BareED - A Brief Introduction

COLLABORATORS			
	TITLE : BareED - A Brief Introduction		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 31, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

BareED - A Brief Introduction

1.1 BareED 0.10 - a dirty introduction

```
Please, do not expect too much from this file nor from BareED itself. BareED ↩
   is currently only available as a pre-version
(beta) that suits my need and perhaps also to yours.
Although BareED is very stable on my system, it is possible that BareED will fail ↩
     on yours!

 - What is BareED.
  BareED is a simple text editor that will use the ASCII one character set for ↩
     Amiga computers.

 - Why another text editor when already hundreds are available?
  I originally designed BareED as replacement for NotePad, the first word- ↩
     processor for the Amiga, Amiga-ED,
  the batch and script editor, and Amiga-MEmacs, the text editor.

  What I have liked most at NotePad was that the colours could be chosen freely ↩
     and that NotePad also gave the
  possibility to use proportional fonts. Although NotePad is much more ↩
     powerful than BareED, it has so many
  limits that only a few people used NotePad.

  The use of Amiga-MEmacs on an intuitive driven platform like the Amiga is a ↩
     pain in the butt. Even today newer
  versions of the Amiga-ED aren't comfortable to use nor can they be used to edit ↩
     normal text files.

  My favourite text editor is the one shipped with the Devpac$^1$ package - clean ↩
     and easy to use!

  BareED is one of the few text editors on the Amiga, who does not use the ↩
     Amiga console device and therefore
  non-rigid colours and non-fixed width fonts can be used.

  BareED also is one of the few text editors that deal correctly with a sizeable ↩
     editor window; even commercial text
  editors have problems therewith. So you can use BareED like a notepad on your ↩
     Workbench - that means that
```

```
  you  reduce  BareED's  editor  window  to a minimum and leave it open anywhere  ↩
     on your Workbench desk while
  you  do other things. When you need BareED, for example to remember important  ↩
     stuff, you activate BareED only
  (without enlarging the editor window to a maximum) and enter the letters.

  Although  BareED  is relatively quick on my system, it is possible that BareED  ↩
     is dramatically slow on yours; that
  is  due  to  the  calculations  that  must  be  done  before  something  is   ↩
     performed;  nothing  is  rigid  in BareED!
  Therefore  it  is  quite modest at consuming memory but can slow down your  ↩
     machine to its limits while you scroll
  around in the text file or enter characters.

  BareED  has  not  been designed for native Amigas (like an A1200 or A4000) but  ↩
     for accelerated machines with
  a  (for  Amigas)  quick  CPU.  As  if  that's  not bad enough, BareED should  ↩
     only be used when a fast interface to
  access  system  memory is available (forget any ZIII memory expansion card!) and ↩
      in addition - when you have
  installed a 3rd party graphic device.

      Additional  informations  can  be  found  here,  if  you  encounter  any   ↩
         problems  click  the help- button in the
       status-bar of this window.

      BareED's primitive button interface

      ARexx supported macro commands, still under construction

      Icon save and icon imagery

      An automatic revision control

      Disadvantages and faults of BareED

      Internals to BareED

      Copyright, distribution



      Click here to install BareED or only parts of this archive


$^1$) Devpac, ©® of HiSoft, UK
```

1.2 This page is under construction....

```
  - System    - the operating system, stuff to make the machine look like she is  ↩
     alive
  - ASCII one   - international standard format of characters that fit into a byte ↩
      (Latin-1)
  - Character set   - format of bit masks representing the keys on your keyboard
  - Notepad   - first Amiga word processor, 1985 (so far I know)
```

- MEmacs - text editor, 1986
- ED - text-editor for batch and script files, 1986, enhanced 1991
- Proportional fonts - visualised character set where each character can differ in width
- Fixed width fonts - visualised character set where each character has got the same width
- Font - character set that can be visualised, dropped as file
- Workbench - platform to perform things without touching the keyboard
- Console device - high level interface to simplify text handling and text output
- ARexx - Amiga specific implementation of Rexx (script interpreter)
- ZIII - expansion slot (Zorro) with 32 bit wide address range, update to ZII (24 bit)

1.3 How to deal with BareED so that no crash occurs...

To make BareED running you must firstly install (if not already done years ago) an Asl library module of at least version
38. If you did this, make sure you own OS 3 and a 68020 or better CPU. Use also the latest available SetPatch.

Click double on the icon labelled "BareED". A window appears. Here you can enter the characters; load files in - delete
or modify characters in a specific file and of course save them back.

Make sure if you try BareED for the first time on your system to firstly save any important things; furthermore, close all 3rd
party applications and make sure to have at least the minimum configuration that BareED presupposes.

If BareED crashes your system it's very likely that some applications in your Workbench start-up drawer are
incompatible with BareED.
I have encountered also a crash that occurs when BareED is started right after a commercial Basic compiler / interpreter
has been quit. The crash occurs when BareED attempts to demand memory.
It's not an error of BareED instead this compiler / interpreter corrupts the memory list of the system. So, be (again)
carefully.

If BareED appears but doesn't display the font which has been described by the icon of the file you want to load in, it is
possible that the disk font library module or the font itself cannot be opened, either because of no more obtainable RAM
or inaccessible files.

If the Asl screen mode requester module does not pop up the settings in BareED's icon have been either set to
FOREIGN or NATIVE and you don't have a 3rd party graphic board support software running - or your machine does
not support a DMA screen mode, respectively.

If you press the "Replace All" button, activate in no case the text editor window under any 3rd party graphic software

emulation! If you do however, you will encounter a software failure that will end up in a GURU! This is caused by a corrupted memory list due to invalid calls of Exec PutMsg() and GadTools GT_PostFilerIMsg(). Since BareED doesn't use PutMsg() and GT_PostFilerIMsg() on its own, it's very likely that this failure is caused by belated react and reply to Intuition messages - since BareED is heavy busy doing a job (replacing strings). BareED 0.9428 tab handling has been re-written so that it is now drastically faster (where necessary - RpPrintLine(), SetCursorXY(), CursorLineEnd()). This should solve the above stated, but on a heavy loaded system and even under a slow CPU it might fail. I've also tried to lock the layer of BareED's editor window for the time being busy, but this has caused in critical circumstances deadlocks.

If BareED crashes your system upon starting from Workbench or Shell take a look at the given stack size. BareED requires on a native Amiga with OS 3 and no 3rd party graphic emulation 2200 bytes of stack. With OS 3 and a 3rd party graphic emulation system it requires at least 3000 bytes of free stack (tested with CyberGraphX and Picasso96). Newer versions of those Amiga graphic emulation systems may need more stacks. So, if you encounter such a problem increase the stack size step by step by 1024 bytes until BareED starts up and runs correctly. This applies not only to BareED but also to all other applications that can handle both, the native Amiga graphic device and any 3rd party. In addition, BareED has got now a stack check. If BareED does not pop up when started from Workbench, the amount of free stack memory has been chosen too small. Increase the stack size in this case as stated 6 lines before. If this happens during the start out of a CLI-environment, a message will appear, telling that BareED cannot continue which such less stack (for your safety).

On my pure system (A4000 Desktop from '92) I have not installed any patches not written by me. The only exception is the Picasso96 software package (used to make that CyberVision 64/3D useable) and thus I do not encounter any problems when using BareED, in mind the hints given here.

Click here for "Amiga Rexx" related problems

1.4 Intro, Install, Requirements, Language, Use

BareED is a new text editor for your Amiga/Draco computer.
BareED is actually only available as beta application (not ready yet) - but some simple files can be edited anyhow.

It has a new concept (as opposite to other Freeware text editors)
- Designed for non-mono-space fonts (proportional fonts).
- Does not use the system's console device so that the colours can be chosen for the background, text, cursor, and text underneath the cursor and for the marked characters.

Currently it’s a really "bare" but further releases may get stronger.

Because I do not have the time to write a complete guide in how to use BareED, ↩
you should read this short introduction
carefully.

%%%

Installing BareED

(1)
To install BareED you have nothing else to do than to drag the program icon of ↩
BareED into your favourite drawer.

(2)
If you wish to store BareED into a currently non-existing drawer select "New ↩
Drawer" from the Workbench window menu.
Make sure you have got selected a suitable medium and drawer where to create ↩
the new one. If the new directory has
been created, execute step one as stated above.

To install the complete package of BareED including Rexx-script, button interface ↩
and the German catalogue simply drag
"BareED’s" drawer to your favourite position on your hard drive. No "ASSIGN" to ↩
"BareED’s" "home-directory" is needed
to run BareED!

%%%

Requirements

BareED can be used up from OS 3.0 (perhaps also up from OS 2.0 when the Asl ↩
library of OS 2.1 is installed, but this has
not been tested by me; furthermore, pen sharing and re-loading will then not work!

There aren’t any specific disk resident libraries and devices used by ↩
BareED other than those your machine was
originally equipped with.

Requirements (minimum):
  Exec   v36
  Dos   v36
  Graphics v33
  Layers   v33
  Intuition   v36
  GadTools v36
  Diskfont   v33
  Icon   v33
  Asl   v38

  CPU 68020, 512Kb RAM, OCS

Better:
  Exec   v36

```
  Dos   v36
  Graphics  v39
  Layers    v39
  Intuition   v39
  GadTools  v39
  Diskfont    v33
  Icon    v33
  Asl   v39
  Locale    v38

  CPU 68030, 2MB 32-Bit RAM, AA

Perfect:
  Complete Kickstart v40 package (OS 3.1)
  68040 with onboard RAM
  3rd party graphic card plugged into Zorro 3 or Draco direct slot
  Installed Picasso96 software package
  Asl library v41/42 (Asl v40 has got several bugs...)
  Several proportional fonts either bitmap related or outlined
  Ttf (true type font engine) package by Richard Griffith

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Installing a suitable catalogue (OS 2.1 (Kickstart v38))
------------------------------------

To  install  the  catalogue  containing  the  native  language  strings  you  have ↩
    first to create the
catalogue:

     Edit the ".ct" file
     Use   "CatComp"   utility   (or   an   equivalent   application)   to    ↩
        produce  the  "catalog"  file.  Create  a
     sub-directory on the medium where you have stored BareED called "catalogs".
     Create  in  this  new  created  directory another one that is called exactly  ↩
        as your native language, e.g.
     "dansk".
     Move the translated (via "CatComp") "catalog" to this location.
     Fully:

      "Work:Edit/Editors/BareED"    - Home-directory of BareED

      1> dir work:edit/editors/BareED
       BareED     BareED.doc
      1> makedir work:edit/editors/BareED/catalogs
      1>
      1> makedir work:edit/editors/BareED/catalogs/dansk
      1>
      1> copy ram:BareED.catalog to work:edit/editors/BareED/catalogs/dansk
      1>
      1> dir work:edit/editors/BareED
       BareED     BareED.doc
       catalogs (dir)
      1>
      1> dir work:edit/editors/BareED/catalogs
       dansk (dir)
      1>
```

```
      1> dir work:edit/editors/BareED/catalogs/dansk
       BareED.catalog
      1>
```

If you are an average user of the Amiga OS you might ask why to create a sub-directory in the drawer where BareED stays instead of using "SYS:locale/catalogs/..../BareED.catalog". The reason why is: If you delete from Workbench the BareED drawer BareED and used by it files will also be deleted, i.e. you don't have to scan through your system-partition to look for BareED used files, simple - eeh?

%%

Currently supported keys

CURSOR-UP - move cursor to previous line
CURSOR-LEFT - move cursor one position to the left
CURSOR-DOWN - move cursor to next line
CURSOR-RIGHT - move cursor one position to the right

SHIFT CURSOR-UP - move cursor to top of page, when cursor already set to this position, cursor is moved to previous page
SHIFT CURSOR-LEFT - move cursor to start of line
SHIFT CURSOR-DOWN - move cursor to bottom of page, when the cursor already set to this position,
cursor is moved to next page
SHIFT CURSOR-RIGHT - move cursor to end of line

SHIFT DELETE - delete from cursor position all characters to end of line
SHIFT BACKSPACE - delete from character left of cursor all characters to start of line
SHIFT RETURN - insert Carridge-return code (character 13) into archive

CTRL CURSOR-UP - move cursor to first character of archive
CTRL CURSOR-DOWN - move cursor to last character of archive

ALT CURSOR-LEFT - find previous word, number or single letter within current line
ALT CURSOR-RIGHT - find next word, number or single letter within current line

ALT RETURN - insert FormFeed code (character 12) into archive

Amiga RETURN - terminate line with a linefeed (character 10) and auto-indent the next line

CTRL A - arrange text to block format
CTRL B - enter right margin for block format (requester pops up)
CTRL C - change word's first letter into a capital
CTRL F - find next occurrence (find string must have been already entered in the
Find/Replace requester)
CTRL G - change letters of word underneath of cursor into capitals (association: GREAT)
CTRL K - delete from cursor position all characters to end of line
CTRL M - fence-match, point cursor to one of these characters: ([{ < > }])

```
CTRL N      - find next occurrence (find string must have been already ↩
   entered in the
         Find/Replace requester)
CTRL P      - find previous occurrence (find string must have been already ↩
   entered in the
         Find/Replace requester)
CTRL R      - replace occurrence (find and replace strings must have been ↩
   already entered
         in the Find/Replace requester)
CTRL S      - change letters of word underneath of cursor into lower ↩
   case letters
         (association: SMALL)
CTRL U      - delete from character left of cursor all characters to start of line
CTRL W      - delete word or number underneath of cursor
CTRL X      - delete current line
CTRL Y      - delete current line
CTRL Z      - arrange text to block format but in an AmigaGuide compatible manner


+++ DON'T USE ONE OF THE FOLLOWING KEY COMBINATIONS
 CTRL H   CTRL I    CTRL J    CTRL L
 CTRL O   CTRL Q    CTRL T    CTRL [
++++++++++++++++++++++++++++++++++++++++++++++++++


CTRL DELETE   - delete current line

BACKSPACE   - delete character left of cursor
DELETE      - delete current character
RETURN    - terminate line with a linefeed (character 10) (paragraph)
ENTER     - same as RETURN but with auto-indent of characters


Left mouse button - move cursor to position of mouse pointer

Right Amiga B   - start marking an area
Right Amiga X   - cut away the marked area (goes into clipboard)
Right Amiga C   - copy the marked area (goes into clipboard)
Right Amiga V   - insert earlier in clipboard remembered marked area

Double mouse click  - start marking an area

+++ A HINT +++

If you want to mark a really large number of characters, use the mouse (it's ↩
   faster):
  Click once on the character that represents the first to mark character
  Click again on this character (so called double click)
  Click mouse one character behind the last to mark character
  - Area shown in selected mark-colour
  - You can now cut, copy this area

+++ A REMARK +++

If you lay out your text using the AmigaGuide compatible mechanism (CTRL-Z) ↩
   , ensure that a brace-left character is
```

introduced through the at-sign and that the brace-left character one following ↩ brace-right character has. Otherwise, in
case not, the AmigaGuide compatible mechanism is broken off and it will become ↩ enormously difficult to restore the result
(text layout) to that of your original layout. Another disadvantage is that italic ↩ strings may not be laid out correctly.

Find Requester
---

n & Shift N - find next occurrence
l & Shift L - find previous occurrence
p & Shift P - find previous occurrence

Number Requester

Return & Carridge-return - leave requester with valid result of number gadget

%%

Settings and preferences

When "Create Icons?" is enabled BareED writes along with the archive the settings ↩ you have chosen for this archive, for
example the colours.
BareED offers no possibility to choose the global preferences with three ↩ exceptions: You can enter into BareED's icon
the monitor type you wish the Asl-requester-module to display - and whether you ↩ like to reserve the pens taken for the
"knob-bank", so that other application wont be able to use these pens and no ↩ false colours occur. In addition, you can
tell BareED whether you like to start with the button interface enabled.

All three characteristics are entered normally through the use of the tool types.

MONITORTYPE=ALL|FOREIGN|LIKEWB|NATIVE

Where

ALL ...to display each monitor who is available on your machine

FOREIGN ... only display non-Amiga modes, i.e. you must have already plugged ↩ into a Z3 bus
a 3rd party graphic device (for example)

LIKEWB ... allows only to display such modes which will be supported by the ↩ Workbench, i.e.
no HAM, EHB, DPF, 15 bit and modes with an alpha channel

NATIVE ...to display only those monitors that can be directly displayed by ↩ the Amiga hardware

Combinations of the above stated are allowed, such as

```
  MONITORTYPE=FOREIGN|LIKEWB  (e.g. for the Draco computer)

    Or

  MONITORTYPE=NATIVE|ALL

Please do not combine NATIVE and FOREIGN....


KNOBPENS=RESERVE

  ... instructs BareED to reserve the pens taken for the knob-bank - in order ↩
     to avoid each time re-mapping the
  colours for a newly generated surrounding in case other applications have ↩
     successfully attempted to reserve
  pens for their own purpose, e.g. image viewers (like MultiView), Workbench ↩
     games, icon patches (NewIcons) or
  OS 3.5 icon subsystem.

BUTTONS=YES

  ... instructs BareED to turn on and to display the button groin upon start


The option "IBM compatible Save" allows to transmit your archive from one ↩
    computer platform to another without making
the significant changes on your own. This means, however, too, that the ↩
    complete layout of your archive will disappear
(on your Amiga) but it becomes on an IBM computer very much resemble ↩
    to those on the Amiga created. Those
changes are not visible onscreen, however, when you invite the so modified file ↩
    again! BareED does not do now anything
other than to remove all line terminators (LineFeeds) for a paragraph. Because ↩
     it is in fact difficult to find out which lines
appertain to a paragraph on the Amiga-platform, you must prime first your ↩
    archive. You eliminate all non-standard text
features (for example Escape codes and AmigaGuide macro / attributes). ↩
    As next you select then the right area
boundary in the dialog box (if you use the Helvetica font, you use a right ↩
    edge of 153 signs). Then change your text so
that a block format comes out in this case (tabs are in this case ↩
    allowed). Backup now your archive onto a medium.
BareED will combine now all lines to an individual paragraph, that are ↩
    not split into two parts and do not exceed your
defaulted area boundary.


*IBM    ® International Business Machines

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Because this manual is very short it's recommended that you use method "trial and ↩
     error" when you use BareED. BareED
works just like other text editors.
```

1.5 BareED's button interface

Before you start to use this feature of BareED, you have to make sure that you own a 68040 or better CPU, the operating system of version 3.0 or better and at least 130 Kbytes of free memory (for the graphical details). Ensure also that you are using a 256-colour screen with a resolution of at least 800 x 600 pixels (do not use the button interface if you only have one of the native Amiga graphic devices!). If this does not scare you, then just read ahead.

BareED offers a button interface that is disabled by default, due to the memory consumption and the time needed to set up this button interface. If you own a 68040+ CPU, however, you may wish to use it.

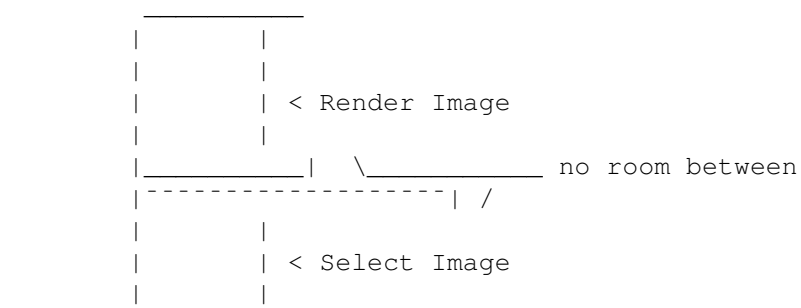
No image for the button bank of BareED is limited to something. This means that the imagery may differ in width, height, depth and colour.

By default no pens of the knobs are reserved in order to give any picture viewer the ability to view any imagery in the closest colour match. This can lead BareED to view those knobs in false colours when there are no more pens free, or when a picture viewer overrides BareED used pens - but this can be changed by BareED at any time, unfortunately, you have to explicitly tell BareED if the button bank appears ugly. If you don't agree with this, use the tool type setting that forces BareED to reserve these pens.

You should, although there is no such hard coded limit, limit your image to a height that can fit into the window when there is also enough room to display one, single text line.

The imagery themselves have to be created through the use of an available paint program, e. g. Personal Paint. Your render image and the select image have to be of same size. You must define the border for the knob on your own - if you want a border to separate the buttons.

When you have made your imagery, backup the render and the select image for each knob to disk as a single file, where the render image is placed at the top and the select image at the bottom of this new created picture - and no row splits them up!



```
  |       |
   -------------------


DETAILED DESCRIPTION:
-----------------------------------------


BareED's  button interface is nothing else than a groin with graphic symbols.  ↩
    While clicking one of these symbols, BareED
carries out an action, e.g. opening a file requester.

BareED offers a button interface that can be changed by the user to form his  ↩
    individual wishes.
Therefore  BareED  allows you to change the imagery for the button interface knobs ↩
     and the order of the knobs. In order to
achieve  this you must edit the text file "BareED.cfg", e.g. with BareED itself, - ↩
     which must be placed in the root directory of
the executable file BareED!
If  you  turn  on the menu item "Use Knob-Bank?", BareED will read in the file " ↩
    BareED.cfg" and examine this file according
to codewords.
Therefore,  indices  are very important since no file names are defaulted but  ↩
    these used indices point at intern functions of
BareED.
Indices,  which  are  supported  by  BareED,  are  settled  in the field of 1 to  ↩
    63; BareED sets up functions at your disposal,
which DOS commands can fire up beginning with index 40.
The  indices,  beginning  with index 40, ARexx macros can also be assigned by  ↩
    means of the DOS command "RX", so that
through a simple mouse click on one of these buttons a macro direct can be started ↩
    .
A  command  line  that  is  understood  by BareED must be introduced by the index  ↩
    (the index is a usual number), followed
from a comma that itself the path and file name of the graphic symbol of this  ↩
    function connect.

A valid coding line would then look like this:

  3,SYS:Images/Save.btn

where  number  3  represents  the  index,  and  "SYS:"  indicates the partition of ↩
     the hard disk. The directory "Images" as a
consequence,  contains  for  our  example  another file by the name of "Save.btn"  ↩
    that the graphic symbol which a function
was assigned contains.
For  the  case,  that  you  do  not like the symbols enclosed by me, you proceed  ↩
    as follows in order to replace the symbols
made out by me through your own symbols:

The first step that you must do is to start a paint program of your choice in  ↩
    order to paint two small pictures.

The  colour  palette  chosen  by  you  is  only being of secondary importance,  ↩
    however, you should choose these pictures
less than 48 screen points in level and upright direction.
It goes without saying that the two pictures with it should have the same scales.
```

Put the two pictures directly on top of each other so that no space occurs between the two pictures.
Now take the two pictures as one pencil and store it in the IFF format with at the most 256 colours.

Now copy the enclosed tool "BitMapSaver" into the C-directory of your System, e.g.

  Copy ...../BitMapSaver to C:

You invoke the tool BitMapSaver as follows:

1> c:bitmapsaver NAME_OF_IMAGE to NAME_OF_KNOB opt raw rawhdr pen cmap rgb32

If the path- or filename of the image file contains spaces, enclose them in double quotes.

The tool BitMapSaver will create an image that can directly be read by the program BareED.

The next step you have to do is to edit the text file "BareED.cfg" which describes this function and the image you want to
change.

We assume as an example that you would like to change the button which describes the backup function.

The index which indicates the backup function was determined by me as function 3. We now search in the file
"BareED.cfg" for the line which begins with number 3. According to number 3, followed by a comma, the path and file name
of the image file used currently stands.
Change the path and file name so that this now points at your newly defined image file. If you now start the program
BareED and turn on the button groin, your image file appears instead of the made out by me.

As mentioned above already, allows BareED it you to carry out a DOS command, by which the required coding line is
hardly treated more heavily than a normal one that points at an intern function of BareED.

In memory; a valid coding line consists of the index which points to a function and the file name of the image file!

nn,path&filename

New is now, that according to the file name of the symbol (image file) separated by a space character and included in
parentheses, a valid AmigaDOS coding line follows.

nn,path&filename (AmigaDOS_command_line)

In case you want to have an ARexx macro file executed by means of the DOS command "RX", the correct coding line
reads like this:

40,path&filename_of_image_file (SYS:RexxC/RX your_rexx_macro.rx)

```
Please  do  not  forget  that  only  the indices which are settled from index 40  ↩
   up to index 63, will allow it, AmigaDOS coding
lines to carry out.

By  the  way,  if  you  would  like  to  start a ARexx macro file by means of the  ↩
   "RX" command, you indicate as command file
"SYS:RexxC/RX"  where  the word "RX" should be capitalized because BareED then  ↩
   starts automatically the Rexx server -
if the eleventh and the twelfth character to the word "RX" show!

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

CHANGE ORDER:
-----------------------------

To  vary  the order of the buttons, it satisfies, the line that the index of the  ↩
   corresponding symbol represents to displace in
the file "BareED.cfg".

Example:
The symbol which saving represents should be strobed before the symbol which  ↩
   loading represents.

The only you have to do, is the line that begin with number 3 (save) to present  ↩
   that line, that begins with number 2 (load).

2,knobs/Load.btn
3,knobs/Save.btn

  changed to

3,knobs/Save.btn
2,knobs/Load.btn

Easy, isn't it?

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

REMARKS:
------------------

In  order  to  insert  remarks  into  the  file  "BareED.cfg"  it  satisfies  ↩
   remarks  to  turn  a  semicolon on. BareED will ignore
everything  behind  a  semicolon  up  to  end  of  line.  Semicolons  are  also  ↩
   suitable to eliminate statements instead of to
remove completely this!

Examples:
2,knobs/Load.btn  ; the load image
; 3,knobs/Save.btn  removed save image off button bank

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

INDICES:
-------------
```

```
Following indices are assigned to these functions:

* 0 exit BareED
* 1 move gadget within BareED's right window border
  2 load a new file
  3 save existing file out of editor
  4 create a new, blank editor surround, ready to enter characters
* 5 close the current editor surrounding
  6 print document
  7 select screen mode and use it
  8 select tab-step and use it
  9 select a new font and use it
  10  select pencils for document
* 11  change palette of screen
  12  snapshot marked block and save it as clip to the clipboard
  13  insert clip in clipboard into current document
  14  undo one, remembered action
  15  write marked block to disk
  16  insert a file into the current document
  17  open the find-requester
  18  open the replace-requester
  19  open the go-to-requester
  20  arrange current paragraph to block layout
  21  arrange current paragraph to block layout but in an AmigaGuide compatible ↩
     manner
* 22  open requester to execute an ARexx-macro
* 23  set global preferences

(* = Unsupported)

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

BACKGROUND PATTERN:
-------------------------------------------

BareED let you also choose the background pattern and colour beside the ↩
   imagery and the vertical and
horizontal distance between each image and the border of the knob-bank. There are ↩
   synonyms to choose it.

FILPAT    synonym for fill-pattern
PATPEN  synonym for pattern-pen
VSPACE  synonym for vertical spacing
HSPACE  synonym for horizontal spacing
ONEPAL  synonym for one palette


In order to make them available to BareED these synonyms have to be entered ↩
   in the same file where the knobs have
been described, Button.cfg .

An example could look like this:

FILPAT=0
PATPEN=0
VSPACE=0
HSPACE=0
```

```
ONEPAL=0

The  FILPAT  is a long word representing a mask for set and unset pixels. It can  ↩
   be entered as a decimal, hexadecimal or
dual number.

FILPAT=1431677610 ; hexadecimal ($) 5555AAAA, dual (%)  ↩
   01010101.01010101.10101010.10101010
PATPEN=2    ; use a white pattern (on my system)
VSPACE=1    ; use between each image (and between the pattern border and the image ↩
    itself)
      ; one vertical line
HSPACE=1    ; ditto for horizontal
ONEPAL=1    ; all imagery will have the same palette thus tell BareED to compute  ↩
   the palette
      ; only once (30% faster)

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

GUIDELINES:
-----------------------

I  strongly insist you to follow these guidelines:
  Rexx-scripts  for  BareED  should  be placed in a single drawer and this drawer  ↩
     should be in the root directory of
  BareED, e. g. :

    Work:Tools/BareED/    <- main (home) drawer of BareED
    Work:Tools/BareED/BareED  <- objectfile of BareED
    Work:Tools/BareED/Button.cfg  <- configuration file
    Work:Tools/BareED/knobs/  <-drawer for knobs
    Work:Tools/BareED/rexx/   <-drawer for rexx scripts
    Work:Tools/BareED/rexx/Info.rx  <- a rexx script with a suffix

By  the way, if your rexx script does not contain a suffix "ARexx" will add one on ↩
    its own. This suffix is ever ".rexx". So "Info"
becomes "Info.rexx". You should remember this when you choose a name for your rexx ↩
    script.

Now  something  smart:  BareED  will  transmit  his  Rexx  port  name  into  the  ↩
   clip  variable  "BAREED".  The clip variable
"BAREED"  refers  in this case always to the running copy of BareED which you used ↩
    with the button interface (knob bank)
in order to call the Rexx- command / script. Example:

/* Rexx Demo - first line */

BareED_HOST = GetClip('BAREED') /* Get the name, e.g. BareED.1, BareED.2 and so on

IF BareED_HOST = '' THEN DO   /* Ensure we got it */
  CALL SetClip('BAREED')    /* Otherwise remove it from ClipNode */
  EXIT 5        /* Error, no name! */
  END

ADDRESS VALUE BareED_HOST   /* Refer from now on to this running copy of BareED */
```

```
CALL SetClip('BAREED')      /* Remove from ClipNode */

.... here you can now let you script start
```

Of course it's not necessary to remove the clip via SetClip(BAREED) but this saves memory thus I use it.

1.6 The Amiga Rexx Interface of BareED

First of all, this chapter and in general the ARexx-interface of BareED are under construction!

ARexx scripts can indirectly drive BareED, but at the moment BareED does not offer to execute ARexx scripts with one
exception: the button interface of BareED gives you direct access to such scripts.
BareED is able to deal with strings containing up to three macro-commands where the first is called "Command-Index",
the second "Object-Index" and the third "Parameter-Index".

So instead of writing:

```
GetAmountChars
 You write
Get Amount Chars
```

Which is first better to read and second for me easier to parse. The disadvantage is that it may collide with reserved
ARexx symbol names or functions, e.g. Set Error Off, where ·"ERROR"· is a reserved ARexx command.

One goal of BareED's ARexx interface is the non-case-sensitive handling of passed strings. So you can write:

```
  get amount chars
```

Or

```
  'get amount chars'      «« enclosed in single quotes
  'get aMOunt chARs'
  'GET AMOUNT CHARS'
```

It's non important to BareED! BareED splits a command line (string) into the above stated index' where hash values are
used instead of simple string comparison.

Currently BareED has got an ARexx interface with more than 80 macro-commands but only a few call the appropriate
routines (under construction). Supported are:

```
        additional string
        ------------------------------|
              |
Command-Index Object-Index  Parameter-Index «astr»  RESULT
-------------------------------------------------------------------------
```

```
SET
    ERROR   ON  |  OFF    -none-  ( useful when debugging )
    ECHO    ON  |  OFF    -none-  ( useful when debugging )
    FONT    "times.font,13"   -none-  should be clear....
    FONTSTYLE 0,1,2,3 or combined -none-    try!
    CLIPUNIT  0 - (almost) ~    -none-  0 suggested for textfiles
    FIND    STRING    "string"  -none-  the string to look for
    REPLACE STRING    "string"  -none-  find string replaced with this one

    FINDMODE  -none-      -none-  Set default mode, case sensitive but no word  ↩
       only search
        NORMAL      -none-  non case sensitive search
        WORDONLY    -none-  word only search

    MARGIN  RIGHT   "number" -none- set right margin in number of letters
    TAB   SIZE    "number" -none- "number" in amount of characters
    BLOCKSTART  -none-      -none-  set start point for marked area or erase it
    BLOCKEND  -none-      -none-  set end point for marked area or erase it (not  ↩
       necessary, anyhow...)
-------------------------------------------------------------------------------
GET
    ARCHIVESTART -none-     address memory where characters start
    ARCHIVEEND  -none-      address memory where characters end
    REGION    -none-      address allocated memory for storage
    REGION    SIZE      amount allocated memory for storage in bytes
    ARCHIVENAME -none-      filename without path
    FILENAME  -none-      filename inclusive path

    AMOUNT
        CHARS     amount characters in archive
        LINES     amount lines in archive
        CHANGES   amount of modifications
    CURRENT
        CHAR      RC    single character (RC unequal zero if end of archive)
        LINE      RC    complete string line (RC unequal zero if end of archive
                       or only a paragraph)
        COLUMN    offset in number of spaces in current line
    CURSOR
        X     offset (in number of characters) in current line
        Y     current line

    BVERSION  -none-      packed: version (high word), revision (low word)
    FONT    -none-      string: fontname - terminated by a comma - then font  ↩
       height
    TAB   SIZE      amount of space characters a tab takes up
    CHARWIDTH "char"      characters width in number of pixels
    BLOCK   -none-      add contents of clip to archive
-------------------------------------------------------------------------------
PUT
    CHARS   "string"      -none-  add string to archive
    CHAR    "Q"     -none-  add single character to archive
    BLOCK   -none-      -none-  copy marked block (written to clipboard)
-------------------------------------------------------------------------------
LAYOUT
    NORMAL  -none-      -none-  lay out paragraph to normal block format
    GUIDE   -none-      -none-  lay out paragraph Amiga Guide compatible
-------------------------------------------------------------------------------
```

```
MOVE
    CURSOR
        LEFT      -none-
        RIGHT     -none-
        UP      -none-
        DOWN      -none-
        LINESTART   -none-
        LINEEND   -none-
        PAGESTART   -none-
        PAGEEND   -none-
        ARCHIVESTART  -none-
        ARCHIVEEND    -none-

    BLOCK   -none-      -none-  cut and copy marked block (written to clipboard)
-------------------------------------------------------------------------------
GOTO
    linenumber  -none-      -none-
    LINE    linenumber    -none-
    BOOKMARK  1 to 10     -none-
-------------------------------------------------------------------------------
DELETE
    CURRENT
        CHAR      -none-
        LINE      -none-
        WORD      -none-
    TO
        LINEEND   -none-
        LINESTART   -none-
    CHAR
        LEFT      -none-
        RIGHT     -none-

    LINE    -none-      -none-
-------------------------------------------------------------------------------
LOCK
    ON    -none-      -none-  forbid modifications through user interface
    OFF   -none-      -none allow it
-------------------------------------------------------------------------------
ACTIVATE
    WINDOW  -none-      -none-  input stream set to editor window

-------------------------------------------------------------------------------
FIND
    NEXT    WORD      RC   unequal zero if none found
    PREVIOUS  WORD      RC   unequal zero if none found
    NEXT    STRING      RC   unequal zero if none found
    PREVIOUS  STRING      RC   unequal zero if none found
-------------------------------------------------------------------------------
REPLACE
    NEXT    -none-      RC   unequal zero if none found
    ALL   -none-      -none-  but requester pops up
-------------------------------------------------------------------------------
SAVE
    -none-    -none-      -none- (but file-requester pops up!)
    "filename"  -none-      -none-
-------------------------------------------------------------------------------
NEW   -none-    -none-      -none-
```

```
    "filename"  -none-      RC    unequal zero if "filename" could not be loaded ↩
       in
-------------------------------------------------------------------------------
QUIT    -none-    -none-      -none-
-------------------------------------------------------------------------------
TELL
    "string"    -none-      -none-
-------------------------------------------------------------------------------
CASETELL
    "string"    -none-      1 = yes, 0 = no
-------------------------------------------------------------------------------

Some notes at this point:
  With  "storage"  I  refer  to the allocated memory region where characters can ↩
     be placed within. It is not identical
  to  the  physical  address  of  the  first  character  of  the  file  that  has ↩
      been  previously  loaded in, or to the first
  character you have entered in the text editor window.
  Instead,  "archive"  represent  the  group  of  single  characters  that  are  ↩
     combined  together and which can be
  written to a medium as a single file.

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

The find/replace functions are still under construction (as the complete BareED ↩
    Rexx port).
The macro "FINDMODE" offers three choices:
    FINDMODE - without any arguments, to set the initial state: case sensitive, no ↩
        search for "words only"
    FINDMODE - NORMAL, to search case insensitive
    FINDMODE - WORDONLY, to ignore combined words

I  strongly  urge  you  to  call  the  FINDMODE  command  without  an  argument  ↩
    before  you  begin  to search for a string,
otherwise you run with unknown settings and unexpected results may happen.

move cursor archivestart
set find string "BareED"
set replace string "bare editor"
set findmode
set findmode wordonly

reps = 0

do while RC = 0
  find next string
  if RC ~= 0 then
    break
  replace next
  reps = reps + 1
end

tell "Found" reps "occurrences to replace!"

NOTE:
  If  you  use  "REPLACE  NEXT" on its own - and currently the cursor does not  ↩
     point to the string you are looking
```

```
  for,  "REPLACE NEXT" will only move the cursor to the next occurrence without  ↩
     replacing it. No error is returned,
  which means that you cannot distinguish between a simple cursor move and an  ↩
     exchange.

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

If you want to obtain information's from BareED then don't forget to use:

OPTIONS RESULTS

Otherwise BareED does not return a value or string. Following is an example that  ↩
   cares about all hints above stated:

/* Demo - First line */
address BareED.1  /* Refer from now on to first running copy of BareED */
options results   /* Tell BareED to return values in case requested */

'set echo on'   /* BareED should display any incomings */
'set error off'   /* Pass through any encountered errors */

get amount chars  /* Get amount of used chars, warning: zero possible - when  ↩
   archives empty! */

amount = result   /* AMOUNT  is a macro name of BareED, here used as variable, so  ↩
   the next time
         we refer to amount; we refer to the variable and not to the macro name!!! ↩
             */

'get amount lines'    /* Using single quotes prevents ARexx to refer to the  ↩
   variable, the string is passed
         through to BareED as it! */
lines = result    /* LINES is also a BareED macro name, so see above */

get archivestart    /* Get memory address of first character or letter, thus  ↩
   archive pointer */
aptr = result

say "Archive at 0x"D2X(aptr)", size in bytes:" amount "- where the archive  ↩
   contains" lines "lines."


       Click here for "Amiga Rexx" related problems


By BareED reserved keywords in alphabetical order.

ACTIVATE  ALL   AMOUNT  ARCHIVE   ARCHIVEEND
ARCHIVENAME ARCHIVESTART BLOCK    BLOCKEND  BLOCKSTART
BOOKMARK  BVERSION  CASETELL  CHANGES CHAR
CHARS   CHARWIDTH CLIPUNIT  COLOR   COLORS
COLUMN  COMPUTE CURRENT CURSOR  DELETE
DEPTH   DOWN    DRAWERNAME  ECHO    ERROR
FILE    FILENAME  FIND    FINDMODE  FIRST
FONT    FONTSTYLE GET   GOTO    GUIDE
HEIGHT    INACTIVATE  INFOWINDOW  INITIALX  LAST
LAYOUT  LEFT    LEFTX   LENGTH  LINE
```

```
LINEEND LINES   LINESTART LOCK    MARGIN
MARK    MOVE    NEW   NEXT    NORMAL
OFF   ON    PAGE    PAGEEND PAGESTART
PENS    POSITION  PREVIOUS  PUT   QUIT
REGION    REPLACE REQUEST RESTORE RIGHT
RPORT   SAVE    SCREEN  SET   SIZE
STRING    TAB   TELL    TO    UP
USED    WIDTH   WINDOW  WORD    WORDONLY
WORDS   X   Y


Reserved symbols by ARexx - not available to BareED

Abbrev()  Abs()   Addlib()    Address Address()
AllocMem()  Arg   Arg()   B2C()   BAddr()
BitAnd()    BitChg()    BitClr()    BitComp() BitOr()
BitSet()    BitTst()    BitXor()    Break   Break_C
Break_D   Break_E   Break_F   C2B()   C2D()
C2X()   Call    Center()    Centre()    Close()
ClosePort() Compare() Compress()  Copies()    D2C()
D2X()   Datatype()  Date()    Delay()   Delete()
DelStr()    DelWord() Digits()    Do    Drop
Echo    Else    End   Eof()   Error
ErrorText() Exists()    Exit    Export()    Find()
Forbid()    Form()    Forward() FreeMem() FreeSpace()
Fuzz()    GetArg()  GetClip() GetPkt()    GetSpace()
Halt    Hash()    HI    If    Import()
Index()   Insert()    Interpret   IoErr   Iterate
LastPos() Leave   Left()    Length()    Lines()
MakeDir() Max()   Min()   Next()    Nop
NoValue Null()    Numeric   Offset()    Open()
OpenPort()  Options   Otherwise Overlay() Parse
Permit()    Pos()   Pragma()  Procedure Pull
Push    Random()  RandU()   RC    ReadCh()
ReadLn()  RemLib()  Rename()  Reply()   Result
Return    Reverse() Rexx...   Right()   RX
RXC   RXSET   Say   Seek()    Select
SetClip()   Shell   Show()    ShowDir() ShowList()
SigL    Sign()    Signal    SourceLine()  Space()
StateF()    StdErr    StdIn   StdOut    Storage()
Strip()   SubStr()    SubWord() Symbol()  Syntax
TCC   TCO   TE    Time()    Trace
Trace()   Translate() Trim()    Trunc()   TS
TypePkt() Upper()   Value()   Verify()    WaitPkt()
When    Word()    WordIndex() WordLength()  Words()
WriteCh() WriteLn() X2C()   X2D()   XRange()
```

1.7 Amiga Rexx problems

```
__
 I  Unknown command
----
```

When you encounter a problem that you cannot track down it's very likely that a function either of the rexx master or a rexx support library returned an error or even no error in the RC variable. In case that occurs ARexx tells you that the command (here BareED) returned the value 10.

For example I used this and similar fragments which caused a lot of error messages until it was solved:

```
Address BareED.1
Options Results

Delay( 2)
Put Char '0A'X
```

Seems to be okay to me. I opened the "rexxsupport.library" and therewith the Delay() function could be used. The mistake that I made was that I didn't cared about the RC variable where a function passes the result of the function.

In the ARexx manual is stated that a returned function code may not be explicit called, it's automatically done. I know that but never thought that the result of Delay() is passed immediately to BareED. In the example code above the ARexx-Server made following of the code:

```
Address BareED.1
Options Results

Delay( 2)
0       <<<< !!!!
Put Char '0A'X
```

When BareED encountered the character zero ("0") it didn't know how to handle it so it returned 10 (not known by me)! This mistake is not only visible when "Options Results" is used (application function, do and return a result).

This problem to solve is very easy; call a function so that ARexx knows that you don't care about a result. In the code fragment above it is done through:

```
Address BareED.1
Options Results

Call Delay( 2)      <<<< !!!!
Put Char '0A'X
```

The "CALL" command indicates that the RC variable has got no influence on the further "programme course". Thus RC is taken as unset by ARexx and therewith ignored, which leads ARexx to continue with "Put Char '0A'X" instead of "0". Nevertheless, the return code of the function Delay() can be check; it's not placed in the RC variable but in the RESULT variable!

___
 I I BareED’s macros become not recognize

As already stated somewhere in this document, BareED has got a non case- ↩
sensitive interface to Arexx where a letter
is ever treated like it would have been entered in upper case. The ↩
next is that BareED doesn’t use single macro
commands, e.g. "MoveCursorArchiveend", instead it will support "Move Cursor ↩
Archiveend". This has got the advantage
to be more readable but can put anybody in trouble if he/she doesn’t know which ↩
symbols ARexx and BareED reserve!

So a simple line like:

Set Error Off

Will cause trouble because "ERROR" is a reserved ARexx and BareED macro name! To ↩
avoid this enclose any probably
from both used macro name in single quotes, e.g.:

’Set Error Off’

Now this string is passed through to BareED instead of being analysed by ARexx, ↩
because ARexx treats the three words
as one single line and therewith as a macro (which it does not understand).

Also, variable names can cause trouble, e.g.:

```
Get Amount Lines
Lines = RESULT    < OK

....bla bla bla

Get Amount Lines    < ERROR
Lines = RESULT
```

In the example I misused a reserved macro name as variable name, "Lines". ↩
Until the second "Get Amount Lines" is
encountered all goes as it should, when now ARexx analysis’s the second ↩
"Get Amount Lines" it encounters the
"Amount" macro name followed by the "Lines" variable. If "Lines" has got the ↩
value 1300 ARexx would pass this string to
BareED:

```
Get Amount 1300
```

Of course BareED cannot handle this. To avoid this there are several ↩
solutions, first: do not use variable names, which
will also be used as macro commands by BareED, second: enclose the macro ↩
name, which collides with the variable
name in single quotes, e.g.:

> Get Amount ’Lines’ <, or which is in my opinion better, enclose all macro names ↩
in single quotes:

```
'Get Amount Lines'

As already told, it doesn't bother BareED if you write in lower or upper case or ↩
    in mixed form. So you can even write:

'gEt aMoUnT lineS'

BareED knows how to handle this.


_____
 I I I  Not know problems by author
-------

Room for your extraordinary experiences with BareED and ARexx...
```

1.8 Disadvantages and mistakes - oh no

```
Disadvantages and mistakes, known to the author

Cannot  be  used  in connection with Nico François-PowerSnap-Versions-xx. This  ↩
    means only, that PowerSnap cuts out
not the correct characters to discover, because BareED reserves between every line ↩
     of text a separate row.

Marking  a  field on a screen with less than 4 colours causes a small problem:  ↩
    Cursor and marked field are represented in
the same colour. Thus you cannot distinguish where the marked field ends.

Blank lines are not represented in the selected marker colour if these are within  ↩
    a marked field.

Only  decimal  numbers  can  be  entered  as  tool  type values: as I already  ↩
    mentioned in the source code of BareED, the
functions  provided  by  my  compiler-provider  cause a complete computer-crash as ↩
     for example "atoi". Therefore I used
the  built-in  OS  function  StrToLong() the DOS library. This function accepts  ↩
    only decimal numbers. With the introduction
of OS 4.0 this function could support also dual and hexadecimal numbers.

Slow  delete  /  insert  from characters: Currently BareED copies and removes  ↩
    every character immediately. This could be
changed  with  a  line buffer. The problem with a line buffer is that this would  ↩
    have an inflexible size - the line, however, not,
hmmm.  The  next  problem is that the inner cursor of BareED which represents the  ↩
    visible cursor will not have any ground
under its feet in this special case. So I do not like to re-work at the moment the ↩
     routines.

A  subfunction  added  -  that  was  written  in  assembler  -  which  reduced  ↩
    drastically  the time to displace and to copy
characters.
```

Pen selection not very friendly: A plan for a future version of BareED is to offer a friendlier interface in order to choose the pens.

If you press a key and nothing happens, then the system cannot assign any more memory to BareED. In this special case it is also not anymore possible to backup that archive onto disk since the "Asl"-file-requester cannot be shown due to the low available memory. Perhaps a future version of BareED spends a warning if you run danger to fall below a lower memory limit.

The whole user interface of BareED is font-sensitive; BareED uses the title font of the screen, this may be proportional. If the font is now so big that the window that is computed cannot be opened - basing on the width and height of this font -, BareED will not resort on the font topaz-8 and compute a new window due to the new scales. The next bug is, that it not spends this mistake.

After all, currently BareED informs you only about occurred mistakes that are caused and treated by the Amiga-DOS.

System function ObtainBestPenA() inconsequent: First four and last four colours are used always from intuition screens (MultiColor mode); when ObtainBestPenA() is used, existing colours (pen register) are not returned even if they are the same as the demanded ones. Thus I must do myself my thoughts for the future how to implement a better relief of sharing screen pens.

Existing pen number are not reduced onto the actual, represent able number of coloured pencils. This happens then, when you use a screen depth of for example 32 colours at the present time and then switch back to (say) 8 colours.

At the moment BareED supports not more than 256 pens of a screen, this becomes change in (far) future, when BareED allows too, to modify the colour-values for the pens.

Created catalogue, which one was created with 'CatComp' and where a short cut is invalid for a keyboard contraction (or twice uses) is not recognized by BareED as such - and thus either does not patch!

Underscorings for short cuts are not set and handled although GadTools 37 could be available.

The editor-window can be damaged if a font is supposed to be represented with the connotation 'italic'. This is due to the fact that such a font can show a bend to the right side - some fonts even draw also some points out to the left side!

Currently, BareED attempts calculating the needed room to the right and left of the drawing area in once, however, it can fail.

```
Render engine was written in C ('CHUNKY TO PLANAR') and with that it is slow. ↩
    The next one is that this routine always
uses eighth 'bit' deep 'bit-planes' - even if the target area is not so deep.

BareED is very susceptible to mistakes made of third party software. ↩
     This can result in crashes. Are warned!
Non-appropriate to rules and inconsistently written patches can initiate ↩
    often BareED to the descend! Since BareED
works only with the memory that was assigned to him by the system, no third ↩
    application has the right to damage memory
requested by BareED.
BareED uses the distributed resources in their full size. The ↩
    uncontrolled changing only of such a byte can end in a
disaster.
```

1.9 BareED is able to save icon imagery

```
BareED has got as default a 4-colour built-in icon image. If you, for example, ↩
     prefer icon of MagicWB, NewIcon or even
GlowIcon type you would be disappointed when BareED would only save a 4- ↩
    colour icon image to disk. To solve the
problem, I implemented a routine which checks first if there is already ↩
     an icon image on disk, i.e. you save a newer
version of the text file. If it is, this one is taken instead of the built ↩
    -in 4-colour icon image. If there is currently no icon
image on disk, BareED checks if in its home directory is a directory ↩
    labelled "defs" that contains an icon image which
suffix correspond with the one of the file you are going to write to disk. Example ↩
    :

You want to save the file "BareED.guide" to disk.

In this example BareED will look for an icon in the drawer "defs" ↩
    labelled "def_guide.info". The file "def_guide.info" is a
normal Workbench icon. If BareED finds the file, this file (icon) is ↩
    saved along with the text file instead of the 4 colour
built-in icon of BareED.

You should note that a suffix must not be longer than 6 characters:
  BareED.guide    -> 5, ok
  CpyLib.asm    -> 3, ok
  Startup.c   -> 1, ok
  Kernal.cpp    -> 3, ok
  CreateKnobs.script  -> 6, ok
  Man.postscript    -> 10, wrong - here the 4-colour icon would be used!

In the above example these icons must be present in the "defs" drawer:
  def_guide.info
  def_asm.info
  def_c.info
  def_cpp.info
  def_script.info

From Workbench those files will be viewed as:
  def_guide
```

```
  def_asm
  def_c
  def_cpp
  def_script
```

In order to save an icon image to disk the menu item "Create Icons?" must be turned on. You should pay attention that
no tool type of an existing icon is stored in the new created one.

1.10 Simple Revision Control supported by BareED

As I forgot often to update the date of revising and the revision number itself when I enhanced or only fixed bugs in
BareED source code, I modified BareED in a manner so that it now does this task automatically.

Before you can use this feature of BareED you have to enable it. This is very easy but will not be done for you by BareED
automatically.

First, open the icon of the file that you want BareED to change each time after having loaded it into memory. Enter
somewhere in the tool types:

```
  RVCTYPE=
```

The RVCTYPE describes the desired output type of the date string; it can be »DOS«, »INT«, »USA« or »CDN«. It
depends on the place where you live since I didn't find a date string format which will be accepted all over the world by
the »Version« command due to the different language date formats.

For example you choose »DOS«, then the type should be set as follow:

```
  RVCTYPE=DOS
```

The next step is to choose your lines where the modifications should occur.

Open with BareED your file and remember all lines that should be updated.
Enter in the icon of the file:

```
  REVCONTROLn=
```

where 'n' is a count from zero to nine that belongs to the line that should be changed; example: Line three and seven
should be changed, so enter in the icon of the file:

```
  REVCONTROL0=3
  REVCONTROL1=7
```

Now save these changes.

```
Open with BareED the file; if BareED finds any things to change in your specified ↩
    lines it will do it.

NOTE: The date string format cannot be changed by BareED, this means if ↩
    you're currently using the
  DOS type but you prefer the Canadian type, you have to change it per hand first:

  (23-Sep-96) to  (23.09.96)

  otherwise BareED only ignores this section completely, without any changes!

  Note also that a date string must be surrounded by parentheses, otherwise BareED ↩
      continues without changes.

  Since a revision control should be used for revisions, BareED updates the number ↩
      behind the dot, not in front!

  1.33  to  1.34    but not
  1.33  to  2.33 !    - no no

  There is a limit for the version part of 3 characters, therewith in ↩
      range from 1 to 999. Any version number
  greater 999 will put BareED into trouble.

  998.144 changed to  998.145   but
  1052.144  changed to  1053.144 ! - So watch out.

  BareED can change one value per line, the first that is found will ↩
     change. If a date string in the right format
  follows, this one will be changed, too.
  It's also okay to BareED if a date string stands on its own, examples:

  "$VER: BareED-doc 1.01 (23-Sep-96) Copyright Jörg van de Loo"
  will be changed to (example)
  $VER: BareED-doc 1.2 (25-Nov-00) Copyright Jörg van de Loo"

  or

  #define Revision 145
  will be changed to
  #define Revision 146

  or

  #define VerRevStr "33.37"
  to
  #define VerRevStr "33.38"

  and

  (23-Sep-96)
  to
  (25-Nov-00)


_____________________________
Supported date strings formats are:
```

```
-----------------------------------------------

 /* Demo DOS */
------------------------
#define ID "$VER: BareED-doc 1.01 (23-Sep-96) Copyright Jörg van de Loo"

#define Version 1

#define Revision 01

// ---end


/* Demo  INT */
---------------------
#define ID "$VER: BareED-doc 1.01 (96-Sep-23) Copyright Jörg van de Loo"

#define Version 1

#define Revision 01

// ---end


/* Demo  USA */
------------------------
#define ID "$VER: BareED-doc 1.01 (09-23-96) Copyright Jörg van de Loo"

#define Version 1

#define Revision 01

// ---end


/* Demo  CDN */
------------------------
#define ID "$VER: BareED-doc 1.01 (23.09.96) Copyright Jörg van de Loo"

#define Version 1

#define Revision 01

// ---end
```

1.11 Internals to BareED

NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE

THIS PAGE IS OBSOLETE ! ! ! - AND IT WILL NOT BE UPDATED ANYMORE !

```
However,
  I'd like to thank Gunther Nikl for redesigning and enhancing the start-up-code.
  Thanks, Gunther.


You  may not imagine how much effort I spent to optimise BareED - not only to make ↩
     it faster but also to make
it as short as possible...


- BareED has been written with the complete renunciation of all 3rd party link  ↩
    libraries and object files
- All  standard  functions  like strlen, strcmp, strncmp, strncat and so on have  ↩
    been re-designed in plain C so
    that any compiler can use relative addressing mode to processor register a4 ( ↩
       small data)
- A complete new start-up-code, written entirely in plain C (almost)

o gained about 11Kb of code
o gained about 2700 long relocation entries (~16 Kbytes)
o gained speed when accessing the operating system, especially Exec, GadTools and  ↩
    Intuition
o auto-detach from CLI/Shell
o no assign to PROGDIR needed (even when called from the console)
o 3rd party graphic device compliant (no default PAL/NTSC screen setting)
o coloured editor window
o proportional font support
o real tabulators
o stack check at initialising time
o Draco computer compliant
o Enforcer/Mungwall/IO_Torture tested
o extensive tests already done for OS 3.0, 3.1


Internal limits of BareED:
  Up to 2.1 milliards characters per archive
  Up to 2.1 milliards pixels per line
  Up to 2.1 milliards lines per archive
  Up to 65535 pages per archive
  No screen depth limit (although more than 8 get not really supported through the ↩
       system (yet))
  Any screen size greater than 640 pixels in width and 200 pixels in height
  Up  to 75-pixel height for the font (due to the limit of the visible area;  ↩
      internally BareED can handle fonts
  up to 32767 pixel in height)
  Tabulator  size  up  from 1 to endless (although any tabulator width greater  ↩
      than 2 milliards pixels will put
  BareED into trouble)
  Up to 16.7 million different colours per pen supported (24 bit, sorry no 48 bit  ↩
      support)
  Up to 2.1 milliards clipboard units but currently only one (lonely) block  ↩
      supported
  Only one font per text file supported
  Only one font style per text file supported


BareED consists at the moment of 3 files (system header files do not count here):
```

```
  - startup.c   19799 bytes source     3468 bytes object  -GNU-C compliant
  - BareED.c  351747 bytes source 115388 bytes object   -ditto-
  - cpylib.asm     6282 bytes source       432 bytes object


BareED designed using:
      Maxon's MaxonC++ compiler V1 and V4 in C mode
      HiSoft's Devpac Amiga assembler V3 in 68000 mode
      HiSoft's Devpac Amiga debugger V3
      Cloanto's paint program Personal Paint V7
      Martin Apel's ADis disassember V1
      Author's own graphical converter BitMapSaver V1
      Author's own hunk analyser DropHunk V1

Experimental compile runs under:
  AZTEC-C «« Not tested with BareED source codes higher than version 0.87 .
         Version 0.87 okay (rely on warnings sprinkled out all over the source  ↩
            code!)
  GNU-C   «« large data mode ok., near will cause a lot of internal compiler  ↩
     errors
          (spilled register -while setting up RawDoFmt() and several GadTools  ↩
             functions)
  VBCC    «« Not tested with BareED source codes higher than version 0.72 .
          Version 0.72 okay (3rd pass of optimisation could not be used by me due  ↩
             to lack of RAM)
```

1.12 std

```
"Copyright and Distribution"

The copyright holder is:          email:
  Joerg van de Loo          »Joergloo@aol.com«
  Hoevel 15
  47559 Kranenburg
  Germany



It  may  the  executable  file  'BareED',  other  needed  files  (for  example  ↩
   the  presettings)  and  this  documentation to be
distributed  for  free  if  nobody  demands  money  explicitly  for  that.  BareED ↩
     must  not  be  distributed  onto  floppy disks
(exception:  as part as a cover disk for Amiga journals). It is okay to me if  ↩
   BareED and its files are distributed via nets and
CD-ROMs.  The  source  code  of  BareED  is  freely  acquire-able, however, only  ↩
   through me and no one has the right to
make copies of it and spread it to 'friends'.

If  you  want to receive a copy of BareED's source code then send an already  ↩
   formatted floppy disk (either 880 KB or 1.76
MB)  and  a  self  mapped  envelope  with  5  US$  cash  to  my  postal  address.  ↩
   I will not ship the source code of BareED
through nets!
```

```
As  indicated already in this document, BareED is a beta release that is not  ↩
    faultless (and far away from being perfect). So,
if  you  take  note  of  something  exceptional  that  leads  to  a fatal crash or ↩
     mistake, which in turn has the result in loss of
dates  or something else that anybody can imagine, I refuse to take any liability. ↩
     Again, all use is at your own risk. I cannot
be  made  liable for a probable made mistake or lost dates, including profit loss, ↩
     although error reports are thankfully taken
into account!
```

1.13 Part Installation

```
To  install BareED or only parts of this archive, firstly choose the medium and  ↩
    drawer where this install script should place
the  files.  Then,  click  one  or  more buttons to let the install script know  ↩
    which parts you want to have installed. After that,
choose "Proceed with Install" to start the installation.

You  can  choose different mediums and drawers for installation; to do this  ↩
    firstly choose medium/drawer, then click one or
more "part-buttons", then click "Proceed with Install". Repeat this procedure for  ↩
    different mediums/drawers.


NOTE:
  An  »ASSIGN«  has  been  made  labelled "B_Temp:" that points to your »Ram Disk: ↩
      «. In case you want to install
  BareED  only  temporarily - without installing BareED for real - then please use ↩
       the »B_Temp:« "volume" instead
  of "Ram Disk:", this will suppress faults - thank you!


You  have  to  choose the volume (the medium) and the destination drawer  where   ↩
    you  wish  to  install  BareED  and  by
BareED  used  files!  You can choose a non-existent drawer, in this case a  ↩
    requester appears - asking you if it is okay to
you  to  create  a  new drawer; your response to the appearing requester should be ↩
     »Create«. In case your response was
not »Cancel«, the new drawer and a desktop image for it will be created!

Which parts of BareED do you like to install?
(Click the buttons that represent the things you want to have installed):

      BareED, the program itself (version 0.9623)
      BareED's program pictogram (desktop image; a 4-colour image)
      BareED's configuration file, necessary for BareED's button-bank
      BareED's default button-bank imagery (standard look)
      BareED's ARexx macro files
      BareED's alternate icon imagery (4-colour default imagery)
      Tool directory with the image converter »BitMapSaver«
      Additional font directory (»Excel«)
      Example imagery (in ILBM format - as starting-point for your own)
      Catalogue directory with German catalogue file
```

```
    Files, which can be used as starting-point to translate BareED dependent  ↩
       strings to your native language
    Enclosed source code files (in case you can read/understand/use C and Asm  ↩
       files)

    The enclosed documentation

    German dokumentation, needs the in English written!
      Create icon for the otherwise invisible drawer?

    The complete DAC archive, with source
      Create icon for the DAC drawer?
      Create icon for the otherwise invisible drawer "source"?


To start the installation, click the button below.
 Proceed with Install
```