aFilter_en

COLLABORATORS			
	TITLE : aFilter_en		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 15, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

aFilter_en

1.1 Amiga Filterium 1.0 documentation

```
        PseudoDOS Group presents:

        Amiga Filterium 1.0, 3/2001, FREEWARE
        luxus extendable filetypes processor/ripper

        by 'd7' Andreas 'da Silva' G. Szabo (Andy Silva)
```

```
                 Short description, legality and requirements:
  ---------------------------------------------

  An universal file and memory ripper with interactive
  or automatical file processing, works together with
  standard interfaces like WB, Shell, Opus, AREXX, AppIcon.
  Nearly everything can be configured, for example you
  can specify typespecific directorys and output filename
  extensions for saving. XFD/XAD/XVS can be involved.
  Functions can be enhanced via plugin modules, called
  'filters'. Custom processing methods can be defined
  via macros. See the customisation part of this text
  to learn how.
  This program is FREEWARE and for private usage only.
  it may be spread as well the package is left as is.
  This program is NOT able to do anything illegal,
  any usage is at own risk.

  Every classic Amiga-OS and 680xx cpu will do.
  Runs perfectly with UAE/WinUAE emulation.
  Recommended is 3.x with librarys ASL and XFD.


                General Informations
```

1.2 Amiga Filterium - Updates and Authors Contacts

```
                Filterium Homepage ...... http://www.psi5.com/~silva/afilter/index ↩
                   .html
AmiNet ................................. http://www.aminet.net/~aminet/
Amiga Exotica ................................. http://exotica.fix.no

Andy Silva ....... author/coder of Amiga Filterium ..... silva@psi5.com

Richard 'Zeg' Wagenführer (demoarchive) ..... express@munich.netsurf.de
Crown / Cryptoburners (betausing) ............. khaarla@cc.helsinki.fi
James 'Insane Warrior' Ostrowick & Ray Heasman / Genisys .. (some sound
rippers) ..................... ostrowickj@Newton.parktown.gp.school.za
Kaare Johansen (some sound rippers) .................. kaarej@start.no
Conan / Rebels (some sound rippers) ................................ ?
Richard 'Shred' Körber / TriTech (some snd rip) .... shred@shredzone.de
Laurent Clevy (soundformat desc archive) ...... lclevy@club-internet.fr
Don Adan / Wanted Team (some sound rippers) ............ wt@jho.pol.pl
Dirk Stöcker (XFD/XAD) ................ stoecker@Rsc1.urz.tu-dresden.de
Georg Hörmann / Alex van Niel (XVS) .................... pinkie@dds.nl


                 Credits

                 Contents
```

1.3 Amiga Filterium - General Informations

```
                  Amiga Filterium general informations
  ------------------------------------

This is a new standard tool for classic amiga systems to enhance
standard interfaces like shell, arexx, opus and appicon with common
filetypes recognition capability. A typical usage example is to
select a non-standard packed file (old intros, demos, games, etc)
of unknown type and sort music, text and gfx from inside out for
PRIVATE collecting. Things like XFD/XAD/XVS can be involved even
```

recursively and output can be saved or sent to other programs like
multiview or editors.
 Any processing steps can be configured via easy builtin scripting
language. The power of amiga filterium can even be extended by
external plugin modules to recognize, convert, decruch, modify
(or what ever) a type of data or something, that is somewhere IN a
file. This tool is 100% compatible to all classic amiga-os versions
(1.2 to 3.x) and 100% written in 680xx assembler. As you desire you
can configure to use ASL, REQTOOLS or other filerequesters.

Every classic Amiga-OS and 680xx cpu will do. Runs perfectly with
UAE/WinUAE emulation. Recommended is 3.x with librarys ASL and XFD.

Amiga Filterium is FREEWARE. You may use and spread it as you want,
but nothing may be modified when offered to download somewhere else. If
offered somewhere else, it must be offered for free.

Please SUPPORT this with suggestions and ideas (and bugreports).
If you want to write a PlugIn yourself, please read this.

Credits

Contents

1.4 Amiga Filterium - Credits

CREDITS

Written by ’d7’ Andreas ’da Silva’ G. Szabo (Andy Silva), motivated by
’Zeg’ Richard Wagenführer (one of the greatest collectors of amiga stuff)
and my amiga programing syndrome disease, that forces me to do things
like this.
 Some first technicaly and moral support by ’BuZz’ Jools Smyth
from Amiga Exotica.
 For the xfd part I got some detailed help from Dirk Stöcker
(author of xfd).
 Further greetings go to Gryzor (Prowizard) for his source and
Delirium (DeliTracker).

Special greets to Richard Wagenführer and Crown of Cryptoburners for
beeing the first users of Amiga Filterium.
 Big reagrds to Crown for detecting the Happy New Year
Virus in my system. ;)
 Many thanks to Richard for serious beta testing and USING.
Big Hello to Wolfgang ’Lighttower’ Popp, thank you for source of Pici
which I used for the MemView PlugIn and thank you for UnFlash.

Further credits concerning espesially the UNIRIP PlugIn go to the
following people for described stuff:

```
   **  Richard Körber  **  - (shred) SoundCracker sourcecode
    **  Conan/Rebels  **   - Megaripper sourcecode
   **  Kaare Johansen  **  - Serial Ripper sourcecode
   **  Laurent Clevy  **   - type description archive, calc
  **  James Ostrowick  **  - Jack The Ripper sourcecode
      **  Don Adan  **     - some rip and lenght calc source
```

You can find all these older separate programs and maybe their sources
in Aminet or via the Amiga Exotica homepage!

Authors Contacts & Updates

Well, so far. It’s up to you to tell me where I can improve this program, its documentation or the entire concept. ;)

Another big hello and greetings to all guys that made this piece
of thingy possible to exist. Thanx to my fingers and hands that
seem like they hardly can’t take any further typing...

Greetings to my american sweet Cherry chick in Michigan!!!

Take care and have fun!

d7

Contents

1.5 Amiga Filterium - Installation

INSTALLATION

After unpacking evrything to a directory of your choice,
there should be at least these files:

```
afilter     the main command file
afilter.cfg   configuration and scripts textfile
filters/    subdirectoy for plugins
        ..    some plugins
```

The other files are not needed but for comfort.

If you got a new Filterium release archive and decrunched it
to access it from a Workbech Drawer you do good!
Now Amiga Filterium allready should work from WB by
selecting its icons alone or together with other file(s)
and/or disk(s)/drawer(s) (hold SHIFT key down for multiselect).

In other cases (and expert users):

If the filters/ subdirectory is missing, the PlugIns may be
in the same directory as afilter as well.

```
You also MAYBE have to edit afilter.cfg to set some paths
and maybe put the 'afilter' program to c:, the 'afilter.cfg'
file to s: and the filters/ directory with contents to
libs: (maybe create that directoy there).


               Configuration
```

1.6 Amiga Filterium - Configuration

```
                CONFIGURATION
-------------

Amiga Filterium internally uses the scripting language
'SCOPE' for all configuration and macro stuff.

Use a texteditor like CygnusED to edit 'afilter.cfg'.

A backup of that file is added as afilter/presets/default.cfg
to the original archive.

               configuration keywords

               SCOPE language syntax
                The default configuration file is expected to be:

  <WB-drawer of aFilter>/afilter.cfg
or  <current dir>/afilter.cfg
or  s:afilter.cfg
or  env:afilter.cfg
or  envarc:afilter.cfg

You can also specify a custom path and name with the
argument CFG=filename from shell or icon-tooltypes.


               Usage
```

1.7 Amiga Filterium - Usage

```
                USAGE
-----

There are many ways to use Amiga Filterium. The most easy
method is from WB icons, advanced use is from shell and
expert use is from Opus, AREXX, AppIcon and similar.


               Usage from Workbech
               Usage from Shell / CLI

               Usage with Opus / Magellan / ...

               Other Usage
                To change or enhance the whole behaviour of Filterium
see
               Customisation
              .

Also recommended:
               quick reference
```

1.8 Amiga Filterium - Usage from Workbench

```
                  Usage from Workbench
  --------------------

  You can click icons in the 'afilter' WB drawer alone
  or (hold SHIFT key down) together with other files
  or disk(s)/dir(s) icon(s).

  Maybe a
              filerequester
               would pop up if you selected
  no other icons or (a) disk/directory icon(s).
  The requester would open in/for each disk/dir icon
  you selected (neat if file itself has no icon).


  See
              Customisation
               for how to make your special
  own Filterium icons for sepcial purposes.

  Expert note: the supplied icons, except the
               'afilter' icon, are *project* icons.
               In their tooltypes they both may contain
         all the keywords described at

               Shell & Icon keywords
              .
```

1.9 Amiga Filterium - Filerequesters

```
                 Filerequesters
--------------

Filterium supports more types of filerequesters. To set
```

```
your prefered one, edit the configuration file
afilter.cfg at

FILEOPTIONS ...
{
  ...
  REQUESTER ASL
  ...
}

Change the 'ASL' to what you want, see
               FILEOPTIONS
              .

For selecting multiple files (if supported) hold down
the SHIFT key while clicking file entrys.
```

1.10 Amiga Filterium - Usage from Shell / CLI

```
                Usage from Shell / CLI
  ----------------------

    Enter in the command line:

  afilter         try it

    Standard default processing will be done
    if the DEFAULTCALL keyword in the afilter.cfg
    provides a name of a macro to use (macros also
    defined in afilter.cfg).


  afilter ?       try it

    Read some internal quick version, usage and
    about texts.


  afilter filename filename ...

    Other standard usage. See keyword list for
    detailed description.


  afilter ...

    '...' means one or more of the keywords with
    parameters from below.



               keyword list

               combined examples
```

```
                Keywords can be typed in upper or lower case or mixed
case (all the same).

The equal sing '=' and the " " are not neccessary.
You only need to use the " " if the parameter
contains spaces (like some filenames).

Some parameters here are shown like <parameter> with
these < > signs. That just means it is a name of
a file or something from the config. Allways type
the parameter without these < > signs!

Keywords below are typed UPPERCASE here, parameters
you can add to them are typed lowercase here for
better know what I mean (*S*).

Each keyword is described separately here, but you
may also combine them in one command line. Sometimes
special keywords cause to ignore others. Guess...

Expert: These keywords work also as icon tooltypes!



             keyword list
```

1.11 command line usage examples

```
              Some combined command line examples:

afilter dh0:new

  Per standard this will open a
            filerequester
             to
  choose one or more files to process in dh0:.

afilter dh0: ram:

  Do processing for dh0: and also for ram: files.
  Per standard then a
            filerequester
             will pop up
  for these both directorys, first for dh0: and
  then for ram:. Once you slected all files to
  process, they will be done.

afilter call memview.view dh0:exoticmusic/ n

  This calls the memview plugin view which is
  for viewing datas as a bitplane and hex/asc
  dump together. With a directory for the
  filerequester to start. The 'n' is the NOUSER
  keyword so that you don't need to press
```

```
      <enter> before entering the non-os mode of
      the meview plugin function.
      Please note that memview does not work with
      some gfx-cards (nothing will happen).

    afilter call uni TO ram:

      Use a supplied maco named 'uni' for a file
      or files from somewhere and write output files
      to the ramdisk. This macro accesses
      the unirip plugin to rip music out of these
      files and do nothing else.

    afilter call xfdr

      This uses the not supplied macro 'xfdr' which
      is for decrunching packed files that are
      in packed files and so on with the xfd
      plugin.

      See
                Customisation
                 to learn about macros.

    afilter CALL memview.view SOURCE selectchipmem

      View the chipmemory as bitplane/hex/asc.

    afilter CALL mvc

      Use a macro called 'mvc' (not supplied, you
      have to write it by yourself).
      It should accesses meview.view on the
      chipmemory with the ability to mark a range
      and save it to a file.



                 keyword list
```

1.12 Amiga Filterium Shell & Icon Keywords

```
                    Possible Shell & Icon keywords with parameters are:
    ---------------------------------------------------


    FILE="mydisk:mypath/myfile" FILE= ...

    "mydisk:mypath/myfile" ...

      Both the same, you do *not* need to use the
      FILE keyword. Just the file(s) or disk(s)
      or dir(s) name(s) alone work well.
```

```
  You can specify up to ??? multiple filenames
  or just disk(s): or drectory(s)/ as you
  want.

  They will all be processed like specified
  by other keywords, config or by default.

  If you just use (a) path(s) without filename,
  a
            filerequester
             should pop up for ya to
  select the file you want there! This is really
  nice if you don't remember the filename but
  just the path where it is.


ALL

  Do all files in a directory, but filenames
  matching the pattern from FILEOPTIONS/EXCLUDE
  (in the configuration file) will be skipped!


TO  path

  Specify a savepath that overrides the one
  from the config.


SLIM

  Less detailed text output (try and see).


?         try it

  See some short version and usage inforamtions.


ABOUT         try it

  See some internal about text.


CFG="mypath:mydir/mycfgtextfile"

  Use specified configuration file instead of
  the default one. This keyword is also nice
  to specify where to find the cfg file if
  path problems occur (maybe from Opus).


NOUSER
n

  These are the same. Without this keyword
  Filterium sometimes asks you questions for
  what to do or for names and else.
  * With * this keyword *no* questions appear and
  Filterium does automated processing!
  Use this for 'longrange'-work, for bbs-batches
  or if just no user is available to wait
  while processing and watch out for answering
  questions. See also NOWINDOW.


R
RESTART

  Sometimes you might want to process many files
  from many different directoys without knowing
  the paths and names. You just want to start
  Filterium again and again. Use this keyword.


FILTERS
PLUGS         try it
PLUGINS
LISTFILTERS

  Again all the same. This gives you a list
  overview of what plugins are installed,
  how they are named correctly, what functions
  are in them and the name of these functions.

  This is **VERY** important for you if you
  want to write own processing
            macros
            !
  Rather see
            Customisation
            .


FILTERS pluginname

  Same as above (PLUGS, ect also works) but just
  for one singe plugin where you allready know
  the name of. Just for a better overview for ya!


ABOUT pluginname      try it

  Read some about text on the specified plugin.

TYPES pluginname

  See a list of filetypes supported by a plugin.
  If you don't specify a pluginname, you will see
  a very looong list of the filetypes
  supported by *all* plugins (try that)!


MACROS
```

```
  See a list of macros defined in the given
  configuration file. Does not include 'micros'.


CALL macroname

  Filterium does all its processing by following
  macros. In the afilter.cfg configfile a macro is
  set for default. If you want to use an other,
  specify it here with this keyword.

  See your afilter.cfg for available macros to
  call. See also:
            Macros
                WINDOW "x/y/width/height"

  If no shell output window is allready there
  or provided from other app, these parameters
  will override the internal default settings
  or even those from the coniguration file.


USEWINDOW "x/y/width/height"

  No matter if an output  window is allready
  there, this specified window will be opened
  allways! Also overrides the configuration.


NOWINDOW

  Do not open a text output window if none there
  allready. In most cases then also no user
  Questions could be answered an will be treaten
  with default answers. Also see NOUSER.
  Good in combinantion with Opus or icons, but
  be careful because you maybe won't see
  anything what happens, even not know if the
  program really has finished!


SOURCE pluginname.filtername

  This is for advanced usage only. With this you
  can specify exactly which data providing
  filter should be used before processing that
  data with the default or specified macro.

  Main examples (parameters exactly have to be
           spelled as shown):

  afilter SOURCE loadfile     (default)
  afilter SOURCE selectchipmem    (chipmem)

  Either processing goes on (a) file(s) as it is done
  by default, or unto whole your Amigas chipmemory.
```

```
    Both cases use internal filters. External filters
    (functions) from plugins need to be specified
    by the format  pluginname.filtername  with the
    separating '.' sign.


  SYNTAX

    This does only syntax checking on the specified
    or default configfile afilter.cfg. No further
    processing is done, other keywords will be ignored.




  Note:

  Keywords can be typed in upper or lower case or mixed
  case (all the same).

  The equal sing '=' and the " " are not neccessary.
  You only need to use the " " if the parameter
  contains spaces (like some filenames).

  Some parameters here are shown like <parameter> with
  these < > signs. That just means it is a name of
  a file or something from the config. Allways type
  the parameter without these < > signs!

  Above keywords are typed UPPERCASE here, parameters
  you can add to them are typed lowercase here for
  better know what I mean (*S*).

  Each keyword is described separately here, but you
  may also combine them in one command line. Sometimes
  special keywords cause to ignore others. Guess...
```

1.13 Amiga Filterium - usage from Opus etc

```
                Directory Opus / Magellan / some Dock-Apps
 ------------------------------------------

 To use Amiga Filterium with Opus, the following line should be
 considered as a template for Opus configurations:

 afilter CALL <macroname> CFG <pathandnameofafilter.cfg> TO {dr} {F}

 Where the {F} is for opus where to put all the selected
 filenames. It does not matter to afilter whether you put the
 filenames first or last but for opus it is better to put them
 behind all other arguments beacuse opus might split them into
 more executions if there are many files selected (and the part
 before the {F} will be at first in every execution then).
```

```
The < > marked parameters are names of your purpose and choice,
type them without the < > signs, but maybe you have to put them
in " " if they contain spaces.

With 'TO' you specify a savepath for Filterium, overriding the
one from the configuration file.

You can set up multiple buttons in Opus / Dock-App for
different things to perform.

If you decided to store the Filterium configfile afilter.cfg
in s:, env: or envarc:, you don't need to set the CFG
keyword with path and name of afilter.cfg here. That keyword
is especially thought for the purpose of having multiple
Filterium configurations for different settings to use.

If you want to improve Opus filetype recognition with Filterium,
see
              filetype specific actions and settings
              .
```

1.14 Amiga Filterium - other usage

```
                AppIcon and similar programs
----------------------------

Depends on how that program(s) work. Note that you can use all
keywords described at
              Shell / CLI usage
               also as
icon tooltypes.


Batch / AREXX
-------------

Same as from Shell / CLI. All keywords should work together
with batch variables also (depends on used CON handler and
AREXX and AmigaOS version).
```

1.15 Amiga Filterium - Customisation

```
                Amiga Filterium is highly configurable!

First steps:
               custom config general info

               Customisation with Icons
                Change the tooltypes of the supplied macro-icons or make your
own icons for your purposes (maybe to call your selfmade macros).
```

custom file processing modes with macros
Learn how to change and write macros in the configuration file.

filetype specific actions & settings
Filterium is able to perform specified actions whenever a file or data in a file of a specified type is detected.
You can use this for easy 'allways do the same when ...' or even for better filetype recognition in Opus / Magellan.

further expert notes

You can decide to put 'afilter' to c:, 'afilter.cfg' to s: and the 'filters' directory to libs:, then you have to specify the correct plugins directory path in 'afilter.cfg'.
See also
Configuration
.
Then for wb usage you maybe also have to change the default tool paths in all the project icons (see below). You can also use an assign for where what is and so on (just an idea).

You can keep multiple configuration files and select which one to use with the CFG keyword from command line or icon, see
commandline/icon keywords list
.

Multiple cfg files maybe also make sense, if you want to work from a directory with files of one type you can keep a special cfg file optimized for that type there. That dir should be the current when aFilter is called from shell or batch there.

1.16 Amiga Filterium - customisations of Icons

Filterium Icons and Tooltypes

If you got this whole archive (hope so), there are allready some icons supplied for different purposes. If not, you can create your own. The main program 'afilter' needs an icon of the type *tool* and the other icons for custom parameters need to be of the type *project*.

For a new purpose copy the 'empty' icon out of the drawer afilter/presets/ to where you need it, change its name like you want and then edit its tooltypes.

Icon tooltypes are like command line keywords, but they are

```
saved in the icon file. To edit them from Workbench select
an icon and choose WB menu -> Icons -> Information.

The *Default tool* of a Filterium *project* icon has to be
set to 'afilter' (with path if afilter is not in the same
dir as the icon, maybe you decided to keep afilter in the
c: directory).

At the *tool types* list you can add all keywords with
parameters as described at
              Shell & Icon Keywords
              .
```

1.17 Amiga Filterium - Settings in the Configfile

```
                Configuration file
------------------

The Configuration file per default is  >>>>  afilter.cfg  <<<<

  ...maybe see
              Configuration
               before

The config will be parsed as a scriptfile everytime you start Filterium.
The file is built from keywords that can be grouped together
to sections.

I used my selfdeveloped configuartion script language 'SCOPE',

Scripted Configuration for Object Programing Environments

- description of all Filterium
               configuration keywords
                - learn about the
               SCOPE language syntax
```

1.18 SCOPE language syntax

```
SCOPE language syntax
---------------------

- its absolutely easy

- start simple line comments with a semicolon ;
- multiple line comments between special marks
  as all this info header
- everything is caseUNsensitive

  keyword
```

```
    or  KEYWORD
    or  kEYwOrD

  - the common form of this script is, that parameters are
    assigned to keywords

    example:  AGE = 25

    then, there are many more features based on this

  - if a parameter contains spaces it _must_
    be put in bordering marks

      aparameter
    but   "a parameter"

      11.10.2000
    but   "11. 10. 2000"

    if you like that you can put it in marks allways

      "ilikeparametersinmarks"

  - tabulators work as space(s) also and you can combine both

    keyword parameter
    or  keyword       parameter
    or    keyword   parameter

    it is _not_ relevant how many tabs or spaces you insert
    before, between or after _anything_,
    the only rule of design is that you can understand your work
    after a month or a year, too

  - the equalsign = may be used between keywords and parameters
    but its not neccessary

      keyword parameter
    or  keyword=parameter
    or  keyword   parameter
    or  keyword = parameter

  - subsections of subkeywords to a keyword and subsub and so on
    must look like this:

    keyword ...
    {
      subkeyword ...
      {
        subsub ...
      }
    }

    or  keyword=parameter{subkeyword{subsub...}}

  - or you can type _any_fancy_combination_ of all above
    just as you would like
```

```
- note: - sometimes a parameter is _needed_, sometimes its
          _optional_, sometimes _none_ should be there
  - sometimes subkeywords are allowed or not
  - refer to the docs and other examples to get an idea
    or in last chance write me and tell me what is not
    understandable to you

- script errors will be reported by reason, line and content,
  but the program may continue in most cases (!)
  if fewest neccessary informations are in tact

- if you think 'something wonderful happened' and can not find
  the bug in the script with help of the error reports,
  send me a copy of your fancy work and describe me
  what you wanted to do ... ill try to help you and to improve
  the concept, the parser and the docs :-)
```

1.19 Amiga Filterium - Macros

```
                 Maybe see also
               custom config general info
                before.


Macros
------

Filterium does no processing without a macro that tells exactly what
to do and what not. This is for speed up, prevention of false
detections, aid in automated processing and flexibility in what to do.

In the configfile you can define multiple macros, that also can call
each others as sub-macros and that even recursively. The macros call
internal filtering functions or such ones from the plugins. The
filtering functions from the plugins are called * filters *.

When you run Filterium without a specified macro, the default one
will be used. In the config it is specified like this:

    DEFAULTCALL <macro>

Where <macro> is the name of a defined macro (without the < > signs).
This macro must be defined in the configfile also.

Macro standard defintion looks like this:

  MACRO <name>
  {
    ...
  }

Where <name> is the name of the defined macro (without the < > signs).

Other examples:
  MACRO mymacro
  {
    CALL plugin.filter {CALL ...}
  }

  or

  MACRO mymacro
  {
    CALL myothermacro
    CALL plugin.filter
    {
      CALL ...
      CALL ...
      ...
    }
    ...
    }
```

Where the ... means that it can be continued like guessable (hope so).

As you can see, there are both sequential commands and subcommands
possible. Sequential commands go all on the same data and subcommands
go on the data found _in_ that data. Subsub and subsubsub and so on
commands work perfectly (if you get a script stack overflow error
please report me, but I think you won't reach that).

For subcommands you need to put them in { }.

As you also can see, you can call macros from macros.

After the CALL you type the name of

- a macro

  or the name of

- a filterfunction

See:
how to get a list of filterfunctions
There are builtin filters as well, they should be called without a
leading 'plugin.'. The typical builtin filters are 'loadfile' and
'savefile'.
An other important builtin filter is 'execute'. You can use that to
involve other programs, arexx for example.

See:
filterfunctions described

1.20 Amiga Filterium - filetype specific actions

```
                   Maybe see also
             custom config general info
              before.


      filetype specific actions and settings
      --------------------------------------

      With the CLASS keyword you can set things for one single type
      of data. You can also group together more types of data by
      embedding one CLASS section in an other. Like:

    CLASS typename or dummy_group_name
      {
      ...
      AUTO  macroname or plugin.filter
      ...
      PATH  ram:
      ...
    }

  Where the AUTO keyword means, that the specified macro or
  filter should be called if data of 'typename' found.

  See also:
             how to get a list of filterfunctions
                    and:
             filterfunctions described
               The PATH keyword specifies the savepath for that type
  of data.

    Embedded example:

    CLASS trackers
    {
      AUTO  execute
      {
        ARG "delitracker %"
      }
      CLASS Protracker
      CLASS SoundFX13
      CLASS Prowizard
      {
        PATH  packed
      }
      ... ...
      PATH  modules:
      }

  As you can see, AUTO here can also define an embedded macro.

  The ARG word is used to specify the commandline for the
  builtin filter function 'execute', the % is where a temporary
  filename will be inserted.
```

```
The type names 'Protracker' and 'SoundFX13' will be given
from detection functions in the plugins.

See also:
              how to get a list of supported filetypes
                The embedded CLASS definitons for the both module formats
in the parent CLASS defintion 'trackers' causes the both
module formats to be immediately played with delitracker
when they are detected and the savepath for them both will
be "modules:" here. Except 'Prowizard', they should be saved
in "modules:packed/".

You can use the combination of CLASS and AUTO from Opus for
better filetype recognition.
```

1.21 Amiga Filterium - quick topic reference

```
                referenced topics
-----------------


               Configuration file 'SCOPE' language sytax

               Filerequester

               filetypespecific actions

               filterfunctions described

               keywords in CommandLine & Icon-Tooltypes

               keywords in Configuration File(s)

               list of plugin & filterfunction names

               list of supported filetypes

               macros for custom processing modes

               MultiView, how to involve

               Opus / Magellan and similar
```

1.22 list of plugins

```
              Plugins are stored in a subdirectory called 'filters'. Their ↩
                 filename
in most cases is also their plugin name.
```

```
You need to know the name of a plugin and also the name of a contained
filterfunction to use that from a
                macro
                 or the commandline.

To get a list of the names type to the shell (paths must be current or
configured well):

            afilter PLUGS

To get a list for only one single plugin (won't be too long then):

      afilter PLUGS <name>
    or
      afilter FILTERS <name>

    where <name> is the name of a plugin you allready know.
```

1.23 list of filetypes

```
If you want to configure special settings for a specified filetype or
want to let Filterium process actions whenever such a type of file or
data in a file is detected you need to know the name of that type.

To get a list of the supported filetype names type to the shell
(paths must be current or configured well):

            afilter TYPES

To get a list for only one single plugin (won't be too long then):

      afilter TYPES <name>
    or
      afilter FORMATS <name>

    where <name> is the name of a plugin you allready know.
```

1.24 filterfunctions

```
                Filterium is concepted on filterfunctions, called filters. There ↩
                   are
both internal and external filters. Internals are in the main program,
externals are in the plugin files.

Filters should be used from macros with the keyword CALL like

MACRO mymacro
{
  CALL  filtername
  CALL  filtername
  {
    CALL  filtername
    ...
  }
  CALL  filtername
  ...
}


  Internal filters names and description:

  loadfile  load a file, supports filerequester

  savefile  same but save one single data to file,
      supports some gimmicks with filename
      numbering to prevent overwrite,
      supports separate directorys for
      specified filetypes, see afilter.cfg

  selectchipmem can be used instead of 'loadfile' to select the
      chipmemory for alternative input

  selectkickrom same as above but you get the rom ;)

  selectrange to use within allready loaded or selected
      other buffer, specify a range

      Example:

      CALL  selectrange
      {
        CFG { OFFSET $0, LENGTH $400 }
        CALL  savefile
      }

      This would save the first $400 bytes of what
      ever (perfect for diskimage bootblock).

  execute   do a shell command,
      good for sending output to CED or multiview,
      see afilter.cfg

      standard form to use in a macro is:

      CALL  execute {ARG "pathname %"}

      Where the % is the place where the name of
      the temporary saved data should be placed.

  choice    ask the user for what to do,
      supports multiple options with default
      if no user available ;)

      standard macro form is:

      CALL  choice
      {
        ARG "'f'irst, 's'econd, 't'hird, ... "
        CFG { DEFAULT f }
        CALL  ...
        CALL  ...
        CALL  ...
        ...
      }

      Where the ' ' marked letters are the keys to
      press to activate command in the line number
      after the ARG. Default allways is nothing,
      what means Filterium just continues after the
      'CALL choice { }' section.

      possible CFG keys

      DEFAULT   specify the letter of the
          option to do by default


  findtext  checks data to be readable contents, good for
      ripping scrolltexts out of demos and intros,
      supports output formating

      usage from macros:

      CALL  findtext
      {
        CFG {UPPERCASEONLY MINLEGHT=20 NOGAP}
        CALL ...
        ...
      }

      Embedded CALL(s) will be done if text found
      and will be done _on_ that text (f.E. savefile).

      Possible CFG keys are

      MINLENGTH 40    minimal interesting length
            default is 20

      UPPERCASEONLY   find only BIG letters,
            default is mixed case

      NOGAP     no 2 byte crap-gaps allowed,
            default is with crap-gaps
            (was implemented because of
             control values in scrolltext)

  scanloop  scans through data each two bytes with all
      specified subcommands (see
                macros
                )


Some external filters names and descriptions:
  Availability depends on installed plugins.

  To call a filterfunction from a plugin you need to spell the
  name of that plugin followed by a '.' followed by the name
  of the filterfuction.

  See
                how to get a list of filterfunctions
                )


  xad.unarc Retrieve files with XAD (all modes).

      Example:

      CALL  xad.unarc
      {
        CFG { MAKEDIRS, ALLFILES }
        CALL  ...
        ...
      }

      Possible CFG keys are

      MAKEDIRS    make all directorys
            from the archive in
            the destination path

      ALLFILES    retrieve all files,
            ovveride the pattern
            from
            FILEOPTIONS/EXCLUDE


Planned future implementations:

  It's up to you! Wanna write a plugin? ->
                Author Contact
```

1.25 how to involve multiview and other applications

```
                Sometimes you might want to send data output not (only) to a file ↩
                   but
to annother application like CED or MultiView. It comes in handy for
graphics or text data.

This can be done with the builtin filterfunction...

    >>>>>>>>>>> execute <<<<<<<<<<<<

...described under
                filterfunctions
                .
```

```
Read the parts
                Configuration
                 and
                Customisation
                .

Maybe also see the file 'afilter.cfg' for working examples.
```

1.26 configuration keywords

```
                Keywords in global configuration file(s) like default  afilter.cfg ↩
                     .


                 PLUGINSDIRECTORY
                  path

                 WINDOW
                    xpos/ypos/width/height

                 USEWINDOW
                    xpos/ypos/width/height

                 NOMOUSESTOP

                 NOUSER

                 FILEOPTIONS
                    [ALLWAYSREQUESTER] { ... }

                 DEFAULTCALL
                    macroname


                 SETPLUGIN
                    pluginname { CFG ... { ... } }


                 MACRO
                      macroname { ... }

                 MICRO
                      macroname { ... }


                 CLASS
                      typename { ... }
```

1.27 PLUGINSDIRECTORY

```
  PLUGINSDIRECTORY  path


Set the path where Filterium should look for the plugins.


Example:

  PLUGINSDIRECTORY  f:filters


Default:  LIBS:filters/, current dir + /filters/, current dir
```

1.28 WINDOW

```
                  WINDOW      xpos/ypos/width/height


Specify a console window position and dimension to open if there was none
allready. For example if started from Workbench.


Example:

  WINDOW      0/0/640/256


Default   : 0/11/640/188
Commandline/Icon: this keyword used there overrides config
See also  :
                 USEWINDOW
```

1.29 USEWINDOW

```
                  USEWINDOW   xpos/ypos/width/height


This is like the WINDOW keyword, but this means, the window should be
opened and used _allways_, no matter where started from!


Example:

  USEWINDOW   0/0/640/256


Commandline/Icon: this keyword used there overrides config
See also  :
                 WINDOW
```

1.30 NOMOUSESTOP

```
  NOMOUSESTOP
```

Normally the scanning funtions stop when the left mouse button is pressed. Use this keyword to turn that off.

```
Default   : stop on left mouse button
```

1.31 NOUSER

```
  NOUSER
```

Do not ask any questions (yes/no, etc), go on automatically with the default answers and choices.

```
Default   : ask questions
Commandline/Icon: can be used there also
```

1.32 DEFAULTCALL

```
                  DEFAULTCALL   macroname
```

Specify a macro, that should be done automatically at program start if none is specified within the argument line or by icon tooltypes. Without this keyword and a parameter (needed!) Filterium yet does nothing but showing some information texts.

Example:

```
  DEFAULTCALL   mymacro
```

```
Default   : for default Filterium does well nothing
Commandline/Icon: in the argument line or icon tooltypes use
      CALL instead
See also  :
                 MACRO
```

1.33 FILEOPTIONS

```
                  FILEOPTIONS   [ALLWAYSREQUESTER]
  {
```

```
                 IFEXISTS
                  REQUESTER, SKIP, OVERWRITE, ADDNUMBER

                 REQUESTER
                  ARP, REQTOOLS, ASL, MULTIREQ
    {

                 PATTERN
                  OFF, ON, pattern
      LEFT    xpos
      TOP   ypos
      WIDTH   width
      HEIGHT    height
    }

                 ALLFIXBEHIND

                 PATH
                    path
    {
      LOAD  path
      SAVE  path
      TEMP  path
    }

                 COMMENT
                  NONE or template

                 EXCLUDE
                  pattern
  }
```

Set some options for global filehandling. If ALLWAYSREQUESTER is
added, a requester will open allways for loading and saving. If not
added, the requester will open only before overwriting files
(sub-keyword IFEXISTS REQUESTER must be set) or if no filename is found
by a filter or just a path or paths are specified in the commandline
or icon tooltypes. All sub-keywords are optional.

1.34 FILEOPTIONS/IFEXISTS

```
  FILEOPTIONS
  {
    ...
    IFEXISTS  REQUESTER, SKIP, OVERWRITE, ADDNUMBER
    ...
  }
```

How should Filterium behave when saving files with names that allready
exist? One of the above modes could be set. The meanings:

```
      ASK   everytime ask what to do
      REQUESTER requester before overwriting
      SKIP    no save if file exists
      OVERWRITE allways overwrite
      ADDNUMBER count a number to the name


Default:    ASK
```

1.35 FILEOPTIONS/REQUESTER

```
  FILEOPTIONS
  {
    ...
    REQUESTER ARP, REQTOOLS, ASL, MULTIREQ
    ...
  }
```

```
With one of the above parameters here you can set the type of
requester library to use. The last one is fine on oldest systems
and can be found as multireq.library in aminet,
but note: multireq understands only '*' as jokers in the pattern.
The postition and size options work only for asl really correctly
and arp and multireq support no multiselect.

Default:    as far I know, ARP is default
```

1.36 FILEOPTIONS/ALLFIXBEHIND

```
                  FILEOPTIONS
  {
    ...
    ALLFIXBEHIND
    ...
  }
```

```
Normally the filters supply filename pre- and suffixes when
identifying data. With this subkeyword you can force to put
them all allways behind the filename. For example,
StartrekkerAM 'mod.Name.NT' goes 'Name.mod.NT'. As you can
guess, this is good for usage on PC (with WinUAE).

Default :   pre- and postfix like 'mod.Name.NT', etc
Note  :   you can set the fixes for each filetype
      with the CLASS keyword
See also:
                 CLASS
```

1.37 FILEOPTIONS/PATH

```
                  FILEOPTIONS
  {
    ...
    PATH    path
    {
      LOAD  path
      SAVE  path
      TEMP  path
    }
    ...
  }
```

Main purpose of this is to set the default savepath. I a requester is opened, it will open with that directory. The temppath is for temporary files. With the loadpath you specify where to open a filerequester when loading files. If you specify only one single path just behind the PATH subkeyword, that will be used for all cases. And also the other paths would be take that as root. You can set own savepaths for a filetype, see CLASS. Also the builtin filter ’loadfile’ when used in a macro supports an own open path with the ARG keyword.

Example:

```
    PATH    { SAVE mod: TEMP ram: }
```

Other example: (savepath would be dh0:mymodulesdir/)

```
    PATH  dh0:  { SAVE mymodulesdir TEMP mytempdir }
```

```
Default   : current directory
See also  :
                 CLASS
```

1.38 FILEOPTIONS/REQUESTER/PATTERN

```
                  FILEOPTIONS
  {
    ...
    REQUESTER ...
    {
      ...
      PATTERN OFF, ON, pattern
      ...
    }
    ...
  }
```

With OFF you turn of the displaying of a show-pattern editbox in the filerequesters. ON does turn on that box. Or even you can put a pattern to start with for that box here. For a detailed description on the patterns see FILEOPTIONS/EXCLUDE.

Example:

```
      PATTERN "#?.adf"


Default   : off
See also  :
                 FILEOPTIONS/EXCLUDE
```

1.39 FILEOPTIONS/EXCLUDE

```
  FILEOPTIONS
  {
    ...
    EXCLUDE pattern
    ...
  }
```

```
Use this to select filenames that should be excluded from archives
or in combination with the commandline/icon ALL keyword.

This functions uses the case UNsensitive AmigaOS pattern matching
routines available since dos.library v37, so all fancy unix style
wildcards like described in dos.doc work! But:

 '+' will be translated to the vertical slash also, as I couldn't
     find it on my WinUAE keyboard layout ;(


Example:  EXCLUDE "(#?.(guide+txt)+#?(readme+displayme)#?)"

Default:  no exclusion


Excerpt from AmigaOS developer 'dos.doc':

  ...

  The patterns are fairly extensive, and approximate some of the ability
  of Unix/grep "regular expression" patterns.  Here are the available
  tokens:

  ? Matches a single character.
  # Matches the following expression 0 or more times.
  (ab|cd) Matches any one of the items seperated by '|'.
  ~ Negates the following expression.  It matches all strings
    that do not match the expression (aka ~(foo) matches all
    strings that are not exactly "foo").
  [abc] Character class: matches any of the characters in the class.
  [~bc] Character class: matches any of the characters not in the
    class.
  a-z Character range (only within character classes).
  % Matches 0 characters always (useful in "(foo|bar|%)").
  * Synonym for "#?", not available by default in 2.0.  Available
    as an option that can be turned on.

  "Expression" in the above table means either a single character
  (ex: "#?"), or an alternation (ex: "#(ab|cd|ef)"), or a character
  class (ex: "#[a-zA-Z]").

  ...
```

1.40 FILEOPTIONS/COMMENT

```
                  FILEOPTIONS
  {
    ...
    COMMENT NONE or template
    ...
  }
```

How to define how a comment for a saved file should look like. You
can set this globaly here for all files and also depending on
filetype (see CLASS keyword). With NONE you turn off COMMENT feature
globally, not affecting filetype specific comments.

The comment subkeyword supports the following jokers:

```
    %loadname name of original loaded file
    %type   type of saved data
    %name   name of saved data
    %afilter  'Amiga Filterium'
    %loc    load to $xxxxxxxx original adress
    %jsr    code at $xxxxxxxx (decrunch, start)
        todo:
    %author
    %anno
    %version
```

```
Example   : "%type from %loadname, loc %loc / jsr %jsr"

Default   : "%type - Amiga Filterium"
See also  :
                 CLASS
```

1.41 SETPLUGIN

```
  SETPLUGIN   pluginname { CFG ... { ... } }
```

Some of the plugins support their own configurations. With SETPLUGIN
you can embedd these in the global configuration or specify a
configuration filename for a plugin (or both combined).

Examples:

```
  SETPLUGIN   memview { CFG s:memview.cfg }
  SETPLUGIN   memview { CFG { PLANECOLOR $044 } }

Yet this is the only plugin that supports configuration.
More to come.
```

1.42 afilter.cfg/MACRO

```
                  MACRO   macroname
  {
    DESC    "description of my macro"

    CALL    filtername or macroname
    ...
    {
      CALL  filtername or macroname
      {
        ...
      }
      ...
    }
    ...
  }


Define own processing macros to optimize actions. The text
behind the DESC subkeyword will appear in the list of macros
that you can get by typing 'afilter MACROS' to the shell.
 A macro is like a script or a 'mill'. These tell Filterium what
filters to use and in which order. Use them to build in more or
less questions and branches or to call external programs,
shell or arexx commands.
 You can have a sound-mill, a mysterious-cruncher-mill or what
ever you need.
 Macros can call other macros as well, like subroutines
in programs.

Example:
  MACRO   grabkick
  {
    DESC  "save the kickstart rom to file"

    CALL  selectkickrom
    {
      CALL  savefile
    }
  }


Imagine a destilation tower. A first filter (here 'selectkickrom')
provides something like file, disk or memory. Then a subfilter
works with that data and provides outputs to a sub-subfilter in
the { ... } subsection(s) and so on. The last and deepest filter
in most cases is 'savefile' to save data or 'execute' to send
data outputs to an other program.
```

```
Multiple usage  : yes
See also  :
                 macros
                ,
      supplied configuration file   afilter.cfg
```

1.43 afilter.cfg/MICRO

```
                  MICRO   macroname
  {
    ...
  }
```

```
This is the same like
                 MACRO
                , but if you define a macro with the
MICRO keyword, it will not appear in the macro list that you can
get by entering 'afilter MACROS' to the shell. In a MICRO the DESC
subkeyword has no effect.


Multiple usage  : yes
```

1.44 afilter.cfg/CLASS

```
                  CLASS   name
  {
    CLASS ...
    ...
    PATH  path
    COMMENT NONE or template
    AUTO  filtername or macroname { commands }
    PREFIX  NONE or prefix
    SUFFIX  NONE or postfix
  }
```

```
To globalize reactions if specified type of data encountered
or to set global parameters for a bundle of types of data,
separated savepaths for example.

With PATH you can set an own savepath for a filetype here.
COMMENT works like global FILEOPTIONS/COMMENT.
With AUTO you can call or define a macro that should be done
allways if data of that type found.
PREFIX and SUFFIX override predefined filename fixes.
```

```
multiple usage  : yes
self contained  : yes
default   : some filters give predefined pre-/suffix,
      use PREFIX/SUFFIX to turn them off
      or change them
see also  :
                 filetype specific reactions
```