proutil

COLLABORATORS

	TITLE : proutil	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 20, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

proutil

1.1 proutil.guide

```
================================================================================
                              P R O U T I L S

                                    v1.0
================================================================================
```


This Manual

TABLE OF CONTENTS

These utilities may be useful while developing software. They all must be run from the shell, except otherwise noted. All of them are driven by keyword options in any order indicating the action to be

```
performed.  Quite all utilities display also a command line usage
notice when they are run with no parameters or a question mark (?) as
parameter.  For example:
         1> profiler ?
         Profiler - small run-time statistics utility v1.03
         Copyright © 1993,1994 by Daniel Weber

         Usage: Profiler [options] <program> [<program arguments>]

         Options: -n <num>  display entries with more than <num> hits.
                  -s        display symbol listing.
                  -t <num>  set time interval to <num> milliseconds (def: 2000).
                  -z        permit displaying entries with zero hits.

         Example: Profiler -s -n 50 foo foo_arguments




  Additional Manuals:
   ProAsm    ASX     ProOpts
```

1.2 about this manual

```
About This Manual
=================


    Welcome to the AmigaGuide version of the ProAsm utilities manual.


    This manual is a part of the ProAsm manual, which was originally written
in TeX and texinfo. You are currently looking at the AmigaGuide version of
the ProAsm utilities part. I used various tools to convert the primal TeX
source into AmigaGuide. They did quite a good job, but the result was over
700KBytes of length! As a consequence I had to shorten the whole guide file
by hand. Now it is very possible that when browsing through this manual you
will find sentences like:

          This node (page) intentionally left blank.
or
          This node (page) was intentionally shortened.

Please forgive me for that. I had also to skip various tables that can
be found in the original printed manual.

    Caused by the convertion of the TeX source of the manual into this
AmigaGuide file and the need of some small reorganisation of it, some of
the links go nowhere. Sorry for that, but I had not the time to check all
links. I did check a lot of them, but unfortunately there are still some of
these dead links in the guide file. If you should find any, please feel
```

free to inform me (Daniel Weber).

If you find some strange (unusal) parts in this AmigaGuide file, please keep in mind, that this guide initially was written as a book. You can order a copy of the printed manual, if you wish. Please read the registration document (register.doc or Registration) for further information.

Daniel Weber
Zurich
November 1995

1.3 notices

Copyrights
==========

ProAsm and its associated utilites are copyrighted by Daniel Weber except otherwise noted.

Trademarks
==========

ProAsm and ASX are trademarks of Daniel Weber. MC68000, MC68008, MC68010, MC68020, MC68030, MC68040, MC68060 MC68EC020, MC68EC030, MC68EC040, MC68LC040, MC68881, MC68882, MC68851 and Motorola are trademarks of Motorola, Inc. Amiga is a registered trademark of ESCOM AG. AmigaDOS, Kickstart, and Workbench are trademarks of ESCOM AG. SAS and Lattice are registered trademarks of SAS Institute, Inc. Aztec and Manx are trademarks of Manx Software Systems. ARexx is a trademark of The Wishful Thinking Development Corp. UNIX is a registered trademark of AT&T. OS-9 is a registered trademark of Microware Systems Corporation. All products mentioned in this manual are trademarks of their respective owners.

Disclaimer
==========

The information, the ASX program, and all the associated utilities are provided "as is" without warranty of any kind, either expressed or implied. The entire risk as to the accuracy of the

information herein is assumed by you. Daniel Weber does not warrant, guarantee, or make any representations regarding the use of, or the results of the use of, the information, the ASX program, or the associated utilities in terms of correctness, accuracy, reliability, currentness, or otherwise. In no event will Daniel Weber be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the information, the ASX program, or the associated utilities even if he has been advised of the possibility of such damages.

1.4 profiler

```
Profiler
--------

 - small run-time statistic utility: Profiler [options ] program [program  ↩
    arguments ]
     The Profiler utility displays the run-time statistics for a given
     program with its optionally given program arguments.  It records
     the amount of time spent in each routine that is called while the
     program runs.  When your program ends, Profiler displays the
     resulting run-time statistic.

     In a line of the statistic output you find (in this order):
        * the name of the particular routine,

        * the hunk number where the routine is located,

        * an offset value that describes the begin of the routine
          relative to the hunk start,

        * the number of hits Profiler counted while the program was
          running,

        * the average percentage of time that was spent in this routine,
          compared with the whole program,

        * and the location of the routine in your source code.

     To use Profiler with your program, you have to assemble your
     program with the debug option and symbol information turned on
     (e.g. DEBUG LINE,ALL,CODE,ADDSYM, see Debugging Information, for
     more details).

     Profiler supports the following command line options:
    -n num
          Tells Profiler to display only routines in the run-time
          statistic with more than num hits.  (Where num is a decimal
          number.)

    -s
          This option forces Profiler to display a symbol listing.

    -t num
          Using this option you can specify the time interval, in
```

```
        milliseconds, between the statistic checks Profiler does on
        your program.  If no time interval is given, Profiler
        defaults to 2000.  (Where num is a decimal number higher or
        equal to 100.)

    -z
        To keep the statistic information as compact as possible,
        Profiler does not list routines with zero hits.  However,
        this option forces Profiler to display these routines also.

     Note that a sequence of options can be written separately (e.g. -s
     -z -t 1000) or together (e.g. -szt 1000).
```

1.5 prohunk

```
ProHunk
-------

 - 680x0/688xx Hunk Analyzer: ProHunk object file  [options ]
     ProHunk is an Amiga object file analyzer that issues the binary
     hunk structure of a load or linkable object file.  All hunks up to
     the operating system version 3.1 (v40) are supported.  See 'The
     AmigaDOS Manual', 3rd Edition, Bantam Computer Books, for more
     information about AmigaDOS hunk formats.

     Next to the hunk information, ProHunk can also display optionally
     the contents of code hunks in disassembled form and data hunks as
     hexdump.  The disassembler supports all Motorola processors up to
     the MC68040, the MC68881 and MC68882 floating-point coprocessors,
     and the MC68851 PMMU coprocessor.

     The given object file is the name and complete path of the file
     that is to be analyzed.  The options are optional and should be
     one or more of the following:
    -d
        Output the contents of all data hunks as hexdump.

    -c
        Output the contents of all code hunks in disassembled form.

    -i
        Show contents of all debug hunks (hunk_debug) as hexdump.

    -i0
        Show contents of all known debug hunks (hunk_debug) in a
        line/offset format.

    -q
        Quiet, send no output to StdOut.

    -v
        Verbose, display all entries in symbol (hunk_symbol) and
        hunk_ext hunks.
```

```
Note that a sequence of options can be written separately (e.g. -i
-c) or together (e.g. -ic).

Example:
          1> ProHunk ProHunk
          ProHunk - 680x0/688xx Hunk Analyzer v1.48
          Copyright © 1991-1996 by Daniel Weber - All Rights Reserved
          analyzing file ProHunk, ok...

          hunk_header
               number of hunks: 1  (lo:0, hi:0)
               hunk #0: $0000444C (17484) bytes  (loadtype: public)
          hunk_code
               (hunk #0, size: 15704, bss: 1780)
          hunk_end
```

1.6 pre2src

```
Pre2Src
-------

 - Convert preasm files into equate files: Pre2Src preasm file  output file
     Pre2Src generates an assembler equate file (output file) out of a
     given preasm file.

               1> Pre2Src myProg.pre myProg.equ

     The following error messages can be reported by Pre2Src:
    Couldn't load file 'filename'
          An AmigaDOS error occurred while loading the specified file
          (filename).

    Couldn't open file 'filename'
          Pre2Src could not open the specified file (filename).  Check
          the spelling of filename and that the supplied pathname is
          correct.

    Not a valid preasm file
          The file given as preasm file is not a correct preasm file.
          Check if the correct file was given.

    Unsupported preasm file entry encountered
          Pre2Src encountered an unsupported entry while processing the
          preasm file.  This error only occurs if the specified preasm
          file is crippled.

    Unsupported preasm file version
          You tried to load an unsupported version of a preasm file.
```

1.7 mmuinfo

```
MMUInfo
-------

 - Display information on the MMU: MMUInfo [-option  address ]
     MMUInfo displays various information about the current MMU
     settings and registers, or on a supplied address.

     MMUInfo supports memory management unit of the MC68030, MC68040,
     and the MC68851 processors.  For more information about the memory
     management unit (MMU) refer to the manuals for the specific
     microprocessors.

     If MMUInfo is run without any parameters, it reports the current
     settings of the MMU and its registers contents.

     If an option and an address is given, MMUInfo displays the
     translation and the page attributes of this address according to
     the given option.  Address is the hexadecimal address of the
     memory location the information is requested for.

     The following two options are supported by MMUInfo:
    -d
          Use the -d option to perform a data space access on the given
          address.

    -i
          Use the -i option to perform an instruction space access on
          the given address.

     For example:
               1> MMUInfo -d $fc0000

     The following error messages can be reported by MMUInfo:
    CPU must be 68020/30/40!
          MMUInfo expects a MC68020 or higher processor to run.
          Supported are the memory management unit of the MC68030,
          MC68040, and the MC68851 processors.

    no PMMU found!
          No memory management unit found.
```

1.8 fd2lvo

```
FD2LVO
------

 - Convert fd-files to _LVO equate files: FD2LVO [input path ] output file  [ ↩
    options ]
     The FD2LVO utility generates equate listings with library vector
     offsets using fd files.
```

```
     The utility can be started from the CLI/Shell in three different
     ways:
    FD2LVO input file output file
          The fd-file input file is parsed and converted to library
          vector offsets and written to the equate listings (output
          file).

    FD2LVO input path output file -a
          All valid fd-files in the given directory (input path) are
          parsed and converted to library vector offsets and then
          written to the equate listings (output file).

    FD2LVO output file -a
          If no input path is given FD2LVO scans the FD: assign.  All
          valid fd-files are parsed and converted to library vector
          offsets and then written to the equate listings (output file).

     The following error message can occur while processing the
     fd-files:
    No output generated.
          No output is generated if an error occurred while FD2LVO
          processes a fd-file.

    Couldn't open file '%s' (DOS error: %ld).
          FD2LVO cannot open the specified file.  Check the spelling of
          input file and output file.  The DOS error code may give you
          additional information why the assembler could not open the
          file (see AmigaDOS Error Codes).

    Output file crippled.
          An error occurred while writing data to the output file.  The
          output file may be crippled.
```

1.9 clicalc

```
CLICalc
-------

 - Simple CLI calculator: CLICalc expression
     CLICalc allows you to evaluate a given expression in an easy way.
     The returned result will be displayed as signed hexadecimal,
     unsigned hexadecimal, decimal, binary number, as ascii chars, and
     as octal number.

     The following operators are supported, sorted after their priority
     (see also Operators):

     CLICalc supports the following forms for numbers (see also
     Integer Constants for more details about integer constants and
     their formats):
     $d
    0xd
```

```
      Hexadecimal number.

d
#d
      Decimal number.

 %ddd
      Binary number.

 @d
 0d
      Octal number.

 'c'
 "c"
      ASCII string.

 CLICalc works in the decimal mode, meaning that you do not have to
 write the pound sign (#) in front of each decimal number in the
 expression.

 For example:
          1> CLICalc 14*2-(33/$a)&%011010
          =$1A  =$1A  =#26  =%00000000.00000000.00000000.00011010
          ='....' =@32
          1>

 If an error had occurred while evaluating the given expression, an
 error message is reported.  Below you find a complete list of
 possible error messages:
calcbuffer overflow
      This error should not occur.

divide by zero
      During the evaluation of the expression CLICalc encountered a
      division by zero.  Check the expression for any division by
      zero.

illegal value
      The expression evaluator could not find a valid value (a
      constant or an expression).

invalid number
      An invalid digit was encountered in a number.  For example,
      using a 2 or 3 in a binary number:
                  move.l  #%1010201,d0

missing close brackets
      An unmatched open bracket (() was found in an expression.

missing quote
      An unmatched quote (single or double quote) was found in a
      string.  Make sure the string is correctly enclosed in quotes.
      You may not mix single and double quotes.

not enough memory
      There is not enough memory available for CLICalc to complete
```

```
      its task.

   no value given
      No term given, or the expression evaluator could not find a
      valid term.  A constant or expression might be invalid.

   overflow
      This error is reported if an overflow occurs while evaluating
      an expression.  In other words, the result of the evaluated
      expression is larger than $ffffffff, +$7fffffff, or
      -$80000000.

   too many close brackets
      Too many close brackets ()) are found.  Either more open
      brackets have to be inserted or the superfluous close
      brackets have to be removed to get properly balanced brackets.
```

1.10 fcalc

```
FCalc
-----

 - CLI IEEE double precision calculator: FCalc expression
     FCalc is a command line oriented calculator for IEEE double
     precision numbers.  The returned result will be displayed as a
     double precision number.

     See Floating-point Constants, for details about the format of the
     floating-point numbers.

     The following operators are supported, sorted after their priority:

     Next to the operators listed above, the following standard
     functions can be used:

     Examples:
              1> FCalc 1.0e2*14
              1.4E3

              1> FCalc (3.14e1+2.17e2)*4
              9.936E2

              1> FCalc sin(2.2346)
              7.87654426210142E-1

              1> FCalc ln(abs(tan(7.87654426210142E-1)))
              4.51254094014066E-3

     If an error had occurred while evaluating the given expression, an
     error message is reported.  Below you find a complete list of
     possible error messages:
    calcbuffer overflow
      This error should not occur.
```

illegal value
The expression evaluator could not find a valid value (a constant or an expression).

invalid number
A number was encountered that is not a valid floating-point number. See Floating-point Constants.

missing close brackets
An unmatched open bracket (() was found in an expression.

not enough memory
There is not enough memory available for FCalc to complete its task.

no value given
No term given, or the expression evaluator could not find a valid term. A constant or expression might be invalid.

too many open brackets
Too many close brackets ()) are found. Either more open brackets have to be inserted or the superfluous close brackets have to be removed to get properly balanced brackets.

unknown function
FCalc encountered an unknown function in the expression. Use FCalc ? to display a list of valid functions.

1.11 stripd

StripD

- Strips debugging information from a load file: StripD source destination
StripD is a small utility that removes all symbol and debug information from the object file given as source. The stripped file will then be written to destination.

The following error messages can be reported by StripD while processing a file:
couldn't open file 'filename' (DOS error: DOS error code).
StripD cannot open the specified file. Check the spelling of filename and that the supplied pathname is correct. The DOS error code may give you additional information why the assembler could not open the file (see AmigaDOS Error Codes).

DOS error DOS error code occurred during processing file - abroted.
An AmigaDOS error occurred while processing a file (see AmigaDOS Error Codes). The operation is aborted. There may exist a crippled output file destination.

1.12 fcmp

```
FCmp
----

 - File compare utility: FCMP firstfile  secondfile
     FCmp is a small file compare utility that compares two given files
     and reports the first found difference.

     When a difference is found FCmp displays an appropriate message
     that informs you at what offset the files differ.  The offset,
     relative to the start of the files, is given as a hexadecimal and
     a decimal number.

     The following messages can be reported by FCmp:
    first difference at offset $x (d)
          This message is reported when a difference is found.  It
          informs you at what offset the first difference is found.
          The offset, relative to the start of the files, is given as a
          hexadecimal (x) and as a decimal number (d).

    length of file differs
          When the file lengths differ FCmp generates this message.  No
          further comparisons will be made.  Since the file lengths
          differ the files are definitely not equal.

    error reading file!
          An error occurred while writing the source file.  Check if
          another program makes write accesses to the file.

    invalid parameter line
          The parameter line has not the requested format.  FCmp
          accepts only the names of the two files that are to be
          compared.

    could not open file filename
          FCmp cannot open the specified file.  Check the spelling of
          filename and that the supplied pathname is correct.
```

1.13 bin2dc

```
Bin2DC
------

 - : Bin2DC file  output source  [options ]
     Bin2DC generates an assembler source code out of a given file
     using the DC directive (see Initialized Data).  The assembler
     source is then written to output source.

     One of the following options can be used to format the source code:
    -b
          Use DC.B in the source code.
```

```
    -w
          Use DC.W in the source code.

    -l
          Use DC.L in the source code.

     DC.L is used as default (if no option was specified).

     Example:
              1> Bin2DC Bin2DC ram:bin2dc_source.s -b
```

1.14 bdiff

```
BDiff
-----

 - small binary file compare program: BDiff oldfile  newfile  [options ]
     The BDiff program determines the differences in contents between
     two files and describes these changes in a special output file
     (bdiff file).  The output file will only be generated if any
     changes were found.

     The produced bdiff file can be used later to reconstruct the
     newfile out of the oldfile or vice versa, except if the -s option
     was used while these files were compared.

     BDiff supports the following options:
    -n nnn
          This option sets the minimum number of bytes to nnn that must
          be identical to be considered  a match.  By default this
          value is set to 8 bytes.

    -o filename
          Sends the output to the specified file (filename).  The
          output is either a bdiff file, that contains the description
          of all differences, or the file generated out of an input and
          a bdiff file.

    -s
          This option can be used to reduce the quantity of information
          stored in the resulting difference file.  Only the update
          information from the oldfile to the newfile will be stored.
          Hence only the newfile can be reconstructed.

    -t1
    -t2
          BDiff normally determines by itself which file must be
          generated out of an input file and a bdiff file.  However, if
          BDiff aborts an operation with a file checksums are equal -
          use -t1 or -t2 option. error message, you have to use one of
          these options to tell BDiff what file you want to have
          generated.  -t1 forces the oldfile to be produced and -t2 the
          newfile.
```

-u
This option tells BDiff to reconstruct a file according to the given input file and bdiff file. BDiff determines by itself if either the former oldfile or the former newfile is to be rebuilt (see also the description of the -t1 and -t2 options). Using this option, BDiff expects the input file given as first argument (oldfile) and the bdiff file as second argument (newfile). The -o option must also be used to specify the resulting output file.

-x
Since the generated bdiff file, that contains the descriptions of all differences, is no ASCII text file, you cannot read it using a text viewer or similar utilities. The -x option analyzes and disassembles a given bdiff file, and writes the output to the StdIO in an almost human readable format. It looks similar to the following example:
The first number, the hexadecimal number within the brackets, describes the internal code number and can be ignored. Offset#1 is the current offset in the first file, and consequently, offset#2 describes the offset in the second file. The second number in the offset field is the number of bytes from the end of the last difference to the beginning of the current difference. The length fields contain the number of bytes that do not match. See at the third line of the example above. There, length#1 is four and length#2 is 50 bytes. For example you want to generate the second file out of this information, this tells you that the four byte sequence at offset#1 has to be replaced by the 50 bytes starting at offset#2.

-y nnn
This sets the I/O buffer size to the value specified by nnn. The default buffer size is 4096 bytes and its maximum is limited by the amount of available memory.

The following error messages can be reported by BDiff:
couldn't open file 'filename' (DOS error code)
BDiff cannot open the specified file. Check the spelling of filename and that the supplied pathname is correct. The DOS error code may give you more detailed information why the assembler could not open the file (see AmigaDOS Error Codes).

error occurred during processing files
An AmigaDOS error occurred while the files were compared.

error occurred during writing file
An AmigaDOS error occurred while the output file was written.

file checksums are equal - use -t1 or -t2 option
If the checksums of the former oldfile and newfile were equal, BDiff is not capable to decide whether the oldfile or the newfile should be rebuilt. Use the -t1 or -t2 option to tell BDiff which file has to be reconstructed.

invalid output file checksum - file may be corrupt

After a file is rebuilt (see -u option), BDiff checks its checksum. An invalid checksum indicates that the produced file may be corrupt. This error can occur when a wrong input file is given that, unfortunately, has the same checksum as one of the former compared input files (oldfile or newfile).

invalid parameter line
The parameter line has not the requested format, or one or more unknown parameters have been encountered. Start BDiff with the question mark (?) as only argument to get a small usage information displayed.

invalid source file checksum
When a file has to be rebuilt, BDiff checks the checksum of the input file (source file) to assure that the input file can be used with the given bdiff file. Check if the correct source file was given.

not a valid BDiff file
The given bdiff file may be corrupt or invalid.

1.15 unbdiff

```
UnBDiff
-------
```

- small binary un-bdiff: UnBDiff input file bdiff file -ooutput file

The UnBDiff program can be used to reconstruct a file according to the given input file and bdiff file. UnBDiff determines by itself if either the former oldfile or the former newfile is to be rebuilt.

UnBDiff expects the input file given as first argument and the bdiff file file as second argument. The -o option must also be used to specify the resulting output file.

This utility performs the same job as BDiff input file bdiff file -u -ooutput file does (see its description above). UnBDiff is simpler to use with its fixed command line usage, and can therefore be useful for software and update distributions.

The following error messages can be reported by UnBDiff:

couldn't open file 'filename' (DOS error code)
UnBDiff cannot open the specified file. Check the spelling of filename and that the supplied pathname is correct. The DOS error code may give you more detailed information why the assembler could not open the file (see AmigaDOS Error Codes).

error occurred during writing file
An AmigaDOS error occurred while the output file was written.

invalid output file checksum - file may be corrupt
After a file is rebuilt UnBDiff checks its checksum. An invalid checksum indicates that the produced file may be

corrupt. This error can occur when a wrong input file is given.

invalid parameter line
The parameter line has not the requested format, or one or more unknown parameters have been encountered. Start UnBDiff with the question mark (?) as only argument to get a small usage information displayed.

invalid source file checksum
When a file has to be rebuilt, UnBDiff checks the checksum of the input file to assure that the input file can be used with the given bdiff file. Check if the correct input file was given.

not a valid BDiff file
The given bdiff file may be corrupt or invalid.

1.16 author

Author
======

If you have bugreports, questions, ideas, flames or complaints (constructive criticism is always welcome), or if you just want to contact me, write or send a letter to:

Daniel Weber

Internet: dweber@amiga.icu.net.ch (preferred)
dweber@iiic.ethz.ch

Mail: Daniel Weber
Hoeflistrasse 32
CH-8135 Langnau
Switzerland.