mIRC v4.72 (released 9th January 1997)

Welcome to mIRC v4.72, an Internet Relay Chat Client.

Please read through these sections **before** asking questions!

Overview

mIRC **attempts** to provide a user-friendly interface for use with the **Internet Relay Chat** network.

mIRC has the **following** features:

- Coloured text to ease reading.
- Uncluttered display.
- Comprehensive Alias definitions.
- Configurable multi-level popup menus.
- DCC Send/Get/Chat/Resume.
- Scripting commands.
- Finger client.
- Remote CTCP tool.
- Simple Event handler.
- Some basic fonts.
- Ident server.
- Simple File-server.
- URL Catcher.
- Command history.
- Function Key definitions.
- Toolbar with tooltips.
- Finger server.
- A handful of options.
- Many other useful bits and pieces.

The various parts of this client have been designed with the **aim** of **simplifying** and **speeding** an IRC session.

File Menu

Connect

Setup
- IRC Servers
- Local Info
- Options
- Identd
- Firewall

Options
- IRC Switches
- Notify List
- Control
- Double-click
- Event Beeps
- Drag-Drop
- Sound Requests
- URL Catcher
- Perform
- Flood
- Logging
- Servers
- Extras

Exit

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.

Show about dialog on startup
If enabled, the about dialog pops up 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. mIRC can do nothing about this.

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 Menu

mIRC supports DCC (Direct Client To Client) Send, Get, and Chat which allows it to make a direct connection to another client, bypassing the IRC network. mIRC also has a DCC Resume capability which allows you to resume a file transfer in the event that it was interrupted.

DCC Send
DCC Get
DCC Resume
DCC Chat
DCC Options

On a related note, check out the File Server.

NOTE:
DCC Send and DCC Chat need to use your local host 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 host and IP address explanations for more information.

Window Menu

The Tile, Cascade, and Arrange Icons menu items work the same way they usually do in windows.

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** (the normal way) 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.

Beneath these is shown a list of currently open MDI windows.

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 main **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 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.

Contents

Where you are.

About

Where I am.

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.

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, and one bouncing dot. Where oh where can they be? :)

Other things I can't remember...!

The Author (and part-time Human Bean)

Greetings! :)

I am **Khaled Mardam-Bey**, residing at **68 Melbury Court, London W8-6NJ, United Kingdom**. I can be reached via email at **khaled@mardam.demon.co.uk**, and you can see my homepage at **http://www.mirc.co.uk/khaled**.

Well, it's been around **two** years since I started working on mIRC, and in that time it has grown **far** beyond what I expected, especially for something which was only meant to be a **hobby**! I've had many requests asking me to **continue** working on it and letters of **support** (thanks :) in a **variety** of languages that I decided to continue with it.

However, since I can't afford to work on it anymore without some kind of financial support, I've also decided to make mIRC **shareware**. I really hope that the original **spirit** and **intention** with which mIRC was **created** and **given** isn't diminished by this. It's purpose is still to help people communicate a little better.

I've stuck a dollar **amount** on mIRC which I hope is fair and which I hope **most** people can afford. If you like mIRC please **register** as that will support my work on it. Thanks :)

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 peoples lives.

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

Khaled

The License

mIRC® v4.72 Internet Relay Chat Client

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 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. please contact me for permission.

All **commercial** use interests in mIRC 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 be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Khaled Mardam-Bey has been advised of the possibility of such damages.

mIRC is a registered trademark of mIRC Co. Ltd.

Double-click

You can 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 use:

/query $1

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

Aliases

mIRC allows you to define aliases and scripts to speed up your IRC session or to perform repetitive functions more easily. Each unique alias must be entered on a single line. See the following examples to understand how they work. To use these aliases you must know some IRC commands.

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

Some examples:

/gb /join #gb

Typing "/gb" is now the same as typing "/join #gb".

/j /join $1

We have now added a parameter string. If we type "/j #gb", then that's 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 the client should ask you for that parameter. The parameter you supply will be inserted in the line at that point. So if you type "/jj", a dialog pops 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 the parameter is given. If

you give 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 * 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".

If you had defined the alias as "/slap /me slaps $1 around with $2", then the final command line would have been "/me slaps Sheepy around with a".

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 funnyguy, the final command would be "/me laughs at funnyguy's joke".

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

Parameters are normally separated by a space. To prevent mIRC from inserting a space before appending the next parameter you can use $+. The above line will send "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 "|" (the shifted character usually under the \ key). So to write an alias that kicks and bans someone:

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

If you want greater control over the order of evaluation of identifiers, you can use the [and] brackets. Identifiers within 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 ] ]
```

You can also create multiple-line definitions by using { } brackets.

```
/ofcourse {
  /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 following example does the same as the above except it uses the **goto** command to jump to different points in the script.

```
/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 easily 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 how variables work, 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(4,21)
```

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.

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 that 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 a command with a . (period). For example:

/ignore somenick

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

/.ignore somenick

Popups

mIRC allows you to create **four** different types of popup menu: one for the status window, one for query/chat windows, one for channel windows, and one for the listbox where users nicknames on the current channel are listed. To use these you must know some Basic IRC commands.

If you press the right mouse-button, the popup menu 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. In fact, each menu-item is associated with a set of aliases, just like the alias window. Note that the popup menu-item **can** call the commands defined in the alias window.

You can create as many levels of popup menus as you want. ie. menus within menus.

To understand how popup menus can be defined, see the following examples.

On a single line, enter:

Get Help:/join #irchelp

The words before the ":" (colon) are the name of the menu-item. The words after the ":" are the commands that are to be performed. In this case, the menu-item you would see when you click the right mouse-button in the main channel window is "Get Help". The command that would be performed 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 **Sub-menu**, use a "." (a fullstop/period). Thus:

```
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 sub-menu is "Join a Channel". All the commands following it beginning with a "." are part of the submenu that fits under this menu-item.

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

```
Join a Channel
.Fun Stuff
..Type1 Fun
...Fun1:/join #fun1
...Fun2:/join #fun2
..Type2 Fun
...Fun2:/join #fun2
...Fun3:/join #fun3
.Help
..Get IRC help!:/join #irchelp
..Help1:/join #help1
..Help2:/join #help2
```

.Other Channels
..Visit the folks at #friendly:/join #friendly
..Wibble Wobble:/join #wibble
.Who shall we join?:/join #$$?="Please enter a channel name:"

And another example:

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 $?

The above menu would probably be suitable for the main channel window.

NOTE: You can separate menu items by placing a single "-" (minus sign) on a line by itself.

To use the popup menu for the nickname listbox, you need to select a nickname before the menu will pop up. The following example follows the same design as the one above, except that it is oriented towards performing commands on a nickname. Here is a typical listbox popup menu:

Who Is?:/whois $1
-
Modes
.Give Ops:/mode # +o $1
.Deop:/mode # -o $1
.Deop, Kick, Ban:/mode # -o $1 | /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...?

Activation
To activate/deactivate popup menus, turn the Active option on or off for each one.

The popup menu for the **Query/Chat** and the **MenuBar** works more or less the same as the channel listbox popup menu. The $1 always refers to the nickname of the user you have selected.

Extras

The options in this menu are more or less after-thoughts... thus the highly descriptive title. Since there are many options that could still be added, they will probably all end up here until they grow to warrant a separate dialog box all to themselves.

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.

Windows colours
You can have mIRC use your windows colours, however it will only display monochrome text. This is because mIRC cannot know which other colours are compatible with the background window colour you've chosen.

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

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 **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.

The **ontop** switch allows you to position the switchbar at the top or bottom of the main window.

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. Some people use the "." character to prefix commands. Note that regardless of what 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.

Future Work

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

Writing this client would have been **impossible** without the availability of documentation, source code, and help from many people. In an attempt to understand the workings of winsock and windows routines, source code from many sources was mutilated and recombined in weird and wonderful ways. A year and a half and 10,000's of lines of code later, it's difficult to know not only who to thank but also how much. So, in no particular order, here goes nothing:

Thanks to...

Winsock help files, RFC1459, IRC Faqs, IRC source codes, and a multitude of other documents, as well as all of the source code for winsock and windows programming in the Public Domain, I couldn't have done it without you. I must have looked at a million zip files. Here is a small list of source codes which I found useful in helping me clarify ideas, understand protocols, and in some cases to become even more confused! These are the few sources I can remember. I also used several different programs to help me trace/experiment with winsock calls as well as other sources in Pascal, C, C++, and Basic. I'm sorry I can't remember all of you, thanks!

Peter R. Tattam, whose WinIRC was the inspiration for this program, specifically it's lack of an /action command! It's quite possible that had WinIRC had an action command, I would never have written this client.

Nicolas Pioch, for his short IRC primer, portions of which were used to write the IRC commands section.

Ramji, who kept on asking for new options to be added to the program... remind me to stay away from you the next time I code something ;)

Bibbly, whose accute knowledge of RFC's directed me to the one for IRC, saving me from having to download them all ;)

Stimps, who showed no mercy and uploaded me a slightly large file explaining hows scripts work...

Tjerk, whose emails, comments, flying fish, FAQ, and beta-testing have improved mIRC in too many ways to mention.

Mike, for his user-interface advice, bits of code, and artistic appreciation of my toolbar buttons ;)

James G., who answered question... after question... after question...

Edward, whose diligence in beta-testing is matched only by his anagramatic wizardry...

Bunster, for providing me with an unlimited supply of delicious life-giving queen-size futons...

Peeg, for his moral support and frequent *bonk*s over the head ;)

Viv, may a zillion rose petals land gently on thee (and you know who :)

and ofcourse... the **Beta-testers** and the folks who sent me bug reports and suggestions :)

and special thanks go out to the folks who help out on channel **#mIRC** and all the other mIRC-related channels on **EFnet**, **Undernet**, and **DALnet**.

also to all the folks who have set up mirror sites for the mIRC homepage and FAQ!

and to **M.Hunnibell** for the ident server suggestion and help, **M.Overtoom** for the ctl3d dialog code, **Mark "Too Slick" Hanson** for debugging the DCC routines, **Ron R.** for his firewall code, **Emiliano Valente** for his mIRC logo design, **Dan Lawrence** for helping me debug the firewall routines, and...

all the other wonderful, heart-warming, 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. Thanks! :)

IRC II v2.2pre8 and above copyright (c) 1992, 1993, 1994 matthew green.
WSFtp v2.0 Copyright 1994 John A. Junod
Zircon Copyright (c) 1993 Lindsay F. Marshall Jon Harley
Finger 1.0 by Zoran Dukic e-mail: dukic@olimp.irb.hr
SerWeb Copyright 1993 Gustavo Estrella estrella@cass.ma02.bull.com
TinyIRC Nathan Laredo nathan@eas.gatech.edu
Text Server Version 1.0 Copyright 1993 Lee Murach
Winsock Module in Winsock Chess Copyright (C) D. Munro 1994
Finger Version 3.1 Lee Murach Copyright 1992, 1993 Network Research Corporation
wSMTP Server Copyright 1993 Ian C. Blenke
NCSA Telnet J.Mittelhauser (jonm@ncsa.uiuc.edu) C.Wilson (cwilson@ncsa.uiuc.edu)
WinWhois Koichi Nishitani (njknish@mit.edu) Larry kahn (71434,600 kahn@drcoffsite.com)
WSPing John A. Junod (junodj@css583.gordon.army.mil / zj8549@trotter.usma.edu)

Installation

The **mIRC v4.72 installation program** should have installed the **following** files into an mIRC directory for you:

16-Bit Installation:

mIRC.exe	16-bit mIRC
mLink.exe	16-bit mIRCLink

32-Bit Installation:

mIRC32.exe	32-bit mIRC
mLink32.exe	32-bit mIRCLink

Both 16bit and 32bit Installations:

mIRC.hlp	the help file
IRCintro.hlp	the IRC intro file
readme.txt	this file
versions.txt	version history
mirc.ini	the configuration file
aliases.ini	the aliases file
popups.ini	the popups file
remote.ini	the remote file
servers.ini	the irc servers file
urls.ini	the urls file

To use the **32-bit version** you will need to have either Win32s installed on your system or you must be using a 32-bit operating system. Once you have installed mIRC, run it from windows, fill in the required setup information, and connect to the IRC server.

The **mirc.ini** file stores general mIRC information, whereas the other .ini files store the aliases, popups, and remote section information respectively. You should store these sections separately from the mirc.ini file. These .ini files come with useful examples and pre-written aliases and popup menus that will help you understand how to configure these options.

Please read the **versions.txt** file from now on to find out what has been changed/fixed in newer versions.

To use mIRC you will need a **Winsock.dll** of which several versions exist (both commercial and shareware) and a connection to the internet. If mIRC 32bit **doesn't** work for you, it might be your winsock, so you should try using the mIRC 16bit version to see if that solves the problem.

Some Basic IRC Commands

mIRC does not come with a tutorial and assumes that you are already somewhat familiar with IRC. For those with limited IRC experience, the following list of commands should help you get started. Note that you can join channel #irchelp to get help if you are confused about a command. There are also mIRC commands you can look at.

Once you have connected to an IRC server, you can try some of the following commands:

/JOIN #channel
Join the specified channel.

example: /join #irchelp

This will join you to 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.

/PART #channel
Leave a channel.

example: /part #irchelp

/LIST [#string] [-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 #string then mIRC will only list channels with that string in their title.

example: /list
example: /list -min 5 -max 20
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 whatever you like.

/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 Control

The following commands give you control over both a channel and the users on that channel (assuming you have Op status). It's probably wise not to abuse either of them.

/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.

Good luck!

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.

As a server, mIRC's **default service name** is... **mIRC**. mIRC defaults to DDE Server turned 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.

The following DDE topics are supported:

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 is essentially the same as the remote DO ctcp command, it just executes 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 4.72" or "mIRC32 4.72"
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 [-r] <service> <topic> <item> [data]

The command defaults to sending an XTYP_POKE unless you specify the **-r** switch in which case an XTYP_REQUEST 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.

/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 ""

Tools Menu

Some basic tools...

Aliases
Popups
Remote

Finger
Timer
Font

DCC Resume

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

This will only work with another mIRC client.

If a user tries to send you a file that already exists in your get directory then you will be shown a DCC Get dialog warning that the file exists. The dialog gives you 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 negotiation method is specific to mIRC. It is not standard and will not work with other DCC implementations (especially since most do not have resume capability). The negotiation is automatic, and once the receiving user clicks the resume button, the transfer will commence as normal.

Here is a description of the mIRC DCC Resume Protocol.

Connect

Select this menu item to initiate a connection to IRC. You must have input the required information in the Setup window. This menu item changes to "Cancel connect" while attempting to connect to an IRC server, and it changes to "Disconnect" while you are connected.

IRC Servers

You must enter basic information about yourself in this dialog. mIRC will **not** connect if any of these fields are empty.

IRC Servers
You can build up a list of IRC servers that you use regularly by clicking on **Add** to add a new server, **Edit** to edit an existing server, and **Delete** to delete a server.

You can find lists of and links to IRC Servers on the **mIRC Homepage**.

For each server you can type in:

A **Description**, which can be **any** text you want.

An **Address**, which is the actual IRC server address eg. irc.dalnet.com

A **Port number**, which 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.

A **Group name**, which allows you to group servers together when they are sorted with the **Sort** button.

A **Password**, which is **only** required for special types of users, if you don't **know** if you need one then you probably don't!

If you **double-click** on a server then mIRC will try to connect to it.

If you try to connect to a server and you get **"Unable to resolve IRC server"** then the problem could be that:

(a) Your Internet Provider's DNS isn't working,
(b) the irc server with this address is currently not working,
(c) The address doesn't exist or is invalid, or
(d) You're trying to use the 32bit mIRC with a 16bit winsock.

If you try to connect to a server and you get **"Unable to resolve local host"** then see the **Local Info** section.

Remember that if you try to connect to an IRC server that is geographically far away from you, it will probably take longer to connect to it. In fact, some servers might refuse you a connection if you're too far away. So just try a different, closer server until one catches.

Real Name and Email Address
The first entry can be either your real name or a witty one liner which will appear in your /whois information. You must also enter a **full** email address eg. khaled@mardam.demon.co.uk. The username part of the email address is used to register with the server.

Nickname and Alternate
As well as entering a regular nickname, you can also enter an alternate nickname so that if the first nickname is in use when you try to log on, the alternate will be tried. 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. These values cannot be changed while connected to a server.

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.

User ID 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 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 should usually be 113.

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 fingers a persons address to find out more information about them. If a user has their own personal address, this will only work if they run 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, eg. **/finger xhlec@wmin.ac.uk**

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 does a /userhost on the nickname to find the users address and then fingers that.

On a related note, check out the Finger Server.

Timer

The **online timer** is displayed in the status window next to your nickname.

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 Tool** allows you to specify both how mIRC replies to **CTCP messages** from other users and how it reacts to certain kinds of **IRC events**. This tool is the most **complex** part of mIRC and requires that you understand the format of Aliases and IRC commands precisely.

With this tool you will be able to let mIRC respond to almost any remote command you can think of... in fact, any command that you can type can be executed remotely. You can do things like give your friends Op status or invite them to a channel at their request, you can give them access to your files, or you can let them control your mIRC in other ways.

You will also be able to set up mIRC to react to events such as users joining channels, responding to queries with certain words in them, and opping or deopping users automatically.

The format of remote definitions is exactly the same as Aliases except for the prefix.

The **Remote window** consists of:

Users where you keep a list of all authorized users and their access levels,
Variables where a list of all currently defined variables can be edited,

Commands where you keep a list of ctcp commands to which mIRC should react,
Events where you keep a list of events to which mIRC should react,
Raw where you keep a list of raw numeric server messages to which mIRC should react.

The **Listen** switch turns the Commands, Events, and Raw **on** and **off**.
The **Default level** specifies the highest access level commands that **all users** can access.

The Internal Address List switch is explained here.

The following sections **explain** various aspects of the remote tool.

Remote Commands
Remote Identifiers
Access Levels

The following examples describe, with increasing complexity, how the remote tool can be configured.

Example 1: How to change replies to standard CTCP queries...
Example 2: How to Op, Invite, and Unban your friends...
Example 3: How to control mIRC remotely...
Example 4: How to offer files for people to DCC...
Example 5: How to listen for events...
Example 6: How to listen for raw numeric messages...

Other notes

Remote commands

The following commands add and/or remove users as well as manipulate information in the remote section:

/raw [on | off] - switches the raw section on and off.

/remote [on | off] - switches commands, events, and raw on and off.

/commands [on | off] - switches commands on and off.

/events [on | off] - switches events on and off.

/dlevel <level> - changes the default user level eg. /dlevel 2

/groups [-e | d] lists either all, enabled, or disabled groups in your remote definitions.

/ulist [< | >]<level> - lists users in the remote list with the specified access levels. For example:

/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

/enable and **/disable** are used to make groups of commands active or inactive. See Access Levels.

Maintaining Users and their Access Levels

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.

This is a complicated command and it's best to play around with it for a while to understand what it does and why it does it. The way it handles/replaces levels (both the initial general access level as well as = levels) should hopefully make sense.

/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], a number between 0 and 4. If no [type] is given then the default is type zero.

/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** levels of a user 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 and if a user has no more levels left then the user is also removed.

/rlevel -r 1,5,7,8

/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.

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 define which users can access which commands.

The **default access level is 1** for users that are not listed in the Users list. All users can access level 1 commands. The higher a user's access level is, the more commands that user can access. You can change the default user level to allow unlisted users to access more commands (see the /dlevel command). However, Users that **are** listed will still have the same access level regardless of the default level setting.

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

<access level> : <user nickname/address>

For example:

2:*!*jimmy@widgets.com
5:*!*parrot@crackers.inc
19:*!*xhlec@wmin.ac.uk

You can specify the users nick or the users nick!user@address. You can also use wildcards.

In the **Commands** window you specify a list of commands and their access levels in the format:

<access level> : <ctcp command> : <alias commands and parameters>

For example:

1:MYLEVEL:/notice $nick You are weak! You have at least access level 1
5:MYLEVEL:/notice $nick You are a friend! You have at least access level 5
7:MYLEVEL:/notice $nick I trust you completely! You have at least access level 7
50:MYLEVEL:/notice $nick You are the boss! You have at least access level 50!

The command "MYLEVEL" can be executed by anyone who has an access level equal to or greater than it's corresponding access level.

If jimmy does a /ctcp yournick MYLEVEL, the reply will be "You are weak! You have at least access level 1". For parrot the reply will be "You are a friend! You have at least access level 5", and for xhlec the reply is "I trust you completely! You have at least access level 7".

The remote always replies with the **highest level command** that matches the users ctcp query and that is within the users access level.

Limiting Access

Note that the following control prefixes are sensitive to positioning. It is best to set up your definitions starting with the lowest access levels at the top of the list and increasing numerically towards the bottom of the list.

You can **prevent** a user from accessing any levels by prefixing their level with an **equal sign**:

Users:
=150:bannednick!user@not.nice.com

Commands:
150:*:/notice $nick Go away, you are banned from using this remote.

bannednick now has access **only** to command 150 and will always get this reply. Notice that the ctcp command is a * which means that this will match **any** ctcp command that this user sends.

The = prefix can also be used to give a user access to several specific levels:

Users:
10,=20,=22,=26:bannednick!user@not.nice.com

The first level in the list is treated normally and can be anything. The remaining levels are treated as = levels (even if you dont put the = infront of them). This user can access any commands equal to or lower than level 10, but the user also has access specifically to only level 20, 22, and 26 commands.

You can **limit commands to users with a certain access level** by prefixing a command with a **plus sign**, for example:

Users:
3:*!*rocky@crackers.inc
10:*!*nicky@crackers.inc
17:*!*wacky@crackers.inc

Commands:
+10:OPME:/notice $nick Hi nicky! Only users with level 10 can access this command!

You can also **prevent users with higher access levels from accessing lower access level commands** by using an **equal sign** in the following way:

Users:
17:*!*nicky@crackers.inc

Commands:
3:HELPME:/notice $nick Help is unavailable!
4:HELPUS:/notice $nick Here is your help...
10:HELPME:=

This would prevent users with access levels higher than 10 from accessing matching commands with levels lower than or equal to 10. Thus, nicky cannot access the level 3 HELPME but she can access the HELPUS command.

You can **prevent a command from reacting to a command sent by you** by using the exclamation mark in the following way:

Users:
17:*!*nicky@crackers.inc

Commands:
3:HELPME:/notice $nick Help for level 3 is unavailable!
10!:HELPME:/notice $nick Help for level 10 is unavailable!

If nicky has this definition in her remote setup and sends herself a /ctcp nicky HELPME then even though she has access level 17, the level 10 command will be skipped and she will get the reply from the level 3 command.

Events that require you to have Ops

You can limit events to being executed only when you have Ops using the @ prefix:

Events:
@2:ON JOIN:#mychannel:/mode $chan +o $nick

So if someone joins this channel you will only issue the auto-op command if you have Ops on the channel. This only works for **Events**, and **not** CTCP messages since CTCP messages are not associated with channels.

Creating Groups of Commands/Events

You can create separate groups using the # prefix in this way:

#groupA start
1:ON JOIN:...
1:ON TEXT:...
#groupA end

You can use **/enable #groupA** or **/disable #groupA** to enable/disable group #groupA. A group that is disabled will be ignored by the remote when processing remote commands/events. A disabled group looks like this:

#groupA disabled
1:ON JOIN:...
1:ON TEXT:...
#groupA end

You **cannot** have groups within groups. You can also specify multiple groups with /enable and /disable, eg. /enable #group1 #group2 #group3.

How to change replies to standard CTCP queries...

Here are simple examples that show how you can customize your normal ctcp replies for finger, ping, time, etc... You do not need to modify your remote Users list, just copy the following lines to your Commands list, and turn on the Listen option in the remote window:

1:PING:/notice $nick Heehee, that tickles!
1:FINGER:/notice $nick If you want to know something about me... ask! | /halt
1:TIME:/notice $nick My local time is: $time
1:VERSION:/notice $nick I'm using mIRC v4.72!
1:USERINFO:/notice $nick You have reached a non-working number, please try again... | /halt

Description:

1. The number 1 at the beginning of each line specifies the access level required to execute this command. All users have default access level 1 if they are not specified in the users list, so all users can execute level 1 commands.

2. The word following the number 1 eg. PING is the ctcp query that the user has sent you. It is not case-sensitive.

3. The /notice is the alias command. You should already know how to write aliases if you are reading this section. If not, read the aliases help section first!

4. The $nick is the nickname of the user who sent you the ctcp query. Other variables you can use are $address (which is the address of the user), $parms (the parameters that the user sent), and $me (your own nickname).

5. The /halt at the end of a line prevents the normal reply to this ctcp query from being sent.

How to Op, Invite, and Unban your friends...

This simple example shows how users can ask your remote to execute an IRC command for them.

Users:
2:friendA
5:friendB
6:friendC
9:friendD

Commands:
1:WHOAMI:/notice $nick You are $nick your address is $address and you sent parameters $parms
1:OPME:/notice $nick Sorry, you do not have the required access level for this command
2:INVITE:/invite $nick $parm1
5:UNBAN:/mode $parm1 -b $parm2
7:OPME:/mode $parm1 +o $nick

Description:

1. If any of your friends send you a "/ctcp yournick WHOAMI", the remote will reply with their nick, address, and any parameters they sent.

2. All of your friends can send you a "/ctcp yournick INVITE #channel" and the remote will invite them to the channel they specified (assuming you are on that channel).

3. friendB, friendC, and friendD can send you a "/ctcp yournick UNBAN #channel friendX" and if you have Op status on #channel then the remote will unban them.

4. friendD can send you a "/ctcp yournick OPME #channel" and if you have Op status on #channel then the remote will give friendD Op status. All other users will get the level 1 OPME reply.

How to control mIRC remotely...

Users:
10:yournick!*@youraddress

Commands:
10:PART:/part $parm1 | /notice $nick I have left channel $parm1
10:QUIT:/notice $nick Okay boss, I'm quitting... see you later! | /quit
10:SEND:/dcc send $nick $parms

Description:

1. If you send a "/ctcp yournick PART #channel", the remote will part the specified channel and tell you that it has done so.

2. If you send a "/ctcp yournick QUIT", the remote will quit IRC and tell you that it has done so.

3. With the SEND command, you can ask the remote to send you a file. So if you send a "/ctcp yournick SEND info.zip", the remote will dcc send you this file.

How to offer files for people to DCC...

This example shows how the remote can reply differently for levels of users. Level 1 users cannot request files from you while level 2 users can. Only a level 3 user can get the special file. The level 3 user can also get files #1 and #2.

Users:
1:jimmy!*@widgets.com
2:parrot!*@crackers.inc
3:*!xhlec@wmin.ac.uk

Commands:
1:CMDS:/notice $nick Existing commands are : cmds, mylevel, mycmds, xdcc
1:MYCMDS:/notice $nick Commands available: cmds, mylevel, mycmds
1:MYLEVEL:/notice $nick You have access level 1

2:MYCMDS:/notice $nick Commands available: cmds, mylevel, mycmds, xdcc
2:XDCC LIST:/notice $nick Files offered: #1 readme.txt(4K) ; #2 utils.zip(132K)
2:XDCC SEND #1:/dcc send $nick c:\files\readme.txt
2:XDCC SEND #2:/dcc send $nick c:\files\utils.zip
2:XDCC SEND:/notice $nick You must specify file #1 or #2
2:XDCC:/notice $nick Try XDCC LIST
2:MYLEVEL:/notice $nick You have access level 2

3:XDCC LIST:/notice $nick Hello $nick !! Files offered: #1 readme.txt(4K) ; #2 utils.zip(132K); #3 special.zip
3:XDCC SEND #3:/dcc send $nick c:\files\special.zip
3:XDCC SEND:/notice $nick You must specify file #1, #2, or #3
3:MYLEVEL:/notice $nick You have access level 3

Remote Identifiers

In the parameters supplied to an alias command you can use the following identifiers to refer to certain values. Don't worry if you don't understand how these are used right now, they'll be explained in the examples.

$chan refers to the channel where an event took place. For Events that have no associated $chan value, $chan refers to a channel that both you and the user that triggered the event are on, on which you are an Op. For CTCP commands, $chan is only filled if you received the ctcp via a channel.

Note: CTCP and RAW commands are **not** associated with a channel, so this identifier is never filled for them.

$clevel refers to the command level that was matched for the currently triggered event.

$dlevel refers to the current default user level.

$nick and **$address** refer to the nickname and address of the user sending the ctcp query. You can also use **$site** to refer to the portion of $address after the @.

$fulladdress refers to the full address of the user trigerring the event in the format nick!userid@address.

$maddress(nick!userid@address) returns the first matching address in the Users list.

$numeric refers to the current raw numeric being process in a raw event.

$parms refers to the **whole** parameter line that the user sends with the ctcp query.

If you want to refer to particular parameters within the user supplied parameter line, you can use:

$parmN where N is a number eg. $parm1 or $parm2

To refer to all parameters from a certain position:

$parmN* where N is the parameter position. eg. $parm3*

$parmN1-N2 which means use parameters N1 to N2, eg. $parm2-5

You will see later that it is important to specify a parameter number in certain situations. For example, if you define a command that requires only 1 parameter and the user sends 2 parameters then this could affect the way the command works.

$ulevel refers to the user level that was matched for the currently triggered event.

$wildsite returns the address of the user who triggered an event in the form *!*@host.domain.

Other notes...

It is wise to test out every command you define to make sure it does what you want and **nothing else**. For example:

5:OPME:/mode $parms +o $nick

With this definition, if a user supplies more than one parameter then the /mode won't work or it might do something unpredictable. You should redefine the command as:

5:OPME:/mode $parm1 +o $nick

This makes sure that only the first parameter that the user supplies is used in the final command executed by the remote.

Variables

You can specify Variables instead of a fixed CTCP command. For example:

2:%password:/notice $nick You're access has been authorized!

This allows you to change the value of the %password variable which saves you from having to rewrite all definitions that require a password. So if you set %password to the value "moo" and someone sends you a "moo" ctcp, it will match %variable and the notice will then be performed.

You can also do this with events in two places:

1:ON TEXT:%text:%channel:/notice $nick you just said $parms on channel $chan

The %text will be matched against whatever text the user sent you, and the %channel will consist of variable channel names which will be matched against the channel on which you received the message.

Multiple Channels

You can specify multiple channel names in events:

1:ON JOIN:#irchelp,#ircbar,#mIRC:/notice $nick Welcome to $chan!

Pass through commands

You can allow only certain commands to be executed by doing this:

5:PING
5:VERSION
5:*:!

The remote will now only allow replies to PING and VERSION. All other **level 5** commands will be ignored. The exclamation mark ! at the end of the line tells the remote to halt any further processing of level 5 commands.

Commenting out commands and events

You can comment out definitions by prefixing them with a ; (semi-colon) or REM.

```
5:PING
rem 5:PING
;5:*:PING
```

Ordering of definitions
Remember that 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.

DCC Send

DCC Send allows you to send a file to a user by specifiying their nickname and the name of the file you want to send. Once you have entered the nickname and the filename, select "Send" and mIRC will send information to the nickname you specified telling them that you want to send a file. The user then has to accept your send request, at which point the file transfer should begin.

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. Each DCC Send session can have a different packet size... just change it before you start the DCC Send. mIRC can receive packets up to 4096 bytes.

If you check **minimize** then the dcc send window(s) will be minimized automatically.

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 a better explanation, read this.

NOTE: There is an experimental DCC Fast Send option which you can try out.

DCC Chat

DCC Chat allows you to talk privately by connecting directly to another client, bypassing the IRC network. Enter the nickname of the user and select "OK" and mIRC will send information to the nickname you specified 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 replies to a /dcc chat request by initiating another dcc chat with you then mIRC treats this as an acceptance of it's own request.

If a user sends you a chat request, a dialog will popup asking you whether you want to talk to them. You can then accept or decline.

DCC Get

The DCC Get dialog only appears in response to a send request from another user.

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

DCC Options

Show Get Dialog/Auto-get file/Ignore all
By default, a send request must be accepted by you before transfer begins. However, if you select the auto-get file option then mIRC will automatically accept a send request and begin receiving a file. It will also **minimize** the receive window if you select the minimize option. You can also have mIRC close the get window automatically after a tranfser is complete by selecting the Auto-close get window option. If you select Ignore All then all incoming dcc send requests are ignored.

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

On Session 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.

Max. DCC Sends
This limits the number of simultaneous **remote** DCC Sends. ie. it does not apply to you manually initiating a DCC Send but it does apply to users who request a DCC Send remotely.

Max. Fileservers
This limits the number of users that can be served simultaneously.

Auto-resume existing files
If someone tries to send you a file that already exists in your dcc get directory then mIRC usually pops up a dialog asking you if you want to overwrite/resume/rename the file. Checking this option forces mIRC to always resume a file automatically if it already exists.

Display fileserver warning
If this switch is checked then mIRC will pop up the fileserver warning dialog everytime a fileserver is initiated to someone. To learn more about fileservers see the Fileserver section.

DCC Get Directories
This allows you to specify dcc get directories where recieved files will be placed according to their extension. The default directory is where all files not matching any of your

extensions are placed.

DCC Resume Protocol

In it's current form the 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 mIRC still sends a filename as "file.ext" in both RESUME and ACCEPT.

ON CHAT/SERV

To listen for messages in DCC Chat windows:

1:ON CHAT:boo:/msg =$nick boo back!

If someone says "boo!" in a dcc chat window, the remote replies with "boo back!". **Note** that to send this as a dcc chat message to this user you have to use the standard "=" prefix with the nick. ie. =$nick

1:ON CHATOPEN:/msg =$nick Hi! I'll be with you in a second...

The moment a chat connection is established, this event is triggered allowing you to send the user information etc.

1:ON CHATCLOSE:/notice $me $nick just left the discussion...

Which is triggered whenever a chat connection is closed.

To listen for messages in Fileserver windows:

1:ON SERV:*:/msg =$nick I say $nick, what do you mean by " $parms " ?

This listens for any message from the user and sends the message back. Notice that I can use both **=$nick** for a dcc chat message and also **$nick** for normal messages. In a fileserver session, you can use **$cd** in a remote definition which represents users current directory.

1:ON SERVOPEN:/msg =$nick Fileserver is currently down, please come back later!

The moment a fileserver connection is established, this event is triggered allowing you to send the user information etc.

1:ON SERVCLOSE:/echo Fserve to $nick closed $fulldate

The moment a fileserver connection is closed, this event is triggered allowing you to update information etc.

See the ON TEXT event for an explanation of the uses of the "text" section.

NOTE: The above commands react to all users level 1 and above because of the way dcc chat works.

How to listen for events...

The events window allows you to specify how mIRC reacts to certain types of events. The set of events currently available is limited to the most basic types, each of which will be given simple examples to illustrate how they work.

The event handler listens to events caused by users specified in the Users List (which it shares with the Commands list). Each event has an access level which is matched to a users access level. The default level works the same as in the commands list. You can also define normal command defintions in this window.

The examples cover each type of event:

The ON BAN/UNBAN events are triggered when a user sets a ban in a channel.

The ON CHAT/SERV events listen to messages from users in dcc chat and fileserver windows.

The ON FILESENT/FILERCVD events are triggered when a DCC transfer is successful.

The ON JOIN/PART events are triggered when users join or part a channel.

The ON KICK event is triggered when a user is kicked from a channel.

The ON OP/DEOP events are triggered when a user is opped or deopped.

The ON TEXT event listens to messages from users either on certain channels or in query windows.

The ON VOICE/DEVOICE events are triggered when a user is voiced or devoiced.

And a few Other Events which react to various situations.

ON TEXT

Keys:
? = Any private message
= Any channel
#mirc = Channel #mirc
* = Both private and channel messages

The match text can contain the * wildcard, for example:
* = any text
text = if user said only this word
text* = if user started line with this word
*text = if user ended line with this word
text = if user said this word anywhere

To listen for messages on channels...

1:ON TEXT:hello*:#:/msg $nick Hello back!

This command listens for the message "hello" from any user on any channel and replies with the message "Hello Back!".

1:ON TEXT:hello*:#mIRC:/msg $chan Welcome to #mIRC!

This command listens for the message "hello" from any user on channel #MIRC and replies with the message "Welcome to #mIRC!".

5:ON TEXT:goodbye:#mIRC:/msg $chan Goodbye $nick!

This command listens for the message "goodbye" from any user with access level 5 on channel #MIRC and replies with the message "Goodbye NickName!"

To listen for private messages...

1:ON TEXT:help:?:/msg $nick Please use /ctcp $me help for more information...

This command listens for a private message with the word "help" in it from any user and replies with the message " Please use /ctcp mynickname help for more information..."

1:ON TEXT:*:?:/msg $nick I'm not here! Go away!

This command listens for any private message from any user and replies with the message "I'm not here! Go away!"

To listen for any messages...

1:ON TEXT:*:*:/msg $nick Not Listening...

This command listens for any message both private or public from any user and replies with the message "Not Listening..."

The **ON NOTICE** and **ON ACTION** events work exactly the same way as the ON TEXT event.

The **ON SNOTICE** event listens for server notices and can prevent them from being displayed. The format is slightly different:

1:ON SNOTICE:*Client connecting*:!
1:ON SNOTICE:*Client exiting*:!

This would prevent "client connecting" and "client exiting" server notices from being displayed.

The **ON WALLOPS** events listens for wallops messages and uses the format:

1:ON WALLOPS:*warning*:/echo $nick said $parms

NOTE: You cannot test out these events by typing text to yourself! They are initiated by someone else saying something in a channel or in a private message.

ON JOIN/PART

You can have mIRC perform certain actions whenever a user joins a channel...

2:ON JOIN:#:/msg $chan Welcome $nick!

When a user with access level 2 joins any channel, you say "Welcome nickname" on the channel.

10:ON JOIN:#mIRC,#mIRCScripts,#mIRCRemote:/kick $chan $nick | /msg $nick And OUT of $chan $nick goes! :)

When a user with access level 10 joins channel #mIRC, #mIRCScripts, **or** #mIRCRemote, you immediately kick them out and send the above message to them.

14:ON PART:#:/msg $chan Yahoo! $nick is gone! ;)

When a user with access level 14 parts any channel you tell everyone how happy you are :)

ON KICK

2:ON KICK:#:/kick $chan $nick | /invite $knick $chan | /msg $nick That person is my friend!

When a certain user is kicked out of any channel you kick the person who kicked them, send them a message, and invite the person who was kicked back to the channel.

2:ON KICK:#mIRC,#irchelp:/notice $me Hey! Wake up! $nick was just kicked with the message " $parms "

When a certain user is kicked from channel #mIRC or #irchelp you send yourself a notice telling you what just happened.

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

>=2:ON KICK:#mIRC:/msg $chan $nick kicked $knick (legal)
1:ON 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 line 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 that **this event works on a nickname and not an address.** This is because the IRC server only sends the address of the user doing the kicking and not of the user being kicked.

ON OP/DEOP

These two events are concerned with operator status. The event is performed for each user that is opped/deopped:

100:ON OP:#:/mode $chan -o $opnick | /msg $nick Don't Op that person!

When a user with access level 100 is opped, you immediately deop them and send a mesage to the person that opped them.

9:ON DEOP:#mIRC:/mode $chan +o $opnick | /msg $nick Hey! That person is my friend!

When a user with access level 9 is deopped, you immediately op them and send a mesage to the person that deopped them.

1:ON SERVEROP:#:/mode $chan -o $opnick | /notice $me $opnick was opped by $nick!

When a user is opped by a server I deop the user and send myself a notice. The $nick refers to the server that did the opping.

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

>=2:ON DEOP:#mIRC:/msg $chan $nick deopped $opnick (legal)
1:ON 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 ON DEOP 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 that **these events work on nicknames and not addresses**. This is because the IRC server only sends the address of the user doing the opping/deopping and not of the users affected ie. it only sends the **nicknames** of the users being opped/deopped.

Other Events

1:ON CTCPREPLY:PING:/echo $nick replied to my ping!

This allow you to intercept ctcp replies.

2:ON INVITE:#mIRC:/join $chan | /describe $chan appears in a puff of smoke!

When a user invites you to channel #mIRC, you automatically join the channel and send the above action to it.

1:ON MODE:#mirc:/notice $nick changed mode to $parms

When a mode change occurrs on a specified channel. $parms holds the actual mode change parameters. ON SERVERMODE works the same way but reacts to server mode changes.

2:ON NICK:/describe $newnick thinks $nick was a nicer nickname!

When a user changes their nickname, you tell them you thought their last nickname was nicer. If you are the one who just changed your nickname, the $me identifier is only updated after this event is processed.

3:ON NOTIFY:/msg $me Hey, wake up! $nick just joined IRC!

When a user joins IRC and you have their nick in your notify list and the **notify is turned on** then this will react to the notification.

2:ON QUIT:/notice $me Hey! Wake up! $nick just quit with the message " $parms "

When a user quits IRC you alert yourself with a notice.

2:ON TOPIC:#:/describe $chan admires $nick's new topic!

When a user changes the channel topic.

1:ON NOSOUND:/msg $nick I don't have $filename

Triggered when a user sends you a /sound request and you don't have the sound they requested.

1:ON DNS:/echo $nick ip address: $iaddress named address: $naddress resolved address: $raddress

Triggered when a /dns lookup is complete. If /dns <nick> was used then $nick refers to <nick> otherwise it refers to your own nickname. $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.

1:ON ERROR:*text*:/echo $parms event!

Triggered when a server sends an ERROR line, this usually occurrs on a disconnection. It is

not related to any kind of error in mIRC, it is a server event.

DCC Fast Send

DCC Fast Send is a simple and experimental extension to a normal dcc send. The aim was to see if it was possible to write a dcc send algorithm that would a) increase transfer speed, and b) remain compatible with the standard dcc get routine.

Basically, dcc fast send attempts to maintain a steady flow of packets by sending slightly ahead of acknowledgements. It does not require the receiving client to implement a special DCC Get algorithm. ie. it will/should work with any client.

By varying the packetsize (using /dcc packetsize <size>) and turning on the fast dcc send option, it is possible to achieve a fair increase in transfer speed. The packetsize is crucial to finding a good balance for maintaining a steady packet flow. 4096 bytes is **not** necessarily the best choice. It is probably better to use something like 2048 bytes.

However, how well it works really depends on both the senders and receivers internet setup and connection speed.

You can turn DCC Fast Send on and off using: /fsend [on|off]

Note:
1.This option has been included just to see if it really does work for most people. It's possible that it might not increase transfer speed for you and in fact, it might even make things slower.
2.If you have problems DCC Fast sending files, then just use the normal DCC send.

mIRC Commands

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

/alias <name> <commands>
This adds or removes aliases to the aliases list, however 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

/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]
This adds someone to the auto-op list, see the Control section for a full description.

/ban [-uN] [#channel] <nickname> [type]
This 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 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.

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>
This beeps a number of times with a delay.

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

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

/close [-icfgms] [nick1] ... [nickN]
This 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 = chat
f = fserve
g = get
m = message (query)
s = send
i = inactive dcc windows

For example, if you want to close all **chat** and **fserve** windows for user **Nerp**, you would type:

/close -cf Nerp

/closemsg [messagewindow]
This closes the specified message window.

/creq [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).

/dcc send <nickname> <filename>
This 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.

/dde [-r] <service> <topic> <item> [data]
This allows you to send information to other applications. Please see DDE Control for a full explanation.

/disconnect
This forces a hard and immediate 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 <nick|address>
This is used to resolve addresses. 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.

/echo [colornumber] <-sa|#channel|nick> <text>
This this 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.

To print in the status window use the **-s** switch, to print in the currently active window use the **-a** switch.

Note: This text is only printed in your window, it isn't sent to the server.

/exit
This forces mIRC to closedown and exit.

/finger <nick/address>
This does a finger on a user. 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.

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

/flushini <filename>
This flushes the specified INI file to the hard disk. This is necessary because of the way INI files are cached in memory, so you might need to do this to make sure that your INI file doesnt lose information if you're updating the INI file frequently.

/font
This pops up the font dialog for the active window.

/fsend [on | off]
This allows you to turn dcc fast send on or off.

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

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

/halt
This halts a currently executing script and prevents any further processing from continuing. 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
This brings up the Basic IRC Commands section in the mIRC help file. You can also do /help <keyword> and if the keyword matches a help topic you will be taken to it.

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

/links
This retrieves the servers to which your current server is linked and lists them in a Links List window for easy viewing and access.

/load <-apuvcer> <file.ini>
This is the command line equivalent of the **Load** button in the alias, popups, and remote dialogs and is used to load one of the following sections where a = aliases, p = popups, u =

users, v = variables, c = commands, e = events, r= raw. For example, if you wanted to load aliases from a file you would do:

/load -a aliases2.ini

NOTE: You can only refer to one section at a time, so you can't do /load -uce some.ini.

/log [on | off] [windowname]
This allows you switch logging on and off for a window.

/nick <nickname> [alternate]
This allows you to change your current nicknames.

/notify <nickname> <note>
This allows you to 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]
This parts all of the channels you are currently on. On certain IRC Servers, you can also specify a message.

/perform [on|off]
This allows you to turn the "Perform these commands" section of the Perform dialog on and off.

/play [-cp q# m# rl# t#] [channel/nick] <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 **-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 **must** be a single number specifying the number of lines in 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.

/pop <delay> [#channel] <nickname>
This 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]
This 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]
This allows you to delete 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. These include the mirc.ini, aliases.ini, popups.ini, and remote.ini files. If you do this, you might corrupt your INI files.

/remove <filename>
This deletes the specified file.

/return [value]
This 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.

/run <filename> [parameters]
This allows you to run 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

/save <-apuvcer> <file.ini>
This is the command line equivalent of the **SaveTo** button in the alias, popups, and remote dialogs and is used to save the specified section, where a = aliases, p = popups, u = users, v = variables, c = commands, e = events, and r = raw. For example, if you wanted

to save aliases to a file you would do:

/save -a aliases.ini

NOTE: You can only refer to one section at a time, so you can't do /save -uce some.ini.

/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 **cannot** use this command in the remote section. Use /msg #channel <message> instead.

/server <server/groupname> [port] [password]
This 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
These allow you to define and undefine your own Variables.

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

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

/splay <file.wav|file.mid>
This 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 .mid file from playing.

/sreq [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).

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

/strip +bur-c

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

/timer[N] [-o] [time] <repetitions> <interval in seconds> <commands>
This 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.

/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.

You can force identifiers to be reevaluated 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]
This 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 <text>
This allows you to set the main application titlebar.

/url [-d] [on | off | hide]
This 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]
This pops up a user window showing information about the specified user. It is the same information you would get if you did a "/whois nickname". You can also build up a list of nicknames and their associated real name and machine address. The format is: /uwho nickname

/write [-cid l# s#] <filename> [text]
This allows you to write lines to a file.

/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.

/writeini <inifile> <section> [item] [value]
This allows you to write to files in a standard INI file format which should make storing and retrieving related bits of information a lot easier than with /write.

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. These include the mirc.ini, aliases.ini, popups.ini, and remote.ini files. If you do this, you might corrupt your INI files.

Other features...

These sections **must** be read as they describe various aspects of mIRC that aren't standard.

mIRC Commands

The System Menu
Text Copy and Paste

Bold, Underline, Reverse, Colour Text

Dynamic Help Menu

The File Server

Special Key Combinations
Command Line Parameters

DDE Control

Miscellaneous Stuff

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.

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> <welcome text file>

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

The commands availabe to a user once connected are:

cd - change 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.

The above commands have been greatly limited in the hope that this will prevent a security breach. If the security aspects of the fileserver trouble you, it is recommended that you use your fileserver with a SUBST'd drive. Since the CD command isn't capable of drive changes, this should limit the user to only that portion of your hard disk.

NOTE:
1.It will improve performance a lot if you make sure that your directories are not too large.

If a directory has a large number of files try to split them up into subdirectories.
2.If a user is idle 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):

Window: You can tell mIRC to **remember or forget the position and size** of the status and channel windows. If they are remembered, the next time a window opens up, it will do so in the saved position. If you choose the **reset** menu item, the window will be moved back to it's previously saved position. Note that if a windows 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. The filename is automatically taken from the name of the window.

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 you select this, the text to the window will be logged to a file. This setting stays on across sessions until you switch it off. This function is only available for the status, channel, and query windows. The filename is automatically taken from the name of the window.

Beeping: This appears only in Channel, DCC Chat, and Query windows. If selected then 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.

Desktop: This allows to position a window outside of the mIRC main window on the desktop.

Stay On Top: This appears only when windows are opened as Desktop instead of MDI windows.

Timestamp: This allows you to turn timestamping of events on/off for a window.

Text Copy and Paste

To copy text from a window, 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.

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 :-)

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.

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.

Channel Folder

This folder allows you to store your **favourite** channels for quick access.

The channels are listed in **alphabetical** order and you can store as many as you want.

To join a channel, you can type in its name and press enter or click the Join button.

You can also enter a channel key right after the channel name, if one is required.

You can use Alt-J to pop up this dialog.

List Channels

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 currently works with the three major WWW Browsers.

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.

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 all the URL functions.

Sound Requests

Accept sound requests
If this option is turned on, mIRC will listen for SOUND requests from other users. You can send a sound request to another user using the command:

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

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

The sound file must end in **.wav** or **.mid**, and may be located in the default sounds directory (see below) 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 somewhere.

The accompanying message is not mandatory, however if you provide one it appears exactly like an action command. The user receiving the sound request must already have the specified .wav or .mid file and must be listening for sound requests.

You can also use **/sound [on|off]** to turn accepting sound requests on and off.

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.

Location of wave files
Whenever a sound is requested, mIRC will look in this directory and all of its **subdirectories** for it. This directory is also searched when you use the /splay 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, especially if you are on several channels 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.

New Format for Remote Commands and Events definitions

The **new format** has been introduced in order to simplify writing definitions for the remote.

The new format is essentially identical to the old format except that **now there is no separation between the aliases section and the parameters section** in a definition, thereby simplifying definitions greatly.

For example, in the old format:

1:PING:/notice *1:$nick -POING!-

the new simplified format:

1:PING:/notice $nick -POING!-

mIRC will automatically convert your old definitions to the new format.

NOTE: mIRC now also stores the commands and events sections in the mirc.ini file with a different prefix. If the command or event sections in the mirc.ini file use the new prefix then mIRC will assume that they have already been converted and will not try to convert them again.

Special 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.

Control-Space - cycles through channel windows.

Tab - inserts a "/msg nickname" into the editbox of the window you are currently on, where "nickname" is the nick of the last person that messaged you.

Cursor Up/Down - browses the command line history buffer for a single-line editbox. For a multi-line editbox you must use **Control-Cursor Up/Down** since Cursor Up/Down are used for editing.

Page Up/Down - browses the scrollback buffer of the active window. You can use **Control-Page Up/Down** to browse the buffer a line at a time.

ESCape key - this quickly minimizes the active window, it must be turned on in the Extras dialog.

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

Control key - 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 color in text.

Command Line Parameters

At this time there are only a few parameters you can specify 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.

The Official mIRC Homepage, FAQ, and other information

For the latest version of mIRC, the FAQ, information on bugs, solutions to problems, general IRC help, etc. please see the **Official** mIRC Homepage located at:

http://www.mirc.co.uk

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

The above page has links to Official mIRC homepage mirror sites:

- http://www2.axi.net/mirc
- http://www.moc.kw/mirc
- http://www-2.nijenrode.nl/software/mirc
- http://www.geocities.com/SiliconValley/Park/6000

Other Information
Below you'll find a selection of additional mIRC and IRC related information available on the Web. All of these pages are maintained by IRC enthusiasts. Keep in mind that due to the nature of the World Wide Web some of these addresses might become outdated at some time.

mIRC pages:

mIRC Tricks/Suggestions	http://www.phoenix.net/~lsimon/mirc.html
mIRC Tips	http://leonardo.spidernet.net/Copernicus/831/mirc/home.html
mIRC Tutorial	http://kpt1.tricon.net/Personal/ewheeler/mirctut1.html
One Stop mIRC Shop	http://www.ctfire.com/
mIRC Scripts	http://www.indy.net/~trekkie/

IRC Intros:

IRC Intro	http://atpibm6000.tuwien.ac.at/~acvitkov/IRCIntro.html
New IRC Users page	http://www.neosoft.com/~biscuits/niu.html
Internet Chat	http://www.azstarnet.com/~emdee/chat/

IRC FAQs:

IRC FAQ	http://www.kei.com/irc.html
IRC Info	http://www2.undernet.org:8080/~cs93jtl/IRC.html
IRC Hints	http://irchelp.org/irchelp/opguide.txt

IRC Networks:

Dalnet	http://www.dal.net/
Undernet	http://www.undernet.org/
IRC for Kids	http://www.kidlink.org/IRC/
Lots of Nets	http://www2.undernet.org:8080/~cs93jtl/ircnets.html

IRC Servers:

Dalnet IRC Servers	http://www.xmission.com/~dragon/dalnet/server.html
Dalnet IRC Servers	http://www.bazza.com/sj/irc/servers.html
EFNet IRC Servers	http://users.hol.gr/~shadow/slists/efnet-servers.html
IRCNet Servers	http://www.ludd.luth.se/irc/servers.html
Undernet IRC Servers	http://http2.brunel.ac.uk:8080/~cs93jtl/servers.html
More Networks	http://users.hol.gr/~shadow/server-lists.html

IRC Channels:

IRC Channels	http://www.funet.fi/~irc/channels.html
Yahoo's list	http://www.yahoo.com/Computers_and_Internet/Internet/Chatting/IRC/Channels/
Popular Channels	http://light.lightlink.com/irc/channels.html
Dalnet Channels	http://www.dal.net/howto/channels.suggest.html

Technical IRC docs:

IRC RFC	http://ds.internic.net/rfc/rfc1459.txt
CTCP Documentation	http://www2.undernet.org:8080/~cs93jtl/irc_ctcp.txt
DCC Documentation	http://www2.undernet.org:8080/~cs93jtl/irc_dcc.txt
DCC Resume Protocol	http://www.mirc.co.uk/dccresum.txt
Color Coding	http://www.mirc.co.uk/color.txt
Running a server	http://www.microserve.com/~jjoseph/wircsrv.html
Undernet Development	http://www.wildstar.net/ircd-dev/
EFnet Development	http://www.funet.fi/~irc/server/
IRC3	http://www.the-project.org/

IRC Switches

Actions are purple
If this is checked, any actions in a channel window appear in purple. Otherwise, all actions appear in black like normal channel text. I've left this as an option because there are quite a few colours being used in a channel window already and adding purple might overload the senses a little bit for some people!

Highlight own messages
If this option is selected, anything you type into a window will be highlighted.

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 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
This 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.

Strip Codes
This allows you to strip out the bold/underline/reverse/color codes from incoming private messages or channel messages.

Perform

Highlight text with these words
On a channel, any lines which contain words matching any words in your list will be highlighted in a dark brown. This eases locating messages directed at you or about you eg. you could place your nickname (or variations of it) in this box. The line you enter must consist of words separated by commas.

Beep on highlighted word
This works in conjunction with the highlight option. If a matching word appears and this option is checked then mIRC will beep.

On connect, perform these commands
You can enter a set of commands in this box which will be performed every time you connect to IRC. To understand how commands work, see the Aliases help section. A quick example:

/names #mirc,#irchelp | /join #winsock

Ctcp finger reply
The message a user receives when they /ctcp finger you.

Quit message
The message displayed 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 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
You can have mIRC automatically log the channel and query/dcc chat windows whenever they open up.

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

If you enable this option mIRC will listen for **finger requests** on port 79.

If the "show finger requests" option is turned on, finger requests are displayed in the status window.

You must specify a **Finger 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.

On a related note, check out the /finger command.

Local Info

If you've been receiving the message **"Unable to resolve Local host"** or you have been unable to **initiate DCC sessions**, then fiddling with the settings below might help. Also note that if you receive the **"Unable to resolve Local host"** message with the **32bit version of mIRC**, the problem might be related to your 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.

Basically:

1) If mIRC is having trouble getting your Local Hostname, then turn OFF "Always get local host", and type in your Local Host manually. The Local host you type in will be used to get your IP address. If your IP address is dynamic, keep "Always get IP address" turned ON, otherwise turn it off. If your Local hostname is dynamic, you're in trouble.

2) If mIRC is having trouble getting your IP address, then turn OFF "Always get IP address", and type in your IP Address manually. The IP address you type in will be used to initiate DCC Sends/Chats. If your IP Address is dynamic, you're in trouble.

3) If the above fail, turn OFF both Always get Local host and IP address, and type both these values in manually. If they're the correct values, great. If they're not, you will be able to use IRC just fine, but you won't be able to initiate DCC Sends/Chats.

Confused? So am I! Please don't email me about this... I've said everything I know in the above paragraphs! If you still have trouble, please contact your Internet Provider or System Administrator for help.

IP 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 but you can tell when it has been completed when you see your DCC Send/Chat toolbar buttons enabled, and also when your DCC Send/Chat menu items in the DCC menu available.

Servers

A server listens for and replies to specific types of requests from other users or systems.

DDE Server
Finger Server

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.

Exit

Saves settings and window positions, closes dcc sessions, quits from the server, and exits mIRC.

Mooo! ;)

Firewall

mIRC can access an irc server through a **SOCKS** firewall The main purpose of this option is most likely 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.

Hostname is the machine name of your SOCKS server and can be either a named address or an IP address.

User ID 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).

Password is the password required to access the firewall.

Port should usually be 1080.

NOTE: Only connecting to an IRC server is supported. DCC Send/Chat, etc. are not supported.

Control

Auto-Op
If you are on a channel and you have channel Op status, any users that match the given nicknames or addresses will be automatically given Op status when they join the channel.

From the command line you can use: **/auto [-r] <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.

You can turn auto on and off by typing **/auto on** and **/auto off** respectively.

The format of the auto-op line is:

nick!userid@host.domain,#channel1,#channel2

On IRC, user addresses are specified in the format:

nickname!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**

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
Messages from the nicknames or addresses in the ignore list will be ignored. ie. you will not see them.

From the command line you can use: **/ignore [-rpcntiku#] <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.

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.

You can turn ignore on and off by typing **/ignore on** and **/ignore off** respectively.

The format of the ignore line is:

nick!userid@host.domain,private,invite,ctcp

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.

From the command line you can use: **/protect [-r] <nickname> [#channel1,#channel2,...]**

The **-k** switch ignores colour/bold/underline/reverse control codes in a message.

The **-r** switch indicates that the address is to be removed.

You can turn auto on and off by typing **/protect on** and **/protect off** respectively.

The format of the protect line is:

nickname,#channel1,#channel2

Note: This option is limited to using **nicknames** because of the way IRC servers work.

Notify List

The notify feature allows you to enter a list of nicknames which mIRC will look for when you are connect to an IRC server.

mIRC will inform you whenever a nickname has joined or left the IRC network. You will also be informed if no users in your list are on IRC when you first connect. mIRC checks the server every 40 or so seconds thereafter and updates its notify list window.

You can also add/remove nicknames from the command line using the /notify command:

/notify [-shr] [on | off | <nickname>] [text]

You can turn notify on and off by typing "/notify on" and "/notify off" respectively.

The [-sh] switches can be used to show or hide the notify list window respectively

The [text] 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 force mIRC to check the server by typing /notify by itself with no parameters.

mIRC also allows you to specify a wave file for each nickname. This wave file will be played whenever this nickname joins IRC.

Remember to click on the **Update** button whenever you modify an entry in the notify list.

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.

Dynamic Help Menu

See Help Menu for a description of the dynamic help menu.

Variables

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

All **currently** defined variables are listed in the **variables** section of the Remote dialog 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 <%var> [value]

This sets the value of **%var** to **value**.

/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 characters %test.

/unsetall

This unsets and removes all variables from the variables list.

/inc <%var> [value]

This increases the value of **%var** by **value**.

/dec <%var> [value]

This decreases the value of **%var** by **value**.

You can also use the **=** equal sign to assign values to variables:

%i = 5

%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

You can only perform a single operation in an assignment at this time.

User Central

This dialog pops up when the command **/uwho [nickname]** is used. This is essentially the same information that the **/whois** command gives you about a user. It allows you to build up a list of nicknames and their associated real name and machine address.

Thanks!

Thanks for your registration! :)

ON BAN/UNBAN

1:ON BAN:#panic:/mode $chan -o $nick | /mode $chan -b $banmask

When someone sets a ban, this will de-op them and remove the ban.

You can also use **$bnick** to refer to the users nickname, however note that if the banmask does not include the users nickname then $bnick will not have a value.

Remember that the **$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?

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

>=2:ON BAN:#mIRC:/msg $chan $nick banned $banmask (legal)
1:ON 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.

Th event **ON UNBAN** works in exactly the same way except it triggers when a user is unbanned.

Note: You will often find that two wildcard strings which don't look like they should match actually **do** match because of the wildcards in them. mIRC will not match ip address against normal addresses, and will not match addresses ending in @* since this can match almost any 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 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 information on how to register is always available from the mIRC homepage.

The current registration amount is **US$15.00** (or **UK£10.00**) when registering via snail-mail, and **US$20.00** when registering via credit card.

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
68 Melbury Court,
London W8 6NJ,
Great Britain

Please make sure you send your **full name**, **postal address**, and **email address** with your registration.

If you are going to send anything **other** than a **UK** or **US** check or money/postal order, please email me first to make sure I can accept it. Note that if the postal order is not from the US or the UK, it should be made out in UK Pounds.

2. Registering **online** with your **Visa/Mastercard Credit Card** through CNET at:

http://www.buydirect.com

The registration amount is **US$20.00** when registering with a credit card.

Once I have **received** your registration, either via snail-mail or via CNET, I will send you an **email** with your registration password. 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.

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** windows or **Channel** windows, and different actions will be taken depending on what aliases you associated with that file type.

You can also define different actions for different types of files by **associating** an Alias 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.

How to listen for raw server messages...

The raw window allows you to create definitions that react to **numeric** server messages.

Filtering and **handling** raw messages can be very **time-consuming** because of the large number of messages that a server can send, so any scripts you write in this section should be as **tight** and as **fast** as possible.

Since there are a **large** number of numeric messages, far too many to go into here, it is recommended that you check out the links on the **mIRC homepage** for **technical** information on server messages. The document you are looking for is called RFC1459.

As a **quick** example, the following script filters out the **channels list** numeric returned when you use the /list command. The **/list** result is probably the most intensive test you can find for the raw numeric routines.

```
322:*mirc* {
  set %n 0
  :next
  set %t $token(%n,32,$parms)
  if %t == $null { goto end }
  inc %n
  goto next
  :end
  echo 3 %n tokens in " $+ $parms $+ "
}
```

The above script matches raw numeric 322 (the channels /list numeric) and if the line returned by this numeric has the word **mirc** in it, the script then **counts** the number of tokens one by one, and then prints out the **total** at the end.

Note: You can prevent most raw server messages from printing out the default text by using the /halt command.

ON FILESENT/FILERCVD

The following events react only to **successful** DCC Sends and Gets.

1:ON FILESENT:*.txt,*.ini:/echo Sent $filename to $nick ($+ $address $+)

When any file match *.txt or *.ini has been successfully sent, it prints a message in the status window indicating the success of the transfer.

1:ON FILERCVD:*.txt,*.ini:/echo Received $filename from $nick | /run notepad.exe $filename

Whenever a file ending in *.txt or *.ini has been successfully received, it prints a messages indicating the success of the transfer, and then runs notepad to read the file.

If-then-else statements

The **If-then-else** statement allows you to **compare** values and **execute** different parts of a script.

The basic format is:

if v1 operator v2 { commands }
elseif v1 operator v2 { commands }
else { commands }

You can **nest** as many if-then-else statements as you want inside each other, just make sure to use **() and {}** brackets properly to clarify exactly the type of comparisons you want. If the alias uses **too few** () and {} 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. Using brackets properly **speeds** up processing.

Available comparisons:

==	equal to
!=	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
isin	string v1 is in string v2
iswm	wildcard string v1 matches string v2
ison	nickname v1 is on channel v2
isop	nickname v1 is an op on channel v2
isnum	number v1 is a number in the range v2 which is in the form n1-n2 (v2 optional)
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** the above you can prefix them with an **!** (exclamation mark).

You can use **&&** and **||** for **AND** and **OR** respectively.

Here are a few examples in the form of alias, popup, and remote definitions...

```
number {
  if (($1 >= 0) && ($1 <= 10)) {
    if ($1 < 5) echo Less than five!
```

```
    else {
      echo Greater than five!
    }
    else if ((%x < -10) || (%x > 20)) echo Out of bounds!
  }
}
```

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

```
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.

```
.Give Ops {
  %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.

```
1:ON CTCPREPLY:PING* {
  if ($parm2 == $null) halt
  else {
    %pt = $ctime - $parm2
    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 not too close and not too far
    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

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. Many identifers also perform functions on data that you supply and then return a result.

Identifiers which cannot be evaluated or evalute 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.

$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 formant: Wed Jun 26 21:41:02 1996

$idle
Returns your current idle time (same time as that returned by a ctcp finger).

$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)
Returns the timer id of the Nth timer in the timers list.

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.

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

$chr(N)
Returns the character with ascii number N.

$chr(65) returns A
$chr(42) returns *

$count(string,substring)
Returns the number of times substring occurrs in string.

$count(hello,el) returns 1
$count(hello,l) returns 2

$left(N,text)
Returns the N left characters of text.

$left(4,goodbye) returns good

$len(text)
Returns the length of text.

$len(#mIRC) returns 5

$lower(text)
Returns text in lowercase.

$lower(HELLO) returns hello

$mid(S,N,text)
Returns the N characters starting at position S in text.

$mid(3,4,othello) returns hell

$pos(string,substring)
Returns a number indicating the position of substring in string.

$pos(hello,el) returns 2
$pos(hello,la) returns $null

$longip
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

$rand(v1,v2)
This works in two ways. If you supply it with numbers for v1 and v2, it will return a random number between v1 and v2. If you supply it with letters, it will return 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

$replace(string,substring,replacement)
Replaces any occurrence of substring in string with replacement.

$replace(abcdefg,cd,xyz) returns abxyzefg

$right(N,text)
Returns the N right characters of text.

$right(5,othello) returns hello

$str(N,text)
Returns text repeated N times.

$str(3,ha) returns hahaha

$strip(text)
Returns text with all bold, underline, reverse, and colour control codes stripped out.

$upper(text)
Returns text in uppercase.

$upper(hello) returns HELLO

File and Directory identifiers

$dir, $file, and $hfile
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.

$dir[="Select a file"] <path and filename>
$file[="Select a file"] <path and filename>
$hfile[="Select a file"] <path and filename>

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.

$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

$getdir
Returns the DCC Get directory specified in the DCC Options dialog.

$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 4.5\mirc32.exe) returns 605694

$logdir
Returns the Logs directory as specified in the Logging 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.

$read
Reads a line from a file and inserts it into the current position in an alias command. The format is:

$read [-nl# -stext] <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.

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 could slow down processing. For small files, this will make no difference, but for large files with over 1000 lines the delay will be noticable.

$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 <filename> <section> <item>

/echo $readini mirc.ini mIRC nick

Reads your nickname from the mirc.ini file.

$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.

$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 address 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).

$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

$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

This standard set of masks is also used in other identifiers and commands.

$me
Returns your current nickname.

$nick(N,#)
Returns Nth nickname in the channels nickname listbox on channel #.

$nick(0,#mIRC) returns the the total number of nicknames on #mIRC
$nick(1,#mIRC) returns the 1st nickname on #mIRC
$nick(22,#mIRC) returns the 22nd nickname on #mIRC

$nopnick(N,#)
Returns the Nth non-Op nickname on channel #.

$nopnick(0,#mIRC) returns the the total number of non-Ops on #mIRC
$nopnick(1,#mIRC) returns the 1st non-Op nickname on #mIRC
$nopnick(9,#mIRC) returns the 9th non-Op nickname on #mIRC

$notify(N)
Returns the Nth notify nick currently on IRC.

$notify(0) returns the total number of notify nicks on IRC.
$notify(3) returns the 3rd notify nick on IRC.

$opnick(N,#)
Returns the Nth Op nickname on channel #.

$opnick(0,#mIRC) returns the the total number of Ops on #mIRC
$opnick(1,#mIRC) returns the 1st Op nickname on #mIRC
$opnick(9,#mIRC) returns the 9th 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(0,#mIRC) returns the the total number of selected nicknames on #mIRC
$snick(1,#mIRC) returns the 1st selected nickname on #mIRC
$snick(12,#mIRC) returns the 12th selected nickname on #mIRC

$snotify
Returns the currently selected nickname in the notify list box.

Window identifiers

$active
Returns the full name of the currently active window.

$chan(N/#)
Returns information on channels that you are currently on.

Properties: topic, mode, key, limit

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

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)
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)
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)
Returns the nickname and filename of the Nth open dcc get window.

Properties: ip, status, file, 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 recieved 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)
Returns the name of the Nth open query window.

$query(0) returns the total number of open query windows.
$query(2) returns the name of the 2nd open query window.

$send(N)
Returns the nickname and filename of the Nth open dcc send window.

Properties: ip, status, file, 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 recieved 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

$findtok(text,token,C)
Returns the Nth position of token in text.

$findtok(a.b.c.d,c,46) returns 3
$findtok(a.b.c.d,e,46) returns $null

$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

$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

$remtok(text,token,C)
Removes one matching token from text.

$remtok(a.b.c.d,b,46) returns a.c.d
$remtok(a.b.c.d,e,46) returns a.b.c.d
$remtok(a.c.c.d,c,46) returns a.c.d

$reptok(text,token,new,C)
Replaces one matching token with a new token in text.

$reptok(a.b.c.d,b,e,46) returns a.e.c.d
$reptok(a.b.c.d,f,e,46) returns a.b.c.d

Miscellaneous identifiers

$away
Returns the value **$true** or **$false** depending on whether you are marked as away or not.

if ($away) say I'm away! | else say I'm here!

$cb
Returns the first 256 characters of the clipboard contents.

$cr
Returns the **carriage return** character, the same as $chr(13).

$host
Returns your Local host name.

$ip
Returns your IP address.

$lf
Returns the **linefeed** character, the same as $chr(10).

$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.

$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)
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

$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.

Fonts

This pops up the **Font** dialog for the **currently active** window.

For certain **types** of windows you can set the **selected** font as the **default** font for **all** windows of that type.

You can also pop up the font dialog in a window by typing **/font**.

Flood

This turns on flood protection and attempts to prevent mIRC from flooding a server by counting the number of bytes you send to a server, and initiating a flood check if you exceed a certain maximum number of bytes.

Trigger flood check after...
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 buffering in an attempt to satisfy the maximum number of users as possible 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

For example, you can **underline** a single word in a sentence by:

1.Typing Control-U
2.Typing in the word
3.Typing Control-U again

This way **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.

For example, you can **colour** a single word in a sentence by:

1.Typing Control-K
2.Typing a number between 0 and 15
3.Typing the word
4.Typing 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, where zero is white, 1 is black, etc.

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.

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.

ON VOICE/DEVOICE

These two events are concerned with user voice. The event is performed for each user that is voiced/devoiced:

100:ON VOICE:#:/mode $chan -v $vnick | /msg $nick Don't Voice that person!

When a user with access level 100 is voiced, you immediately devoice them and send a mesage to the person that voiced them.

100:ON DEVOICE:#mIRC:/mode $chan +v $vnick | /msg $nick Hey! That person is my friend!

When a user with access level 100 is devoiced, you immediately voice them and send a mesage to the person that devoiced them.

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

>=2:ON DEVOICE:#mIRC:/msg $chan $nick devoiced $vnick (legal)
1:ON DEVOICE:#mIRC:/msg $chan $nick devoiced $vnick (illegal)

In this situation, if the devoicers level is larger than or equal to the devoiced users level, then it is a legal devoice. Otherwise, it defaults to the second ON DEVOICE line which indicates that it is an illegal devoice. Remember, this is comparing the voicers and voiced users levels and has nothing to do with with the level "2" in the definition.

Note that **these events work on nicknames and not addresses.** This is because the IRC server only sends the address of the user doing the voicing/devoicing and not of the users affected ie. it only sends the nicknames of the users being voiced/devoiced.