Argue

Colin Thompson Stocksmeier

COLLABORATORS			
	TITLE : Argue		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Colin Thompson Stocksmeier	July 16, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Argue

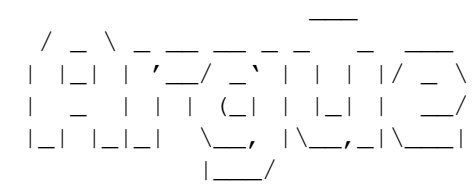
1.1 Note

```
                Hi downloader of Argue!                      16.10.1996

This distribution contains one of the largest programs
I did at all. It took over a half year to get it so
far, and countless hours of chumming the source till it
fit.

I do not request any shareware fee or other greedy
things, but if you like this tool then I'd really enjoy
a small donation, whatever you think is equivalent for
the use of Argue.

You may send all kinds of currencies as far as they come
in notes.

If you don't have money for someone like me, I perfectly
understand you - I'm a poor school boy, too :)

But if you think Argue is quite a nice stuff, and you
just thought about saving the last few living Amiga
programmers on earth, some bucks might help keeping me
at this amazing system :-)

My address:
                Thorsten Stocksmeier
                Lemgoer Strasse 19
                32657 Lemgo
                GERMANY


            To the guide...
                Some years, and all this time will be history...
```

1.2 Shell

```
Argue has a powerful shell mode, one might even use it as
a hugely expanded c:requestchoice.

There are two ways of using the shell mode:

    - enter 'ARGUE "File/A [preset=gaga],Switch/S" nologo' to a
      shell window.

    - enter 'ARGUE RUN foo.bar' where foo.bar is a template
      file like described in this guide somewhere else.

If you want more information about Argue's shell mode, leave
me an email.
```

1.3 Table of Contents



```
     Flav's notes
              (c) Thorsten "Flavour" Stocksmeier

This guide was mainly written by
  Colin Thompson. Thanks Colin!
```

Legal

1.4 flav

Here we go again...

Yet another version of Argue is out to the public. I built in
quite a lot of things, and this nice guide you're reading is
in the distribution.

Argue has been developed on an A1200 with (actually) 10MB of memory
and a 1GB hard disk. Since about one month I have my Blizzard 1230
with a 030/50.

Compilation takes about 1,5 seconds (!) and outputs:

```
Amiga E Compiler/Assembler/Linker/PP v3.2a registered (c) '91-95 Wouter
lexical analysing ...
UNREFERENCED: hook,q,q,q,x,popasl,popasl
parsing and compiling ...
no errors
intermediate code buffer used 66236 (65%) of 102400 (reallocatable).
libraries/generated code buffer used 68848 (30%) of 233027 (fixed).
general/identifier buffer used 113944 (93%) of 122880 (expandable).
label buffer used 6488 (42%) of 15360 (reallocatable).
```

Argue is now over 1/2 year old, and if you knew how Argue 0.3 looked (it
was on Aminet and *worked* ;-) you wouldn't believe how it looks today.

Maybe the best idea of all was to put the template into a file and
make Argue use project icons. I don't want to be mean, but the usual
Amiga user seems not to be able doing anything else than clicking :)

1.5 Have your attorney look this over

The archive and programs contained therein may not be modified.
Users may spread Argue as far as they can, meaning BBSs and
ftp sites and whereever else they can think of.

This program may be put on any CD, if unmodified. The CD makers
MUST check with me to see if they have the latest version. The only
exception to this stipulation is Aminet CDROM.

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.

1.6 The future of Argue is in your hands

Many of the new features in Argue 1.0 were suggested by its users.
This seems to be 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 :-)

Did someone say ARexx?

You see, the future of Argue is in your hands. Each new version
generates more replies, and I hope this will continue.

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

1.7 The inside scoop

Basically, Argue is a black magic algorithm ;-) Even I don't really know
how it works, but anyway, it DOES work quite well. (Maybe I have the right
food ;)

Argue roughly parses the arguments, looking for square brackets, passes the
contents to a private ReadArgs() call and manages all the structures
initialized for each argument.

The arguments are defined in these groups:

    G_STRING,G_INTEGER,G_CHECKMARK,G_MULTIPLE,G_CYCLE,G_BUTTON

String arguments are divided into sub gadget types:

    SG_FILES,SG_PUBSCREENS,SG_FONTS,SG_SECRET,SG_DIRS,SG_SCREENMODE,
    SG_DEVICES,SG_DRIVES,SG_PLAIN,SG_CANDIDATE

In a loop of horror each argument is then transformed to a MUI object
(in general containing a HGroup as father) and glued to some group in
the main window using dynamic MUI layout (OM_ADDMEMBER crap)

By then, Argue has little to do. It just handles some things it HAS to
do (sorting the /M list, checking for menu selections) but all the
major work is done by hooks for taking over Poplists etc.

(For quick reference, click

```
                here
                .)

    When the all startup things are done, Argue will 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 configuration is saved in ENV:MUI/Argue[progname.cfg]

    The saved configuration is loaded at startup time and overrides the
    predefined configuration in the template file.

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

     At the end, Argue starts another loop collecting all the gadget contents,
    putting them together to a string.

    And after all that, it vanishes in a puff of smoke ;-)
```

1.8 The history of the world, Part II

```
0.3 (14.5.96)

    o first working version
    o utilizes my nicegui GUI layout system

0.6 (6.6.96)

    o removed nicegui support
    o added MUI support
    o added /M multi argument list
    o created help bubbles
    o rewrote the old unix/nospaces routines
    o lets the user decide whether he likes the
      arguments in a register group or not

0.7 (20.6.96)

    o implemented a quick addition-parser to allow
      minimal, maximal and default values for integers

0.8

    o changed window ID
    o added five popup buttons for string arguments
      (fonts, pubscreens, files,...)
```

```
    o switches and string gadgets may be preset/preclicked

0.9 (17.7.96)

    o asl multiselect for the /M multiple list
    o screenmode popup finally works
    o added exceptions for failed E memory allocations
    o new /C cycle switch
    o output string size corrected to 10kb
    o new drive list popup
    o help list support for example scripts
    o string gadgets have MUIA_String_AdvanceOnCR on

1.0 (17.8.96)

    o complete argument parsing rewritten
    o new feature allows loading/saving of GUI contents
    o cycle gadgets supports more than two states
    o lots of new examples included
    o user defined bubble help added
    o added NODE addition to enable AmigaGuide help
    o additions are handled via brackets
    o shell mode repaired
    o new MUI layout to support patterns

1.1 (1.9.96)

    o added SHOWFILE tooltype
    o new subwindow and menu
    o candidate popup working
    o replaced old execution routine with a better
      working one
    o important functions got a debug text
    o removed enforcer hit
    o cycle gadgets return their correct state
      when Start is pressed

1.2 (16.9.96)

    o corrected second output buffer size
    o /M multiple lists support "cyclechoices" addition
      and PATTERN/K
    o dynamic adding of gadgets will be initialized and
      finished with the correct MUI methods
    o replaced "Go" through "Check output..."
    o removed nasty bug in ReadArgs() call

1.3 (16.10.96)

    o Now square brackets [] are used instead of
      parantheses ().
    o listviews may be preloaded with DOSGATE/GATEFILE
    o removed a lot of enforcer hits
    o added /B command buttons
    o new RUN argument that adopts an Argue project icon
      even if Argue itself's started from shell
    o optimized tooltype parser
```

- logo images are searched in the project icon's path
- implemented SINGLE addition
- "Check output..." redirected to Argue's SHOWFILE window
- /M multiple lists get an "Adopt..." button to preload on the fly
- added COMMANDLINE argument that will add a nice executor to the main window.
- all elements will have the same size in their group, this was inspired by Thore "CyberAVI" Boeckelmann :)

1.9 Introduction to Argue

Argue reads a text file and creates a MUI GUI from it. This is called a GUI front end. It controls programs that are CLI-ONLY.

You can easily see what Argue does. From the Workbench, enter the Examples directory and click on any of the icons. You'll get the idea right away.

A bit of history

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 so the user can decide what he would

like to have as arguments.

Some months later I learned 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 enmasse, and now, at the time of Argue 1.3, I have a nearly complete interface creation system.

1.10 ReadArgs flags

ReadArgs(), the DOS 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. (Actually Argue only prints the label of such arguments in bold style. This might be changed very soon, please keep an eye on where you set this!)

/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 new flags (just for its own template)

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

Additions

/B A command button. When pressed, a DOS command is executed.

(Most commands have done bad things, so they are killed ;) The command is specified with the PRESET addition. Such a button could be specified like this:

Showpic/B [preset="vt pics:foo.bar"]

Please note Argue does NOT support:

/T Toggle. Absolutely obsolete thing. Changing this to /S is far better and supported by Argue ;-)

Abbrevations may be done with the "=". So "FI=File/A" is perfectly OK.

This is important in UNIX mode!

1.11 How to use Argue

```
                Always launch Argue from a project icon. Shell usage can be ↩
                   done,
but it requires some skills. Look
            here
                Argue will parse the template from a file with the same name ↩
                   as
the icon. So if the icon is called fooGUI.info, the template would be
in the file fooGUI.

The syntax of such a template file is easy. Check out this example:

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

You may add some comments at the beginning. The filename of the
program would be appropriate.


This gui is for the program UNZIP

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

The above example is just to show you what a GUI might look like. In
reality, you would type the template and save it as a file.

 To get a real template, open a shell and type the program's name, a
 space and a question mark. Then press return.

 Ex:

 12.Work:Argue> ZipTool ?
 The template of the program will appear in the shell like this:

 ZipTool v1.2 © 1996 Oliver Hitz
 DEVICE/A,INFO/S,WLOCK/S,RWLOCK/S,UNLOCK/S,TUNLOCK/S,PASSWORD/K,EJECT/S

 This is the template of the program ZipTool. You can get create a
 text file of a template by redirecting the output to a file in RAM:
 like this:

 12.Work:Argue> ZipTool ? >RAM:ZT

 Now press RETURN or enter blblbl or similar to make ReadArgs() fail
```

and the program exit.

Now just load the file RAM:ZT into your text editor and edit it so it looks like this:

```
ZipTool v1.2 © 1996 Oliver Hitz

@NEWFASHION

DEVICE/A
INFO/S
WLOCK/S
RWLOCK/S
UNLOCK/S
TUNLOCK/S
PASSWORD/K
EJECT/S
```

Save the file as ZipToolGUI. Now You need a project icon for the file. Duplicate one of the icons you find in the "Examples" directory. Name it ZipToolGUI.info.

Now you have the basis of a good Argue Script to control the ZipTool program. You can use these instructions for most other programs - just change the filename to the program you want to use.

Continue reading these docs to learn how to add tooltypes to the icon, and options to the GUI script.

1.12 Argue's own template V1.3

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/S ,
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 ,
PAT=PATTERN/K , GATE=DOSGATE/K , GFILE=GATEFILE/K,
MULTI=MULTISELECT/S , PREFIX=PATH/K
```

This might confuse you at first, but it's handy as a reference ;-)

1.13 Compatibility issues with earlier versions

Beginning with Argue 1.3, additions must be surrounded by square brackets []. Your old Argue scripts that use parens () will still function. Square brackets are needed when you specify REQUIRES=13 or later versions.

This will only cause problems when you try to update an older
template file but don't change the brackets.

NOTE: When setting REQUIRES to something over 12 you *MUST* change
the type of brackets!

1.14 Send me email right now!

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 _MUST_ send EMail to
the author. I'll be VERY happy if you send me your own Argue
scripts. I'll add them to the examples drawer at once ;-)

Write to: flavour@teuto.de

The latest version of Argue, and its documentation is always
available at

http://www.teuto.de/~flavour/in_argue.html

1.15 Make the most of Argue

Additions are modifications you may add to each argument in the Argue
script.

The general usage for adding an addition is this:

ARGUMENT/... [put your additions in here]

Square brackets must surround the additions. In earlier versions of
Argue, parentheses () were used to surround additions. This convention was
changed so that some DOS functions could be called. More on this
later.

Additions can be used to do the following:

Add a popup

Add bubble help to a gadget

Context sensitive AG online help

Display a Cycle gadget

Make a Checkbox active

Setting Integer gadget presets

```
              Using the Listview with FILE/M

              The Candidate addition
                  You may use several additions at once. This is allowed:

 FILE/A [filepopup preset="SYS:S/SPAT" node="Shell aliases" help="Pattern  ↩
    matching"]


Please do NOT use commas inside the 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!
```

1.16 Add a popup to your file requester

```
                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
       at Argue's startup time.

   "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. NOTE: This will only give you
       safety as long as you type into the Argue GUI.
       Argue passes the text uncrypted!

   "drivepopup"

       offers a nice drive list when popped up.

   "candidatepopup"

       is something very new :-) It displays multiple items
       and adds them to a MUI poplist. So you may enter
       the strings you prefer for a specific gadget.
       For further help have a look at the
             Candidates
                       interface in the examples drawer.

 Sample Usage:

 FILE/A [filepopup]
 SCREENMODE/K [screenmodepopup preset="NTSC:Hi Res Laced"]
```

1.17 Preset the numbers in an Integer gadget

```
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]

 Integers are displayed as Slider gadgets unless you use the USEKNOBS
 tooltype.
```

1.18 Preset a Checkbox

```
Switches like FILTER/S may be pre-clicked.

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

    FILTER/S [on]
```

1.19 Add a help bubble to a gadget

MUI's help bubbles look really nice and may be used to
give a short helptext to the user of your interface.

If you want HAM8/S to have the help text "Millions of Colors", just write

    HAM8/S [help="Millions of colors"].

If your helptext is too long to fit on one line, you can insert a
newline character. Argue uses the carat (^) or the shifted 6 to force a
new line. Put the carat where you want the line to break.

1.20 Add a Cycle gadget to Argue

If the program you are calling has several Switches that are mutually
exclusive, you may combine any or all of them into a single Cycle gadget.

As an example, a picture viewer might have switches to display the
picture in either grayscale or HAM8 modes. If you did not use a Cycle
gadget, these two switches would be presented as two seperate checkboxes.
You would then have to select one or the other with a checkmark.

If you put these switches on a Cycle gadget, the appearance of the GUI
is better, and the user cannot accidently select both of them.

Here's how to do this:

Using the example outlined above, create a new argument - one
that is not already used by the target program. Add the /C
qualifier.

MODE/C

Now, inside brackets, put the two switches you want to present in the
Cycle gadget. The first one you enter will be presented on "top".

MODE/C ["HAM8/S" "GRAYSCALE/S"]

The /C tells Argue that this argument is not to be sent to the
target program. What is sent is either of the two items inside quotes.

1.21 Add a specific AmigaGuide node to your GUIDE tooltype

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

For example, if the gadget is described in the node called
"Resolution", add this:

DEPTH/N [node="Resolution"]

Now, when the user has the mouse pointer over the DEPTH gadget, and presses the Help key, the AmigaGuide document will load and display the "Resolution" node immediately.

1.22 How to use the Listview gadget

Whenever you specify FILE/M, meaning multiple filenames, Argue will put a listview in the GUI. Three new buttons will be added automatically. The buttons are SORT, DELETE, and REFRESH. They operate on the Listview gadget only.

Listviews are powerful gadgets. They can be made to present the user with many options. The options can be:

    Filenames
    Data Items contained in a textfile

Loading a listview with selected filenames or data items is a pretty straightforward operation. Let's start with filenames.

For our example, let's write a GUI that plays QuickTime movies with the program QT11. QT11 can play movies that have a .mov and a .qt suffix. We will want to load our listview with files that have those suffixes. Assuming we store our movies in a single directory called DH1:qt/movies, we can use the new (to 1.3) addition called DOSGATE.

By invoking the DOSGATE addition, we can use the DOS LIST command to scan the directory QT, looking for .mov and .qt files. Here's what it looks like (all on one line):

```
FILE/M [dosgate="list >t:movies dh1:qt quick nohead sort=name
PAT (#?.mov|#?.qt)" gatefile="t:movies" multi prefix="dh1:qt/"]
```

USAGE:

```
DOSGATE "{DOS list command with qualifiers} PAT {pattern to match}"
GATEFILE {file to write the results}
MULTI
PREFIX {full path to the files}
```

Here's how Argue handles this line:

Following the DOSGATE addition, everything inside the quotes is considered a DOS command. In this instance, LIST is used. Argue will pass this string to DOS to be expanded.

GATEFILE is the file that is created by LIST. It contains a list of filenames that match the PAT specified. Argue puts this list in the listview.

PREFIX is the full path to each of the files. This path is not shown in the listview, but when a selection is made, the path is prepended to the filename.

Please notice the use of the DOS SORT=NAME command. This sorts the list by filename automatically. If you don't need a sorted list, omit this call or sort the list with the SORT button.

MULTI allows multiple selection of filenames. The normal mode is to pass multiply selected filenames all at once. Like this:

FileOne FileTwo FileThree

If the program cannot accept filenames in this manner, add the tooltype SINGLE.

Then when you select several files, Argue calls your program with the first filename selected, and waits until the program finishes. Then it passes the next filename, etc.

Using our example of playing QT movies, you could multiselect several movies to watch, and QT11 will show each movie in order.

The above example works very well when the contents of a directory change. New additions or deletions are reflected in the listview immediately. If your application uses a fixed set of filenames or data items, you can write a file that contains these items and have GATEFiLE display that file in the listview. If necesary, you can use the PREFIX addition to tell ARGUE where the files are located. This speeds up the display of the GUI.

ADDITIONAL NOTES:

The DOSGATE addition will accept many DOS commands. Feel free to experiment.

1.23 ToolTypes

These are the tooltypes you can use in the project icon.

Tooltype	Function
ADDFONTSIZE	String gadgets with fontpopup active won't get the font size cut off when put to the argument string.
ADDNULLS	Even add an integer to the output if it is NULL.
COMMAND	The command to execute with the resulting output line.
COMMANDLINE	Argue will add a nice "Execute command..."-like line to the interface so the user can easily do some additional things fromout the GUI.
DEBUG	Use this to send me a bug report.
GUIDE	A guide file used in conjunction with the "node" keyword

```
            in the template file after the argument name.

HELPFILE    This text file is shown below the interface for
            pure educational use.

LOGOFILE    Requires the path/filename of an alternative logo graphic.
            Logos are 100x49 pixels, and are read by datatypes.
            Make your own logos to customize your GUI!

MIXTURE     Do not sort arguments but take them as they come.

NOHELP      Deactivate user-defined help bubbles.

NOLOGO      Do not show the Argue logo.

NOSPACES    If in unix mode and NOSPACES is on, the output line will
            e.g. be "-e4 -xfoo.bar" instead of "-e 4 -x foo.bar"

NOVIRTUAL   Avoid virtual groups in an interface. If you're wise, you
            won't use this for big interfaces. It's meant for small
            GUIs where virtual groups just bump window size.

OUTPUT      The output window. the default should do...

PAGEGROUP   Put the arguments to three registers. (just for large GUIs)

REQUIRES    The version of Argue the interface was created under.
            DO ALWAYS SET THIS TO THE CORRECT AMOUNT! This will ensure
            future compatibility.

SINGLE      Used only with MULTI. Causes multiply selected items
            to be sent to the target program, one at a time.

SHOWFILE    Argue 1.1 is able to display the command's output in a
            neat MUI window. You could set COMMAND="list >t:foo.bar"
            and SHOWFILE="t:foo.bar".

TITLE       Specify the title of your interface. Defaults to Argue's
            title.

UNIX        Set to unix mode. (for programs that have a unix argument
            parser)

USEKNOBS    Replace those annoying integer sliders (/N gadgets) with
            the "new" amazing MUI knobs.

WINDOW_ID   MUI needs an ID to save preferences etc. DO ALWAYS SET THIS
            TO THE CORRECT AMOUNT! Else the user can't save the interface
            contents etc.

PERMANENT   obsolete. do not use any more!

SHUTDOWNCOMMAND used for deletion of temporary files etc. This command
            is executed every time you shut down Argue. You could use
            SHUTDOWNCOMMAND=delete t:foo.bar if you used foo.bar as a
            temporary file.
```

```
SLIM        obsolete. do not use any more!

TEMPLATE    nearly obsolete. if you have a short template, add it here.
            But you should use a template file instead.
```

1.24 Argue's unix mode

Did you ever have a tool that has a unix-style argument parser? Those plus and minus things? +r -k30 ? Terrible thing. You'd always have a look at the docs to find out what a switch means.

From its first version, Argue had a unix mode.

Let's think of this case: A program wants to have the unix argument +f as a switch for... frequency... something like that ;)

At first, create a new Argue project icon and specify the UNIX tooltype.

Now use this template fragment: +f=Frequency/N

This will make Argue output something like +f 14400. But the tool refuses this. It doesn't like spaces between an arguments name and the value anyway.

No problem. Add the NOSPACES tooltype.

Argue will output +f14400, and this is exactly what you wanted.

Don't stick on integer arguments, this also works for strings, buttons etc.

Please note: UNIX mode will hold Argue away from using ""s to cover filenames. This works in 99% of all cases.

1.25 Argue's built-in buttons

Argue does not allow you to program any buttons. The buttons in your GUI will alway be at the bottom of the GUI. Here are the buttons and what they do:

Start & Quit

> This starts the action, and then Argue shuts down the GUI. It's like a "one time shot".

Start

> Starts an action, but leaves the GUI on the screen for further operations.

Quit

Shuts down the GUI immediately

When a listview is displayed, four more buttons are available. These buttons pertain to the listview only.

Delete

This button removes the active entry in the listview - ONLY.
It does not delete the file!

Sort

Sorts the entries.

Refresh

Reads the directory or datafile again. This is very useful if the operation you are doing changes the directory entries.

Adopt...

You may preload a listview on-the-fly with a text file by clicking this button. For example, select your s:user-startup.
NOTE: The file you select will not be touched anyway, Argue just reads its contents.

1.26 Argue's Menus serve up a feast of options

Project

About A?

A huge self-congradulatory splash, designed to stroke my ego. Humor me - look at it once :-)

[I hate you, Colin ;-) Flavour.]

About MUI

An ode to Stefan Stuntz.

Check output AC

This is a debugging tool you can use when constructing GUIs. It sends the output of the GUI to a window for you to inspect.

Quit AQ

Duh:(Select it and see what it does.

Settings

Last Saved AL

Load and use the settings you saved last.

Use Settings AU

Load and use the settings defined in the Argue script.

Save Settings AS

Save the current settings as the default.

1.27 The Candidate addition

Candidate is a popup listview gadget you may use. It creates a text box with the title you give it. Here is the usage:

```
Media   [can cc "Phone" "Computer" "Sound card :D" "CD-ROM"]
```

When the start button is pressed, selections will be sent to the program, but the titles will not be sent. This can be used when you are using a DOS Script as a target "Program". The selections can be taken in with the .key command as parameters.