rxlibnet_eng_guide

COLLABORATORS

	TITLE : rxlibnet_eng_guide	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 31, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

rxlibnet_eng_guide

1.1 RxLibnet 30.0

```
RxLibnet 30.0 © 2000, 2001 Alfonso Ranieri

1. Introduction
2. Author
3. WRID
4. Terms
5. Structures
6. Functions
7. Note
```

1.2 RxLibnet Introduction

```
Introduction


  This library contains net utility and stack specific functions.

  This library uses rxsocket.library API so it needs rxsocket.library
  installed.

  The environment is macro-private: each macro opens bsdsocket.library
  and what else must be private (e.g. miami.library) and stores a list
  of "things" that must be freed on exit.

  See RxSocket documentation for more info.
```

1.3 RxLibnet Author

```
Author


  I am Alfonso Ranieri
```

```
    My e-mail address is alforan@tin.it

    My home page is at http://web.tiscalinet.it/amiga/"
```

1.4 RxLibnet Warning, Requirements, Installation and Distribution

```
  Warning, Requirements, Installation and Distribution


    Warning
        THIS SOFTWARE AND INFORMATION ARE PROVIDED AS IS.
        ALL USE IS AT YOUR OWN RISK, AND NO LIABILITY OR
        RESPONSIBILITY IS ASSUMED. NO WARRANTY IS MADE,

    Requirements
        The library needs:
            o AmigaOS, version 2 or higher
            o a TCP/IP stack.
            o RxSocket version 9 or higher

    Installation
        o Run the installer script.
        o The library dispatch offset is -30

    Distribution
        rxlibnet.library is Freeware

        You are free to distribute it as long as the original archive
        is kept intact. Commercial use or its inclusion in other
        software package is prohibited without prior written consent
        from the Author.
```

1.5 RxLibnet Terms

```
  Terms


    The main terms used are:

        o stem or stemName
          a valid ARexx variable name e.g. var var.0 var.name

        o socket
          the named space created by socket()

        o socketfd
          the socket descriptor id as an integral value

        o addr or address
          an Internet address in dotted form. An Internet address is
          a 32 bits unsigned long, represented in the "dotted" form
          as "a.b.c.d" "a.b.c" "a.b" "a" or as a symbolic name.
```

```
  In RxSocket addresses are passed/returned in dotted form,
  e.g. resolve() return the dotted form of its argument.

o types of arguments: the types used are:

  D   any data            --
  N   numeric             /N      ARexx integral number
  S   symbol              /S      ARexx valid symbol
  V   stemName            /V      As S but with length<20
```

1.6 RxLibnet Structures

```
Structures


  The main structures passed to or returned from functions are:

    o group set by GetGRGID() and GetGRNam()
        . NAME
        . PASSWD
        . GID
        . MEMBERS.0, ... ,MEMBERS.last (last = MEMBERS.NUM-1)
        . MEMBERS.NUM

    o passwd set by GetPWNam() and  GetPWUID()
        . NAME
        . PASSWD
        . UID
        . GID
        . GECOS
        . DIR
        . SHELL

    o sockaddr_in needed by MiamiSetSocksConn()
        . ADDRFAMILY
        . ADDRADDR
        . ADDRPORT
```

1.7 RxLibnet Functions

```
Functions


  Usergroup
   crypt
   GetPass
   GetGRGID
   GetGRNAM
   GetPWNAM
   GetPWUID
   GetSalt
```

```
Low level
 CreateICMP
 CreateIP
 CreateTCP
 CreateUDP
 ReadICMP
 ReadIP
 ReadTCP
 ReadUDP


Miami
 MiamiClosePF
 MiamiCreatePF
 MiamiDisallowDNS
 MiamiGetPid
 MiamiIFIndexToName
 MiamiIFNameToIndex
 MiamiIsOnline
 MiamiOnOffLine
 MiamiPCapCompile
 MiamiPCapMatch
 MiamiPFNext
 MiamiSetSocksConn
 MiamiSupportsIPV6
 SockatMark


Genesis
 GenesisGetGlobalUser
 GenesisGetUser
 GenesisGetUserName
 GenesisIsOnLine
 GenesisReloadUserList
 GenesisSetGlobaLuser


Encoding
 DecodeB64
 EncodeB64
 MD5
 URLEncode
 URLDecode


Various
 CloseRxLibnet
 help
 ParseURI
```

1.8 RxLibnet Functions - CloseRxLibnet

```
CloseRxLibnet - closes local structures
```

```
Synopsis
  res = CloseRxLibnet()
  -

Function
  When the TCP/IP stack is closed, it sends a ctrl_c to all the
  processes that are using its libraries. It means that even if
  your macros closed all the sockets or all the Miami packet
  filters it used, it may receive a ctrl_c and so be compelled
  to exit. To prevent that, you may use this function. It will
  try to close all the libraries bases, so that you may go on
  without any problem.

Result
  res - an ARexx boolean that indicates if all the libraries
        base were closes.
```

1.9 RxLibnet Functions - CreateICMP

```
CreateICMP - create an ICMP packets

Synopsis
  icmp = CreateICMP(stem)
  <stem/V>

Function
  Creates and returns an icmp header reading its fields from stem.
  The fields are:
      o DATA          the data of the icmp packet, needed to compute its  ↩
         checksum
      o TYPE          icmp type
      o CODE          type/code
      o PPTR
      o GWADDR
      o ID
      o SEQ
      o VOID
      o PMVOID
      o NEXTMTU
      o NUMADDRS
      o WPA
      o LIFETIME
      o OTIME
      o RTIME
      o TTIME
      o IP
      o RADV
      o MASK

Inputs
  stem - an ARexx stem name

Result
  icmp - the icmp packet
```

1.10 RxLibnet Functions - CreateIP

```
CreateIP - creates an IP packet

Synopsis
  ip = CreateIP(stem)
  <stem/V>

Function
  Creates and returns an ip header reading its fields from stem.
  The fields are:
    o V         default 4
    o HL        default 5
    o TOS       default 0
    o LEN       default 20
    o ID
    o OFF
    o TTL       default IPDEFTTL
    o P
    o SRC       source ip addr in dotted form
    o DST       dest ip addr in dotted form

Inputs
  stem - an ARexx stem name

Result
  ip - the ip packet
```

1.11 RxLibnet Functions - CreateTCP

```
CreateTCP - creates a TCP packet

Synopsis
  tcp = CreateTCP(stem)
  <stem/V>

Function
  Creates and returns a tcp header reading its fields from stem.
  The fields are:
    o DATA     needed to compute the tcp checksum
    o SPORT
    o DPORT
    o SEQ
    o ACK
    o OFF
    o FLAGS
    o WIN
    o URP
    o SRC      source addr in dotted form, needed to compute tcp checksum
    o DST      dest addr in dotted form, needed to compute tcp checksum

Inputs
  stem - an ARexx stem name
```

```
Result
  tcp - the tcp packet
```

1.12 RxLibnet Functions - CreateUDP

```
CreateUDP - creates an UDP packet

Synopsis
  udp = CreateUDP(stem)
  <stem/V>

Function
  Creates and returns an udp header reading its fields from stem.
  The fields are:
      o DATA    needed to compute the udp checksum
      o SPORT
      o DPORT
      o SRC     source addr in dotted form, needed to compute tcp checksum
      o DST     dest addr in dotted form, needed to compute tcp checksum

Inputs
  stem - an ARexx stem name

Result
  udp - the udp packet
```

1.13 RxLibnet Functions - crypt

```
crypt - performs password encryption.

Synopsis
  cpasswd=crypt(passwd,set)
  <passwd>,<set>

Function
  The crypt function performs password encryption.
  The algorithm used for encryption is implementation-dependent.

  The first argument is the clear password, the second is a salt,
  that can be created via GetSalt.

  Refer to usergoup.doc .

Inputs
  passwd - the password
  set - the salt

Result
  cpasswd - the password crypted

Example
```

```
  o to generate a password from an user/pass:
      passwd=crypt(pass,GetSalt(user))

  o expression to verify a user/pass login:
      cript(pass,passwd)==passwd
```

1.14 RxLibnet Functions - GenesisGetGlobalUser

```
GenesisGetGlobalUser - gets Genesis global user

Synopsis
  res=GenesisGetGlobalUser(stem)
  <stem/V>

Function
  Writes in stem the Genesis global user, if any.
  The fields set are:
      o NAME
      o PASSWD
      o UID
      o GID
      o GECOS
      o DIR
      o SHELL
      o FLAGS
      o MAXTIME
      o TIMESERVER    (not yet supported)

Inputs
  stem - an ARexx stem name

Result
  res - an ARexx boolean

Note
  This function works iff Genesis is installed.
```

1.15 RxLibnet Functions - GenesisGetUser

```
GenesisGetUser - gets a Genesis user

Synopsis
  res=GenesisGetUser(stem,login,passwd,title,flags)
  <stem/V>,[login],[passwd],[title],[flags/N]

Function
  Writes in stem an user from Genesis database.
  Open an "user request window" if the user must be identified.
  The fields set are:
      o NAME
      o PASSWD
      o UID
```

```
        o GID
        o GECOS
        o DIR
        o SHELL
        o FLAGS
        o MAXTIME
        o TIMESERVER    (not yet supported)

  Inputs
    stem - an ARexx stem name

  Result
    res - an ARexx boolean

  Note
    This function works iff Genesis is installed.
```

1.16 RxLibnet Functions - GenesisGetUserName

```
  GenesisGetUserName - returns a Genesis user

  Synopsis
    res=GenesisGetUserName(userNumber)
    <userNumber/N>

  Function
    Returns the name of the user number userNumber if it exists.

  Inputs
    userNumber - the number of the user

  Result
    res - an ARexx boolean

  Note
    This function works iff Genesis is installed.
```

1.17 RxLibnet Functions - GenesisIsOnLine

```
  GenesisIsOnLine - controls Genesis online status

  Synopsis
    res=GenesisIsOnLine(flags)
    [flags]

  Function
    Controls Genesis online status.

    Flags can be one of:
        o ASKUSER
        o FORCE
```

```
Inputs
  flags - see above

Result
  res - an ARexx boolean

Note
  This function works iff Genesis is installed.
```

1.18 RxLibnet Functions - GenesisReloadUserList

```
GenesisReloadUserList - instructs Genesis to reload users list

Synopsis
  res=GenesisReloadUserList()
  -

Function
  Instructs Genesis to reload the users list.
  That operation should be performed after an
  application modified the users database.


Inputs
  none

Result
  none

Note
  This function works iff Genesis is installed.
```

1.19 RxLibnet Functions - GenesisSetGlobalUser

```
GenesisSetGlobalUser - Logins a new user

Synopsis
  res=GenesisSetGlobalUser(stem,login,passwd,title,flags)
  <stem/V>,[login],[passwd],[title],[flags]

Function
  Logins a new user and sets stem with the user fields.

  The fields set are:
      o NAME
      o PASSWD
      o UID
      o GID
      o GECOS
      o DIR
      o SHELL
      o FLAGS
```

```
        o MAXTIME
        o TIMESERVER    (not yet supported)

    Flags is one or more of:
        o ASKUSER
        o FORCE
        o STAYOPEN

    STAYOPEN
        Many applications use genesis.library to find out which user
        is currently logged.
        E.g. YAM uses Genesis current logged user, if any.
        If you are using a different tcp/ip stack, you can still use
        this Genesis feature with this flags.
        It tells rxlibnet to stay opened so that also genesis.library
        remains opened and the user logged, till you unlog using this
        function with an empty user name.
        Of course, all that has sense only for non-Genesis user.
        See GenesisLogin example in the examples drawer.

  Inputs
    stem - an ARexx stem name
    login - the name of the user to login
    passwd - the password of the user
    title - the title of the requester
    flags - see above
```

1.20 RxLibnet Functions - GetGRGID

```
  GetGRGID - searches for a group by ID

  Synopsis
    res=GetGRGID(GID,group)
    <GID/N>,<group/V>

  Function
    Searches the group database for the given group id, stopping
    at the first found. Fills "group" with a group structure.

  Inputs
    GID - the group id to search
    group - an ARexx stem name

  Result
    res - an ARexx boolean
```

1.21 RxLibnet Functions - GetGRNam

```
  GetGRNam - searches for a group by name

  Synopsis
    res=GetGRNam(name,group)
```

```
    <name>,<group/V>

  Function
    Searches the group database for the given group name, stopping
    at the first found. Fills "group" with a group structure.

  Inputs
    name - the group name to search
    group - an ARexx stem name

  Result
    res - an ARexx boolean
```

1.22 RxLibnet Functions - GetPass

```
  GetPass - requests a password

  Synopsis
    res=GetPass(prompt)
    <prompt>

  Function
    Displays a prompt and read in a password.

  Inputs
    prompt - the prompt to show

  Result
    res - the password the user entered
```

1.23 RxLibnet Functions - GetPWNam

```
  GetPWNam - searches for an user by name

  Synopsis
    res=GetPWNam(name,pass)
    <name>,<pass/V>

  Function
    Searches the user database for the given name, stopping
    at the first found. Fills "pass" with a passwd structure.

  Inputs
    name - the user name to search
    pass - an ARexx stem name
```

1.24 RxLibnet Functions - GetPWUID

```
GetPWUID - search an user by ID

Synopsis
  res=GetPWUID(UID,pass)
  <UID/N>,<pass/V>

Function
  Searches the user database for the given UID, stopping
  at the first found. Fills "pass" with a passwd structure.

Inputs
  UID - the user ID to search
  pass - an ARexx stem name
```

1.25 RxLibnet Functions - GetSalt

```
GetSalt - computes a salt string

Synopsis
  salt=GetSalt(user)
  <user>

Function
  GetSalt creates a text string that is suitable to be passed to
  crypt() as a settings string.

Inputs
  users - the user name

Result
  salt - the salt to pass to crypt()
```

1.26 RxLibnet Functions - help

```
help - returns rxlibnet.library functions strings

Synopsis
  help=help(funName)
  <funName>

Function
  Returns the arguments mask string of rxlibnet.library function "funName".

Inputs
  funName - a rxlibet.library function name

Result
  help - the hep string
```

1.27 RxLibnet Functions - MiamiClosePF

```
MiamiClosePF - closes a packet filter

Synopsis
  call MiamiClosePF(pfID)
  <pfID/N>

Function
  Closes a packet filter created with CreatPF .

Inputs
  pfID - the packet filter ID

Result
  none

Note
  This function works only if Miami is running.

See
  MiamiCreatePF MiamiPFNext
```

1.28 RxLibnet Functions - MiamiCreatePF

```
MiamiCreatePF - creates a packet filter

Synopsis
  pfID =  MiamiCreatePF(dev,signal,maxPks)
  <dev>,<signal/N>,[maxPks/N]

Function
  Creates a packet filter and returns its id .
  A packet filter will receive every in-out coming packet on
  the interface "dev" .

  "signal" is a rmh.library/AllocSignal allocated signal, that
  will be set any time a packet is received .

  maxPks is the max number of packets to store; it must be an
  integer greater than 1 .

Inputs
  dev - an interface name
  signal - the signal to use
  maPks - max number of packets to store

Result
  pfID - an integer:
              o -1 error
              o the ID of the filter otherwise

Example
  pf.rexx
```

```
Note
  This function works only if Miami is running.

See
  MiamiClosePF MiamiPFNext
```

1.29 RxLibnet Functions - MiamiDisallowDns

```
MiamiDisallowDns - controls extern dns-lookup

Synopsis
  call MiamiDisallowDns(1|0)
  [status/N]

Function
  Controls extern DNS lookup.
  If status is 0 extern dns-lookup is disabled.
  If status is 1 extern dns-lookup is enabled.
  Default value for status is 0.

Inputs
  status - the status

Result
  none

Note
  This function works only if Miami is running.
```

1.30 RxLibnet Functions - MiamiGetPid

```
MiamiGetPid - returns Miami's internal process descriptor

Synopsis
  pid = MiamiGetPid()
  -

Function
  Returns Miami's internal process descriptor as packed chars.
  This value is needed when you want to manipulates routes
  directly (see ip2if.rexx).

  If you need just a Process ID (e.g. like in icmp echo packets)
  use pragma("ID") .

Inputs
  none

Result
  pif - the ID
```

```
Note
  This function works only if Miami is running.
```

1.31 RxLibnet Functions - MiamiIFIndexToName

```
MiamiIFIndexToName - returns an interface ID from an interface name

Synopsis
  ifname = MiamiIFIndexToName(index)
  <index/N>

Function
  Returns an interface name from an interface index.

Inputs
  index - an interface index

Result
  ifname - an interface name or an empty string if index
           is not valid
Note
  This function works only if Miami is running.
```

1.32 RxLibnet Functions - MiamiIFNameToIndex

```
MiamiIFNameToIndex - returns an interface name from an interface ID

Synopsis
  ifindex = MiamiIFNameToIndex(name)
  <name>

Function
  Returns an interface index from an interface name .

Inputs
  name - an interface name

Result
  ifindex - an interface index or -1 if name is not valid

Note
  This function works only if Miami is running.
```

1.33 RxLibnet Functions - MiamiOnOffLine

```
MiamiOnOffLine - controls Miami online status

Synopsis
  call MiamiOnOffLine(interface,status)
  <interface>,[status/N]
```

```
Function
  Switch the status of the interface.

  If status is 0 the interface is put offline.
  If status is 1 the interface is put online.
  Default value for status is 0.

  The functions doesn't wait for the switching to
  complete and always returns 1.

Inputs
  interface - an interface name
  status - the status to put the interface

Result
  none

Note
  This function works only if Miami is running.
```

1.34 RxLibnet Functions - MiamiIsOnLine

```
MiamiIsOnLine - checks the status of an interface

Synopsis
  res=MiamiIsOnLine(interface)
  <interface>

Function
  Checks if the given interface is online.

Inputs
  interface - an interface name

Result
  res - an ARexx boolean

Note
  This function works only if Miami is running.
```

1.35 RxLibnet Functions - MiamiPFNext

```
MiamiPFNext - returns the next packet from a packet filter queue

Synopsis
  pkt = MiamiPFNext(pfID)
  <pfID>

Function
  Gets the next packet on a packet filter queue.
  pfID is a packet filter id.
```

```
Inputs
  pfID - a packet filter ID

Result
  pkt - the packet or an empty string

Note
  This function works only if Miami is running.

See
  MiamiClosePF MiamiCreatePF
```

1.36 RxLibnet Functions - MiamiPCapCompile

```
MiamiPCapCompile - compiles a pcap expression

Synopsis
  filter = MiamiPCapCompile(expr,interface,prom)
  <expr>,[interface],[prom/]

Function
  Compiles the pcap expression for the specified interface,
  or the "suitable" one if any, with promiscuous set if
  specified.

  Returns a string that can be used with MiamiPCapMatch() or Null()
  for failure; the reason of the failure can be found in "PACAPERR" .
  The filter can be freed with DROP .

Inputs
  expr - the pcap expression
  interface - the interface to compile expr for
  prom - promiscuous flag

Result
  filter - the compiled expression or null() for failure

Note
  This function works only if Miami is running.

See
  MiamiPCapMatch
```

1.37 RxLibnet Functions - MiamiPCapMatch

```
MiamiPCapMatch - matches a compiled pcap expression on a packet

Synopsis
  res = MiamiPCapMatch(filter,packet)
  <filter>,<packet>
```

```
Function
  Matches a filter created with MiamiPCapCompile() with a
  packet returned by MiamiPFNext() (or what else).

Input
  filter - a compiled pcap expression
  packet - the packet

Result
  res - an ARexx boolean

Note
  This function works only if Miami is running.

See
  MiamiPCapCompile
```

1.38 RxLibnet Functions - MiamiSetSocksConn

```
MiamiSetSocksConn - set the dest addr for the next bind

Synopsis
  res=MiamiSetSocksConn(remote)
  <remote/V>

Function
  Sets the destination address for the next SOCKS bind() request.

Inputs
  remote - an ARexx stem name set as a sockaddr_in

Result
  res - an ARexx boolean

Note
  This function works only if Miami is running.
```

1.39 RxLibnet Functions - MiamiSupportsIPV6

```
MiamiSupportsIPV6 - checks if Miami supports IPv6

Synopsis
  res=MiamiSupportsIPV6()
  --

Function
  Checks if the running version of Miami supports the IPv6 protocol.

Inputs
  none

Result
```

```
  res - an ARexx boolean.

Note
  This function works only if Miami is running.
```

1.40 RxLibnet Functions - ParseURI

```
ParseURI - parses a URI

Synopsis
  res=ParseURI(uri,stem)
  <uri>,<stem/V>

Function
  Parses the URI 'uri' and writes in 'stem' the fields:
      o Scheme
      o Hostinfo
      o User
      o Password
      o Hostname
      o Port
      o Path
      o Query
      o Fragment

Inputs
  uri  - the uri to parse
  stem - where to write the fields

Result
  res - an ARexx boolean

Note
  This function is really primitive, parse only http scheme
  and it is supposed to grow in the future. Anyway, the
  fields used will remain valid.
```

1.41 RxLibnet Functions - ReadICMP

```
ReadICMP - parses an ICMP packet

Synopsis
  call ReadICMP(pkt,stem)
  <pkt>,<stem/V>

Function
  Fills stem with an icmp header read from pkt.
  pkt must at least 28 bytes or an error 18 is generated.

  The fields set are:
      o TYPE
      o CODE
```

```
      o CKSUM
   (Yessss too lazi to make a better icmp parser  :)

Inputs
  pkt - the packet
  stem - an ARexx stem name

Result
  none
```

1.42 RxLibnet Functions - ReadIP

```
ReadIP - parses an ip packet

Synopsis
  call ReadIP(pkt,stem)
  <pkt>,<stem/V>

Function
  Fills stem with an ip header read from pkt.
  pkt must at least 20 bytes or an error 18 is generated.

  The fields set are:
      o V
      o HL
      o TOS
      o LEN
      o ID
      o OFF
      o TTL
      o P
      o SUM
      o SRC in dotted form
      o DST in dotted form

Inputs
  pkt - the packet
  stem - an ARexx stem name

Result
  none
```

1.43 RxLibnet Functions - ReadTCP

```
ReadTCP - parses a TCP packet

Synopsis
  call ReadTCP(pkt,stem)
  <pkt>,<stem/V>

Function
  Fills stem with a tcp header read from pkt.
```

```
  pkt must at least 20 bytes or an error 18 is generated.

  The fields set are:
    o SPORT
    o DPORT
    o SEQ
    o ACK
    o OFF
    o X2
    o FLAGS
    o WIN
    o SUM
    o URP

 Inputs
   pkt - the packet
   stem - an ARexx stem name

 Result
   none
```

1.44 RxLibnet Functions - ReadUDP

```
ReadUDP - parses an UDP packet

Synopsis
  call ReadUDP(pkt,stem)
  <pkt>,<stem/V>

Function
  Fills stem with an udp header read from pkt.
  pkt must at least 8 bytes or an error 18 is generated.

  The fields set are:
      o SPORT
      o DPORT
      o ULEN
      o SUM

Inputs
  pkt - the packet
  stem - an ARexx stem name

Result
  none
```

1.45 RxLibnet Functions - SockAtMark

```
SockAtMark - checks if a socket is in OOB status

Synopsis
  res=SockAtMark(socketfd)
```

```
  <socketfd/N>

Function
  Checks if the socket is in out of band status.

Inputs
  socketfd - the socket to check

Result
  res - an ARexx boolean

Note
  This function works only if Miami is running.
```

1.46 RxLibnet Functions - DecodeB64

```
DecodeB64 - decode base64 data

Synopsis
  res=DecodeB64(source,dest,opt)
  <source>,<dest>,[opt]

Function
  Decodes a base64 encoded stream of data.

  source may be a string or a file name.
  dest may be an ARexx var name or a file name.

  opt is one of:
      o STRING
        source is a string rather than a file name
      o VAR
        dest is an ARexx var name rather than a
        file name
      o NTCHECKERR
        don't check for illegal chars or incomplete
        data during decoding

  If dest is an ARexx var, data size must be < 32767.


Inputs
  source - the sorce of data
  dest - where to store the Result
  opt - options


Result
  res - an ARexx boolean
        On failure, RC contains the reasone:
        1 - AmigaDOS error
        2 - incomplete stream of data
        3 - illegal chars in data
```

```
See
  EncodeB64
```

1.47 RxLibnet Functions - EncodeB64

```
EncodeB64 - encodes data

Synopsis
  res=EncodeB64(source,dest,opt,maxLineLen)
  <source>,<dest>,[opt],[maxLineLen/N]

Function
  Base64 encodes a stream of data.

  source may be a string or a file name.
  dest may be an ARexx var name or a file name.

  opt is one of:
      o STRING
        source is a string rather than a file name
      o VAR
        dest is an ARexx var name rather than a
        file name
      o UNIX
        add a '\n' at end of lines rather than a
        '\r\n'

  If dest is an ARexx var, data size must be < 32767.


Inputs
  source - the sorce of data
  dest - where to store the Result
  opt - options
  maxLineLen - if dest is a file (VAR not specified
               in opt), lines are cut every 72 chars
               by default; maxLineLen specifies a
               new value. If 0 lines cutting is
               suppressed. It must be a non negative
               integral value.

Result
  res - an ARexx boolean
        On failure, RC contains the reasone:
        1 - AmigaDOS error

See
  DecodeB64
```

1.48 RxLibnet Functions - URLEncode

```
URLEncode- RFC 1738 data encoding

Synopsis
  res=URLEncode(url)
  <url>

Function
  Encodes 'url' based on RFC 1738 rules.
  E.g.
      "http://www.serchit.com?search=< Amiga >" --->
      "http://www.serchit.com?search=%3C%20Amiga%20%3E"


Inputs
  url - the string to encode


Result
  res - 'url' encoded


See
  URLDecode
```

1.49 RxLibnet Functions - URLDecode

```
URLDecode- RFC 1738 data decoding

Synopsis
  res=URLDecode(url)
  <url>

Function
  Decodes 'url' based on RFC 1738 rules.
  E.g.
      "http://www.serchit.com?search=%3C%20Amiga%20%3E"  --->
      "http://www.serchit.com?search=< Amiga >"


Inputs
  url - the string to decode


Result
  res - 'url' decoded


See
  URLEncode
```

1.50 RxLibnet Functions - MD5

```
MD5 - Computes the MD5 Message-Digest

Synopsis
  dig=MD5(source,opt)
  <source>,[opt]

Function
  Computes the MD5 Message-Digest for 'source'.

  'opt' may be one of:
      o STRING - 'source' is a string (default)
      o FILE   - 'source' is a file name


Inputs
  source - the source of data
  opt - options regarding 'source' type


Result
  dig - the MD5 digest string for 'source'
        or an empty string if 'source' was
        a file, that can't be opened
```

1.51 RxLibnet Note

```
Note


  1. Pointers to deallocate the local environment in the library
     base is saved in the fields pr_ExitCode and pr_ExitData of the
     Process structure of the macro. At exit a chain of
     pr_ExitCode(pr_ExitData) is called. Details are avaible on
     request.

  2. Some functions are avaible only if a peculiar stack is running
     or installed:

       o  Miami functions are avaible only if Miami is running;
       o  Miami packets filter functions are avaible only if a
          registered version of Miami is running;
       o  Genesis functions are avaible only if Genesis is installed;
       o  usergroup functions are avaible only if the stack running
          offers the usergroup.library .

       When a function is not avaible, the user is informed via a
       requester and an ARexx error 15 (function not found) is
       returned.

       Miami-registered-only functions returns error if used with
       Miami not registered.

       rxsocket.library/IsLibOn() can be used to test the environment.
```