hyper.hyper

COLLABORATORS

	TITLE : hyper.hyper	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 26, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

hyper.hyper

1.1 Hyper v1.0 - © Koessi 92

```
                ********************************************************* ↩

*                                                           *
*               This program is SHAREWARE !                 *
*   To register and to support further development send    *
*                                                           *
*    ---------->> DM 10,00/$ 10,00/£ 5,00 <<----------      *
*                                                           *
*   to: Koessi, Peterstr.60, W5609 Hueckeswagen, Germany    *
*                                                           *
*-----------------------------------------------------------*
*                                                           *
*         Bugreports & special wishes are welcome!          *
*                                                           *
* A file called  "strings.c"  is included.  If U  want a    *
* version in your own language, feel free to change that    *
* file  and  send it to me.   No problem,   but keep the    *
* strings short. Also I'm strongly interested in all the    *
* developers supporting stuff from Commodore  -  I don't    *
* know where and how to get it  -   perhaps somebody can    *
* help me - thanx and have fun with your machine.           *
*                                                           *
*************************************************************

What is it good for ?

Hyper will show documents that are written to be used with
the legendary
       Am*gaGu*de
        from Commodore. Several authors
do already use it, but I, as a more normal user, have no
access to it. So I decided to write my own version.

Where is the problem ?

Hyper needs Am*gaDOS 2.0 ! U2 \8-( ?
and a lot of memory ...
```

1.2 Getting started

Hyper can be called from the CLI by typing:

[prompt]> Hyper <-b> <document> <nodename> <return>

The option '-b' will make Hyper stay in the background until it is called via its
ARexx-Port
. Use it in the startup-sequence if U have enough memory. Without that option and if U don't supply a
document-file
to read in, Hyper
will open the asl-requester first. If the directory containing the hyper-docs is assigned as "HYPER:" this one is the default. Supplied docnames are searched first in the current directory, then in HYPER: and there also with the suffixes ".hyper" and ".guide" appended.

If no nodename is given or if it is not found in the document, the "main" node is shown as the first page.

On WorkBench it behaves completely normal: just doubleclick its icon. U may shift-select the first project to work with.

The
window
will appear on the WorkBenchScreen (default size is 640x200).

1.3 Compatibility with `Am*gaGu*de'

The only source I found were some documents published in the Fish ↩
-Library.
I don't know wether there are more keywords possible, than those guys used. Hyper also supports 'links' to nodes inside other docs (and it reminds the way back).
Hyper ignores the @width and @height informations, I can't find any use for them. If the nodes have an extra name, Hyper presents it as a headline on top of the page. The windowtitle is taken from the 'main'-node.
I've heard, that 'Am*gaGu*de' is a shared library, but I don't know any of it's calling conventions, so Hyper cannot emulate that calls.
Instead I've implemented an
ARexx-Port
.

1.4 The Hyper window

```
                The window will appear on the WorkBenchScreen (default size is ↩
                   640x200).
U may change the size in the usual ways.
The gadgets at the bottom of the window control the following items:

 Return      - if U want to quit Hyper and remove it from memory
 Load Doc    - if U want to read in another document
 First Page  - <f>, <F>, <home>
 Prev Page   - will go back one chapter in the document
 Next Page   - will move forward to the next page in the document
               (this may be confusing, because the chapters do not need to
                be arranged in a logical order)
 Sleep       - will make Hyper close its window and wait for a message to
                its
                ARexx-Port
                . Also an AppMenuItem is
                added to the WorkBench's Tools-menu named "WakeUpHyper".

On the right edge of the window a scroller gadget allows to move up and
down inside the chapter.

There are several shortcuts build in to access the gadget-functions via
keyboard:

 Return      - <r>, <R>, <return>, <enter>, <esc>
 Load Doc    - <l>, <L>
 First Page  - <f>, <F>, <home>
 Prev Page   - <p>, <P>, <backspace>, <cursor-left>, <page-up>
 Next Page   - <n>, <N>, <space>, <cursor-right>, <page-down>
 Sleep       - <s>, <S>

 Scroll-up   - <cursor-up>
 Scroll-down - <cursor-down>

If the
                document-file
                 contains links, they will appear in
the shown text in inversed colors. Doubleclick on those keywords to jump
into the assotiated chapters - that's it.
```

1.5 The Hyper ARexx Port

```
                To make Hyper accessable for your applications I've build in an ↩
                   ARexx port.
To keep things simple there's only one way to use it:

Send a RexxMsg with only one string in the first RexxArg slot.
This string should have this format:

                  "<docname>/<NODENAME>"
```

```
 where the nodename must be in capital letters only.

The portname is "HYPER_RXPORT".

An example ARexx-script called
               test.rexx
                is part of this distribution.

Using the port is one way to wake Hyper up while it is in sleepmode.
The easiest way is to call hyper again, because it will detect the port,
bring up that's window and terminate itself. Another way is to use the
provided tool "
               help
               ", that will do the same being much smaller.
```

1.6 The `help' tool

```
                The easiest way to wake Hyper up from CLI while it is in  ↩
                   sleepmode, is
to use the provided tool "help".

            [prompt]> help <document>/<NODENAME> <return>

where the nodename must be in capital letters only.

Supplied docnames are searched first in the current directory, then in
HYPER: and there also with the suffixes ".hyper" and ".guide" appended.

If no nodename is given or if it is not found in the document, the "main"
node is shown as the first page.

The
               C-source
                of "help" is part of this distribution, because it may give
U an idea on how to use Hyper from inside your own programs.
```

1.7 Author & Credits

```
          Koessi © 9.92

          phone germany 02192 7630

          thanx to ...

          Matt Dillon     - done this with DICE only - have fun!
          Fred Fish       - for the Library

          ... and all authors of nonorlesscommercial Am*ga software.
```

1.8 test.rexx

```
/*  call hyper */

PARSE ARG doc ' ' node

IF (SHOW('P', 'HYPER_RXPORT')) THEN DO

  IF (node != "") THEN
    doc = doc'/'node

  ADDRESS 'HYPER_RXPORT'

  doc

END
```

1.9 help.c

```
/*  Compile with DICE:          */
/*  dcc help.c -ohelp -rr -2.0  */

#include <exec/types.h>
#include <exec/execbase.h>
#include <exec/memory.h>
#include <dos/dos.h>
#include <dos/dostags.h>
#include <rexx/storage.h>

/* Prototypes */
#include <clib/exec_protos.h>
#include <clib/dos_protos.h>
#include <clib/alib_protos.h>
#include <clib/rexxsyslib_protos.h>

#define  MSG struct Message
#define RMSG struct RexxMsg
#define MSGP struct MsgPort
#define SIZE 32

extern void _main(int, char *);

char portname[] = "HYPER_RXPORT";
char command[]  = "SYS:Utilities/hyper -b";

void
_main(int arglen, char *argptr)
{
  MSGP *reply_port;
  MSGP *rx_port;
  void *rx_msg;       /*  casted to parts of a RexxMsg struct */
  char *msgtxt = argptr;

  while (*argptr && *argptr != '\n')
```

```
    ++argptr;

  *argptr = '\0';

  if (!FindPort(portname))
  {
    SystemTags(command,
               SYS_Asynch, TRUE,
               SYS_Output, NULL,
               SYS_Input,  NULL,
               TAG_DONE);
    Delay(10);
  }

  if (reply_port = CreateMsgPort())
  {
    if (rx_msg = AllocVec(sizeof(RMSG), MEMF_PUBLIC|MEMF_CLEAR))
    {
      ((struct Node *)rx_msg)->ln_Type      = NT_MESSAGE;
      ((MSG         *)rx_msg)->mn_ReplyPort = reply_port;
      ((MSG         *)rx_msg)->mn_Length    = sizeof(RMSG);
      ((RMSG        *)rx_msg)->rm_Args[0]   = msgtxt;

      Forbid();
      if (rx_port = (MSGP *)FindPort(portname))
      {
        PutMsg(rx_port, (MSG *)rx_msg);
        Permit();
        WaitPort(reply_port);
        ReplyMsg(GetMsg(reply_port));
      }
      else
        Permit();

      FreeVec(rx_msg);
    }
    DeleteMsgPort(reply_port);
  }
}
```