable to get local hostname** then you will have to fill in your local hostname manually. mIRC will then use whatever you have entered to get your IP address...

IP Address
This will normally be filled in by mIRC and is here mainly for your information. mIRC looks up your IP address and stores it in the mirc.ini file for future reference. This way it doesn't have to look it up every time you want to connect.

If mIRC is having trouble getting your IP address then you can enter this value manually and mIRC will assume that it is correct. If this value is wrong you will still be able to log on to IRC but you will not be able to initiate DCC Send/Chat sessions (you will only be able to accept them).

On connect, always get...
These options are here because of the different types of internet connections people have. Some people have a fixed Local hostname and IP address, other have a dynamic Local Hostname, others a dynamic IP address, and yet others have both. If you don't know what kind of connection you have, leave both of these checked.

Selecting **Always get Local Host** automatically turns on the **Always get IP address** option. De-selecting the **Always get IP address** option automatically turns off the **Always get Local Host** option.

Lookup Method
If you find that mIRC is not resolving your IP address correctly you might try changing from **Normal** to **Server** or vice versa, which might solve the problem.

With the **Normal** method, mIRC relies on your winsock to reply with the correct information.

With the **Server** method, mIRC looks up your local host through the IRC Server, and then performs a /dns on it to resolve it to an IP address.

The Server method will most likely be slower, you can tell when it has been completed when you see your local host name and IP address displayed in the status window.

Note: If changing the above switches **still** doesn't solve the problem, or you don't know what to enter for your local host or ip address, contact your **internet provider** or **system administrator**.

Long File Names With.Spaces In Them

The DCC protocol doesn't take into account the possibility of a filename containing spaces, so most if not all IRC clients will incorrectly interpret the following DCC Send message:

PRIVMSG nick :DCC SEND This is a long file name with.spaces in it ipaddress port filesize

Thus mIRC gives you the **Fill Spaces** option; this fills spaces in a filename with the underscore character _, which should then allow other clients to interpret the message correctly. So other clients would see:

PRIVMSG nick :DCC SEND This_is_a_long_file_name_with.spaces_in_it ipaddress port filesize

If the Fill Spaces option isn't selected then mIRC sends **"Long File Names with.Spaces in them"** enclosed in quotes. For example:

PRIVMSG nick :DCC SEND "This is a long file name with.spaces in it" ipaddress port filesize

As far as I know, only mIRC can send and receive messages of this form (and only versions 3.8 and onwards), so if you try using this dcc send message with other clients it probably won't work.

Moo! ;)

Firewall

mIRC can connect to an IRC Server through a **SOCKS4** or **SOCKS5** firewall.

The **main** purpose of this option is to allow someone to access IRC through a SOCKS server at work, or more rarely through a network set up at home. The majority of home users should keep this option turned **off**.

SOCKS Settings

Hostname
The machine name of your SOCKS server, can be either a named address or an IP address.

User ID
Can be the account or user name on your system. For most people this will be the User ID portion of their email address (the text before the @ sign).

Password
The password required to access the firewall.

Port
This is usually be 1080.

Note: SOCKS4 is far more limited in functionality, eg. it doesn't support authentication, domain name resolution, or listening on a socket for a connection. The **most** you will be able to do with a SOCKS4 server is to connect to an IRC server.

SOCKS5 and DCC Send/Chat

mIRC supports an **experimental** method for DCC Send/Chat through a **SOCKS5** firewall. This will **not** work with older versions of mIRC or other IRC clients because there wasn't a practical way of implementing it to do this.

mIRC uses a **passive** protocol to establish DCC connections when a client is behind a SOCKS5 firewall. The method will work with SOCKS5 firewalls that both:

i) Support listening/binding to a port of Zero for an incoming connection.
ii) Assign outgoing connections an IP address that's the same as the SOCKS firewall IP address.

The DCC Send/Chat CTCP messages are extended by adding an extra number to the end of the message which uniquely identifies the negotiation, and a port of Zero is specified to indicate that this is a passive connection request. All other fields are identical to a standard DCC Send/Chat message.

DCC Chat Passive Protocol

Client A, initiating the DCC Chat, sends the passive connection request below to Client B. The port is set to zero, and the id number is a unique integer identifying the connection:

```
DCC CHAT nickname address port id
```

If Client A is behind a firewall, the address is the ip address of its SOCKS firewall.

Client B receives the message and sets up a listening socket, and sends the IP address and port back to Client A, specifying the id number that identifies Client A's request:

```
DCC CHAT nickname address port id
```

If Client B is behind a SOCKS5 firewall, it requests a listening socket from the SOCKS5 firewall and specifies Client A's IP address as the binding address.

Client A then proceeds to connect to this address to chat. If Client A is behind a SOCKS firewall, it sends a connection request to it.

DCC Send Passive Protocol

Client A, initiating the DCC Send, sends the passive connection request below to Client B. The port is set to zero, and the id number is a unique integer identifying the connection:

```
DCC SEND filename address port filesize id
```

If Client A is behind a firewall, the address is the ip address of its SOCKS firewall.

Client B receives the message and sets up a listening socket, and sends the IP address and port back to Client A, specifying the id number that identifies Client A's request:

```
DCC SEND filename address port filesize id
```

If Client B is behind a SOCKS5 firewall, it requests a listening socket from the SOCKS5 firewall and specifies Client A's IP address as the binding address.

Client A then proceeds to connect to this address to begin the transfer. If Client A is behind a SOCKS firewall, it sends a connection request to it.

Note: The DCC Resume and Accept protocols in mIRC are also extended by adding the id number to the end of the CTCP message, but otherwise work in exactly the same way.

Control

The following settings are related to channel and user control.

Auto-Op

If a user joins a channel where you have Ops and that user's address is listed in the auto-op list, they will be given Op status. You can add an address to the list in the following format:

nick!userid@host.domain,#channel1,#channel2

On IRC, user addresses are specified in the format:

nick!userid@host.domain

So if my nickname is **MadGoat** and my address is **khaled@mardam.demon.co.uk** then to put me in your list, you would use:

madgoat!khaled@mardam.demon.co.uk

If I change nicknames a lot, then you would use:

***!khaled@mardam.demon.co.uk**

If I change my nickname and userid a lot, then:

***!*@mardam.demon.co.uk**

The /auto command

/auto [-r] <on|off|nickname/address> [#channel1,#channel2,...] [type]

The **-r** switch indicates that the address is to be removed.

If you do **not** specify a type then only the users nick is added to the op list. If you specify a type then the users address is looked up via the server and added to the list.

Random delay auto-op

This option introduces a random 1 to 7 seconds delay in the auto-op routine. This is to prevent channel windows from filling up with mode notifications whenever a nickname is in the auto-op list of several users. If at the end of the random delay the user has already been opped then mIRC does not perform an Op.

Ignore

If a user sends you a message, whether on a channel or in private, and that user's address is in the ignore list, the users message will be ignored and won't be displayed. You can add an address to the list in the following format:

nick!userid@host.domain,private,invite,ctcp

The /ignore command

/ignore [-lrpcntikxu#] <on|off|nickname/address> [type]

Where p = private, c = channel, n = notice, t = ctcp, i = invite, k = control codes.

The **-u#** switch specifies a delay in seconds after which the ignore is automatically removed.
The **-r** switch indicates that the address is to be removed.
The **-x** switch indicates that this address should be excluded from ignores.
The **-l** switch displays the list of ignored addresses which match the specified switches.

If you do **not** specify a type then only the users nick is ignored. If you specify a type then the users address is looked up via the server and all messages coming from this address will be ignored.

Protect

If you are on a channel and you have channel Op status, any users that match the given nicknames will be automatically protected. mIRC does this by kicking or de-opping anyone who tries to kick or de-op your protected users. You can add an address to the list in the following format:

nickname,#channel1,#channel2

Note: This option is limited to using **nicknames** because of the way IRC servers work.

The /protect command

/protect [-r] <on|off|nickname> [#channel1,#channel2,...]

The **-r** switch indicates that the address is to be removed.

Variables

Variables are temporary storage areas to which you can assign **values** which you can use later in your scripts. All **currently** defined variables are listed in the remote variables section and can be edited there.

If a variable is referred to and it doesnt exist, it returns the value **$null**. The **$null** value can be used in comparisons in if-then-else statements to control branching etc.

The following **commands** allow you to create and set the values of variables.

/set [-zuN] <%var> [value]
This sets the value of %var to the specified value.

If you specify the **-uN** switch, %var is unset after N seconds, assuming it is not set again by another script.

The **-z** switch decreases %var until it reaches zero and then unsets it.

/unset <%var>
This unsets and removes the specified variables from the variables list. If you specify a variable with **wildcard** characters then all matching variables will be removed.

/unset %test*

This will **remove** all variables beginning with the word %test.

/unsetall
This unsets and removes all variables from the variables list.

/inc [-czuN] <%var> [value]
This increases the value of %var by value.

If you specify the **-uN** switch, %var is increased by the value **once** and then %var is unset N seconds later, assuming it is not set again by another script.

The **-c** switch increases %var once per second.

The **-z** switch decreases %var until it reaches zero and then unsets it.

/dec [-czuN] <%var> [value]
This decreases the value of %var by value.

If you specify the **-uN** switch, %var is decreased by the value **once** and then %var is unset N seconds later, assuming it is not set again by another script.

The **-c** switch decreases %var once per second.

The **-z** switch decreases %var until it reaches zero and then unsets it.

You can also use the **equal** sign to **assign** values to variables:

```
%i = 5
%xyzi = 3.14159
%count = $1
```

And you can perform the following **operations** on variables when using the **equal** sign:

```
%x = 5 + 1
%x = 5 - %y
%x = %x * 2
%x = %z / $2
%x = $1 % 3
%x = 2 ^ %w
```

You can only perform a **single** operation in an assignment at this time.

You can also use the **$calc()** identifier which allows you to perform complex calculations.

```
//echo 1 $calc(3.14159 * (2 ^ %x % 3) - ($ticks / (10000 + 1)))
```

For **floating point** numbers you can also use the **$round(N,D)** and **$int(N)** identifiers to handle precision of the decimal digits. The number of decimals is currently limited to 5 digits.

on BAN/UNBAN

The **on BAN** and **on UNBAN** events trigger when a user on a channel is banned or unbanned.

Format:	on <level>:BAN:<#[,#]>:<commands>
Example:	on 1:BAN:#mirc,#irchelp:/msg $nick Sorry but you're not allowed on $chan

Examples

on 9:BAN:#newbies:/mode $chan -o $nick | /mode $chan -b $banmask

This triggers when someone bans a user with access level 9. **$banmask** refers to the banmask used to ban the user.

on 1:UNBAN:#:/msg $bnick You have just been unbanned

This triggers when any user is unbanned from any channel. **$bnick** refers to the banned users nickname, however this would actually only be filled if the banmask itself includes a nickname. If the banmask does not include a nickname, $bnick is $null.

Remember that **$banmask** is usually a **wildcard string** which means that it will be matching wildcard strings in your remote users section. For example, if someone sets a ban of ***!k*d@*.uk** it will match users:

*!khaled@mardam.demon.co.uk
*!kha*d@*am.d*mo?.co.*
!k@*.u?

Comparing levels

You can **compare the levels** of the banner and the banned by prefixing the line with <,>,<=,=>,<>, or =, in the following way:

on >=2:BAN:#mIRC:/msg $chan $nick banned $banmask (legal)
on 1:BAN:#mIRC:/msg $chan $nick banned $banmask (illegal)

In this situation, if the banners level is larger than or equal to the banned users level, then it is a legal ban. Otherwise, it defaults to the second ON BAN line which indicates that it is an illegal ban. Remember, this is comparing the banners and banned users levels and has nothing to do with with the level 2 in the definition.

Note: These events only work on nicknames because the IRC server only sends the nickname of the user being banned/unbanned and not an address. Also, IP addresses will not be matched against named addresses, and banmasks ending in @* will be ignored since this can match almost any user address.

How to Register

As described in the License, mIRC is a **Shareware** program, which means that you can use it legally for 30 days free of charge to **evaluate** it. If during, or at the end of, that period you decide that you would like to continue using it, please **register** your copy. Your single-user registration will license you to use your copy of mIRC, will support work on future versions, new features, and bug fixes, and will provide you with technical support via email.

The **latest registration information** can always be found on the mIRC website.

The current registration amount is **US$20.00** (or **UK£10.00**). This is a one-time registration fee for your current version of mIRC.

You can register mIRC either by:

1. Sending a **US** or **UK** personal **check** or **money/postal order** in my name to:

Khaled Mardam-Bey
Apt. 347,
56 Gloucester Road,
London SW7 4UB,
UK

Please make sure to include your **full name**, **postal address**, and **email address** with your letter.

I can also handle EuroCheques in UK pounds, and Canadian and Australian Postal Money Orders in UK or US currency. If you are going to send something **other** than a **UK** or **US** check or money/postal order, please email me first to make sure I can accept it.

2. Registering **online** with your **Credit Card**, you can find out how at:

http://www.mirc.co.uk/register.html

The registration amount is **US$20.00** when registering with a credit card.

Once I **receive** your registration, either via Postal mail or via your online credit card order, I will send you an **email** with your registration information. If you've submitted a registration and haven't received a reply, you can email me at khaled@mardam.demon.co.uk to check up on the status of your registration.

You can then **enter your registration information** in the **registration dialog** by going to the **Help menu** in the menubar (not the toolbar) in mIRC, and selecting **Register** at the bottom of the menu.

Drag and Drop

The **Drag and Drop** feature allows you to **pick up files** from other programs eg. a file manager, and **drop** them on either **Message** or **Channel** windows. Depending on the type of file you drop onto a window, a different action will be performed according to the commands you've defined for that file type.

You can define different actions for different types of files by **associating a command** with **a file ending**.

For example, if you wanted to mIRC to **read** a text file to another user whenever you drop files ending in **.txt** then you might make an association such as:

```
*.txt:/play $1 $2-
```

In this case, **$1** stands for the name of the user or channel where the file has been dropped, and **$2-** stands for the name of the file that was dropped. The /play command would send each line in the specified text file to the user.

You can also have different aliases executed for the same file type depending on whether you hold down the **shift key** or not when you drop the file.

Currently the **default** settings for dropping files with **no shift key** pressed are:

```
*.wav:/sound $1 $2-
*.*:/dcc send $1 $2-
```

Which means that if you drop a Wave file, it will be played with the /sound command, and if you drop any other type of file it will initiate a dcc send to that user.

The **default** settings for dropping files with the **shift key** held down are:

```
*.*:/dcc send $1 $2-
```

Which is the same as above, initiating a dcc send to the active user.

Note: if you drop a file which has a space in it, it is enclosed in "" quotes.

Raw Events

Raw events are special IRC Server messages that are identified only by a number. There are a large number of raw events, far too many to go into here, so it is recommended that you check out the links on the mIRC Homepage for technical information. The document you are looking for is called **RFC1459**. There is also a numerics help file available which is more up-to-date than this document.

Filtering and **handling** raw messages can be very **time-consuming** because of the large number of messages that a server can send, so you should try to make sure that your scripts process this information as quickly as possible. The format of the raw event definition is:

```
raw <numeric>:<matchtext>:<commands>
```

Examples

As a **quick** example, the following script filters out the **channels list** numeric when you use the /list command.

```
raw 322:*mirc*:/echo 5 $1-
```

The above script matches numeric 322 which is the channels /list numeric and if the line returned by this numeric has the word **mirc** in it, it is printed out in the status window.

Note: You can prevent most raw server messages from printing out their default text by using the /halt command.

on FILESENT/FILERCVD

The **on FILESENT** and **on FILERCVD** events trigger when a dcc send or dcc get succeeds.

Format:	on <level>:FILESENT:<filename[,filename]>:<commands>
Example:	on 1:FILESENT:*.txt:/msg $nick I have successfully sent you the $filename text file

The **on SENDFAIL** and **on GETFAIL** events use the same format as above, and trigger when a dcc send or dcc get fails.

Examples

on 1:FILESENT:*.txt,*.ini:/echo Sent $filename to $nick $address

This triggers when a dcc send succeeds in sending a .txt or .ini file to a user. **$filename** refers to the filename that was transmitted.

on 1:FILERCVD:*.txt,*.ini:/echo Received $filename from $nick | /run notepad.exe $filename

This triggers when a dcc get succeeds in getting a .txt or .ini file from a user.

on 1:SENDFAIL:*.txt:/echo I failed to send the text file $filename to $nick

This triggers when a dcc send failed to send a .txt file to a user.

on 1:GETFAIL:*.zip:/echo I failed to get the zip file $filename from $nick

This triggers when a dcc get failed to get a .zip file from a user.

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
//	is a multiple of
\\	is not a multiple of
&	is a 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
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)
ison	nickname v1 is on channel v2
isop	nickname v1 is an op on channel v2
isvo	nickname v1 has a voice on channel v2
ishelp	nickname v1 is a helper on channel v2
ischan	if v1 is a channel which you are on.
isauto	if v1 is a user in your auto-op 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.

$ifmatch

Returns the first parameter of matching if-then-else comparison. So, in the case of this comparison:

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

$ifmatch will return "text"

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.

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.

Identifiers

Identifiers return specific values eg. **$time** would return the **current time**. Whenever mIRC finds an identifier in your command, it **replaces** it with the **current value** of that identifier. Many identifers also perform functions on data that you supply and then return a result.

There are also special types of identifiers used in remote scripts and socket connections.

Identifiers which cannot be evaluated or evaluate to no value return the value **$null**. The **$null** value can be used in comparisons in if-then-else statements to control branching etc.

For all of the following identifiers, you can place **other** identifiers or variables **inside** the brackets.

The identifiers are listed below according to group.

Time and Date identifiers

$asctime(N)
Converts time values returned by $ctime (also the ping reply) into a full date in text format.

$ctime
Returns total number of seconds elapsed since 00:00:00 GMT, January 1, 1970 based on your system time.

$ctime(text)
Returns the number of seconds elapsed since 00:00:00 GMT, January 1, 1970 based on the date that you specify.

$ctime(January 1 1970 00:00:00)
$ctime(3rd August 1987 3:46pm)
$ctime(21/4/72 1:30:37)
$ctime(Wed 1998-3-27 21:16)

$date
Returns the current date in day/month/year format.

For the date in US format you can use $adate.

$day
Returns the name of the current day ie. Monday, Tuesday, etc.

$duration(N)
Returns the specified number of seconds in a week/day/hour/minute/second format.

$fulldate
Returns the current date in the format: Wed Jun 26 21:41:02 1996

$idle
Returns your current idle time (same time as that returned by a ctcp finger).

$ltimer
Returns the number of the last timer that was started by the /timer command.

$online

Returns the number of seconds elapsed in the Timer dialog.

$ticks
Returns the number of ticks since your O/S was first started.

$time
Returns the current time in hour:minute:second format.

$timer(N/name)
Returns the timer id of the **Nth** timer in the timers list. You can also specify a timer **name** instead of a number. This identifier works in conjunction with the /timer command.

Properties: com, time, reps, delay, type

$timer(0)	returns the number of active timers
$timer(1)	returns the timer id of the 1st timer in the list
$timer(1).com	returns the command for the 1st timer in the list
$timer(3).type	returns online/offline status for the 3rd timer in the list

$timestamp
Returns the current time in [xx:xx] format.

$timezone
This returns your current timezone setting in seconds. For the 16bit version the return value depends on the TZ environment variable.

Text and Number identifiers

$abs(N)
Returns the absolute value of number N.

$abs(5)	returns 5
$abs(-1)	returns 1

$asc(C)
Returns the ascii number of the character C.

$asc(A)	returns 65
$asc(*)	returns 42

$calc(operations)
Returns the result of the specified operations. This identifiers allows you to perform multiple operations easily. For example:

```
$calc(3.14159 * (2 ^ %x % 3) - ($ticks / (10000 + 1)))
```

$chr(N)
Returns the character with ascii number N.

$chr(65)	returns A
$chr(42)	returns *

$cos(N), $acos(N)
Return the cosine and arccosine of N.

$count(string,substring)

Returns the number of times substring occurrs in string.

$count(hello,el)   returns 1
$count(hello,l)   returns 2

$int(N)
Returns the integer part of a floating point number with no rounding.

$int(3.14159)   returns 3

$left(text,N)
Returns the N left characters of text.

$left(goodbye,4)   returns good

$len(text)
Returns the length of text.

$len(#mIRC)   returns 5

$longip(address)
Converts an IP address into a long value and vice-versa.

$longip(158.152.50.239)   returns 2660774639
$longip(2660774639)   returns 158.152.50.239

$lower(text)
Returns text in lowercase.

$lower(HELLO)   returns hello

$mid(text,S,N)
Returns N characters starting at position S in text.

$mid(othello,3,4)   returns hell

If N is zero, it returns the number of characters from S to the end of the line.

$pos(text,string,S)
Returns a number indicating the position of string in text starting the search from position S in text.

$pos(hello,el,1)   returns 2
$pos(hello,la,1)   returns $null

Note: if S is zero, it returns the number of times string appears in text.

$rand(v1,v2)
This works in two ways. If you supply it with numbers for v1 and v2, it returns a random number between v1 and v2. If you supply it with letters, it returns a random letter between letters v1 and v2.

$rand(a,z)   returns a letter in the range a,b,c,...,z
$rand(A,Z)   returns a letter in the range A,B,C,...,Z
$rand(0,N)   returns a number in the range 0,1,2,...,N

$remove(string,substring,...)
Removes any occurrence of substring in string.

$remove(abcdefg,cd)   returns abefg

You can also specify **multiple** remove parameters:

$remove(abcdefg,a,c,e,g)   returns bdf

$replace(string,substring,newstring,...)
Replaces any occurrence of substring in string with newstring.

$replace(abcdefg,cd,xyz)   returns abxyzefg

You can also specify **multiple** replace parameters:

$replace(abcdefg,a,A,b,B,c,C,d,D)   returns ABCDefg

$right(text,N)
Returns the N right characters of text.

$right(othello,5)   returns hello

$round(N,D)
Returns the specified floating point number rounded to the Dth decimal digit.

$round(3.14159,2)   returns 3.14

$sin(N), $asin(N)
Return the sine and arcsine of N.

$str(text,N)
Returns text repeated N times.

$str(ho,3)   returns hohoho

$strip(text)
Returns text with all bold, underline, reverse, and colour control codes stripped out.

$tan(N), $atan(N)
Return the tangent and arctangent of N.

$upper(text)
Returns text in uppercase.

$upper(hello)   returns HELLO

File and Directory identifiers

$alias(N/filename)
Returns the filename for the Nth loaded alias file. If you specify a filename, it returns $null if the file isn't loaded.

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

$dir, $file, $hfile, and $sdir

Allow you to select a filename which is then inserted into an alias. The $dir identifier pops up a full directory and file dialog, while the $file just pops up a quick file dialog. The $hfile is the same as $file except that it lists files horizontally. The $sdir allows you to select a directory.

$dir[="Select a file"] <path and filename>
$file[="Select a file"] <path and filename>
$hfile[="Select a file"] <path and filename>
$sdir[="Select a directory"] <path>

For example, in a popup definition:

Play A Wave:/splay $file="Choose a wave!" c:\mywaves\*.wav

Note: You should not use $file or $dir with the /dcc send command which has it's own built-in dcc send dialog.

$exists(filename)
Returns $true if a file exists and $false if it doesn't.

$exists(c:\mirc\mirc.exe)   returns $true or $false.

$filtered
Returns the number of lines that were filtered when using the /filter command.

$finddir(dir,dirspec,N)
Searches the specified directory and its subdirectories for the Nth directory name matching the filespec and returns the full path and directory if it is found.

$finddir(c:\,mirc*,1)   returns the first directory name beginning with "mirc"

If you specify a custom @window name instead of the N parameter, mIRC will fill the custom @window with the results.

$findfile(dir,filespec,N)
Searches the specified directory and its subdirectories for the Nth filename matching the filespec and returns the full path and filename if it is found.

$findfile(c:\mirc,*.exe,1)   returns c:\mirc\mirc.exe

If you specify a custom @window name instead of the N parameter, mIRC will fill the custom @window with the results.

$getdir
Returns the DCC Get directory specified in the DCC Options dialog.

$getdir(filespec)
Returns the DCC Get directory for the specified file type.

$getdir(*.txt)   returns c:\mirc\text\ (for example)

$lines(filename)
Returns the total number of lines in the specified text file.

$lines(c:\irc\kicks.txt)   returns the total number of lines in c:\irc\kicks.txt

$lof(filename)

Returns the length of the specified file in bytes.

$lof(c:\net\mirc\mirc32.exe)   returns 605694

$logdir
Returns the Logs directory as specified in the Logging section of the Options dialog.

$mididir
Returns the Midi directory specified in the Sound Requests section of the Options dialog.

$nofile(filename)
Returns the path in filename without the actual filename.

$nopath(filename)
Returns filename without a path if it has one.

$nopath(c:\mirc\mirc.exe)   returns mirc.exe

$mircdir
Returns the current directory of the mIRC program.

$mircini
Returns the name of the main .ini file, usually mirc.ini.

$read and $readn
Reads a line from a file and inserts it into the current position in an alias command. The format is:

$read [-nl# -swtext] <filename>

$read will insert any text, even commands with identifiers, and these will work like normal commands.

/say $read c:\funny.txt

Reads a random line from funny.txt and inserts it at that position in the command.

/say $read -l24 c:\funny.txt

Reads line 24 from funny.txt and inserts it at that position in the command.

/kick # $1 $read kicks.txt

Reads a random kick line from kicks.txt and uses it in the kick command.

/say $read -smirc info.txt

Scans the file info.txt for a line beginning with mirc and uses the rest of the line in the alias.

/say $read -w*help* help.txt

Scans the file help.txt for a line containing the word "help" anywhere in the line.

The **$readn** identifier returns the line number that was matched when using the **-s** or **-w** switches.

If you specify the **-s** or **-w** switches, you can also specify the **-lN** switch to specify which line to start searching after in the file, eg.:

/echo -l100 -w*mirc* versions.txt

If the **-n** switch is specified then the line read in will not be evaluated and will be treated as plain text.

Note: If the first line in the file is a single number, it must represent the total number of lines in the file. This speeds up the $read considerably. If you don't specify the total number of lines in the file on the first line, then mIRC will need to count all of the lines itself which slows it down. For small files, this will make no difference, but for large files the delay will be very noticable. If you specify a line number of 0, mIRC returns the value of the first line if it's a number.

$readini
Reads information from an INI file and inserts into in the current position in an alias command. It works in conjunction with the /writeini command to read information from INI files. The format is:

$readini <-n> <filename> <section> <item>

/echo $readini mirc.ini mIRC nick

Reads your nickname from the mirc.ini file.

If the **-n** switch is specified then the line read in will not be evaluated and will be treated as plain text.

$shortfn(filename)
Returns short version of a long filename, only works in 32bit mIRC. In 16bit, returns same filename.

$wavedir
Returns the Waves directory specified in the Sound Requests section of the Options dialog.

Nickname and Address Identifiers

$address(nickname,type)
Searches the Internal Address List for the address associated with the specified nickname.

$address(nick,1)   returns nick!userid@domain.host

If the Internal Address List doesn't contain a matching nickname, the identifier returns $null.

See **$mask()** for a list of types.

$comchan(nick,N)
Returns the names of channels which both you and nick are on.

Properties: op, help, voice,

$comchan(nick,0)	returns the total number of common channels
$comchan(nick,1)	returns the first common channel name
$comchan(nick,1).op	returns $true if you're an op on the channel

$hnick(#,N/nick)
Returns the Nth Helper nickname on channel #.

$hnick(#mIRC,0)	returns the the total number of Helpers on #mIRC
$hnick(#mIRC,1)	returns the 1st Helper nickname on #mIRC

Note: This identifer has been added to support the upcoming mode +h on TS4 servers.

$ial(mask,N)
Returns the Nth address matching mask in the Internal Address List.

Properties: nick, user, host, addr

$ial(*!*@*.demon.co.uk,0)	returns the total number of addresses in the IAL matching *!*@*.demon.co.uk
$ial(*!*@*.demon.co.uk,3)	returns the 3rd address in the IAL matching *!*@*.demon.co.uk
$ial(*!*@*.com,4).nick	returns the nick of the 4th matching address ending in .com
$ial(*!*@*.com,4).user	returns the userid of the 4th matching address ending in .com

To scan each address in the IAL you can use $ial(*,N).

$ialchan(mask,#,N)
Returns the Nth address on the specified channel matching mask in the Internal Address List.

This works the same way as the $ial() identifier above.

$level(address)
Finds a matching address in the remote users list and returns its corresponding levels list.

$level(*!*@mardam.demon.co.uk)   returns =5,10,20,21,32

$link(N)
Returns the Nth item listed in the server Links window.

Properties: addr, ip, level, info

$link(0)	returns the total number of links in the links window
$link(1)	returns the Nth server address in the links window

$mask(address,type)
Returns address with a mask specified by type.

$mask(nick!khaled@mardam.demon.co.uk,1)	returns *!*khaled@mardam.demon.co.uk
$mask(nick!khaled@mardam.demon.co.uk,2)	returns *!*@mardam.demon.co.uk

The available types are:

```
  0: *!user@host.domain
  1: *!*user@host.domain
  2: *!*@host.domain
  3: *!*user@*.domain
  4: *!*@*.domain
  5: nick!user@host.domain
  6: nick!*user@host.domain
  7: nick!*@host.domain
  8: nick!*user@*.domain
  9: nick!*@*.domain
```

You can also specify a type of 10 to 19 which correspond to masks 0 to 9, but instead of using a * wildcard to replace portions of the host.domain, mIRC uses ? wildcards to replace the numbers in the address.

This standard set of masks is also used in other identifiers and commands.

$me
Returns your current nickname.

$nhnick(#,N/nick)
Returns the Nth non-Helper nickname on channel #.

$nhnick(#mIRC,0)	returns the the total number of non-Helpers on #mIRC
$nhnick(#mIRC,1)	returns the 1st non-Helper nickname on #mIRC

Note: This identifer has been added to support the upcoming mode +h on EFnet TS4 servers.

$nick(#,N/nick)
Returns Nth nickname in the channels nickname listbox on channel #.

$nick(#mIRC,0)	returns the the total number of nicknames on #mIRC
$nick(#mIRC,1)	returns the 1st nickname on #mIRC

$nopnick(#,N/nick)
Returns the Nth non-Op nickname on channel #.

$nopnick(#mIRC,0)	returns the the total number of non-Ops on #mIRC
$nopnick(#mIRC,1)	returns the 1st non-Op nickname on #mIRC

$notify(N/nick)
Returns the Nth nickname in your notify list.

Properties: ison, note, sound, whois

$notify(0)	returns the number of nicknames in your notify list.
$notify(3)	returns the 3rd nickname in your notify list.
$notify(3).ison	returns $true if this user is on IRC, $false if not.

$nvnick(#,N/nick)
Returns the Nth non-voiced nickname on channel #.

$nvnick(#mIRC,0)	returns the the total number of non-voiced nicknames on #mIRC
$nvnick(#mIRC,1)	returns the 1st non-voiced nickname on #mIRC

$opnick(#,N/nick)
Returns the Nth Op nickname on channel #.

$opnick(#mIRC,0)	returns the the total number of Ops on #mIRC
$opnick(#mIRC,1)	returns the 1st Op nickname on #mIRC

$snicks
Returns a string of the currently selected nicknamess in the active channel listbox in the form:

nick1,nick2,nick3,...,nickN

$snick(#,N)
Returns the Nth selected nickname in the channel listbox on channel #.

$snick(#mIRC,0)	returns the the total number of selected nicknames on #mIRC
$snick(#mIRC,1)	returns the 1st selected nickname on #mIRC

$snotify

Returns the currently selected nickname in the notify list box.

$vnick(#,N/nick)
Returns the Nth voiced nickname on channel #.

$vnick(#mIRC,0)   returns the the total number of voiced nicknames on #mIRC
$vnick(#mIRC,1)   returns the 1st voiced nickname on #mIRC

Window Identifiers

$active
Returns the full name of the currently active window in mIRC.

$appactive
Returns **$true** if mIRC is the active application, otherwise it returns **$false**.

$chan(N/#)
Returns information on channels that you are currently on.

Properties: topic, mode, key, limit, ial

If you specify a number N, the Nth channel name is returned.

$chan(0)   returns the number of channels you are on
$chan(2)   returns the name of the 2nd channel you are on
$chan(2).key   returns the key of the 2nd channel you are on
$chan(4).ial   returns $true if IAL contains addresses of all users on this channel

If you specify a channel name, information on that channel is returned but only if you are on that channel already.

$chan(#mIRC).mode   returns the mode of channel #mIRC

$chat(N/nick)
Returns the name of the Nth open dcc chat window.

Properties: ip, status

$chat(0)   returns the total number of open dcc chats.
$chat(1)   returns the nickname of the 1st dcc chat window.
$chat(2).ip   returns the ip address of the 2nd open dcc chat window.

$fserv(N/nick)
Returns the name of the Nth open dcc chat window.

Properties: ip, status, cd

$fserv(0)   returns the total number of open fserves.
$fserv(1)   returns the nickname of the 1st fserve.
$fserv(1).cd   returns the current directory of the 1st fserve.

$get(N/nick)
Returns the nickname and filename of the Nth open dcc get window.

Properties: ip, status, file, path, size, rcvd, cps, pc

$get(0)	returns the total number of open dcc gets.
$get(2)	returns the nickname of the 2nd dcc get.
$get(2).rcvd	returns the number of bytes received for the 2nd dcc get.
$get(2).cps	returns the character per second rate for the 2nd dcc get.
$get(3).pc	returns the percent complete of transfer for the 3rd dcc get.

$query(N/nick)
Returns the nickname of the Nth open query window.

Properties: address

$query(0)	returns the total number of open query windows.
$query(2)	returns the name of the 2nd open query window.

$query(N).address   returns the address of the Nth query, however note that this address is not available until after the user has sent you a message, and that this address **may not** be correct.

$send(N/nick)
Returns the nickname and filename of the Nth open dcc send window.

Properties: ip, status, file, path, size, sent, lra, cps, pc

$send(0)	returns the total number of open dcc sends.
$send(2)	returns the nickname of the 2nd dcc send.
$send(2).sent	returns the number of bytes sent for the 2nd dcc send.
$send(2).lra	returns the last received ack for the 2nd dcc send.
$send(3).pc	returns the percent complete of transfer for the 3rd dcc send.
$send(3).status	returns active, inactive, or waiting for the 3rd dcc send.

Token identifiers

$addtok(text,token,C)
Adds a token to the end of text but only if it's not already in text.

$addtok(a.b.c,d,46)	returns a.b.c.d
$addtok(a.b.c.d,c,46)	returns a.b.c.d

$deltok(text,N,C)
Deletes the Nth token from text.

$deltok(a.b.c.d,3,46)	returns a.b.d

$findtok(text,token,N,C)
Returns the position of the Nth matching token in text.

$findtok(a.b.c.d,c,1,46)	returns 3
$findtok(a.b.c.d,e,1,46)	returns $null

If you specify **zero** for N, it returns the total number of matching tokens.

$gettok(text,N,C)
Returns the Nth token in text.

$gettok(a.b.c.d.e,3,46)	returns c
$gettok(a.b.c.d.e,9,46)	returns $null

You can also specify a range of tokens:

$gettok(a.b.c.d.e,2-,46)	returns 2nd token onwards b.c.d.e
$gettok(a.b.c.d.e,2-4,46)	returns tokens 2 through 4 b.c.d

$instok(text,token,N,C)
Inserts token into the Nth position in text, even if it already exists in text.

$instok(a.b.d,c,3,46)	returns a.b.c.d
$instok(a.b.d,c,9,46)	returns a.b.d.c

$matchtok(tokens,string,N,C)
Returns tokens that contain the specified string.

$matchtok(one two three, e, 0, 32) returns 2
$matchtok(one two three, e, 2, 32) returns three

If you specify **zero** for N, it returns the total number of matching tokens.

$puttok(text,token,N,C)
Overwrites the Nth token in text with a new token.

$puttok(a.b.c.d,e,2,46)	returns a.e.c.d

$remtok(text,token,N,C)
Removes the Nth matching token from text.

$remtok(a.b.c.d,b,1,46)	returns a.c.d
$remtok(a.b.c.d,e,1,46)	returns a.b.c.d
$remtok(a.c.c.d,c,1,46)	returns a.c.d

$reptok(text,token,new,N,C)
Replaces the Nth matching token in text with a new token.

$reptok(a.b.c.d,b,e,1,46)	returns a.e.c.d
$reptok(a.b.c.d,f,e,1,46)	returns a.b.c.d
$reptok(a.b.a.c,a,e,2,46)	returns a.b.e.c

$wildtok(tokens,wildstring,N,C)
Returns the Nth token that matches the wildcard string.

$wildtok(one two three, t*, 0, 32) returns 2
$wildtok(one two three, t*e, 1, 32) returns three

If you specify **zero** for N, it returns the total number of matching tokens.

Miscellaneous identifiers

$?*!="message"
This identifer allows you to request input from a user by popping up an input dialog.

//echo $?="What is your name?"

If the user enters their name in the editbox and presses the OK button, $? will return whatever the user entered. If the user clicks on the Cancel button, $? returns nothing.

//echo $?*="What is your password?"

In this case the $?* makes any text that the user types into the editbox appear as ***** characters to prevent anyone seeing what is being entered.

//echo $?!="Shall I continue?"

In this case, a Yes/No dialog pops up. If the user clicks on Yes, $true is returned, otherwise $false is returned.

The input dialog is extended vertically to display the whole message if it is very long. You can also make text appear on different lines by using the $crlf identifier to separate the lines, eg.

//echo $?="This is on the first line. $crlf $+ And this is on the 2nd line."

Note: This identifier cannot be used from within a script event that is reacting to a server event. One way around this is to use a /timer to initiate an input request.

$away
Returns the value **$true** or **$false** depending on whether you are marked as away or not.

if ($away) say I'm away! | else say I'm here!

$bits
Returns 32 for the 32bit mIRC, or 16 for the 16bit mIRC.

$cb
Returns the first 256 characters of the clipboard contents.

$cb(N)
Returns CRLF delimited lines from text currently in the clipboard.

Properties: len

$cb(0)	returns the number of lines in the clipboard
$cb(0).len	return the total length of all lines in the clipboard
$cb(1)	returns line 1 from the clipboard
$cb(1).len	returns the length of line 1

$colour(name)
Returns the N index of the specified colour name eg. $colour(action text). If you don't specify the full name the first partial match is returned eg. $colour(action)

$cr
Returns the **carriage return** character, the same as $chr(13).

$creq
Returns current /creq settings in the DCC Options chat section dialog.

$crlf
Returns a carriagereturn/linefeed combination.

$ddename
Returns the current dde service name in use by the DDE Server.

$editbox(window)

Returns the text in the editbox of the specified window.

$email
Returns the email address specified in the Setup dialog.

$hash(text,B)
Returns a hash number based on text where B is the number of bits to use when calculating the hash number.

$host
Returns your Local host name.

$ifmatch
Returns the first parameter of matching if-then-else comparison.

In the case of this comparison:

if (text isin sometext) { ... }

$ifmatch returns "text"

$ignore(N)
Returns the Nth address in the ignore list.

Properties: type

$ignore(0)	returns the total number of addresses in the ignore list
$ignore(1)	returns the 1st address in the ignore list
$ignore(2).type	returns the ignore flags for the 2nd address in the ignore list

$iif(C,T,F)
Returns T or F depending on whether the evaluation of the Conditional C is true or false.

$iif(1 == 2, yes, no)   returns "no"

$iif() returns F if the conditional returns zero, $false, or $null. For any other value $iif() returns T.

If you don't specify the F parameter, $iif returns a T value if the condition is true, and returns nothing if it's false.

$iif(1 == 2, yes)   returns nothing

You can find out more about conditionals in the if-then-else section.

$inmidi
Returns $true if a midi file is currently playing, otherwise returns $false.

$inwave
Returns $true if a wave file is currently playing, otherwise returns $false.

$ip
Returns your IP address.

$isalias(name)
Returns $true if the specified name is an alias command that exists in your aliases or scripts.

Properties: fname, alias

$isalias(join)	returns $true if you have an alias for /join
$isalias(join).fname	returns the filename in which the alias exists
$isalias(join).alias	returns the alias definition for /join

$lf
Returns the **linefeed** character, the same as $chr(10).

$os
Returns the version number of the operating system. The reply can be 3.1, 95, 98, or NT.

Note: the 16bit mIRC is limited to replying either 3.1 or 95.

$port
Returns the **port number** of the server to which you're currently connected.

$result
Stores the number value returned to a calling routine by the **/return** command.

$rgb(name)
Returns RGB value of specified system colour name which can be one of the following: face, shadow, hilight, frame, and text.

$server
Returns the name of the server to which you are currently connected.

If you're not currently connected to a server, it returns $null.

$server(N/address)
Returns the address of the Nth server in your irc servers list.

Properties: desc, port, group

$server(0)	returns the total number of servers in the servers list
$server(2)	returns the address of the 2nd server
$server(2).desc	returns the description of the 2nd server
$server(3).port	returns the port(s) of the 3rd server

If you specify an irc server address and it is in your servers list, it returns its associated info.

$show
Returns $false if a command is prefixed with a . to make it quiet, otherwise returns $true.

$sreq
Returns current /sreq settings in the DCC Options send section dialog.

$url
Returns the **currently active** URL in your Web Browser.

$url(N)
Returns the Nth address in your URL list.

Properties: desc, group

$url(0)	returns the total number of items in the URL list

$url(2)	returns the address of the 2nd item in the list
$url(2).desc	returns the description of the 2nd item in the list
$url(3).group	returns the group of the 3rd item in the list

$usermode
Returns your current usermode on the irc server.

$version
Returns the version of mIRC that is being used.

Flood Protection

Flood protection attempts to prevent you from **flooding** a server with messages sent in **response** to requests from other users via CTCP or a script.

Flooding usually results in your being **disconnected** from IRC servers since they place a limit on how much information you can send at one time.

mIRC will count the number of bytes you send to a server, and will initiate a flood check if you exceed a certain maximum number of bytes.

Trigger flood check after...
This is the number of bytes at which mIRC should check if it might be flooding the server or not. Setting this greater than 500 bytes isn't too helpful since that might be the maximum amount a server allows. The lower the number, the more sensitive mIRC will be, and the slower it will reply. Default 400.

Max. lines in buffer
The maximum number of lines mIRC will buffer.

Max. lines per person
The maximum number of messages a user can have have in the buffer.

Ignore person for...
How long to ignore a user who has exceeded their maximum number of buffered messages. If zero, no ignore is done.

The **/flood** command also allows you to change these settings. Typing /flood with no parameters gives you the current flood status.

```
/flood 200 10 2 30
```

Here mIRC will check for flooding if it has sent 200 or more characters to the server, will buffer a maximum of 10 lines and ignore the rest, will only allow each user 2 buffered lines, and will ignore a user for 30 seconds if that user exceeded the maximum number of buffered messages.

The flood protection method also performs intelligent queuing in order to satisfy the maximum number of users as quickly as possible, so that no single user can hog the queue.

You can type **/flood on** to turn flood protection on with the default settings.

Bold, Underline, Reverse, and Colour

mIRC interprets control codes in text for **Bold, Underline, Reverse, and Colour** and displays text in the specified format.

You can use the following key combinations to insert control codes in text:

Control-B for **bold** text
Control-U for **underlined** text
Control-R for **reverse** text
Control-K for **coloured** text
Control-O for **plain** text

Examples

To **underline** a single word in a sentence:

1.Type Control-U
2.Type in the word
3.Type Control-U again

Only the text that is **enclosed** by the start and end codes will be affected. You can use this method with all of the other control codes.

The **Control-K** control code is slightly different because it allows you to specify a colour number. To **colour** a single word in a sentence:

1.Type Control-K
2.Type a number between 0 and 15
3.Type the word
4.Type Control-K again

If you also want to change the **background** colour of a word, you would need to type **two numbers** separated by a **comma** instead of just one number. The first number is the **text** colour, the second number is the **background** colour. The colours range from 0 to 15, and the indexes are:

0 white	8 yellow
1 black	9 lightgreen
2 blue	10 cyan
3 green	11 lightcyan
4 lightred	12 lightblue
5 brown	13 pink
6 purple	14 grey
7 orange	15 lightgrey

If you want to enclose **existing** text in control codes, just **select** the text with your cursor, and then type the Control code. This will insert both starting and ending control codes around the text you selected.

You can enclose text in multiple control codes, so for example you could have a bold, underlined, and coloured word.

You can use colour 99 to indicate a transparent colour.

Note: If you have the **pop up colour index** switch turned on in the Extras section of the options dialog, mIRC will pop up a small colour index showing you each colour and it's associated number so you don't have to memorize them.

If you want to strip out control codes that other people send you in private or channel messages, you can either change the strip settings in the IRC Switches dialog, or you can use the /strip command.

Accepting Files on IRC and the Internet in General

Sharing files on IRC is part of what makes IRC fun, however it's important to be **careful** about **who** you accept files from and **which** types of files you accept.

Although the **majority** of files are safe, there are always a few out there that may be infected with a virus, or may be malicious programs that try to damage your computer. Since it's impossible to know in advance whether a file that is being sent to you might cause a problem, following a few common sense rules usually helps to avoid problems:

1. Only accept files from people that you know and trust. You should **never** accept files from people you don't know, and **never** accept files without knowing what their purpose is.

2. Files ending in .BAT, .COM, .EXE, .DLL have the most potential to cause problems. You should **not** accept such files from people you don't know, or download them from web/ftp sites which don't appear trustworthy.

3. Aliases, Popups, or Scripts that can be **loaded** or **typed** into your IRC client can also cause problems. mIRC, and most other IRC clients, allow you to create scripts that perform **very** useful functions, but these can also cause problems if misused. You should make sure that you know and trust the source of these files before using them.

4. Certain types of Document files can contain **macros** which are run by your Word Processor when you open the document to view it, so these are also potentially harmful. You should make sure that you have macro-warnings turned on in your Word Processer. It is also usually safer to view any documents that you receive in a plain-text editor first if possible.

5. If you have an anti-virus program, you should use it to scan all files that you download **before** you use them. However, IRC is a highly interactive medium where information spreads very quickly, so using an anti-virus program does not guarantee that a file will be safe since it takes time for anti-virus programs to be updated.

Ctcp Events

CTCP stands for **Client-To-Client-Protocol** which is a special type of communication between IRC Clients. By creating CTCP events, you can make your mIRC react to commands or requests from other users. CTCP events use the format:

```
ctcp <level>:<matchtext>:<*|#|?>:<commands>
```

The **level** is the access level required to access this event, the **matchtext** is the actual CTCP being sent, the ***#?** specify whether to react to any message, to channel messages, or to private messages respectively, and the **commands** are the commands that will be performed if this event triggers successfully.

Examples

The following examples should give you an idea of how to create simple CTCP events.

A Basic CTCP event

```
ctcp 1:help:*:/msg $nick help yourself!
```

The above ctcp event would react to a **/ctcp yournick help** message either in a channel or private message. Since it has access level 1, this means that any user can access it because 1 is the lowest access level.

Giving Op status to a friend

```
=5:*!khaled@mardam.demon.co.uk
```

```
ctcp 5:opme:?:/mode $1 +o $nick
ctcp 5:inviteme:?:/invite $nick $1
```

These definitions would allow the above level 5 user to send you the ctcp **/ctcp yournick opme #mIRC** and if you are on an Op on channel #mIRC, the above script would automatically Op him. The user can also send you the ctcp **/ctcp yournick inviteme #mIRC**, and you would invite him to channel #mIRC.

Note: By giving the user access level =5, the user is limited only to level 5 events. If I had given the user access level 5, then the user would be able to access all events which have access level 5 **and** below.

Changing a standard CTCP reply

```
ctcp 1:ping:?:/notice $nick Ouch! | /halt
```

This will react to the standard ping CTCP and will reply with "Ouch!". The /halt at the end of the line prevents the standard ping reply from being sent. If you don't use the /halt, the standard reply to PING will be sent.

```
ctcp 1:time:?:/notice $nick The time here is around $time | /halt
```

This will react to the standard time CTCP and will reply with the above message. Again, the /halt prevents the standard time reply from being sent.

Note: You can't prevent the standard version reply from being sent.

Controlling your mIRC remotely

100:*!khaled@mardam.demon.co.uk

ctcp 100:quit:?:/notice $nick Okay boss, I'm quitting... see you later! | /quit

The above definition shows how you can give yourself a high access level to access the quit event, and you can tell your mIRC to quit IRC from another IRC Client.

ctcp 100:send:?:/dcc send $nick $1-

This definition allows you to ask your mIRC to send you whatever file you specify to the IRC Client you're using from another location, for example by using the ctcp **/ctcp yournick send homework.txt**.

ctcp 100:*:?:$1-

This event definition allows you to execute any command remotely. So if you send a **/ctcp yournick echo Hi!**, your script will execute the command **echo Hi!**. This is a potentially dangerous event definition since if you allow anyone else to access it, they will be able to perform any command they want on your computer.

Wildcards and Variables

ctcp 1:*help*:#:/notice $nick I can see that you need some help

By using the *** and ? wildcard characters**, you can match any text you want in a sentence. So if a user sends you a ctcp which has the word help in it anywhere, the above notice will be sent.

ctcp 1:%password:?:/notice $nick Your access has been authorized

By using variables in the matchtext section, you can change the value of %password whenever you want without having to change the event definition. So if you set %password to the value "moo" and someone sends you a "moo" ctcp, it will match %variable and the notice will the above message will be sent.

on SNOTICE

The **on SNOTICE** event triggers when you receive a server notice.

Format:	on <level>:SNOTICE:<matchtext>:<commands>
Example:	on 1:SNOTICE:*client connecting*:/halt

For an explanation of **matchtext** see the on TEXT event.

Examples

on 1:SNOTICE:*hack*:/splay hack.wav

This triggers when a server notice contains the word hack.

Note: You can prevent the default server notice from being displayed by using /halt.

on WALLOPS

The **on WALLOPS** event triggers when you receive a wallops message.

Format:	on <level>:WALLOPS:<matchtext>:<commands>
Example:	on 1:WALLOPS:*warning*:/echo $nick issued warning at $time

For an explanation of **matchtext** see the on TEXT event.

Examples

on 1:WALLOPS:*oink*:/splay oink.wav

This triggers when a wallops notice contains the word oink.

Note: You can prevent the default wallops mesage from being displayed by using /halt.

on OPEN/CLOSE

The **on OPEN** and **on CLOSE** events trigger for various events relating to the opening and closing of a window of different types of windows. In the case of dcc sessions, they trigger when a dcc connection has opened or closed.

Format:	on <level>:OPEN|CLOSE:<?|@|=|!|*>:<commands>
Example:	on 1:CLOSE:?:echo -s closed $target query window

Examples

on 1:OPEN:?:/echo -s Just opened $target query window

This triggers just before a query window is about to be opened because of an incoming private message. If you use /halt in this event, it prevents the window from being opened.

on 1:CLOSE:?:/echo -s you just closed $target query window

This triggers when you close a query window.

on 1:OPEN:=:/msg =$nick Hi! I'll be with you in a second...

This triggers when a dcc chat connection is first established. The equal sign in **=$nick** is required to send the reply as a dcc chat message. If no equal sign is used, the message is sent as a private irc server message.

on 1:CLOSE:=:/notice $me $nick just left the discussion!

This triggers when a dcc chat session ends, or when you close your chat window manually.

on 1:OPEN:!:/msg =$nick Welcome to my fileserver!
on 1:CLOSE:!:/echo -s $nick just ended her fileserver session

These trigger when a dcc filserver session is first established and when it is closed.

Note: DCC events react to all users level 1 and above because of the way DCCs work.

on 1:CLOSE:@:/echo -s Just closed $target custom window

The above triggers when a custom window is closed. The OPEN event does not trigger for custom windows.

Note: These event do not trigger for any other types of windows. Channel windows are handled by the JOIN and PART events.

on NOTIFY/UNOTIFY

The **on NOTIFY** and **on UNOTIFY** events trigger when a user in your notify list joins or leaves IRC.

Format:	on <level>:NOTIFY:<commands>
Example:	on 1:NOTIFY:/msg $nick Hiya! :)

Examples

```
on 1:NOTIFY:/msg $nick Hi! I'm in #mIRC_Lounge, come over!
```

This triggers when a user in your notify list joins IRC.

```
on 1:UNOTIFY:/notice $me $nick just left IRC *sniff*
```

This triggers when a user in your notify list leaves IRC.

on INVITE

The **on INVITE** event triggers when a user invites you to a channel.

Format:	on <level>:INVITE:<#[,#]>:<commands>
Example:	on 1:INVITE:#mIRC:/join $chan

Examples

on 2:INVITE:#:/join $chan | /timer 1 3 /describe $chan appears in a puff of smoke!

This triggers when a user with access level 2 invites you to any channel.

Note: If you want to automatically join a channel when someone invites you, it's easier to turn on the auto-join option in the Options dialog.

on CTCPREPLY

The **on CTCPREPLY** event triggers when a user sends a standard ctcp reply to a ctcp that you initiated.

Format:	on <level>:CTCPREPLY:<matchtext>:<commands>
Example:	on 1:CTCPREPLY:VERSION*:/echo $nick is using IRC client: $1-

For an explanation of **matchtext** see the on TEXT event.

Examples

on 1:CTCPREPLY:PING*:/echo $nick replied to my ping!

This triggers when a user replies to a ctcp PING that you sent.

on MODE

The **on MODE** event triggers when a user changes a channel mode.

Format:	on <level>:MODE:<#[,#]>:<commands>
Example:	on 1:MODE:#mIRC:/notice $me $nick changed $chan mode to $1-

The **on SERVERMODE** event uses the same format, and triggers when an IRC Server changes a channel mode.

Examples

on @1:MODE:#:/notice $me $nick changed $chan mode to $1-

This triggers when a user changes a mode on any channel where you have Ops (the @ at sign specifies the Op requirement, see the Access Levels section for more info). The actual parameters of the mode change are stored in **$1-** which you would need to parse yourself to enforce a particluar mode.

Note: This event only triggers on channel mode changes not user mode changes such as ops, bans, etc. It also does not trigger when **you** make a mode change.

on NICK

The **on NICK** event triggers when a user changes nickname while on the same channel as you.

Format:	on <level>:NICK:<commands>
Example:	on 1:NICK:/echo $newnick was previously know as $nick

Examples

```
on 1:NICK:/describe $newnick thinks $nick was a nicer nickname!
```

This triggers when a user changes their nickname on a channel.

on QUIT

The **on QUIT** event triggers when a user quits IRC while on the same channel as you.

Format:	on <level>:QUIT:<commands>
Example:	on 1:QUIT:/notice $me $nick just quit IRC with the message $1-

Examples

on 1:QUIT:/ame waves bye-bye to $nick *sniff*

This triggers when a user quits IRC while on the same channel as you. The **$1-** parameters hold the user's quit message.

on TOPIC

The **on TOPIC** event triggers when a user changes a channel topic.

Format:	on <level>:TOPIC:<#[,#]>:<commands>
Example:	on 1:TOPIC:#mIRC:/msg $chan Hmm, odd topic!

Examples

```
on 1:TOPIC:#mIRC4Dummies:/describe $chan admires $nick's new topic!
```

This triggers when a user changes the topic on channel #mIRC4Dummies. The **$1-** parameters hold the actual text of the new topic.

on NOSOUND

The **on NOSOUND** event triggers when a user sends a /sound request to play a sound and you don't have that sound.

Format:	on <level>:NOSOUND:<commands>
Example:	on 1:NOSOUND:/notice $me Oops, $nick has $filename and I don't!

Examples

```
on 1:NOSOUND:/msg $nick ! $+ $nick $filename
```

This triggers when you don't have the requested sound. **$filename** refers to the name of the file that was requested.

Note: This will trigger whether the warn if sound doesn't exist option is turned on or off.

on DNS

The **on DNS** event triggers when a /dns query either succeeds or fails.

Format:	on <level>:DNS:<commands>
Example:	on 1:DNS:/notice $me Resolved: $raddress

Examples

on 1:DNS:/echo $nick ip address: $iaddress named address: $naddress resolved address: $raddress

This is triggered when a /dns completes. If **/dns <nick>** was used to initiate the DNS lookup then **$nick** refers to the specified nickname otherwise it is $null. **$iaddress** refers to the ip address, and **$naddress** refers to the named address. **$raddress** is the resolved address. If **$raddress** is $null then that means that the address could not be resolved.

Note: This event is also triggered if you try to /dns a nickname, and the nickname is not on IRC.

on ERROR

The **on ERROR** event triggers when an IRC Server sends an ERROR message, this usually occurs on a disconnection.

Format:	on <level>:ERROR:<matchtext>:<commands>
Example:	on 1:ERROR:*server full*:/echo Try another server!

For an explanation of **matchtext** see the on TEXT event.

Examples

on 1:ERROR:*banned*:/echo I'm banned from this server *mumble*!

This triggers when you try to connect to a server and it tells you you are banned. The **$1-** parameters refer to the full error message.

Note: This event is not related to any kind of error reporting in mIRC itself.

on MIDIEND/WAVEEND

The **on MIDIEND** and **on WAVEEND** events trigger when mIRC finishes playing a midi or wave file.

Format:	on <level>:MIDIEND:<commands>
Example:	on 1:MIDIEND:/splay jazzy.mid

Examples

on 1:WAVEEND:/echo Finished playing wave file!

Triggers when a wave file finishes playing. You can use $inmidi and $inwave to test if a sound is currently playing.

Note: These events will not trigger if you use /splay to play another sound or to stop a sound being played, it only triggers if the sound finishes playing completely.

on INPUT

The **on INPUT** event triggers when you enter text in an editbox and press enter.

Format:	on <level>:INPUT:<*#?=!@>:<commands>
Example:	on 1:INPUT:#mIRC:/echo You entered the text " $1- " in the #mIRC window

Examples

on 1:INPUT:#:/echo I just mumbled " $1- " in a channel

Triggers when you enter text in an editbox in a channel window and press enter. The **$1-** parameters refer to the text that you entered. If you /halt this event, you can prevent mIRC itself from processing your message.

on 1:INPUT:?:/echo I just mumbled " $1- " in a query window
on 1:INPUT:=:/echo I just mumbled " $1- " in a dcc chat
on 1:INPUT:!:/echo I just mumbled " $1- " in a fileserver

Note: You can use commands such as /say with on INPUT and they will send the message to the window inwhich you're typing, however most commands/events don't work this way and require you to specify the actual destination of a message.

on USERMODE

The **on USERMODE** event triggers when you change your usermode.

Format:	on <level>:USERMODE:<commands>
Example:	on 1:USERMODE:/echo You changed your usermode to $1-

Examples

```
on 1:USERMODE:/echo Usermode for $nick is now $1-
```

Triggers when your usermode changes. The **$1-** parameters refer to the new usermode.

on LOAD/START

The **on LOAD** event triggers the first time a script file is ever loaded.

Format:	on <level>:LOAD:<commands>
Example:	on 1:LOAD:/echo mIRC Script Loaded!

The **on START** event uses the same format, and triggers the first time a script is ever loaded and also every time after that when scripts are loaded when mIRC is first run.

Examples

on 1:LOAD:/echo Performing one-time initialization for this script!

Triggers the first time a script is ever loaded. The purpose of this event is to perform a one-time initialization of settings.

on 1:START:/echo Performing regular initialization for this script!

Triggers the first time a script is ever loaded, and also every time after that when scripts are loaded when mIRC is first run. The purpose of this event is to perform general initialization settings.

Note: When a file is loaded in the remote dialog, it's initialization sections are not run until after the dialog is closed. Only **one** of each of these events is allowed in a script.

Chatting Privately

As well as being able to chat on public **channels**, mIRC also allows you to chat **privately** with someone.

If you're **on a channel**, and you see someone you'd like to chat with, you can **double-click** on their nickname in the nickname listbox and a private query window will open up. You can then start chatting privately to them through the query window. Alternatively, you can **click your right mouse button** in the nickname listbox and a **popup menu** will appear with various options, one of which will be to open a private query window to the person selected in the nickname listbox.

If you're **not on a channel**, you can type the command **/query nickname**, where nickname is the person you want to chat with. Press the enter key, and a query window will open up and you can start chatting privately, assuming of course that the person is on IRC. You can **find out** if a person is on IRC by using the **/whois nickname** command.

There is **another** way to chat privately called DCC Chat. This method is more secure and usually faster because it doesn't rely on the IRC Server to relay your messages. Instead it connects **directly** to the other person's IRC Client. However it does need to use the IRC Server to **initiate** the chat session.

To DCC Chat someone, you can click on the **Chat button** in the toolbar, and a DCC Chat dialog will pop up. Enter the person's nickname, and click on the Chat button, and if the persons accepts your DCC Chat request, you will be able to start talking to them privately. For more information about DCC Chat you should read the DCC Chat Section.

Changing Colours

mIRC uses **black text** on a **white background** by default because this allows **coloured** text to appear clearly and crisply. However, if you find the colour scheme too bright, or if you want something a bit more fun, you can change the text and background, as well as other specific types of text eg. your own messages, to other colours.

To change the colour settings, select the **Colours dialog** from the **Tools Menu**. It will popup up and show you the different types of messages in their current colour settings.

To **change** the colour of an **item**, just **click** on it and you'll see it's **name** appear in the **listbox** at the bottom. Then **click** on any of the **coloured squares**, and you'll see the colour of the text for that item change accordingly.

To **change** the **background colour**, just **click** on the background itself and select a colour as usual. You can also click on the background of the **editbox** and the **listbox** to change their colours as well.

If you don't like the changes you've made you can either click **cancel** or you can click the **reset** button which will reset the colours back to the **default** mIRC colours.

To **customize** the colour of one of the colour boxes, just click your **right mouse button** on it and a custom colour dialog will pop up allowing you to choose a new colour for it.

Once you've **finished** making changes, click on the **Ok** button, and hey presto you should now feel a bit more colourful!

mIRC also allows you to **interactively** change the colour, as well as the **appearance**, of text as you type. You can find out how to do this in the Bold, Underline, Reverse and Colour section. However remember that other people might have a background colour that's **different** from yours, so what appears clearly on your background colour might be hard to read on someone else's.

Custom Windows

The **/window** command, along with a few **other** related commands listed below, allows you to **create** and **manipulate** custom windows. It's format is:

/window [-abcdefhl[N]noprsx] [-tN,..,N] [+bfmnstx] <@name> [x y [w h]] [/command] [popup.txt] [font [size]]

Switches and Parameters

You can specify the following switches and parameters either when first creating a window or to manipulate an existing window.

The **first** set of switches:

a = activate window
b = update horizontal scrollbar width for listbox
c = close window
d = open as desktop window
e = editbox
f = indicates that w h are the required width and height of the
    text display area as opposed to the window size
h = hide the window (only appears in Window list)
l[N] = listbox, if N is specified then a side-listbox N characters wide is created
n = minimize window
o = if opened on desktop, place ontop
p = creates a picture window
r = restore window
s = sort the main window, whether text or listbox
S = sort the side-listbox
u = remove ontop setting of a desktop window
x = maximize window

The **-t** switch used to set the **tab positions** in a listbox.

t[N,...,N]= specifies the tab positions in a listbox, if text contains tabs it will be
    spaced out according to these tab settings

The **third** set of switches is used to change the **appearance** of a window.

b	= border
f	= dialog frame
n	= minimize box
s	= sizable
t	= titlebar
x	= maximize box

Note: Some switches may automatically turn others on/off.

@name	window name (must prefix with a @)
x,y,w,h	left top width height
popup.txt	popup filename, loaded when needed (must be a plain non-ini text file)
/command	default command (executed whenever you enter text in an editbox)
font/size	font name and size (defaults to status window font)

If you want the custom window to use a menu definition in a remote script, you can specify the custom window's name as the popup filename instead of an actual popup.txt filename.

Commands

The following commands allow you to manipulate the **lines** in a custom window.

/aline [c] <@name> <text>	add line to list
/cline [c] <@name> <N>	changes the colour of the Nth line to colour C
/dline [c] <@name> <N[-N2]>	delete Nth line through N2th line
/iline [c] <@name> <N> <text>	insert line at Nth line
/rline [c] <@name> <N> <text>	replace Nth line
/sline [c] <@name> <N>	select Nth line

The **c** parameter specifes a colour number for the line.
The **-s** switch selects the line that was just added and clears the current selections.
The **-a** switch selects a line without clearing the current selections.
The **-h** switch highlights the window's icon if it's currently minimized.
The **-p** switch forces the line of text to be wrapped if it's too long to fit on one line.

If you are referencing a window which uses a side-listbox, you can specify the **-l** switch in the above commands to act on the side-listbox.

/renwin <@oldname> <@newname> [topic]
This allows you to change the name of an existing window to a different one, and an optional topic can be specified.

Identifiers

The following identifiers return custom window information.

$window(N/@name)
Returns properties for a window.

Properties: x,y,w,h,dx,dy,dw,dh,bw,bh,mdi,title,state,font,fontsize

x,y,w,h	return the left, top positions, and the width and height of the window repsectively.
dx,dy	return the left, top positions of the window relative to the desktop.
dw,dh	return the width and height of the text display area.
bw,bh	return the width and height of the bitmap for a graphic window.
mdi	returns $true if the window is mdi, otherwise returns $false
state	returns minimized/maximized/normal/hidden
title	returns the text in the titlebar of the window
font	returns the name of the current font
fontsize	returns the size of the current font

Note: You can also get the .w and .h properties by specifying: -1 for the width and height of the screen, -2 for the main mIRC window, and -3 for the MDI window where all other windows inside mIRC are displayed.

$line(@name,N,T)
Returns the Nth line of text in the specified window. If N is zero, it returns the total number of lines in the window.

If you are referencing a window which uses a side-listbox, you can set T to zero to reference the display area, or set T to one to reference the side-listbox.

$fline(@,wildtext,N,T)
Returns Nth position of line which matches the specified wildcard text. If T is not zero, it acts on the side-listbox.

$sline(@name,N)
Returns the Nth selected line in a listbox (only works in listboxes). If N is zero, it returns the total number of selected lines in the window.

Properties: ln

$sline(@name,N).ln	returns the **line number** of the Nth selected line

Examples

/window @test

This would create a window named @test with all of the default settings.

/window -els @clones 10 10 clones.txt

This would create a window with a sorted listbox and an editbox, positioned near the top left of the mIRC window, and using the popup file clones.txt which would appear whenever you click the right mouse button in the listbox.

Note: The popup text file must be plain text in a non-ini format using a non-ini extension.

Connection Problems

The most **common** problems related to connecting to an IRC Server are described below.

Unable to resolve IRC Server

If you try to connect to a server and see this message, the problem could be:

Your Internet Provider's DNS isn't working
This happens occasionally and is a temporary problem with your Internet Provider. This problem would also result in your being unable to connect to other internet services such as web sites. You should try again later.

An Invalid or Non-working IRC Server address
You might be trying to connect to an IRC Server that is currently not working, or perhaps is an old address and doesn't exist anymore. You should try another IRC Server.

Trying to use the 32bit mIRC with a 16bit winsock
Try the 16bit mIRC to see if that solves the problem. If you are running a 32bit operating system, you may have an old version of winsock.dll on your machine. This causes all 32bit internet applications to stop working. You should have a file called winsock.dll,   dated 1995-07-11 or later, probably 42k in size.

Unable to connect to IRC Server

If you try to connect to a server and see this message, the problem could be:

IRC Server isn't working
You might be trying to connect to an IRC Server that is currently not working. This is the most likely problem. You should try another IRC Server.

Invalid Port number
The IRC Server address might be correct but you've specified the wrong port. Most servers operate at least on port 6667, you should try that port to see if it solves the problem.

Other messages

If you try to connect to a server and get **Disconnected** and see the message **Closing Link** followed by a comment such as **No Authorization**, or **No More Connections**, etc., it might be that you're too far away geographically from that server, or that the server is full and can't handle anymore users, or there may be other reasons. You should try a different, closer IRC Server until one works for you.

Note: If you try to connect to a server and you get **Unable to resolve local host** then see the Local Info section.

Example Script

The following example script shows you how you can place related aliases, popups, and events in a single file making it easier to distribute a whole script to other people.

```
;Moo Script v1.0 - contains moo related functions

;This menu definition adds a submenu to your channel popup menu

menu channel {
   Moo
   .happily:/describe # moos happily
   .woefully:/describe # moos woefully
   .philosophically:/describe # MUs
   .colourfully:/describe # moos in several hues
}

;These add aliases for shortcuts to often used messages

alias how /msg $1 How now brown cow?
alias moo /sound moo.wav mooooos

;This adds a ctcp command which reacts to a moo ctcp from someone

ctcp 1:moo:/notice $nick Sorry, I'm all out of moos right now.

;These add events which react to specific words said on a channel

on 1:text:*moo*:#:/msg $chan okay, who let the cow loose?
on 1:text:*grass*:#:/describe $chan dribbles hungrily

;These add join and part events which react to a user joining/parting
;the channel #moo

on 1:join:#moo:{
   /msg $nick Welcome $nick to channel #moo!
   /msg $nick This is a herd-oriented channel, there are calfs present!
   /msg $nick Please refrain from profaine mooing and/or bleating
   /msg $nick Mammals enaging in such acts will be promptly demooted
}

on 1:part:#moo:/msg $nick Thanks for grazing with us on #moo!

;The following line is processed while you're doing a channels list. It
;prints to the status window any channel name/topic that has the
;word moo in it

raw 322:*moo*:/echo -s $2-
```

on DCCSERVER

This event triggers when someone tries to connect to your DCC Server. The purpose of this event is to allow you to monitor connections and to prevent someone from connecting to your server by using /halt.

Format:	on <level>:DCCSERVER:<Chat|Send|Fserve>:<commands>
Example:	on 1:DCCSERVER:Send:echo $nick $address wants to send you $filename

Examples

```
on 1:DCCSERVER:Chat:/echo $nick $address wants to chat with you!
```

```
on 1:DCCSERVER:Send:if (.exe isin $filename) /halt
```

The above event checks triggers when someone wants to send you a file, and if the file they're sending ends in .exe, it cancels the send by using the /halt command.

Address Book

The address book allows you to store information about your friends on IRC. It consists of the main addresses section, the Whois results section, and the Notify section, described below.

The address book can be displayed by accessing it through the toolbar or by using the /abook [nickname] command. This command pops up the address book, and if a nickname is specified it shows the information for that nickname.

You also can press the **Alt-B** key combination to pop up the address book.

Addresses

The address book allows you to save information about your friends on IRC, such as name, email address, personal website, IP Address, and other notes.

Nickname

This is the persons usual nickname on IRC. Information in the Address Book is stored according to nickname.

Real Name

This is the persons real name.

Email

This is the persons email address. If you click on the **Email** button, mIRC will start up your email program (assuming that there is an email program registered on your system).

Note: The address in the /whois dialog may not be an actual email address, it is mainly an indication of the users internet provider.

Website

This is the persons personal website if they have one. If you click on the **View** button, mIRC will start up your web browser.

IP Address

This is the persons IP Address. If you click the Chat button, mIRC will initiate a DCC Chat directly to this persons DCC Server instead of using the IRC Server.

Note: Because of the way certain internet providers work, a person may not always have the same IP Address.

Notes

The notes section allows you to enter miscellaneous information about a person.

Picture

If the person sent you their picture, you can associate it with their nickname in the address book, and it will be displayed whenever their information is displayed.

Note: The picture must be in the standard BMP format, mIRC cannot handle any other picture formats at this time.

Whois

The Whois section displays the result of the /uwho command which looks up the information on a user from the IRC server that you are on. The format of the /uwho command is:

/uwho [nick] [nick]

This performs a /whois on the specified nickname to look up their server information and then displays it in the /whois section of the address book. Because of the way IRC Servers work, you may need to specify the nickname a second time in order to look up information such as idle time or the away message, however this information usually takes far longer to retrieve.

Notify List

The notify list allows you to specify nicknames which you want mIRC to look for when you're connected to an IRC Server. mIRC will **notify** you whenever a nickname **joins** or **leaves** the IRC network you are on. You will also be informed if no users in your list are on IRC when you **first** connect.

mIRC checks the server and updates its notify list every **40 seconds or so** thereafter.

Adding a User

To add a user you can enter the following information and then click on the **Add** button.

Nickname

The nickname of a user that you want notify to look for.

Note

An optional note or reminder that will appear next to the users nickname.

Play sound

The sound that you want played whenever the user joins IRC.

Perform /whois

This option makes mIRC perform a **/whois** on the user when they join IRC to look up their address. You should only use this option if you really need to, if you use it with too many nicknames then the IRC server might disconnect you for flooding.

Add/Update/Delete

To add or delete users from the list. After you make a change to the settings of an existing user, you must click the **Update** button.

Show notifies in active window

The default is to show notifies in the status window, however checking this option will also show notifications in the current active window.

Only show notifies in notify window

This will make mIRC display notifies only in the notify window.

Pop up notify window on connect

This will pop up the notify list window when you connect to an irc server.

The /notify command

You can also add/remove nicknames from the notify list by using the /notify command.

/notify [-shr] <on|off|nickname> [note]

You can turn notify on and off by typing **/notify on or off** respectively.

The **-sh** switches can be used to **show or hide** the notify list window respectively

The **-r** switch removes the specified nickname from your notify list.

The **note** is optional and allows you to specify a little note for each nickname.

If you prefix a nickname with a **+** sign then mIRC will do a **/whois** on the nickname as part of the notify. However, if you do this on too many nicknames then the IRC server might disconnect you for flooding, so it's best to use it only if you really need to.

You can manually force mIRC to update the notify list by typing **/notify** with no parameters.

Note: Some IRC networks might let you use a full address instead of just a nickname, the only way to see if it works is to try it out.

DCC Server Protocol

Chat Protocol
Client connects to Server and sends:
100 clientnickname
When Server receives this, it sends:
101 servernickname
Connection is established, users can now chat.

Fserve Protocol
Client connects to Server and sends:
110 clientnickname
When Server receives this, it sends:
111 servernickname
Connection is established, user can now access fserve.

Send Protocol
Client connects to Server and sends:
120 clientnickname filesize filename
When Server receives this, it sends:
121 servernickname resumeposition
Where resumeposition is between 0 and filesize, and is required.
Connection is established, and Server dcc gets the file.

Get Protocol
Client connects to Server and sends:
130 clientnickname filename
When Server receives this, it sends:
131 servernickname filesize
When Client receives this, it sends:
132 clientnickname resumeposition
Where resumeposition is between 0 and filesize, and is required.
Connection is established, and Server dcc sends the file.

Other
If server receives unexpected information, or doesn't recieve info 15 seconds after initial connection, it closes the connection.

If service is unavailable, server sends:
150 unavailable

If server rejects connection, it sends:
151 rejected

Notes:
The Get protocol has been implemented in this way mainly because I'm assuming:
1) The client may not be able to open a socket to listen for and accept a connection (firewall etc.)
2) The DCC Server may only be able to listen for connections on port 59 (firewall etc.)
3) Since the client was able to connect to the DCC Server the first time, it should have no problem connecting to the same port again.

Currently the Get Protocol is **only** used by the Fileserver when a user who has connected to a Fileserver via the DCC Server requests a "get filename". All other attempts to Get a file via the DCC

Server are ignored.

Binary Files

mIRC allows you to read and write **binary files**, and to modify binary variables, by using the following commands and identifiers.

/bread <filename> <S> <N> <&binvar>
This reads N bytes starting at the Sth byte position in the file and stores the result in the binary variable &binvar.

/bwrite <filename> <S> [N] <text|%var|&binvar>
This writes N bytes from the specified text, %var, or &binvar, to the file starting a the Sth byte position. Any existing information at this position in the file is overwritten.

Note: If S is -1, the bytes are appended to the end of the file.

/bset <&binvar> <N> <asciivalue>
This sets the Nth byte in binary variable &binvar to the specified ascii value.

If you try to /bset a variable that doesnt exist, it is created and zero filled up to N bytes. If &binvar exists and you specify an N position beyond it's current size, it is extended to N bytes.

$bvar(&binvar,N)
Returns the ascii number of the Nth byte. If N is 0, it returns the length of the binary variable.

Notes on &binvar binary variables
Binary variables have a maximum size of 4096 bytes, can **only** be accessed by commands /bread and /bwrite, so they can't be printed or assigned or edited, and are automatically destroyed when a script finishes processing.

Sockets

Socket support allows you to create your own raw socket connections in order to send and receive information. You should already be an **expert** at writing Aliases, Popups, and Scripts before attempting to use sockets.

Sockets are a **limited resource** so it is important that you understand how these commands work before trying to use them. Sockets should always be closed after they have been used to make them available to other applications.

Socket Identifiers

$sock(name,N)
This returns information about a socket connection that you created using the socket commands.

Properties: name, port, ip, status, sent, rcvd, sq, rq, ls, lr, mark

.name is the name you give to a connection to identify it.
.sent and **.rcvd** return the number of bytes sent and rcvd over that connection so far.
.sq and **.rq** return the number of bytes queued in the send and receive buffers respectively.
.ls and **.lr** return the number of seconds since the connection last sent and last received info.
.mark is a user storage area max. 512 bytes (see /sockmark).

Note: you can use a wildcard name to quickly reference matching entries. The N parameter is optional, if it isn't specified it is assumed to be 1.

$sockname
$sockname is the name given to a connection to identify it. This identifier can be used in events to know which connection an event is related to.

$sockerr
$sockerr is set to a value after each socket command/event and **must** be checked after each socket command and before processing an event to see if an error occurred.

$sockbr
$sockbr is set to the number of bytes read by a /sockread command. It is used to test whether any information was in fact read from the buffer (see below for more info).

$portfree(N)
Returns $true if the specified port number is not in use, otherwise returns $false.

Socket Commands and Events

The following information lists associated commands and script events together for easy reference.

Listening for and Accepting incoming connections

/socklisten <name> [port]
The /socklisten command listens on the specified port for connections to that port. If a port isn't specified, the port is selected randomly from the range specified in the DCC Options dcc ports section.

on 1:socklisten:name:commands
The socklisten event is triggered when someone tries to connect to a port that you are listening on. If you want to accept the connection you **must** do it in this event using the /sockaccept command,

otherwise the connection is closed.

/sockaccept <name>
The /sockaccept command accepts the current connection to your listening port and assigns it a name to identify it.

/sockrename <name> <newname>
The /sockrename command assigns a new name to an existing connection.

Opening and Closing connections

/sockopen <name> <address> <port>
The /sockopen command initiates a connection to the specified address and port. You can specify either an ip address or a named address (which will be resolved to an ip address).

on 1:sockopen:name:commands
The sockopen event is triggered when a /sockopen command is successful and a connection has been made.

/sockclose <name>
The /sockclose command closes the connection with the specified name. If you specify a wildcard name, all connections that match the wildcard are closed.

on 1:sockclose:name:commands
The sockclose event is triggered when a connection is closed by the remote connection (not you).

Sending information

/sockwrite [-tn] <name> <text|%var|&binvar>
The /sockwrite command queues info to be sent on the specified connection. mIRC will then try to send that info as quickly as it can. Once it has finished sending the info, it triggers the on sockwrite event so you can send more info if you need to.

If you specify the -t switch, it forces mIRC to send anything beginning with a & as normal text instead of interpreting it as a binary variable. The -n switch appends a CRLF to the line being sent if it's not a &binvar and if it doesn't already have a CRLF.

Note: you can use a wildcard name to send the same information at once to all connections that match the wildcard.

on 1:sockwrite:name:commands
The sockwrite event is triggered when mIRC has finished sending all of the info which you previously queued for sending.

Note: if you try to /sockwrite while there is still info queued in the send buffer, your new info will just be added to the end of the queue up to a maximum of 8192 bytes. Any attempt to queue more than that will result in an error message, so you should check how much info is currently queued by using $sock().sq (send queue) before trying to queue info on a socket.

Reading information

on 1:sockread:name:commands
The sockread event is triggered when there is info waiting to be read on the specified connection. You can read this info using the /sockread command.

Note: If this event triggers but no /sockread is performed to attempt to read the buffer, it is assumed that no script exists that is handling this buffer, so it is cleared and the info it contained is lost.

/sockread [-f] [numbytes] <%var|&binvar>
The /sockread command reads bytes from the receive buffer into the specified variable.

If you specify a %var variable, a line of text ending with a Carriage Return/LineFeed is read into %var. The CRLF are stripped off (this may result in %var being $null if the line only consisted of CRLF).

If you specify a &binvar then [numbytes] of info is read into the binary variable. If no [numbytes] is specified it defaults to 4096 bytes.

If you specify the **-f** switch with a %var variable, this forces mIRC to fill the %var variable with whatever text is in the receive buffer, even if it doesn't end in a CR/LF.

Note: a single /sockread may not be enough to read the entire buffer. You should keep reading until $sockbr (bytes read) is set to zero. This is far faster than letting mIRC re-trigger the event. If your script doesn't read the whole buffer, the on sockread event is re-triggered if:
   a) you were reading into a &binvar.
   b) you were reading into a %var and there is still a CRLF terminated line in the buffer waiting to be read.

Example:
This example shows you how you should process a sockread event. The socket has already been opened and has received information, so the sockread event is triggered. The socket name is **testing**. There is an explanation of each step below the sample script.

```
on 1:sockread:testing:{
  if ($sockerr > 0) return
  :nextread
  sockread %temp
  if ($sockbr == 0) return
  if (%temp == $null) %temp = -
  echo 4 %temp
  goto nextread
}
```

If **$sockerr** is larger than zero then there is a socket error. mIRC will automatically close the socket, so all you have to do is **return** from the event.

sockread %temp reads a CRLF terminated line of text and stores it in %temp. If the buffer did not contain a CRLF terminated line, %temp is not filled with anything, and $sockbr returns zero, so you should just return from the event without further processing.

If **%temp is $null** then that means the line consisted only of a CRLF which mIRC has automatically stripped out of the line, so only an empty line remains. In this case, I'm setting %temp to a dash to represent an empty line but you can do whatever you wish here.

I then **echo** the final line that was read to the status window.

Finally, I use **goto** to jump back and to continue reading remaining lines in the socket's receive buffer.

Marking a socket

/sockmark <name> [text]
The /sockmark command fills the .mark attribute of a socket with the specified info for later reference via the $sock().mark property. If you do not specify any text, the mark is cleared. The mark can hold up to

512 bytes.

Note: you can use a wildcard name to set the same information at once for all connections that match the wildcard.

Picture windows

The **picture window** is a special type of custom window that can display a combination of text, graphics, and pictures, and can trigger script events relating to mouse clicks and movements.

Once you have opened a picture window using the /window command, you can use the following commands and identifiers to draw and monitor activity in this window.

Drawing commands

/drawdot -ihnr @ <colour> <size> <x y> [<x y>...]
Draws a dot using the specified colour and size at the x,y co-ordinates. Multiple co-ordinates can be provided.

The **-i** switch draws in inverse mode.

The **-h** switch highlites the windows icon if it's minimized.

The **-n** switch prevents the display from being updated immediately. This allows you to make changes to the window in the background and then display the results only when you've finished. You can update the display by using any of the /draw commands with only the window name specified.

The **-r** switch indicates that the colour is in RGB format. You can use $rgb(N,N,N) to create an RGB value.

The whole window can be cleared by using the /clear command, eg. /clear @name. You can also specify the **-n** switch in /clear to delay the effect as described above.

/drawline -ihnr @ <colour> <size> <x y> <x y> [<x y>...]
Draws a line from the first <x y> co-ordinate to the second, if more co-ordinates are specified, the line is continued. The switches are the same as those in /drawdot.

/drawrect -ihnrfec @ <colour> <size> <x y w h> [<x y w h>...]
Draws a rectangle at the specified co-ordinates of the specified width and height. If more co-ordinates are specified these are also drawn as separate rectangles.

The **-f** switch fills the rectangle with the current colour.

The **-e** switch draws an Ellipse instead of a rectangle.

The **-c** switch draws a focus rectangle.

The remaining switches are the same as those in /drawdot.

/drawfill -ihnrs @ <colour> <colour> <x y> [filename] [<x y>...]
Fills an area with the specified colour starting at the specified co-ordinates.

The **-s** switches indicates that the second colour parameter is the colour that should be filled (surface fill). If no **-s** is specified, the second colour is the border colour at which filling should stop (border fill).

The optional **[filename]** specifies a bitmap .BMP file that is 8 by 8 pixels in size and is used as a fill pattern.

The remaining switches are the same as those in /drawdot.

/drawtext -hnrpboc @ <colour> [colour] [fontname fontsize] <x y [w h]> <text>
Draws text at the specified co-ordinates.

The **-p** switch processes and interprets colour/bold/etc. codes in the text.

The **-b** switch indicates that you have specified the second colour parameter as the background colour for the text.

The **-o** switch means the specified font should be in bold.

The **-c** switch means that you have specified the [w h] values as the rectangle in which text should be printed. Text will be clipped if it extends beyond this rectangle.

The remaining switches are the same as those in /drawdot.

/drawscroll -hn @ <x> <y> <x y w h> [<x> <y> <x y w h>...]
Scrolls the region inside the specified rectangle. The first <x> and <y> parameters represent the distance to scroll and can be positive or negative.

The remaining switches are the same as those in /drawdot.

/drawcopy -ihnt @ [colour] <x y w h> @ <x y [w h]>
Copies part of a picture to a different position in the same window or to another window. If the second [w h] parameters are specified, the picture is stretched/squeezed to fit.

The **-t** switch indicates that you have specified the [colour] RGB value as a transparent colour in the source bitmap.

The remaining switches are the same as those in /drawdot.

/drawpic -ihntsc @ [colour] <x y [w h]> [x y w h] <filename>
Loads and draws a Bitmap .BMP picture at the specified co-ordinates. If the first [w h] are specified, it is stretched or squeezed to fit. The second [x y w h] rectangle specifies which portion of the loaded bitmap should be displayed ie. a bitmap could contain multiple pictures.

The **-t** switch indicates that you have specified the [colour] RGB value as a transparent colour in the specified bitmap.

The **-s** switch indicates that you have specified the first [w h] parameters (as explained above) to squeeze/stretch the bitmap.

The **-c** switch indicates that the bitmap should be **cached**. This greatly speeds up subsequent references to this bitmap. If you specify **-c** and the bitmap is already in the cache, it is loaded and used from the cache, otherwise it is reloaded from the file. You can clear the entire cache with /drawpic -c.

If you try to load and cache a bitmap and there are already 30 bitmaps in the cache, the bitmap with the lowest reference count is freed and replaced by the new bitmap.

If you try to load a bitmap and there isn't enough memory, mIRC repeatedly frees the least referenced bitmap and tries to load again.

The remaining switches are the same as those in /drawdot.

Events and Identifiers

Mouse events can be defined in a scripts menu definition.

```
menu @test {
   mouse:/echo mouse moved at $mouse.x $mouse.y
   sclick:/echo single click at $mouse.x $mouse.y
   dclick:/echo double click at $mouse.x $mouse.y
   uclick:/echo mouse released at $mouse.x $mouse.y
   drop:/echo drag and drop at $mouse.x $mouse.y
}
```

The **mouse** event is triggered when you move a mouse inside a picture window. You can use the **$mouse** identifier (see below) for the x and y position of the mouse.

The **sclick** event is triggered when you click once inside a picture window. It will also trigger if you double-click.

The **dclick** event is triggered when you double-click inside a picture window.

The **drop** event is triggered if you clicked in the window, held the button down, moved the mouse, and then let go of the button.

$mouse
Returns the x, y position of the current mouse event, and whether the left mouse button, shift key, and/or control key are pressed.

Properties: win, x, y, mx, my, dx, dy, key

The $mouse identifier can be used in the mouse/click events. For $mouse.key you can use the **&** bitwise operator to check if the left button, shift key, and/or control key are pressed:

```
if ($mouse.key & 1) echo left button is pressed.
if ($mouse.key & 2) echo shift key is pressed.
if ($mouse.key & 4) echo control key is pressed.
```

The following properties can be used:

The **.win** property returns the name of the active window.

The **.x/.y**, **.mx/.my**, and **.dx/.dy** properties return the x and y position of the mouse relative to the active window, the main mIRC window, and the desktop respectively.

$click(@,N)
This stores a history of x,y clicks for a window.

Properties: x, y

You can use /clear -c @name to clear the history of clicks for a window. If you use $click() with no properties it returns x y.

$inrect(x,y,x,y,w,h)
Returns $true if the point x y is inside the specified rectangle, and $false if it isn't.

$inpoly(x,y,a1,a2,b1,b2,...)
Returns $true if the point x y is inside the polygon defined by the specified points, and $false if it isn't.

$rgb(N,N,N)

Returns an RGB colour value for use in /draw commands. If you use only **one** parameter, it assumes it is an actual RGB colour value and returns N,N,N.

$getdot(@,x,y)
Returns the RGB colour value of the dot at the specified position.

$height(text,font,size)
Returns height of text in pixels for the specified font.

$pic(filename)
Returns the size, width, and height of a bitmap file.

Properties: size, width, height

$width(text,font,size,B,C)
Returns width of text in pixels for the specified font.

If **B** is non-zero, the font is bold, if **C** is non-zero, control codes are processed.

Halting default text

mIRC displays its own default text for various types of IRC Server events, such as users joining or parting a channel, however you can modify or suppress this by using a script.

The ^ event prefix

You can prevent the default text for an event from being shown by using the ^ prefix in an event definition. This allows you to show your own custom event messages.

```
on ^1:JOIN:#:echo $chan Joins: $nick | halt
```

This line is triggered by a JOIN event and shows your own custom join event message, /halt prevents the normal message from being shown.

The ^ events **don't replace** your existing events; your normal events are independent and are **still** processed whether there is a ^ event in a script or not.

If you only want to halt the default text without /halting the entire script, you can use the **/haltdef** command.

You can check if a script has already halted the default text by using the **$halted** identifier; it returns $true if a user has used /halt or /haltdef in a ^ event, and $false if not.

The ^ event prefix currently works **only** on the following types of events: BAN, CHAT, DEHELP, DEOP, DEVOICE, HELP, INVITE, JOIN, KICK, MODE, NICK, NOTICE, OP, PART, TEXT, UNBAN, USERMODE, VOICE, QUIT, SERV, SERVERMODE, SNOTICE, TOPIC, WALLOPS.

Note: Halting the default text for an event affects how mIRC displays the most basic information about IRC events to a user, so it should be used carefully.