IRC

COLLABORATORS			
	TITLE : IRC		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 1, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

IRC

1.1 IRC II-Command-Help V1.2, done in 1992 by Thomas Witt (`Bach')

```
                                IRC II - Help.guide   Main-Menu
                      *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*
Commands:

                 !

                 #

                 ADMIN

                 ASSIGN

                 AWAY

                 BASICS

                 BYE

                 CD

                 CHANNEL

                 CLEAR

                 COMMANDS

                 COMMENT

                 CONNECT

                 DATE

                 DESCRIBE

                 DIE

                 DMSG
```

NICK

NOTICE

NOTIFY

OPER

PART

QUERY

QUIT

QUOTE

REDIRECT

REHASH

RESTART

RULES

SAVE

SAY

SEND

SERVER

SIGNOFF

SLEEP

SQUIT

STATS

SUMMON

TIME

TOPIC

TRACE

TYPE

USERS

VERSION

WAIT

```
 WALL

 WALLOPS

 WHILE

 WHO

 WHOIS

 WHOWAS

 WINDOW

 XECHO
Command-Groups:

 ALIAS

 BIND

 CTCP

 DCC

 IRC II

 LOAD

 NOTE

 ON

 SET
```

1.2 Help for IRC II - Help :)

```
                Welcome to the  IRC II - Help.guide Version 1.2
                *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*

             This helpfile was done on 15.11.2992 by Thomas Witt.

             This file is FD, but please don't change my Name, the
                  copyrights and all other things like these.

           Send Suggestions/Money/Love-Letters/Flowers/Computers to:

                                 Thomas Witt
         IRC-Nick: Bach / InterNet E-Mail: mozart@zelator.in-berlin.de

---

                                Greetings to:

             Alectron, Amarie, Gollum, Guenthi, Hawk, Helmut, LP,
```

Macbeth, Mumsa, OB, Ray, Strubi, Waz, Wolfi, Wuerg.

FNM, Guardian, Ill, Laire, mlelstv, Pjotr, Radix, ScottE,
Stargazer, Tron, Udance, Zop.

and to all other important people on the #Amiga-Channel. ;)

Amit, Axel, Britta, Carsten, David, Erik, Ernst, Frank, Frederik,
Inga, Jan, Johannes, Maja, Thilo, Wolfgang.

1.3 Recall previous commands

```
              Usage: ![history number|history match]
The ! command is used to recall previous commands in your command
history for re-execution.  The ! command is unique in that when it
is used, it leaves the matching history entry in the input line for
re-editting.  You can specify a history entry either by its number
in the history list, or by a match with a given wildcard expression.
For example:
  /!10
will put entry 10 in the history list into the input line.
The following:
 /!/MSG
will search the history list for a line beginning with /MSG and
put it in the input line (an * is implied at the end of /MSG).  When
using ! command with a wildcard expression, subsequent uses of
the ! command continue their search in the history list from where
they left off.  This is reset to the end of the list when a command
is executed that adds an entry to the history list. Also, if a
wildcard expression is used once, subseqent uses of ! with no
expression will use the previous wild card expression.  For example,
the following:
  /!/MSG
  /!
The first call returns the first match of /MSG in the history list,
and the second returns the next match, and so on.  This is useful
in the following key binding:
  /BIND ^R PARSE_COMMAND /!$"Search: "
With this, you can hit ^R and you will be prompted for a search
string in the history list.  If you want to then repeat the search
you simply have to hit ^R and hit return, since it will use
the previous search by default.

Lastly, ! may be used in command aliases as well.  When it is
embedded in an alias it simply executes the matching history entry
without first putting it in the input line for re-editting.

See Also:

               HISTORY
```

1.4 Comment-Sign for Scripts

```
Usage in scripts and aliases.
It's identical to COMMENT and does nothing.
```

1.5 Shows a server-administrator

```
Usage: ADMIN [server]
  Displays the administrative details about the given server.  If
  no server is specified, the server you are connected to is
  used.
```

1.6 Sets a variable

```
                Usage: ASSIGN [-][user-variable] [whatever]
  ASSIGN lets you create user variables which are automatically
  expanded by IRCII in aliases using the $variable-name format.
  For example, if you had defined:
    /ASSIGN me The Big Cheese
  Then, in an alias, $me would expand to "The Big Cheese".  These
  are straight textual substitutions.  You can use these kind
  of variables as counters and indexes into lists as well.  For
  example:
  /ASSIGN index 0
  /ASSIGN index ${index+1}
  The first line sets "index" to 0, the seconds increments index by 1.
  Note that the second will always be expanded if used in another
  alias (where $ are normally expanded).  If you want it to be
  expanded if typed at the input line, you must first set
  INPUT_ALIASES to ON.

  Note: In the above example, the mathematical expression must be
  enclosed by {}'s  otherwise the + would be treated as an alias
  delimiter and the addition not performed.

  Suppose now you wanted to use index to get a single word in a list,
  you could do:
  /ALIAS WORD /ECHO $($index)
  Then doing:
  /WORD This is a test
  would display the index'th word in the list.
  The $($index) format does the following: it parses the text
  between the () as a sub-alias.  The "$index" is thus evaluated
  and returns it's value.  This value is then replaced for the ()
  expression and evaluated.  So, if you had:
  /ASSIGN index 1
  Then, the following would be evaluated:
  $index becomes 1 which given
  $(1) which becomes simply $1
  Thus the $1 arguement is used.
  This functionality can be nested. This the following:
  /ASSIGN A Hey You!
  /ASSIGN B A
```

```
/ASSIGN C B
/ALIAS NESTING /echo $($($C))
will cause /NESTING to display "Hey You!"

The following format are also legal:
  $#name  evaluates to the number of words in name
  $@name  evaluates to the number of characters in name
You can use these with the mathematical expressions as well,
for example:
  /ASSIGN foo Testing One Two
  /ECHO $#foo $@foo
  /ASSIGN junk ${#foo + @foo}
The ECHO line will display 3 15 and the variable junk have the
value 18.

Additionally, values assigned to FUNCTION_RETURN are taken to be
the return value of a function.

See Also:

               ALIAS

               IF
```

1.7 Marks you as away

```
              Usage: AWAY [away message]
Marks you as "away".  Whenever someone sends you a MSG or a
does a WHOIS on you, they automatically see whatever message
you set.  While you are away, all messages you receive will be
timestamped and you will optionally get beeped (see SET
BEEP_WHEN_AWAY) for each message.  Using AWAY with no
parameters marks you as no longer being away.

See Also:

               SET BEEP_WHEN_AWAY
```

1.8 Some basics for newusers

```
                Irc is a multi-user, multi-channel chatting network.  It allows
people all over the internet to talk to one another in real-time.
Each irc user has a nickname they use.  All communication with
another user is either by nickname or by the channel that they or
you are on.  All IRCII commands begin with a / character.
Anthing that does not begin with a / is assumed to be a message
that is sent to everyone on your channel.  Here is a list of
basic commands to help you get started:
    /
                 LIST
                             Lists all current irc channels, number of
```

```
                        users, and topic.
    /
                NAMES
                           Shows the nicknames of all users on each
                        channel
    /
                JOIN
                <channel>   Join the named channel.  All non-commands
                        you type will now go to everyone on that
                        channel
    /
                MSG
                <nick> <msg> Sends a private message to the specified
                        person.  Only the specified nickname will
                        see this message.
    /
                NICK
                            Change your nickname
    /
                QUIT
                            Exits irc.
    /
                HELP
                <topic>     Gets help on all IRCII commands.
    /
                WHO
                <channel>    Shows who is on a given channel,
                        including nickname, user name and host,
                        and realname.
    /
                WHOIS
                <nick>     Shows the "true" indentity of someone

These commands should get you started on irc.  Use the /
                HELP
               command to find out more about things on irc, or ask question of
people... most would be happy to help you out.
```

1.9 Exit IRC

```
               Usage: BYE
  Quits your IRC session.

                QUIT

                EXIT
                and
                SIGNOFF
                are identical.
```

1.10 Change directory

```
Usage: CD [path]
  If a path is specified, this changes IRCIIs working directory.
  This is the default directory for EXEC's and other things
  (such as logs) that don't have absolute paths specified.
  CD with no arguments shows you the current directory.
```

1.11 Join a channel

```
                Usage:  CHANNEL [<channel>|-INVITE|-NICK <nickname>]
  Changes your current channel to the channel specified.  If the
  -INVITE switch is used, you will join the channel that you were
  last INVITED to.  Or, if we wish to be grammatically correct,
  you will join the channel to which you were last invited.  If
  the -NICK format is used, you will join the channel that the
  specified nickname is on.  If no parameters are given, your current
  channel is displayed.

  Information about the different channels:
    Channel 0 is the limbo channel.
    Channels 1 thru 999 are public.
    Channels 1000 and up are private channels
    Channels below 0 are private/invisible channels.
    Channels may contain alphabetic characters if they
      begin with a + character.  For example:
  +My_Channel
      is a valid channel name.
    With server version 2.6 and beyond, there is a new channel
    type.  A channel may begin with a # character.  The
    following characters may be alphanumeric like +
    channels.  However, you may join more than one #
    channel at a time.  You can still be on 1 numeric
    or + channel and up to (at last check) 10 # channels.

  CHANNEL and
                 JOIN
                 are identical.
  See HELP MODE for information about changing channel characteristics.
```

1.12 Clear Window

```
Usage: CLEAR [-ALL]
  Clears the current window.  If the -ALL flag is used, clears all visible
  windows.
```

1.13 Some basics for newusers

```
                All IRCII commands must begin with one of the CMDCHAR settings
(see
                 SET CMDCHARS
```

). By default, this is set to /. Thus, any input line whose first character is a / is treated as an IRCII command. For example:

```
/MSG BigCheese Howdy!
```

This executes the MSG command, which sends a private message to "BigCheese", and that message is "Howdy!"

You can create aliases for commands or redefine what an existing command does. This is done with the ALIAS command. For example:

```
/ALIAS M /MSG
```

This example creates a new command, M, which does the same thing as MSG , you can now do:

```
/M BigCheese Howdy!
```

and it will work exactly like MSG. Suppose however you did this:

```
/ALIAS MSG //MSG BigCheese
```

Now, the MSG command will only send messages to BigCheese:

```
/MSG Howdy!
```

Note that in the alias, there are two / in front of MSG. Putting two of the CMDCHARS in front of a command tells IRCII that you want to use the original command, and not any alias of it. Therefore, even if you had the above alias for MSG, you could still do the following:

```
//MSG Cheese I can still use the original MSG command.
```

Furthermore, if you wish you can turn off your display (See SET DISPLAY) for the duration of a command. This is done by putting a ^ character between the / and the command word. For example, if you do:

```
/^MSG BigCheese Hello
```

You will not see any visible effect to this command (unless the nickname BigCheese does not currently exist).

Lastly, if you do the following:

```
/ /this is a test
```

```
Note there is a space after the /.  This form forces the following
text to be sent to your current channel no matter what.  You can
thus send lines that begin with your
                 CMDCHARS
                 .  It can also
be used to force a message to your current channel when you are
using
                 QUERY
                .
```

1.14 Sets a comment in a script

```
Usage: COMMENT <anthing you want>
  This is a comment, does nothing.  Useful in .ircrc files.
```

1.15 Attempts to connect another server (Operator-Command)

```
Usage remote: CONNECT <serverX> <port> <serverY>
  Attempts to connect serverY to serverX on the given port.
```

1.16 Shows time/date

```
                Usage: DATE [server]
  Shows the current time of day and date.  If a server is
  specified, the time of day and date are reported from that
  server.  DATE and
                 TIME
                 are identical.
```

1.17 Describes an Action

```
                Usage: /DESCRIBE <nickname|channel> <action description>
    This command can be used to send a description of what you are doing
    or how you are feeling or just about anything concerning you, to the
    person or channel you pass as first argument.

/DESCRIBE BigCheese takes a flask of whisky out of the fridge.

    is supposed to produce a line on BigCheese’s screen stating:

Lynx takes a flask of whisky out of the fridge.

    This command makes use of CTCP and is not understood by all clients.
    If you get an error message, your description has not arrived properly.
```

```
  See Also:

               ME

               LOAD ACTION

               LOAD MUD

               CTCP

               CTCP CLIENTINFO
```

1.18 Shutdown a server (Operator-Command)

```
Usage: DIE
  Causes your server to exit, die, pack it in, give up the ship,
  etc.
```

1.19 Send message across a direct client client connection

```
                Usage: DMSG nickname
    DMSG sends a message across a direct client connection to the given
    nickname. The connection must first have been established with DCC
    CHAT. DMSG messages are the most secure messages you can send to
    another user on IRC, as they are passed directly from your client
    to the client of the recipient, without passing through servers.
  See Also:

               MSG

               DCC CHAT
```

1.20 Starts a DCC conversation with someone

```
                Usage: DQUERY [nickname]
  Starts a DCC conversation with [nicknames].  All text you
  type that would normally be sent to your channel now goes to
  [nicknames] in the form of DMSGs.  To cancel a DCC
  conversation, use DQUERY with no arguments.

  See Also:

               DCC CHAT

               DMSG
```

1.21 Print a text

```
              Usage: ECHO <anything you want>
This simply displays all of it’s arguments.  However, it’s not
that simple.  ECHO will work even if DISPLAY is set to OFF,
making it useful in your .ircrc file.  Also, ECHO will work in
the action portion of an ON command in all modes.  This can be
used to redefine the way messages appear.

See Also:

               ON

               SET DISPLAY

               XECHO
```

1.22 Encrypt MSG’s

```
              Usage: ENCRYPT [nickname|channel] [key]
ENCRYPT allows you to hold an encrypted conversations with
a person or a whole channel.  Once a nickname/channel and
key is specified, all messages you send to that
nickname/channel will automatically be encrypted using the
given key.  Likewise, any messages from that nickname/channel
will automatically be decrypted.  IRCII is smart enough to know
if the incoming message isn’t encrypted and will not attempt to
decrypt it.  If you received an encrypted message from someone
for whom you haven’t specified a key, it will be displayed as
[ENCRYPTED MESSAGE].

The [key] can be any text which is to be used an they key
for encryption/decryption of the conversation.  It is up to
you and the people you wish to talk to about how to agree upon
a key and how to communcate it to one another.

For example, if user CheeseDog wishes to talk encryptedly with
user DogCheese, they must first agree on an encryption key
(case sensitive), say foo.  Then user CheeseDog must issue a
  ENCRYPT DogCheese foo
and user DogCheese must issue a
  ENCRYPT CheeseDog foo
Thereafter, all messages sent between CheeseDog and DogCheese
will be encrypted and decrypted automatically.

If ENCRYPT is given with a nickname but no key, then encrypted
conversation is ended with that user.  If ENCRYPT is given with
no arguments, the list of encrypted user and keys are
displayed.

IRCII uses a built in encryption method that isn’t terribly secure.
You can use another if you so choose, see SEE
               SET ENCRYPT_PROGRAM
                for information about this.
If you are sending encrypted messages to a user or channel, you
can toggle it off and on in a message line by inserting the
control-E character in the input line.  This is usually done
by hitting control-Q then control-E.  An inverse video E will
appear in the input line.
```

1.23 Some basics for newusers

```
                HOW TO BEHAVE ON IRC

Authors:      Lea Viljanen (LadyBug)  viljanen@kreeta.helsinki.fi
              Ari Husa     (luru)     so-luru@tolsun.oulu.fi
Modified by:  Troy Rollo   (Troy)     troy@plod.cbme.unsw.oz.au



1) Language

   The most widely understood and spoken language on IRC is English.
However! As IRC is used in many different countries, English is by
no means the only language. If you want to speak some other language
than English (for example with your friends), go to a separate channel
and set the topic (with /
                 Topic
                 ) to indicate that. For example
   /topic Finnish only!
would mean that this channel would be reserved for Finnish discussion.
On the other hand, you should check the topic (with /list command)
before you move to a channel to see if there are any restrictions about
language.
   On a channel not restricted by /topic, please speak a language
everybody can understand. If you want to do otherwise, change channels
and set the topic accordingly.


2) Hello/Goodbye

   It's not necessary to greet everybody on a channel personally.
Usually one "Hello" or equivalent is enough. And don't expect everybody
to greet you back. On a channel with 20 people that would mean one
screenful of hellos. It's sensible not to greet, in order not to be rude
to the rest of the channel. If you must say hello to somebody you know,
do it with a private /
                 Msg
                 . The same applies to goodbyes.

   Also note that using IRCII's /
                 ON
                 facility to automatically say hello
or goodbye to people is extremely poor etiquette. Nobody wants to receive
autogreets. They are not only obviously automatic, but even if you think
you are being polite you are actually sounding insincere and also
interfering with the personal environment of the recipient when using
autogreets. If somebody wants to be autogreeted on joining a channel,
they will autogreet themselves.


3) Discussion

   When you come to a new channel it's advised you to listen
for a while to get an impression of what's discussed. Please feel free
to join in, but do not try to force your topic into the discussion
if that doesn't come naturally.


** 4) Limits

   On channel numbers above 10 there can be only 10 persons
per channel. So if the channel is full and you don't actively
participate, please move to another channel to let someone else in.


5) {}|[]\

   IRC has quite a lot of people from Scandinavian countries,
the above characters are letters in their alphabet. This
has been explained on IRC about a thousand and one times, so
read the following, do not ask it on IRC:

   {     is an A with 2 dots over it
   }     is an A with a small circle above it
   |     is either an O with 2 dots over it or an O with a dash (/) through it
   [, ], and \ are the preceding three letters in upper case.


6) ATTENTION!

   Remember, people on IRC form their opinions about you only by
your actions, writings and comments on IRC. So think before you type.
```

1.24 Executes an UNIX-Command

```
              Usages:
  EXEC <shell commands>
  EXEC -NAME <name> <shell commands>
  EXEC -OUT [%<process id>|<shell commands]
  EXEC -MSG <nickname> [%<process id>|<shell commands]
  EXEC -IN %<process id> <text to send to process>
  EXEC -WINDOW [%<process id>|<shell commands>]
  EXEC -<signal> %<process id>
  EXEC -CLOSE %<process id>

  EXEC allows you to start subprocesses in IRCII and manipulate
  them in various ways.  You can start multiple subprocesses
  simultaneously and access them via a process number assigned
  by IRCII.  You can list all currently running subprocesses by
  using EXEC with no parameters.  The process id of a process
  is the number assigned by IRCII for that process, or the
  name of the process given by the -NAME flag.  If a NAME is
```

```
given to a process, that name may be used anyway in place of
the process number assigned by IRCII.

The first form of EXEC will simply start a subprocess and send
it's output to your display.

The second form tells IRCII to send the output of the process
to your current channel.  For example:
  EXEC -OUT ls
sends the output of ls to your channel.
  EXEC -OUT %1
tells IRCII to send the output of subprocess 1 to your
channel.  Subprocess 1 must exist already by a previous call
to EXEC.

The third form is much like the second, except that it sends
to the specified nickname or nicknames (the format of the
nicknames is the same as for MSG).  As with the second form,
you can start a subprocess with -MSG, or you can change an
already running process to send it's output to the given
nicknames.

The fourth form lets you send a line of input to a running
subprocess.  For example:
  EXEC -IN %shell This is a test.
Sends "This is a test." to subprocess 0.  This processes must
have prevously been started with a call to EXEC -NAME shell.
An alternate method of sending text to processes is using the
MSG or QUERY command.  In the place of a nickname, you may
specify %n, when  n is a current running processes id.   For
example:
  MSG %shell This is a test.
is equivalent to the previous example.

The fifth form lets you specify that you want all output from
the process to go to the current window.  Normally, output
from processes goes to whichever window has a level setting of
CRAP.  This locks the output into the current window.
  EXEC -WINDOW %1
Sends the output of process 1 to the current window.

The last form lets you send various signals to subprocesses.
The allowable signals are:
  HUP     INT     QUIT     ILL     TRAP     IOT     EMT
  FPE     KILL    BUS      SEGV    SYS      PIPE    ALRM
  TERM    URG     STOP     TSTP    CONT     CHLD    TTIN
  TTOU    IO      XCPU     XFSZ    VTALRM   PROF    WINCH
  LOST    USR1    USR2
What these signals do depends on the process running, etc.
  EXEC -KILL %0
Sends a KILL signal to process 0, forcing it to exit
immediately.  If you want to read more about these signals, do
a "man kill" at your shell prompt.

The last form is for really ornery processes that simply won't
die.  Sometimes this is because an EXEC'd process has forked
off subprocesses which don't die when you use -KILL (or other
flag).  Doing a:
  EXEC -CLOSE %0
closes all of IRCII’s connections to that processes.  This means
that even if the processes is still sending output you won’t see
it.  This also means (in most cases) that the process will be
killed by a SIGPIPE when it tries to send to IRCII.

See Also:

               SET SHELL

               SET SHELL_FLAGS

               SET SHELL_LIMIT

               SET NOTIFY_ON_TERM
```

1.25 Exit IRC

```
              Usage: EXIT
Quits your IRC session.

               BYE

               QUIT
               and
               SIGNOFF
               are identical.@endnode
```

1.26 Flush server-output

```
              Usage: FLUSH
Flushes all pending output from the server.  This has the
effect of stopping long lists from the server, such as a
               LINKS
                command which can get quite lengthy.
```

1.27 Gives you help on IRC (not needed, you have THIS :))

```
              Usage: HELP [command]
Shows help on the given command.  The help documentation is
set up in a heirarchical fashion.  That means that certain
help topics have sub-topics under them.  For example, doing
  /HELP ADMIN
gives help on the admin command, while:
  /HELP SET
gives help on the set command and also displays a list of
sub-topics for SET.  If you are using IRCIIHelp, then to get
help on the subtopics, you would do:
 /HELP SET <subtopic>
```

```
where <subtopic> is one of the subtopics.  If you are using the
built in help, then you need only type the subtopic name.  The
input prompt will indicate what help level you are on.  Hitting
return will move you up one level.

At any time, you can specify a ? to get a list of subtopics
without the associated help file, for example:
  /HELP ?
gives a list of all main help topics.  The follwoing:
  /HELP BIND ?
gives the list of all BIND subtopics.  If you use a ? with
a topic that has no subtopics, you will simply get the
standard help file for that topic.

See also:

               About this IRC.guide - File
```

1.28 Shows you the command history

```
              Usage: HISTORY [number]
Displays the command history to the screen.  You can specify
the number of history entries you wish to view as well.

See Also:

               SET HISTORY
```

1.29 Execute command when a boolean expression is true/false

```
              Usage: IF boolean true-command [false-command]
IF is a method of executing different commands depending on
the truth of a boolean expression.  The boolean in the
IF command can be one of the following:
exp
exp1=exp2
exp1!exp2
exp1>exp2
exp1<exp2
Here, exp, exp1, and exp2 can either be numeric or string data.
If just exp is used, then the true-command is executed if exp is
either a non-empty string or a non-zero number.   Otherwise,
false-command is executed.  If false-command is not present,
no command is executed.
In the other expressions, if both exp1 and exp2 are numeric, then
the following cause true-command to be executed:
=   if exp1 is equal to exp2
! if exp1 is not equal to exp2
> if exp1 is greater than exp2
< if exp1 is less than exp2
Otherwise, false-command (if present) is executed.
If both exp1 and exp2 are non-numeric, then the following cause
```

```
  true-command to be executed:
  = if exp1 is the same as exp2
  ! if exp1 is not the same as exp2
  > if exp1 is alphabetically greater than exp2
  < if exp1 is alphabetically less than exp2
  All string comparisons are case-insensitive: so "Hello=HELLO"
  is true.
  NOTE: No spaces are allowed in the boolean expression.  You can
  however put double quotes around a boolen expression to have it
  include more than one word.

  Both the true-command and false-command may either be a single one
  word command, or you may surround a command and it's arguments in
  double quotes.

  Examples:
  /IF "$^!^=^<^>C" "/set input_prompt $C" "/set -input_prompt"
  /IF "$^!^=^<^>C!0" "/set input_prompt $C" "/set -input_prompt"
  Both of the above evaluate the same way, since $C will either expand
  to your current channel or 0 if none.  If you aren't familiar with what
  $^!^=^<^>C means,  I'll explain.  $C expands to the current channel.
  Putting in ^c (where c is any character) tells IRCII to quote c with
  a \ before substitution.  Using ^!^=^<^> will cause any operators
  embedded in channel names to be quoted and not interfere with the
  operation.

  NOTE: if you type the above examples in at the command line, the $C
  will only be converted if you have
               INPUT_ALIASES
               ON.  These examples
  will all work fine if embedded in an
               ALIAS
               .

  /IF "$USER=ms5n" "/echo Hello Mike!" "/echo Who the hell are you?"
  This example will look for an ASSIGN'd variable named USER, and if found
  replace it and compare to ms5n.  If no ASSIGN'd variable exists, it
  will check the environment variables for USER and return that.

  /IF "$index>${max*2}" "/echo Limit reached"
  This checks the ASSIGN'd variable against the ASSIGN'd variable max times 2
  and displays "Limit reached" if the first is greater than the second.
  It does nothing if this is not true.
```

1.30 Ignore MSG's/Notices from a user on IRC

```
Usages:
  IGNORE [nick] [-|+|^][message type]
  IGNORE [user@host] [-][message type]

  The standard form of IGNORE allows you to specify which type of
  messages you wish to ignore from a user.  The message type
  parameter can be one of the following:
    MSGS            All MSGs received
    NOTICES         All NOTICEs received
    PUBLIC          All normal channel conversation
    INVITES         All INVITEs received
    WALLS           All WALLs received
*    WALLOPS         All WALLOPS received
    NOTES           All NOTEs received
    ALL             All of the above message types
    NONE            No message types

  You can ignore by nickname or by specifying a userid@hostname
  format.  Wildcards may be used in all formats.  You can specify
  multiple types of messages to ignore on the command line.
  Preceeding a type with a - indicates removal of ignoring of
  that type of message.  For example:
     IGNORE BigCheese PRIVATE INVITES
  ignores MSGs and INVITES from BigCheese.
     IGNORE *@*.cmu.edu ALL -WALLS
  ignores all types of messages except WALLS from anyone from
  CMU.  The user@host format is considerably slower than the
  nickname format and doesn't function for PUBLIC messages.

  You may also specify a + before any message type to cause messages
  of that type from that user to be displayed with the nickname
  highlighted.  For example:
    IGNORE *Cheese* +PRIVATE
  will cause all MSGs from anyone with Cheese in their nickname
  to appear with their nickname highlighted.
    IGNORE * +WALLS +NOTES
  will highlight all WALLS and NOTES received.  The + has no effect with
  the user@host name format.

  If you specify a ^ before a message type, then messages from the
  matching nicknames will be displayed no matter what.  This allows
  you to exclude certain nicknames from larger matching lists.  For
  example:
    /IGNORE * MSG
    /IGNORE BigCheese ^MSG
  This will ignore private messages from everyone except BigCheese.

  All forms of IGNORE use the - to remove ignore attributes,
  and the type NONE will remove the user from the list
  completely.

  All forms of IGNORE will match against the nickname with the
  most true matching characters.  What this means is if you have:
    IGNORE BigCheese PRIVATE
    IGNORE * +ALL
  Then MSG's from BigCheese will be ignored, and all other types
  of messages from BigCheese will be unaffected.  All messages
  of all types from any other user will be highlighted.
```

1.31 Shows information about the IRC creators

```
              Usage: INFO
  Shows information about the IRC creators.
  See also:

                 About this IRC.guide - File
```

1.32 Invite a User to a channel

```
Usage: INVITE <nickname> [channel]
  Invites another user to a channel.  If no channel is specifed,
  your current channel is used.
```

1.33 Join a channel

```
                Usage:  JOIN [<channel>|-INVITE|-NICK <nickname>]
  Changes your current channel to the channel specified.  If the
  -INVITE switch is used, you will join the channel that you were
  last INVITED to.  Or, if we wish to be grammatically correct,
  you will join the channel to which you were last invited.  If
  the -NICK format is used, you will join the channel that the
  specified nickname is on.  If no parameters are given, your current
  channel is displayed.

  Information about the different channels:
    Channel 0 is the limbo channel.
    Channels 1 thru 999 are public.
    Channels 1000 and up are private channels
    Channels below 0 are private/invisible channels.
    Channels may contain alphabetic characters if they
      begin with a + character.  For example:
  +My_Channel
      is a valid channel name.
    With server version 2.6 and beyond, there is a new channel
    type.  A channel may begin with a # character.  The
    following characters may be alphanumeric like +
    channels.  However, you may join more than one #
    channel at a time.  You can still be on 1 numeric
    or + channel and up to (at last check) 10 # channels.


                ~CHANNEL
                 and JOIN are identical.
  See HELP
                 MODE
                 for about changing channel characteristics.
```

1.34 Kick a user off a channel

```
Usage: KICK <channel> <user>
  Kicks specified user off a given channel.  Only channel operators
  are privileged to use this command.  Channel operator privileges
  can be given to other users of channel by command
  /MODE <channel> +o <user>
  and taken away by command
  /MODE <channel> -o <user>
```

1.35 Kill a user from IRC (Operator-Command)

```
Usage: KILL <nickname> [comment]
  Removes a person from irc.  This is an operator command
  and should be used with extreme caution (if at all). The
  optional [comment] is usually supplied to indicate to
  other operators the reason for the kill.  If you don't give
  a comment, you will probably be bombarded with messages
  saying "why the kill?"
```

1.36 Shows the contents of the lastlog

```
              Usage: LASTLOG [flag] [number of entries|string] [from entry]
  Displays the contents of the lastlog.  This is a list of the
  most recent messages that have appeared on the screen.  LASTLOG
  is useful to redisplay the screen if you inadvertantly do a
  /CLEAR or miss messages for other reasons.  If no arguments are
  given, the entire lastlog is displayed.  If the first argument
  is a number, it determines how many log entries to show.
  Otherwise it is searched for in every lastlog entry.  The
  second argument determines how many lines back to start display
  from.  Thus
    /LASTLOG 4 8
  Shows 4 lines of the lastlog, starting at line 8...  lines 8,
  9, 10, 11 are displayed.  Furthermore:
    /LASTLOG bigcheese
  displays only those lastlog entries that contain the word
  "bigcheese".

  You can further limit the display of the lastlog by specifing
  one of the following flags:
    -PUBLIC   Normal channel messages
    -MSG    Private messages
    -NOTICE   Notices
    -NOTE   Notes
    -WALL   Walls
    -WALLOP   Wallops
    -CRAP   Anything not included in the other
      categories
  The lastlog will only display messages of the type specified
  by the flag.

  See Also:

               SET LASTLOG

               SET LASTLOG_LEVEL
```

1.37 Leave a channel

```
Usage: LEAVE <channel>
  Leave a channel.  This is used to leave a # channel while
  remaining on all other channels you are on.  Of course, you can
  use LEAVE to leave from any channel, not just a # channel.
```

1.38 Shows the serverconnections

```
Usage:  LINKS [servername]
  Shows all of the servers currently connected to the irc
  network.  If servername is specifed, LINKS shows any servers
  that match the given servername.  The servername may contain
  wildcards.
```

1.39 Lists all channels

```
Usage: LIST [flag [arg]] [channel]
  LIST gives you a listing of channels which includes channel name,
  number of users, and a topic (if one is set).  If no channel
  is specified, all channels are shown, otherwise only channels
  that match are displayed (the channel may contain wildcards)
  The displayed list may also be limited by using one or more of
  the following flags:
  -MIN n    When n the minimum number of user.  Channels
      with less than n users are not shown.
  -MAX n    When n the maximum number of user.  Channels
      with more than n users are not shown.
  -PUBLIC   Only shows Public channels
  -PRIVATE  Only shows Private (*) channels
  -ALL    Overrides previous -PUBLIC and/or -PRIVATE
  -TOPIC    Always show channels with a topic set
      (regardless of other switches)
```

1.40 Shows you some stats about IRC

```
Usage: LUSERS
  Gives a brief listing of the number of users, the number of
  servers and the number of operators.
```

1.41 Informations when you get UNIX-Mail

```
  The MAIL command in previous versions of IRCII has been changed
  to NOTE.  This is to minimize confusion between the server's
  delayed messaging capability (now the NOTE command), and the
  unix mail indicators built into IRCII.  If you get a message
  which says "You have new mail" or you see (Mail:1) in your
  status line, this means that you have unix mail and you should
  leave irc to read it.
```

1.42 Action-description

```
                Usage: /ME <action description>
 or    /<your_nickname> <action description>

    This command can be used to send a description of what you are doing
    or how you are feeling or just about anything concerning you, to the
    current channel or query. When I type:

/ME opens up the fridge.

    it’s supposed to produce a line on my talking partners screens stating:

Lynx opens up the fridge.

    You can also use your own nickname as command, that is you can type the
    line with a leading slash, like this:

/lynx reaches out for the orange juice.

    Hopefully resulting into:

Lynx reaches out for the orange juice.

    It is absolutely good style to not forget the period at the end!!

    This command makes use of CTCP and is not understood by all clients.
    If you get an error message, your description has not arrived properly.

  See Also:

                 DESCRIBE

                 LOAD ACTION

                 LOAD MUD

                 CTCP

                 CTCP CLIENTINFO
```

1.43 Changes the mode of a channel

```
Usage: MODE <channel> [+|-]<modechars> <parameters>
  Mode command is quite complicated and it allows channel
  operators to change channel mode.  <modechars> is one of
  the following:
    m           - channel is moderated (only channel operators talk)
    s           - channel is secret
    p           - channel is private
    l <number>  - channel is limited, where <number> is the
                  maximum number of users allowed
    t           - topic limits, only the channel operators may
                  change it
    a           - channel is anonymous
    o <nick>    - Makes <nick> a channel operator
    i           - channel is invite only
    n           - No MSGs to the channel are allowed
  A + or - sign determines whether the specified mode should be
  added or deleted.
```

1.44 Shows the servermessage-of-the-day

```
Usage: MOTD [server]
  Gives the message-of-the-day for the named server.  If no
  server is given, your server is used.
```

1.45 Sends a private message to a user or a channel

```
              Usage: MSG <nicknames|<channel>|%n|-CHANNEL <nickname>> <text of  ↩
                 message>
  Sends a private message to the nicknames or list of nicknames
  specified.  A list of nicknames should by separated by commas
  (no spaces).  Two special case nicknames are defined.  If the
  nickname is "," (a comma), the message is sent to the last
  person who sent you a /MSG.  If the nickname is "."  (a
  period), the message is send to the last person to whom you
  send a message.

  The second form allows you to specify a channel number or
  channel name to which to send the message.  The message will go
  to everyone on that channel.

  The third form lets you specify a process number to which the
  message will go, where n is the process number.  This works
  just like the EXEC -IN command.  For example:
    MSG %1 Homina homina homina
  will send the text to process 1.  Process 1 must be running
  already using a previous call to EXEC.

  The last form is similar to the second except that you can
  specify the channel to send a message to by giving the nickname
  of one to the people on the channel.  IRCII will look up the
  proper channel number or name and send the message.

*  There is also a use of wildcards for operators.
*  For example : /msg $*.edu will address all users having .edu
*                in their serverinfo.

  See Also:
```

```
                 SET BEEP_ON_MSG

                 EXEC

                 DMSG
```

1.46 Shows all users who are currently on IRC

```
Usage: NAMES [flag [arg]] [channel]
  Shows the nicknames of all users on all visible channels.
  If no channel is specifed, all channels are shown, otherwise
  only matching channels are shown (the channel may contain
  wildcards).  The displayed list may also be limited by using
  one or more of the following flags:
  -MIN n    When n the minimum number of user.  Channels
      with less than n users are not shown.
  -MAX n    When n the maximum number of user.  Channels
      with more than n users are not shown.
  -PUBLIC   Only shows Public channels
  -PRIVATE  Only shows Private (*) channels
  -ALL    Overrides previous -PUBLIC and/or -PRIVATE
```

1.47 Some news in IRC II V2.1.5

```
_____________________________________________________________________________
                             ircII 2.1.5
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> The command /who -h and -s is no more supported.
> Introduced { and } in scripts, see script/finger for an example.
  New also: You can indent things in a script. See Also /help load.
> Behaviour of HOLD_MODE has changed, won't hold when you are being active
  in that window. I like it a lot more, it's not so disturbing anymore.
> Away messages are displayed only once, when talking to someone who is away
  unless you /set show_away_once off.
> You don't get disturbed by cryptic but trivial CTCP requests, unless
  you /set verbose_ctcp on.
> Improved handling of process prompts, but INPUT_PROMPT must be set.
> ircII can now load scripts and help files in compressed form.
> All the scripts have been updated and many helpfiles written and
rewritten.  Many bugs have been fixed, support of newer server features
added, this client is already 2.7-capable.

> There have been major changes in the command parser. Commands are not
expected to have a leading / unless entered interactively, if you
experience incompatibilities, you will have to add 'say' and 'send'
wherever you have made aliases that send to the channel or query.
See Also /help ircII programming for this. Also the ';' has a new
meaning as command seperator within ALIAS, BIND and ON. You might have
to \escape it.

> The good news: Scripts are much more readable and understandable, look
a lot less ugly now  :)   COMMAND_MODE gives command-oriented freaks the
```

```
chance to use ircII that way. A special script for MUDders has been added
that emulates an Aber/LPMUD-like environment on irc, without having to
type / before "smile" and allowing "'<text>" for "say" etc.

> CTCP ACTION has been implemented which permits you to send descriptions
of actions/'feelings' to other people on channels and even through
queries.  You can use it with /me and the action script.
See Also: /help load action.
```

1.48 Some basics for newusers

```
                    NEW USER INFORMATION FOR IRC

This file contains some caveats for people new to IRC. It is not a
guide to commands for IRC. For a brief guide to commands for new
users, see /HELP INTRO.

See /HELP
                 ETIQUETTE
                 for a guide to good manners on IRC.

IRC is an international network servicing 20 or more countries. There
are over 2000 registered users, and a similar number of regular
unregistered users. Do not expect everybody to speak English.

The primary means of identification is currently by nickname. This can
be modified with /
                 NICK
                 , and registered with NickServ. (/MSG NickServ HELP
for information on NickServ). Note, however that some nicknames are
duplicated, and some people will impersonate others by using their
nicknames. If you have any doubts about the identity of somebody using
a given nick, use /
                 WHOIS
                 NickName to find out more. This gives detailed
information on the person using the NickName. For example, if you
are getting abusive messages from "Fred", type:

/WHOIS Fred

If the information displayed is not the same as that which you would
expect for Fred, it is probably a case of impersonation.

Recently some users have been tricking others into allowing them to
control their IRC sessions or damaging their files. If somebody asks
you to type a command and you don't know what it does, use
/HELP CommandName to find out more about it first. In particular,
/ON has been used to cause trouble and is now initially disabled
for new users. Additionally, /QUI is short for /QUIT and will
terminate your IRC session. If you see any message asking you to
type /QUI or /QUIT, ignore it.

Some new users have been baffled by "CTCP" messages appearing on
the IRC session. These are messages from other users, asking your
client to perform some service for them. They are generally
```

```
generated by somebody typing in a CTCP command. See /HELP
                 CTCP
                 .

If you have not already done so, read the HELP files INTRO and

                 ETIQUETTE
                 .
```

1.49 Changes your nickname

```
Usage: NICK <nickname>
  Changes your nickname to whatever you like.   The nickname may be at
  most 9 characters.  If you specify a nickname that is already in
  use, you will be prompted to enter a new nickname.
```

1.50 Sends a private message to a user or a channel

```
                Usage: NOTICE <nicknames|<channel>|-CHANNEL <nickname>> <text of ↩
                   message>
  Sends a private message to the nicknames or list of nicknames
  specified.  Unlike MSG, no automated response messages (such as
  generated by INGORE or an automaton) will be sent in response
  to a NOTICE.  Other than that, it's just like a MSG.

  For a description of parameters and switches see:

                 MSG
```

1.51 Give Message when a notified user leaves/enters IRC

```
Usage: NOTIFY [-]<nickname> { [-]<nickname> ... }
  Use of this command lets you mark certain nicknames such that
  you will be warned when they signon or off of IRC.  When
  someone on your notify-list signs on, you will see a message
  "Signon by <nickname> detected".  Likewise, when they signoff,
  you will see "Signoff by <nickname> detected".  Someone will be
  shown as signed off if they change nicknames as well, so keep
  this in mind.

  You can modify how the signon and signoff messages are
  displayed using the ON NOTIFY_SIGNON and ON NOTIFY_SIGNOFF
  commands.

  Specifying a - before a nickname will remove notification
  checking for that nickname.  If no parameters are specified,
  the current status of each user on your notify list is
  displayed.  If you specify a + without any additional
  parameters:
    /NOTIFY +
  you will be shown a list of only those people on your
  notification list who are currently on irc.
```

1.52 Gives you Operator-Priviledges (Operator-Command)

```
Usage: OPER <nickname> <password>
  Gives you operator priviledges if the correct nickname and
  password are given.  If password is not given, you will be
  prompted for one.  If no nickname is given, your current
  nickname will be used.
```

1.53 Leave a channel

```
Usage: PART <channel>
  Leave a channel.  This is used to leave from a # channel while
  remaining on all other channels you are on.  Of course, you can
  use PART to leave from any channel, not only # channels.
```

1.54 Start a conversation with a user

```
                Usage: QUERY [nicknames|%n]
  Starts a private conversation with [nicknames].  All text you
  type that would normally be sent to your channel now goes to
  [nicknames] in the form of MSGs.  To cancel a private
  conversation, use QUERY with no arguments.  The %n arguments
  allows you to send the query text to a process number n.  The
  process must have already been started using a previous call
  to EXEC.

  See Also:

                 EXEC
```

1.55 Exit IRC

```
                Usage: QUIT
  Quits your IRC session.

                 BYE

                 EXIT
                 and
                 SIGNOFF
                 are identical.
```

1.56 Sends a command directly to your server

```
Usage: QUOTE <server command>
  Send a command directly to the server.  No parsing is done by
  the client.
```

1.57 Send output from a command to a nickname/channel/process

```
Usage: REDIRECT <nick|channel|process> <cmd>
  This will send the output from the given command to the
  specified nickname, channel, or started process.  For
  example:
    /REDIRECT +Har /whois bigcheese
  This sends the result of "/whois bigcheese" to channel +Har.
  There is an implicit WAIT built into the REDIRECT command.
  This means that all output will be redirected until the
  given command has finished executing.  This can have some
  unpredictable results, so use it with caution.
```

1.58 Let the server re-read its configuration file (Operator-Command)

```
Usage: REHASH
  Forces the server to re-read its configuration file.
```

1.59 Restart the server (Operator-Command)

```
Usage: RESTART
  Causes your server to restart itself, or more correctly, to start up
  the binary with the path you have defined in the ircd's config.h.
```

1.60 Some basics for newusers

```
                Definitions:

   IRC-ADMIN - Person who have access to all files concerning ircd.
   IRC-OP    - Person who have privilegies given to him by an IRC-ADMIN.
               And who is in charge at HIS server.
   USER      - Person who is using IRC.

------------------------------------------------------------------------------

            Users Behavior on the Internet Relay Chat system



The Internet Relay Network (IRC) is a system for comunicating with other
peopples.
```

```
IRC is a free speech system. Users may exchange viewpoints with other
users. But to protect people from abuse there are certain rules that the
user MUST respect.

If a USER doesn't respect the guidelines/rules stated below, then the
IRC-ADMIN may suspend or reduce the availability for the USER.

These things are prohibited:

   * Using offensive words in channel topics.

   * Harassing another user. Harassment is defined as behavior towards
     another user with the purpose of annoying them.
     Harassment is a matter of opinion of the IRC-OP.

   * "Dumping" a lot of text to a channel.

   * Annoying another user or a channel with constant beeping.

   * Any behavior reducing the functionality of IRC.

What kind of action a user is doing to break these rules are up to the
IRC-OP to decide.

Violation should be straighted out via a civilized conversation between
the IRC-OP and the USER.
If the USER is not on-line then the USER should be notified by EMail.
If the USER wont respect what he's been told then it is up to the IRC-ADMIN
what to do with the USER.


If someone finds a USER violating these rules he may contact the USER's
IRC-ADMIN.
To find a USERS IRC-ADMIN one may use the /
                 ADMIN
                 <nick-name>.


Ove Ruben R Olsen
IRC-ADMIN for Bergen Edu. College, Norway.

EMail: rubenro@viggo.blh.no
```

1.61 Save currently IRC-Settings to a file

```
                Usage: SAVE [filename]
  This will save all of the settings of IRCII to a file.  It
  saves all of the Key Bindings, Variables, and Aliases (in that
  order) in a format such that they can be loaded into IRCII
  using the LOAD command or the -l switch.  If no filename is
  given, the your default .ircrc file will be used (unless
  changed with the -l switch).

  See Also:
```

LOAD

IRCII COMMAND_LINE_ARGS

1.62 Send a message to the current channel

```
Usage: SAY <text>
```

This command is to permit aliases, binds, scripts and similar to send a message to the current channel as if it were typed. This is equivalent to typing '/ <text>'.

See Also:

SEND

IRCII PROGRAMMING

1.63 Send a message to the current channel

```
Usage: SEND <text>
```

This command is to permit aliases, binds, scripts and similar to send a message to the current channel or query as if it were typed.

See Also:

SAY

IRCII PROGRAMMING

1.64 Changes your server

```
Usage: SERVER [+|-][<name of server>|<number in server_list>] [ ↩
   port number]
```

Switches your primary server to the server specified at the specified port number. If no port nunmber is given, the default port number is used. Your channel and AWAY status will remain unchanged. Occasionally, you can switch servers faster than the irc network can send out the information that you have left a server. So don't be suprised if it says your nickname is in use... just wait a moment and set it with
NICK
. IRCII
maintains an internal list of servers you have connected to. To see that list, use /SERVER with no arguments. You may also specify one of the servers in the server list by its number in the list, by issuing:

```
  /SERVER <number in list>
Likewise, you can automatically switch to the next or previous
server in the server list by doing:
  /SERVER +
or
  /SERVER -
to switch to the next and previous server in the list,
respectively.
Also, you can set a particular server to be associated with a
particular IRCII window.  To do this, set the current window to the
window you wish to use and do:
  /SERVER +servername
In this case, servername may again be a name, IP address, or number
of a server in your server list.
This will connect you to the given server as a secondary server in
that window.  Doing a /SERVER with no parameters will show you your
primary server.  If you kill the last window associated with a
server, the connection to that server will close.  If you kill the
last window associated with your primary server, a new open server
will be chosen as the primary server.  If there are open servers you
will not be allowed to close the connection to your primary server.
The following will also close a connection to a specific server:
  /SERVER -servername
When you create new windows, they will use as their associated
server whatever was the server for the current window when the new
window was created.  You can therefore have more than one window for
any server, but still only one primary server.

Every time you switch to a new server, it gets
added to the server list automatically.  To set up initial
servers in the server list, you can specify them on the command
line as such:
  irc <nickname> server1 server2 server3
Each host may have it's own port number and password specified
on the command line as well:
  irc <nickname> server1:6667 server2:7000:blah server3::foo
In this case, server1 will use port 6667, server2 will use port
7000 with password blah, and server3 will use the default port
number with password foo.  The -p switch on the command line
allows you to specify the default port number.  Also, if no
password is specified and one is needed to connect, you will be
prompted to enter a password before you can connect.  You can
also specify the default server list in the IRCSERVER
environment variable using the same format as above.

See Also:
```

IRCII SERVER_LISTS

IRCII COMMAND_LINE_ARGS

IRCII ENVIRONMENT_VARS

1.65 Exit IRC

```
                Usage: SIGNOFF
  Quits your IRC session.

                 BYE

                 EXIT
                 and
                 QUIT
                 are identical.
```

1.66 Suspend IRC for the given number of seconds

```
Usage: SLEEP <number of seconds>
  This suspends IRCII for the number of seconds given.  NOTHING happens
  during this time.  This is really only useful for automatons.
  Maybe not even then.
```

1.67 Sends a server quit to the named server (Operator-Command)

```
Usage: SQUIT <servername>
  Sends a server quit to the named server.
```

1.68 Shows some stats about IRC

```
Usage: STATS [L|C]
  Shows some irc server usage statistics.  If L is specified,
  more specific information is displayed on the current irc
  server connections.  If C is specified, the server's connections
  as specified in it's config file are displayed.
```

1.69 Summons a user to IRC

```
Usage: SUMMON <userid>[@server]
  Summons a user to irc.  The user currently must be logged into
  the server machine for this to function.
```

1.70 Shows time/date

```
                Usage: TIME [server]
  Shows the current time of day and date.  If a server is
  specified, the time of day and date are reported from that
  server.
                 DATE
                 and TIME are identical.
```

1.71 Changes the topic

```
Usage: TOPIC [<channel>] [<topic for channel>]
  Changes the topic for the named channel.
```

1.72 Shows the server connection of a server

```
Usage: TRACE <server>
  Shows the server connection of the given server.
```

1.73 Simulate keyboard keystrokes

```
Usage: TYPE <keystrokes>
  The TYPE command simulates keyboard keystrokes.  You can
  include control characters as well as normal characters and
  they will respond as they would as if typed at the keyboard.
  A control key is specified by a ^ before the letter.  For
  example:
    TYPE Testing^B^B^B go^E
  will type the word "Testing", move the cursor left three
  spaces, then type " go" and move the cursor to the end of the
  line,  leaving "Test going" on the input line.  This will
  probably happen too fast to see.
```

1.74 Shows the user logged into a server

```
Usage: USERS <server>
  Shows the users logged into the machine where the specified
  server is running.  If no server is given, the current server
  is used.
```

1.75 Shows the IRC II version number of client and server

```
Usage: VERSION [server]
  Shows the IRCII version number and the version number of the
  server specifed.  If no server name is specifed, the version
  of your current server is shown.
```

1.76 Wait for something

```
Usage: WAIT [%process]
  There are two forms of this command.  The first is:
    /WAIT
  with no arguments.  This form waits for all server output to finish
  before continuing.  This is useful if you have an alias in which you
  change a variable that you wish to change back after completion of
  the command, or if you wish to insure that your operator privs are
  activated in your .ircrc before continuing: For example, in your
  .ircrc:
    /OPER <nick> <password>
    /WAIT
    /TRACE
  The /WAIT insures that the server has responded to the /OPER before
  continuing.  If you didn't /WAIT, you would get an Unknown command
  error from the /TRACE.

  The second form of the command is:
    /WAIT %process
  Where %process is a valid running processes id or name.  This is
  used to cause IRCII to wait for an EXECd process to finish before
  continuing.  If the given process doesn't exist, return is
  immediate. Here is an example of it's use:
    /on ^exec uptime /assign uptime $1
    /on ^exec_exit uptime
    /alias uptime /exec -name uptime uptime\n/wait %uptime
  Now, whenever you do /uptime, this runs the uptime program and
  ASSIGNs the uptime variable to the time of day (that's $1 in the
  return from uptime).  This can be used for things like:
    /alias dotime /uptime\nThe time is: ${uptime}
  which will send "The time is: 10:42pm" to your channel (with the
  correct time of course).
```

1.77 Sends a message to all users on IRC

```
Usage: WALL <text of message>
  Sends a message to all users on irc.
```

1.78 Sends a message to all operators currently on IRC

```
Usage: WALLOPS <message>
  Sends a message to all operators currently on irc.
```

1.79 A While-command

```
                Usage: WHILE boolean cmd
  This will execute the given cmd while the given boolean returns
  true.  The boolean has the same format as for the
               IF
               command, except
  that it is re-evaluated at each loop interation.  The same is true
  for cmd.  It is re-evaluated at each loop interation.  This mean
  that if you have a $ expression that you wish to test at each loop
  interation you must put two $.  This is because the first $ will be
  consumed when the command is initially parsed, and the second will
  be done at each loop interation.  For example:
    /alias repeat /^assign blah 0\n/while "$${blah}<$0" "$^"^$1-\\n/^assign blah ↩
        $${blah+1}"\n/^assign -blah
  This can be used as follows:
    /repeat 10 /msg bigcheese This repeats 10 times
  The repeat alias is broken down into three parts (between \n’s):
    /^assign blah 0
    /while "$${blah}<$0" "$^"^$1-\\n/^assign blah $${blah+1}"
    /^assign -blah
  The first is to initialize blah to 0 and the last part is to remove
  blah when done.  The WHILE portion is described below:
  The boolean for the while loop is "$${blah}<$0".  When the
  alias is first parsed, this is becomes "${blah}<10" using the above
  example.  This is what is then used at each loop interation.  At the
  first iteration this will be "0<10",  at the next "1<10", and so on.
  The cmd part of this looks like "$^"^$1-\\n/^assign blah $${blah+1}"
  When the alias is first parsed, this becomes
      "/msg bigcheese This repeats 10 times\n/^assign blah ${blah+1}"
  This is executed at each loop interation.  It sends the MSG to
  BigCheese and then it increments blah by 1.

  I hope this clears things up for you.  It gets a bit confusing
  knowing when to put two $ and when not to.  Just remember that each
  alias is parsed once through before the sub-command parts are
  evaluated.  The WHILE command then uses that and evaluates again the
  boolean and cmd portions of the WHILE.
```

1.80 Gives a listing of the users on IRC

```
Usage: WHO [-switch [arg]] [wildcard expression]
  Gives a listing of the users on irc.  If no flags are
  specified, the wildcard expression you supply will
  match any field returned by WHO.  However, if the wildcard
  expression is "*" (an asterisk), the users on the current
  channel are shown.  If the expression is 0, all users on irc are
  shown.  The following switches are also recognized and used
  to limit the listing:

    -OPERATORS            - lists only operators
    -CHOPS      - lists only channel operators
    -NAME <username>      - lists only people with matching
                            user-ids
    -FILE <filename>      - lists only people listed in the
          file

  The switches may be abbreviated unambiguously.  All matching of
  usernames, hostnames, and servernames may contain wildcards (*).
  Wildcards in the form "*.edu" are also correctly recognized
  by the IRC server (so there’s no need for ircII to do that).
```

1.81 Shows more informations about a user

```
              Usage: WHOIS <nickname>
       WHOIS <nick1>,<nick2> [...]
       WHOIS <server> <nickname>[,<nick2>...]

  Shows more detailed information about the nickname specified.
  This info includes the users name, host, server, "real" name,
  and away message.  As you can see in the second line you can
  use /whois on multiple nicknames. The third line demonstrates
  how you can send the /whois request to a distant server. If
  you /whois a person on his server you will receive extra info
  about his idle time.

  See Also:

               WHOWAS
```

1.82 Shows information about who userd the given nickname last

```
Usage: WHOWAS <nickname>
  Shows information about who used the given nickname last, even
  if no one is currently using it.
```

1.83 Manipulate/open/close windows in IRC II

```
              Usage: WINDOW [<argument> [<argument> <argument> ...]]
  The WINDOW command lets you manipulate multiple "windows" in
  IRCII.  Windows are horizontally divided sections of the
  screen in which different bits of irc information can be
  displayed.  WINDOW lets you create, manipulate, and remove
  such windows.  The <argument> parameter to WINDOW can be one
  of the following:
    NEW            Creates a new window by splitting the current
                   window evenly.
    KILL           Removes the current window.  This does not
                   remove you from any channel you were on in that
                   window.
    GOTO n         Set the current window to the n’th window visible
                   on the screen.  They are numbered from the 1 being
                   the topmost window.
    GROW <n>       Increases the size of the current window by n
                   lines.
    SHRINK <n>     Decreases the size of the current window by n
                   lines.
    CHANNEL <c>    Directs all conversation with channel <c> to the
                   current window.  If you are not current on
                   channel <c>, you will join it.  If you are on
                   channel <c> it will be diverted to the current
                   window, possibly removing channel <c> from
                   another window.
    BALANCE        Causes all windows to resize so they are as
                   close as possible to evenly sized.
```

```
SCROLL <f>      Where <f> is ON, OFF, or TOGGLE.  This controls
                scrolling for the current window.
HOLD_MODE <f>   Where <f> is ON, OFF, or TOGGLE.  This
                controls hold mode for the current window.
LEVEL <L>       Where <L> is a comma separated list of one of
                the following: CRAP, PUBLIC, MSG, NOTICE,
                WALL, WALLOP, NOTE, ALL, or NONE.  This controls
                what types of messages will go to the current
                window.
LOG <f>         Where <f> is ON, OFF, or TOGGLE.  This turns on
                logging for the current window only.  The logfile
                name is created from the current setting of the
                LOGFILE variable and the name of the window channel,
                query user, or simply the word "Window" with the
                windows internal reference number aadded to the end.
                For example, if LOGFILE is set to "IrcLog", and you
                are on channel "+Har", the log file for the window
                will be "IrcLog.+Har".
NAME s          Sets the "name" for the window to s.  This name may
                now be used in place of the refnum assigned by IRCII
PROMPT s        Sets the input prompt for the window.  It will replace
                $T as the window target when used in the INPUT_PROMPT
                variable.  The window prompt is set automatically when
                EXEC detects a prompt from a process.
NOTIFY          Toggles notification for the current window when it is
                invisible. If set to ON, the first text written to the
                window after it is hidden will cause a line to be
                displayed in the current window at that time.
BACK            Set the current window to the previous current window
MOVE n          Moves the current window n positions on the screen.
                MOVE 1 will move the current window down 1 on the screen.
         Window proportions are maintained using MOVE
REFNUM r      Change the current window to the window with refnum r
HIDE     Make the current window "hidden".  It still exists in
   every way except it takes up no space on the screen.
SHOW r     Makes the hidden window with refnum r visible again and make
   it the current window.  If the window is already visible,
   just make it the current window.
SWAP r          Swaps the current window with the hidden window with refnum r.
   The proportions of the window are maintained.
ADD n         Adds n to window namelist, where n is either a nick or
   channel name.  All communcation with n will go to that window.
   Also, n may be a comma separated list (no spaces).
LIST     Show a list of all windows.
PUSH     Push the current window onto the window stack.
POP      Pop the top window off the window stack making it the current
   window.  If it is hidden, the make it visible first.
STACK    Show all the windows in the window stack.
HIDE_OTHERS     This hides all visible windows but the current window.
KILL_OTHERS     This kills all visible windows but the current window.
LASTLOG_LEVEL <L>   Where <L> is the same as for LEVEL, this
                    controls what types of messages displayed
                    in the current window should be saved to
                    the windows lastlog.
SERVER <s:n:p> This allows you to specify a different server to
                connect to for the current window.  The <s:n:p>
                refers to the servername, portnumber, and password.
```

The portnum and password are optional. If you are already connected to the specified server, the window will switch to that server. Doing a /WINDOW with no parameters shows you what server is associated with it. See HELP SERVER for more about this.

The <L> parameters for LEVEL and LASTLOG_LEVEL are described more fully in SET LASTLOG_LEVEL.

The settings of the system variables
SCROLL
and
HOLD_MODE
are
used as defaults for new windows created. Executing a SET SCROLL or SET HOLD_MODE will change the settings for the current window as well.

The "current window" is denoted by "^^^^^^^^^^" in the status line for that window. You can switch the current window by activating the key functions
PREVIOUS_WINDOW
or
NEXT_WINDOW
.
These are bound, by default, to ^Xp and ^Xn, respectively.

Each window can have the conversation of a channel going to it, which can be specified by either using WINDOW CHANNEL <c> or by executing the
JOIN
or
CHANNEL
commands in a window.
Each window can also have it's own
QUERY
user. When QUERYing
someone in a window, all conversation with that person is directed to that window.

No information is ever lost. Anything that doesn't otherwise have a specific window to go to (because it is a window's current channel or QUERY user) will either go to the current window, or the window who's LEVEL setting contains an appropriate type for that message. By default, the first window created has a LEVEL of ALL, meaning that all messages will go to that window unless otherwise specified. This includes all output from NAMES, LIST, WHO, etc. These things fall under the CRAP category for LEVEL. If you want, for example, these messages to go to your current window, you can do this in the original window:

```
  WINDOW LEVEL ALL,-CRAP
```

Now, all "crap" will go to your current window, while all private messages, walls, wallops, mail, and notices will go to the root window.

1.84 Output a given text

```
              Usage: XECHO [flags] <text>
  XECHO is like
                ECHO
                , except that it takes flag arguments.

Flags:
  -LEVEL <level>  displays the message with the specified lastlog
                  level. This is useful to chane the window some
                  messages are displayed in. This is the only way
                  to generate a message with the USERLOG1-4 lastlog
                  levels.
  See Also:

                ECHO

                ON

                SET DISPLAY
```

1.85 Help on ALIAS-Commands

```
                                 IRC II - Help.guide  ALIAS-Sub-Menu
                     *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*


                ALIAS

                FUNCTIONS

                QUOTE

                SPECIAL

                WIDTH
```

1.86 Help on BIND-commands

```
                                 IRC II - Help.guide   BIND-Sub-Menu
                     *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*


                BIND

                KEYS

                EXAMPLES

                BACKSPACE

                BACKWARD_CHARACTER
```

```
BACKWARD_HISTORY

BACKWARD_WORD

BEGINNING_OF_LINE

CLEAR_SCREEN

COMMAND_COMPLETION

DELETE_CHARACTER

DELETE_NEXT_WORD

DELETE_PREVIOUS_WORD

END_OF_LINE

ERASE_LINE

ERASE_TO_END_OF_LINE

FORWARD_CHARACTER

FORWARD_HISTORY

FORWARD_WORD

HELP_CHARACTER

META1_CHARACTER

META2_CHARACTER

NEXT_WINDOW

NOTHING

PARSE_COMMAND

PREVIOUS_WINDOW

QUIT_IRC

QUOTE_CHARACTER

REFRESH_SCREEN

SELF_INSERT

SEND_LINE

STOP_IRC

SWITCH_CHANNEL
```

```
TOGGLE_INSERT_MODE

TOGGLE_STOP_SCREEN

TRANSPOSE_CHARACTER

TYPE_TEXT

UNSTOP_ALL_WINDOWS

YANK_FROM_CUTBUFFER
```

1.87 Help on CTCP-commands

```
                     IRC II - Help.guide   CTCP-Sub-Menu
     *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*


CLIENTINFO

CTCP
```

1.88 Help on DCC-commands

```
                     IRC II - Help.guide    DCC-Sub-Menu
     *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*


CHAT

CLOSE

DCC

GET

LIST

RENAME

SEND

TALK

TMSG
```

1.89 Help on IRC II in general

```
                    IRC II - Help.guide  IRCII-Sub-Menu
     *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*


ARGUMENTS

COPYRIGHT

ENV-VARS

IRC II

PROGRAMMING

SERVERLIST

STATUSLINE

About this IRC.guide - File
```

1.90 Help on LOAD-commands and files

```
                    IRC II - Help.guide   LOAD-Sub-Menu
     *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*


ACTION

LALIAS

BASICAL

BRC

COLUMNS

COMMANDER

CURSOR

DEUTSCH

ENGLISH

EVENTS

FINGER

FNET

IRCRC

LOAD
```

LOCAL

MSG

MUDLIKE

OPER

PREFIX

REPEAT

SERVICE

SILENT

SUGGESTS

WINDOW

1.91 Help on NOTE-commands

```
                     IRC II - Help.guide   NOTE-Sub-Menu
     *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*
```

CAT

CHANNEL

COUNT

DENY

FIND

FLAG

LOG

LS

NOTE

PLOG

RM

SAVE

SEND

SENT

```
SPY

STATS

USER

WAITFOR

WALL

WALLOPS

WHOWAS
```

1.92 Help on ON-commands

```
                  IRC II - Help.guide     ON-Sub-Menu
     *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*


ON

ON: DETAILS

CHANNEL_NICK

CHANNEL_SIGNOFF

CONNECT

CTCP

CTCP_REPLY

DCC_CHAT

DISCONNECT

EXEC

EXEC_ERRORS

EXEC_EXIT

EXEC_PROMPT

FLOOD

HELP

IDLE

INPUT

INVITE
```

```
JOIN

LEAVE

LIST

MAIL

MODE

MSG

MSG_GROUP

NAMES

NICKNAME

NOTE

NOTICE

NOTIFY_SIGNOFF

NOTIFY_SIGNON

NUMERIC

PUBLIC

PUBLIC_MSG

PUBLIC_NOTICE

PUBLIC_OTHER

RAW_IRC

SEND_DCC_CHAT

SEND_MSG

SEND_NOTICE

SEND_PUBLIC

SEND_TALK

SERVER_NOTICE

SIGNOFF

TALK

TIMER
```

```
TOPIC

WALL

WALLOP

WHO

WINDOW

WINDOW_KILL
```

1.93 Help on SET-commands

```
                     IRC II - Help.guide    SET-Sub-Menu
     *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*

SET

ALWAYS_SPLIT_BIGGEST

AUTO_UNMARK_AWAY

AUTO_WHOWAS

BEEP

BEEP_MAX

BEEP_ON_MSG

BEEP_WHEN_AWAY

CHANNEL_NAME_WIDTH

CLOCK

CLOCK_24HOUR

CLOCK_ALARM

CMDCHARS

COMMAND_MODE

CONTINUED_LINE

DISPLAY

ENCRYPT_PROGRAM

EXEC_PROTECTION

FLOOD_AFTER
```

```
FLOOD_RATE

FLOOD_USERS

FLOOD_WARNING

FLOOD_STATUS_LINE

HELP_PAGER

HELP_WINDOW

HIDE_PRIVATE_CHANNELS

HISTORY

HISTORY_FILE

HOLD_MODE

HOLD_MODE_MAX

INDENT

INPUT_ALIASES

INPUT_PROMPT

INSERT_MODE

INVERSE_VIDEO

LASTLOG

LASTLOG_LEVEL

LOG

LOGFILE

MAIL

MINIMUM_SERVER

MINIMUM_USERS

NOTIFY_ON_TERMINATION

NOVICE

PAUSE_AFTER_MOTD

SCROLL

SCROLL_LINES
  SEND_IGNORE_MSG
SHELL
```

```
SHELL_FLAGS

SHELL_LIMIT

SHOW_CHANNEL_NAMES

SHOW_END_OF_MSGS

SHOW_NUMERICS

STATUS_AWAY

STATUS_CHANNEL

STATUS_CLOCK

STATUS_DQUERY

STATUS_FORMAT

STATUS_HOLD

STATUS_INSERT

STATUS_MAIL

STATUS_MODE

STATUS_NOTIFY

STATUS_OPER

STATUS_OVERWRITE
  STATUS_QUERY

STATUS_SERVER

STATUS_USER

STATUS_WINDOW

SUPPRESS_SERVER_MOTD

WARN_OF_IGNORES
```

1.94 Sets an alias

```
              Usage: ALIAS [-][alias] [commands]
Creates a command alias for [commands].  Commands do not need to be
prefixed with a command character, in fact they shouldn't!
If you want an alias to type something to the channel or query
use SEND or SAY. You can use ';' to put multiple commands in a row.

Special character sequences, usually ircII variables, are prefixed
```

```
  with '$'.  There are certain $ sequences built in to IRCII for getting
  such information as your nickname and current channel, plus you can
  expand your own variables and system variables using the $ format.
  You can also expand functions, substituting their return values.
  See ASSIGN for more on this.

  Here are some simple alias examples:
    /ALIAS HELLO MSG $0 Hello, $0! $1-
  This alias can be used by typing:
    /HELLO BigCheese How are you?
  This would act as though you had typed:
    /MSG BigCheese Hello, BigCheese! How are you?
  ALIAS with just the [alias] argument shows the current alias
  for that word.  ALIAS with no argument shows all current
  aliases.  To remove an alias, use ALIAS -[alias].

  To use more than one command in an alias, seperate them by \n.

  For more specific information about the $ uses and more examples,
  see the specific help files listed below.

  See Also:

                 SET INPUT_ALIASES

                 SEND

                 SAY

                 IRCIIPROGRAMMING

                 ASSIGN
```

1.95 Alias/IRC II-functions

```
IRCII Functions
  IRCII functions are substituted with the format $%FUNCTION(arguments).
  A function is an ALIAS which assigns a value to FUNCTION_RETURN.
  For example:
        /ALIAS DOUBLE assign FUNCTION_RETURN $({$0+$0})
        /ALIAS SHOWDOUBLE echo $%DOUBLE($0)
  will cause
        /SHOWDOUBLE 7
  to display "14".

  The following built-in functions already exist and cannot be overriden:
  LEFT(COUNT STRING)     Returns the COUNT leftmost bytes from the STRING.
  RIGHT(COUNT STRING)    Returns the COUNT rightmmost bytes from the STRING.
  MID(INDEX COUNT STRING) Returns COUNT bytes starting at position INDEX
                         in STRING.
  INDEX(CHARLIST STRING) Returns the index to the first character in STRING
                         which appears in CHARLIST.
  RINDEX(CHARLIST STRING) Returns the index to the last character in STRING
      which appears in CHARLIST.
  TIME()                 Returns the current system time as a long integer
  STIME(TIMEVAL)         Returns the date and time in English that corresponds
                         to the long integer TIMEVAL.
  RAND(LIMIT)            Returns a random number x such that 0<=x<LIMIT
  SRAND(SEED)            Seeds the random number generator and returns nothing.
                         The seed may be a long integer, although only the low
                         integer is used.

NOTE The argument separator for functions is the space character, not the
     comma.
```

1.96 Quoting modifier

```
              ALIAS MODIFIER:
  This is a special quoting modifier.  Use of this modifier tells
  IRCII that you wish certain characters in the converted text to be
  quoted using the \ character.  The form of this modifier is:
    $^c<sequence>
  where c is the character to be quoted, and <sequence> is one of the
  special sequences.  For example, you may wish to do the following:
    /alias foo /if 1 "/echo $^"*"
  and you do:
    /foo "This is a Test"
  Since $ expressions are parsed first, the $^"* is the first thing
  parsed.  The ^" part tells IRCII to quote and " in the resultant
  string.  So, the $^"* is expanded just like $* and becomes:
    "This is a Test"
  and the ^" causes it to quote all ", which becomes:
    \"This is a Test\"
  This is then replaced in the original alias for parsing, so /foo
  becomes:
    /if 1 "/echo \"This is a test\""
  Notice that the \" won't interfere with the parsing of the " in the
  IF statement.
  The ^ modifier must be first after the $.  You may specify more than
  one character to be quoted simply be adding more ^c to the
  expression:
    $^"^'B
  will quote all occurences of " and ' in the body of the last message
  you received.

  Remember, you can use both forms of modifiers, but they must be in
  the correct order.  All ^ modifiers must be first, followed by any
  [] modifier.  For example:
    $^.[-10]S
  This will right justify your server name and quote any . in the
  server name with \.

  See Also:

               ALIASWIDTH
```

1.97 Special character sequences for ALIAS

```
Special character sequences for ALIAS:
  All special character sequences begin with a $.  In their simpliest
  form, the following have special meaning in aliases:
    $*   Expands to the rest of the arguments on the command line.
    $n   Where n is a non-negative number, expands to the nth arg.
    $n-m Where n and m are non-negative numbers, expands to the
         nth thru mth arguments inclusive.
    $n-  Where n is a non-negative number, expands from the nth
         argument to the end of the argument list.
    $-m  Where m is a non-negative number, expands from the
         beginning of the argument list to the mth argument.
         This is the same as $0-m.
    $,   Expands to the nickname of the last person who send you
         a /MSG
    $.   Expands to the nickname of the last person to whom you
         sent a /MSG
    $variable  Expands to the value of one of the following:
           1) Matching ASSIGN'd variable
           2) Matching IRCII SET variable
           3) Matching environment variable
           4) Nothing
         It checks in the order shown.  Thus, if 1 doesn't match, 2 is
         tried.  If 2 doesn't match, 3 is tried, etc.
         See ASSIGN for more details.
    $"Prompt"  Will prompt you (using the text between the double
               quotes) for input which will be replaced in the
               alias.
    $(sub-alias)  This expands out the sub-aliases, then uses that
         result in place of the (sub-alias) expression.  For example
         $($0) will first expand $0... suppose it expands to S.  Then
         it replaces that in the original text, giving you $S, which
         is then expanded to the name of your current server.
    $!history!  This expands to a matching entry in your command
         history.  The text between the ! may contain wildcards.
    $:   Expands to the nickname of the last person to join your
         channel
    $;   Expands to the nickname of the last person to send a public
         message to your channel
    $A   Expands to the text of your AWAY message
    $B   Expands to the body of the last MSG you sent.
    $S   Expands to the name of your server
    $Q   Expands to the nickname of the person you are QUERYing.
    $C   Expands to your current channel
    $T   Expands to the 'target' of your input (either a QUERY nick or
         a current channel)
    $N   Expands to your nickname
    $K   Expands to the current value of CMDCHAR.  Useful to have
         aliases work even when you change CMDCHAR.
    $I   Expands to the name of the channel to which you were last
         INVITED
    $S   Expands to the name of the server you are on
    $L   Expands to the current contents of the input line
    $Z   Expands to the time of day
    $A   Expands to your current AWAY message
```

```
*    $O   Expands to the value of STATUS_OPER if you are currently an operator
    $P   Expands to "@" if you are a chanop on the current channel
    $D   Expands to the nickname of the person whose sign-on was last
         detected by the NOTIFY mechanism.
    $$   Expands to $

  Argument to aliases will automatically be appended to the expanded
  alias unless you use one of the following forms in the alias:
     $*
     $n
     $n-m
     $-m
     $n-
     $(sub-alias)
  If one of these forms is used in the alias arguments are not appended.
  For example:
    /ALIAS M /MSG
  will be treated as:
    /ALIAS M /MSG $*
  However,
    /ALIAS M /MSG $0 $1-
  will not have the arguments appended.  If you have an alias and you wish to
  prevent arguments from being appended, add $() to the alias.  The $() with
  nothing between the parenthesis expands to nothing and prevents arguments
  from being appended.

  Any alias may be surrounded by {}s so that it can be imbedded within
  another string.  For example:
    a${N}a
  will expand to (assuming your nickname is BigCheese):
    aBigCheesea
  Aliases are automatically delimited by certain characters.  For example:
    "$N"
  expands to:
    "BigCheese"
```

1.98 Width-modifier

```
              ALIAS MODIFIER:
  This modifier is a width specifier for any of the forms of
  alias.  This is done by placing [<width>] after the $ (and
  after and ^c expressions).  For example:
    $[10]C
  This expand to your current channel, but it will truncate any
  characters beyond 10 in the channel name.  If the channel is
  less than 10 characters, it will be padded on the right with
  blanks.  Specifying a negative width will justify the field on
  the right and pad on the left with blanks.  A width of 0 is
  treated as though no width specifier was used.  If you get real
  tricky, you can do things like this:
    $([$CHANNEL_NAME_WIDTHD]C)
  The value of the IRCII variable CHANNEL_NAME_WIDTH will be expanded
  in the $() expression.   If CHANNEL_NAME_WIDTH is 10, this will result
  in $[10]C which will then expand as described above.
  Remember, you can use both forms of modifiers, but they must be in
  the correct order.  All ^ modifiers must be first, followed by any
  [] modifier.  For example:
    $^.[-10]S
  This will right justify your server name and quote any . in the
  server name with \.

  See Also:

               ALIAS QUOTE
```

1.99 Bind a keystroke (-sequence) to an IRC function

```
              Usage: BIND [key] [function] [string]
  Binds a keystroke sequence to an IRC function.  The
  QUOTE_CHARACTER key (by default ^Q) may be used to override any
  key binding and have IRC insert the actual value of the key in
  the input line.

  See Also:

               BIND KEYS
                             for information on allowable [key] parameters

               BIND EXAMPLES
                         for some examples.

               BIND-Functions Overview
               for help on individual key binding functions
```

1.100 Some examples about BIND

```
The following are allowable key sequences for use with the BIND
function:
  c          where c is any key (case sensitive)
  ^c         where c is one of A thru Z or ? or [ or ]
  METAn-c    where n is 1 or 2 and c is the same as the
             first form above
  METAn-^c   where n is 1 or 2 and c is the same as the
             second form above
  mc         where m has been previously bound to META1_CHARACTER
             or META2_CHARACTER and c is the same as the first
             form above.  m may be either of the first two forms
             above (m or ^m).
  m^c        where m has been previously bound to META1_CHARACTER
             or META2_CHARACTER and c is the same as the second
             form above.  m may be either of the first two forms
             above (m or ^m).
```

1.101 Backspace

```
Usage: BIND <key> BACKSPACE
  The BACKSPACE function deletes the character to the left of the
  cursor and moves the cursor one space to the left.  If the
  cursor is at the first position in the input line, BACKSPACE
  has no effect.
```

1.102 Cursor backward

```
Usage: BIND <key> BACKWARD_CHARACTER
  The BACKWARD_CHARACTER function moves the cursor one space to
  the left.  If the cursor is at the first character in the input
  buffer, this function has no effect.
```

1.103 Backward in history

```
Usage: BIND <key> BACKWARD_HISTORY
  Replaces the contents of the input buffer with the previous
  command in the command history buffer.  The command history
  buffer is a circular buffer that wraps at the end.
```

1.104 Backward one word

```
Usage: BIND <key> BACKWARD_WORD
  Moves the cursor to the first character of the previous word.
  If the cursor is in the middle of a word, the cursor is moved
  to the first character of the same word.  If the cursor is on
  the first character of a word or on whitespace, the cursor is
  moved to the first character of the previous word.
```

1.105 Beginning of line

```
Usage: BIND <key> BEGINNING_OF_LINE
  Moves the cursor to the first character in the input buffer.
  This need not be the first character visible on the screen,
  since the input buffer wraps when the line gets too long.  This
  is always the first characters in the input buffer.
```

1.106 Clear screen

```
Usage: BIND <key> CLEAR_SCREEN
  This function will clear the display and restart a screen that
  has been stopped by HOLD_MODE or by the TOGGLE_STOP_SCREEN
  function.
```

1.107 Complete a command

```
Usage: BIND <key> COMMAND_COMPLETION
  The COMMAND_COMPLETION function will cause IRCII to attempt to
  complete the command you have typed in the input line.  If a
  match is found, it is expanded to its full length in the input
  line.  If multiple matches are found, the complete list of
  matching commands is displayed.  If no match is found, the
  input line is left unchanged.
```

1.108 Delete character

```
Usage: BIND <key> DELETE_CHARACTER
  Deletes the character under the cursor.  The cursor is not
  moved by this operation.  If the cursor is at the end of the
  input buffer, this function has no effect.
```

1.109 Delete next word

```
Usage: BIND <key> DELETE_NEXT_WORD
  Deletes from the cursor position to the end of the next word.
  If the cursor is in the middle of a word, all characters from
  the cursor position to the end of the word are deleted.  If the
  cursor is on whitespace, all following whitespace up through
  the end of the next word is deleted.
```

1.110 Delete previous word

```
Usage: BIND <key> DELETE_PREVIOUS_WORD
  Deletes to the beginning of the previous word.  If the cursor
  is in the middle of a word, all characters from the left of the
  cursor to the beginning of the word.  If the cursor is on
  whitespace, all whitespace through to the beginning of the
  previous word are deleted.
```

1.111 End of line

```
Usage: BIND <key> END_OF_LINE
  Moves the cursor to the last character of the input buffer.
```

1.112 Erase line

```
Usage: BIND <key> ERASE_LINE
  Erases the contents of the input buffer, leaving it very empty.
```

1.113 Erase to end of line

```
Usage: BIND <key> ERASE_TO_END_OF_LINE
  Erases from the cursor position to the end of the input buffer.
```

1.114 Cursor forward

```
Usage: BIND <key> FORWARD_CHARACTER
  Moves the cursor to the right one position.  If the cursor is
  at the end of the input buffer, this function has no effect.
```

1.115 Forward in history

```
Usage: BIND <key> FORWARD_HISTORY
  Replaces the contents of the input buffer with the next entry
  in the command history buffer.  The command history buffer is
  circular and wraps at the end.
```

1.116 Forward one word

```
Usage: BIND <key> FORWARD_WORD
  Moves the cursor to the end of the next word.  If the cursor is
  in the middle of a word, it is moved to the end of that word.
  If the cursor is already at the end of a word or on whitespace,
  it is moved to the end of the next word.
```

1.117 Help on currently entered command

```
Usage: BIND <key> HELP_CHARACTER
  This shows help on the current input buffer without disturbing
  the contents of the buffer.  It is as though you had typed
  /HELP followed by whatever is in the input buffer.
```

1.118 Set META1-character

```
Usage: BIND <key> META1_CHARACTER
  This function switches to an "alternate" keymap.  That is,
  after this character is hit, the next key hit can have a new
  definition.  This is how the Escape sequences are done.  In
  fact, the escape key’s default binding is META1_CHARACTER.  For
  example:
    BIND ^X META1_CHARACTER
  binds this function to ^X.  You may then bind meta key
  sequences with either of the two following formats:
    BIND ^X? HELP_CHARACTER
  or
    BIND META1-? HELP_CHARACTER
```

1.119 Set META2-character

```
Usage: BIND <key> META2_CHARACTER
  This function switches to an "alternate" keymap.  That is,
  after this character is hit, the next key hit can have a new
  definition.  This is how the Escape sequences are done.  In
  fact, the escape key's default binding is META1_CHARACTER.  For
  example:
    BIND ^X META2_CHARACTER
  binds this function to ^X.  You may then bind meta key
  sequences with either of the two following formats:
    BIND ^X? HELP_CHARACTER
  or
    BIND META2-? HELP_CHARACTER
```

1.120 Go to next window

```
Usage: BIND <key> NEXT_WINDOW
  Switches the "current window" to be the next window in the
  window list.  This is the window "below" the current window on
  the screen.  At the bottom of the screen, this will jump to the
  top most window on the screen.
```

1.121 Do nothing

```
Usage: BIND <key> NOTHING
  The NOTHING function does nothing.  Effectively, it disables
  the key it is bound to (to which it is bound for you
  grammatical types).
```

1.122 Execute an command

```
                Usage: BIND <key> PARSE_COMMAND <string>
  The PARSE_COMMAND function cause the supplied string to be
  executed as an ircII command (or alias). It doesn't need to be
  prefixed by the command character. This function does not disturb
  the contents of the input line and is not added to the command history.
  You may also include any of the special $ sequences available in ALIAS
  and they will be expanded before the line is parsed.  The
  sequences that deal with command line arguments ($*, $n, $-n)
  are expanded as though there were no command line arguments.

  See Also:
```

```
                 ALIAS

                 IRCII PROGRAMMING
```

1.123 Go to previous window

```
Usage: BIND <key> PREVIOUS_WINDOW
  Switches the "current window" to be the previous window in the
  window list.  This is the window "above" the current window on
  the screen.  At the top of the screen, this will jump to the
  bottom most window on the screen.
```

1.124 Exit IRC

```
Usage: BIND <key> QUIT_IRC
  Exit irc and return to whatever excuse for a shell you're
  using.  Sorry.  Didn't mean that.
```

1.125 Quote character

```
Usage: BIND <key> QUOTE_CHARACTER
  This function "quotes" the next key hit.  What this really
  means is that it overrides the key binding for the next key and
  forces it to insert itself into the input buffer.  For example,
  if you have Control D bound to delete character, and you hit
  the quote character (defaults to Control Q) then Control D, it
  will insert a Control D (shows up as an inverse video D) and
  does not delete the character.  Got it?
```

1.126 Refresh screen

```
Usage: BIND <key> REFRESH_SCREEN
  This redraws the screen, in case some other nasty program has
  munged it up.
```

1.127 Insert a character

```
Usage: BIND <key> SELF_INSERT
  Causes the key it is bound to to insert itself into the input
  buffer.  This is the default binding for the alphabet keys,
  numbers, etc.  If this is bound to a meta sequence, it will
  only insert the last key hit in the meta sequence.
```

1.128 Send line

```
Usage: BIND <key> SEND_LINE
  This "sends" the input buffer to the server.  Well, really it
  first expands aliases, etc...  but you get the idea.  It is
  normally bound to the return key or enter key...  or both.
```

1.129 Stop IRC (go to shell)

```
Usage: BIND <key> STOP_IRC
  Sends a Stop signal to IRC II, returning you to the shell.
  Normally, you can then return to IRC II by typing fg.  Please
  read about your shell to find out about this.
```

1.130 Switch channel(s)

```
Usage: BIND <key> SWITCH_CHANNELS
  Changes the current channel for the current window.  This will
  only switch to a channel not currently directed to another
  window.  That is, if you are on three channels, #Foo, #Bar,
  and #Cheese, and #Foo and #Bar are currently directed to two
  windows but #Cheese isn’t, then SWITCH_CHANNELS will only
  toggle between #Cheese and either #Foo or #Bar (depending upon
  which window this is done in)
```

1.131 Toggle insert-mode

```
Usage: BIND <key> TOGGLE_INSERT_MODE
  Toggles the INSERT_MODE variable.  This function is equivalent
  to doing:
    /SET INSERT_MODE TOGGLE
```

1.132 Toggle stop screen

```
Usage: BIND <key> TOGGLE_STOP_SCREEN
  Stops and starts the screen display if it is scrolling madly
  out of your control.  The display is also restarted after the
  SEND_LINE function is executed.
```

1.133 Swaps two characters

```
Usage: BIND <key> TRANSPOSE_CHARACTERS
  Swaps the two characters before the cursor.  Thats all.
```

1.134 Type text into input buffer

```
Usage: BIND <key> TYPE_TEXT <string>
  Causes the given string to be typed into the input buffer using
  the current setting of the INSERT_MODE variable.
```

1.135 Unstop all windows

```
Usage: BIND <key> UNSTOP_ALL_WINDOWS
  When activated, all currently stopped windows will be restarted.
```

1.136 Restores the last deleted thing

```
Usage: BIND <key> YANK_FROM_CUTBUFFER
  Restores the last deleted thing.  The following functions store
  what they deleted to the cutbuffer:
    DELETE_NEXT_WORD
    DELETE_PREVIOUS_WORD
    ERASE_LINE
    ERASE_TO_END_OF_LINE
```

1.137 Examples of BIND

```
Here are some examples of the BIND function:
    /BIND ^X META2_CHARACTER
  Makes control X a meta key.
    /BIND ^Xl parse LIST
  Binds the meta sequence control X followed by l to perform a
  /LIST.  This binding is only valid if ^X was previously bound
  to a meta function (as was done in the first example).
    /BIND META1-u BACKWARD_HISTORY
  by default, the escape key is bound to META1_CHARACTER, so this
  binding makes the escape u key sequence show the last command
  history entry.  Note that more than one key may be bound to
  *any* function.
    /BIND ! type Bang!
  This binds the !  (exclamation mark) to type the word "Bang!"
  into the input line whenever it is hit.
```

1.138 Shows you the known CTCP commands from a client

```
Usage: /ctcp <nick> CLIENTINFO
                    CLIENTINFO <CTCP_COMMAND>

CLIENTINFO returns the known CTCP commands from the other client in
a list.
```

```
You can inspect commands further by calling 'CLIENTINFO VERSION' for
example. You will (or should) be given a one-line explanation of
what this command is supposed to do.

To find out about your own's client services execute a CLIENTINFO
on yourself.

Warning: Some CTCP commands are not supposed to be sent by hand,
    in particular ERRMSG, DCC, SED and ACTION.
```

1.139 Client-to-client-protocol

```
              Usage: CTCP <nickname> <command> [args]
  CTCP allows you access to the client-to-client protocol used
  to perform certain client specific actions between different
  clients on the network.  The CTCP mechanism works by sending
  a specially coded message to another user whose client is
  supposed to reply with a reply message of the type, or with
  an error message, unless you sent it to a channel.

  For example, if you do:
  /CTCP BigCheese VERSION
  you will receive:
  *** CTCP REPLY VERSION from BigCheese: IRCII 2.1 Unix

  The <command> field may be of several types, new ones are
  introduced all the time, but there is a mechanism for you to
  find out what you can use: see CLIENTINFO.

  See Also:

                 CTCPCLIENTINFO

                 ON CTCP

                 ON CTCP_REPLY
```

1.140 Initiate a DCC chat-connection

```
              Usage: DCC CHAT NICK
   DCC CHAT initiates a direct client connection chat to the given
   nick, who must respond with DCC CHAT. This is the most secure
   form of communication available on IRC. Messages sent via a DCC
   CHAT connection are not sent through IRC, but are sent by a direct
   connection between your client and the remote client. Messages
   are sent over a DCC CHAT connection with DMSG
  See Also:

                 DMSG
```

1.141 Close a DCC connection

```
              Usage: DCC CLOSE type nick [arguments]
   DCC CLOSE closes an unwanted DCC connection or offer. The type,
   nick and arguments are the same as those shown by DCC LIST.
   If the arguments are not specified, the oldest connection of
   the specified type is closed.
  See Also:

               DCC LIST

               DCC RENAME
```

1.142 Direct client client connection

```
              Usage: DCC FUNCTION [ARGUMENTS]
   DCC handles direct connections to remote clients. The behaviour of DCC
   is determined by the FUNCTION specified.
  See Also:

               DMSG
```

1.143 Transfer a file

```
              Usage: DCC GET nick filename
   DCC GET accepts a file transfer by direct client connection.
   The sender must first have offered the file with DCC SEND.
  See Also:

               DCC RENAME

               DCC SEND
```

1.144 List DCC connections

```
              Usage: DCC LIST
   DCC LIST lists all the current DCC connections showing the type
   of connection, the nick of the person on the other end of the
   connection, the current state of that connection and any other
   information associated with that connection.
  See Also:

               DCC CLOSE
```

1.145 Rename a DCC-Get

```
             Usage: DCC RENAME nick [filename1] filename2
  DCC RENAME renames a file prior to a DCC GET from filename1 to
  filename2. If filename1 is not specified, the oldest file
  connection to the given nick is renamed.
 See Also:

              DCC GET

              DCC LIST
```

1.146 Transfer a file

```
             Usage: DCC SEND nick filename
  DCC SEND initiates a file transfer by direct client connection.
  The recipient must accept your offer of a file transfer with
  DCC GET.
 See Also:

              DCC GET
```

1.147 DCC-talk-connection to a user

```
             Usage: DCC TALK user[@host]
  DCC TALK initiates a talk connection to the user at the named host,
  or responds to one initiated by that user if one exists. Messages
  are sent across a DCC TALK connection with DCC TMSG
 See Also:

              DCC CHAT

              DCC TMSG
```

1.148 Send a message across a DCC-talk-connection

```
             Usage: DCC TMSG user[@host]
   DCC TMSG sends a message across a talk connection to the given
   user. If only one connection with the given username exists, the
   host need not be supplied.
 See Also:

              DMSG

              MSG

              DCC TALK
```

1.149 IRC II - Arguments

```
The [switches]:
   -c <channel> joins <channel> at startup.  If not specified, you
                start at channel 0.
   -p <port>    default server connection port (usually 6667).
                This default port number can be over-riden by
                specifying a port number in the server list (see below)
                or using the SERVER command (see SERVER).
   -f           your terminal uses flow controls (^S/^Q), so IRCII shouldn't
                You may want to rebind ^Q and ^S so you can still
                use those functions.  ^Q is usually bound to
                QUOTE_CHARACTER, and ^S is usually bound to
                TOGGLE_STOP_SCREEN.  See BIND about this.
   -F           your terminal doesn't use flow control (default).
                Opposite of -f, forces IRCII to take over ^Q/^S.
   -s           Start IRCII without using the server process (ircserv)
                even if it can find it.  This is useful for use with
                irciid and automatons which don't need the
                separate process (since they never ^Z).
   -d           runs IRCII in "dumb" terminal mode.  IRCII then
                uses no fancy screen stuff or fancy input editting.
                All output goes to stdout, and all input is read from
                stdin.  Input editting is only done by your tty (if any).
                No IRCII keybinding have any effect in dumb mode.
   -l <file>    loads <file> in place of your .ircrc
   -a           adds default servers and command line servers to server
                list.  Normally, if you specify servers on the command
                line, the IRCSERVER and default server are not added to the
                server list.  This forces all servers known to be added.
                The order in which servers appear in you server list
                are as follows:
                   1. command line servers (from left to right)
                   2. IRCSERVER servers (from left to right)
                   3. Default IRCII servers
   -b           runs IRCII in dumb mode but accepts no user input.
                This is useful for automatons written totally under
                IRCII which you wish to run in the background.  IRCII
                will automatically fork into the background with
                this switch (unless -n is used).
   -n           Prevents IRCII from forking into the background with
                either the -b or -e switches..
   -e <process arg list>  runs IRCII by reading stdin from <process> and
                sending stdout to <process>.  All other args are passed
                to <process>,  so this must be the last switch.  IRCII
                will fork into the background unless -n is used.
```

1.150 IRC II - Copyright

```
                IRCII is Copyright (c) 1991 by Michael Sandrof. Versions 2.11
and above Copyright (c) 1991 by Troy Rollo. You have the
right to copy, compile, and maintain this software.  You also
have the right to make modifcations to this code for local use
only.  You have the right to distribute modified versions of
```

```
the code as long as you CLEARLY indicate that any problems
with the modified code be reported to you.

One module of the source code, scandir.c is copyright (c) 1983 by
the Regents of the University of California.  All rights are
reserved by the University of California.

This software is provided ``as is'' and without any express or
implied warranties, including, without limitation, the implied
warranties of merchantibility and fitness for a particular
purpose.


                 About this IRC.guide - File
```

1.151 IRC II - Environment Variables

```
The following environment variables are recognized by IRCII.

IRCNICK    -  Specifies your default irc nickname
IRCNAME    -  Specifies any lunacy you want instead of your real name
IRCSERVER -  Specifies a default server list for IRCII
IRCPATH    -  Specifies a list of directories IRCII should search when
              the /LOAD command is used to find IRCII scripts.
IRCRC      -  Specifies a file to use instead of your $HOME/.ircrc.
TERM       -  Specifies your terminal type
HOME       -  Tells IRCII where your home directory is
```

1.152 About IRC II...

```
                IRCII is a termcap based interface to the IRC Network.  All
commands to IRCII begin with a '/' (or whatever your current
CMDCHAR is set to, see HELP SET CMDCHAR).  Any other text you
type on the command line is sent directly to the channel you are
on.

IRCII may be ftp'ed from plod.cbme.unsw.oz.au [129.94.128.121]
                   or freebie.engin.umich.edu [141.212.68.23]
           or ftp.informatik.uni-oldenburg.de [134.106.1.9]

Send bug reports, comments, or questions to
   Troy Rollo (troy@plod.cbme.unsw.oz.au)
or to the IRCII mailing list
   +dist+~ms5n/src/irc/IRCII.dl@andrew.cmu.edu
or to Troy on irc.


                 About this IRC.guide - File
```

1.153 About programming in IRC II

```
                This is about programming in ircII, because ircII is not a client ↩
                   program,
it’s an operating system.. :)  And the language is just as simple as SMAIL
(that is: it is horrendous) but if you want to get into it, here’s a little
note for you:

The command character (usually ’/’) is only necessary when you type commands
interactively, when you program things it is no more needed, it used to be
though. If you want to type to the channel from within an alias, on or
binding, you have to use SAY or SEND.

The ’;’ has a special meaning in aliases, bindings and ons: It is treated
as command seperator, that means you can execute multiple commands in
a row seperated by semicolons. The semicolons are not considered seperators
when you use them interactively (to be able to type  ;-)) and within an
ircII script file. You can escape the meaning of ; in an alias with \;.

-lynX


                 About this IRC.guide - File
```

1.154 IRC II - Serverlist

```
The formats of an IRCII server list is as follows:
  servername
  servername:portnum
  servername:portnum:password
  servername::password
The servername is the name of the server, the portnum is the port
number of that server to connect to, and the password is a
connection password for that server.  The first format above
uses the default port number and no password.  The second format
specifies a port number for connection to the specified server
and no password.  The third format specifies both a port number
and password for connection.  The final format uses a password
and the default port number.

And, since we call these "server lists", you may specify as many
of the above formats as you wish.  These may be used on the IRCII
command line (See HELP IRCII COMMAND_LINE_ARGS) and in the
IRCSERVER environment variable (see HELP IRCII ENVIRONMENT_VARS).
```

1.155 IRC II - Statusline

```
                The IRCII status line shows some important pieces of information.
It shows your current nickname (with an asterisk next to it if
you have OPER privs), your current channel and QUERY user (if you
are QUERYing anyone), current Insert or Overwrite mode (I or O),
and your AWAY status.
```

```
The status line may be customized using certain IRCII variables.
Please see the following for more information on this:
  HELP
                 SET
                  HELP
                 SET STATUS_FORMAT
                  HELP
                 SET STATUS_AWAY
                  HELP
                 SET STATUS_CLOCK
                  HELP
                 SET STATUS_HOLD
                  HELP
                 SET STATUS_INSERT
                  HELP
                 SET STATUS_OVERWRITE
                  HELP  SET STATUS_QUERY
  HELP
                 SET STATUS_USER
                  HELP
                 SET STATUS_MAIL
                  HELP  SET FULL_STATUS_LINE
```

1.156 Loads a IRC II - script

```
                Usage: LOAD [-args] <filename> [arg0 arg1 ...]
  Loads the given file into irc, treating each line as an irc
  command, just as if it were typed into the command line, but it
  expects commands to not have leading command characters, for
  compatibility it is however allowed. LOAD is the command used
  automatically at irc startup to load the .ircrc file.

  In a LOAD script it is allowed to have leading spaces and tabs
  before the commands.

  A long line, like a long alias definition, can be split over
  multiple lines by putting { and } around it like this:

alias follow {
  say sorry folks, got invited away
  join -i;bow
  say here i am especially for you
  grin
}

  This format _MUST_ be used, you cannot put the { in the next
  line and you cannot put the } elsewhere than by itself.
  It is also not allowed to use { { } } constructs.

  If the -args flag is specified, then each
  line of the loaded file is alias expanded before it is executed,
  just as if you had INPUT_ALIASES set to ON.  The optional
  arguments, arg0 arg1 etc, are then used to expand the argument
  expandos ($*, $1, $2, etc) for each line.  The -args switch
  takes precedent over any changes of the setting of INPUT_ALIASES
  in the loaded file.  If you only want to expand certain lines
  and not others, use the INPUT_ALIASES variable since it too will
  used the arguments on the command line.

  Commands seperated by ';' are not recognized by LOAD, that syntax
  is for ALIAS, BIND and ON only.

  See Also:

                SET INPUT_ALIASES
```

1.157 Some aliases...

```
Usage: /LOAD action

  Several Multi-User-Dungeon-like aliases are defined here.
  They make use of the /me command, and are transmitted to
  the current channel or the current query via CTCP ACTION.
  Old client programs might not understand this kind of
  messages.

  The simple aliases (without arguments) are:
  APPLAUD   BOUNCE    BOW   BURP
  CHUCKLE   CLAP    COUGH   CRY
  GASP    GIGGLE    GRIN    GROAN
  LAUGH   MOAN    NOD   PURR
  SCREAM    SHIVER    SHRUG   SIGH
  SING    SMILE   SNAP    SNEEZE
  SNORE   WAVE    WHISTLE   WIGGLE
  WINK    YAWN

  These instead, require a nickname as argument:
  COMFORT   CUDDLE    DANCE   HUG
  KISS    LOVE    POKE    SHAKE
  SLAP    SPIT    THANK

  You can also make up your own sentences with the /ME or the
  /DESCRIBE command.  e.g. "/ME is hungry." will send
  "BigCheese is hungry." if you are the big cheese.
```

1.158 Some aliases...

```
Usage: /LOAD alias
  Aliases defined here are:
  /MA <text>  Send another message to the person you last sent one.
    Same as: /m . <text>
  /MR <text>  Send a reply to the person who last sent you a msg.
    Same as: /m , <text>
  /WA   Show the WHOIS of the person you last sent a message to.
  /IA   Invite the person you last sent a message to.
  /UNALIAS <aliasname>  An alias to unalias an alias ;-)
  /UNSET <variablename> An alias to unset a set.. huh? :)
  /ALARM <time>   Set the alarm of the clock.
  /CLOCK <on/off> Switch the clock.

  /NO = /NOTICE to avoid collision with /NOTIFY
  /LA = /LASTLOG  to avoid collision with /LAUGH from ACTION
  /LF = /SET LOGFILE  for lazy ones..
  /"  = /QUOTE    "       "

  Contributed by various ircers...
  /IGNORELAST   ignore messages and notices from the person that last sent you a ↩
     msg
  /PROTECT    make your channel invite-only and writeprotected
  /UNPROTECT    remove that state of protection again
  /CHOPS    get a list of the channel operators on the channel
  /WJOIN <channel>  join a channel and open a special window for it
  /TALKTO <target>  open a window and start a query with <target> in it
  /OOPS <right_person>  excuse yourself for a msg that went to the wrong person  ↩
     and send it
      to the person that should have got it
```

1.159 Some aliases...

```
Usage: /LOAD basical
  Basical aliases are defined here, so to make ircII compatible
  to the old irc. This script should be loaded from the global ircrc.

  However, some additional aliases are defined:

  /CHOP <nick>      Make a person channel-op of your current channel.
  /K <nick>         Kick person from current channel.
  /MO <modechange>  Make a mode change on the current channel.
  /UNCHOP <nick>    Remove channel-operator status from person.

  /WDETECTED        /Whois of the last person detected by /notify.

  /H+ and /H-       Quickies for /set hold_mode on and off.

  ^r is bound to type the 'reverse' character.

  Also some output representations are changed.
```

1.160 Some aliases...

```
Usage: /LOAD bitnet

  BITNET aliases and settings are defines here to make ircII look
  and feel like the BITNET Relay Chat (BiRC).

  Some of the BiRC commands can't be found in IRC, and some of the
  commands will display different outputs than those in BiRC.
  The display will reassemble as much of the BiRC that is possible.

  This script is version 1.20, Patchlevel 1
  This script is Copyright (C) Ove Ruben R Olsen (Gnarfer)

  The commands below are taken from the BiRC /Help command.

  **************************** Relay Commands *****************************
  * /BYE. . . . . . . . . . . . . . . . . . . . . . . . . Signoff from Relay
  * /Channel name . . . . . . . . . . . . . . . . . . Change to channel <name>
  * /COntact nick | relay . . . . . . . . . . . . . .Show Relay contact info
  * /GETOP. . . . . . . . . . . . . . . . . . . . Try to summon a Relay Operator
  * /H. . . . . . . . . . . . . . . . . . . . . . . . . . . . .Print this list
  * /HElp . . . . . . . . . . . . . . . . . . . . . . . . . . Gives ircII help
  * /IGNore <nickname>. . . . . . . . . . . . . . . . . . . . .  Ignore a user
  * /INFo . . . . . . . . . . . . . . . . . . . . . . . . . Send RELAY INFO file
  * /Invite nickname. . . . . . . . . . . . . . . . .Invite user to your channel
  * /LCL relay. . . . . . . . . . . . . . . . . . . .Show users on a given Relay
  * /LINKMsg <ON | OFF> . . . . . . . . . . . . Turn link messages on or off
  * /LINks. . . . . . . . . . . . . . . . . . . . . . . . . .Shows active Relays
  * /List . . . . . . . . . . . . . . . . . . . . . . . . . List active channels
  * /Msg nick text. . . . . . . . . . . . . . . . . . . . . .Sends private message
  * /NAmes <channel>. . . . . . . . . . . . . . . . . . . . Show users with name
  * /NIck newnick . . . . . . . . . . . . . . . . . . . . . Change your nickname
  * /OPNames. . . . . . . . . . . . . . . . . . . . . . . . .Show active operators
  * /Rates. . . . . . . . . . . . . . . . . . . . . . . . . .Display message rates
  * /REGISTER full name . . . . . . . . . . . . . . . . . . . Register full name
  * /RULES. . . . . . . . . . . . . . . . . . . . . . . Sends the RELAY RULES file
  * /SErvers <node> . . . . . . . . . . . . . . . . . . Show relays serving node
  * /SIGNOFF. . . . . . . . . . . . . . . . . . . . . . . . . Signoff from Relay
  * /Signon <nick> <chan> . . . . . . . . . . . . . . . . . . . .Signon to Relay
  * /SINce. . . . . . . . . . . . . . . . . . . . . .Show recently arrived users
  * /ST . . . . . . . . . . . . . . . . . . . . . . . . . Display Relay statistics
  * /SUmmon userid@node . . . . . . . . . . . . . . . . . . Invite user to Relay
  * /SWho <channel> . . . . . . . . . . . . . . . . . . . . . . . Show nicknames
  * /TList. . . . . . . . . . . . . . . . . . . . List channels which have a topic
  * /Topic <subject>. . . . . . . . . . . . . . . . . . . . Topic for your channel
  * /UNIGNore nickname. . . . . . . . . . . . . . . . . . . . . . .Unignore a user
  * /Users. . . . . . . . . . . . . . . . . . . . . . . . . Show number of users
  * /Who <channel | *>. . . . . . . . . . . . . . . . . .Show users and nicnames
  * /WHOis <nick> . . . . . . . . . . . . . . . . . . . . . . . .Identify a nickname
  *************************** End of List ********************************
```

1.161 Some aliases...

```
Usage: /LOAD columns
  A little script by BigCheese to demonstrate the usage of /echo $[].
  It displays all in and outgoing channel talk formatted to the 11th
  column.
```

1.162 Some aliases...

```
Usage: /LOAD commander

  This file contains a big set of key bindings, one command for
  almost every key you have on your keyboard, prefixed by ESC.
  Both the ESCAPE and ` keys are defined as META1 control key,
  so if you do not have ESC you can use `.
  To obtain a ` you must type ` twice. If you type ESCAPE twice
  you'll instead get the ircII command completion function.
  TAB is defined as alternative META2 key, to toggle insert/overwrite mode
  type TAB twice. ESC-TAB clears the current window.
  Here's the definition of the keys after pressing ESC or `:

    A Prompts you for an away message, ENTER will un-away you.
    C Shows you the /NAMES entry of your current channel.
    D Shows the Who-list of people of your country. ( = /countrywho )
    E Prompts you for the launch of an external command with /EXEC.
    F Follows an invitation to a channel.
    G "Greet" - Enter a query with the last person whose signon has been
    detected by /notify.
    H "Here" - Shows Who-list of users from your site. ( = /localwho )
    I Invite the user who last sent you a message to the current channel.
    J Shows the WHOIS of the person who last joined your channel.
    L List of channels with at least 3 persons on unless with topic.
    M "More" - Enter a query with the person you last sent a msg to.
    N Names list of public channels having at least 2 persons on.
    O Who-list of operators online.
    P Make your channel private.
    Q Exit a query.
    R "Reply" - Enter a query with the person who last sent you a msg.
    S Prompts you for a new server, ENTER gives a list of stored ones.
    T Prompts you for a topic for the current channel.
    U Tells you how many users and servers are on IRC.
    W Gives you the Who-list of the people on your channel.
    X "eXamine" - WHOIS of the person who last sent you a message.
    Y Toggles the SCROLL mode.
    Z Toggles the HOLD_MODE.
    , Flushes the server output.
    . Clears the window.
```

1.163 Some aliases...

```
Usage: /LOAD cursor

    I don't know why, it works on some terminals, on others it doesn't:
This should setup ircII to make the cursor keys on your terminal move
the cursor in ircII left/right and forward/back in command history.

Try it, maybe you are lucky.
```

1.164 Some aliases...

```
Usage: /LOAD deutsch

This is only for the german-speaking part of the IRC community.

Hier ist nix sinnvolles drin, nur eine kleine Sammlung von Tastenbelegungen,
die mit META2 erreichbar ist. META2 ist per default auf ^w gelegt (CTRL-W),
wer aber 'commander' verwendet hat es auch auf TAB zur Verfuegung.

  f *fauch*
  g *grins*
  G *gaehn*
  k *kicher*
  l *lach*
  M moin <letzter joiner>!
  R rehi <letzter joiner>!
  s *seufz*
  S *schmoll*
  t *troest*
  T tach <letzter joiner>!

Hiermit habe ich also mein Geheimnis fuer schnelle virtuelle Emotivitaet
preisgegeben. Man kann sich echt dran gewoehnen! *grins*   ;)

Boah ey!  :)
```

1.165 Some aliases...

```
Usage: /LOAD english

   For those who didn't like that ircII has special features for germans
only, here is just about the same silly thingie in english language.
This might however become obsolete if /me & co. become a standard.

If you load this script you get the following strings typed into your
ircII line as you type META2, that is ^w (CTRL-W) or TAB for those
who make use of 'commander' plus the character with or without SHIFT.

  a *applaud*
  A *gasp*
  b *blush*
  B *bow*
  c *cry*
  C *cough*
  f *frown*
  g *grin*
  G *giggle*
  h *hug*
  k *kiss*
  l *laugh*
  r *purr*
  s *sigh*
  S *shrug*
  y *yawn*
```

1.166 Some aliases...

```
Usage: /LOAD events

  This script contains a collection of /ON commands which change the looks
  of some 'events' to something that I like more, maybe you like it, too?

  The person's nickname that last sent you a message will appear in
  your status-line, if you have %U in your STATUS_FORMAT variable.
  The output of /NAMES will look more compact.
  The output of /WHO will look different, too.
  When you type to a channel your nickname will be displayed as prefix to
  the line instead of just '> '.
  You will be informed about the time every full hour.
```

1.167 Some aliases...

```
Usage: /LOAD finger

    The mythical /finger alias by BigCheese. Apparently it works again.
You can type /finger <nickname> and a 'finger' command is launched with the
user@host info for that nickname.
```

1.168 Some aliases...

```
IRC ForumNet Compatibility System

To use the compatability package, you should do:
  /LOAD fnet
If this doesn't work, check with your IRCII installer.

Group commands
  /group  name  Change to group +name
  /status [s|p|h|u] Change group status
  /boot nick  Expel nick from group
  /invite nick  Invite nick to the group
  /pass nick  Make nick a group moderator
  /nm nick  Remove moderation from nick
  /w    List groups and users
  /topic  topic Change the topic of the group

Message commands
  /m  nick  Send a message to nick
  /beep nick  Send a beep to nick
  /hclear   Clear /m history

General commands
  /commands Show this command list
  /display n  Redisplay the last n messages
  /nick newnick Change your nickname to newnick
  /motd   Read the current message of the day
  /q    Quit IRC
  /fset var Set or display an IRC variable.
```

1.169 The startup-file

```
You could make up a nice .ircrc to start with, simply by writing these 3
lines into it:

load commander
load events
load alias

Read the helps to learn about what you get by using these scripts.
They are designed to bring much of ircII's power to the user's
disposition which is otherwise hard to get, because you'd have to learn
ircII's complex and unfriendly programming language.

About the key definitions, you get much faster in discussion if
you get acquainted to its use, especially of the fast /query bindings
which can help you switch between multiple conversations quickly.
```

1.170 Some aliases...

```
Usage: /LOAD local

  Local definitions, done by your ircII administrator are stored in here.
  Local is (should be) executed at ircII startup automatically.

  The most important aliases in here, are these two aliases:

  /COUNTRYWHO Who-list of the people of your country.
  /LOCALWHO Who-list of the people at your site.

  With 'commander' you have these commands on ESC-d and ESC-h respectively.
```

1.171 Some aliases...

```
                Usage: /LOAD msg

    Contains some handy key bindings:
A variable LAST_WHOIS will always contain the person you last /whois'd.
A variable LAST_NOTIFY will contain the person that was last detected to
sign onto irc.

By typing META2 with ? you will be prompted '/msg <last_whois> '
With META2 and ! you'll get '/msg <last_notify> '

Additionally you also have META2-, which expands to
 '/msg <last person that sent you a msg> '
and META2-. to '/msg <last person you sent a msg to> '
analogue to the standard IRC shortcuts ('/msg .' and ',').
```

```
META2 is defined by /bind, by default it is ^x.

  See Also:

                 LOAD COMMANDER
```

1.172 Some aliases...

```
Usage: /LOAD mudlike

    ircII interface for people who like the way one communicates on MUDs.
It makes ircII act almost the same way:

Removes the / before commands, everything you type is supposed to be
a command. To speak to a channel use 'say <text>', to speak to a person
use 'tell <person> <text>' or use 'query <person>' and 'send <text>'.

When you 'load action' you get the 'smile' 'grin' commands which
ALSO transmit over a query! So be careful if you are in a query or not.

The ' symbol itself is a quickie for the 'send' command which either
sends to the query or the channel. If you type one space before text,
that is space as a command like this: ' <text>', it goes to the channel.

Much of the irc output is modified in a way that it looks more familiar
to MUDders.

'emote' behaves like 'me', see /help ME. Read also /help DESCRIBE

For more precise information try to read the script file itself.
```

1.173 Some aliases...

```
Usage: /load OPER

A collection of aliases for IRC operators:

/stick <new> <target> /connect <new> 6667 <target>
/joinzone   /join #Twilight_Zone
/partzone   /part #Twilight_Zone
/joineuro   /join #EU-Opers
/parteuro   /part #EU-Opers

/loud     Show the wallops
/silent     Swallow the wallops away
/lastwallops    Should show the last wallop, but I think it doesn't
```

1.174 Some aliases...

```
Usage: /LOAD prefix

  For those who use multiple channels without opening windows
  for each of them. The channel name talk is coming from or going
  to is displayed as prefix to each line.
```

1.175 Some aliases...

```
This is a sample of using the WHILE command in an alias.  Once
loaded, here is it's usage:

Usage: REPEAT <cnt> <cmd>
  This will perform <cmd> for the number of times specified by
  <cnt>.  For example:
    /REPEAT 5 /msg bigcheese Hello!
  This will send the message "Hello!" to bigcheese 5 times.
  The above example can get annoying, so try to not to use
  this to really bother people.
```

1.176 Some aliases...

```
Usage:  /LOAD service
  This script contains some aliases for the handling of
  NickServ and NoteServ which are shortcuts for the
  /msg NickServ/NoteServ - Sequences.
  You have to setup a Variable which contains your NickServ-
  Password with /ASSIGN PASSWD password in your .ircrc.
  When you then /LOAD SERVICE that password will automatically
  be sent to the NickServ.

  For NickServ :

   /.identify    /.whois

  For NoteServ :

   /.list     /.tell (or /.msg)
   /.read     /.query
   /.away     /.delete
   /.await (or /.waitfor)
```

1.177 Some aliases...

```
Usage: /LOAD silent

  A series of /on commands, which will make all of the *** join, leave,
  signoff nickname and mode change messages disappear. Meant to be used
  in high-traffic channels such as +report.
  If you have %U somewhere in your STATUS_FORMAT variable, the messages
  will appear in the status line instead, which is much less annoying,
  than to have them in the ircII windows.
```

1.178 Suggestions...

```
Hello!

If you have suggestions about the ircII script system, the scripts themselves,
or even have some contributions please send them to my address for me to
examine them and include them into distribution of ircII. Consider also
including help files for your scripts.

My address is:

  Carl 'lynX' v. Loesch:  <loesch@informatik.uni-oldenburg.de>

I'm always curious about what people do with the ircII language... :)
```

1.179 Some aliases...

```
Usage: /LOAD window
  This script contains bindings for Window handling in an EMACS-like
  fashion. After pressing the META2 key (either CTRL-X or TAB) you
  can press one of these:

  2 Open a new window.
  + or z  Grow the current window by one line.
  - or ^z Shrink the current window by one line.
  = Balance the size of all windows.
  0 Kill the current window!
  1 Kill all other windows!
```

1.180 Note commands

```
Usage: NOTE [CAT] [$passwd] [+-FLAGS] [#ID] <nick!username@host>
  Displays nick/username/tohost sent from your nickname and username
  including the content of the message. See HELP LS for more info.
  See HELP FLAG for more info about flag settings.
```

1.181 Note commands

```
Usage: NOTE [CHANNEL] [$pwd] [+-FLAGS] [+-maxtime] <nick!username@host> <msg>
  CHANNEL is an alias for USER +M (default max 7 days)
  This command is for sending a message to recipient only if this is on
  channel +note.
  Note: This channel will be used for broadcast messages about the note
```

```
      system. Join and leave to check if it's any messages at any time.
  Example: CHANNEL foobar This is a test
           CHANNEL +N +7 foobar!username@*.edu Hi there!
```

1.182 Note commands

```
Usage: NOTE [COUNT] [$passwd] [+-FLAGS] [#ID] <nick!username@host>
  Displays the number of messages sent from your nick and username. See
  HELP LS for more info. See HELP FLAG for more info about flag setting.
```

1.183 Note commands

```
*Usage: NOTE [DENY] [$passwd] [+-FLAGS] [+-maxtime] <nick!user@host> <msg>
*  DENY is an alias for USER +RZ (default max 1 day)
*  This command makes it impossible for any matching recipient to
*  use SEND option.
```

1.184 Note commands

```
Usage: NOTE [FIND] [$passwd] [+-FLAGS] [+-maxtime] <nick!username@host> <msg>
  FIND is an alias for USER +FLR (default max 1 day)
  This command makes the server search for any matching recipient, and
  send a note message back if this is found. Max matching is set as MUM.
  <msg> field should be the realname of the person if specified.
  Example: FIND -4 foo*!username@host
           FIND @host Internet*
```

1.185 nnote

```
Usages:
  NOTE [USER] [$passwd] [+-FLAGS] [+-maxtime] <nick!username@host> <msg>
-  or  [SEND | CHANNEL | SPY | FIND | WAITFOR] <see USER args.>
*  or  [SEND|CHANNEL|SPY|FIND|WAITFOR|PLOG|WALL|WALLOPS|DENY] <see USER args.>
  NOTE [LS|CAT|COUNT|RM|LOG] [$pwd][+-FLAGS][#ID] <nick!user@host> [date]
  NOTE [FLAG] [$passwd] [+-FLAGS] [#ID] <nick!username@host> <+-FLAGS>
-  NOTE [SENT] [NAME|COUNT]
*  NOTE [SENT] [NAME|COUNT|USERS] <f.nick!f.name@host> <date> [RM|RMAB]
-  NOTE [WHOWAS] [+-FLAGS] [#max] <nick!name@host> [date1] [date2]
*  NOTE [WHOWAS] [+-FLAGS] [#max] <nick!name@host> [date1] [date2] [RM|RMS]
-  NOTE [STATS] [MSM|MSW|MUM|MUW|MST|MSF|USED]
*  NOTE [STATS] [MSM|MSW|MUM|MUW|MST|MSF|USED|RESET] [value]
*  NOTE [SAVE]

  The NOTE system let you send one line messages to irc users
  which they will get if and when they next sign on to irc,
  change nick or channel. If they are on irc already, they'll
  get the note message immediately.
  NOTE has built in a lot of other features too, as possibility to
  log users and lots of other message handling as find users,
  tell if an user change nickname or sign on/off... and a lot more.

  Suggestions should be sent to jarlek@ifi.uio.no (Wizible on IRC).
```

1.186 Note commands

```
Usage: NOTE [FLAG] [$passwd] [+-FLAGS] [#ID] <nick!username@host> <+-FLAGS>
  You can add or delete as many flags as you wish with +/-<FLAG>.
  + switch the flag on, and - switch it off. Example: -S+RL
  Following flags with its default set specified first are available:
  -S > Message is never saved. (Else with frequency specified with /MSF)
  -P > Repeat the send of message(s) once.
  -M > Ignore message if recipient is not on a channel named NOTE.
  -Q > Ignore message if recipient’s first nick is equal to username.
  -V > Ignore message if recipient is not on a server in same country.
  -I > Ignore message if recipient is not on this server.
  -W > Ignore message if recipient is not an operator.
  -U > Ignore message if recipient is not signing on or off IRC.
  -Y > Ignore message if sender is not on IRC.
  -N > Let server send a note message if message is sent to recipient.
  -D > Same as N, but msg. is sent to sender if this is on IRC.
  -R > Repeat the message until timeout.
  -F > Let server send note info for matching recipient(s). Any message
       specify what to match with the realname of the recipient.
  -L > Same as F, but message is sent to sender if this is on IRC.
  -C > Make sender’s nicks be valid in all cases username is valid.
  -X > Let server display if recipient signs on/off IRC or change
       nickname. Any message specified is returned to sender.
  -E > Ignore message if nick, name and host matches the message text
       starting with any number of this format: ’nick!name@host nick!... ’
*  -A > Generate a whowas list for all matching nick/name/host. Timeout
*       for whowas messages is set identical to this flag message timeout.
*       However, separate timeout is set if specified as a message.
*  -J > Whowas list log new time for each user every day.
*  -B > Let server generate a header message to matching nick/name/host
*       Message is only sent to each matching recipient once.
*  -T > A or B flag list includes nicks for all matching names and hosts.
*  -K > Give keys to unlock privileged flags by setting that flags on.
*       The recipient does also get privileges to queue unlimited msg.,
*       list privileged flags and see all stats.
*  -Z > Make it impossible for recipient to use SEND option.
  Other flags which are only displayed but can’t be set by user:
  -O > Message is sent from an operator.
  -G > Notice message is generated by server.
-  -T > Whowas list has included nicks for all matching names and hosts.
-  -J > Whowas list log new time for each user every day.
-  -B > Broadcasting message sent once to all who matches recipient name.
*  -H > Header message generated using flag B.
-  Notice: Message is not sent to recipient using F, L, R or X flag.
-  Using this flags, no message needs to be specified.
*  Notice: Message is not sent to recip. using F, L, R, X, A, K, Z or H
*  flag (except if B flag is set for R). For this flags, no msg. needed.
```

1.187 Note commands

```
Usage: NOTE [LOG] [$passwd] [+-FLAGS] [#ID] <nick!username@host>
  Displays the last time matching person was on IRC. This works only
  after use of NOTE Spy. The log is protected to be seen for other
  users than the person who queued the SPY request.
```

1.188 Note commands

```
Usages:
  NOTE [USER] [$passwd] [+-FLAGS] [+-maxtime] <nick!username@host> <msg>
-  or  [SEND | CHANNEL | SPY | FIND | WAITFOR] <see USER args.>
*  or  [SEND|CHANNEL|SPY|FIND|WAITFOR|PLOG|WALL|WALLOPS|DENY] <see USER args.>
  NOTE [LS|CAT|COUNT|RM|LOG] [$pwd][+-FLAGS][#ID] <nick!user@host> [date]
  NOTE [FLAG] [$passwd] [+-FLAGS] [#ID] <nick!username@host> <+-FLAGS>
-  NOTE [SENT] [NAME|COUNT]
*  NOTE [SENT] [NAME|COUNT|USERS] <f.nick!f.name@host> <date> [RM|RMAB]
-  NOTE [WHOWAS] [+-FLAGS] [#max] <nick!name@host> [date1] [date2]
*  NOTE [WHOWAS] [+-FLAGS] [#max] <nick!name@host> [date1] [date2] [RM|RMS]
-  NOTE [STATS] [MSM|MSW|MUM|MUW|MST|MSF|USED]
*  NOTE [STATS] [MSM|MSW|MUM|MUW|MST|MSF|USED|RESET] [value]
*  NOTE [SAVE]

  The NOTE system let you send one line messages to irc users
  which they will get if and when they next sign on to irc,
  change nick or channel. If they are on irc already, they'll
  get the note message immediately.
  NOTE has built in a lot of other features too, as possibility to
  log users and lots of other message handling as find users,
  tell if a user change nickname or sign on/off... and a lot more.

  Suggestions should be sent to jarlek@ifi.uio.no (Wizible on IRC).
```

1.189 Note commands

```
*Usage: NOTE [PLOG] [$passwd] [+-FLAGS] [+-maxtime] <nick!user@host> <msg>
*  PLOG is an alias for USER +AR (default max 1 day)
*  This command is to activate the public log system. The system will
*  generate a list which can be read by anybody using WHOWAS.
*  +I flag log only users on your server.
*  +T flag log all nicks used by an account (only last is default).
*  +J flag makes a new log each day of any matching recipient.
*  You may queue severale log requests at the same time as specified here;
*  1. PLOG +i +365 * +60
*  2. PLOG +t +365 @*.edu -12
*  3. PLOG +365 * +1
*  This system log all user@host using your server for last 7 days,
*  log all users including nick-names for everybody on host @*.edu
*  for last 12 hours, and finally log every user@host on the IRC
*  for one day (24 hours). This process is repeated for 365 days.
```

1.190 Note commands

```
Usage: NOTE [RM] [$passwd] [+-FLAGS] [#ID] <nick!username@host>
  Deletes any messages sent from your nick and username which matches
  with nick and username@host. Use FLAGS for matching all messages which
  have the specified flags set on or off. See HELP FLAG for more info
  about flag settings, and HELP LS for info about time.
```

1.191 Note commands

```
*Usage: NOTE [SAVE]
*  SAVE saves all messages with the save flag set. Notice that the
*  messages are automatically saved (see HELP STATS). Each time server is
*  restarted, the save file is read and messages are restored. If no users
*  are connected to this server when saving, the ID number for each
*  message is renumbered.
```

1.192 Note commands

```
Usage: NOTE [SEND] [$passwd] [+-FLAGS] [+-maxtime] <nick!username@host> <msg>
  SEND is an alias for USER +D (default max 7 days)
  This command is for sending a message to recipient, and let the server
  send a note + a notice to sender if this is on IRC - if message is sent.
  Example: SEND foobar Hello, this is a test.
           SEND +7 !username@*.edu Hello again!
```

1.193 Note commands

```
-Usage: NOTE [SENT] [NAME|COUNT]
*Usage: NOTE [SENT] [NAME|COUNT|USERS] <f.nick!f.name@host> <date> [RM|RMAB]
  Displays host and time for messages which are queued without specifying
  any password. If no option is specified SENT displays host/time for
  messages sent from your nick and username.
  NAME displays host/time for messages sent from your username
  COUNT displays number of messages sent from your username
*  USERS Displays the number of messages in [], and names for all users
*  who have queued any message which matches with spec. nick/name/host.
*  RM means that the server removes the messages from the specified user.
*  RMAB means same as RM, but only messages with A or B flag is removed.
```

1.194 Note commands

```
Usage: NOTE [SPY] [$passwd] [+-FLAGS] [+-maxtime] <nick!username@host> [msg]
  SPY is an alias for USER +RX (default 1 max day)
  SPY makes the server tell you if any matching recipient sign(s)
  on/off IRC or change nick name. No message needs to be specified.
  However, if a message is specified this is returned to sender including
  with the message about recipient. Message could for example be one or
  more Ctrl-G characters to activate the bell on senders machine.
  The system logs last time the last matching person was on IRC for
  each SPY request is queued in the server. See NOTE LOG for this.
  Example: SPY foobar!username@host <ctrl-G>
           SPY +10 foobar
           SPY $secret * <ctrl-G>
```

1.195 Note commands

```
-Usage: NOTE [STATS] [MSM|MSW|MUM|MUW|MST|MSF|USED]
*Usage: NOTE [STATS] [MSM|MSW|MUM|MUW|MST|MSF|USED|RESET] [value]
  STATS with no option displays the values of the following variables:
  MSM: Max number of server messages.
  MSW: Max number of server messages with username-wildcards.
  MUM: Max number of user messages.
  MUW: Max number of user messages with username-wildcards.
  MST: Max server time.
  MSF: Note save frequency (checks for save only when a user register)
  Notice that 'dynamic' mark after MSM means that if there are more
  messages in the server than MSM, the current number of messages are
  displayed instead of MSM.
  Only one of this variables are displayed if specified.
*  You can change any of the stats by specifying new value after it.
*  RESET sets the stats to the same values which is set when starting the
*  server daemon if no note file exist. Notice that this stats are saved
*  in same file as the other messages.
```

1.196 Note commands

```
Usage: NOTE [USER] [$passwd] [+-FLAGS] [+-maxtime] <nick!username@host> <msg>
  With USER you can queue a message in the server, and when the recipient
  signs on/off IRC, change nick or join any channel, note checks for
  valid messages. This works even if the sender is not on IRC. See HELP
  FLAGS for more info. Password can be up to ten characters long. See
  HELP FLAGS for info about flag settings. Username can be specified
  without @host. Do not use wildcards in username if you know what it
  is, even if it's possible. Max time before the server automatically
  remove the message from the queue, is specified with hours with a
  '-' character first, or days if a '+' character is specified, as
  -5 hours, or +10 days. Default maxtime is 7 days.
  Note: The received message is *directly* displayed on the screen,
  without the need for a read or remove request.
  NOTE USER $secret +WN +10 Wizible!jarlek@ifi.uio.no Howdy!
  - is an example of a message sent only to the specified recipient if
  this person is an operator, and after receiving the message, the server
  sends a note message back to sender to inform about the delivery.
  NOTE USER +XR -5 Anybody <ctrl-G>
  - is an example which makes the server to tell when Anybody signs
  on/off irc, change nick etc. This process repeats for 5 hours.
  NOTE USER +FL @*.edu *account*
  - is an example which makes the server send a message back if any real-
  name of any user matches *account*. Message is sent back as note from
  server, or directly as a notice if sender is on IRC at this time.
```

1.197 Note commands

```
Usage: NOTE [WAITFOR [$pwd] [+-FLAGS] [+-maxtime] <nick!username@host> [msg]
  WAITFOR is an alias for USER +YD (default max 1 day)
  Default message is; <<< WAITING FOR YOU ON IRC RIGHT NOW>>>
  This command is for telling the recipient if this appears on IRC that
  you are waiting for him/her and notice that this got that message.
  Example: WAITFOR foobar
           WAITFOR -2 foobar!username@*
           WAITFOR foobar Waiting for you until pm3:00..
```

1.198 Note commands

```
*Usage: NOTE [WALL] [$passwd] [+-FLAGS] [+-maxtime] <nick!user@host> <msg>
*  WALL is an alias for USER +BR (default max 1 day)
*  This command is for sending a message once to every matching user
*  on IRC. Be careful using this command. WALL creates a list of
*  persons received the message (and should not have it once more time)
*  with H flag set. This list can be displayed using LS +H from the
*  nick and username@host which the WALL request is queued from.
*  Removing the headers (H) before WALL request is removed would cause
*  the message to be sent once more to what users specified in list.
*  WALL +7 @*.edu Do not do this! - Makes it clear for all users using
*  IRC on host @*.edu the next 7 days how stupid it is to send such WALL's ;)
```

1.199 Note commands

```
*Usage: NOTE [WALLOPS] [$passwd] [+-FLAGS] [+-maxtime] <nick!user@host> <msg>
*  WALLOPS is an alias for USER +BRW (default max 1 day)
*  This command is same as WALL, except only opers could receive it.
```

1.200 Note commands

```
-Usage: NOTE [WHOWAS] [+-FLAGS] [#max] <nick!name@host> [date1] [date2]
*Usage: NOTE [WHOWAS] [+-FLAGS] [#max] <nick!name@host> [date1] [date2] [RM|RMS]
  Displays information about who used the given nickname, username or
  hostname the first and last time (every day or any time) this person
  was on IRC. If a time is specified, only usernames from that day are
  displayed. If another time is specified, all users which are dated from
  time1 to time2 are displayed. See HELP LS for more info about time.
*  RM or RMS means that the specified username is deleted. RM displays
*  the deleted usernames, but RMS does not display that.
  If no names are specified, information is displayed about what group
  of users is registered in the whowas system.
  Following flags with its default set specified first are available;
  -I > Lists persons on this server only.
  -O > List only operators.
  -C > Counts number of matches.
  -U > List only one nick per username and host.
  -T > List only log generated with T flags.
  -J > List only log generated with J flags.
```

1.201 About the ON-command

```
              Usage: ON [+|-|^]<event> [-|^]<match> [action]
  The ON command lets you set up actions which will occur when
  certains events happen.  For example, you can create
  "personalized" away messages for different individuals,
  periodic actions that occur at specific times, just to name
  a few.

  Note that using ON JOIN and ON LEAVE to automatically greet
  people entering or leaving your channel is extremely poor
  etiquette. See /HELP ETIQUETTE.

  The <event> parameter specifies what irc event to monitor for
  possible execution of the action.  All possible events are listed
  after this file, and more specific help is available for each.

  There are four modes for each event.  Different modes
  are specified by preceeding the event with one of the
  following characters:
    +           Make event very noisy.  It will display
                everything it does.
    -           Make event quiet.  It will not display any of
                it’s actions.
    ^           Make event silent.  Just like - above except it
                inhibits the normal event display as well.  The
                only command that will cause output in this mode
    is the ECHO command.  This can be used to redefine
                how event messages look.
  If no character is specifed, you will be warned upon activation
  of an event (this is somewhere between the + and - modes).

  The <match> parameter may be a single word, or a list of words
  surrounded by double quotes (").  The <match> determines
  specifically which events of the given <event> will cause
  the action to take place.  The <match> may contain wildcards,
  and the specifics of the <match> parameter are dependant on
  the <event> type.

  If the <match> is preceeded by a -, that entry will be
```

```
removed from the action list.  For example:
/ON PUBLIC -BigCheese
/ON INVITE  -"* +Har"
The first will remove "BigCheese" from the PUBLIC list, the second will
remove -"* +Har" from the INVITE list.
If you specify a - without a <match>, all entries on that list will
be removed.  For example:
/ON MSG -
will remove all entries from the MSG list.

If the <match> is preceeded by a ^, then no action will be taken
for a match for that event (this is used when you wish
to exclude a nickname or list of nicknames from a larger
wildcard match, see HELP ON FLOOD for an example).

The [action] parameter is that action that will occur when the
specified event occurs which matches the given <match>.
The action parameter is parsed by the command parse in exactly
the same way as an IRCII alias is parsed, using the same inline
variable exapansions (See ALIAS).  Also, depending on the
event that is activated, there will be certain values passed
as arguements to the action, just as if they were arguments to
an alias.  You may use the $0, $1, $2... etc variables to
expand out these arguments.  The arguments passed for each
event are described in the individual help files for each event.

Here are some examples:
  /ON -MSG *cheese* NOTICE $0 No Cheese Messages Please!
Whenver a MSG is received from anyone with a nickname with
cheese in it, a NOTICE will be sent out to that user.  This
action will generate no noticable effect to you (it is marked
as "quiet").

  /ON MSG ^BigCheese
If used with the first example above, this will prevent the
action from being taken when a message is sent from BigCheese.

Here is an example of how to redefine the way a
message looks:
  /ON ^MSG * ECHO :$0: $1-
All MSGs received from now on will look:

:BigCheese: Hello!

The same can be done with the other event types.

More examples are given for the individual event types.

/ON will not function if the NOVICE variable is set to ON.

See Also:

               IRCII PROGRAMMING

               ALIAS
```

1.202 Details about the ON-command

```
Using of ON:
  The <event> parameter specifies what irc event to monitor for
  possible execution of the action.  All possible events are listed
  after this file, and more specific help is available for each.

  There are four modes for each event.  Different modes
  are specified by preceeding the event with one of the
  following characters:
    +          Make event very noisy.  It will display
               everything it does.
    -          Make event quiet.  It will not display any of
               it's actions.
    ^          Make event silent.  Just like - above except it
               inhibits the normal event display as well.  The
               only command that will cause output in this mode
               is the ECHO command.  This can be used to redefine
               how event messages look.
  If no character is specifed, you will be warned upon activation
  of an event (this is somewhere between the + and - modes).

  The <match> parameter may be a single word, or a list of words
  surrounded by double quotes (").  The <match> determines
  specifically which events of the given <event> will cause
  the action to take place.  The <match> may contain wildcards,
  and the specifics of the <match> parameter are dependant on
  the <event> type.

  If the <match> is preceeded by a -, that entry will be
  removed from the action list.  For example:
        /ON PUBLIC -BigCheese
        /ON INVITE  -"* +Har"
  The first will remove "BigCheese" from the PUBLIC list, the second will
  remove -"* +Har" from the INVITE list.
  You can remove all entries from a given event list by doing
  the following:
  /ON <event> -
  For example:
  /ON MSG -
  removes all entries in the ON MSG list.

  If the <match> is preceeded by a ^, then no action will be taken
  for a match for that event (this is used when you wish
  to exclude a nickname or list of nicknames from a larger
  wildcard match, see HELP ON FLOOD for an example).

  The [action] parameter is that action that will occur when the
  specified event occurs which matches the given <match>.
  The action parameter is parsed by the command parse in exactly
  the same way as an IRCII alias is parsed, using the same inline
  variable exapansions (See ALIAS).  Also, depending on the
  event that is activated, there will be certain values passed
  as arguements to the action, just as if they were arguments to
  an alias.  You may use the $0, $1, $2... etc variables to
  expand out these arguments.  The arguments passed for each
  event are described in the individual help files for each event.

  Here are some examples:
    /ON -MSG *cheese* /NOTICE $0 No Cheese Messages Please!
  Whenver a MSG is received from anyone with a nickname with
  cheese in it, a NOTICE will be sent out to that user.  This
  action will generate no noticable effect to you (it is marked
  as "quiet").

    /ON MSG ^BigCheese
  If used with the first example above, this will prevent the
  action from being taken when a message is sent from BigCheese.

  Here is an example of how to redefine the way a
  message looks:
    /ON ^MSG * /ECHO :$0: $1-
  All MSGs received from now on will look:

  :BigCheese: Hello!

  The same can be done with the other event types.

  More examples are given for the individual event types.
```

1.203 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]CHANNEL_NICK [-|^]"<channel> [<nick>]" [action]
  This is activated whenever someone on a channel you are on
  changes their nickname. The parameters for this function are:
    $0  The channel from which that person has changed nick
    $1  Original nickname for that person
    $2  New nickname for that person
```

1.204 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]CHANNEL_SIGNOFF [-|^]"<channel> [<nick>]" [action]
  This is activated whenever someone on a channel you are on
  leaves irc.  Normally, the display would show:

  *** Signoff: <nickname>

  but, of course, you can change this.  If <nick> is your own
  nickname, then the action will be taken when you exit
  from irc.  The parameters are as follows:
    $0  The channel from which the person signed off
    $1  Nickname of person who left irc
```

1.205 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]CONNECT [-|^]<server> [action]
  This is activated whenever you receive a connect to a server.
  The parameters for the action are as follows:
    $0    The name of the server you connected to.
```

1.206 An ON-command (do an action when something happens)

```
                Usage: ON [+|-|^]CTCP [-|^]<parameters> [action]
  This event is activated whenever someone sends a client-to-client
  protocol (CTCP) request.  Certain CTCP types are have predefined
  actions that you can't override with the ^, but you can use the
  to create your own CTCP protocols.  The parameters are:
        $0      nick of person who send CTCP
        $1      who the ctcp was to (either your nick or a channel)
        $2      The CTCP command word
        $3-     Any additional arguments
  For example, if you want to set up a new ctcp function called
  CHEESE, you  would do:
        ON -CTCP "* * CHEESE" /CTCP $0 $2 Say Cheese!
  Using the /CTCP in an ON CTCP automatically sends the proper
  reply format.  If someone no send you a:
        CTCP <yournick> CHEESE
  you will see:
        *** CTCP CHEESE from <theirnick>:
  and they will see:
        *** CTCP REPLY CHEESE from <yournick>: Say Cheese!

  See Also:
```

CTCP

ON CTCP_REPLY

1.207 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]CTCP_REPLY [-|^]<parameters> [action]
  This event is activated when you receive a reply from a CTCP
  request that you made (with the CTCP command).  For example,
  if you did:
  /CTCP BigCheese VERSION
  you would receive a:
  *** CTCP REPLY VERSION from BigCheese: IRCII 2.1beta2 Unix
  from BigCheese.  However, if you had:
  /ON ^CTCP_REPLY "* VERSION" /echo $0 is using $2-
  and did the CTCP show above, you would get something like:
  BigCheese is using IRCII 2.1beta2 Unix
  The parameters are:
  $0  nick of person who sent the reply
  $1  The CTCP command being replied to
  $2- The reply itself (depends on $1)
  Note: You may NOT use the CTCP command in an ON CTCP_REPLY.
  It simply will report an error message to you and not work.
  It is illegal to do this and it is not supported.
```

1.208 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]DCC_CHAT [-|^]<nicknames> [action]
  This is activated whenever you receive a DMSG from someone.
  The parameters for the action are as follows:
    $0    nickname of person who sent the DMSG
    $1-   The message

  Warning: As DMSG is outside the IRC protocol, it allows you to send
  a DMSG from an ON DCC_CHAT. This has the potential to create loops.
  If you intend to send a DMSG from DCC_CHAT you are responsible for
  ensuring that it will not create a loop.
```

1.209 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]DISCONNECT [-|^]<parameters> [action]
  This is activated whenever you are disconnected from an
  irc server and IRCII's automatic reconnection techniques
  can't get you reconnected.  This would be any situtation
  which you would see "*** Use /SERVER to connect to a server".
  Parameters:
  $0  Last server you were connected to
```

1.210 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]EXEC [-|^]<process number> [action]
  This is activate whenever an EXEC'd process produces output
  to its standard output channel.  The process number is the
  number assigned by EXEC to the process (see HELP EXEC).  The
  parameters to this function are:
    $0  The process number that activated the ON
    $1- The line of output to stdout
```

1.211 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]EXEC_ERRORS [-|^]<process number> [action]
  This is activate whenever an EXEC'd process produces output
  to its standard error channel.  The process number is the
  number assigned by EXEC to the process (see HELP EXEC).  The
  parameters to this function are:
    $0  The process number that activated the ON
    $1- The line of output to stderr
```

1.212 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]EXEC_EXIT [-|^]<parameters> [action]
  This is activated when any EXEC'd process exits.  The parameters
  are:
  $0  Process number or logical name (if assigned)
  $1  Signal that killed the process (or 0 if it exited normally)
  $2  Exit code for process (non-zero only if $1 is 0)
```

1.213 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]EXEC_PROMPT [-|^]<parameters> [action]
  This is activated whenever an EXEC'd process displays a
  "prompt".  A "prompt" is defined as any line of output from
  a process that doesn't end in a carriage return - line feed.
  The parameters are:
  $0  process number or logical name (if assigned)
  $1  The prompt string
```

1.214 An ON-command (do an action when something happens)

```
                Usage: ON [+|-|^]FLOOD [-|^]<nick> [action]
  The FLOOD type is activated whenever someone sends a lot of
  information very rapidly to you, your channel, or the system at
  large (ie: a wall).  Usually, this is caused when someone dumps
  a file to your channel or to you.  The parameters for this are:
    $0  The nick of the flooder
    $1  The type of flooding being done
    $2- The content of the flooding message

  General use for ON FLOOD is as follows:
  /ON ^FLOOD *
  This will automatically prevent flooding messages from being
  displayed and automatically disable itself when the flooding
  has stopped.

  If you use services like IRCIIHelp or NickServ, these can appear to
  IRCII as floods, but if you add to following ON lines:
   /ON FLOOD ^IRCIIHelp
   /ON FLOOD ^NickServ
  These will not be treated as floods and will let their messages through.

  Warning:  There are occasions where ON FLOOD can be inadvertantly
  activated (such as after a net-burp, or after you have ^Z'd for a
  while).  Therefore, I strongly advise against using KILL or even
  KICK as the action taken by an ON FLOOD.

  See Also:

                  SET FLOOD_AFTER

                  SET FLOOD_RATE
```

```
                 SET FLOOD_USERS

                 SET FLOOD_WARNING
```

1.215 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]HELP [-|^]<parameters> [action]
  ON HELP is activated for each line of the HELP command
  that is displayed.  There is probably no other use
  for this except by IRCIIHelp.  So ignore it.
```

1.216 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]IDLE [-|^]<parameters> [action]
  This is activated when IRCII has been idle for a period of
  time.  Idleness is in minutes from the last keystroke.
  Parameters:
        $0       minutes idletime

  So, /ON IDLE 5 /ECHO FINISHED will print out "FINISHED"
  after a period of 5 minutes without any key pressed.
```

1.217 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]INPUT [-|^]<parameters> [action]
  This is activated whenever you send a line of input to either
  a channel or a query user.  This means that if you type in a
  message and hit return, ON INPUT can be activated.  It is NOT
  activated by IRCII commands (any line starting with a CMDCHAR).
  Parameters are:
  $0  Channel or query user or 0 if neither.
  $1- Text of message
```

1.218 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]INVITE [-|^]<nicknames> [action]
  This is activated whenever you receive an INVITE.  The
  parameters for the action are as follows:
    $0    nickname of person who invited you
    $1    the channel to which you are being invited
```

1.219 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]JOIN [-|^]<nicknames> [action]
  This is activated whenever someone joins a channel you are on.
  The parameters for the action are as follows:
    $0    nickname of person who joined the channel
    $1    the channel which was joined
```

1.220 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]LEAVE [-|^]<nicknames> [action]
  This is activated whenever someone leaves a channel you are on.
  The parameters for the action are as follows:
    $0    nickname of person who left the channel
    $1    the channel which which was left
```

1.221 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]LIST [-|^]<paramters> [action]
  When /LIST is activated any outputline will be modified
  by [action].
  For example:
    /ON LIST "* 5 *" /ECHO NUMBER OF USERS: 5

  The parameters are as follows:
    $0    channel name
    $1    number of users on channel
    $2-   topic of channel
```

1.222 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]MAIL [-|^]<count> [action]
  This is activated whenever you receive new mail (non-irc mail)
  or when you start up irc and you have mail.  You can use this
  to replace the standard "*** You have new mail" banner with
  anything you like.  The parameters for this function are:
    $0  The number of new mail messages
    $1  Total number of mail messages
  On startup, new mail messages will be all messages found.
  After that, for example if you had 2 messages and a new
  message came in, new messages would be 1.  Here are a few
  examples of how to use this:
    /ON ^MAIL 1 /echo *** You have $0 new mail message, total of $1
    /ON ^MAIL * /echo *** You have $0 new mail messages, total of $1
  These will display the number of new messages that arrive.
  Here is another example:
    /ON ^MAIL * /comment
  This will disable the display of any message when you have
  mail.  Setting the MAIL variable to OFF will also have this
  effect, but this will also disable the total mail messages
  count in the status line.
```

1.223 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]MODE_CHANGE [-|^]<nick> [action]
  This is activated whenever someone on a channel you are on
  changes the mode for that channel (see HELP MODE).  The
  parameters for this function are:
    $0  The nickname of the person who changed the mode
    $1  The channel on which the mode was changed
    $2- The new mode settings for that channel
```

1.224 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]MSG [-|^]<nicknames> [action]
  This is activated whenever you receive a MSG from someone.
  The parameters for the action are as follows:
    $0    nickname of person who sent the MSG
    $1-   The message

  Warning: If you attempt to send a MSG from within a MSG
  action, it will be automatically converted to a NOTICE, as
  this will prevent MSG loops from being sent between two or
  more users.  Also, you will not be permitted to send either
  MSGs or NOTICEs from within a NOTICE action.  The irc protocol
  states that NOTICEs may not generate automatic replies so this
  will prevent loops of automatic messages from flooding the net.
```

1.225 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]MSG_GROUP [-|^]<parameters> [action]
  This is activated whenever you receive a message sent to a
  "group" of people, such as a message from an operator to
  everyone on a given server.
  Parameters:
        $0      Sender
        $1      Group being sent to
        $2-     Text of message
```

1.226 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]NAMES [-|^]<paramters> [action]
  When /NAMES is activated any outputline will be modified
  by [action].
  For example:
   /ON NAMES * /echo $0 $1-

  The parameters are as follows:
    $0    name of the channel
    $1    all users on this channel
```

1.227 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]NICKNAME [-|^]<nick> [action]
  This is activated whenever someone on a channel you are on
  changes their nickname. The parameters for this function are:
    $0  Original nickname for that person
    $1  New nickname for that person
```

1.228 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]NOTE [-|^]<nicknames> [action]
  This is activated whenever you receive an irc NOTE.
  The parameters for the action are as follows:
    $0    nickname of person who sent the note
    $1    The user@hostname of the person who sent the note
    $2    The note flags (see HELP NOTE FLAGS)
    $3-7  The time the note was queued.
    $8    The server from which the note was queued
    $9-   The message itself
```

1.229 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]NOTICE [-|^]<nicknames> [action]
  This is activated whenever you receive a NOTICE from someone.
  The parameters for the action are as follows:
    $0    nickname of person who sent the NOTICE
    $1-   The message

  Warning: If you attempt to send a MSG from within a MSG
  action, it will be automatically converted to a NOTICE, as
  this will prevent MSG loops from being sent between two or
  more users.  Also, you will not be permitted to send either
  MSGs or NOTICEs from within a NOTICE action.  The irc protocol
  states that NOTICEs may not generate automatic replies so this
  will prevent loops of automatic messages from flooding the net.
```

1.230 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]NOTIFY_SIGNOFF [-|^]<nicknames> [action]
  This is activated when someone on your notify list (see
  NOTIFY) signs off of irc.  The parameters to the action are:
    $0  Nickname of person who signed off
```

1.231 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]NOTIFY_SIGNON [-|^]<nicknames> [action]
  This is activated when someone on your notify list (see
  NOTIFY) signs onto irc.  The parameters to the action are:
    $0  Nickname of person who signed on
    $1  Channel they signed on to
```

1.232 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]<numeric> [-|^]<number> [action]
  This is activated whenever a numeric message of that number arrives.
  The legal range of values for <numeric> are from 001 to 999.
  Numeric messages are special replies send by the server to the client
  in the raw irc protocol.  They are useful to intercept if you are
  writing a service using IRCII.  For more details about what numbers
  represent what messages from the server, please see the irc server
  source code.
  The parameters for the action are as follows:
    $0    number of numeric message
    $1-   parameters (depends on the numeric message)
```

1.233 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]PUBLIC [-|^]<nicknames> [action]
  This is activated whenever you receive a message from someone
  on a channel which is also one of your current channels.
  The parameters for the action are as follows:
    $0    nickname of person who sent the message
    $1-   the message
```

1.234 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]PUBLIC_MSG [-|^]<nicknames> [action]
  This is activated whenever you receive a message to your channel
  from someone not on that channel.
  The parameters for the action are as follows:
    $0    nickname of person who sent the message
    $1    the channel to which the message was sent
    $2-   the message
```

1.235 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]PUBLIC_NOTICE [-|^]<nicknames> [action]
  This is activated whenever you receive a NOTICE to your channel
  from someone not on that channel.
  The parameters for the action are as follows:
    $0    nickname of person who sent the NOTICE
    $1    the channel to which the NOTICE was sent
    $2-   the message
```

1.236 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]PUBLIC_OTHER [-|^]<nicknames> [action]
  This is activated whenever you receive a message from someone
  on a channel that you are also on which is not one of your
  current channels.
  The parameters for the action are as follows:
    $0    nickname of person who joined the channel
    $1    name of channel message was sent to
    $2-   the message
```

1.237 An ON-command (do an action when something happens)

```
                Usage: ON [+|-|^]RAW_IRC [-|^]<parameters> [action]
  This is activated whenever a message is received from a server which is
  not understood by IRCII. Under normal circumstances this will never
  happen, however if used in conjunction with /QUOTE, this ON function
  can be used to test new features being added to a server. This will
  normally be used in conjunction with /ALIAS

  See Also:

                 ALIAS

                 QUOTE
```

1.238 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]SEND_DCC_CHAT [-|^]<nicknames> [action]
  This is activated whenever you send a DMSG to someone else.
  The parameters for the action are as follows:
    $0    nickname of person to whom you sent the DMSG
    $1-   The message itself
```

1.239 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]SEND_MSG [-|^]<nicknames> [action]
  This is activated whenever you send a MSG to someone else.
  The parameters for the action are as follows:
    $0    nickname of person to whom you sent the MSG
    $1-   The message itself
```

1.240 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]SEND_NOTICE [-|^]<nicknames> [action]
  This is activated whenever you send a NOTICE to someone else.
  The parameters for the action are as follows:
    $0    nickname of person to whom you sent the NOTICE
    $1-   The message itself
```

1.241 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]SEND_PUBLIC [-|^]<nicknames> [action]
  This is activated whenever you send a message to a channel you
  are on.
  The parameters for the action are as follows:
    $0    Name of the channel to which the message is going
    $1-   The message itself
```

1.242 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]SEND_TALK [-|^]<user[@host]> [action]
  This is activated whenever you send a TALK MSG to someone else.
  The parameters for the action are as follows:
    $0    user[@host] of person to whom you sent the DMSG
    $1-   The message itself
```

1.243 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]SERVER_NOTICE [-|^]<text> [action]
  This is activated whenever you receive a message from the server
  as a NOTICE. This is basically anything beginning with "***" which
  is sent to you by the server, and the MOTD.

  The parameters for the action are as follows:
    $0    The first word of the text sent -  probably "***" or
          "MOTD"
    $1-   The message
```

1.244 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]SIGNOFF [-|^]<nick> [action]
  This is activated whenever someone on a channel you are on
  leaves irc.  Normally, the display would show:

  *** Signoff: <nickname> (<reason>)

  but, of course, you can change this.  If <nick> is your own
  nickname, then the action will be taken when you exit
  from irc.  The parameters are as follows:
    $0  Nickname of person who left irc
    $1- Serverprovided reason for the sign-off.
```

1.245 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]TALK [-|^]<user> [action]
  This is activated whenever you receive a TALK message from someone.
  The parameters for the action are as follows:
    $0    user name of person who sent the TALK message
    $1-   The message
```

1.246 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]TIME [-|^]<time> [action]
  This is activated whenever the real-time clock is equal to the
  time specification you gave it.  The time specifications should
  be one of two forms:
  HH:MM[AM|PM]  where 1<=HH<=12 and 0<=MM<=59
  HH:MM   where 0<=HH<=23 and 0<=MM<=59
  which form you use depends on the current setting of
  CLOCK_24HOUR.  Time specification may include wildcards as
  well.  The matching of the real-time clock and the time
  specification is a pure textual match.  Of course, wildcards
  may be used in the time specification.
  The parameters for the action are as follows:
    $0    The time (format depends on CLOCK_24HOUR variable)
```

1.247 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]TOPIC [-|^]"<nickname> channel" [action]
  This is activated whenever someone changes the topic
    $0    nickname of person who changed the topic
    $1    The channel on which the topic was changed
    $2-   The new topic
```

1.248 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]WALL [-|^]<nicknames> [action]
  This is activated whenever you receive a WALL (a global
  message from an irc operator).
  The parameters for the action are as follows:
    $0    nickname of person who sent the WALL
    $1-   The message
```

1.249 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]WALLOP [-|^]<nicknames> [action]
  This is activated whenever you receive a WALLOP (a message
  sent to all operators on irc).
  The parameters for the action are as follows:
        $0      Sender
        $1      + if sender is an oper, - if sender is a normal user,
                S if sender is a server
        $2-     text of message
```

1.250 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]WHO [-|^][<channel>|'Channel'] [action]
  This is activated whenever the reply of a /WHO comes in.
  The parameters for the action are as follows:
    $0    The channel the user is on, or 'Channel' for the /who header line.
    $1    Nickname.
    $2    Status. (Here, Gone, *Operator, [ @ChannelOperator ])
    $3    Username (login name)
    $4    Host (host machine)
    $5-   Full Name or other User Data.
```

1.251 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]WINDOW [-|^]<parameters> [action]
  This is activated whenever text is sent to a matching window.
  This can be used to echo information from one window to a
  channel.  WARNING!!!  If you aren't careful you can get into some
  infinite sending loops!  Use with care!
  Parameters:
  $0  Refnum or name of window
  $1- text of line sent to window
```

1.252 An ON-command (do an action when something happens)

```
Usage: ON [+|-|^]WINDOW_KILL [-|^]<paramters> [action]
  Whenever a window is killed (checked out by parameter * or number
  for the window (see HELP WINDOW)), action is done.
  The parameters are:
    $0    Refnum or name of the window
```

1.253 Set a variable in IRC

```
Usage: SET [-][variable name] [value]
  Sets a specified variable to a given value.  If SET is used
  with no parameters, all variables and their current settings
  are listed.  If SET is used with a variable name and no value,
  that variables current setting is listed.  If a - preceeds
  a variable whose value is a string of text, it sets that
  variable to nothing.

  See HELP SET ? for a list of all variables
```

1.254 Set a variable to set up IRC

```
Usage: SET ALWAYS_SPLIT_BIGGEST [ON|OFF|TOGGLE]
  If ON, all new windows displayed will split in half the largest
  visible window to make room for themselves.  If OFF, the
  current window is always split unless it is too small, then the
  largest is split.
```

1.255 Set a variable to set up IRC

```
Usage: SET AUTO_UNMARK_AWAY [ON|OFF|TOGGLE]
  When ON, you will automatically be unmarked as being away (as
  if you had issued an AWAY with no arguments) if you send a
  message to a channel or send a private message.  When OFF, your
  away status will remain unchanged until you issue the AWAY
  command with no arguments.
```

1.256 Set a variable to set up IRC

```
Usage: SET AUTO_WHOWAS [ON|OFF|TOGGLE]
  If set to ON, this will automatically generate a
    WHOWAS <nickname>
  for any nick command that results in
    <nickname>: No such nickname
  If OFF, you will not be bothered with WHOWASs.
  This variable has no effect for servers before 2.6.
```

1.257 Set a variable to set up IRC

```
              Usage: SET BEEP [ON|OFF|TOGGLE]
  Turns ON or OFF the audibleness of ^G's when received.  If ON,
  ^Gs should beep the terminal.  When OFF, ^G's will be treated
  as other control characters and show up as G in the display.
  The setting of BEEP does not affect the beeps
  generated by the BEEP_ON_MSG variable and the BEEP_WHEN_AWAY
  variable.

  See Also:

                 SET BEEP_ON_MSG

                 SET BEEP_WHEN_AWAY
```

1.258 Set a variable to set up IRC

```
Usage: SET BEEP_MAX [number of beeps]
  This allows you to limit the maximum number of beeps you will hear
  for any given messages.  For example, if you set BEEP_MAX to 3,
  and someone sends a message with more than 10 beeps in it, you
  will only hear the first three.  Setting BEEP_MAX to 0 means
  you will hear all beeps in a message.
```

1.259 Set a variable to set up IRC

```
Usage: SET BEEP_ON_MSG [ALL|NONE|[-]<level> [-] <level> ...]
  This variable lets you specify certain types of messages
  which will cause an audible beep when you receive one.
  The possible message levels are:
    PUBLIC          Channel conversation only
    MSG             MSGs only
    NOTICE          NOTICEs only
    WALL            WALLs only
*    WALLOP          WALLOPs only
    NOTE            NOTE only
    CRAP            Is not used.
  See SET LASTLOG_LEVEL for more on these levels
```

1.260 Set a variable to set up IRC

```
Usage:  SET BEEP_WHEN_AWAY [value]
  This sets the number of beeps that you will hear when you
  receive a message while you are /AWAY.  Setting the value to 0
  means you will hear no beeps.
```

1.261 Set a variable to set up IRC

```
Usage: SET CHANNEL_NAME_WIDTH <value>
  Lets you adjust the amount of space used to display channel
  names in the LIST and NAMES output and in the status line of
  IRCII.  If CHANNEL_NAME_WIDTH is set to 0, no channel width
  formatting is performed and channel names are displayed at
  full width.
```

1.262 Set a variable to set up IRC

```
Usage: SET CLOCK [ON|OFF|TOGGLE]
  Turns ON or OFF the status line clock.  The clock can be set to
  show 24 hour time as well as 12 hour time (See HELP SET
  CLOCK_24HOUR).  You can also set an alarm to warn you of the
  time (See HELP SET CLOCK_ALARM).
```

1.263 Set a variable to set up IRC

```
Usage: SET CLOCK_24HOUR [ON|OFF|TOGGLE]
  When ON, the status line clock will be displayed in 24 hour
  format.  When OFF, the status line clock will be displayed in
  12 hour format with either AM or PM after it.
```

1.264 Set a variable to set up IRC

```
Usage: SET CLOCK_ALARM [time|OFF]
  Sets an alarm clock that will beep and send you a message when
  the time arrives.  The time format may be either 12 hour format
  (with AM or PM) or 24 hour format.  The CLOCK does not have to
  be ON for the alarm to function.
```

1.265 Set a variable to set up IRC

```
Usage: SET CMDCHARS <character set>
  Sets the set of characters which may be used to specify a
  command to IRCII.  You can specify as many characters as you
  wish to be command characters.
```

1.266 Set a variable to set up IRC

```
              Usage: /set COMMAND_MODE ON|OFF

This switch was designed for those who are used to MUDs and computer
freaks who like command oriented environments. It will disable the
command character! Every input of yours will be expected to be a command.
If you want to send something to your current channel or query, you'll
have to type "send <text>" or "'<text>".
"say <text>" will always send to the channel even if you're in a query.
With ":<text>" you get the same effect as with "me <text>".

The "'" and ":" abbreviations are introduced for MUD compatibility and
are only available in COMMAND_MODE.

  See Also:

                SAY

                SEND

                LOAD ACTION

                LOAD MUDLIKE

                ME
```

1.267 Set a variable to set up IRC

```
              Usage: SET CONTINUED_LINE <some text>
  This allows you to change the text which is displayed when a
  line is greater than than width of the screen.  By default,
  this text is a +, and each line greater than the screens width
  will have this prepended to all lines after the first.   You
  may use CONTINUED_LINE in conjunction with INDENT, or you may
  disable the CONTINUED_LINE character by setting it to <EMPTY>.

  See Also:

                SET INDENT
```

1.268 Set a variable to set up IRC

```
Usage: SET DISPLAY [ON|OFF|TOGGLE]
  Turns ON or OFF all output.  It can be useful to SET DISPLAY
  OFF at the beginning of your .ircrc so you don't have to see
  everything it does every time you start irc.  Just remember to
  SET DISPLAY ON again at the end of your .ircrc.  NOTE: Setting
  this variable produces no visible result on the screen!
```

1.269 Set a variable to set up IRC

```
Usage: SET ENCRYPT_PROGRAM <encryption program path>
  Sets the program used to encrypt and decrypt messages.  The
  program selected must take an encryption key as the first
  command line argument to work with IRCII.
```

1.270 Set a variable to set up IRC

```
Usage: SET EXEC_PROTECTION [ON|OFF|TOGGLE]
  If you saw a warning message telling you to read this, beware!
  If  anyone on irc told you that typing "/ON ^MSG * $1-" or
  "/ON ^MSG <nick> $1-" would speed speed things up or otherwise
  make life better for you, they are lying.  The above commands
  allow people to send you MSGs and have them executed as commands
  by your IRCII.  This can be a major security problem, since the
  person who told you to do this can then execute any command you
  could, including EXEC commands.  This would give them control over
  your account while you are on irc.

  Anyway, if you did see this warning, it could be that someone is
  trying to abuse your account.  But, don't worry, the EXEC command
  that it warned you about was not executed.  If you are unsure about
  what is going on, please contact your local irc operator for help.

  If you know what you're doing, you can set EXEC_PROTECTION to OFF
  and EXEC will be allowed within ON commands.  Only do this if
  you are sure you understand what is going on.
```

1.271 Set a variable to set up IRC

```
                Usage: SET FLOOD_AFTER <number of messages>
  Used in conjunction with ON FLOOD, this lets you specify the
  number of flooding messages you will see before ON FLOOD
  is activated.

  See Also:

                 ON FLOOD

                 SET FLOOD_RATE

                 SET FLOOD_USERS
```

1.272 Set a variable to set up IRC

```
                Usage: SET FLOOD_RATE <messages per second>
  FLOOD_RATE can be set to the number of messages per second you
  wish to  activate flooding. If messages from a user outpace
  FLOOD_RATE for  FLOOD_AFTER number of messages, ON FLOOD is
  activated.  If FLOOD_RATE is larger then FLOOD_AFTER, then you
  will end up seeing at least FLOOD_RATE messages before flood
  activation (If FLOOD_RATE is 5 and FLOOD_AFTER is 3 then you must
  receive at least 5 messages before the flood rate can be 5
  messages per second).

  See Also:

                ON FLOOD

                SET FLOOD_AFTER

                SET FLOOD_USERS
```

1.273 Set a variable to set up IRC

```
               Usage: SET FLOOD_USERS <number of users to monitor>
  FLOOD_USERS sets the maximum number of users you can have
  flood protection from at one time.  If this number is too large,
  you may see performance degradations and the flood protection
  might not work as well as it should.

  See Also:

                ON FLOOD

                SET FLOOD_RATE

                SET FLOOD_AFTER
```

1.274 Set a variable to set up IRC

```
               Usage: SET FLOOD_WARNING [ON|OFF|TOGGLE]
  If ON, you will see a warning when flood protection
  is activated.  This is used in conjuction with:
    /ON ^FLOOD *
  which otherwise will simply cut off flooding messages
  without a warning to you.

  See Also:

                ON FLOOD
```

1.275 Set a variable to set up IRC

```
Usage: SET FULL_STATUS_LINE <ON|OFF|TOGGLE>
  When ON, IRCII will always draw it's status line to fill the
  entire width of the screen.  When OFF, the status line will
  only be as long as it needs to be to accomodate the displayed
  information.
```

1.276 Set a variable to set up IRC

```
Usage: SET HELP_PAGER [ON|OFF|TOGGLE]
  When set to OFF, the HELP function will not use it's built
  in paging mechanism but will use the normal window paging
  (hold mode).  If ON, then help uses it's own pager.
```

1.277 Set a variable to set up IRC

```
Usage: SET HELP_WINDOW [ON|OFF|TOGGLE]
  This will cause the HELP function to create a new window for
  displaying help.  If OFF, the current window will be used.
```

1.278 Set a variable to set up IRC

```
Usage: SET HIDE_PRIVATE_CHANNELS [ON|OFF|TOGGLE]
  When ON, this will suppress display in your status line of channels
  whose mode is +s (secret) or +p (private).  When OFF, channels
  are shown as usual in the status line.
```

1.279 Set a variable to set up IRC

```
                Usage: SET HISTORY <value>
  Sets the size of the command history circular buffer.  Setting
  HISTORY to zero disables the command history functions unless you
  have HISTORY_FILE set.  You must disable HISTORY_FILE to completely
  turn off command history.

  See Also:

                 SET HISTORY_FILE
```

1.280 Set a variable to set up IRC

```
                Usage: SET HISTORY_FILE <path for history file>
  If HISTORY_FILE is set, then the command history is saved in
  this file rather than in memory.  The advantage to this is
  that the size of this file is limited by your disk space and
  doesn't increase the memory usage of IRCII.  The disadavantage
  is that history access might be slower (or it might not).  To
  disable the HISTORY_FILE, set it to <EMPTY>.  To completely
  disable the command history functions, you must also set
  HISTORY to 0.

  See Also:

                 SET HISTORY
```

1.281 Set a variable to set up IRC

```
               Usage: SET HOLD_MODE [ON|OFF|TOGGLE]
  When ON, IRCII will pause at the end of each page of output.
  Output will continue when the TOGGLE_STOP_SCREEN function is
  activated (bound, by default, to cntl-S) or when the
  SEND_LINE function is activated (bound, by default, to the
  Return or Enter key).

  See Also:

                SET STATUS_HOLD
```

1.282 Set a variable to set up IRC

```
Usage: SET HOLD_MODE_MAX <number of lines>
  This specifies the maximum number of lines that will be held
  in HOLD_MODE before the display is automatically restarted.
  If HOLD_MODE_MAX is 0, the number of lines held in HOLD_MODE
  is unlimited.
```

1.283 Set a variable to set up IRC

```
               Usage: SET INDENT [ON|OFF|TOGGLE]
  Setting INDENT to ON will cause lines of text that are longer
  than the screen width to be indented so that all lines after
  the first start underneath the second word in the first line.
  That’s a long sentence, but that’s what it does.  If you also
  have CONTINUED_LINE set, this will pad the CONTINUED_LINE
  characters out to the second word in the first line, unless
  CONTINUED_LINE is longer.  CONTINUED_LINE will not be
  truncated.  To disable the CONTINUED_LINE, set it to <EMPTY>.
  INDENT will never indent beyond 1/3 the width of the screen.

  See Also:

                SET CONTINUED_LINE
```

1.284 Set a variable to set up IRC

```
Usage: SET INPUT_ALIASES [ON|OFF|TOGGLE]
  Turns ON or OFF the expanding of aliases in the inputline,
  .ircrc and LOADed files.  If you set INPUT_ALIASES ON and type:
    /ECHO $S
  you will see your server name echoed to your window.  If it is
  OFF, you will see $S echoed to your window.  If you turn this on
  in your .ircrc, it is usually advisable to do it as one of the
  last lines in your .ircrc, otherwise other occurences of $ in your
  .ircrc may expand out strangely.

  NOTE: If you do turn on INPUT_ALIASES in the .ircrc then the arguement
  expandos ($*, $0, $1, $3-4, etc) will expand to the command line
  arguments passed to IRCII on startup.
```

1.285 Set a variable to set up IRC

```
Usage: SET INPUT_PROMPT <input prompt string>
  Allows you to change the prompt that will be displayed in the
  input line before any text you type.  Normally, there is no
  prompt, but you can specify any string.  To turn off a prompt,
  set INPUT_PROMPT to <empty>.
```

1.286 Set a variable to set up IRC

```
Usage: SET INSERT_MODE [ON|OFF|TOGGLE]
  Turns ON or OFF insert mode.  While on, characters typed are
  inserted into already existing text.  When off, new characters
  overwrite existing ones.  The TAB key toggles this variable.
```

1.287 Set a variable to set up IRC

```
Usage: SET INVERSE_VIDEO [ON|OFF|TOGGLE]
  When OFF, inverse video sent to your display by using the ^B
  character will not show up in inverse.  This will not affect
  the status line or input line.
```

1.288 Set a variable to set up IRC

```
Usage: SET LASTLOG <value>
  Sets the size of the lastlog buffer (See HELP LASTLOG).  This
  buffer keeps an in memory record of messages sent and received.
  If LASTLOG is set to 0, the lastlog function is disabled.
```

1.289 Set a variable to set up IRC

```
Usage: SET LASTLOG_LEVEL [ALL|NONE|[-]<level> [-]<level> ...]
  The setting of this variable determines which types of
  messages are stored in the lastlog.  If ALL is specified,
  everything that shows up on the screen is saved in the
  lastlog.  If NONE is specified, nothing is saved in the
  lastlog.  The <level> specification can be one of the
  following:
    PUBLIC          Channel conversation
    MSGS            Private messages
    NOTICES         NOTICEs
    WALLS           WALLs
*    WALLOPS         WALLOPs
    NOTE            NOTEs
*    OPNOTES         Operator notifications
    SNOTES          Server notices
    ACTIONS         In and outgoing CTCP ACTION lines
    USERLOG1        Reserved for the user
    USERLOG2        Reserved for the user
    USERLOG3        Reserved for the user
    USERLOG4        Reserved for the user
    CRAP            Anything not covered by the above categories
  You can combine these on a command line to make the lastlog
  save just what you want.  Also, by putting a - before any
  level, you remove that one from the list.  For example, if you
  wish to save everything except NOTEs and all that CRAP, you
  could do:
    SET LASTLOG_LEVEL ALL -NOTE -CRAP
  which is the same as doing:
-    SET LASTLOG_LEVEL PUBLIC MSG NOTICE WALL ACTIONS
*    SET LASTLOG_LEVEL PUBLIC MSG NOTICE WALL WALLOP OPNOTES ACTIONS
```

1.290 Set a variable to set up IRC

```
Usage: SET LOG [ON|OFF|TOGGLE]
  Turns the session log ON or OFF.  While ON, a record of your
  IRC session is recorded to file (see HELP SET LOGFILE).  When
  OFF, no log is made.  This log will contain everything that
  appears on your screen, no matter what window, in the order it
  comes in.  To log an individual window, use the:
    WINDOW LOG [ON|OFF|TOGGLE]
  command.  See HELP WINDOW.
```

1.291 Set a variable to set up IRC

```
                Usage: SET LOGFILE <filename>
  Sets the name of the file to be used for the session log.  New
  session log messages are appended to the end of this file.
  See Also:

                 SET LOG
```

1.292 Set a variable to set up IRC

```
                Usage: SET MAIL [0|1|2]
  When non zero, you will be informed when you have new mail.  This
  is your real unix mail and not irc-mail.  An indicator will light
```

```
up in your status line and a message will be sent to your screen
telling you about the new mail. If set to 2, the more significant
headers from the mail are shown.

See Also:

               ON MAIL

               SET STATUS_MAIL
```

1.293 Set a variable to set up IRC

```
              Usage: SET MINIMUM_SERVERS <number of servers>
When connecting to a server, this variable is checked versus
the number of servers currently connected to the ircnet.  If
the number of servers is less than the setting of
MINIMUM_SERVERS, you will automatically be disconnected from
that server and IRCII will attempt to connect to the next
server in you server list (as though you had done /SERVER +).
This is useful if the net is fragmented to keep you on a more
populated portion of the net.  This will be rechecked if you
issue a /LUSERS.  This will only affect your primary server.
Secondary servers are not checked vs this variable.

See Also:

               SET MINIMUM_USERS

               SERVER

               LUSERS

               IRCII SERVER_LIST
```

1.294 Set a variable to set up IRC

```
              Usage: SET MINIMUM_USERS <number of users>
When connecting to a server, this variable is checked versus
the number of users currently connected to the ircnet.  If
the number of users is less than the setting of
MINIMUM_USERS, you will automatically be disconnected from
that server and IRCII will attempt to connect to the next
server in you server list (as though you had sone /SERVER +).
This is useful if the net is fragmented to keep you on a more
populated portion of the net.  This will be rechecked if you
issue a /LUSERS.  This only affects your primary server.
Secondary servers are not affected by this variable.

See Also:
   SET MINIMUM_SERVERS

               SERVER
```

```
                 LUSERS

                 IRCII SERVER_LIST
```

1.295 Set a variable to set up IRC

```
Usage: SET NOTIFY_ON_TERMINATION [ON|OFF|TOGGLE]
  When ON, all EXEC'ed processes will inform you when they exit.
  You will be told which process is exiting and the reason it
  exited.  If termination was normal, you will receive the exit
  status code.  If termination was by signal, you will be told
  which signal.  If OFF, you will not be informed when processes
  exit.
```

1.296 Set a variable to set up IRC

```
                Usage: SET NOVICE [ON|OFF|TOGGLE]

The NOVICE variable causes IRCII to disallow certain actions by the
user (such as ON commands) and to show a request to read the NEWUSER
help file on startup. It is set to ON by default, and should normally
be turned off in your .ircrc file

  See Also:

                 ON

                 NEWUSER
```

1.297 Set a variable to set up IRC

```
Usage: SET PAUSE_AFTER_MOTD [ON|OFF|TOGGLE]
  When ON, this will cause IRCII to wait for a keystroke after
  the local IRCII motd is displayed then clear the screen and
  continue.  If OFF, the local motd will be displayed,
  immediately followed by the server banners and motds.  If
  there is no local IRCII motd, this variable has no effect.
  PAUSE_AFTER_MOTD only works for the IRCII motds and *not* for
  server motds.
```

1.298 Set a variable to set up IRC

```
Usage: SET SCROLL [ON|OFF|TOGGLE]
  Turns ON or OFF screen scrolling.  While OFF, when the cursor
  reaches the bottom of the screen, it jumps to the top and
  overwrites its way back down.  This mode is recommended for
  *extremely* dumb terminals.  When ON, the screen will scroll
  old information off the top.
```

1.299 Set a variable to set up IRC

```
Usage: SET SCROLL_LINES [number of screen lines to scroll]
  This allows you to set the number of lines the screen will
  scroll each time the cursor reaches the bottom of the screen.
  If SCROLL_LINES is set to 0, the SCROLL is turned OFF as well,
  and must be turned back on before scrolling can be resumed.
  The maximum number of lines that may be scrolled is the size of
  the display (the number of lines on the screen minus 2).
  Negative numbers are automatically changed to positive.
```

1.300 Set a variable to set up IRC

```
Usage: SET SEND_IGNORE_MSG [ON|OFF|TOGGLE]
  When OFF, prevents the sending of "You are being ignored"
  messages when you /IGNORE someone.  When ON, these messages are
  sent when then /IGNORE is initiated for PRIVATE or ALL
  messages.
```

1.301 Set a variable to set up IRC

```
                Usage: SET SHELL <shell path>
  Sets the name of the shell to be used by the EXEC command.
  Normally, this would be a standard csh or sh, but you can make
  it anything you like.
  See Also:

                 SET SHELL_FLAGS

                 SET SHELL_LIMIT

                 EXEC
```

1.302 Set a variable to set up IRC

```
Usage: SET SHELL_FLAGS <flags>
  Sets any additional flags that the shell (set with SET SHELL)
  might need.
```

1.303 Set a variable to set up IRC

```
Usage: SET SHELL_LIMIT <value>
  Sets the maximum number of lines of output from any EXEC'd
  process.  This is useful to prevent yourself from accidentally
  starting a process with EXEC that spits out soooo much output
  that everything you know and love grinds to a halt.  Setting
  it to 0 puts no limit on the number of output lines from a
  process.
```

1.304 Set a variable to set up IRC

```
Usage: SET SHOW_CHANNEL_NAMES [ON|OFF|TOGGLE]
  When ON, this will show the names of all the users on a channel when
  you join it.  When OFF, no names will be displayed.
```

1.305 Set a variable to set up IRC

```
Usage: SET SHOW_END_OF_MSGS [ON|OFF|TOGGLE]
  When ON, tells IRC II to display the "End of list" messages for
  the /NAMES, /LIST, /LINKS, and other commands.  When OFF, no
  "End of list" messages are displayed.
```

1.306 Set a variable to set up IRC

```
Usage: SET SHOW_NUMERICS [ON|OFF|TOGGLE]
  If ON, then any numeric protocol messages from the server will
  have their corresponding number shown on the line.
```

1.307 Set a variable to set up IRC

```
Usage: SET STATUS_AWAY <Status line text when in away>
  The contents of STATUS_AWAY are replaced in the STATUS_FORMAT
  variable for any occurence of %A while you are away.  If you
  are not away, %A is replaced by nothing.
```

1.308 Set a variable to set up IRC

```
Usage: SET STATUS_CHANNEL <Status line text for display of channel>
  The contents of STATUS_CHANNEL are replaced in the
  STATUS_FORMAT variable for any occurence of %C.
  This variable may contain any text, plus it may contain
  a single occurence of %C, which is replaced by the name or
  the number of the channel you are currently on.
```

1.309 Set a variable to set up IRC

```
Usage: SET STATUS_CLOCK <text of display for status line clock>
  The value of STATUS_CLOCK is replaced in the STATUS_FORMAT
  variable for any occurence of %T if CLOCK is ON.  If CLOCK is
  OFF, nothing is displayed for %T in STATUS_FORMAT.  Also, the
  STATUS_CLOCK variable may contain one occurence of %T which
  will be replaced with the current time of day in the string.
```

1.310 Set a variable to set up IRC

```
Usage: SET STATUS_DQUERY <Status line text when DCC querying>
  This variable is replaced in the STATUS_FORMAT for occrunces of
  %Q when you are DCC querying someone.  This variable may contain
  any text, plus it may contain a single occurence of %D, which
  is replaced by the name of the user you are DCC querying.  For
  example, you can set STATUS_DQUERY to "DQuerying %D", where %D
  is replaced by a the DCC query nickname.
```

1.311 Set a variable to set up IRC

```
                Usage: SET STATUS_FORMAT <format description for status line>
  Setting the variable allows you to alter the appearance of the
  status line.  Certain special characters are recognized in
  this format line and replaced with various bits of status line
  information.  Each of these special characters is preceeded by
  a ’%’.  Here is a list of these characters:
       N          Your current nickname.
       C          Your current channel.
       R          Current window refnum or name.
       W          Value of STATUS_WINDOW variable. *
       +          Value of STATUS_MODE variable. *
       Q          Value of STATUS_QUERY variable. *
       I          Value of STATUS_INSERT variable. *
       S          Value of STATUS_SERVER variable. *
       O          Value of STATUS_OVERWRITE variables. *
       A          Value of STATUS_AWAY variable. *
       T          Value of STATUS_CLOCK variable. *
       U          Value of STATUS_USER variable. *
       H          Value of STATUS_HOLD variable. *
       *          Value of STATUS_OPER variable. *
       M          Value of STATUS_MAIL variable. *
       V          Current IRCII version
       %          Replaced by %
  Those marked with * are only displayed in the status line when
  certain conditions are met.  Please get help on the variables
  mentioned above for specifics about each one.

  For example, the standard default IRCII status line format
  looks like this:

  *** IRCII: /HELP for help  %H%N%*  Channel %C%Q (%I%O) %A%T***
  The %H is replaced by the value of STATUS_HOLD if display
  output is stopped (See SET HOLD_MODE), otherwise nothing is
  displayed.  %N is replaced by your current nickname.  %* is
  replaced buy the value of STATUS_OPER if you have operator
  status.  %C is replaced by your current channel.  %Q is
  replaced by the value of STATUS_QUERY if you are querying
  someone, otherwise nothing is displayed.  %I is replaced by the
  value of STATUS_INSERT if you are in insert mode.  %O is
  replaced by the value of STATUS_OVERWRITE if you are not in
  insert mode.  %A is replaced by the contents of STATUS_AWAY if
  you are away.  %T is replaced by the current time of day of the
  variable CLOCK is ON.

  See Also:

                SET STATUS_WINDOW

                SET STATUS_MODE
                    SET STATUS_QUERY

                SET STATUS_INSERT
                    SET STATUS_OVERWRITE

                SET STATUS_AWAY

                SET STATUS_CLOCK

                SET STATUS_HOLD

                SET STATUS_SERVER

                SET STATUS_OPER

                SET STATUS_USER

                SET STATUS_MAIL

                SET STATUS_NOTIFY

                SET HOLD_MODE

                SET CLOCK
```

1.312 Set a variable to set up IRC

```
Usage: SET STATUS_HOLD <text of display for status line hold mode>
  The value of STATUS_HOLD is replaced in the STATUS_FORMAT
  variable for any occurence of %H if the display is current on
  hold (See SET HOLD_MODE).  If the display is not on hold, %H
  is replaced with nothing.
```

1.313 Set a variable to set up IRC

```
Usage: SET STATUS_INSERT <Status line text when in insert mode>
  The content of STATUS_INSERT are replaced in the STATUS_FORMAT
  variable for any occurence of %I while INSERT_MODE is ON.  If
  INSERT_MODE is OFF, nothing is displayed for %I.
```

1.314 Set a variable to set up IRC

```
Usage: SET STATUS_MAIL <Status line text when you have mail>
  The contents of STATUS_MAIL are replaced in the STATUS_FORMAT
  variable for any occurence of %M while you get new mail.  The
  STATUS_MAIL may contain occurences of %M which will be replaced
  by the total number of mail messages you have.  If you have
  no mail, %M is replaced by nothing in the STATUS_FORMAT.
```

1.315 Set a variable to set up IRC

```
Usage: SET STATUS_MODE <text displayed for channel mode>
  The value of STATUS_MODE is replaced in the STATUS_FORMAT
  variable for an occurence of %+ if the window's current
  channel has any of the mode settings active.  The STATUS_MODE
  variable may contain one occurence of %+ which will be
  replaced with the current channel mode (if applicable).
```

1.316 Set a variable to set up IRC

```
Usage: SET STATUS_NOTIFY <Status line text>
  The contents of STATUS_NOTIFY are replaced in the STATUS_FORMAT
  variable for any occurence of %F when there has been activity in
  a hidden window. STATUS_NOTIFY may contain occurences of %F which
  will be replaced by the list of windows in which there has been
  activity. If there has been no activvity in hidden windows, %F
  is replaced by nothing in the STATUS_FORMAT.
```

1.317 Set a variable to set up IRC

```
Usage: SET STATUS_OPER <text of display for status line operator mode>
  The value of STATUS_OPER is replaced in the STATUS_FORMAT
  variable for any occurence of %* if you have operator status.
  If you are not an operator, %* is replaced with nothing.
Usage: SET STATUS_OVERWRITE <Status line text when in overwrite mode>
  The content of STATUS_OVERWRITE are replaced in the STATUS_FORMAT
  variable for any occurence of %O while INSERT_MODE is OFF.  If
  INSERT_MODE is ON, nothing is displayed for %O.
```

1.318 Set a variable to set up IRC

```
Usage: SET STATUS_OVERWRITE <Status line text when in overwrite mode>
  The content of STATUS_OVERWRITE are replaced in the STATUS_FORMAT
  variable for any occurence of %O while INSERT_MODE is OFF.  If
  INSERT_MODE is ON, nothing is displayed for %O.
```

1.319 Set a variable to set up IRC

```
Usage: SET STATUS_QUERY <Status line text when querying>
  This variable is replaced in the STATUS_FORMAT for occrunces of
  %Q when you are querying someone.  This variable may contain
  any text, plus it may contain a single occurence of %Q, which
  is replaced by the name of the user you are querying.  For
  example, you can set STATUS_QUERY to "Querying %Q", where %Q
  is replaced by a the query nickname.
```

1.320 Set a variable to set up IRC

```
Usage:  SET STATUS_SERVER <Status line text for display of server>
  The value of STATUS_SERVER is replaced in the STATUS_FORMAT
  variable for any occurence of %S.
  This variable may contain any text, plus it may contain
  a single occurence of %S , which is replaced by the server
  you are currently on.
```

1.321 Set a variable to set up IRC

```
Usage: SET STATUS_USER <text of status line user information>
  The value of STATUS_USER is replaced in the STATUS_FORMAT
  variable for any occurence of %U. If STATUS_USER is set to
  <EMPTY>, nothing will be replaced for %U.
```

1.322 Set a variable to set up IRC

```
Usage: SET STATUS_WINDOW <text displayed in current window status line>
  The value of STATUS_WINDOW is replaced in the STATUS_FORMAT
  variable for an occurence of %W if the window is the current
  window.  If the window is not the current window (or if there
  is only one window visible), then %W is replaced with nothing.
```

1.323 Set a variable to set up IRC

```
Usage: SET SUPPRESS_SERVER_MOTD [ON|OFF|TOGGLE]
  This will prevent you from seeing the motd from the server when you
  connect to a server.  This is not to be confused with the IRCII motd
  which your installation may or may not have.  You will still see the
  server motd if you use the /MOTD command.
```

1.324 Set a variable to set up IRC

```
Usage: SET WARN_OF_IGNORES [ON|OFF|TOGGLE]
  Turns OFF or ON warning messages that occur when you try to
  send a message to a nickname that you are ignoring.  When ON,
  you get a warning each time you attempt this.  When OFF, you
  get no warning.
```