in

COLLABORATORS

	TITLE : in		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 18, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

in

1.1 main

```
                 Argue 1.0 documentation
    (c) Thorsten "Flavour" Stocksmeier


                Readme first!
                 <- legal stuff / license


                 Introduction

                 ReadArgs()

                 Usage

                 Additions

                 Tooltypes

                 Main window

                 History

                 Future

                 Known bugs

                 Argue scripts

                 Sorry!

                 The author
                Note: I'm not a good guide writer, and I know that ;-) If you miss ←
                    something
      then it's no problem to include it in the guide.

Please remember:
      Argue is written in hours of hard work :-)
```

1.2 legal

```
The software "Argue" is copyrighted. The distribution itself may not
be modified. (This excludes archivers, of course)

All users may spread Argue as far as they can, meaning BBSs and
ftp sites and whatever they can think of.

This tool may be put on any CD if unmodified. The CD makers should
check whether they got a modified version.

Regular (i.e. official) versions of Argue are _first_ pushed to
Aminet and THEN spread around, not the other way round.

There will *never* be Argue beta versions available to the public.
If you get one, tell me where you got it immediately.
```

1.3 scripts

```
For advanced users or developers, Argue can do a very nice job when
used in shell or arexx scripts.

Argue parses the template from the TEMPLATE shell argument, so in
such a script file you might write

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

ed blablabla
argue >t:argue_out "DUMDUM/K/A,AMAZING/S"
c:fascinator `type t:argue_out` blurk

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

Well, this is a rather silly example, but perhaps it shows, what you
can do with Argue in a script file. You may even use it as a GUI
interpreter to ask the user for settings (like RequestFile or so)

It would even be possible to write a preferences editor with Argue
and store the result in env:xxx.config.

Just try to experiment :-)
```

1.4 bux

```
- Argue doesn't free all its memory. I'm hunting for those leaks,
  but far not all are yet localized. (Argue uses E's memory tracking
  to avoid dramatic leaks ;-)
```

1.5 future

```
Big parts of the new things in Argue 1.0 were suggested by its users,
and I think this is the best kind of cooperation developers can have.

I hope a lot of people have now realized they can help to make Argue
better and better. As long as I get feedback, as long as I know
folks still use Argue and are interested in further development, I
will spend my time for them and build in what you like.

(BTW: In the worst of all cases (if I sell my Amiga ;) I will release
the complete sourcecode to allow others to include the neat features
of a new OS or GUI system. Argue should not die :-)

Now my hopes are that Argue’s users get creative and think about what
they would like to have in a new Argue. Feel free to flame :-)

You see, the future of Argue is in your hands. From version to version
I get more replies, and I hope this will continue.

Tell me! Write to: flavour@aventure.teuto.de and inform yourself
about the newest Argue on http://www.teuto.de/~flavour/in_argue.html
```

1.6 history

```
                          ... to be continued ...


1.0 - complete argument parsing rewritten, interface contents can be
      easily load and saved, cycle gadget now features as many items
      as specified, shell mode disabled, hoard of new examples supplied,
      user defined bubble help, arguments may now be linked to a helpnode
      in an AmigaGuide document, template file configuration now amazingly
      user friendly ;-), shell mode enabled again, but former icon
      config no more supported, MUI layout completely reworked (should
      do a much better job with patterns now.)                [August 1996]

0.9 - ASL multiselect for the /M multiple gadget causing two new internal
      hooks :), screenmode popup works now, failed New()s and String()s
      raise an IMEM (insufficient memory) exception. (for safety), added
      the new cycle switch, output string size now 10kb., new drive list
      popup, help file support added for the example scripts, string
      gadgets advance on carriage return (MUIA_String_AdvanceOnCR)
                                                                [July 1996]

0.8 - major improvements: pre-settable switches and string gadgets,
      windowID no more "MAIN" but "ARG!", five new popup buttons
      for several arguments containing keywords. (PubScreen etc.)
      [Argue 0.8 distribution contained the 0.3 executable. Silly thing....]

0.7 - template parsing has been widely extended to allow min/max/actual
      extensions for integer arguments. fixed this and that bug. (as
      always :)                                                 [June 1996]
```

```
0.6 - new release version, added multi argument list, help bubbles,
      unix/nospaces routines rewritten, lets the user decide whether
      he likes the arguments in a register group or not         [June 1996]

0.5 - all argue development was switched to MUI, which is really
      predestinated for those dynamic gadget things ;)

0.4 - interim release of 0.3 :)))

0.3 - first release version using the nicegui system. not a very good gui
      layout system, but it did the job                        [April 1996]

0.2 - again some alpha things. they DID work, but how...

0.1 - never released alpha version (didn't work ;-)          [January 1996]
```

1.7 window

When all startup things worked, Argue will now begin to prepare
its window, work a bit on it and then open that main window.

You will see the interface elements in a virtual group. Slide
the scroller at the side down and up to reach all elements.

It is now finally possible to save the actual state of the interface,
meaning ALL the stuff you entered and clicked. Just click "Save this"
in the Project-menu. This nice feature is again a neat MUI feature
I saw when flying over the autodocs ;-)

The saved configuration is load at startup time and dominates the
predefined configuration in the template file.

You may manually reload your settings with "Load" from the menu.

1.8 introduction

It was around 1992 when Commodore released their new Amiga OS 2.0. With
this, there were amazing changes for developers and users. All looked
a bit more professional, and a lot of things were just easy and better
to handle than in former times.

Earlier, developers had to write their own argument reading system. Often
it was really unpractically and difficult to understand.

The guys at Commodore knew that and thought about a new standard for
argument parsing to avoid confusion about all that. What they finally
got was ReadArgs(), a system function that parses arguments automatically.

Developers now only had to write a template to specify, what arguments
they would like to have. A template looks like this: FILE/A,SWITCH/S...

From now on, all the users could have a look at this template by adding a question mark to the program's name to execute.

But all in all, there was a problem. Folks still had to go "down" into a shell and type in all the arguments by hand.

So there are still a lot of people that write external interfaces for a specific tool. Some of them are even shareware!

This was really annoying as there was no tool that could manage ALL tools.

In early 1996 I developed a GUI layout system called NiceGUI. It was crap, but on this way I created the first version of Argue.

Argue's job was and is to read other tool's argument templates and prepare a nice user interface where the user can decide what he would like to have as arguments. Argue 0.3 was quite bad, but it was the first basis for further development.

Some months later I invented how to write MUI applications. It was very easy, and I implemented a new version of Argue with it. This was called Argue 0.6 and released to some BBSs here in Germany.

From then on Argue made giant steps towards user friendliness and efficiency. New features were added in masses, and now, at the time of Argue 1.0, there is a (near ;) complete interface creation system.

1.9 flags

ReadArgs(), the function that eats the templates, supports ↩ several flags
that are linked to the argument name in the template. So a switch will be called switch/S.

The most important flags:

/M	a multiple gadget. Can be fed with as much arguments as given by the user.
/A	this argument MUST be given.
/S	this is a switch.
/T	rarely used. same as /S. You should write /S.
/N	a number. may be positive or negative.

If no /N or /S flags are given, the argument is meant to be a string.

Argue even offers a new flag (just for its template)

/C	offers a nice cycle gadget with as many items as you like. These items are specified by

```
                Additions
                    Please note Argue does NOT support:

/M/N    this is absolutely rarely used.

/T      write /S in the template instead :-)
```

Abbrevations may be done with the "=". So "FI=File/A" is perfectly OK.
This is important in UNIX mode!

1.10 usage

Argue prefers to read its arguments as tooltypes from a project icon with itself as the standard tool. It will also accept shell arguments, but you should no more use them.

First it looks, whether there is a TEMPLATE tooltype. This specifies the template it should build a user interface around. But even this is now out of date.

Argue likes to parse the template from a file with the same name as the icon. (So if the icon is called foo.info, the template would be in the file foo)

The syntax of such a template file is easy. Look:

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

you may add some comments at the beginning!
blah...

@NEWFASHION   <- the marker
              <- one empty line
FILE/A
SWITCH/S
INTEGER/N
   .
   .
   .  <- do not leave a blank line at the end!
-------------------------------------------------------------------
```

This is the basic template file. Please do NOT forget the @NEWFASHION marker, as it is very important.

1.11 reference

Template additions are read by ReadArgs() with this template:

```
     fl=FILEPOPUP/S,fn=FONTPOPUP/S,ps=PUBSCREENPOPUP/S,
     sm=SCREENMODEPOPUP/S,sc=SECRETPOPUP/S,dr=DRIVEPOPUP/S,
     no=NOPOPUP/S,de=DEVICEPOPUP/S,can=CANDIDATEPOPUP/K,
     min=MINIMUM/K/N,def=DEFAULT/K/N,max=MAXIMUM/K/N,
     cc=CYCLECHOICES/K/M,on=SWITCHACTIVE/S,help=BUBBLEHELP/K,
     preset=PRESETSTRING/K,node=HELPNODE/K
```

1.12 sorry

A big sorry to all my users. I was not able to keep compatibility
to former Argue versions when used with templates in a file because
there were major changes in reading the minimal and maximal values
for integers and so on.

Argue will refuse any interface that has REQUIRED set to something
less than 10.

But I promise, all version from now on WILL support your Argue 1.0
interfaces! Really! :-)

1.13 email

Argue is submitted "as is", the author is not responsible for any
damage this tool may cause.

Argue is EMailware. If you use it, you should write an EMail to
the author. I'll be VERY happy if you send me your own Argue
interfaces you made and I'll add them to the examples drawer
at once ;-)

Write to: flavour@aventure.teuto.de

And see Argue's information bulletin for newest information about
development.

Have a look at http://www.teuto.de/~flavour/in_argue.html

1.14 additions

(For quick reference, click
here
.)

Argue likes to have specifications for each argument. This offers a
great compatibility for a lot of tools.

So if I now write, "xyz" should be added, then

```
    Switch/S
```

becomes

```
    Switch/S (xyz).
```

And if I write, you should also add "blah", then

```
    Switch/S (xyz)
```

becomes

```
    Switch/S (xyz blah).
```

Please do NOT use commas in brackets, this will confuse Argue and disturb the whole interface. Not even in help strings. If you get weird problems, this may be the cause!

Argue features nice popup buttons for string gadgets. This means if you press it and select something from the list comming up, it will be taken to the string line. This is very useful!

Please note you may only add ONE popup for each argument.

You may add...

```
   "nopopup"   is the default and disables any popup buttons the
               string gadget might have.

   "filepopup"   for a file popup button. When the user presses it,
                 an ASL requester will open and he can click on a
                 file. The filename will then be added to the
                 string gadget.


   "screenmodepopup"   if a tool wants to have the name of a monitor
                       (for example "Multiscan: Productivity") then
                       this will help. A screenmode popup will open.

   "devicepopup"    nice for terminal programs etc. You may choose
                    from a list of *.device files that are available
                    on your computer. (This list is read from DEVS:
                    when Argue initializes)

   "pubscreenpopup"   do you have lots of tools being able to open
                      their window on a public screen? With this
                      popup button  you can choose one out. Note:
                      The public screen list is just read once.

   "secretpopup"    is for arguments that have something to do with
                    passwords etc. Every character will be represented
                    by a dot, not by a character, so nobody will
                    see what you enter.

   "drivepopup"   offers a nice drive list when popped up.
```

So "File/A" would get "File/A (filepopup)". Please DO leave a space between the first bracket and the argument’s name/flags.

You may also preset string gadgets to ease the use of your interface.

Now if you'd like to have File/A the preset "foo.bar", why not write

```
    File/A (preset="foo.bar")
```

Now there are /N gadgets. They specify integer values. If you want to limit, how far they may go or what the default shall be, there are again things you may add in brackets after the argument's name. (For one argument all the additions are ALL in ONE pair of brackets!)

Let's say you want to limit Int/N to 80 and the default shall be 20. Its negative limit must be -50. No problem. Int/N will then be

```
    Int/N (min=-50 default=20 max=80)
```

Switches like Blah/S may be pre-clicked if they are very useful :-)

Just add "on" to the brackets. So "Blah/S" will be

```
    Blah/S (on)
```

Do you know MUI's help bubbles? They look really nice and may be very important to give a short help to the user of your interface.

So if you want File/A to have the help text "tictac", just write

```
    File/A (help="tictac").
```

Perhaps you already read Argue supports a new cycle gadget. The elements for it are also specified as additions in brackets.

So you may write

```
    Numbers/C (cyclechoices "one" "two" "three").
```

If you specify an AmigaGuide document with the GUIDE tooltype, you may give each object a specific AmigaGuide node it belongs to.

Now if you have a Blgrmbl/S switch and you want to reference to a AmigaGuide node that explains it, why not add

```
    Blgrmbl/S (node="Switches")
```

Where "Switches" is a valid node in an AmigaGuide document.

If you didn't understand everything I wrote here, why not have a look to all the Argue examples that come with it? Nearly all features described above are in them, and you may learn a lot more than from this guide :-))

1.15 tooltypes

```
    Argue also likes tooltypes at the project icon.

---

    If you have a tool that persists on old UNIX argument style like

      +c24 +F

    there is the UNIX switch. Argue will stop using ""s to cover
    arguments and use the abbrevations of argument names for the
    output.

    So if the old program awaits a file name after +r, just write

        +r=File/S.

    If your tool even does not like spaces between the argument
    identifier (+r) and the filename, use the NOSPACES tooltype.
    This will disable argument spacing.

---

    Argue has three modes of output. If you press the "Try" button
    the tool will be executed with Argue's output as arguments,
    but Argue will not pop down. If you press "Use", the arguments
    you enter are pushed in right order and written to the
    output shell.

    If you would like Argue also to execute the command when "Use"
    is pressed, add the COMMAND tooltype and in follow the commmand
    name.

    Why not write "COMMAND=list all" :)

---

    MUI offers nice looking integer gadgets that look like knobs.
    If you like them and want Argue to use them for any integer,
    add the USEKNOBS tooltype.

---

    Argue puts a title above your interface. For default this is
    something like "Argue x.x" (where x.x is the version)

    But if you want to specify an own title (you might explain
    what a GUI this is ;) write

    TITLE=blahblah

---
```

```
    If you have a big bunch of arguments for Argue it may be wise
    to divide them into three groups and put them into a register
    group. Just add PAGEGROUP to do that.

---

    Argue usally won't add integer arguments to the template if
    they are NULL. If your tool needs even the nulls, add the
    ADDNULLS tooltype.

---

    You hate help bubbles? Specify the NOHELP tooltype :)

---

    If you have a string gadget that has a font popup, MUI will
    add the font name and the font size to it. Argue's default
    handling will cut off the size when you press "Use" or
    "Try". If your tool awaits the font size, just add the
    ADDFONTSIZE tooltype.

---

    You find this damn Argue logo a waste of space? Simply add the
    NOLOGO tooltype and it will vanish like magic ;-)

---

    The MIXTURE tooltype enables Argue's native mode. It will
    not sort arguments from the template file any more and
    display any argument as it comes.

---

    You need some help text below the interface? No problem.
    Write that text to a file and add its name after the
    HELPFILE tooltype.

---

    To make MUI remember your interface window as a unique one,
    add a nice identification string after WINDOW_ID. Argue will
    use "ARGUE [your id]" as window id.

---

    Always write Argue's version behind the REQUIRES tooltype
    that you used to create the interface.

---

    If you have an AmigaGuide document that explains some of your
    arguments, add its name after the GUIDE tooltype.

    If you then want to reference to a specific node for an
```

```
argument, add (node="mynode") behind the argument's name in
the template file.
```