CygnusEd_Manual

Bruce Dawson

COLLABORATORS			
	TITLE : CygnusEd_Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Bruce Dawson	January 2, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

CygnusEd_Manual

1.1 CygnusEd_Manual.guide

CygnusEd Professional Online Manual
Copyright © 1987-1999 CygnusSoft Software
Written by Bruce Dawson, updated by Olaf Barthel

Welcome to the CygnusEd Professional online documentation. CygnusEd Professional is a powerful and fast text editor, designed to be easy to learn just by looking through the menus. However, even the best designed program has some hidden or non-obvious features, and CygnusEd's menus are large enough that it's easy to miss some important features.

This documentation is setup so that it can easily be used as an introduction to CygnusEd, or as a reference. All of the buttons in the block below contain important information about how CygnusEd works, what the important features are, and what the philosophy behind these features is. When learning CygnusEd it is recommended that you at least glance at all of these sections, to see what CygnusEd has to offer, and then read in detail those that interest you.

- expand CygnusEd through ARexx

ARexx reference
- A detailed reference of all ARexx command

The menu buttons are best used as a reference when you have a question on how a specific menu item works. All of CygnusEd’s menu commands are documented here.

The following sections describe features that are not covered by ↩ the menu documentation:

1.2 Default icon

Default icon

When saving icons with files, CygnusEd will use its built-in default project icon. You can supply your own icon, just create a drawer "ENVARC:CED" and copy the project icon file under the name "def_CEDFile.info" into it.

1.3 Setting the tabulator size

Setting the tabulator size

The "File" environment menu is for setting the size of tabulator stops, but this can take place only after a file has been loaded or just before you load a new file. There is an alternative way to specify the tabulator size to use when editing a file. CygnusEd will scan the first ten lines of a file just loaded for a special keyword. For example, to set the tabulator size to four spaces, add the following line to a text file:

```
:ts=4
```

Now open this text file; CygnusEd will automatically adjust the tabulator size after loading the file. ":ts=" is actually an abbreviation for ":tabsize=".

1.4 Keyboard operations

Keyboard operations

Depending on what qualifier keys ([Shift], [Alt], [Ctrl], [Caps Lock]) are being held, a number of keys trigger different actions.

[Return] key:

No [Shift]/[Ctrl] key held

Breaks the line at the cursor position and puts the cursor at the beginning of the next line.

With [Shift] key held

Adds a new, auto-indented line below the current line and puts the cursor there. This *DOES NOT* break the current line. The auto-indentation is accomplished by duplicating all the leading blank space characters in the current line.

With [Alt] key held

Breaks the line at the cursor position and adds a new, auto-indented line below the current line.

With [Ctrl] key held

Breaks the line at the cursor position and inserts a carriage return character.

Function keys with the [Caps Lock] key held insert printer control codes and the current time of day and date:

	Set attribute	Clear attribute	Set foreground
Qualifier	F1	F2	F3
[Ctrl]	Underline	Underline	Colour 3
[Shift]	Italic	Italic	Colour 2
[Alt]	Bold	Bold	Colour 1
None	Clear all	Clear all	Colour 0

	Set background	Set pitch	Clear pitch
Qualifier	F4	F5	F6
[Ctrl]	Colour 3	Enlarged	Enlarged

[Shift]	Colour 2	Condensed	Condensed
[Alt]	Colour 1	Elite	Elite
None	Colour 0	Normal	Normal

	Set pitch	Clear pitch	Miscellaneous
Qualifier	F7	F8	F9
[Ctrl]	Super	Super	1/6 lines
[Shift]	NLQ	NLQ	1/8 lines
[Alt]	Doublestrike	Doublestrike	Proportional off
None	Shadow	Shadow	Proportional on

	Time and date
Qualifier	F10
[Ctrl]	-
[Shift]	Date
[Alt]	Time
None	Time and date

Any combination of qualifier keys is allowed.

1.5 Mouse operations

Mouse operations

CygnusEd allows you to use the mouse for moving the text cursor, text scrolling and for marking text blocks. Movement is accomplished by holding down the left mouse button and dragging the mouse. To mark text, you can use two different techniques:

1) Double-click the left mouse button; this will toggle block marker mode.

2) Hold down either [Shift] key and drag the mouse; this will start marking a text block unless you are already in block marker mode.

1.6 Workbench support

Workbench support

This support comes in two forms: through icons CygnusEd can store along with the files you save, through the Workbench 'Tools' menu and CygnusEd's window.

Icons

Workbench 'Tools' menu

CygnusEd window

1.7 Icons

Icons

CygnusEd can store icons with every single file it saves to disk (see Icon Creation?). Other than making these files 'visible' from Workbench, CygnusEd stores state information in these icons as follows:

TABSTOPS

These are the tabulator settings, as available through the Customize tabs... command.

MARK

This is a list of line/column pairs the text marks are placed at (see Mark/Jump Location).

CURSOR

This sets the file's cursor position. CygnusEd will place the text cursor there after loading the file (instead of placing the cursor at the beginning of the document).

1.8 Workbench 'Tools' menu

Workbench 'Tools' menu

If the Workbench is active when CygnusEd is started (note that this is not the case if you launch CygnusEd from your 'S:User-Startup' script) a menu item 'Open with CygnusEd' will be added to the Workbench 'Tools' menu. To open files with CygnusEd, select their icons, then pick this menu item. Note that for each new file a new view will have to be created. The number of views is limited, so CygnusEd may not open all the files you selected.

1.9 CygnusEd window

The CygnusEd window

If opened on the Workbench screen, the CygnusEd window acts as a so-called 'AppWindow', i.e. you can drag icons into it. Depending on whether you hold down any [Shift] or [Ctrl] key different actions will follow:

No [Shift] or [Ctrl] key held:

CygnusEd will try to load the file into a new view.

Any [Shift] key held:

The file will be included at the position you let go of the left mouse button.

[Ctrl] key held:

The name(s) of the file(s) will be inserted at the current cursor position. If more than a single file is given, their names will be separated by line feeds.

1.10 Project menu

Project menu

The commands in this menu are, as you would expect, mostly to do with files, projects, printing and other similar operations.

About...

Clear

Open new - Keyboard shortcut [Amiga]+?

Open... - Keyboard shortcut [Amiga]+o

Include file... - Keyboard shortcut [Amiga]+O

Save clip as... - Keyboard shortcut [Amiga]+n

Save - Keyboard shortcut [Amiga]+w

Save as... - Keyboard shortcut [Amiga]+W

Save all changes

Change current directory...

Spawn new CED [Ctrl]+Esc

Activate next CED [Ctrl]+N

Print clip... - Keyboard shortcut [Amiga]+p

Print file... - Keyboard shortcut [Amiga]+P

Quit - Keyboard shortcut [Amiga]+q

Save & quit - Keyboard shortcut [Amiga]+Q

Quit & die

1.11 Environment menu

Environment menu

The Environment menu is a uniquely CygnusEd three part menu that contains all of the settings for how CygnusEd operates.

Global Settings

File Settings

View Settings

1.12 Global Settings

Global Settings

The ’Global Settings’ menu is the first part of the peculiar three part ’Environment’ menu. The ’Global Settings’ menu contains all of those settings which affect all of the files and views in CygnusEd. This includes such things as screen resolution, macro definitions, fonts, etc. These settings are stored in your ceddefaults file if you select ’Save Environment’, at the bottom of this menu. When you first run CygnusEd the ’ceddefaults’ file is loaded in with global, file and view settings. Whenever a new file is loaded in, new extension specific settings from the ’File Settings’ menu will be loaded in if a ceddefaults file with the same extension (for example, ceddefaults.c, or ceddefaults.txt) is found. However the global defaults are only loaded once, or if you explicitly load them. Therefore it is important that when you change any of the global settings that you save those settings in the ceddefaults file (no extension) using the ’Save Environment’ command.

Open on public screen...

Set screen type and size...

Duplicate Workbench size

Macro definitions

Begin short invocation macro - Keyboard shortcut [Amiga]+m

Begin long invocation macro - Keyboard shortcut [Amiga]+M

Define startup macro

Quote key

Clear definitions

Load definitions... - Keyboard shortcut [Amiga]+;

Save definitions... - Keyboard shortcut [Amiga]+'

Priority

Autosave

Set icon tool name...

Colours

File save method

Icon creation?

Hot-Start enabled?

Auto-expand views? - Keyboard shortcut [Amiga]+[

Keypad = movement? - Keyboard shortcut [Amiga]+8

Select font

Rendering choices

Load environment

Save environment... - Keyboard shortcut [Amiga]+E

1.13 File Settings

File Settings

The 'File Settings' menu contains settings that are kept separately for each file loaded into CygnusEd. These settings are stored on disk in separate ceddefault files with each default file having a different extension. If you load a file with a '.c' extension then CygnusEd looks for a file called 'ceddefaults.c', in the current directory and then in S:. If it finds such a file then it automatically loads in all of your file and view settings from that settings file and applies them to the current view. This allows you to easily have different tab sizes and other settings for text files versus C source files.

Tab size

Customize tabs... - Keyboard shortcut [Amiga]+t

Set right border... - Keyboard shortcut [Amiga]+^

Set scroll jump

Max scroll   xx...

Layout? - Keyboard shortcut [Amiga]+5

Word wrap? - Keyboard shortcut [Amiga]+6

Insert mode? - Keyboard shortcut [Amiga]+7

Tabs = spaces? - Keyboard shortcut [Amiga]+0

Editable file?

1.14 View Settings

View Settings

CygnusEd allows you to have multiple cooperating views on a single file. All of these views are windows into the same data. Changes in one view are reflected in the others. In fact, if several views are displaying the same area of a particular file, you can see changes happening in all of the views simultaneously. The settings in this menu are stored separately for different views of a particular file. This is occasionally useful.

Status line

White spaces

Scroll bar

Set scroll borders...

Views overview

1.15 Special menu

Special menu

The Special menu is, as the name suggests, a grab bag of miscellaneous functions that don’t fit into any of the other neat categories.

View operations

Previous view - Keyboard shortcut [Amiga]+,

Next view - Keyboard shortcut [Amiga]+.

```
                 Split view - Keyboard shortcut [Amiga]+d

                 Expand view - Keyboard shortcut [Amiga]+]

                 Grow view [Ctrl]+[Alt]+Up

                 Shrink view [Ctrl]+[Alt]+Down

                 Format

                 Post period spaces...

                 Send DOS/ARexx command...

                 Install DOS/ARexx command...

                 Load DOS/ARexx commands...

                 Save DOS/ARexx commands...

                 Send DOS/ARexx output to...

                 Enter ASCII... - Keyboard shortcut [Amiga]+-

                 Center cursor - Keyboard shortcut [Amiga]+=

                 Center line - Keyboard shortcut [Amiga]+\

                 Repeat key/menu... - Keyboard shortcut [Amiga]+''

                 Find matching bracket - Keyboard shortcut [Amiga]+h

                 Mark/Jump Location
```

1.16 Cut/Paste menu

```
                                             Cut/Paste menu

  The Cut/Paste menu is primarily concerned with various ways of cutting
out part of file and, optionally, pasting it in somewhere else. CygnusEd has
the fairly unique feature of having many different cut and paste buffers.
The standard cut/copy/paste commands can use any of the Amiga clipboards 255
different units, and the word and line specific cut and paste commands each
have their own clip buffers also.


                 Mark - Keyboard shortcut [Amiga]+b

                 Mark columnar - Keyboard shortcut [Amiga]+B

                 Cut - Keyboard shortcut [Amiga]+x

                 Copy - Keyboard shortcut [Amiga]+c
```

Paste - Keyboard shortcut [Amiga]+v

Erase

Mark word

Set clipboard unit...

Rot marked

Strip CR marked

Shift in marked

Shift out marked

Change marked spaces to tabs

Change marked tabs to spaces

Change marked to lower case

Change case marked

Change marked to upper case

Delete word [Ctrl]+Del

Undelete word [Ctrl]+[Alt]+Del

Bck Spc word [Ctrl]+BckSpc

UnBck Spc word [Ctrl]+[Alt]+BckSpc

Delete line - Keyboard shortcut [Amiga]+k

Delete to EOL - Keyboard shortcut [Amiga]+y

Undelete line - Keyboard shortcut [Amiga]+l

1.17 Search/Replace menu

Search/Replace menu

The Search/Replace menu primarily contains commands to invoke and control CygnusEd's high-speed searching and replacing. In addition this menu contains commands for changing the case of words and letters, and controlling CygnusEd's undo ability.

Repeat search backwards - Keyboard shortcut [Amiga]+a

Repeat search forwards - Keyboard shortcut [Amiga]+s

Search for... - Keyboard shortcut [Amiga]+S

Repeat replace - Keyboard shortcut [Amiga]+r

Replace... - Keyboard shortcut [Amiga]+R

Clip to search buffer

Clip to replace buffer

Set ASCII zero alias for search...

Change case letter - Keyboard shortcut [Amiga]+g

Change case word - Keyboard shortcut [Amiga]+G

Upper case word

Lower case word

Undo

1.18 Move menu

Move menu

CygnusEd's move menu contains all of the cursor movement commands. Most of the menu items, the bottom fifteen, are there purely as a place to put the keyboard shortcuts for the cursor movement commands, as the ultimate in online documentation. You can select these commands from the menu with the mouse, but few people actually do.

Jump to line... - Keyboard shortcut [Amiga]+j

Jump to auto-mark - Keyboard shortcut [Amiga]+4

Jump to byte... - Keyboard shortcut [Amiga]+J

Jump to last change

Cursor key movement

1.19 About...

About...

The 'About' menu command brings up an informative requester telling you what version of CygnusEd you are running.

1.20 Clear

Clear

The 'Clear' command clears out the current file, including it's name. All views of the current file will be cleared. If you have made any changes then you will be asked to confirm the clear before proceeding. You can undo this operation, but the file name is permanently lost.

1.21 Open new - Keyboard shortcut [Amiga]+?

Open new - Keyboard shortcut [Amiga]+?

The 'Open New' command creates a new, empty file buffer. If the current file has multiple views on it then the current view will be used for the new file buffer. If the current file has just one view on it, then a new view will be created for it.

1.22 Open... - Keyboard shortcut [Amiga]+o

Open... - Keyboard shortcut [Amiga]+o

The 'Open' command is used to load a new file from disk into CygnusEd. The file is always loaded into the current view - a new view is not created for the new file. If the current file has multiple views on it then the current view will be used for the new file. If the current file has just one view on it, then the new file will replace the current one. If the current file has been changed then you will be aksed to confirm the overwrite. The standard ASL file requester is used to select the file, or files, to open. If multiple files are chosen then additional views are opened for the additional files.

1.23 Include file... - Keyboard shortcut [Amiga]+O

Include file... - Keyboard shortcut [Amiga]+O

The 'Include file' command is used to insert a file into the current file. Including a file in this way is similar to pasting from the clipboard, except that you are pasting from a file instead of from a clipboard. Note that the keyboard shortcut for this command is [Amiga]+[Shift]+o, which is very similar to the open command's keyboard shortcut of [Amiga]+o. This similarity reflects the fact that the two commands are similar, in that they both load in files. The standard ASL file requester is used to select the file to include.

1.24 Save clip as... - Keyboard shortcut [Amiga]+n

Save clip as... - Keyboard shortcut [Amiga]+n

The 'Save clip as' command saves the current contents of the clipboard as a file on disk. This is very convenient if you want to save part of a file as a new file on disk, or if you want to break a file into two parts. Alternately you can use the 'Open new' command, paste into the new buffer, and save the new buffer with the 'Save' or 'Save as' commands.

1.25 Save - Keyboard shortcut [Amiga]+w

Save - Keyboard shortcut [Amiga]+w

The 'Save' command saves the current file under its current name. If the current file is unnamed, then the standard ASL file requester appears to allow you to select a file name. The save will occur even if no changes have been made to the file.

1.26 Save as... - Keyboard shortcut [Amiga]+W

Save as... - Keyboard shortcut [Amiga]+W

The 'Save as' command brings up the ASL file requester to allow you to save the current file under a new name. The save will occur even if no changes have been made to the file.

1.27 Save all changes

Save all changes

The 'Save all changes' command will save all loaded files that have been changed. Changed files are indicated with an asterisk ('*') in the title bar. The number of changes made since the last save can also be displayed (see the 'Status Line' command in the 'View Settings' menu).

1.28 Change current directory...

Change current directory...

If you make use of CygnusEd's ability to execute DOS commands and ARexx scripts from within CygnusEd then you sometimes need the ability to change CygnusEd's directory, because these commands inherit CygnusEd's current directory. The standard ASL directory requester is used to select a new current directory.

1.29 Spawn new CED [Ctrl]+Esc

Spawn new CED [Ctrl]+Esc

Although CygnusEd can edit multiple files, and have multiple views on each of those files, it sometimes makes sense to make use of the multitasking abilities of the Amiga by running multiple copies of CygnusEd. This can give you extra flexilibity in the arranging of your document windows, even allowing you to have different copies of CygnusEd running on different screens. Additional copies can be run from the CLI, from the workbench, or by selecting 'Spawn new CED'.

1.30 Activate next CED [Ctrl]+N

Activate next CED [Ctrl]+N

If you run multiple copies of CygnusEd you will frequently need to switch between them. This command makes it easy to do this.

1.31 Print clip... - Keyboard shortcut [Amiga]+p

Print clip... - Keyboard shortcut [Amiga]+p

The 'Print clip' command starts up a background process to print the contents of the clipboard to your printer. Because the printing runs in the background you can resume work immediately. However, you cannot exit CygnusEd until printing has completed.

1.32 Print file... - Keyboard shortcut [Amiga]+P

Print file... - Keyboard shortcut [Amiga]+P

The 'Print file' command starts up a background process to print the contents of the current file to your printer. Because the printing runs in the background you can resume work immediately, even editing the document that is being printed, without affecting the results. However, you cannot exit CygnusEd until printing has been completed.

1.33 Quit - Keyboard shortcut [Amiga]+q

Quit - Keyboard shortcut [Amiga]+q

The 'Quit' command does not necessarily tell CygnusEd to completely quit. It would be more accurate to describe it as a 'Close view' command. The current view on the current file is closed. If that is the last view on that file then the file is cleared from memory (after confirmation from the user if there were any changes to the file). If there is only one view open when this command is chosen then then CygnusEd's main window closes also. However, even then CygnusEd may not completely terminate. If you are using the 'Hot Start' feature of CygnusEd, then CygnusEd remains dormant, ready to be invoked by the hot key ([Alt]+[Shift]+Return) or the CygnusEd invoker program, ed.

1.34 Save & quit - Keyboard shortcut [Amiga]+Q

Save & quit - Keyboard shortcut [Amiga]+Q

The 'Save & quit' command is just like the 'Quit' command except that the current file is first saved. Again, this command is best thought of as 'Save & close view' rather than 'Save & quit'.

1.35 Quit & die

Quit & die

If you are using the 'Hot Start' feature of CygnusEd then closing down the last view does not actually terminate the program - unless you use the 'Quite & die' command. This removes CygnusEd from memory after closing the last view.

1.36 Open on public screen...

Open on public screen...

One of the wonderful new features of the AmigaOS 2.0 operating system is public screens. Many programs, such as Mand2000, Art Department Professional, Directory Opus and many others open what are known as 'public screens'. If one of these programs is running when you select this menu item then a requester will appear listing all the currently available public screens that are at least 640 by 200 pixels in size. You can select any of these screens, and suddenly CygnusEd is running on Mand2000's screen, so that you can write letters while you explore fractals! The workbench screen is the one public screen that is always available, so that this command is how you get CygnusEd to open on the workbench screen. If there is only one public screen available, typically the workbench screen, then no requester appears and CygnusEd moves directly to that screen. If you have multiple copies of CygnusEd running and if CygnusEd is setup to declare its own screen to be public then you can open a second copy of CygnusEd on another CygnusEd's screen.

1.37 Set screen type and size...

Set screen type and size...

This command invokes the standard ASL screen mode requester. This requester allows you to select any standard Amiga screen mode, including the new AGA screen modes. It also allows you to pick a wide variety of screen modes that become available if you install one of many third party graphics boards that supply intuition compatible screen modes. Such boards include the Picasso, Retina, EGS Spectrum and many others. CygnusEd is highly compatible with all of these graphics boards.

1.38 Duplicate Workbench size

Duplicate Workbench size

The ’Duplicate Workbench size’ command is provided based on the assumption that as monitors and graphics boards change, people frequently need to change the screen modes that they work with, and that it’s a nuisance to have to change the screen modes for potentially dozens of different programs. This problem can be particularly annoying if the program you are trying to change comes up in a no longer visible screen mode, forcing you to reconfigure it blind. If you set CygnusEd to ’Duplicate Workbench size’ then CygnusEd will always open up in exactly the same screen mode and resolution as you workbench screen, meaning that you will never have to reconfigure it again. CygnusEd will open up on a different screen from the workbench, but this screen will be identical.

1.39 Macro definitions

Macro definitions

CygnusEd features a very powerful macro definition ability that makes it possible to assign any sequence of commands to any key on the keyboard, with no programming required. Creating macros in CygnusEd and remapping the keyboard is so easy that CygnusEd users frequently create ’throwaway’ macros for key sequences that they will only be repeating a couple of times. The basic way of defining macros in CygnusEd is to type [Amiga]+m, the key you want to assign the macro to, do the actions you want recorded (all the commands are processed while you do this) and then stop defining by typing [Amiga]+m again. That’s it.

Begin short invocation macro - Keyboard shortcut [Amiga]+m

Begin long invocation macro - Keyboard shortcut [Amiga]+M

Define startup macro

Quote key

Clear definitions

Load definitions... - Keyboard shortcut [Amiga]+;

Save definitions... - Keyboard shortcut [Amiga]+'

1.40 Begin short invocation macro - Keyboard shortcut [Amiga]+m

Begin short invocation macro - Keyboard shortcut [Amiga]+m

This is the most commonly used command for defining macros. Select this command (or, more commonly, type [Amiga]+m) and then follow the prompts. The first thing you will be prompted for is a key to assign the macro to. This can be a single key, such as F1, 'B', or 'Enter', or it can be a modified key, such as [Ctrl]+U, [Alt]+[Shift]+8, or [Ctrl]+[Alt]+[Shift]+[Amiga]+Tab. Every key on the keyboard, with the exception of the control, alt, shift and Amiga keys, can have a macro assigned to it. Because the Amiga key is a valid qualifier key, you can override the supplied command key shortcuts. Because control, alt and shift are also valid qualifier keys, you can have as many as sixteen commands attached to one key. In order to end the definition of a macro, select this command again. In other words, this command is simultaneously stop and start definition.

Macros can contain any keystroke, including [Amiga]+key menu shortcuts, indeed they can contain any menu command at all. If a menu command in a macro brings up a requester, such as the search requester, then CygnusEd will query you as to whether you would like the contents of the requester stored in the macro. If you answer yes then the search string and flags will be inserted at playback time, and the requester will not be displayed - the same string will be searched for each time. If you answer no then the requester will come up each time you playback the macro. If you want your macro to search for whatever string is in the search buffer at the time the macro is invoked, then you would need to use a search command that doesn't bring up a requester at all - such as 'Repeat search forwards'. The same principles apply to the 'Open' and 'Save as' requesters, the 'Send DOS/ARexx command'' and other requesters.

If in the course of defining a new macro you invoke a previously defined macro then the contents of the previously defined macro are played back into the new macro. The new macro receives a copy of the previous macros contents. If at a later date you change the original macro, the new macro with its copy is unaffected.

The only menu commands that can not be included inside a CygnusEd macro are the macro menu commands.

To delete an existing macro, either use metamac, or assign an empty macro to that key. That is, select 'Begin short invocation macro', then enter the keystroke, then select 'Begin short invocation macro' again.

The commands in the macro menu cannot be added to macros.

In practice we suggest using the function keys and control-key sequences for creating commands.

```
  Some of my favourite macros are listed here, along with the keyboard
shortcuts I use for them:

  I attach the following macro to [Ctrl]+Return.  It has the convenient
(for  C  programmers)  effect of creating a new brace block and leaving the
cursor on an indented line.
  shift-Return (auto-indent)
  '{'
  shift-Return (auto-indent)
  '}'
  up arrow
  shift-Return (auto-indent)
  TAB

  I  attach  the following macro to [Ctrl]+S.  It has the useful affect
(for dyslexics) of swapping the next two characters.
  Delete
  right-arrow
  Alt-Delete (undelete last deleted character)
  left-arrow


  If  you wish to edit macros that you have created, see the separate
utility program 'MetaMac'.
```

1.41 Begin long invocation macro - Keyboard shortcut [Amiga]+M

```
        Begin long invocation macro - Keyboard shortcut [Amiga]+M

  To increase potential compatibility with other text-editors such as
Micro-Emacs, that have rather long and convoluted multi-character keyboard
commands, CygnusEd allows you to assign macros to key sequences. The
keyboard shortcut for this command is [Amiga]+[Shift]+m, as opposed to [Amiga]+m,
to reflect the similarity with the 'Begin short invocation macro' command.
With this command, after you type the first character to use in invoking the
command, you will be prompted as to whether you want to use additional
keystrokes to invoke this command. You can define as long an invocation
sequence as you want, but realistically, anything more than two characters
is unmanageable. You could, for instance, assign the 'Paste' command to the
rather non-standard sequence [Ctrl]+P,[Ctrl]+A,[Ctrl]+S,[Ctrl]+T,[Ctrl]+e. Some ↩
   might
even argue that this is an intuitive and wonderful way to set up your text
editor. Others, more sensible, might argue that the punishment for mistyping
a single character is too annoying.

  In order to end the definition of a macro, select this command again. In
other words, this command is simultaneously stop and start definition.

  As with the single character invocation macros, long invocation macros
can be edited with the separate utility program 'MetaMac'.
```

1.42 Define startup macro

Define startup macro

If you like to have your text editor always great you by typing in a cheery 'Hello jack', or if, like the majority of us who aren't named Jack, you'd rather have your text editor invoke a particular ARexx script on startup, the startup macro is perfect for you. Simply select this command and then enter the sequence of commands you wish played back at startup. To end the macro definition, select 'Begin short invocation macro'.

1.43 Quote key

Quote key

You may end up assigning a macro to a key, such as [Ctrl]+S, which you occasionally need to type into a document. If you need to do this, select the 'Quote key' command followed by the key in question. The 'Quote key' command tells CygnusEd to ignore any macros assigned to the keystroke which follows.

1.44 Clear definitions

Clear definitions

The 'Clear definitions' command clears all currently loaded macro definitions.

1.45 Load definitions... - Keyboard shortcut [Amiga]+;

Load definitions... - Keyboard shortcut [Amiga]+;

The 'Load definitions' command loads a CygnusEd macro file in, first clearing out all existing macro definitions.

1.46 Save definitions... - Keyboard shortcut [Amiga]+'

Save definitions... - Keyboard shortcut [Amiga]+'

The 'Save definitions' command saves the currently loaded CygnusEd macro definitions. The macros can be saved to any filename, but if you save them to s:cedmacros (the default name) then they will be automatically loaded in when you run CygnusEd the next time. If you save them to cedmacros in some directory other than s: then they will only be loaded in if the cedmacros file is in you current directory when you run CygnusEd.

1.47 Priority

```
                                Priority

   Inherit?
   Priority  x?
Set priority...
```

In order to cooperate well in the Amiga’s multitasking environment it is important that CygnusEd have its multitasking priority set to the appropriate level, so that it doesn’t steal more CPU time than is necessary, but it doesn’t run sluggishly. There are two modes which CygnusEd can operate in for deciding it’s priority. One is to inherit it’s priority from its parent. Its parent would either be the workbench, if you run it from there, or a CLI, if you run it from there. The priority of a CLI can be changed with the ChangeTaskPri command. If the priority of a CLI is changed after CygnusEd is run, that has no effect on CygnusEd’s priority.

The other more common way is to set CygnusEd to a specific priority. The default is one. This is usually a good choice because if you are compiling or ray tracking at a priority of zero or lower, CygnusEd will run at top speed, unaffected by your ray tracing. This is probably what you want, since speed and responsiveness are terribly important in a text editor, but not even possible in a ray tracer. If CygnusEd is sitting idle, when you aren’t using it, CygnusEd will use zero CPU time, so there is rarely a reason to lower CygnusEd’s priority. The one operation that should have a higher priority than CygnusEd is any telecommunication programs you may run, as downloads may experience errors if CygnusEd steals too many CPU cycles from them.

The ’Inherit’ and ’Priority n?’ check boxes (where ’n’ represents the currently selected fixed priority) decide whether CygnusEd inherits its priority or sets it absolutely. The ’Set priority...’ command lets you set the fixed priority, but it has no affect if ’Inherit?’ is set.

1.48 Autosave

```
                                Autosave

   xx min?
Set timer...
```

The autosave command is an extremely handy feature for making sure that you don’t lose work in case of an unexpected crash. Although CygnusEd is an extremely stable and torture tested program, other programs, especially ones in development, may unexpectedly take down your Amiga. Although autosave can’t completely protect you, it can minimize the damage.

If you enable autosave (select the ’n min?’ command to make a check box appear) then all changed files will be saved to temporary files every n minutes (you set the interval with ’Set timer...’). If you are editing ’letter.txt’ then an autosave copy will be saved to ’letter.txt.auts’. If you later manage to successfully save your changes, or if you close the file without saving, then the autosave file will be automatically deleted.

However if your system crashes unexpectedly, then the autosave files will be left behind. The next time you go to edit 'letter.txt.auts', CygnusEd will tell you that there is an autosave version, and will give you the option to load it in instead of the older version. It will then load the autosave version in under the proper file name, without '.auts'. So even if you don't realize that you've lost changes, CygnusEd will remind you. If you do realize that you've lost some changes, you can also look for them by searching for '#?.auts' files.

If you are editing a new file that you have not yet saved (an unnamed file) then CygnusEd will autosave the file as 'CygnusEdUnNamedN.auts', where the 'N' represents a number, necessary to ensure that the filename is unique. Because the file was never saved by you, CygnusEd can't warn you of the unsaved version the next time you load the real version. Therefore after a crash, if you think you may have been working on new files it is important that you look for any #?.auts files.

1.49 Set icon tool name...

Set icon tool name...

When files are saved from CygnusEd they can, optionally, be saved with an icon, for easy access from the workbench. The 'Set icon tool name' allows you to specify what the default tool should be for the icons. The 'default tool' is the program that will be run when you double click on the icon. The default 'default tool' is 'ed', the CygnusEd invoker. An alternate, more flexible way of controlling the icon used for files saved from CygnusEd is to create the icon you want used, complete with imagery, tool types and save it to 'env:ced/def_cedfile.info'. You'll have to create the 'env:ced' directory, and you'll probably want to save the icon to 'envarc:ced/def_cedfile.info' as well, to make your preferred icon a permanent setting.

1.50 Colours

Colours

Cycle colours - Keyboard shortcut [Amiga]+$

Adjust colours...
Use WBench colours

CygnusEd allows limited control over the colours it uses, and then only when it is using its own screen (ie; CygnusEd will not allow you to change the colours of the workbench screen or some other screen that CygnusEd is 'visiting'). The cycle-colours command, a rarely used command that is granted a keyboard shortcut for anachronistic reasons, merely shuffles the four available colours through all of the twenty-four mathematical permutations and possibilities. 'Adjust colours' brings up a colour selection requester, which allows you to customize the colours any way that you want. 'Use WBench colours' tells CygnusEd to grab the colours from the workbench screen. It also tells CygnusEd to grab the workbench colours every time CygnusEd is run, meaning that any changes in workbench colours will soon be reflected in CygnusEd's colour scheme.

1.51 File save method

File save method

CygnusEd offers you a number of different ways of saving your files, depending on how strongly you value safety versus speed. The simplest method is 'Simple saves'. With this method the file is simply saved out directly. This is slightly faster than the other methods, it avoids ever having multiple copies of file on disk (handy if you are running very low on space) and, because it doesn't create a brand new file, it can avoid some permission problems when saving to a file server. The main problem with 'Simple saves' is that if your Amiga crashes while CygnusEd is in the process of writing out the file, not only are the new changes lost, it's quite likely that the old file is lost as well, leaving you with a zero length file. If this does happen to you, stop your Amiga's reboot with a well placed [Ctrl]+D and look at the disaster recovery section of this manual.

Next up on the safety scale is 'Safe saves'. With this method the file is saved to a temporary filename, and when all of the data has been safely written to disk, the old file is deleted and the new one is renamed into its place. With this method, if your Amiga crashes, it is quite likely that either the old file, the new file, or both, will survive unscathed. The old file will typically be found unaltered, and the new file may be found listed under a name such as 'CygnusEdTemp.0Na', where 'N' represents some number.

Finally, the safest method, my favourite, my recommended choice, is 'Backup to *.bak'. This method is very similar to safe saves, and has all of its benefits. However instead of deleting the old file, the old file is rename to '*.bak'. If you are editing 'file.txt' then the old version will get renamed to 'file.txt.bak' (deleting any older versions of file.txt.bak if necessary). Not only does this make the saving process easier, but it allows you to revert to a previous version, long after you finish saving.

1.52 Icon creation?

Icon creation?

The 'Icon creation?' toggle governs whether CygnusEd saves an icon with each file it saves. See 'Set icon tool name' for more information.

1.53 Hot-Start enabled?

Hot-Start enabled?

In order to sate the power hungry desires of impatient programmers, CygnusEd has an optional mode, used by almost all CygnusEd users, called 'Hot-Start'. If you enable this mode it means that when you exit CygnusEd it doesn’t really exit. It sits waiting for your next command, invisible to the untrained eye. To awaken CygnusEd you can simply type ’[Alt]+[Shift]+Return’ - easily drummed out with the right hand, and CygnusEd instantly pops into view. If you are at the command line then you can type ’ed’ or ’ed filename’ or ’ed *.c’ and CygnusEd will pop up into view and load the requested files. For more information, see ’Quit & Die’ and ’Ed: the CygnusEd invoker’

1.54 Auto-expand views? - Keyboard shortcut [Amiga]+[

Auto-expand views? - Keyboard shortcut [Amiga]+[

CygnusEd can have many different files open and many different views on these files, Some people, like myself, like most of these views to be open so that I can see the contents of many files simultaneously. Others feel that only the currently active file is of interest, and they want all others to shrink away to nothing. ’Auto-expand views?’ turns on a mode in which whenever a new view becomes active, either through being selected with the mouse or through the next/previous view commands, that view is expanded to maximum size, while the surrounding views are shrunk down to the minimum one line height.

1.55 Keypad = movement? - Keyboard shortcut [Amiga]+8

Keypad = movement? - Keyboard shortcut [Amiga]+8

For those who work on IBM compatible computers all day, and yearn for the thrill of a numeric keypad that doesn’t type numbers, this command is for you. Enabling this command turns the numeric keypad into a directional pad, with 2, 4, 6, and 8 being arrow keys, 1 and 7 being end of line and beginning of line (end and home), 3 and 9 being bottom of screen and top of screen (page down and page up), 0 toggling insert mode, and ’.’ being the same as delete.

1.56 Select font

Select font

Select disk font...
Topaz 80 column
Topaz 60 column

CygnusEd was never intended to be a word processor, so don’t expect fancy font capabilities. CygnusEd does not allow multiple fonts in one file, it simply allows you to select alternate fonts for all of your files, to allow you to choose the appropriate font for your particular graphics card, monitor, and visual acuteness. You can either select Topaz 80 or Topaz 60, or use the standard ASL font requester to choose any mono-spaced font from disk.

1.57 Rendering choices

Rendering choices

Use custom scrolling routines?
Use system scrolling routines?
Make screen public?

CygnusEd's lightning fast screen update and scrolling have always been two of the features that have ensured its success. However being the fastest on the block doesn't help if everybody is moving to a graphics board that you're not compatible with. Therefore CygnusEd 3.5 offers a number of options to allow you to choose the balance between system compatibility and speed that is right for you.

The fastest routines are chosen by selecting 'Use custom scrolling routines'. With this choice CygnusEd writes characters directly to display memory (when on its own screen) and programs the blitter directly, for scrolling and redraws that happen faster than the eye can see.

Direct blitter programming can cause CygnusEd to sop up every single bus cycle available and, for reasons never quite explained, can cause some terminal programs to experience data errors while downloading files. All sorts of explanations as to why this can't happen have been given, but it continues happening. So, the simple solution is to select 'Use system scrolling routines?'. In this mode the scrolling is slightly slower (but still very fast) and the characters are written directly to the screen at a minimum speed of 30,000 characters per second.

The most system compatible mode, compatible with most third party graphics boards, is to make CygnusEd's screen public. In this mode CygnusEd strictly follows the rules and allows other programs to share its screen. This allows you to, for instance, open Mand2000 on CygnusEd's screen, and do some four colour fractal exploration while writing the great Canadian novel. Or you open Art Department Professional on your CygnusEd screen so that you can watch your ARexx scripts process images while you write e-mail.

1.58 Load environment

Load environment

Default - Keyboard shortcut [Amiga]+e
Specify...

All of the settings above can be saved to CygnusEd's environment files. Typically those settings are automatically loaded in, but if you wish you can force those settings to be reloaded. 'Default' loads in the default settings file, based on the current file extension. If the current file is called 'fred.txt' then the file 'ceddefaults.txt' will be looked for, in the current directory and then in 'S:'. If that file isn't found, then 'ceddefaults' (no extension) will be looked for, also in the current directory and then in 'S:'. All settings, global, file and view will be loaded.

'Specify' lets you select any settings file and load it in.

1.59 Save environment... - Keyboard shortcut [Amiga]+E

Save environment... - Keyboard shortcut [Amiga]+E

This command allows you to save all of the current environment settings. All of the global, file and view settings are saved. The default file name supplied is 'S:ceddefaults.ext' where '.ext' is the extension of the current file you're editing. If you have made changes to the global settings, you should probably save the environment to 'S:ceddefaults' as well as the global defaults are by default loaded from there initially, and are never loaded again, unless specifically requested by the user.

1.60 Tab size

Tab size

CygnusEd supports true tabs, of any size ranging from 1 to ten spaces long. Typical sizes are five or eight for writing letters, two, three or four for writing programs. Files that are displayed with a tab size other than the one they were created with can lose their formatting and look quite terrible.

Interesting animations can be made by swiftly changing the tab size of an excessively indented C file. Try holding down the shift key and drag selecting all ten tab sizes, then watch as CygnusEd changes tab sizes ten times. Or make a macro with even more tab size changes.

1.61 Customize tabs... - Keyboard shortcut [Amiga]+t

Customize tabs... - Keyboard shortcut [Amiga]+t

Some people, particularly assembly language programmers, aren't satisfied with evenly spaced tabs. They want a tab for opcodes, another for operands, and a third, way out there, for comments. Selecting 'Customize tabs' lets you specify any arbitrary arrangement of tab stops that you would like. Note that you are limited to setting tab stops in 160 columns; beginning with the 161th column, the tab stop settings will "wrap" around. Due to implementation constraints, CygnusEd will always assume that there is a tab stop in column #160.

1.62 Set right border... - Keyboard shortcut [Amiga]+^

Set right border... - Keyboard shortcut [Amiga]+^

The 'Set right border' command allows you specify a right border for your documents. This right border affects word wrap, paragraph reformatting, and centering of lines.

1.63 Set scroll jump

Set scroll jump

CygnusEd can scroll your files faster than the eye can follow, but sometimes an unreadable blur of text isn’t actually what you want. ’Set scroll jump’ controls how quickly text scrolls. Larger numbers mean faster scrolling.

1.64 Max scroll xx...

Max scroll xx...

One of the important design goals of CygnusEd was to ensure that, whenever possible, cursor moves from one part of a file to another were done by scrolling, rather than jumping to the new location and redrawing. The advantage of scrolling is that it means that you intuitively keep track of how far you’ve gone and where you are. If the screen redraws, you lose all visual clues as to where you are. It can be hard to differentiate between a move of several thousand lines and a move to the next line in the file.

’Max scroll’ lets you control this. It sets the maximum distance that CygnusEd will scroll. If you ask CygnusEd to move the cursor any further than this maximum number of lines, either by using the ’Jump to line’ command, the search command, or even the page down command, then CygnusEd will redraw the screen rather than scrolling. I strongly feel that ’Max scroll’ should be set to at least several lines more than the number of lines visible on the screen, and to at least thirty or so. However, some misguided people think that speed is more important than knowledge, so they like to set it down to ten, or less. These small maximum scroll jumps can speed things up, but the price you pay is an increased difficulty in following your travels through the file.

1.65 Layout? - Keyboard shortcut [Amiga]+5

Layout? - Keyboard shortcut [Amiga]+5

Normally CygnusEd does not allow the cursor to go beyond the end of lines. The cursor is normally only allowed to go where there are characters. Sometimes, particularly when using the vertical block command, or creating tables, it can be useful to allow the cursor to go out beyond the end of lines. Layout mode does this. Turn this on if you want the cursor to move beyond the end of lines.

If you are out beyond the end of a line and you type a character, CygnusEd will automatically fill in the necessary spaces to allow the character to be typed at that location. If you then delete that character, CygnusEd will then remove those spaces. Additionally, CygnusEd will remove any end of line spaces and tabs as you scroll around the file, if you are in layout mode.

1.66 Word wrap? - Keyboard shortcut [Amiga]+6

Word wrap? - Keyboard shortcut [Amiga]+6

Although CygnusEd is certainly not a word processor, it is frequently used for writing of letters and other non-programming documents, especially for e-mail. It has the advantage over word processors of being extremely quick, keeping up with even the swiftest typist. To support those who write paragraphs as well as functions, CygnusEd has a 'Word wrap' function. If you enable this mode then whenever the cursor reaches the right border (see 'Set right border') the current word and the cursor are wrapped around onto the next line.

However, because CygnusEd does not store text as paragraphs (it stores text as individual lines which is the standard ASCII file way to store files) and because CygnusEd does not store any hidden formatting codes, that's about all the the word wrap can do. If you delete words from the middle of a paragraph, the paragraph will not be reformatted. If you add words in the middle of a paragraph, no reformatting will happen until you reach the end of the line, at which point the entire end of the line will be wrapped.

If you edit a paragraph and destroy the formatting, CygnusEd can still help you out with its paragraph formatting command. See 'Format' for more information.

1.67 Insert mode? - Keyboard shortcut [Amiga]+7

Insert mode? - Keyboard shortcut [Amiga]+7

When entering new characters CygnusEd normally pushes all previous characters out of the way. If you put CygnusEd into 'Insert mode' then each new character typed replaces the one previously under the cursor. Insert mode does not effect deleting.

1.68 Tabs = spaces? - Keyboard shortcut [Amiga]+0

Tabs = spaces? - Keyboard shortcut [Amiga]+0

When you press the Tab key, CygnusEd normally enters a real tab character (ascii value nine). This is the most space efficient way of storing tabs, and it allows you to then delete the tab with a single press of the delete or backspace key. However if you move your document to another text editor that does not support the tab size that you have chosen, your document's formatting will change. Therefore CygnusEd gives you the option to have all new tabs entered as spaces. If you enable 'Tabs = spaces?' then pressing the Tab key enters the number of spaces required to move the cursor to the next tab stop.

1.69 Editable file?

Editable file?

The 'Editable file?' command allows you to tell CygnusEd that a particular file should not be changed. Any attempts to do so will cause an error message. The editable file flag will also get set if you load in a file that you do not have write permission for. If this happens you can enable editing in the file by selecting 'Editable file?', but you may not be able to write your changes out to the file because of insufficient permissions.

1.70 Status line

Status line

CygnusEd's status line is broken up into two parts. The left hand part normally displays as much of the path name of the loaded file as possible, but can be made to display other information through the controls below. The left half also normally displays a view number at the left edge if there is more than one view on a particular file.

The right half of the status line always displays the same information, in the following format:

```
        KLWIT * line 1000 col 234
```

These symbols mean:

'K' - if this is present, CygnusEd is in Keypad=Movement mode, and the numeric keypad is setup for cursor movement instead of typing numbers. Type [Amiga]+8 to toggle this.
'L' - if this is present, that file is in Layout mode and the cursor can go beyond the end of lines. Type [Amiga]+5 to toggle this.
'W' - if this is present, that file is in Word wrap mode, and words will wrap around when typed beyond the right border. Type [Amiga]+6 to toggle this.
'I' - if this is present, that file is in Insert mode, and characters typed will push other characters out of the way. Type [Amiga]+7 to toggle this.
'T' - if this is present, that file is in tabs=spaces mode and any tabs that you enter will be replaced with the appropriate number of spaces. Type [Amiga]+0 to toggle this.

'*' - if this is present it means that the file has been changed, that the change count (see below) is non-zero.

The remainder of the right half of the status line displays the current line number and column number for the current view.

The 'Status line' sub menu allows control over what information is displayed in the left half of a particular view's status line. The 'On/Off?' command turns the entire status line on or off. If the status line is off then the files name is showed instead.

If 'changes' is selected then the number of changes made to the current file since it was last saved are shown. Each character that you type into CygnusEd counts as a change. Each character that you delete counts as a change. Pasting 50,000 characters into a document counts as one change. Replacing all one thousand occurences of 'the' with the identical text 'the' counts as one thousand changes. So, the number of changes can give you an idea as to how much has been done to your file since you last saved it, but it does not give you an accurate measurement of how different your file is from the saved version. Note that the number of changes can be negative, if you saved the file and then undid some of the changes - undoing a change reduces the change count by the number that it originally went up by. If you type 'R' and then type 'Backspace' the change count will be two. If you type 'R' and then select 'Undo' then the change count will be zero. In both cases the file will be unchanged.

If 'pages' is selected then the number of pages in the document will be displayed. The number of pages is obtained simply by dividing the number of lines by 66, a common printer page length.

If 'Show ASCII values' is enabled then the status line displays the numerical ASCII value of the character under the cursor. This can be very useful when editing binary files (yes, CygnusEd gives you a limited ability to edit binary files). The ascii value replaces the KLWIT display.

1.71 White spaces

White spaces

If you are very concerned about precisely what is in your files, tabs and spaces can give you some problems, because they look identical. Ordinarily the only way that you can differentiate between them is to try deleting them. The commands in the 'White spaces' menu allow you to make tabs, spaces and end of line (EOL) characters visible. This can make your screen incredibly cluttered, but if you want to make sure that tabs or spaces are used consistently in your files, these options are very useful. Please note that this feature requires that you have selected 'Topaz 80 column' as the text font.

The 'Esc codes visible?' toggle is a peculiar beast which demonstrates CygnusEd's small time word processing replacement aspirations. You can enter printer or console escape codes directly into your documents to alter the appearance of text when printed. However entering these codes destroys the formatting of the document in CygnusEd so badly that it's very hard to get the desired appearance. If you disable 'Esc codes visible?' then all recognized escape codes are hidden - the only sign of their existence is that the following character is printed in inverse. This feature is a bit anachronistic, and I don't recommend that you use it.

1.72 Scroll bar

Scroll bar

Scroll bars, especially proportional scroll bars, can be incredibly useful gadgets, both for giving you smooth analog control over your location in a file, but also for giving you instant intuitive feedback about the size of your file and your location in it. However, some people feel that scroll bars waste too many columns of valuable screen real estate. And some people running earlier versions of AmigaOS occasionally run into a bug whereby they can’t move their mouse to the right edge of the screen. Therefore CygnusEd allows you to decide whether or not you want scroll bars, and which side you want them. You can even have each view having scroll bars on a different size. I’m not sure why you’d want to though...

1.73 Set scroll borders...

Set scroll borders...

The ’Set scroll borders’ command is a very important one for giving CygnusEd its distinctive smooth scrolling and its uncanny ability to always ensure that you have enough context. Far too many text editors happily let you move your cursor right to the bottom of the window, seemingly never realizing that if your cursor is at line 199, you’re probably also interested in what’s on line 200. The scroll borders define a box inside the window which CygnusEd tries to keep the cursor inside of. Whenever the cursor goes outside of this box, by getting too close to any of the edges, CygnusEd smooth scrolls the necessary number of lines of text onto the screen. The only time CygnusEd will allow the cursor outside of the scroll borders is if there is no additional data to scroll on - ie; if the cursor is at either end of the file, or the left edge, or the extreme right edge of the longest visible line.

1.74 View operations

View operations

Although the Amiga’s wonderful multitasking abilities mean that there is technically no need for a program to handle multiple files simultaneously (after all, you can always just run another copy of the program) it turns out that is frequently very useful to have many files loaded into a simple program at one time. It simplifies window management, cutting and pasting, and keeping everything together on one screen. CygnusEd allows you to load multiple files, and it also allows you to have multiple views on one file. So, for instance, you can easily be looking at the opening paragraph of your doctoral thesis in one view while looking at a supporting paragraph on page 97 in another view. Because the two views are actually looking at the same document, changes made to one view are also made to the other one - you never need to worry about them getting out of sync.

To create a new view select the ’Split view’ command. This takes the current view and splits into two roughly equal sized views. If necessary it will shove other adjacent views out of the way. At this point you have created two cooperating views on the same file. A common use for this would be that when you are writing code that calls a function, you may want to look at the functions definition earlier on in the file, or you may want to look at some variable declarations earlier on in the same function, but currently off screen. If you scroll back to look at these other areas then you have to return to your original location before you can use the information - by which time you may have forgotten. Even using marked locations to make the jumping back and forth easy won't allow you to examine both areas simultaneously. CygnusEd's multiple views and simple window management commands solve this problem. Simply type '[Amiga]+d' to split the current view in half. Your cursor will be in the bottom view, and both views will be displaying roughly the same area of the same file. Jump, search and scroll to the area of interest and cut or paste the relevant code. Now type '[Amiga]+,' (Amiga-comma) to jump to the previous view - the original one. Type or paste in your code, while still looking at the sample code in the view below. When your finished, type '[Amiga]+.' (Amiga-period) to move to the next view (the recently created one) and then type '[Amiga]+q' to close that view down and resume your work. It's simple, fast and powerful. You never have to take your hands off the keyboard. New views can be created on a whim and destroyed seconds later, having served their purpose.

If you want to look at a different file, perhaps an include file, a slight variation on the above process applies. After doing the split view, use the '[Amiga]+o' shortcut to open up a new file. This new file will be placed into the newly created view. From that point on the instructions are identical. In this case '[Amiga]+q' not only closes the new view, it also removes the new file from memory.

Of course you don't have to close the new view seconds after you created it. You can have up to twenty different views on a range of different files open all the time, and you can cut and paste between them, use the next/previous view commands to move among them, or use the mouse to select a new view or size them, or size them from the keyboard with macros attached to the 'Grow view' and 'Shrink view' commands.

Now, onto the individual function descriptions.

1.75 Previous view - Keyboard shortcut [Amiga]+,

Previous view - Keyboard shortcut [Amiga]+,

This command moves the cursor to the previous view, the one above the current one. If the top view is active, the bottom one will be selected. If only one view is open this command has no effect.

1.76 Next view - Keyboard shortcut [Amiga]+.

Next view - Keyboard shortcut [Amiga]+.

This command moves the cursor to the next view, the one below the current one. If the bottom view is active, the top one will be selected. If only one view is open this command has no effect.

1.77 Split view - Keyboard shortcut [Amiga]+d

Split view - Keyboard shortcut [Amiga]+d

The 'Split view' command takes the current view and splits it into two roughly equal sized views on the same file. If the current view is less than three lines high then the surrounding views will be pushed out of the way to make room. If there is no room to expand the other views, or if the number of views has already reached twenty then this command will fail.

1.78 Expand view - Keyboard shortcut [Amiga]+]

Expand view - Keyboard shortcut [Amiga]+]

This command expands the current view out to the maximum possible size.

1.79 Grow view

Grow view

This command moves the title bar of the current view up one line. If there is no room to move the title bar up, the title bar of the view below is moved down one line. This command is rather useless as a menu command with no keyboard shortcut, so I always use a macro to bind this command to a key. My usual choice is F8.

1.80 Shrink view

Shrink view

This command moves the title bar of the current view down one line. If there is no room to move the title bar down, the title bar of the view below is moved up one line. This command is rather useless as a menu command with no keyboard shortcut, so I always use a macro to bind this command to a key. My usual choice is F9.

1.81 Format

Format

The format commands continue CygnusEd's imitation of a simple word processor. Selecting one of these two commands tells CygnusEd to treat all the text from the current line down to the first blank line or line starting with a space or a tab, as a single paragraph. It then reformats this paragraph to fit neatly within the right border (see 'Set right border'). The left border is always assumed to be the first column. If you use the 'With fill' option then spaces are added as necessary to fill exactly to the right border. If you select 'Without fill' then no extra spaces are added. Normally one space is added after each word, with additional spaces added after punctuation such as periods, exclamation marks, etc. See 'Post period spaces'.

1.82 Post period spaces...

Post period spaces...

This command, added exclusively for my demanding European customers, controls how many spaces should be placed after punctuation such as periods and exclamation marks when formatting paragraphs. The default is two.

1.83 Send DOS/ARexx command...

Send DOS/ARexx command...

This command allows you execute any DOS or ARexx command from within CygnusEd. Simply type the name of the command, and any parameters which it takes and the command will be run. The output will by default be sent to a standard console window which will only open up if the program or script being run produces some output. The real power of this program shows up when you embed it within a macro. If you say yes when the macro creation process asks if you would like the contents of the requester stored in the macro then when you play back the macro it will automatically execute the desired script or command. This allows you to attach an unlimited number of ARexx commands to arbitrary keys on your keyboard.

1.84 Install DOS/ARexx command...

Install DOS/ARexx command...

CygnusEd allows you to make a limited number of ARexx commands (ten) show up in the ARexx menu. The 'Install DOS/ARexx command...' command lets you select these. Select the number, from one to ten, of the slot that you would like to use. Then enter the command that you would like executed. After installing a new command you should probably use the 'Save DOS/ARexx commands' command to save your new configuration. The command is not actually run at this point. It can be run either by pressing the appropriate function key, or by selecting the command from the menus. Don't forget to save your ARexx command definitions afterwards.

1.85 Load DOS/ARexx commands...

```
                        Load DOS/ARexx commands...

  This command loads a set of ten DOS and ARexx command names from a file
(default name s:RexxCommands) and places them in the user configurable ARexx
commands menu.
```

1.86 Save DOS/ARexx commands...

```
                        Save DOS/ARexx commands...

  This command saves the current set of ten DOS and ARexx command names
into a file (default name S:RexxCommands). If you use the default name then
the command names will be automatically loaded in again when you next run
CygnusEd.
```

1.87 Send DOS/ARexx output to...

```
                        Send DOS/ARexx output to...

  Executing Dos and ARexx commands frequently produces output. If you
don't ever want to see this, use this command to set the output file name to
'nil:'.
use the public domain 'null:' device, which has the same effect.

  Alternately, you can send the output to a file on disk, or to a console
window. The default output file is:

"con:0/0/640/90/Ced<->Dos & Rexx/Inactive/Auto/Close/Wait"

  Roughly translated this means a console window, opening up in the upper
left hand corner of the workbench screen (0,0), 640 by 90 pixels in size,
with a title bar saying 'Ced<->Dos & Rexx'. The console window will open
inactive, it won't open until the commands actually print something, it will
have a close gadget and it will allow you close it even though CygnusEd
still has it open.

  CygnusEd opens the output file, whatever it may be, once only (unless
you change it). That means, in the case of the default console file, that if
you execute a number of commands, each producing output, all of their output
will remain for you to view at your leisure, and it will all be in one
console window. The downside is that it means that the window doesn't
automatically go away when the commands finish executing. However, since it
doesn't come up active, and since you can make it as small as desired, or
use the 'null:' device, that shouldn't really matter. Finally, if you want,
you can use the standard 'publicscreen' keyword to cause the output console
window to open up on any public screen, including CygnusEd's. This can allow
you, for instance, to look at the results of a testrun of your program on
the same screen as your source code. Other conole windows, including regular
CLI shells can also be made to appear on CygnusEd's screen, allowing you to
do directory listings and other DOS commands from within CygnusEd.
```

1.88 Enter ASCII... - Keyboard shortcut [Amiga]+-

```
               Enter ASCII... - Keyboard shortcut [Amiga]+-

  The 'Enter ASCII...' command is used to enter a character into the
current file by typing in its ascii value. For instance, the ASCII value of
the space character is 32, the ASCII value of the number one is 49. The
ASCII values range from zero to 255, and can be entered in decimal,
hexadecimal (base 16 - precede the numbers with either a dollar sign '$' or
'0x'), or even base 2 (precede the numbers with a percent sign '%'). The
Enter Ascii requester keeps on returning until you select 'Cancel' or hit
return when it is empty. This allows you to easily enter sequences of ASCII
values. Any character can be entered into CygnusEd.
```

1.89 Center cursor - Keyboard shortcut [Amiga]+=

```
               Center cursor - Keyboard shortcut [Amiga]+=

  The 'Center cursor' command attempts to place the cursor, and the line
of text that it is on, in the center of the current window. It scrolls the
current file up or down in order to do this. The only thing that will stop
CygnusEd from vertically centering the cursor is if the cursor is too close
to the either end of the current file, in which case the scrolling will stop
when the top or bottom of the file meets the top or bottom of the window.
The cursor is not horizontally centered. The text is not altered.
```

1.90 Center line - Keyboard shortcut [Amiga]+\\

```
                Center line - Keyboard shortcut [Amiga]+\

  The 'Center line' command takes the current line of text, strips all
white space (space characters and tabs) from each end, and then centers the
result in the area between column zero and the user defined right edge. The
centering is done by padding the beginning of the file with spaces. If the
line is too long to fit then it is not changed at all.
```

1.91 Repeat key/menu... - Keyboard shortcut [Amiga]+”

```
             Repeat key/menu... - Keyboard shortcut [Amiga]+"

  The 'Repeat key/menu' command is a powerful command for mechanizing
repetitive work. When you select this command q requester appears asking for
the desired repeat count. After you enter it, CygnusEd waits for you to type
any key or select any menu item, at which point it repeats it the desired
number of times. If you need to type in exactly eighty '*' characters, or if
you need to move the character down exactly 93 lines, this command is
perfect. If you need to do one of these operations frequently you can define
a macro which contains a repeat command - the repeated keystrokes will be
recorded in the macro just as if you typed them yourself. Additionally, if
```

you have a macro which you would like repeated dozens of times, this command will do that. And don’t forget that you can repeat any keystroke and any menu - there doesn’t even have to be a keyboard shortcut for the menu, just select it with the mouse if you want.

1.92 Find matching bracket - Keyboard shortcut [Amiga]+h

Find matching bracket - Keyboard shortcut [Amiga]+h

The ’Find matching bracket’ command is extremely useful for programmers. If you place the cursor over any of the parenthetical characters (’(’, ’)’, ’[’, ’]’, ’<’, ’>’, ’{’ or ’}’) then CygnusEd will search for its mate. In doing so CygnusEd will respect any other parenthetical pairs inside of the same type. For instance, if your cursor is on the first line of this text ’((Testing)EndTest)’ then find matching bracket will move the cursor to the end of the text. In this example, ’(]TestingEndTest)’, the cursor will also move from the beginning of the text to the end, because ’]’ doesn’t pair up with ’(’. Using the ’Find matching bracket’ command twice in a row will always return you to your starting location.

1.93 Mark/Jump Location

Mark/Jump Location

Mark location 1, 2, 3 - Keyboard shortcuts - [Amiga]+[Shift]+1, 2, 3
Jump to mark 1, 2, 3 - Keyboard shortcut [Amiga]+1, 2, 3

The ’Mark location’ commands record the cursors current position in the file so that you can easily return to a particular place in a file. The shortcuts were chosen so that, for instance, [Amiga]+[Shift]+2 will mark a spot and [Amiga]+2 will jump to that spot. If you mark a particular line and then add additional lines before that, the mark location will automatically be updated so that the same line of text will be jumped to.

1.94 Mark - Keyboard shortcut [Amiga]+b

Mark - Keyboard shortcut [Amiga]+b

The ’Mark’ command is used for starting to mark a block for cutting or copying. The cursor location when this command is selected is used as one end of the block and the cursors position is used as the other end of the block. Therefore, immediately after selecting this command, the area selected contains no characters. Moving the cursor forwards or backwards adds one character to the selected area. When you move the cursor forwards through the file, the character under the cursor when you selected ’Mark’ is included in the block, if you move backwards through the file it is not.

Note that unlike most text editors and word processors, when CygnusEd highlights a block, it only highlights out as far as there is actually text.

This makes the block marking very useful for tracking down over length lines or problems with extra or missing text.

The contents of the clipboard are not affected unless you select 'Cut' or 'Copy'.

Marking a block can be cancelled by selecting 'Mark' again.

1.95 Mark columnar - Keyboard shortcut [Amiga]+B

Mark columnar - Keyboard shortcut [Amiga]+B

The 'Mark columnar' command is similar to the 'Mark' command except that the marked out area is always a rectangle. This is very useful for removing columns from tables, adjusting indenting, creating script files and much more. If you paste in a columnar block then all of the lines of the block are pasted in at the same column number in adjacent lines of the file, padding the lines with spaces if necessary. The other important difference when pasting a vertical block is that the cursor position is adjusted differently. Pasting a normal block leaves the cursor one column to the right of the last character inserted. Pasting a columnar, or vertical block, leaves the cursor one row below the first character of the last line inserted. This is very handy if you want to insert the same columnar block multiple times because you can just keep on pasting and the blocks all stack downwards. The most common use for this is to do a columnar cut on a tab character and then hold down [Amiga]+V to repetitively paste this tab character in, thus indenting a section of text or code. Doing this with a regular block would require extra cursor movements after each paste, or a macro.

Marking a columnar block can be cancelled by selecting 'Mark columnar' again. Selecting 'Mark' after 'Mark columnar' merely changes the type of block being marked from a columnar block to a normal block, and vice versa.

Columnar blocks frequently enclose areas with lines of varying lengths. In this situation it is sometimes necessary, when cutting or when pasting, to move the cursor beyond the end of the current line to allow cutting out text on a longer line earlier on. For this reason, columnar blocks are frequently used in 'Layout mode' (see layout mode)

1.96 Mark word

Mark word

This command marks the word under the cursor, taking care not to mark any blank spaces next to it.

1.97 Cut - Keyboard shortcut [Amiga]+x

Cut - Keyboard shortcut [Amiga]+x

The 'Cut' command cuts the currently marked area to the Amiga clipboard. If CygnusEd is not in mark or mark columnar modef, the warning 'No area selected' appears. If no characters are selected, either because the cursor is on top of the block start or because the cursor is right below the block start in columnar mode, a requester warning 'No area marked' or 'No area selected' appears. Text cut to the clipboard can be pasted into the same file, other files, other copies of CygnusEd or other Amiga applications, including the standard console. The selected text is removed from the file and 'mark' mode is exited and the highlighting cleared. If in columnar block mode, all of the lines in the block range are shortened by the number of characters removed. If the block is a columnar block then an extra IFF chunk is placed in the clipboard to tell CygnusEd this. This chunk will be ignored by all other applications, so other programs will paste CygnusEd's columnar blocks as if they were regular blocks of text.

1.98 erase

Erase

The 'Erase' command works just like the 'Cut' command in that it removes the marked text. However, the marked text does not get transferred into the clipboard.

1.99 Copy - Keyboard shortcut [Amiga]+c

Copy - Keyboard shortcut [Amiga]+c

The 'Copy' command is just like the 'Cut' command except that the file is not altered.

1.100 Paste - Keyboard shortcut [Amiga]+v

Paste - Keyboard shortcut [Amiga]+v

The 'Paste' command pastes the current contents of the Amiga clipboard into the current file at the cursor location. Normally the clipboard will contain text cut or copied out of CygnusEd but it may also contain blocks of data from other programs. The contents of the clipboard are unaffected by the paste command.

1.101 Set clipboard unit...

Set clipboard unit...

The Amiga clipboard actually consists of 256 different clipboards. Most applications only use clipboard zero, but occasionally having extra places to store temporary text can be very useful. You can temporarily set the clipboard unit to a different number, cut or copy some text, to back to clipboard unit zero, do some more editing, including using the clipboard, and then switch back to the different clipboard number to retrieve the text you cut out much earlier, unaffected by intermediate clipboard operations. The only thing to be careful of is that, since most applications support clipboard unit zero only, you will not be able to cut and paste to most other applications with the other clipboards. The main way that people use this command is to setup three macros that switch to a different clipboard unit, cut, paste or copy, and then switch back. This lets you select the appropriate clipboard without having to adjust your settings manually. CygnusEd allows you to store the clipboard numbers in the macro just by saying 'yes' when asked when you are defining the macro.

1.102 Rot marked

Rot marked

Occasionally text files on the internet are 'encrypted' using a method known as 'rot13'. The purpose of this 'encryption' is not to stop people from reading the text but to stop them from accidentally reading the text. This allows people to post rot13 rude jokes with a non-encrypted warning so that people can read the warning and then decide whether or not to read the encrypted text. The main advantage to the rot13 method is that if you do it twice, you end up back where you started from - therefore encrypting text is done the same way as unencrypting it.

The method, for those of you interested, is very simple. Just move all alphabetic characters forwards thirteen positions, wrapping around from 'Z' back to 'A'. If you do that twice then every character has been moved forward twenty six positions, putting it back where it started.

The 'Rot marked' command rotates the currently selected text, and deselects it. Non alphabetic characters are unaffected.

1.103 Strip CR marked

Strip CR marked

MS-DOS based machines terminate the lines of their text files with a carriage return (ascii 13) and a line feed (ascii 10), whereas Amiga text files are terminated with just a line feed. This command quickly strips out all of the annoying carriage returns in the selected text, then deselects the text.

1.104 Change case marked

Change case marked

The 'Change case marked' command changes all of the selected alphabetic characters to upper case if they are lower case and to lower case if they are upper case.

1.105 Change marked to lower case

Change marked to lower case

This command changes all of the selected alphabetic characters to lower case.

1.106 Change marked to upper case

Change marked to upper case

This command changes all of the selected alphabetic characters to upper case.

1.107 Shift in marked

Shift in marked

This command is for indenting program source code and ordinary text by adding a 'tab' character at the beginning of each line in the marked area. Blank lines will remain blank.

1.108 Shift out marked

Shift out marked

This command is for removing indentations in program source code and ordinary text by stripping a leading 'tab' character from the beginning of each line in the marked area. No change will take place if there are no 'tab' characters at the beginning of the lines in the marked area.

1.109 Change marked spaces to tabs

Change marked spaces to tabs

This command will replace all blank spaces in the marked block by the correct amount of tabulator characters. The text layout will stay the same.

1.110 Change marked tabs to spaces

```
                       Change marked tabs to spaces

  This command will replace all tabulator characters in the marked
block by the correct amount of blank spaces. The text layout will stay the
same.
```

1.111 Delete word [Ctrl]+Del

```
                           Delete word [Ctrl]+Del

  The 'Delete word' command deletes the next word in the file and puts it
in a special 'delete word buffer'.
```

1.112 Undelete word [Ctrl]+[Alt]+Del

```
                       Undelete word [Ctrl]+[Alt]+Del

  The 'Undelete word' command pastes the last deleted word into the file
at the current cursor position.
```

1.113 Bck Spc word [Ctrl]+BckSpc

```
                      Bck Spc word [Ctrl]+BckSpc

  The 'Backspace word' command deletes the next word in the file and puts
it in a special 'Backspace word buffer'. This is separate from the 'delete
word buffer'. Backspace word deletes words to the left of the cursor instead
of the right.
```

1.114 UnBck Spc word [Ctrl]+[Alt]+BckSpc

```
                     UnBck Spc word [Ctrl]+[Alt]+BckSpc

  The 'Unbackspace word' command pastes the last backspaced word into the
file at the current cursor position.
```

1.115 Delete line - Keyboard shortcut [Amiga]+k

```
                Delete line - Keyboard shortcut [Amiga]+k

  The 'Delete line' command deletes the entire current line and puts it in
a special 'deleted line buffer'.
```

1.116 Delete to EOL - Keyboard shortcut [Amiga]+y

```
               Delete to EOL - Keyboard shortcut [Amiga]+y

  The 'Delete to EOL' command deletes the current line from the cursor
position to the end of the line, not including the end of line character. It
shares the 'deleted line buffer' with the 'Delete line' command.
```

1.117 Undelete line - Keyboard shortcut [Amiga]+l

```
               Undelete line - Keyboard shortcut [Amiga]+l

  The 'Undelete line' command pastes the contents of the deleted line
buffer (filled either by the 'Delete line' or 'Delete to EOL' commands into
the file at the current cursor position.
```

1.118 Repeat search backwards - Keyboard shortcut [Amiga]+a

```
          Repeat search backwards - Keyboard shortcut [Amiga]+a

  The 'Repeat search backwards' command is used to search again for the
last sequence of characters searched for, going backwards. The keyboard
shortcut was chosen so that its position, to the left of the 'repeat search
forwards' shortcut would suggest searching to the left, or backwards. If no
search string has been specified yet, the search requester is brought up.
```

1.119 Repeat search forwards - Keyboard shortcut [Amiga]+s

```
           Repeat search forwards - Keyboard shortcut [Amiga]+s

  The 'Repeat search forwards' command is used to search again for the
last sequence of characters searched for, going forwards. The keyboard
shortcut was chosen so that its position, to the right of the 'repeat search
backwards' shortcut would suggest searching to the right, or backwards. It
was decided to use the same shortcut as for 'Search for', except for the
lack of the shift key, to suggest the close ties between these commands. As
with the mark/jump to mark commands, the setup command uses the shift key,
the reuse command does not, on the assumption that the setup command
(marking a location or doing an initial search) will be done less frequently
than the reuse command (jump to mark or repeat search). If no search string
has been specified yet, the search requester is brought up.
```

1.120 Search for... - Keyboard shortcut [Amiga]+S

Search for... - Keyboard shortcut [Amiga]+S

The 'Search for' command is used to start searching for a sequence of characters. A requester appears with a string gadget for typing in the characters to be searched for. Additionally there are eight toggles for controlling the search.

Ignore case - when this is checked (the default) then the search is case insensitive - 'a' is considered equal to 'A'.

Wildcards - when this is checked the '*' character will match any single character. Obviously this is a fairly weak wild card capability, but it can be very useful.

Forwards - this controls the direction of the search. By default it is selected, meaning that the search proceeds forwards through the file. Unchecking it means that the search goes backwards.

Only whole words - this check box is very useful when searching for small words, such as 'the', or a variable name such as 'i'. Normally these sequences of characters would be found inside other words, such as 'then', 'there' and 'whether' or 'while', 'int' and 'main'. When 'only whole words' is checked, it is only considered a match if the characters before and after the character sequence are neither alphanumeric nor an underscore. In other words, it only counts as a match if the sequence of characters are found as an entire word, rather than the beginning, end or middle of a word.

Wrap around - If no matching text is found in the given direction, the search process will wrap around at the end/beginning of the file. This allows you to find/replace all occurences of a string without having to invoke this function twice.

Begin at top - Regardless of where the cursor is positioned at the time you open this requester, the search/replace process will always begin at the beginning/end of the file.

Expand escape codes - This will turn embedded escape sequences like \n, \r etc. in the search/replace strings into the appropriate characters. Supported are:

```
       \x<hex> with <hex> being a hexadecimal number, two digits max.
       \<oct>  with <oct> being an octal number, three digits max.
       \a
       \b
       \f
       \n
       \r
       \t
       \v
       \'
       \"
       \?
       \\
```

Show summary - When the search/replace process has completed, CygnusEd will show a summary of how many patterns have been replaced or whether the search pattern could be found. This switch allows you to turn this message off, which can be handy for macros.

There are two features of the search requester which are frequently not noticed. The first is that there are menus, which give [Amiga] key access to all of the gadgets, and also allow cutting and pasting from the Amiga clipboard into the search or replace gadgets. This can make searching much easier and less error prone, as you frequently don’t need to type in the text to search for. There is also an ’erase’ menu command to clear out the search buffer.

The other hidden feature is the search history. The up and down arrow keys can be used to look through the last twenty commands searched for. If you hold down the shift key while using the arrow keys then a search is done for the next or previous search entry that starts with the characters to the left of the cursor.

As search can be started either with the [Amiga]+S, or by hitting return or enter. To replace text, type [Amiga]+E or press the "Replace" button. For more information, see
Replace...
.

CygnusEd’s search routine is written in carefully optimized assembly language which allows CygnusEd to do a full, case sensitive or insensitive search with simple wildcards at the rate of 100,000 characters per second on a 7Mhz 68000. Obviously on faster processors the search speed is much faster.

1.121 Repeat replace - Keyboard shortcut [Amiga]+r

Repeat replace - Keyboard shortcut [Amiga]+r

The ’Repeat replace’ command repeats the last replace command, in the direction that the previous replace went. Note the similar keyboard shortcuts for these two commands, evoking there similarities, and showing the same consistent relationship used in the search commands the mark/jump to mark commands.

1.122 Replace... - Keyboard shortcut [Amiga]+R

Replace... - Keyboard shortcut [Amiga]+R

The ’Replace’ command uses the same requester as the ’Search for’ command. To start replacing text, just press the "Replace" or the "Replace all" button.

After a replace command is started, either by typing [Amiga]+E, selecting the ’replace’ button or hitting return, CygnusEd searches for the desired text. If the text is found the title bar for that file changes to the message:

'(Y)es/(N)o/(L)ast(G)lobal/(T)urbo/(Q)uit?'

Cygnused then waits for you to type one of the capital letters to state your preference. If you type 'Q', or any unrecognized character then the replace is cancelled, and CygnusEd returns to normal.

If you type 'Y' then the text under the cursor is replaced and the search/replace continues after the replaced text.

If you type 'L' then the text under the cursor is replaced and the search/replace stops.

If you type 'G' then the text is replaced and CygnusEd continues searching and replacing without prompting you. Once you start this mode you can stop it at any time by pressing any key. When CygnusEd does a global replace it still updates the screen, scrolling as needed, so that you can see exactly what is going on. If this is too slow for your tastes...

If you type 'T' then CygnusEd does a 'Turbo' replace. This is similar to the 'Global' replace in that no further promting is done and that you can cancel the replacing by hitting any key. However the screen is not updated (except for a running count of the number of replacements done in the title bar). This alone speeds up the replacing considerably, but additional internal optimizations are done to ensure that as little data movement as possible is done. This tends to make the Turbo replace run dozens to thousands of times faster than global replace. This makes global operations on multi-megabyte files a matter of just a few seconds.

The other difference with 'Turbo' mode is that, for purposes of undo, all of the separate replacements are treated as one operation. The undo command will undo 100,000 replacements at once, whereas with the other replacement options each replacement is undone separately.

If you type [Amiga]+P or press the "Replace all" button, CygnusEd will go into 'Turbo' replace mode without displaying the '(Y)es/(N)o/(L)ast(G)lobal/(T)urbo/(Q)uit?' message.

1.123 Clip to search buffer

Clip to search buffer

To avoid the tedious and error prone typing in of search strings CygnusEd allows you to cut or copy text out or your files and then paste them into the search buffer. This can either be done with the paste command in the search/replace requester (see the menus) or with the 'Clip to search buffer' command. The main reason for having the 'Clip to search buffer' command is because it can more easily be put into macros and ARexx scripts.

1.124 Clip to replace buffer

Clip to replace buffer

To avoid the tedious and error prone typing in of replace strings CygnusEd allows you to cut or copy text out or your files and then paste them into the replace buffer. This can either be done with the paste command in the search/replace requester (see the menus) or with the 'Clip to replace buffer' command. The main reason for having the 'Clip to replace buffer' command is because it can more easily be put into macros and ARexx scripts.

1.125 Set ASCII zero alias for search...

Set ASCII zero alias for search...

CygnusEd will allow you to load virtually any file in, anything from source code to binaries. All of the characters of the file are displayed and can be deleted, copied, pasted and searched for - except for the character represented by the ascii value zero. The character '0', ascii 48, is no problem, but ascii zero is a problem, because this special value is used to terminate strings in the C language. 'Set ASCII zero alias for search' allows you to specify a character, which when typed into the search requester, will be understood to represent ASCII zero. You should choose some character which you don't frequently need to search for, because when using this option you won't be able to search for this character, as it will always get translated to ascii zero when searching starts.

1.126 Change case letter - Keyboard shortcut [Amiga]+g

Change case letter - Keyboard shortcut [Amiga]+g

The 'Change case letter' command simply changes the case (upper to lower, lower to upper) of the character under the cursor if it is an alphabetic character. Whether it is or not, the cursor is moved to the next character, allowing you to toggle the case of a large sequence of characters just by repeating the command.0

1.127 Change case word - Keyboard shortcut [Amiga]+G

Change case word - Keyboard shortcut [Amiga]+G

The 'Change case word' command changes the case of the character under the cursor and all subsequent characters until it comes to the end of the current word. See next/prev word for the definition of a word. The cursor is moved to the end of the word, to allow easy repetition of the command.

1.128 Upper case word

Upper case word

The 'Upper case word' command upper cases the character under the cursor and all subsequent characters until it comes to the end of the current word. See next/prev word for the definition of a word. The cursor is moved to the end of the word, to allow easy repetition of the command.

1.129 Lower case word

Lower case word

The 'Lower case word' command lower cases the character under the cursor and all subsequent characters until it comes to the end of the current word. See next/prev word for the definition of a word. The cursor is moved to the end of the word, to allow easy repetition of the command.

1.130 Undo

Undo

```
Undo - Keyboard shortcut [Amiga]+u
Redo - Keyboard shortcut [Amiga]+U
Max undo levels...  xxx
Max undo memory...  xxxxx
```

CygnusEd sports a very powerful undo capability, a feature that no text editor (or other program for that matter) should be without. CygnusEd's undo is multi-level, allowing you to change your mind easily.

Undo is useful for:

Undoing accidental deletions of important text.
Returning to previous versions of code or text if a new version doesn't work out.
Reminding you what you were just working on (undo, then redo - the cursor will go to the last change).

Each time you select 'Undo' you go further into the past. Each time an additional change that you made to the file is reversed. Each character typed is a change, each block cut or pasted is a change - copying a block is not a change. Only operations which actually change the file buffer count as undoable changes - anything that updates the change counter in the status bar.

Changes can only be undone in the strict reverse order that they were done in, and they can only be undone one at a time (although you could make a macro, or use the repeat command to undo multiple ones in one operation). These limitations stem from the fact that trying to undo an operation out of place frequently doesn't make sense - it may mean deleting text that hasn't been typed in yet, or some other contradictory thing.

When you undo you are moving backwards through time. You can then decide that you want to move forwards through time again by using the 'Redo' command. This plays back your changes. However, and this is very important, if you undo several operations and then make a change to the file, like typing in a single character, then the redo information is necessarily wiped out. If you try to undo that new operation, planning to then redo the previous operations, you will find that redoing will simply redo the typing of that character. In other words, although CygnusEd supports time travel through undo and redo, CygnusEd does not support multiple concurrent paths through time - the contradictions and troublesome philosophical conundrums are just too weird to ponder.

Note that CygnusEd's undo allows you to undo past saves. If you do this then the change count in the status bar actually goes negative! This is normal and expected behaviour. The only time that this causes a problem is if you make some changes to a file and then save it, thus zeroing the change count. If you then undo one of these changes, the change count goes to negative one. If you then change the file again, the change count increments to zero, and CygnusEd thinks that no changes have been made, when actually two changes have been made since the last save.

The number of levels of undo that CygnusEd supports is limited only by available memory. Since different people have very different amounts of memory and very different feelings on the importance of having a thousand levels of undo, CygnusEd allows you to configure how much memory should be devoted to the undo buffers, per file loaded. As the names suggest, 'Max undo levels' specifies that maximum number of undo events that should be stored per file. Up to 9999 levels are supported, but 100 is a more typical number. Usually just a dozen or so levels of undo are used, but occasionally you may find yourself heavily editing a file, only to realize after an hour or so that you are completely on the wrong track - at times like that, being able to reverse all your changes can be invaluable. If you haven't saved then you could use your previous version on disk, or your backup version (automatically created by CygnusEd if you enable that option), but if you've saved several times, then undo may be your only hope.

Since it is the memory consumed by CygnusEd's undo which is your real concern, CygnusEd allows you to set the maximum amount of memory which CygnusEd will devoted to undo buffers, per file. The number of undo levels supported is the smallest of the maximum number of levels specified and the maximum number that will fit in the maximum amount of memory specified. Since the size of undo events varies wildly (storing the typing of a single character in the undo buffer is cheap - storing the pasting in of a 100K block is expensive) the number of events in the buffer may vary considerably. Obviously setting both numbers high gives you many levels at the cost of much memory, and setting the both low gives you few levels at the cost of little memory. Setting the number of levels high but the memory low means that the number of undo events stored will be high as long as the undo events are simple - small blocks or simple typing, and the amount of memory used will be very consistently around the maximum. Setting the number of levels low but the memory high means that there will almost always be the number of levels stored that you requested, and that the amount of memory will usually be very small, but occasionaly quite big. Unless you are very short on memory, the defaults, or even larger numbers, work very well.

1.131 Jump to line... - Keyboard shortcut [Amiga]+j

Jump to line... - Keyboard shortcut [Amiga]+j

As expected the 'Jump to line' command moves the cursor to the requested line number. This requester allows you to enter negative line numbers; these will move the cursor relative to the end of the file, e.g. a value of -10 in a file with 30 lines will move to line 20.

1.132 Jump to auto-mark - Keyboard shortcut [Amiga]+4

Jump to auto-mark - Keyboard shortcut [Amiga]+4

The 'Jump to auto-mark' command is a somewhat unique CygnusEd feature. It derived from the observation that people quite frequently briefly jump to a very different location in the file they are editing - to the beginning or end or to a searched for string - and then want to return. Clearly the way to do this is to mark their current location before doing the initial jump, but I usually forget, and I suspect that other people do to. So CygnusEd attempts to recognize when you are doing this and automatically set a mark for you. CygnusEd's algorithm for deciding when to set the auto-mark is simple - if the user moves, by any means, to a location that is too far away to scroll to, then the auto-mark is set at the old location. Simple enough. You can then use 'Jump to auto-mark' to return to your old location. It doesn't always work, because you might, for instance, jump to the beginning of the file and then search for something that causes another jump - now the auto-mark points to the beginning of the file. However it works enough that I hate to use an editor that doesn't have it.

The only other time that the auto-mark gets set is when you use it. If you do a 'Jump to auto-mark' then, even if the destination is close enough to scroll to (it can happen, for subtle reasons), the auto-mark is set, allowing you to toggle back and forth.

Note that 'close enough to scroll to' is a user settable thing. See 'Max scroll'.

1.133 Jump to last change

Jump to last change

This command will place the cursor at the position of the last undoable editing operation.

1.134 Jump to byte... - Keyboard shortcut [Amiga]+J

Jump to byte... - Keyboard shortcut [Amiga]+J

The 'Jump to byte' command moves the cursor to a specific character or byte location in the file. The bytes in the file are numbered sequentially starting at zero. Note that the cursor column is not necessarily related to the byte number because some characters can expand to fill multiple columns (tab characters) and some characters can take up no space (escape characters, see 'Esc codes visible?'). This command can be very useful for

stripping the MacBinary header off of a file, as this is the first 128 bytes. Just do a 'mark', then jump to byte 128, then cut.
The requester allows you to enter negative offsets. These will move the cursor relative to the end of the file. For example, with a file 256 bytes in size, an offset of -10 will move to byte offset 250.

1.135 Cursor key movement

Cursor key movement

Shift+Cursor keys

CygnusEd was carefully designed to make the cursor movement commands natural and easy to remember. Instead of using [Amiga]+T to go the the top of the file, the arrow keys, with their clearly implied directions, are used with various qualifier keys.

The text in the menus gives a brief description of what the different the cursor keys do, always in the order up-arrow, down-arrow, left-arrow, right-arrow.

Holding down the shift key moves the cursor to the edge of the current page, roughly speaking. Shift plus the up or down arrow keys moves the cursor to the bottom or bottom of the current screen - as far as it can move without scrolling. If the cursor is already at the top or bottom then it scrolls up or down one page. Shift plus the right or left arrow moves the cursor to the left or right edge of the current line, even if this means moving beyond the edge of the page.

Holding down the alt key moves cursor twelve character positions, columns or rows, in the appropriate direction. This is one of my favourite ways of navigating a file, as it is less harsh and easier to follow than moving by pages, but not as slow and over delicate as moving a line at a time.

With the control keys the cursor movements are quite different in the horizontal versus the vertical direction. Holding down the control key when typing the up or down arrow keys moves to the beginning and the end of the file respectively. Holding down the control key when typing the left or right arrow keys moves the cursor to the previous and next words respectively.

CygnusEd's word movement algorithm is as follows. It was defined to make deleting words as useful as possible, by trying to ensure that CygnusEd doesn't delete too much excess text. If the cursor is on and end of line character, the cursor is moved one character in the appropriate direction and the move is over. If the cursor is over an alphanumeric character, it skips over that character and over any subsequent alphanumeric characters, and then skips over any tabs or spaces at the end, if moving forwards. If the cursor starts on a non-alphanumeric character that isn't a tab or a space then it skips that single character and any following spaces or tabs, if moving forwards. If the cursor started out on a tab or a space then it moves over all consecutive tabs and spaces. If it's moving forwards then it stops then. If moving backwards it continues on to deal appropriately with any alphanumeric or non-alphanumeric characters that follow.

The one remaining way of navigating around files in CygnusEd is with 'turbo mode'. This is a variable speed smooth scroll way of swiftly getting to where you want to be, while allowing you to slow down at any time to make sure you know where you are. You can smoothly adjust the scroll speed from being slow enough to glimpse words in the file, to fast enough that only the shape of the written areas can be discerned. To start turbo mode use the alt key together with the up or down arrow keys. Initially this moves twelve lines at a time, but as soon as the arrow key starts auto-repeating you're in turbo mode. At this point you can lift off the alt key to slow yourself down, or press the shift key to speed up. You can smoothly move between these three speeds (holding down the shift key only is the same as holding down the alt key only) as you look for the text you want to find.

1.136 MetaMac

MetaMac

MetaMac is CygnusEd's macro editor. CygnusEd macros are usually created from within CygnusEd, by simply doing the actions you want turned into a macro while having CygnusEd record them. However if you make a mistake while defining a macro, if you want to adjust what a macro does, or if you just want to see what macros you have defined, MetaMac is the program for you.

When you first run MetaMac two windows appear. The left window contains a list of all of the macros that you have defined. The right window displays the contents of the currently selected macro, if any.

Using the left window is quite easy and intuitive. You can change which macro is selected by using the arrow keys or the mouse. Macros can be deleted with the delete key or with the delete macro button. The project menu work as you would expect. The trickiest part is the 'Add Macros' menu. The three menu entries here let you find or create the three basic types of CygnusEd macros - one key invocation macros, multi-key invocation macros, and a startup macro. If you choose 'Find/Add short invocation macro' then MetaMac asks you what key you would like to assign the macro to. If that key is already in use it selects that macro for you. If it isn't in use, it creates an empty macro assigned to that key. If you never define actions for that macro key then you have effectively disabled that key.

Using the right window is somewhat more confusing. First of all note that the complete CygnusEd menus are attached to this window. Any keystrokes or menu selections that you make in this window are added to the currently selected macro, if any. This is necessary in order to allow you to add any keystrokes and macros to a CygnusEd macro. However it means, for instance, that you can't use the arrow keys to move through a macro definition, and you can't use the delete key to delete one line of a macro entry. Instead these keystrokes will get added to the macro definition. Moving around this window must be done with the mouse, by selecting a line or by moving the scroll bar. Deleting lines must be done with the 'Delete entry' button. The one other operation you can do in the right window is to double click on an entry that contains some requester data, such as a numeric value or a filename. This will allow you to edit it in the appropriate requester. See 'Insert Special' for more details.

New

Open... - Keyboard shortcut [Amiga]+o

Save - Keyboard shortcut [Amiga]+w

Save as... - Keyboard shortcut [Amiga]+W

Undo - Keyboard shortcut [Amiga]+u

Print macro list... - Keyboard shortcut [Amiga]+p

About...

Quit - Keyboard shortcut [Amiga]+q

Find/Add short invocation macro - Keyboard shortcut [Amiga]+m

Find/Add long invocation macro - Keyboard shortcut [Amiga]+M

Find/Add startup macro

Delete Macro

Delete Entry

Insert Special

1.137 New

New

The 'New' command clears out the currently loaded macro file. If any changes have made you are asked to confirm before MetaMac proceeds.

1.138 Open... - Keyboard shortcut [Amiga]+o

Open

The 'Open' command is used to load another macro file into MetaMac. Since MetaMac can only have one macro file loaded at a time, the currently loaded macro file is first cleared. If the file has been changed at you are asked to confirm before MetaMac proceeds.

1.139 Save - Keyboard shortcut [Amiga]+w

Save

The ’Save’ command is used to save the macro file that you are currently working on to the file name associated with it. If there is no file name associated with it then the standard ASL file requester is brought up to allow you to select a file name. If you want CygnusEd to automatically load the macro file in when it is run then you should save the macro file as s:cedmacros or as cedmacros in the current directory that you will be running CygnusEd from.

1.140 Save as... - Keyboard shortcut [Amiga]+W

Save as...

The ’Save as’ command is used to the macro file that you are currently working on to a new file. You can save the macros to any file name but CygnusEd will only automatically load the macro file at startup if it is saved as s:cedmacros or as cedmacros in the current directory that you will be running CygnusEd from.

1.141 Undo - Keyboard shortcut [Amiga]+u

Undo

MetaMac has a very limited undo facility. If you make some changes to a macro then you can undo those changes if you have not selected another macro since. All of the changes are undone at once - there is no way to selectively undo some changes to a macro but not others. If you accidentally delete a macro this can not be undone, because you necessarily select another macro and thus clear the undo buffer. If you accidentally delete a valuable macro, your best course of action would probably be to reload the macro file without saving the changes.

1.142 Print macro list... - Keyboard shortcut [Amiga]+p

Print macro list...

You may want to keep a list of all macros handy CygnusEd knows. This command will print the information you see displayed in both MetaMac windows (first the keystrokes that introduce the macro, then the commands this macro will invoke). Alternatively, you can send the macro list to a file rather than to the printer.

1.143 About...

About...

The 'About' command merely brings up a box showing the version number and other fascinating information.

1.144 Quit - Keyboard shortcut [Amiga]+q

Quit

The 'Quit' command tells MetaMac to exit. If you have made changes to the currently loaded macro file, MetaMac asks you to confirm before continuing.

1.145 Delete Macro

Delete Macro

The 'Delete Macro' button tells MetaMac to delete the currently selected macro. No warnings are given if the macro has been changed.

1.146 Delete Entry

Delete Entry

The 'Delete Entry' command is the only way to delete entries from a macro definition, other than deleting the entire macro. The reason that the usual options, such as using the 'Del' key or marking a block and cutting, don't work is because these options are instead recorded into the macro.

1.147 Insert Special

Insert Special

The 'Insert Special' command is used to add requester information to CygnusEd macros. This information is only relevant if it follows a command which brings up a requester. Examples are commands like 'Open', 'Send DOS/ARexx command', 'Search for', 'Jump to line' and 'Select disk font...'. These five commands are not the only commands that this applies to, but they do show off the five different types of requester data that CygnusEd and MetaMac support within macros. These are:

Filename data - 'Open', 'Save as', any requester that just asks for a file or directory name.
Text data - 'Send DOS/ARexx command', any requester that just asks for a single line of text.
Search and replace data - 'Search for', 'Replace', or any other command that makes the search/replace requester appear.

Numeric data - 'Jump to line', 'Jump to byte', any requester that asks for a number.
Font data - 'Select disk font...'.

When you select this command a requester appears asking you which type of data you would like to insert. MetaMac will let you insert any type of requester data, whether it makes sense or not. If it does not make sense in the context of the previous command, CygnusEd will silently ignore it.

When you select the type of data that you would like, a standard CygnusEd requester of the appropriate type appears. Dismiss the requester when you have selected the desired values and the data will be entered into the macro and displayed in the right hand window.

If you wish to edit existing requester values, double click on the line containing the requester values and the appropriate requester will appear.

1.148 Find/Add short invocation macro - Keyboard shortcut [Amiga]+m

Find/Add short invocation macro - Keyboard shortcut [Amiga]+m

This command is used to find an existing short invocation macro (macro that is invoked with just one keystroke) or to create a new one. When you select this command you are asked to type the keystroke used to invoke this macro. If that keystroke is in use, then that macro is selected. Otherwise a new macro is created, attached to that keystroke.

Note that the keyboard shortcut for this is the same as it is in CygnusEd.

1.149 Find/Add long invocation macro - Keyboard shortcut [Amiga]+M

Find/Add long invocation macro - Keyboard shortcut [Amiga]+M

This command is used to find an existing long invocation macro (macro that is invoked with a sequence of keystrokes) or to create a new one. When you select this command you are asked to type the first keystroke used to invoke this macro. After each keystroke you are asked if there anymore. If that sequence of keystrokes is in use, then that macro is selected. Otherwise a new macro is created, attached to that sequence of keystrokes.

Note that the keyboard shortcut for this is the same as it is in CygnusEd.

1.150 Find/Add startup macro

Find/Add startup macro

This command is used to find an existing startup macro or to create a new. If there is already a startup macro it is selected, otherwise an empty startup macro is created. A macro file can have at most one startup macro. Having any empty startup macro is functionally the same as having no startup macro at all.

1.151 RecoverCEDFiles

RecoverCEDFiles

Occasionally while working on your Amiga an errant program may cause your system to crash. If you are working on valuable files when this happens then this can be quite annoying. With most programs you are just out of luck. CygnusEd, however, gives you a second chance. RecoverCEDFiles is a program which will scan memory for any files left in memory by a previous copy of CygnusEd. If RecoverCEDFiles is run immediately after a reboot it will frequently be able to find all of your files still intact in memory, and allow you to save them!

This is not a guaranteed thing, so don't go rebooting with unsaved files loaded, counting on RecoverCEDFiles to find them. Two things can go wrong. One is that the magic cookies which RecoverCEDFiles uses to locate files in memory may get destroyed, in which case the files can not even be located. This can happen because your startup-sequence uses that area of memory, or because you have a memory board that doesn't continue refreshing memory while rebooting.

The other thing that can go wrong is even worse. Sometimes the magic cookies survive, but the file data does not. There is no way that CygnusEd can distinguish good data from corrupted data. CygnusEd does not maintain any data integrity checksum for this purpose. Therefore, when recovering files with RecoverCEDFiles is is VITAL that you save the recovered files to a different file name, and check to see whether the changes from the previous version are the changes that you typed in, or random memory errors or overwrites. If you use RecoverCEDFiles to save over top of the previous version of the file then RecoverCEDFiles may unwittingly participate in the destruction of your data.

To increase the change of RecoverCEDFiles succeeding, it is important to run RecoverCEDFiles as early on in the rebooting process as possible. We recommend that you use [Ctrl]+D to abort your startup sequence, because the loading of commodities and the allocation of disk buffers could easily overwrite the memory that CygnusEd was using, making recovery impossible.

1.152 Views overview

Views overview

CygnusEd allows you to have multiple cooperating views on a single file. All of these views are windows into the same data. Changes in one view are reflected in the others. In fact, if several views are displaying the same area of a particular file, you can see changes happening in all of the views simultaneously.

Although the Amiga's wonderful multitasking abilities mean that there is technically no need for a program to handle multiple files simultaneously (after all, you can always just run another copy of the program) it turns out that is frequently very useful to have many files loaded into a simple program at one time. It simplifies window management, cutting and pasting, and keeping everything together on one screen. CygnusEd allows you to load multiple files, and it also allows you to have multiple views on one file. So, for instance, you can easily be looking at the opening paragraph of your doctoral thesis in one view while looking at a supporting paragraph on page 97 in another view. Because the two views are actually looking at the same document, changes made to one view are also made to the other one - you never need to worry about them getting out of sync.

CygnusEd's text output routines are so fast, especially when using an eight by eight font such as Topaz80, that you can have ten different views displaying the same lime of text, type as fast as you can, and have all ten views show the changes, without ever falling behind. And all this works on a 68000 based Amiga 1000!

You can clearly recognize when you have multiple views on a single file by the #1, #2, etc. text that appears at the left edge of the status bar if you are displaying file names in the status bar.

These are the commands for setting view specific settings.

Status line

White spaces

Scroll bar

Set scroll borders...

These are the command for doing view operations, such as creating and destroying views and moving between views.

View operations

Previous view - Keyboard shortcut [Amiga]+,

Next view - Keyboard shortcut [Amiga]+.

Split view - Keyboard shortcut [Amiga]+d

Expand view - Keyboard shortcut [Amiga]+]

Grow view

Shrink view

1.153 Ed

Ed

Ed is a command line or workbench program that is used to invoke CygnusEd. The advantage of using Ed rather than using CygnusEd directly is that if CygnusEd is already running, Ed will tell it to load the requested files. If CygnusEd is not running, Ed will run it and tell it to load the requested files. That way, you never need to worry about whether or not CygnusEd is running.

The basic syntax for Ed is that you can specify any number of file names, with optional wildcards, and you can specify a number of command line arguments, starting with a dash, to modify the treatment of those names. For example:

```
  ed #?.c
  ed  s:user-startup   source:ced/main.c
```

Unless otherwise specified, you can always specify as many files as desired, and you always specify file specific options after the file you want them to affect.

-a Activate previously active file

The '-a' command is the only option in this section that does not apply to a specific file. The '-a' option tells CygnusEd to load in the specified files, and then move the cursor back to where it was before Ed was invoked. This is particularly useful if you have a script which will be loading files into CygnusEd at unpredictable times - this ensures that the cursor won't be yanked

-f Force loading

The '-f' option is a powerful but dangerous option. It tells CygnusEd to load the specified file, overwriting the file if it is already loaded into CygnusEd. This option is dangerous because the previously loaded file will be lost, even if there were unsaved changes made to it, without asking the user. '-f' stands for 'force'.

-sticky
-s

There are some applications where an application needs to let the user edit a file, and then needs to do something with the file when they have finished editing it. Usually these programs let the user specify what editor they would like to use, and many people choose CygnusEd for this purpose. However the default behaviour for Ed, the preferred way to invoke CygnusEd, is to return immediately, without waiting for the user to finish editing the file. Normally this behaviour is what is desired - but not in this case. This is precisely what the '-sticky' option was made for. If you go 'Ed -sticky filename' then CygnusEd will be run if necessary, the specified file will be loaded into the new copy of CygnusEd, or an existing copy if there is one, and Ed will not return until the user finishes editing the specified file. The user can indicate that they are finished editing the file either by closing all views on the file (with the Quit command) or by loading another file on top of the file (with the Open command). Only one file name can be specified on the command line when using the '-sticky' command. '-s' is a synonym for '-sticky'. Unlike most of the file specific options, the -sticky command can go either before the file name or after. This means that you can define the name of your editor to be 'ed -sticky' and let the other application tack the file name on at the end.

Please note that this feature works only if you specify a single file name on the command line. Multiple 'sticky' files are not supported.

-i Ignore

The '-i' command line option tells CygnusEd to load the specified file in if it isn't already loaded. If the file is already loaded it tells CygnusEd to move the cursor to the view containing that file. This is particularly useful for when compiler's need to ensure that a particular file is loaded so that they can place the cursor on a syntax error. If you don't specify this option, or '-o' described below, then CygnusEd will warn you that the file is already loaded and will ask you to confirm before proceeding. The 'i' stands for 'ignore', as in 'ignore the file on disk if it's already loaded'.

-o Overwrite

The '-o' command line option tells CygnusEd to load the specified file in even if is already loaded. If the file is already loaded it tells CygnusEd to overwrite the previously loaded copy. If any changes have been made you are asked to confirm before CygnusEd continues. If you don't want the requester to appear, see the '-i' or '-f' options.

-v View only

The '-v' command tells CygnusEd to load the file to mark it as non-editable. This setting can be changed after the file is loaded with the 'Editable file?' menu item under the Environment, File settings menu.

Different options can be combined as long as they don't contradict each other. If CygnusEd is not yet in memory, the CygnusEd command line options can be used also. For example:

```
ed file1 -va
```

This tells CygnusEd to load in file1, make it non editable and then reactivate the previously active view.

```
ed file.c -i ram:err/file.err -fa
```

This tells CygnusEd to load in file.c, if it isn't already loaded, then load in ram:err/file.err, overwriting any previously loaded version, then put the cursor back to the previously active view. This line ensures that the needed source file is loaded, without losing any changes, and it ensures that the most recent error file is loaded, overwriting changes if necessary.

See also:
Command line options

1.154 Command line options

Command line options

When running CygnusEd there are a number of command line options that you can specify to alter its behaviour. All of these can also be used if you are using Ed, the CygnusEd invoker, to run CygnusEd. Some of the options don't make sense if you are using Ed to pass new file names to an already running copy of CygnusEd, and are therefore ignored. However we recommend that you always use Ed to run CygnusEd and to load files into CygnusEd, because that way you never need to worry about whether CygnusEd is already running or not.

If you use the command line options below with 'Ed' then they will only have an effect if CygnusEd has not yet been loaded into memory. If CygnusEd has already been loaded into memory they will be ignored. These options can be specified together with file names, except for -r.

-r

If you use CygnusEd on a regular basis you will probably want to put it into your user-startup script so that CygnusEd is automatically run for you each time you reboot your Amiga. Normally when you run CygnusEd its screen immediately opens up, but that is probably not what you want when running CygnusEd from the user-startup script. The '-r' option tells CygnusEd to run but to go immediately into dormant mode - 'R'esident mode. This means that CygnusEd has loaded in and is ready to start editing on a seconds notice, but has not yet opened its screen. You can make CygnusEd appear either by typing the [Alt]+[Shift]+Return hot-start sequence, or by using Ed to wake up CygnusEd, and possibly pass in the names of some files to be loaded. If you specify some file names on the same command line as the '-r' option then CygnusEd ignores the '-r' option, opens the screen and displays the files.

-keepio

Normally when you run CygnusEd from a CLI it disconnects itself completely from the CLI, allowing you to close the CLI at any point. Normally this is what you want. However it means that if you run any DOS commands from within CygnusEd, a new CLI window will have to be opened up to display the output of the commands. If you would prefer that the output of DOS commands run from CygnusEd go to the CLI that CygnusEd was started from, specify the '-keepio' option when first running CygnusEd. The '-keepio' option can be used with the Ed command, but will only have an effect if CygnusEd has not yet been run - once CygnusEd has detached itself from the CLI it can never reattach itself. If you use the '-keepio' command then the CLI that CygnusEd was run from can not be close until CygnusEd exits completely.

-pubscreen=

CygnusEd can be made to open up on a public screen by using the environment menu and then saving the environment. However if you want to be able to specify what public screen CygnusEd should open up on when you run CygnusEd, you can use the '-pubscreen=' command. The name of the public screen that you want CygnusEd to use should appear directly after the equals sign, with no intervening spaces. Case may be significant, depending on what version of AmigaOS you are running.

PORTNAME=

This option allows you to override the name of the ARexx port CygnusEd should use instead of the default.

SETTINGS=

By default, CygnusEd tries to read three settings files (ceddefaults, cedmacros and RexxCommands) first from the current directory, then it looks into 'S:'. With this option you can tell the program which defaults file to read, CygnusEd will then look for the remaining two settings files (cedmacros and RexxCommands) in the same directory.

See also:
Ed - the CygnusEd invoker

1.155 ARexx

ARexx

DOS/ARexx interface

One of the things that makes the Amiga an extremely powerful and versatile computer is its support of a common scripting language which is designed to handle communication between multiple programs. CygnusEd has a powerful ARexx interface which allows it to control other programs, be controlled by other programs, or it allows you to control it more precisely, by writing your own complex macros in ARexx and executing them from within CygnusEd. By executing these ARexx scripts from within macros that can be attached to any key, you can completely remap CygnusEd's keyboard. Some people have even gone to the extreme of using ARexx and CygnusEd to implement small databases or data entry programs.

This portion of the documentation assumes some familiarity with ARexx programming. Although it is possible to figure out how to write ARexx programs for CygnusEd just by reading this documentation, and by looking at the sample ARexx scripts, it is much easier if you get a copy of one of the excellent ARexx programming manuals.

ARexx is a powerful scripting language that has been almost universally adopted on the Amiga. It allows different programs to talk to each other and it allows the end user to add new features to programs that the original authors never thought of.

CygnusEd was one of the first applications to contain an ARexx interface. CygnusEd's ARexx interface is very powerful and lets you attach

ARexx scripts and DOS commands to any key on the keyboard. These scripts, or other scripts, can then control CygnusEd, or other programs.

CygnusEd’s ARexx commands are, by and large, the very same commands which you see in the menus. Exactly the same wording as what you see. For instance, to clear the current file:

```
CLEAR
```

To save the current file under a new name:

```
SAVE AS NewName.txt
```

As you can see, ARexx programming for CygnusEd needn’t be difficult.

When specifying menu commands from ARexx, case is not important. Keyboard shortcuts, such as [Amiga]+A, or [Ctrl]+Esc, should not be specified. Neither should the ellipses, which indicate that requesters will appear.

If you wish to send commands to CygnusEd from a script that was run from CygnusEd then you don’t need to worry about the name of CygnusEd’s ARexx message port - your commands will automatically be sent to it. However, if you want to invoke your scripts from outside of CygnusEd, you need to be able to tell ARexx whom you want to talk to. The default port name for CygnusEd is ’CYGNUSED’. For compatibility with older ARexx scripts, the older naming scheme ’rexx_ced’..’rexx_ced9’ is still supported, though. If you use the older naming scheme you must be very careful to enclose the port name in quotes, or else ARexx will upper case it and communication will fail. If multiple copies of CygnusEd are run, the additional port names will be of the form ’CYGNUSED.1’, ’CYGNUSED.2’, etc. Invoke the About command to see the current port name.

Results are returned from ARexx programs in the Result variable. However this is only done if you have enabled returning of results with the ’OPTIONS RESULTS’ ARexx command. Additionally, if an error occurred in a command then the RC variable will be set, and Results will be undefined.

If you wish to add a new command to CygnusEd, by making an ARexx command that can be executed from within CygnusEd, one way is by using the ’Install DOS/ARexx command’ command. To use this command, select it from the Special, DOS/ARexx Interface menu, choose a command slot number from one to ten, then enter the name of the ARexx command. The advantage to this method is that the ARexx command actually gets added to the menus. Don’t forget to save your ARexx command definitions afterwards.

The other way allows you to attach an ARexx script to any key on the keyboard. Begin by defining a macro attached to the desired key. Then use the ’Send DOS/ARexx command’ command to send the desired command. You will be asked whether you want the contents of the requester added to the macro - say yes, and end the macro definition. Don’t forget to save your macros afterwards.

Many of the commands in CygnusEd’s menus bring up requesters. Obviously an ARexx script isn’t much use if it can’t fill in the values required by those requesters without bothering the user. CygnusEd’s ARexx interface allows you to respond to these requesters in a simple, orthogonal way. All file requesters are responded to identically. All yes/no requesters are responded to identically, and so on. Any time you choose not to supply data for a requester, the requester will come up. If there is a situation where a requester may or may not come up it is best to always supply data for the requester, as it will be ignored if not needed.

Some commands will bring up two requesters, such as the open command when the current file is changed, is non-zero in length and has no other views. In this case a yes/no requester will come up asking if it's okay to proceed and lose changes. If this is answered yes to then a file requester will appear, asking for the new file name. If you detect this condition in ARexx you can invoke the Open command like this:

```
  OPEN 1 'ram:newfile.txt'
```

Note that if you do this when the above conditions do not apply then the yes/no requester will not be called and the '1' will be used for the new file name - generally not what you want.

Yes/no and 'okay' requesters:

The response to these requesters can be given by putting either '1' or '0' after the command' '1' means yes, '0' means no. Either response is legal for a single gadget 'okay' requester.

Filename and directory name requesters:

Commands, such as Open, Save as, etc., that require a file name should simply have the file name, typically enclosed in quotes after the command name. For instance:

```
  SAVE AS 'ram:file.txt'
```

Numeric requesters:

Numeric requesters, brought up by such commands as 'ENTER ASCII', should be supplied with a numeric argument in the legal range for that particular command. For instance:

```
  ENTER ASCII 10
```

String requesters:

String requesters, brought up by such commands as 'SET ICON TOOL NAME' should be supplied with the desired string, typically enclosed in quotes. For instances:

```
  SET ICON TOOL NAME 'C:ed'
```

Font requesters:

If you wish to change CygnusEd's font from an ARexx script you can use the 'SELECT DISK FONT' command. Simply follow the command with the name and size of the desired font. For instance:

```
  SELECT DISK FONT emerald.font 12
```

Search/replace requesters:

Search data needs to be supplied for the ’SEARCH FOR’ command. The ’SEARCH FOR’ command can be specified with no arguments, in which case the search requester will appear. It can be specified with just a string to search for, or it can be specified with a string and five boolean parameters. The five booleans, which must be specified in order as either 1 or 0, control the four check boxes in the search/replace requester. They are, in order - case sensitivity, wild cards, forwards/backwards, only words. For instance:

```
  SEARCH FOR
  SEARCH FOR "Hello"
  SEARCH FOR "Hello" 1 0 1 1
```

The first variation brings up the search requester. The second variation searches for the string Hello, using the current search flags. Note the quotes to prevent ARexx from upper casing the string. The third variation searches for Hello after setting the flags to: case insensitivity on, wild cards off, forwards on and only words on.

The replace command has a similar syntax, with four different variations:

```
  REPLACE
  REPLACE "Hello" "Goodbye"
  REPLACE "Hello" "Goodbye" 1 0 1 1
  REPLACE "Hello" "Goodbye" 1 0 1 1 1 t
```

The first variation brings up the replace requester. The second variation will search for the first instance of "Hello" and replace it with "Goodbye", using the current search flags. The third variation is the same as the second except that it specifies the search flags. The fourth variation specifies ’t’ for ’turbo’ replace. This allows you to do a global replace in one command. The other option for where the ’t’ is is ’g’ for ’global’ replace. Note: When specifying the ’t’ or ’g’ option you MUST specify a fifth boolean variable and it MUST be 1.

1.156 Detailed ARexx reference

```
                The general rules
=================
```

Two general rules apply to all CygnusEd ARexx commands. These are:

1. All commands set RC and RESULT as indicated above.

2. All commands are case-insensitive. However, arguments may be case sensitive

If a command normally brings up one of the information requesters (like the "All macro definitions cleared" message after using the "CLEAR DEFINITIONS" command) or one of the notify or response requesters (like the "n changes have been made to file. They will be lost. Ok to continue" that shows up if you try to exit without saving), you can get ARexx to respond to the requester for you.

If you put a 1 (for yes) or a 0 (for no) after the command that will normally bring up the requester, the 1 or 0 will make the requester not appear, and CygnusEd will act as if you hit the corresponding button.

Therefore, if you want to CLEAR the current file and you don’t care if you lose any changes, go like this:

```
   CLEAR 1
```

Creative quoting

ARexx always strips at least one set of quotation marks from all strings. Therefore, in order to provide for the proper treatment of strings with imbedded spaces, some creative quoting may be needed.

For example:

```
   GETFILENAME '"Default name"' '"The Title"'
```

This sends "Default Name" and "The Title" to CED.

ARexx variables which might have imbedded spaces in their contents must be handled carefully. For example:

```
TheName = "The FileName"
TheTitle = "The Title"
GETFILENAME '"'TheName'"' '"'TheTitle'"'
```

This makes sure that ARexx gets to strip off a set of quotation marks, causes its variables to be replaced with their contents, and still provides for a set of quotation marks around the resulting strings.

ARexx command reference

The following list contains those ARexx commands whose functionality has not already been covered by the menu descriptions. Every single menu command is also available from ARexx: just enclose its label in quotes as in ’shift in marked’. So if you find something missing in the following list, look up the menu descriptions.

ACTIVATE NEXT CED

ADJUST COLOURS

AUTO-EXPAND VIEWS

AUTO-INDENT

BACKSPACE

BACKTAB

BACKUP TO *.BAK

```
BCK SPC WORD

BEG OF FILE

BEG OF LINE

BEG OF SCREEN

BEG/END DEFINITION

BLOCK TO SEARCH BUFFER

CEDTOBACK

CEDTOFRONT

CENTER CURSOR

CENTER LINE

CHANGE CASE BLOCK

CHANGE CASE LETTER

CHANGE CASE MARKED

CHANGE CASE WORD

CHANGE CURRENT DIRECTORY

CHANGES

CHECK FILE

CLEAR

CLEAR DEFINITIONS

CLIP TO SEARCH BUFFER

COPY

COPY BLOCK

CUSTOMIZE TABS

CUT

CUT BLOCK

CYCLE COLOURS

DEFAULT

DELETE

DELETE LINE
```

DELETE TO EOL

DELETE WORD

DM

DOWN

DOWN 12 LINES

EDITABLE FILE

END OF FILE

END OF LINE

END OF SCREEN

ENTER ASCII

EOLS VISIBLE

ESC CODES VISIBLE

EVERY N MIN

EXPAND VIEW

FIND MATCHING BRACKET

FORCE CUSTOM SCREEN

FORWARDTAB

GETCHAR

GETDIRNAME

GETFILENAME

GETLINE

GETNUMBER

GETSTRING

GETWORD

GROW VIEW

HOT-START ENABLED

ICON CREATION

INCLUDE FILE

```
INHERIT

INSERT BLOCK

INSERT MODE

INSTALL DOS/AREXX COMMAND

JUMP TO AUTO-MARK

JUMP TO BYTE

JUMP TO FILE

JUMP TO LINE

JUMP TO MARK

JUMPTO

KEYPAD = MOVEMENT

LASTKEY

LAYOUT

LEFT

LEFT 12 CHARS

LL

LOAD DEFINITIONS

LOAD DOS/AREXX COMMANDS

LOCKGUI

LOWER CASE WORD

MARK

MARK BLOCK

MARK COLUMNAR

MARK COLUMNAR BLOCK

MARK LOCATION

MAX SCROLL

MAX UNDO LEVELS

MAX UNDO MEMORY

MENU
```

```
NEXT VIEW

NEXT WORD

NO SCROLL BAR

OKAY1

OKAY2

ON/OFF

OPEN

OPEN NEW

OW

PAGES

PASTE

POST PERIOD SPACES

PREV WORD

PREVIOUS VIEW

PRINT BLOCK

PRINT CLIP

PRINT FILE

PRIORITY

PUBSCREEN

QUIT

QUIT & DIE

RAWKEY

REDO

REPEAT KEY/MENU

REPEAT REPLACE

REPEAT SEARCH BACKWARDS

REPEAT SEARCH FORWARDS

REPLACE
```

RESET ALIAS

RIGHT

RIGHT 12 CHARS

ROT BLOCK

ROT MARKED

RX

SAFE SAVES

SAVE

SAVE ALL CHANGES

SAVE AS

SAVE BLOCK TO FILE

SAVE CLIP AS

SAVE DEFINITIONS

SAVE DOS/AREXX COMMANDS

SAVE ENVIRONMENT

SCREEN HEIGHT

SCREEN WIDTH

SCROLL BAR ON LEFT

SCROLL BAR ON RIGHT

SEARCH FOR

SELECT DISK FONT

SEND DOS/AREXX COMMAND

SEND DOS/AREXX OUTPUT TO

SET CLIPBOARD UNIT

SET ICON TOOL NAME

SET PRIORITY

SET RIGHT BORDER

SET SCREEN SIZE AND TYPE

SET SCROLL BORDERS

```
SET SCROLL JUMP

SET TIMER

SETSCREEN

SHOW ASCII VALUES

SHRINK VIEW

SIMPLE SAVES

SPACES VISIBLE

SPAWN NEW CED

SPECIFY

SPLIT VIEW

STATUS

STRIP CR BLOCK

STRIP CR MARKED

TAB SIZE

TABS = SPACES

TABS VISIBLE

TEXT

TOPAZ 60 COLUMN

TOPAZ 80 COLUMN

UNBCK SPC WORD

UNDELETE LINE

UNDELETE WORD

UNDO

UNLOCKGUI

UP

UP 12 LINES

UPPER CASE WORD

USE WBENCH COLOURS
```

```
                 VERSION

                 WITH FILL

                 WITHOUT FILL

                 WORD WRAP
```

1.157 ACTIVATE NEXT CED

```
Syntax: ACTIVATE NEXT CED

This command brings the next spawned copy of CED, if one exists, to the
front of all other screens or windows and makes it active.

If there are multiple copies of CED running, issuing the ACTIVATE NEXT CED
via ARexx will return the ARexx port name of this next copy of CED in the
RESULT variable. Otherwise, RESULT will be set to 0.

Result: ARexx port of next copy of CED
        0 if no other copy could be found
```

1.158 ADJUST COLOURS

```
Syntax: ADJUST COLOURS

This command will display the colour requester for you to change the
screen’s colour scheme. There is currently no way to specify colours from
ARexx.

Result: always 1
```

1.159 AUTO-EXPAND VIEWS

```
Syntax: "AUTO-EXPAND VIEWS"

This command will toggle the status of the "Auto-Expand Views?" mode.

Note that the quotes around the command are necessary due to the hyphen in
the command name. Failure to place this command in quotes will result in an
ARexx syntax error.

You can query the current state of the toggle using the "STATUS AUTOEXPAND"
command.

Result: 1 if auto-expand view mode is switched on
        0 if auto-expand view mode is switched off
```

1.160 AUTO-INDENT

```
Syntax: "AUTO-INDENT"

This command simulates typing Shift+Return.

Note that the quotes around the command are necessary due to the hyphen in
the command name. Failure to place this command in quotes will result in an
ARexx syntax error.

Result: always 1
```

1.161 BACKSPACE

```
Syntax: BACKSPACE

This command is the same as pressing the Backspace key. It will move the
text cursor one character to the left deleting the character that was there.

Result: always 1
```

1.162 BACKTAB

```
Syntax: BACKTAB

This command is requivalent to [Ctrl]+[Alt]+Left. It will move the text cursor
left to the next tab stop. The text cursor will not move off the current
line.

Result: always 1
```

1.163 BACKUP TO *.BAK

```
Syntax: "BACKUP TO *.BAK"

This command enables "Backup to *.bak" mode. Note that this command must
always be enclosed in quotation marks because of the embedded asterisk.

You can find out which file save method is currently being employed by using
"STATUS SAFESAVES". This ARexx command will return an integer ranging
between 0 and 2 corresponding to which type of file saving will be
performed.

Result: always 1
```

1.164 BCK SPC WORD

```
Syntax: BCK SPC WORD

This command will execute the equivalent of [Ctrl]+Backspace and will delete
the word to the left of the cursor.

Result: always 1
```

1.165 BEG OF FILE

```
Syntax: BEG OF FILE

This command moves the cursor to the beginning of the current file

Result: always 1
```

1.166 BEG OF LINE

```
Syntax: BEG OF LINE

This command moves the cursor to the beginning of the current line.

Result: always 1
```

1.167 BEG OF SCREEN

```
Syntax: BEG OF SCREEN

This command moves the cursor to the top of the current scrolling area.

Result: always 1
```

1.168 BEG/END DEFINITION

```
Syntax: None

Due to changes in the way macros are defined, this command is not supported
anymore through ARexx. There is no other equivalent ARexx command.
```

1.169 BLOCK TO SEARCH BUFFER

Syntax: BLOCK TO SEARCH BUFFER

This command is simimlar in function to the newer "CLIP TO SEARCH BUFFER" command, but has been preserved for compatibility with scripts created with previous versions of CED. Please consult the "CLIP TO SEARCH BUFFER" command for specific information.

Result: always 1

1.170 CEDTOBACK

Syntax: CEDTOBACK

If CED is running as a window on the Workbench, its window will be pushed behind all other windows on the screen. If CED is running on its own screen, its screen will be pushed behind all other screens.

Result: always 1

1.171 CEDTOFRONT

Syntax: CEDTOFRONT

If CED is running as a window on the Workbench, its window will be brought to the front and activated. If CED is running on its own screen, its screen will be brought to the front and activated.

Result: always 1

1.172 CENTER CURSOR

Syntax: CENTER CURSOR

This command scrolls the current line to the center of the view.

Result: always 1

1.173 CENTER LINE

Syntax: CENTER LINE

This command horizontally centers the current line within the current left and right borders

Result: always 1

1.174 CHANGE CASE BLOCK

```
Syntax: CHANGE CASE BLOCK

This command is similar in fucntion to the newer "CHANGE CASE MARKED"
command, but has been preserved for compatiblity with scripts created with
previous versions of CED. Please consult the "CHANGE CASE MARKED" command
for specific information.

Result: always 1
```

1.175 CHANGE CASE LETTER

```
Syntax: CHANGE CASE LETTER

This command inverts the case of the character underneath the text cursor.

Result: always 1
```

1.176 CHANGE CASE MARKED

```
Syntax: CHANGE CASE MARKED

This command inverts the case of each letter in the currently
defined block.

Result: 0 if block is defined
        1 if no block is defined
```

1.177 CHANGE CASE WORD

```
Syntax: CHANGE CASE WORD

This command invertes the case of each letter in the word underneath and to
the right of the text cursor.

Result: always 1
```

1.178 CHANGE CURRENT DIRECTORY

```
Syntax: CHANGE CURRENT DIRECTORY
        CHANGE CURRENT DIRECTORY directoryname

The first form will cause a directory requester to appear for you to select
a new current director.

The second form will cause CED to change its current directory to the named
```

```
directory without putting up any requester.

You can use the "STATUS CURRENTDIR" command to query the value of the
current directory.

Result: 1 if current directory was changed
        0 if current directory was not changed
```

1.179 CHANGES

```
Syntax: CHANGES

This command will cause the alternate status line to be displayed and set to
the "changes" style.

Result: always 1
```

1.180 CHECK FILE

```
Syntax: CHECK FILE filename

This command will determine if a given file (you should specify the full
path to the file, in addition to the actual name of the file, in filename)
is loaded into one of CED's views.

This command goes well with the "OW" command. Pass "CHECK FILE" a filename
and if the file is in the editor, it will activate its view; if it isn't,
then it will try to load the file.

Result: 1 if file is loaded
        0 if file is not loaded
```

1.181 CLEAR

```
Syntax: CLEAR
        CLEAR 0
        CLEAR 1

The first form will cause the "CLEAR" command to be executed on the current
view. If the execution of the command generates a requester, the user will
have to answer the requester manually.

The second form will cause the "CLEAR" command to be executed on the current
view. If the execution of the command would have generated a requester, the
requester is automatically anwered with a negative response and the clear
operation is cancelled. No requester will actually be shown.

The third form will cause the same effect as the second form, except that a
positive response will be given.
```

```
Result: 1 if view was cleared
        0 if view was not cleared
```

1.182 CLEAR DEFINITIONS

```
Syntax: CLEAR DEFINITIONS

This command erases all macro definitions currently defined.

Result: always 1
```

1.183 CLIP TO SEARCH BUFFER

```
Syntax: CLIP TO SEARCH BUFFER

This command copies the contents of the current clipboard unit to the search
buffer.

Result: 1 if search buffer was updated
        0 if the clipboard was empty.
```

1.184 COPY

```
Syntax: COPY

This causes the paste buffer to assume the contents of the block currently
being defined. The block in the text is not disturbed. If no block is
currently being defined when this command is executed, then the contents of
the paste buffer is undisturbed.

Result: 1 if the block was copied to the clipboard
        0 if no block was defined
```

1.185 COPY BLOCK

```
Syntax: COPY BLOCK

This command is similar in function to the newer "COPY" command, but has
been preserved for compatibility with scripts created with previous versions
of CED. Please consult the "COPY" command for specific information.
```

1.186 CUSTOMIZE TABS

```
Syntax: CUSTOMIZE TABS
        CUSTOMIZE TABS tab_settings
```

The first form causes the "customize tabs" requester to appear soliciting a new set of tab stops.

The second form causes the new tab stops to be taken from the supplied string which is composed of either dashes (where no tab stop is desired) or with "T"s (where tab stops are desired).

You can receive a string composed of dashes and "T"s by executing "STATUS TABS".

Result: always 1

1.187 CUT

Syntax: CUT

This causes the paste bufer to assume the contents of the block currently being defined. The block itself is deleted from the text. If no block is currently being defined when this command is executed then the contents of the pase buffer are undisturbed.

```
Result: 1 if the block as cut to the clipboard
        0 if no block was defined
```

1.188 CUT BLOCK

Syntax: CUT BLOCK

This command is similar in function to the newer "CUT" command, but has been preserved for compatibility with scripts created with previous versions of CED. Please consult the "CUT" command for specific information.

1.189 CYCLE COLOURS

Syntax: CYCLE COLOURS

This command will cause the values of the four colour registers used by CED to interchange amongst themselves in a way which only author of CED understands. Suffice to say, however, that if you do this command 24 times you get back to the original combination of four colours.

```
Result: 1 if colours were cycled
        0 if colours cannot be cycled
```

1.190 DEFAULT

```
Syntax: DEFAULT

This command will cause the default name (with a base name of
"S:CEDDefaults") based on the current view's filename extension to be
loaded.

Result: always 1
```

1.191 DELETE

```
Syntax: DELETE

This command is the same as pressing the Del key. It will erase the
character to the right of the text cursor.

Result: always 1
```

1.192 DELETE LINE

```
Syntax: DELETE LINE

This command will remove the current line and place it in the line buffer.

The contents of the global line buffer can be read from ARexx using the
command "STATUS DELETELINEBUFFER".

Result: always 1
```

1.193 DELETE WORD

```
Syntax: DELETE WORD

This will execute the equivalent of [Ctrl]+Del and will delete the word to the
right of the cursor.

Result: always 1
```

1.194 DELETE TO EOL

```
Syntax: DELETE TO EOL

This command will delete the text from the cursor position to the end of the
current line and place it in the line buffer.

The contents of the global line buffer can be read from ARexx using the
```

```
command "STATUS DELETELINEBUFFER".

Result: always 1
```

1.195 DM

```
Syntax: DM
        DM message

The first from resets CED’s title bar to the default message (i.e. the
program’s name, version number and copyright notice).

The second form displays the specified message in CED’s title bar.

This command is directly compatible with the commands used in SAS/C’s scmsg
program.

Result: 1 if message was displayed/reset
        0 if message was not displayed/reset
```

1.196 DOWN

```
Syntax: DOWN

This command is the same as pressing the cursor down key. It will move the
text cursor down within the same column if possible.

Result: always 1
```

1.197 DOWN 12 LINES

```
Syntax: DOWN 12 LINES

This command will move the text cursor down 12 lines, or as far as possible
if less than 12 lines exist between the current line and last line in the
file.

Result: always 1
```

1.198 EDITABLE FILE

```
Syntax: EDITABLE FILE

This command will set the RESULT variable to a boolean value indicating the
new state of the toggle.

You can query the state of the "Editable file?" flag by using the Arexx
```

```
command "STATUS EDITABLE".

Result: 1 if file is made editable
        0 if file is made uneditable
```

1.199 END OF FILE

```
Syntax: "END OF FILE"

This command will move the cursor to the last cursor position in the file.
Note that the quotation marks are required because "END" is a keyword in
ARexx.

Result: always 1
```

1.200 END OF LINE

```
Syntax: "END OF LINE"

This command will move the cursor to the rightmost position of the current
line. Note that the quotation marks are required because "END" is a keyword
in ARexx.

Result: always 1
```

1.201 END OF SCREEN

```
Syntax: "END OF SCREEN"

This command moves the cursor to the ottom of the current scrolling area.
Note that the quotation marks are required because "END" is a keyword in
ARexx.

Result: always 1
```

1.202 EOLS VISIBLE

```
Syntax: EOLS VISIBLE

This command toggles the state of the "EOLs visible?" menu subitem.

From ARexx the state of the toggle can be queried using "STATUS VISIBLEEOL".

Result: 1 if eols vislble mode is switched on
        0 if eols vislble mode is switched off
```

1.203 ENTER ASCII

```
Syntax: ENTER ASCII
        ENTER ASCII n

The first form will cause the CED numeric requester to appear to solicit the
decimal value of the character you wish so insert.

The second form will enter the ASCII equivalent of the decimal value n. No
requester will appear.

Result: 1 if value was entered
        0 if value was not entered
```

1.204 ESC CODES VISIBLE

```
Syntax: ESC CODES VISIBLE

This command toggles the state of the "ESC codes visible?" menu subitem.

From ARexx the state of the toggle can be queried using "STATUS VISIBLEESC".

Result: 1 if esc codes visible mode is switched on
        0 if esc codes visible mode is switched off
```

1.205 EVERY N MIN

```
Syntax: EVERY N MIN

This command toggles on and off whether or not you will be reminded to save
your work every n minutes.

Result: 1 if auto-save timer is switched on
        0 if auto-save timer is switched off
```

1.206 EXPAND VIEW

```
Syntax: EXPAND VIEW

This command attempts to enlarge the current view’s visible size to its
maximum size.

Result: always 1
```

1.207 FIND MATCHING BRACKET

```
Syntax: FIND MATCHING BRACKET

This command will attempts to find a corresponding token for the token
underneath the cursor.

Result: 1 if a matching token exists
        0 if no matching token exists
       -1 if no token is under the text cursor
```

1.208 FORCE CUSTOM SCREEN

```
Syntax: None

Due to changes in the way screens are invoked, this command is not supported
anymore through ARexx. Please use either the "SET SCREEEN SIZE AND TYPE" or
"SETSCREEN" command instead.
```

1.209 FORWARDTAB

```
Syntax: FORWARDTAB

This command is the same as pressing [Ctrl]+[Alt]+Right. It will move the text
cursor forward to the next tab position. It will not move the text cursor
beyond the current end of line nor will it move the text cursor to the next
line.

Result: always 1
```

1.210 GETDIRNAME

```
Syntax: GETDIRNAME
        GETDIRNAME defaultdir
        GETDIRNAME defaultdir title

The first form displays a directory requester from which a directory can be
selected. No default directory is given.

The second form displays a directory requester with the contents of the
given ‘defaultdir’ directory shown.

The third form displays a directory requester with the contents of the given
‘defaultdir’ directory shown and the text ‘title’ in the requester’s title
bar.

Result: name of selected directory, if one was selected
        blank or RESULT if no directory was selected
```

1.211 GETFILENAME

```
Syntax: GETFILENAME
        GETFILENAME defaultname
        GETFILENAME defaultname title

This command allows you to use the CED file requester to solicit file names
for your own purposes.

The first form displays a file requester from which a file can be selected.
No default filename is given.

The second form displays a file requester with the specified file
defaultname) as the default filename. The default directory will be CED's
idea of the current directory (see "CHANGE CURRENT DIRECTORY") if the
specified filename does not contain any contrary path information.

The third form displays a file requester with the specified file
(defaultname) as the default filename. The file requester has text, as
defined by title, on its title bar. The default directory will be CED's idea
of the current directory (see "CHANGE CURRENT DIRECTORY") if the specified
filename does not contain any contrary path information.

Result: selected filename, including path if file was selected
        blank or RESULT if file was not selected
```

1.212 GETNUMBER

```
Syntax: GETNUMBER [default [[title] [[min] [max]]]]

This command gives access to CED's numeric requester. Depending on how many
parameters you provide, input will be collected differently:

   default - Default value displayed in the requester

   title - The requester title to display, this must be the
           second parameter if any.

   min - Minimum acceptable value to be entered, this must be
         the third parameter if any.

   max - Maximum acceptable value to be entered, this must be
         the fourth parameter if any.

Result: entered number
        blank, if requester was cancelled
```

1.213 GETSTRING

```
Syntax: GETSTRING [[default value] [default title]]

This command gives access to CED's text requester. Depending on how many
```

```
parameters you provide, input will be collected differently:

   No parameter - Displays the text requester with a default
                  title and no default value.

   default value - Displays the text requester with the given
                   value and a default title.

   default title - Displays the text requester with the given
                   value and the given title; this must be the
                   second parameter if any.

Result: entered text
        blank or RESULT if requester was cancelled.
```

1.214 GETWORD

```
Syntax: GETWORD

This command retrieves the word under the cursor. If the cursor is above a
whitespace character (space, tab, carriage return) a NULL string is
returned.

Result: word under the cursor
        0 if there is no word under the cursor
```

1.215 GETLINE

```
Syntax: GETLINE

This command retrieves the contents of the entire line under the cursor.
If the line is empty, an empty string will be returned. Please note that
the line does not include the terminating 'end of line' character (e.g.
line feed, carriage return).

Result: contents of line under the cursor
```

1.216 GETCHAR

```
Syntax: GETCHAR

This command retrieves the character under the cursor. If the cursor is
above the end of the file or a NULL character, a blank string will
be returned.

Result: word under the cursor
```

1.217 GROW VIEW

```
Syntax: GROW VIEW

This command attempts to enlarge the current view by one line.

Result: 1 if view was changed in size
        0 if view did not change in size
```

1.218 HOT-START ENABLED

```
Syntax: "HOT-START ENABLED"

This command toggles the state of the "Hot-start enabled?" menu item.

Note that the quotes around the command are necessary due to the hyphen in
the command name. Failure to place this command in quotes will result in an
ARexx syntax error.

You can query the current state of the toggle using the "STATUS
KEEPRESIDENT" command.

Result: 1 if Hot-start enabled mode is switched on
        0 if Hot-start enabled mode is switched off
```

1.219 ICON CREATION

```
Syntax: ICON CREATION

This command toggles the state of the "Icon creation?" menu item.

You can query the current state of the toggle using the "STATUS
ICONCREATION" command, which will return a value of 1 if icon creation is
enabled.

Result: 1 if icon creation mode is switched on
        0 if icon creation mode is switched off
```

1.220 INCLUDE FILE

```
Syntax: INCLUDE FILE
        INCLUDE FILE filename

The first form causes the file requester to appear to solicit the file to be
inserted.

The second form causes CED to attempt to insert the named file.

Result: 1 if the file was included
        0 if the file was not included
```

1.221 INHERIT

```
Syntax: INHERIT

This command cuases CED to be set to the state where it will take its
process priority from the task that invoked it.

Result: always 1
```

1.222 INSERT BLOCK

```
Syntax: INSERT BLOCK

This command is similar in function to the newer "PASTE" command, but has
been preserved for compatibility with scripts created with previous versions
of CED. Please consult the "PASTE" command for specific information.

Result: always 1
```

1.223 INSERT MODE

```
Syntax: INSERT MODE

This command toggles the state of the "Insert Mode?" menu item.

Result: 1 if insert mode is switched on
        0 if insert mode is switched off
```

1.224 INSTALL DOS/AREXX COMMAND

```
Syntax: "INSTALL DOS/AREXX COMMAND"
        "INSTALL DOS/AREXX COMMAND" n
        "INSTALL DOS/AREXX COMMAND" n command

The first form causes a requester to appear for your manual dispatching.
Note that the quotation marks around the command are significant due to the
presence of the "/" in the command.

The second form displays a requester allowing you to enter a command for
function key n.

The third form binds the command string specified by command to the function
key n.

Result: 1 if the command was installed
        0 if the command was not installed
```

1.225 JUMP TO AUTO-MARK

```
Syntax: "JUMP TO AUTO-MARK"

This command moves the cursor to the current auto-mark.

Note that the quotation marks are necessary due to the embedded hyphen in
the command string.

The current location of the current view’s auto-mark can be found with the
following commands:

        STATUS AUTOMARKY

                this will return the line number pointed to by
                the "auto-mark".

        STATUS AUTOMARKX

                this will return the column number pointed to
                by the "auto-mark".

Result: always 1
```

1.226 JUMP TO BYTE

```
Syntax: JUMP TO BYTE
        JUMP TO BYTE n

The first form causes the numeric requester to appear to solicit the desired
destination byte number.

The second form automatically moves the cursor to the specified byte n
without opening a requester.

Result: always 1
```

1.227 JUMP TO FILE

```
Syntax: JUMP TO FILE filename

This command moves the cursor to a specific on-screen view. This can be
quite handy in a large number of ways.

If a view is present which corresponds to the file given infilename, that
view will be made the current view.

Result: 1 if the named view was found
        0 if the named view was not found
```

1.228 JUMP TO LINE

```
Syntax: JUMP TO LINE
        JUMP TO LINE n

The first form causes the numeric rquester to appear to solicit the desired
destination line number.

The second form automatically moves the cursor to the specified line n
without opening a requester.

Note that if you give a larger or smaller number, it will display a numeric
requester showing the current minimum and maximum line numbers.

Result: 1 if text cursor moved to the new location
        0 if text cursor did not move
```

1.229 JUMP TO MARK

```
Syntax: JUMP TO MARK n

This command will cause the current text cursor position to be moved to the
location remembered in mark n.

The current values of the marks can be queried by the following ARexx status
commands.

        Mark    Line            Column
        ------- --------------- -------------
        1       status marky1   status markx1
        2       status marky2   status markx2
        3       status marky3   status markx3

If a mark has not yet been set, it will contain an undefined value. Keep
this in mind when querying the values of the marks.

In addition to the three standard bookmarks, CED maintains a transient mark
called the auto-mark. The auto-mark is defined as the last position the
cursor was in before CED had to redraw an entire screen. If you were in the
middle of a large file and jumped far away due to an executed command such
as "SEARCH FOR" or "BEG OF FILE", then auto-mark would be defined as the
character position you were in before making the long distance jump.

Since the auto-mark is defined automatically, no command exists for setting
its value. You can instantly travel to the location of the auto-mark using
its built-in keyboard binding Amiga+4.

Result: 1 if command was successful
        0 if command was not successful
        1 or RESULT if given an invalid mark number
```

1.230 JUMPTO

```
Syntax: JUMPTO LINE column

This command is an extension to the "JUMP TO LINE" command found in the
"Move" menu that allows for the specification of a column number.

Result: always 1
```

1.231 KEYPAD = MOVEMENT

```
Syntax: "KEYPAD = MOVEMENT"

This command toggles the state of the "Keypad = Movement?" menu item.

Note that the quotes around the command are necessary to the the "=" in the
command name. Failure to place this command in quotes will result in an
ARexx variable named "KEYPAD" being assigned a NULL string.

You can query the state of "Keypad = Movement?" from ARexx using the command
"STATUS KEYPADMOVEMENT".

Result: 1 if keypad = movement mode is switched on
        0 if keypad = movement mode is switched off
```

1.232 LASTKEY

```
Syntax: LASTKEY

This command provides a rudimentary method of obtaining asynchronous user
input from an ARexx program. It will return the rawkey code of the last key
pressed by the user as well as its qualifier. Note that the qualifier value
must be adjusted before it can be used (by subtracting the number 32768).

Result: -1 if key was not pressed
        Raw key code and qualifier as two numbers separated by a space
        character
```

1.233 LAYOUT

```
Syntax: LAYOUT

This command toggles the state of the "Layout?" menu item.

Result: 1 if layout mode is switched on
        0 if layout mode is switched off
```

1.234 LEFT

```
Syntax: LEFT

This command is the same as pressing the cursor left key. It will move the
text cursor one position to the left. The text cursor will wrap to the
preceding line if necessary.

Result: always 1
```

1.235 LEFT 12 CHARS

```
Syntax: LEFT 12 CHARS

This command moves the text cursor 12 characters to the left of its current
position or to the beginning of the line, whichever ocurs first.

Result: always 1
```

1.236 LL

```
Syntax: LL line
        LL line column

The first form moves the cursor to the specified line number.

The second form moves the cursor to the specified line and column number.

Line and column numbers start at 1.

This command is directly compatible with the commands used in SAS/C’s scmsg
program. Note that the rather strange syntax which scmsg defaults to,
whereby DM is specified at the end of the LL command line works with LL and
DM only.

Result: always 1
```

1.237 LOAD DEFINITIONS

```
Syntax: LOAD DEFINITIONS
        LOAD DEFINITIONS filename

The first form causes the file requester to appear, soliciting the name of
the file from which to load a new set of macro definitions.

The second form causes CED to attempt to load a new set of macro definitions
from the file named filename. The file requester will not appear.

Result: 1 if macros were loaded
        0 if macros were not loaded
```

1.238 LOAD DOS/AREXX COMMANDS

```
Syntax: "LOAD DOS/AREXX COMMANDS"
        "LOAD DOS/AREXX COMMANDS" filename
```

The first form causes the CED file requester to appear for manual operation. Note that quotation marks must surround the command since it contains an embedded "/".

The second form will attempt to make CED load the named file. No requester will appear.

```
Result: 1 if commands were loaded
        0 if commands were not loaded
```

1.239 LOWER CASE WORD

```
Syntax: LOWER CASE WORD
```

This command switches the case of each letter in the word currently underneath and to the right of the text cursor.

```
Result: always 1
```

1.240 MARK

```
Syntax: MARK
```

This command will cause a horizontal block definition to commence at the current text cursor location.

Note that if a block was already in the process of being defined, the block definition will be cancelled. In this respect, the command can be considered to be a toggle.

```
Result: 1 if block operation was started
        0 if block operation was stopped
```

1.241 MARK BLOCK

This command is similar in function to the newer "MARK" command, but has been preserved for compatibility with scripts created with previous versions of CED. Please consult the "MARK" command for specific information.

1.242 MARK COLUMNAR

```
Syntax: MARK COLUMNAR

This command will cause a vertical block definition to commence at the
current text cursor location.

Note that if a block was already in the process of being defined, the block
definition will be cancelled. In this respect, the command can be considered
to be a toggle.

Result: 1 if columnar block operation was started
        0 if columnar block operation was stopped
```

1.243 MARK COLUMNAR BLOCK

```
This command is similar in function to the newer "MARK COLUMNAR" command,
but has been preserved for compatibility with scripts created with previous
versions of CED. Please consult the "MARK COLUMNAR" command for specific
information.
```

1.244 MARK LOCATION

```
Syntax: MARK LOCATION n

This command will cause the current text cursor position to be remembered as
mark n.

Result: 1 if position was marked
        0 if location number was invalid
```

1.245 MAX SCROLL

```
Syntax: MAX SCROLL
        MAX SCROLL n

The first form will cause the numeric requester appear for you to specify a
max scroll value.

The second form sets the max scroll value to be n. No requester will appear.

You can query the value of the max scroll distance using "STATUS
MAXSCROLLY".

Note that setting max scroll 1 to 1 effectively eliminates scrolling for all
screen updates, except single line scrolls. This tends to produce a great
deal of screen flashing since CED has to constantly redraw the entire
screen.

Setting max scroll to a huge number can also produce some very undesirable
results. For example, if it were set to 10000 and you wnted to move 5000
```

```
lines away from the current position, you would scroll all the way there,
which could take a long time.

Result: 1 if scroll value was in range
        0 if scroll value was not entered or out of range
```

1.246 MAX UNDO LEVELS

```
Syntax: MAX UNDO LEVELS
        MAX UNDO LEVELS n

The first form will cause the numeric requester to appear soliciting the new
value for max undo levels.

The second for will cause max undo levels to be set to n. No requester will
appear.

Result: 1 if new undo level was set
        0 if new undo level was not set
```

1.247 MAX UNDO MEMORY

```
Syntax: MAX UNDO MEMORY
        MAX UNDO MEMORY n

The first form will cause the numeric requester to appear soliciting the new
value for max undo memory.

The second form will cause max undo memory to be set to n. No requester will
appear.

Result: 1 if new undo memory value was set
        0 if new undo memory value was not set
```

1.248 MENU

```
This command exists solely for backwards compatibility with older CED
releases. It purpose was to invoke a menu item by specifying the menu, item
and subitem numbers. Since the contents of the menus and their arrangement
can change from one CED release to another, this command has been frozen to
use the menu ordering as used by CygnusEd Professional release 3.5.
```

1.249 NEXT VIEW

```
Syntax: NEXT VIEW

This command moves the cursor to the view below the current one.
```

```
Result: always 1
```

1.250 NEXT WORD

```
Syntax: NEXT WORD

This command moves the cursor to the beginning of the first word to the
right of the current cursor position.

Result: always 1
```

1.251 NO SCROLL BAR

```
Syntax: NO SCROLL BAR

This command removes the scroll bar from the current view.

The state of this command can be queried indirectly using the "STATUS
LEFTPROPGADG" and "STATUS RIGHTPROPGADG" ARexx commands.

Result: always 1
```

1.252 OKAY1

```
Syntax: OKAY1 text

This command provides access to CED’s notify requester. The notify requester
will present the supplied text along with a single button marked "Continue".

Note that the notify requester will generally appear near the present mouse
pointer position.

Multiple lines of text can be separated by line feed characters (denoted in
ARexx as ’0A’X).

Result: always 1
```

1.253 OKAY2

```
Syntax: OKAY2 text

This command provides access to CED’s response requester. This command will
present the supplied text along with two buttons marked "Ok" and "Cancel".

Note that the response requester will generally appear near the present
mouse pointer position.
```

```
Multiple lines of text can be separated by line feed characters (denoted in
ARexx as '0A'X).

Result: 1 if Ok was selected
        0 if Cancel was selected
```

1.254 ON/OFF

```
Syntax: "ON/OFF"

This command toggles the status line between its standard and alternate
values. The quotation marks are necessary since the command contains an
embedded "/".

Result: 1 if status line is switched on
        0 if status line is switched off
```

1.255 OPEN

```
Syntax: OPEN
        OPEN 0
        OPEN 1
        OPEN 0 filename
        OPEN 1 filename
        OPEN 1 filename 1

The first form will display a file requester to solicit a file to load. If
there are any outstanding changes made to the current view and there are no
other cooperating views open at the time, a response requester will appear
asking you if you are sure about opening up a new file without first saving
the changes in the current view. Then, the file requester will appear
soliciting a file to open.

The second form is similar to the first, but if there are any outstanding
changes made to this view nd there are no other cooperating views open at
the time, this command will abort and return a 0 result. No requesters will
appear. If there were no outstanding changes to the current view or there
wre cooprerating views open, then the file requester will appear soliciting
a file to open.

The third form is similar to the ones above, but the current view will be
cleared even if there are outstanding changes and there are no other
cooperating views. No response requester will be presented. Then, the file
requester will appear to solicit a file to open.

The fourth form is similar to the second form, but if there were no
outstanding changes to the current view or there were cooperating views
open, CED will attempt to open the named file (filename).

Note that this form or if the command has very little utility because if
there were no outstanding changes or there was at least one other
```

```
cooperating view, the file which would be opened would be named "0", not
"filename". This is because the 0 would be the first utilized argument since
there was no reason to bring up the response requester.

The fifth form is similar to the fourth form, except that the current view
will be cleared even if there are outstanding changes and tehre are no other
cooperating views. No response requester will be presented. Then, CED will
attempt to open the named file (filename).

Note that you should use this construction only if you are sure that calling
the open command would result in the response requester asking if you are
sure you want to sacrifice the unsaved changes in the current view.

The sixth form is similar to the fifth form, except that no response
requester will be presented. Then, CED will attempt to open the named file
(filename).

Note that you should use this construction only if you are sure that calling
the open command would result in the response requester asking if you are
sure you want to sacrifice the unsaved changes in the current view and you
are sure that the supplied filename does not exist. Otherwise you are likely
to have a file called "1" opened.

Result: 1 if file was loaded
        0 if file was not loaded
```

1.256 OPEN NEW

```
Syntax: OPEN NEW

This command causes a new empty view to be opened.

Result: 1 if new view was opened
        0 if new view could not be opened
```

1.257 OW

```
Syntax: OW filename

This command puts the cursor into a view displaying the specified file. If
the file does not exist in a view, it will be loaded.

This command is directly compatible with the default commands used in
SAS/C's scmsg program.

Result: 1 if the file was either loaded or its view made active
        0 if the file was not loaded
```

1.258 PAGES

```
Syntax: PAGES

This command will cause the alternate status line to be displayed and set to
the "pages" style.

Result: always 1
```

1.259 PASTE

```
Syntax: PASTE

This command will cause the contents of the clipboard to be inserted into
the current view at thte current text cursor position. The contents of the
paste buffer is not disturbed.

Result: 1 if text was pasted
        0 if clipboard was empty
```

1.260 POST PERIOD SPACES

```
Syntax: POST PERIOD SPACES
        POST PERIOD SPACES s

The first form will display the numeric requester, from which you may enter
a new number of spaces.

The second form will set the number of spaces to the given value (s).

Result: 1 if a valid number is given
        0 if an invalid number is given or if the user cancels the requester.
```

1.261 PREV WORD

```
Syntax: PREV WORD

This command moves the text cursor to the beginning of the first word to the
left of the current cursor position.

Result: always 1
```

1.262 PREVIOUS VIEW

```
Syntax: PREVIOUS VIEW

This command moves the text cursor to the view above the current one.

Result: always 1
```

1.263 PRINT BLOCK

```
Syntax: PRINT BLOCK
        PRINT BLOCK file_or_device_name
        PRINT BLOCK file_or_device_name t2s s2t indent
```

This command is similar to the newer "PRINT CLIP" command, but has been preserved for compatibility with scripts created with previous versions of CED. Please consult the "PRINT CLIP" command for specific information.

1.264 PRINT CLIP

```
Syntax: PRINT CLIP
        PRINT CLIP file_or_device_name
        PRINT CLIP file_or_device_name t2s s2t indent
```

The first form will cause the CED file requester to appear to solicit a destination for the output of the print function. The print requester will then appear to solicit any further modifiers to the function.

The second form will direct the output of the print function to the specified file or device. The CED file requester will not appear. The print requester will be brought up to allow the user to specify any additional modifiers to the print function.

```
        Tab processing          t2s     s2t
        ----------------------- ------- -------
        tabs to spaces          0       0
        leave tabs alone        1       0
        spaces to tabs          0       1
```

The third form will direct the output of th function to the specified file or device. The file requester will not appear. The two flags will be used to set the state of the tab processing. The indent parameter sets the number of spaces to indent the output of the print function.

```
Result: 1 if text was printed
        0 if text was not printed.
```

1.265 PRINT FILE

```
Syntax: PRINT FILE
        PRINT FILE file_or_device_name
        PRINT FILE file_or_device_name t2s s2t indent
```

The first form will cause the CED file requester to appear to solicit a destination for the output of the print function. The print requester will then appear to solicit any further modifiers to the function.

The second form will direct the output of the print function to the specified file or device. The CED file requester will not appear. The print requester will be brought up to allow the user to specify any additional

```
modifiers to the print function.

        Tab processing          t2s     s2t
        ----------------------- ------- -------
        tabs to spaces          0       0
        leave tabs alone        1       0
        spaces to tabs          0       1

The third form will direct the output of the function to the specified file
or device. The file requester will not appear. The two flags will be used to
set the state of the tab processing. The indent parameter sets the number of
spaces to indent the output of the print function.

Result: 1 if text was printed
        0 if text was not printed.
```

1.266 PRIORITY

```
Syntax: PRIORITY

This command makes CygnusEd use the preset Task priority, as controlled
via the "Global/Priority" menu.

Result: always 1
```

1.267 PUBSCREEN

```
Syntax: PUBSCREEN name

This command will move the current CED screen or window as a visitor window
on the specified public screen. Remember that screen names are case
sensitive.

If the screen does not exist, then the screen or window will open on the
Workbench screen.

Result: always 1
```

1.268 QUIT

```
Syntax: QUIT
        QUIT 0
        QUIT 1

The first form will cause the current view to be exited. If there are any
outstanding changes to this view and there are no other views cooperating
with this one, then CED will ask you for confirmation before quitting.

The second form will cause the current view to be exited only if no
requesters would be generated. That is, the current view will be exited only
```

if there are no outstanding changes made for it or there are other cooperating views present.

The third form will cause the current view to be exited. No requester will be presented and any changes made to this view will be lost if there are no other views cooperating with this one.

At this time we do not recommend the user of this command from ARexx or from CED macros.

```
Result: 1 if view was closed (or quit)
        0 if view was not closed (or quit)
```

1.269 QUIT & DIE

Syntax: "QUIT & DIE"

NOTE: This command has no built-in keyboard binding and is not recommended
~~~~~~for use in macros. In ARexx programs, you should use "QUIT" instead.

```
Result: 1 if view was closed (or quit)
        0 if view was not closed (or quit)
```

1.270 RAWKEY

Syntax: RAWKEY code qualifier

This command allows you to simulate the typing of any key on the keyboard. This command takes two parameters, the first is the raw key code of the key to be pressed, the second is its qualifier.

For your convenience, the table or raw key codes, as available with the "usa" keyboard layout, is provided.

Raw key value	Unshifted key	Shifted key
0	`	~
1	1	!
2	2	@@
3	3	#
4	4	$
5	5	%
6	6	^
7	7	&
8	8	*
9	9	(
10	0	)
11	-	_
12	=	+
13	\	|
14	none	none
~~~~~~

15	0 (keypad)	0 (keypad)
16	q	Q
17	w	W
18	e	E
19	r	R
20	t	T
21	y	Y
22	u	U
23	i	I
24	o	O
25	p	P
26	[	{
27	]	}
28	none	none
29	1 (keypad)	1 (keypad)
30	2 (keypad)	2 (keypad)
31	3 (keypad)	3 (keypad)
32	a	A
33	s	S
34	d	D
35	f	F
36	g	G
37	h	H
38	j	J
39	k	K
40	l	L
41	;	:
42	’ "	
43	none	none
44	none	none
45	4 (keypad)	4 (keypad)
46	5 (keypad)	5 (keypad)
47	6 (keypad)	6 (keypad)
48	none	none
49	z	Z
50	x	X
51	c	C
52	v	V
53	b	B
54	n	N
55	m	M
56	,	<
57	.	>
58	/	?
59	none	none
60	. (keypad)	. (keypad)
61	7 (keypad)	7 (keypad)
62	8 (keypad)	8 (keypad)
63	9 (keypad)	9 (keypad)
64	space	space
65	backspace	backspace
66	tab	tab
67	enter	enter
68	return	return
69	esc	esc
70	del	del
71	none	none

72	none	none
73	none	none
74	- (keypad)	- (keypad)
75	none	none
76	cursor up	cursor up
77	cursor down	cursor down
78	cursor right	cursor righ
79	cursor left	cursor left
80	F1	F1
81	F2	F2
82	F3	F3
83	F4	F4
84	F5	F5
85	F6	F6
86	F7	F7
87	F8	F8
88	F9	F9
89	F10	F10
90	((keypad)	((keypad)
91	) (keypad)	) (keypad)
92	/ (keypad)	/ (keypad)
93	* (keypad)	* (keypad)
94	+ (keypad)	+ (keypad)
95	help	help
96	left-shift	left-shift
97	right-shift	right-shift
98	caps lock	caps lock
99	ctrl	ctrl
100	left-alt	left-alt
101	right-alt	right-alt
102	left-amiga	left-amiga
103	right-amiga	right-amiga

Note that raw keycodes 90 through 94 are not found on the A1000 keyboard.

Acceptable values for the qualifier include combinations of the following:

Value	Meaning
1	left-shift
2	right-shift
4	caps lock
8	ctrl
16	left-alt
32	right-alt
64	left-amiga
128	right-amiga

1.271 REDO

Syntax: REDO

This command performs the actions that were previously undone using the "UNDO" command.

```
Result: 1 if last operation was undo
        0 if last operation was not undo
```

1.272 REPEAT KEY/MENU

```
Syntax: "REPEAT KEY/MENU" repeat_count
        "REPEAT KEY/MENU" repeat_count key_code
        "REPEAT KEY/MENU" repeat_count key_code qualifier_code
```

This command is somewhat cumbersome to execute from ARexx and has some limitations. The command must be enclosed in quotation marks due to the presence of the "/" character.

The repeat_count is the count of the number of repetitions to make.

The key_code is the raw keycode for a single key on the Amiga keyboard. A listing of the raw keycodes can be found under the discussion of the "RAWKEY" ARexx command. Since you can only supply one keycode, an obvious limitation of the "REPEAT KEY/MENU" command is that it can execute only single key commands (plus possibly multiple qualifiers).

The qualifier_code is a keyboard qualifier as defined in the key qualifier table under the discussion of the "RAWKEY" ARexx command.

The first form requires the user to select the menu item, subitem or key stroke to be repeated.

The second form assumes a qualifier of 0.

The third form allows you to specify the repeat count, raw keycode , and qualifier all in one statement.

From ARexx, this command only allows repetition of keystrokes, not menu events. However, if a menu item has a keyboard shortcut, then this can be used to repeat it. For instance, the following line tells CED to execute Amiga+1 one time:

```
        "REPEAT KEY/MENU 1 1 64"
```

This is equivalent to the "JUMP TO MARK 1" command.

Note that one drawback of using this command in your ARexx programs is that it makes your programs quite unintelligible.

```
Result: 1 if command was successful
        0 if command was not successful
```

1.273 REPEAT REPLACE

```
Syntax: REPEAT REPLACE
```

This command will perform a replace operation from the current text cursor

```
position.

Note that when called from ARexx, this command will search and replace using
the string which was last entered manually into the Search/Replace
requester. CED will restore the search and replace strings each time "SEARCH
FOR" and "REPLACE" are called from an ARexx program or macro. To be sure of
the strings you will be using (from ARexx), use multiple "REPLACE" commands
rather than the "REPEAT REPLACE" command.

Result: 1 if text was replaced
        0 if text was not replaced
```

1.274 REPEAT SEARCH BACKWARDS

```
Syntax: REPEAT SEARCH BACKWARDS

This command searches for the previously specified search string beginning
at the current text cursor position and search backward through the file.

Note that when called from ARexx, this command will search for the string
which was last entered manually into the search requester. CED will restore
the search and replace strings each time "SEARCH FOR" and "REPLACE" are
called from an ARexx program or macro. To be sure of the string you will
search for (from ARexx), use multiple "SEARCH FOR" commands rather than the
"REPEAT SEARCH BACKWARDS" command.

Result: 1 if text was found
        0 if text was not found
```

1.275 REPEAT SEARCH FORWARDS

```
Syntax: REPEAT SEARCH FORWARDS

This command searches for the previously specified search string beginning
at the current text cursor position and searching forward through the file.

Note that when called from ARexx, this command will search for the string
which was last entered manually into the search requester. CED will restore
the search and replace strings each time "SEARCH FOR" and "REPLACE" are
called from an ARexx program or macro. To be sure of the string you will
search for (from ARexx), use multiple "SEARCH FOR" commands rather than the
"REPEAT SEARCH FORWARDS" command.

Result: 1 if text was found
        0 if text was not found
```

1.276 REPLACE

```
Syntax: REPLACE
        REPLACE string1 string2
        REPLACE string1 string2 u w f o 1
        REPLACE string1 string2 u w f o 1 option

The first form causes the search/replace requester to appear. You must deal
with it manually.

The second form will replace the first occurence of string1 with string2.
The search/replace requester will not appear. The search criteria will be
taken to be whatever they were the last time "REPLACE" was executed. Note
that in order to cause CED to search/replace for more than one word, you
must arrange for a set of quotation marks to surround the search argument.
This can sometimes require creating quoting. The search/replace operation
will be executed exactly once.

The third form is similar to the second, except that the search criteria
will be set by the boolean values supplied in u, w, f and o. These flags
stand for:

        u = "Upper case = Lower case"
        w = "Wildcards"
        f = "Forwards"
        o = "Only words"

Please note that an additional 1 must follow these boolean values. The
search/replace operation will be performed exactly once.

The fourth from is similar to the third, but in addition to the option
argument is required. option may be "g" (for global) or "t" (for turbo).
This is how global and turbo replacements may be performed.

Note that if "REPLACE" is exeecuted from ARexx or from a macro, it will
preserve the previous string to be sought after an replaced, unless the
first form indicated above is used. That is, if the user does not physically
type in a string to search for or replace, the old string will be preserved.
So:

        REPLACE

will not preserve the previous string to search for or replace, but:

        REPLACE "string1" "string2"

will preserve the previous contents of the strings. This same principal
holds true for the "SEARCH FOR" command. Clearly, this change in behavior
affects how and if the "REPEAT REPLACE" commands should be used.

Result: 1 if text was replaced
        0 if text was not replaced
```

1.277 RESET ALIAS

```
Syntax: RESET ALIAS
```

```
This command resets the ASCII zero alias back to ASCII 0.

Result: always 1
```

1.278 RIGHT

```
Syntax: RIGHT

This command will move the text cursor one character position to the right,
wrapping to the next line if necessary.

Result: always 1
```

1.279 RIGHT 12 CHARS

```
Syntax: RIGHT 12 CHARS

This command moves the text cursor to the right 12 character positions or to
the end of the file, whichever occurs first.

Result: always 1
```

1.280 ROT BLOCK

```
Syntax: ROT BLOCK

This command is simliar in function to the newer "ROT MARKED" command, but
has been preserved for compatibility with scripts created with previous
versions of CED. Please consult the "ROT MARKED" command for specific
information.
```

1.281 ROT MARKED

```
Syntax: ROT MARKED

This command performs Ceasarian encryption on the currently defined block of
text.

Result: 1 if block was rotated
        0 if no block was defined
```

1.282 RX

```
 Syntax: RX script_name
Version: Requires CygnusEd V4.9 or higher

This command executes the ARexx script with the name "script_name".

Result: always 1
```

1.283 SAFE SAVES

```
Syntax: SAFE SAVES

This command sets the "Safe saves" mode.

Result: always 1
```

1.284 SAVE

```
Syntax: SAVE
        SAVE filename

The first form causes the current view to be saved. If the current view has
no name then the file requester will appear.

The second form also causes the current view to be saved, the supplied
filename will be ignored unless the current view has no name. In this case,
the supplied filename will be used rather than causing the file requester to
appear.

Result: 1 if view’s contents were saved
        0 if view’s contents were not saved
```

1.285 SAVE ALL CHANGES

```
Syntax: SAVE ALL CHANGES
        SAVE ALL CHANGES file1 file2 ... filen

The first form will cause the file requester to appear if any of the changed
views are not already associated with a file.

The second form, although not recommended since it can easily lead to
confusion, does work. This will cause each changed view to be saved. The
first view which does not already have a filename associated with it will
take on the name specified in the first argument. The second view which does
not have a filename associated with it will take on the name specified in
the second argument, and so on.

Result: 1 if all unsaved views were saved
        0 if all unsaved views were not saved
```

1.286 SAVE AS

```
Syntax: SAVE AS
        SAVE AS filename
        SAVE AS filename 0
        SAVE AS filename 1

The first form causes the current view to be saved in the file solicited
from the user by the file requester.

The second form causes the current view to be saved in the named file. If
the named file already exists, you will be asked to confirm your intentions.

The third form causes the current view to be saved in the named file. If the
named file already exists, the save will be aborted. No requesters will be
generated.

The fourth form causes the current view to be saved in the named file. If
the named file already exists, it will be overwritten. No requesters will be
generated.

Result: 1 if view’s contents were saved
        0 if view’s contents were not saved
```

1.287 SAVE BLOCK TO FILE

```
Syntax: SAVE BLOCK TO FILE
        SAVE BLOCK TO FILE filename
        SAVE BLOCK TO FILE filename 0
        SAVE BLOCK TO FILE filename 1

This command is similar in function the newer "SAVE CLIP AS" command, but
has been preserved for compatibility with scripts created with previous
versions of CED. Please consult the "SAVE CLIP AS" command for specific
information.
```

1.288 SAVE CLIP AS

```
Syntax: SAVE CLIP AS
        SAVE CLIP AS filename
        SAVE CLIP AS filename 0
        SAVE CLIP AS filename 1

The first form causes the CED file requester to appear to solicit the name
of the file into which the contents of the current clipboard unit will be
saved.

The second form saves the current clip to the named file (filename). The
file requester will not appear. If the named file already exists, a
requester will be generated.

The third form saves the current clip to the named file (filename) if it
```

```
does not already exist. No requesters will be generated.

The fourth form saves the current clip to the named file (filename). If the
named file already exists, it will be overwritten. No requesters will be
generated.

Result: 1 if the clipboard contents were saved
        0 if the clipboard contents were not saved
```

1.289 SAVE DEFINITIONS

```
Syntax: SAVE DEFINITIONS
        SAVE DEFINITIONS filename

The first form will cause the file requester to appear, soliciting a name
under which the current macro definitions will be stored.

The second form will attempt to store the current macro definitions in the
file named filename. The file requester will not appear.

Result: 1 if macros were saved
        0 if macros were not saved
```

1.290 SAVE DOS/AREXX COMMANDS

```
Syntax: "SAVE DOS/AREXX COMMANDS"
        "SAVE DOS/AREXX COMMANDS" filename

The first form will cause the file requester to appear soliciting the name
of a file under which to save the current DOS and ARexx command bindings
(those bound to the 10 function keys).

The second form will attempt to make CED save the current DOS and ARexx
commands (those bound to the 10 function keys) to the given file. The file
requester will not appear.

Result: 1 if the commands were saved
        0 if the commands were not saved
```

1.291 SAVE ENVIRONMENT

```
Syntax: SAVE ENVIRONMENT
        SAVE ENVIRONMENT filename

The first form displays a file requester from which you can specify a
filename under which which to save the current environment.

The second form stores the current environment under the specified filename.

Result: 1 if the environment was saved
```

```
        0 if the environment was not saved
```

1.292 SCREEN HEIGHT

```
Syntax: None

Due to changes in the way screens are invoked, this command is not supported
anymore through ARexx. Please use either the "SET SCREEN SIZE AND TYPE" or
the "SETSCREEN" command instead.
```

1.293 SCREEN WIDTH

```
Syntax: None

Due to changes in the way screens are invoked, this command is not supported
anymore through ARexx. Please use either the "SET SCREEN SIZE AND TYPE" or
the "SETSCREEN" command instead.
```

1.294 SCROLL BAR ON LEFT

```
Syntax: SCROLL BAR ON LEFT

This command places the scroll bar on the left side of the current view. The
state of this command can be queried using the "STATUS LEFTPROPGADG" ARexx
command.

Result: always 1
```

1.295 SCROLL BAR ON RIGHT

```
Syntax: SCROLL BAR ON RIGHT

This command places the scroll bar on the right side of the current view.
The state of this command can be queried using the "STATUS RIGHTPROPGADG"
ARexx command.

Result: always 1
```

1.296 SEARCH FOR

```
Syntax: SEARCH FOR
        SEARCH FOR string
        SEARCH FOR string u w f o 1

The first form causes the search requester to appear. You must deal with it
```

manually.

The second form will search for the next occurence of the specified string. The search requester will not appear. The search criteria will be taken to be whatever they were the last time "SEARCH FOR" was executed. Note that in order to cause CED to search for more than one word, you must arrange for a set of quotation marks to surround the search argument. This can sometimes require creating quoting.

The third form is similar to the second, except that the search criteria will be set by the boolean values supplied in u, w, f and o. These flags stand for:

```
        u = "Upper case = Lower case"
        w = "Wildcards"
        f = "Forwards"
        o = "Only words"
```

Please note that an additional 1 must follow these boolean values.

Note that executing this command from ARexx or from a macro will preserve the previous string to be sought after, unless the first form indicated above is used. That is, if the user does not physically type in a new string to search for, the old string will be preserved. So:

```
        SEARCH FOR
```

will not preserve the previous string to search for, but:

```
        SEARCH FOR "string"
```

will preserve the previous contents of the search string. This same principal holds true for the "REPLACE" command. Clearly, this change in behavious affects how and if the "REPEAT SEARCH" commands should be used.

NOTE: Release 3 of CED changes where the search command begins searching. This change affects ARexx scripts only. In previous releases, the search command began at the character following the character the cursor was positioned on. This allows multiple invocations of the search command to find successive occurences of the string. However, this prevented a match from being returned if the matched string were the first characters in the file. As of release 3, the search command begins searching at the current character. This means successive calls to the search command (by opening the search requester and not just performing a repeat search command) will mach exactly the same string, unless interspersed with a call to "RIGHT" to move the cursor past the beginning of the matched string. This allows strings which begin a file to be found with the "SEARCH FOR" command.

```
Result: 1 if text was found
        0 if text was not found
```

1.297 SELECT DISK FONT

```
Syntax: "SELECT DISK FONT"
        "SELECT DISK FONT" fontname.font size
```

The first form will cause the font requester to appear to solicit a new font.

Note that the quotation marks around the command are necessary sicne there is an ARexx keyword "SELECT". Executing this command will cause the font requester to appear to solicit an alternative font.

The second form will select the given font in the given size. No requester will be generated. Note that the font name must have the ".font" string appended to it.

You can query the name of the current font and its size using the ARexx command "STATUS FONTINFO".

```
Result: 1 if font was changed
        0 if font was not changed
```

1.298 SEND DOS/AREXX COMMAND

Syntax: None

This command cannot be executed from ARexx. Use ARexx's "ADDRESS COMMAND" facility to execute DOS or ARexx commands.

Result: always 1

1.299 SEND DOS/AREXX OUTPUT TO

```
Syntax: "SEND DOS/AREXX OUTPUT TO"
        "SEND DOS/AREXX OUTPUT TO" output
```

The first form displays the requester with the input field asking for the destination.

The second form sets the output to the 'output' argument.

Result: always 1

1.300 SET CLIPBOARD UNIT

```
Syntax: SET CLIPBOARD UNIT
        SET CLIPBOARD UNIT unit_number
```

The first form displays a numeric requester from which the clipboard unit can be specified.

The second form sets the clipboard unit to the specified value

```
(unit_number).

If this command succeeds, it will set the RESULT variable to one more than
the unit number specified.

Result: unit_number+1 if unit was changed
        0 if unit was not changed
```

1.301 SET ICON TOOL NAME

```
Syntax: SET ICON TOOL NAME
        SET ICON TOOL NAME filename

The first form will cause the text requester to appear for user input.

The second form will set the icon tool name to filename without bringing up
the text requester.

Result: 1 if tool name was changed
        0 if tool name was not changed
```

1.302 SET PRIORITY

```
Syntax: SET PRIORITY
        SET PRIORITY value

The first form causes the numeric requester to appear to solicit a new
priority value.

The second form attempts to set the priority to the given value. Note that
the value must be enclosed in quotation marks if it is negative.

Result: 1 if priority was changed
        0 if priority was not changed
```

1.303 SET RIGHT BORDER

```
Syntax: SET RIGHT BORDER
        SET RIGHT BORDER value

The first form will cause the right border to be set manually using the
mouse.

The second form will set the right border to the value specified by value.

You can query the value of the right border from ARexx using the "STATUS
RIGHTBORDER" command.

Result: 1 if right border has changed
        0 if right border has not changed
```

1.304 SET SCREEN SIZE AND TYPE

```
Syntax: SET SCREEN SIZE AND TYPE

This command displays the screen mode requester.

To actually specify a screen to open, use the "SETSCREEN" command.

Result: 1 if screen type and size were changed
        0 if screen type and size were not changed
```

1.305 SET SCROLL BORDERS

```
Syntax: SET SCROLL BORDERS

This command allows the user to change the scroll borders for the current
view.

Result: 1 if scroll borders were modified
        0 if scroll borders were not modified
```

1.306 SET SCROLL JUMP

```
Syntax: SET SCROLL JUMP n

This command sets the number of pixels each line will move as a result of a
scroll action.

The n value above may range from 0 to 3, corresponding to 1 to 8 pixels per
scroll.

The ARexx command which returns the current view's scroll jump setting,
"STATUS SCROLLJUMP", returns a value starting from 1, not 0.

Result:  1 if new scroll jump value was accepted
        -1 if new scroll jump value was invalid (out of range).
```

1.307 SET TIMER

```
Syntax: SET TIMER
        SET TIMER n

The first form causes the numeric requester to appear for user input on the
timer value.

The second form sets the timer value to n without causing the numeric
requester to appear.

Result: 1 if timer was changed
        0 if timer was not changed
```

1.308 SETSCREEN

```
Syntax: SETSCREEN mode_id width height

This command specifies the type and dimensions of the custom screen upon
which to open the CED window. For legal values of mode_id, see the Amiga
operating system documentation.

Result: 1 if screen type and size were changed
        0 if screen type and size were not changed
```

1.309 SHOW ASCII VALUES

```
Syntax: SHOW ASCII VALUES

This command causes CED to display (in the title bar of the view) the
decimal value of the character currently underneath the text cursor.

You can query whether the title bar is displaying ASCII codes by using the
"STATUS SHOWASCII" command.

Result: 1 if show ascii values mode is switched on
        0 if show ascii values mode is switched off
```

1.310 SHRINK VIEW

```
Syntax: SHRINK VIEW

This command attempts to size the current view smaller by one line.

Result: 1 if current view changed in size
        0 if current view did not change in size
```

1.311 SIMPLE SAVES

```
Syntax: SIMPLE SAVES

This command sets the simple saves mode.

Result: always 1
```

1.312 SPACES VISIBLE

```
Syntax: SPACES VISIBLE

This command toggles the state of the "Spaces visible?" menu subitem.
```

```
Result: 1 if spaces visible mode is switched on
        0 if spaces visible mode is switched off
```

1.313 SPAWN NEW CED

```
Syntax: SPAWN NEW CED

This command creates another running copy of CED.

Result: always 1
```

1.314 SPECIFY

```
Syntax: SPECIFY
        SPECIFY filename

The first form will cause the CED file requester to appear to solicit an
environment file to be loaded.

The second form will cause CED to attempt to load the specified file as the
new environment.

Result: 1 if environment was loaded
        0 if environment was not loaded
```

1.315 SPLIT VIEW

```
Syntax: SPLIT VIEW

This command attempts to break the current view into two cooperating views.

"STATUS NUMVIEWS" is used to count the number of views remaining on the
screen.

Result: 1 if current view was split
        0 if current view was not split
```

1.316 STATUS

```
                Syntax: STATUS status_description
        STATUS status_number

The "STATUS" command is the workhorse of CED ARexx programs. This command
allows you to query the values of quite a few internal variables and
settings.

Any command (including the "STATUS" command) which passes results back to
```

the calling ARexx program requires that the following line be executed early on:

```
        OPTIONS RESULTS
```

The enables the ARexx program to receive the result codes which are returned by all CED commands.

Listed in the following table are the "STATUS" commands which are currently supported. The numbers corresponding to the descriptions follow each entry.

ACTUALSIZE	16
AUTOEXPAND	72
AUTOMARKX	30
AUTOMARKY	26
AUTOSAVETIMER	77
BACKSPACECHARBUFFER	64
BACKSPACEWORDBUFFER	61
BLOCKBUFFER	60
BLOCKSCREENX	70
BLOCKY	69
CLIPUNIT	94
CURRENTDIR	75
CURSORCOLUMN	46
CURSORLINE	47
CURSORMEMORYX	87
CURSOROFFSETX	44
CURSOROFFSETY	45
CUSTOMHEIGHT	5
CUSTOMROUTINES	86
CUSTOMWIDTH	6
DELETECHARBUFFER	65
DELETELINEBUFFER	63
DELETEWORDBUFFER	62

DESIREDCOLUMN	52
DIRNAME	20
DISPLAYCOLUMNS	50
DISPLAYLINES	51
EDITABLE	82
FILEMEM	15
FILENAME	19
FONTINFO	81
FORCECUSTOM	3
ICONCREATION	4
ICONTOOLNAME	73
INSERTMODE	12
INTERLACE	1
KEEPRESIDENT	2
KEYPADMOVEMENT	80
LAYOUTMODE	11
LEFTLINE	42
LEFTPROPGADG	40
LINEBUFFER	63
LINEBUFFEROFFSET	56
LINEMEMORYLENGTH	58
LINENUMBER	57
LINESCREENLENGTH	59
MARKX1	27
MARKY1	23
MARKX2	28
MARKY2	24
MARKX3	29

MARKY3	25
MAXSCROLLY	78
MODEID	92
NUMCHANGES	18
NUMLINES	17
NUMVIEWS	22
PIXELLEFTEDGE	48
PIXELTOPEDGE	49
PORTNUMBER	91
PUBSCREENNAME	93
PRIORITY	76
RESTNAME	21
RIGHTBORDER	14
RIGHTPROPGADG	68
SAFESAVES	79
SCRIPTBIT	83
SCROLLBORDERBOTTOM	39
SCROLLBORDERLEFT	36
SCROLLBORDERRIGHT	37
SCROLLBORDERTOP	38
SCROLLJUMP	10
SEARCHINFO	74
SERIALNUMBER	88
SHOWASCII	85
STATUSLINE	35
STATUSLINETYPE	41
TABS	7
TABSARESPACES	13
TABSIZE	8

TASKADDRESS	90
TOPLINE	43
TOTALNUMFILES	67
TOTALNUMVIEWS	66
VERTICALBLOCK	71
VISIBLEEOL	33
VISIBLEESC	34
VISIBLETAB	31
WINDOWHEIGHT	54
WINDOWNUMBER	84
WINDOWWIDTH	53
WORDWRAP	9
ZEROMAPCHAR	89

```
          Result: specific status information if successful
        undefined if status number is invalid or description is unknown
```

1.317 STRIP CR BLOCK

```
Syntax: STRIP CR BLOCK

This command is similar in function to the newer "STRIP CR MARKED" command,
but has been preserved for compatibility with scripts created with previous
versions of CED. Please consult the "STRIP CR MARKED" command for specific
information.
```

1.318 STRIP CR MARKED

```
Syntax: STRIP CR MARKED

This command will set the ARexx RESULT variable to a value of 1 if the
operation actually took place.

Result: 1 if block was stripped
        0 if no block was being defined
```

1.319 TAB SIZE

```
Syntax: TAB SIZE tab_spacing - 1

This command sets the size of tabulator gaps.

Note than when setting regular tab stops from ARexx using this method, that
the value specified in the ARexx command is one less than the desired tab
setting. Therefore, the allowable parameters are 0 to 9.

Also note that the ARexx command which returns the size of the regularly
spaced tabs, "STATUS TABSIZE" returns the actual tab setting.

Result: 1 if new tab setting was accepted
        0 if new tab setting was out of range
```

1.320 TABS = SPACES

```
Syntax: "TABS = SPACES"

This command toggles the state of the "Tabs = spaces?" menu item.

Note that the quotation marks are necessary to execute this command from
ARexx due to the presence of the "=" sign in the command’s text.

From ARexx, the "STATUS TABSARESPACES" command will return a boolean
indicating whether or not "Tabs = spaces?" mode is enabled.

ARexx users should note that when in "Tabs = spaces?" mode, calls to the
"TEXT" command run much more slowly then when not in "Tabs = spaces?" mode.
So, if you are doing lots of text insertions using the "TEXT" command, you
might want to toggle out of "Tabs = spaces?" mode temporarily to
dramatically increase execution speed.

Result: 1 if "Tabs = spaces" mode is switched on
        0 if "Tabs = spaces" mode is switched off
```

1.321 TABS VISIBLE

```
Syntax: TABS VISIBLE

This command toggles the state of the "Tabs visible?" menu subitem.

Result: 1 if "Tabs visible" mode is switched on
        0 if "Tabs visible" mode is switched off
```

1.322 TEXT

```
Syntax: TEXT string

This command will insert its arguments into the current view as the current
```

```
cursor position as if they were letters typed at the keyboard.

This command runs much more slowly if the current view is in "Tabs =
spaces?" mode. If you will be doing a lot of "TEXT" calls, you might want to
check if the curent view is in "Tabs = spaces?" mode and temporarily toggle
out of it, if necessary.

Note that for text strings with embedded spaces, you do not need two levels
of quotes as you do in other commands, such as those than can take
filenames.

Result: 1 if text was entered
        0 if text was not entered (i.e. file is uneditable)
```

1.323 TOPAZ 60 COLUMN

```
Syntax: TOPAZ 60 COLUMN

This command causes CED to select the Topaz 9 ROM-based font.

Result: always 1
```

1.324 TOPAZ 80 COLUMN

```
Syntax: TOPAZ 80 COLUMN

This command causes CED to select the Topaz 8 ROM-based font.

Note that when using Topaz 8 CED is significantly faster in all screen
rendering than when using a different font.

Result: always 1
```

1.325 UNBCK SPC WORD

```
Syntax: UNBCK SPC WORD

This will execute the equivalent of [Ctrl]+[Alt]+Backspace and will insert the
contents of the left word buffer.

Result: 1 if word was "un-backspaced"
        0 if word buffer was empty
```

1.326 UNDELETE LINE

```
Syntax: UNDELETE LINE

This command will insert the contents of the line buffer.

The contents of the global line buffer can be read from ARexx using the
command "STATUS DELETELINEBUFFER".

Result: 1 if line was undeleted
        0 if line buffer was empty
```

1.327 UNDELETE WORD

```
Syntax: UNDELETE WORD

This command will execute the equivalent to [Ctrl]+[Alt]+Del and will insert the
contents of the right word buffer.

Result: 1 if word was undeleted
        0 if the word buffer was empty
```

1.328 UNDO

```
Syntax: UNDO

The command will cause the ARexx RESULT variable to contain a boolean value
indicating whether or not an "UNDO" operation actually occured.

Result: 1 if last operation was undone
        0 if no previous operation existed
```

1.329 VERSION

```
Syntax: VERSION

The command will return the version number of CygnusEd. You can and
should use this command to find out which ARexx commands the editor
supports. The command set may be enhanced with subsequent program
revisions.

Result: Program version, e.g. "4.1"
```

1.330 UP

```
Syntax: UP

This command is the same as pressing the cursor up key. The text cursor will
move up in the same or similar column.
```

Result: always 1

1.331 lockgui

                Syntax: LOCKGUI

This command inhibits the use of the graphical user interface. It does not nest, i.e. a single call to UNLOCKGUI will unlock the user interface regardless of how many times you called LOCKGUI.

Result: always 1

1.332 unlockgui

                Syntax: UNLOCKGUI

This command enables the use of the graphical user interface after it has been disabled via the LOCKGUI command.

Result: always 1

1.333 UP 12 LINES

Syntax: UP 12 LINES

This command will move the text cursor up 12 lines, or as far as possible if less than 12 lines exist between the current line and beginning of the file.

Result: always 1

1.334 UPPER CASE WORD

Syntax: "UPPER CASE WORD"

This command switches the case of each letter in the word currently underneath and to the right of the text cursor.

Result: always 1

1.335 USE WBENCH COLOURS

```
Syntax: USE WBENCH COLOURS

This command causes CED to read the color settings from the currently
defined AmigaOS system preferences structure.

Result: always 1
```

1.336 WITH FILL

```
Syntax: WITH FILL

This command will reformat the current paragraph using full (both left and
right) justification.

Result: always 1
```

1.337 WITHOUT FILL

```
Syntax: WITHOUT FILL

This command will reformat the current paragraph using left justification.

Result: always 1
```

1.338 WORD WRAP

```
Syntax: WORD WRAP

This command toggles the state of the "Word wrap?" menu item.

From ARexx, the "STATUS WORDWRAP" command will return a boolean indicating
whether or not "Word wrap?" mode is enabled.

Result: 1 if word wrap mode is switched on
        0 if word wrap mode is switched off
```

1.339 actualsize

```
Get the number of characters in the file displayed in the current view.
```

1.340 AUTOEXPAND

```
Determine whether auto-expand views mode is on (RESULT=1) or off (RESULT=0).
```

1.341 automarkx

Get the column number corresponding to the auto-mark. See also "STATUS AUTOMARKY".

1.342 automarky

Get the line number corresponding to the auto-mark. See also "STATUS AUTOMARKX".

1.343 autosavetimer

Get the number of minutes contained in the autosave timer. It does not mean autosave is enabled.

1.344 backspacecharbuffer

Get the contents of the left undelete character buffer. In order word, the last character deleted with the Backspace key.

1.345 backspacewordbuffer

Get the contents of the left undelete word buffer. In order words, the last word deleted with [Ctrl]+Backspace.

1.346 blockbuffer

Get the contents of the current clipboard unit. Since the clipboard may be huge, you should be prepared to accomodate a very large block. Note that due to how ARexx works, the buffer cannot be larger than 65535 characters.

1.347 blockscreenx

If a block is currently being defined, the current text cursor position defines one end of the block. This command gets the column number containing the other end of the block. Unlike "STATUS BLOCKY", this command will return a non-negative answer even if no block is currently being defined.

1.348 blocky

If a block is currently being defined, the current text cursor position defines one end of the block. This command gets the line number containing the other end of the block. If no block is being defined when this command is executed, it will return a value of -1. You can use this feature to determine if a block is currently being defined.

1.349 clipunit

Get the current clipboard unit being used.

1.350 currentdir

Get a string which corresponds to CED's notion of its current directory. The current directory can be set by the "Change current directory..." command (in the "Project" menu).

1.351 cursorcolumn

Get a value one less than the text column that the text cursor is presently in.

1.352 cursorline

Get a value one less than the line number on which the text cursor is presently on.

1.353 cursormemoryx

Get the number of characters to the left of an on the same line as the cursor.

1.354 cursoroffsetx

Get the horizontal position of the text cursor relative to the first displayed character. This provides a screen absolute position of the text cursor irrespective of any possible horizontal scrolling.

1.355 cursoroffsety

Get the vertical position of the text cursor relative to the first displayed character. This provides a screen absolute position of the text cursor.

1.356 customheight

Get the height of the CED window (if running on the Workbench screen) or the screen ifself (if running on a custom screen).

1.357 customroutines

Get the type of screen rendering mode currently being used. This is a number between 0 and 2, corresponding to the "Rendering choices" submenu items.

1.358 customwidth

Get the width of the CED window (if running on the Workbench screen) or the screen ifself (if running on a custom screen).

1.359 deletecharbuffer

Get the contents of the right undelete character buffer. In order words, the last character deleted with the Del key.

1.360 deletelinebuffer

Get the contents of the delete line buffer.

1.361 deletewordbuffer

Get the contents of the right undelete word buffer. In other words, the last word deleted with [Ctrl]+Del.

1.362 desiredcolumn

Get the column that the text cursor would like to be in, but may not actually be in. For example, if you execute a "DOWN" command from the end of a very long line to avery short one, the text cursor will not go past the end of the very short ine (assuming "Layout mode?" is not on). If you were to then eecute an UP command, the text cursor would return to its previous position at the end of the long line. This is because the text cursor "wanted" to be in a different colun than it actually was in when it was on the very short line.

1.363 dirname

Get the path of the displayed in the current view but not the file's name.

1.364 displaycolumns

Get the number of text columns displayable in the current CED screen or window. This command takes into account the presence or absence of a scroll bar. If CED is running as a window on the Workbench, this command will take into account the window borders. If the width of CED's screen or window changes or a scroll bar is added or removed, the value returned by this command will change as well.

1.365 displaylines

Get the number of lines which are displayable in the current view. If the height of the current view changes,the value returned by this command will also change.

1.366 editable

Determine whether the current view is (RESULT=1) or is not (RESULT=0) an editable view. That is, is the current view a read-only view?

1.367 filemem

Get the actual address of the start of the data in the file displayed in the current view. CED promises to null-terminate the data in memory but since binary files may be edited and null butes may be entered by the user there may be more than one null byte in the file. You are severely cautioned against abusing this information. Treat the data at the address returned by this command as read-only. Do not modify the data directly in memory under any circumstances.

1.368 filename

Get the complete path name of the file displayed in the current view. Beware of views which have not been associated with a filename yet since they will return an incomplete path. By checking the view's filename with "STATUS RESTNAME", you can determine if this view has ever been saved or not (or if it hasn't been saved, then "STATUS FILENAME" will return an incomplete path).

1.369 fontinfo

Get a string containing the current font, font size, and qualifier if a disk based font is being used. If Topaz 80 column or Topaz 60 column is being used, the string will simply be either "Topaz 80 column" or "Topaz 60 column".

1.370 forcecustom

Determine whether CED is running on a custom screen (RESULT=1) or as a window on the Workbench screen (RESULT=0).

1.371 ICONCREATION

Determine whether icons will (RESULT=1) or will not (RESULT=0) be saved with files.

1.372 icontoolname

Get a string containing the present icon tool name. The icon tool name can be set using "Set icon tool name..." (in the "Environment", "Global settings" menu).

1.373 insertmode

Determine whether "Insert mode" is on (RESULT=1) or off (RESULT=0).

1.374 interlace

Determine whether the CED screen (Workbench or a custom screen) is interlaced (RESULT=1) or non-interlaced (RESULT=0).

1.375 keepresident

Determine whether "Hot-Start mode" is enabled (RESULT=1) or disabled (RESULT=0).

1.376 KEYPADMOVEMENT

Determine whether "Keypad = Movement" mode is on (RESULT=1) or off (RESULT=0).

1.377 layoutmode

Determine whether "Layout" mode is on (RESULT=1) or off (RESULT=0).

1.378 leftline

Get the number of columns on the left side which are not currently displayed in the current view. This is usually zero. However, this may become non-zero if the current view is horizontally scrolled.

1.379 leftpropgadg

Determine whether the scroll base is (RESULT=1) or is not (RESULT=0) on the left edge of the current view. Use this in conjunction with "STATUS RIGHTPROPG" to determine if there is a scroll bar present at all.

1.380 linebuffer

Get a string containing the entire contents of the line the text cursor is currently on.

1.381 linebufferoffset

Get the number of bytes separating the start of the current line from the beginning of the file.

1.382 linememorylength

Get the number of bytes in the line the text cursor is currently on.

1.383 linenumber

Get a value one less than the current line number.

1.384 linescreenlength

Get the length of the current line in on-screen columns. This may be different than the value returned by "STATUS LINEMEMORYLENGTH" due to the presence of tabs or escape codes.

1.385 markx1

Get the column number corresponding to bookmark 1. See also "STATUS MARKY1".

1.386 marky1

Get the line number corresponding to bookmark 1. See also "STATUS MARKX1".

1.387 markx2

Get the column number corresponding to bookmark 2. See also "STATUS MARKY2".

1.388 marky2

Get the line number corresponding to bookmark 2. See also "STATUS MARKX2".

1.389 markx3

Get the column number corresponding to bookmark 3. See also STATUS MARKY3.

1.390 marky3

Get the line number corresponding to bookmark 3. See also "STATUS MARKX3".

1.391 maxscrolly

Get the number of lines past which CED will force a screen redraw, rather than a scroll (i.e., the "Max scroll..." setting).

1.392 modeid

Get the current screen mode identifier, as described in the Amiga operating system documentation.

1.393 numchanges

Get the number of changes made to the current view.

1.394 numlines

Get the number of lines in the file displayed in the current view.

1.395 numviews

Get the number of cooperating views which are displaying data from the same file as the current view. This is handy for determining if the current view represents the last cooperating view editing a given file.

1.396 pixelleftedge

Get the number of pixels from the left edge of CED’s screen or window that text information will begin to be displayed. For example, if CED is on its own custom scren and the scroll bar is not on the left, this command will return a value of 0. If CED is running as a window on the Workbench screen and there is no scroll bar on the left, then this command will return a value of 3.

1.397 pixeltopedge

Get the number of pixels from the top edge of CED’s screen or window that text information for the current view will begin to be displayed. As the height or position of the current view changes relative to other views, the result returned by this command will change.

1.398 portnumber

Get the ARexx port number (as specified by the numeric suffix of the ARexx port name, shown in the "About..." panel) of the current running copy of CED. The first copy of CED that is run will return a value of 0, since its port name ("CYGNUSED") contains no numeric suffix.

1.399 pubscreenname

Get the name of the public scren CED is currently being run on. This command will return a NULL string if CED is running on a custom, non-public screen.

1.400 PRIORITY

Get the value which can be found in the "Priority" submenu. It may or may not be the actual priority to which CED is currently set.

1.401 restname

Get the name of the file displayed in the current view. If the current view corresponds to a file which is new and has never been saved, the string returned by this command will be empty.

1.402 rightborder

Get the column number of the current view's right border.

1.403 rightpropgadg

Determine whether the current view does (RESULT=1) or does not (RESULT=0) have a scroll bar on the right. Use this in conjunction with "STATUS LEFTPROPGADG" to determine if there is a scroll bar present at all.

1.404 safesaves

Get the current file save method. This returns a number between 0 and 2, corresponding to the three file saving methods available.

1.405 scriptbit

Determine whether or not the current view does (RESULT=1) or does not (RESULT=0) have its "script" bit turned on. This is set externally, but preserved by CED.

1.406 scrollborderbottom

Get the number of lines form the bottom edge of the screen where the bottom edge of the current view's scroll border starts.

1.407 scrollborderleft

```
Get the number of columns form the left edge of the scren where the left
edge of the current view's scroll border starts.
```

1.408 scrollborderright

```
Get the number of columns from the right edge of the screen where the right
edge of the current view's scroll border starts.
```

1.409 scrollbordertop

```
Get the number of lines from the top edge of the screen where the top edge
of the current view's scroll border starts.
```

1.410 scrolljump

```
Get the curent scroll jump setting. This will be a number between 1 and 4,
corresponding to the four preset scroll jump settings available.
```

1.411 searchinfo

```
Get two strings delimited by quote marks followed by four boolean values.
The first string corresponds to the last string sought for. The second
string represents the last replacement string used in a "Replace..."
command. The four booleans indicate the values of the four qualifiers to the
last "Search for..." or "Replace..." command.
```

1.412 serialnumber

```
Get the serial number of the currently running CED.
```

1.413 showascii

```
Determine whether "Show ASCII values" mode is on (RESULT=1) or off
(RESULT=0).
```

1.414 STATUSLINE

```
Determine whether the status line is in on (RESULT=1) or off (RSULT=0)
state.
```

1.415 statuslinetype

Get the tye of status line would be displayed if the status line were turned on in the current view. If "Changes" is enabled, set RESULT to 0; otherwise set it to 1. Next, if "Show ASCI values" is enabled, add 16 to RESULT; otherwise do not add anything.

1.416 tabs

Get a string describing the current tab stops, where "-" signifies no tab stop and "T" means tab stop.

1.417 tabsarespaces

Determine whether tabs will (RESULT=1) or will not (RESULT=0) be converted to spaces.

1.418 TABSIZE

Get the actual size of the regularly spaced tabs if they were to be enabled.

1.419 taskaddress

Get the current tasks's memory address.

1.420 topline

Get the number of lines preceding the topmost displayed line in the current view. This corresponds to one less than the line number of the topmost displayed line.

1.421 totalnumfiles

Get the nmber of distinct (non-cooperating) files open in CED right now.

1.422 totalnumviews

Get the number of views currently present in the CED window or screen.

1.423 verticalblock

Determine whether the block currently being defined is (RESULT=1) or is not (RESULT=0) a verticalblock. You may use "STATUS BLOCKY" to determine if a block is being defined, then use this command to determine which type of block it is.

1.424 visibleeol

Determine whether end-of-line (EOL) characters will (RESULT=1) or will not (RESULT=0) be visible in the current view.

1.425 visibleesc

Determine whether escape codes will (RESULT=1) or will not (RESULT=0) be visible in the current view.

1.426 visibletab

Determine whether tabs will (RESULT=1) or will not (RESULT=0) be visible in the curent view.

1.427 windowheight

Get the height in pixels of the entire CED window or screen.

1.428 windownumber

Get an integer corresponding to which view is the current view. Views are numbered from 0, with 0 corresponding to the topmost view.

1.429 windowwidth

Get the width in pixels of the entire CED window or screen.

1.430 WORDWRAP

Determine whether "Word Wrap" mode is on (RESULT=1) or off (RESULT=0).

1.431 zeromapchar

```
Get the current ASCII zero alias for searches, as defined by the "Set ASCII
zero alias for search..." requester.
```