ARexxBox.hyper

COLLABORATORS

	TITLE : ARexxBox.hyper	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 5, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ARexxBox.hyper

1.1 ARexxBox.doc

1.2 ARexxBox/ARexxDispatch

```
                  NAME
ARexxDispatch -- get ARexx command from MsgPort and execute it

 SYNOPSIS
ARexxDispatch( rexxhost );

void ARexxDispatch( struct RexxHost * );

 FUNCTION
ARexxDispatch fetches and executes all queued commands from
the given RexxHost's message port.
```

```
If a reply for some previously (with
              SendRexxCommand()
              ) sent
command comes in, the counter variable for still outstanding
replies will be decreased by one and the RexxMsg and it's
associated memory will be freed by
              FreeRexxCommand()
              .

In the main program, you should just check for the signal
of the host's message port and call ARexxDispatch()
without actually getting the message. All work will be
done by the dispatcher.

 INPUTS
rexxhost - pointer to an active RexxHost structure with
     a valid MsgPort

 RESULTS

 SEE ALSO

              SendRexxCommand()
              ,
              SetupARexxHost()
              ,
              DoShellCommand()
```

1.3 ARexxBox/CloseDownARexxHost

```
                  NAME
CloseDownARexxHost -- close & free ARexx host

 SYNOPSIS
CloseDownARexxHost( rexxhost );

void CloseDownARexxHost( struct RexxHost * );

 FUNCTION
CloseDownARexxHost() waits until replies for all pending
ARexx commands have been received and then closes the
ARexx port and frees all memory associated with that
RexxHost structure.

All messages sent to a closing host will be replied
immediately with an error "Host closing down".

 INPUTS
rexxhost - the RexxHost to close down

 RESULTS

 SEE ALSO
```

SetupARexxHost()
,
SendRexxCommand()

1.4 ARexxBox/CommandShell

NAME
CommandShell -- process Commands from a file

SYNOPSIS
CommandShell(rexxhost, fhin, fhout, prompt);

void CommandShell(struct RexxHost *, BPTR, BPTR, char *);

FUNCTION
CommandShell() sets the Flag ARB_HF_CMDSHELL in the
RexxHost's flag field and then processes input from fhin
until EOF or the CmdShell flag in the RexxHost being cleared
(e.g. by the standard Rexx command "CMDSHELL CLOSE").

The input is read line-wise, with newline as EOL. Each
line will be parsed and executed just like a built-in custom
ARexx command, exactly like it was called via an ARexx host
messageport.

The parsing and execution of each line is done by the
function
DoShellCommand()
.

If fhout is not NULL, the output of the commands will be
printed to fhout. The output of the commands will NOT be
assigned to any variables, as there is no underlying ARexx
script program that could hold these variables. Instead,
the output will be formatted to be human-readable.

The prompt string (if not NULL) will be printed to fhout
as an input request before reading an input line.

New (ARB 0.99d): The rexxhost parameter has to point to a
valid RexxHost structure. This is for identifying which
command shell belongs to which window/instance of the main
process.

New(ARB 0.99e): To support the "RX" command sending
asynchronous messages to ARexx, this function now catches
the replies of those messages and frees them using

FreeRexxCommand()
. Messages sent _to_ this host will be
replied immediately with an error "CommandShell Port".

INPUTS
rexxhost - an initialized RexxHost structure

```
fhin - the input FileHandle (see dos.library/Open())
fhout - the output FileHandle (or NULL)
prompt - the prompt string (or NULL)

 RESULTS

 SEE ALSO

              DoShellCommand()
              ,
              ARexxDispatch()
              , dos.library/Open()
```

1.5 ARexxBox/DoShellCommand

```
                   NAME
DoShellCommand -- parse & execute a command line

 SYNOPSIS
DoShellCommand( rexxhost, commandline, fhout );

void DoShellCommand( struct RexxHost *, char *, BPTR );

 FUNCTION
DoShellCommand parses the given string assuming it contains
an ARexx-style command line.

New (ARB 0.99e):  If normal parsing fails, the external
function
              ExpandRXCommand()
               will be called to expand any
macros.  If the expansion fails or the expanded command
can't be recognized either, an error will be returned.

If no errors occur during parsing, it tries to execute the
command with the given arguments. The results of the command's
execution will be printed in a human-readable format to fhout
if fhout is not NULL.

If errors occur, DoShellCommand prints a string describing
the error to fhout (if not NULL).

New (ARB 0.99d): The rexxhost parameter has to point to a
valid RexxHost structure. This is for identifying which
command shell belongs to which window/instance of the main
process.

 INPUTS
rexxhost - an initialized RexxHost structure
commandline - the string to be parsed & executed
fhout - the output FileHandle (or NULL)

 RESULTS
none
```

```
 SEE ALSO

              CommandShell()
              ,
              ExpandRXCommand()
              , <dos/dos.h>
```

1.6 ARexxBox/ExpandRXCommand

```
                  NAME
ExpandRXCommand -- expand macros and/or aliases (V0.99e)

 SYNOPSIS
newcommand = SendRexxCommand( rexxhost, oldcommand )

char *ExpandRXCommand( struct RexxHost *, char * );

 FUNCTION
This is an 'external' function you should provide if you
want to have command aliases or the like. The minimal
version of this function is just a return(NULL) as generated
in the rxif module.

ExpandRXCommand() will be called by the parser if it doesn't
know how to interpret a command string. Expansion could now
for example be a look up in the host's macro table.

Any strings returned by this function have to be allocated
explicitly using the standard C memory functions.  The
calling parser will free() them.

 INPUTS
rexxhost - the RexxHost we are working on
oldcommand - the commandline the parser doesn't know

 RESULTS
newcommand - an explicitly allocated memory area con-
       taining the expanded command (or NULL)

 SEE ALSO

              DoShellCommand()
              ,
              ARexxDispatch()
```

1.7 ARexxBox/FindRXCommand

```
 NAME
FindRXCommand -- search the ARexxBox command table (V0.99e)
```

```
 SYNOPSIS
rxscmd = FindRXCommand( command )

struct rxs_command *FindRXCommand( char * );

 FUNCTION
This function returns a pointer to the given command's entry
in the ARexxBox-generated command table.  It exists to
support those functions working on/with commands, like HELP
or ENABLE/DISABLE.

This function does no macro expansion.  The comparisons are
case independant so you don't have to convert your input to
upper case beforehand.  As this is exactly the routine used
by the parser to find a command, it will handle
abbreviations.

 INPUTS
command - the command name to search for

 RESULTS
rxscmd - the rxs_command structure of that command
   (or NULL if command not found)

 SEE ALSO
```

1.8 ARexxBox/FreeRexxCommand

```
                 NAME
FreeRexxCommand -- free the associated memory of a RexxMsg

 SYNOPSIS
FreeRexxCommand( rexxmessage );

void FreeRexxCommand( struct RexxMsg * );

 FUNCTION
This is basically a PD ARexx routine provided by William S. Hawes.

It frees all memory associated with a particular (previuosly sent)
ARexx message structure. It will also close any stdin/stdout
channels associated to that Rexx message.

You normally shouldn't have to bother with this one because the
dispatcher will call it for you.

 INPUTS
rexxmsg - the rexx message to free

 RESULTS

 SEE ALSO

              SendRexxCommand()
```

1.9 ARexxBox/ReplyRexxCommand

```
                  NAME
ReplyRexxCommand -- reply a rexx message from rexxmast

 SYNOPSIS
ReplyRexxCommand( rexxmsg, primary, secondary, result );

void ReplyRexxCommand( struct RexxMsg *, long, long, char * );

 FUNCTION
This is a PD ARexx routine provided by William S. Hawes.

It replies a given rexx message to the rexx master process,
filling in a primary and a secondary return code plus
optionally a supplied result string.

The result string will only be converted to an ARexx string,
if the primary return code equals 0, and will then destroy the
contents of the secondary return code. So you provide either
primary and secondary return codes or a result string.

You normally shouldn't have to call this function!
It is only mentioned here, because it is not part of the
ARexxBox routines, but part of the original ARexx distribution
by William S. Hawes.

New (ARB V0.99d): Now creates an ARexx variable "RC2" for the
secondary return code. If primary is positive, secondary is
interpreted as a long, if primary is negative, secondary is
interpreted as a char *. RC will become positive in any case.

RC2 will only be assigned if the ARexx RESULT flag is set.

 INPUTS
rexxmsg - the message structure to reply
primary - the primary return code (rc)       ( >0      <0   )
secondary - the secondary return code (rc2) (long or char *)
result - the result string

 RESULTS

 SEE ALSO

              SendRexxCommand()
              ,
              FreeRexxCommand()
```

1.10 ARexxBox/SendRexxCommand

```
                 NAME
SendRexxCommand -- invoke rexx command script

 SYNOPSIS
rexxmsg = SendRexxCommand( rexxhost, command, filehandle )

struct RexxMsg *SendRexxCommand( struct RexxHost *, char *, BPTR );

 FUNCTION
This is basically a PD ARexx routine provided by William
S. Hawes.

This function sends the given command string to the ARexx
master process for execution as an ARexx command. The
command string contains the file name of the ARexx script
to be started. If the filehandle is not NULL, it will be
used as stdin and stdout for the Rexx script. If it is
NULL, the Rexx program will use stdin/stdout of the
calling process.

If necessary, the default extension (defined in the
generated header file under the name REXX_EXTENSION) will
be added to the file name.

Messages sent using this function will be replied to by
the ARexx master process as soon as the execution of the
command script stops. The application MUST NOT close
it's messageport before all replies have been received!
To simplify things, ARexxBox does this book-keeping for you.

              CloseDownARexxHost()
               will wait for all missing replies to
arrive before closing down the messageport.

The dispatcher will automagically detect any replies,
count them and do a
              FreeRexxCommand()
               for each reply, so
you don't have to bother with this either.

 INPUTS
rexxhost - the RexxHost to be used to send the command
command - the file name of the ARexx script
filehandle - Filehandle for stdin/stdout or NULL

 RESULTS
rexxmsg - the sent rexx message structure (for comparisons)

 SEE ALSO

              FreeRexxCommand()
              ,
              CloseDownARexxHost()
              ,
              ARexxDispatch()
```

1.11 ARexxBox/SetupARexxHost

```
                  NAME
SetupARexxHost -- initialize and open an ARexx host

 SYNOPSIS
rexxhost = SetupARexxHost( basename );

struct RexxHost *SetupARexxHost( char * );

 FUNCTION
This function allocates and initializes a RexxHost
structure. It opens a public message port under the
given basename. If no basename (NULL) was specified,
the default basename as entered in the ARexxBox window
will be used instead.

Anyway, if a public port of that name already exists,
SetupARexxHost() will start adding numbers to the name
until a unique name is found. So if for example the
basename is "myhost" and there is already a port of that
name in the system, the name will be changed to
"myhost.1" (then to "myhost.2" and so on).

The actual name will be copied to the portname field of
the RexxHost structure. It is a good idea to tell the
user about the actual port name of the new host.

 INPUTS
basename - the messageport basename or NULL

 RESULTS
rexxhost - the initialized RexxHost structure, ready to
     go. Pass this pointer to
               CloseDownARexxHost()
               ,

               ARexxDispatch()
                and
               SendRexxCommand()
               .

 SEE ALSO

               CloseDownARexxHost()
               ,
               ARexxDispatch()
               ,
               SendRexxCommand()
```