LinPac - Packet Radio Terminal for Linux

Version 0.16

(c) 1998 - 2002 by Radek Burget OK2JBG

User manual

Contents

1 What is LinPac

LinPac is an attempt to create the packet radio terminal for Linux that allows wide configurability and easy addition of new functions and special functions needed by the user. The aim was to minimize the amount of 'hard coded' functions and create the complete set of applications that can be easy expanded and/or completely reconfigured.

All functions described in this manual agree with the standard configuration that comes with the distribution package.

2 First configuration

When linpac is started for the first time, it automaticaly creates the directory called `LinPac` in your home directory. This directory contains your personal LinPac configuration. When creating this directory LinPac asks for basic information and prepares a useable configuration for you.

LinPac functions are based on the very simple interpreted language (actually it's not a language, it's something like the batch files in DOS). In the LinPac home directory there is the subdirectory 'macro', that contains all the scripts written in this language, let's call them macros. Each macro can be run in LinPac and it implements some action. The simplest case of the macro is normal text file, that is just printed to the screen (or sent to the peer) when executed. The language is described in chapter 6.

There is a macro called `init.mac` in the `macro` directory. This macro is executed each time LinPac is started and contains the commands to setup the callsigns and other settings. You can modify this file to change the initial configuration of LinPac.

You may also want to change following files:

ctext.mac - this file is executed when the peer connects and should contain the greeting text
quit.mac - this is called when user enters the Quit command (end of connection). It should print some farewell and disconnect.
info.mac - contains the information about your system

After this you should be able to make your first connection.

3 LinPac controls

After running LinPac the main screen apears. In the standard configuraion it's divided to five parts (described from top of the screen to bottom):

- **QSO window**: this window contains the text that came from the peer and also the sent text and some special messages (different text colours).
- **Status line**: displays your callsign, current time and some information about the current connection.
- **Editor**: allows you to enter the text you want to send to the peer or the commands you want to execute
- **Channel list**: displays the list of channels whith the callsign of connected users. The currently selected channel is highlighted.
- **Monitor** : displays the traffic on your radio ports

LinPac is mostly driven by commands. The commands are entered using the editor and must start with the colon (':') in the first column. Lines that don't begin with the colon are sent to the peer.

LinPac allows to make eight connections simultaneously. For each connection one channel is used. You can switch between channels by pressing the F1 - F8 keys. Each has its own QSO window, status line and the editor. The channel list and the monitor are common for all the channles.

There is a special channel invoked by presing F10. This channel has no QSO window and doesn't allow to make a connection. The text written is sent out immediately using UI frames (beacon).

3.1 Keyboard

Following list contains the important shortcuts:

Global
F1 - F8 : switch to channel 1 - 8
F10 : switch to monitor
Alt-X : end of LinPac

QSO Window
PgUp, **PgDn**, **ctrl-R**, **ctrl-V** : scroll one page up / down
ctrl-E, **ctrl-X** : scroll one line up / down
ctrl-B : skip to end of buffer

Editor
Cursor keys, **Home**, **End**, **Backspace**, **Del** : text editing
ctrl-Y : delete current line
Enter : send current line

Some applications (e.g. mailer) can use the whole screen. On each channel can run only one such application at the same time. It's possible to switch to this application using **Alt-<*channel_number*>** (e.g. Alt-5) and switch back to terminal using **F1 - F10**.

3.2 Entering commands

Each command is called using its name. Some commands can be abbreviated. In this manual the mandatory part is always written by capital letters. The rest is optional. Some commands require some extra arguments. The arguments are written behind the command and are seperated by one or more spaces. If you want to enter the argument containing more than one word, the argument must be entered in quotation marks.

Example:
`:color red blue` - calls the 'color' command with two arguments - 'red' and 'blue'
`:color 'light red'` or
`:color "light red"` - calls the 'color' command with one argument 'light red'

Most of the command work on the currently selected channel. If you want to execute the command on some other channel, you can enter the number of the channel this way:

```
:connect@5 OK0PAB
```

In this case the 'connect' command is executed on channel 5.

The complete list of commands with descriptions is available in Appendix A.

3.3 Initiating the connection

To initiate the connection the `:Connect` command is used. Just switch to the channel you want to use by pressing **F1** - **F8** and enter the following command:

`:connect` *`CALLSIGN`*

Replace the *CALLSIGN* with the real callsign of the station you want to connect. The command can be abbteviated to the first leter only. Example:

```
:c OK0PAB
```

This command will initiate the connecting sequence to OK0PAB. For closing the connection, you can use the `:Disconnect` command by entering

```
:d
```

When your system has multiple radio ports, you can specify its name before the callsign like this:

```
:c 2:OK0PAB
```

This command will try to connect OK0PAB via port 2. When no port name is used, the default one is considered. At the begining the default port is the first port in the `/etc/axports` (your system AX.25 port configuration file). If you want to change the default port, just use the command `:port`.

```
:port 2
```

This will change the default port name to '2'. In some cases it is useful to set another default port for some selected channels. For this the variable `CHN_PORT` can be used (see chapter 4). When set, the content of this variable overrides the default port selection for the particular channel. For example, when you set the variable for channel 4 using

```
:set CHN_PORT@4 1
```

the port '1' will be used as the default one for the channel 4. For other channels, the previously set default port will be used.

3.4 Receiving and sending files

The standard distrubution can receive files using plain text or binary transfer and using file transfers protocols YAPP and Autobin. LinPac will automaticaly start to receive the file when the peer begins to send using the YAPP or Autobin protocols. The 7+ files are automaticaly saved too. When you want to save the incomming text you have to use the command

`:Write <filename>`

The incomming text will be saved until you stop the saving using

`:Write off`

For receiving the plain binary file corresponding command `:WBin` can be used. This way of transferring binary files is not recommended, use the autobin or yapp protocol instead.

There are following commands available for sending files:

`:rprg <filename>` - sends the file using the Autobin protocol
`:yput <filename>` - sends the file using the YAPP protocol
`:rbin <filename>` - sends the binary file (no protocol - not recommended)
`:read <filename>` - semds the text file

3.5 Remote commands

LinPac allows the remote user to enter the commands. For remote control all LinPac commands are available but there can be (and should be) some restrictions for each user.

The remote command must start with the `//` sequence. For example if some connected users sends you a text '`//info`' your terminal will send back the station information.

The remote commands can be disabled using the command `:remote off` and enabled by `:remote on`. You can also specify only some commands to be available for remote users. The default list of available remote commands is defined in the **init.mac** file (the DEF_RCMD line). There is also the posibility to enable various commands for each user. This is described in chapter 5.

3.6 Character encoding

In some countries there are used more different national character encodings for some historical reasons. An user who has his Linux console configured for example for some of standard ISO encodings may be incompatible with another one using a traditional encoding. To solve this LinPac allows to translate the input and output of each channel

using a translation table. The translation tables are stored in files `*.ctt` in the LinPac home directory.

All known encodings have to be defined in the file called `encodings` in the LinPac home directory. This file contains a single line for each encoding that specifies its *alias* (name which will identify the encoding in LinPac), *encoding name* (an official name such as iso-8859-1) and optionaly the name of the *translation table file* to be used (without the extension `.ctt`).

Current encoding can be switched using the **:TRanslate** command separately for each channel. To specify the default encoding for each user you can add the line

```
ENC=<alias>
```

to the appropriate record in station database (station database is described in chapter 5). When no encoding is specified for the user, the default one is used. The default encoding alias is stored in the DEF_ENC@0 variable which is set in the macro init.mac.

When the conversion table is not specified in the encodings file LinPac only changes the name of currently used encoding but doesn't provide any conversion. However some applications (such as QLinPac which works in unicode) are able to do their own conversions.

3.7 Huffman compression

Some packet radio terminals and BBS software allows the compression of transferred text. When switched on, the sender does the compression of all data before sending them to the other station and the recipient has to decompress the data after receiving them. This makes the communication more reliable and reduces the load of the radio link.

The line compression in LinPac is activated using the `:comp` command. The compression is switched on using `:comp on` and switched off using `:comp off`. To ensure that the compression is activated or deactivated on both ends of the link simultaneously LinPac sends the remote command `:comp 1` or `:comp 0` to the other station automaticaly. The arguments 1 and 0 have the same effect as the `on` and `off`, but they don't cause sending any command to the other station.

In case that the remote system doesn't support the `:comp` command it's necessary to switch on the compression on the remote system manually and then use the `:comp 1` command in LinPac.

4 Variables

Each channel has its own set of variables. Some of the variables are used to store the configuration data. User can create and remove the variables and change the values of existing variables using following commands:

`:set <variable> <value>` - sets the value of the variable. If the variable doesn't exist, new one is created.
`:get <variable>` - prints the value of the variable
`:unset <variable>` - removes the variable

Some examples:

```
:set NAME John
:set WHOLE_NAME 'John Big'
:get NAME
:unset NAME
```

The name of the variable can contain the specification of the channel. For example the variable `NAME@5` is the variable `'NAME'` defined on channel 5.

When LinPac founds the character `'%'` followed by the name of variable, automaticaly replaces this text with the value of the variable. Considering previous example the text `%NAME` will be replaced with John.

4.1 Special variables

There are some special internal variables that don't allow changing their value. Their value is set and changed directly by LinPac and this variables can be used to add some actual information to the text. The list follows:

`%V` - LinPac version (e.g. 0.02)
`%C` - The callsign of connected station
`%N` - The name of connected station (when known), else is replaced by the contents of %U macro
`%Y`- Channel callsign (mycall)
`%K`- Current channel number
`%T` - Current time
`%D` - Current date
`%B` - Audible bell
`%Z` - Current time zone
`%U` - The text used when the name is unknown. This can contain other macros (typicaly %C).
`%P`- The port number
`%M`- The number of connected users
`%A`- The time since the last operator activity

`%_`- End of line (CR)
`%<`- Contents of the last line received, this is cleared by reading
`%#number` - Replaced by a character with an ASCII value <number> (e.g. %#27 means ESC)
`%(command)` - Replaced by the command result.
`%[expression]` - Replaced by the result of mathematical expression

Variables for use in macros only:

`%R` - the number of macro arguments (up to 9)
`%0` - the name of the macro
`%1` - `%9` - macro arguments

For example try to write following text in the editor and press enter:

```
The time is %T and the date is %D.
```

5 Station database

The station database holds various information about known stations. All the information is stored in the '`station.data`' file and can be changed using the normal text editor or using the LinPac :`Name` command.

5.1 The station.data file format

The information about each station is written in the paragraph starting with the station callsign in the square brackets. Each line in the paragraph contains one defininion like

```
<item_name>=<value>
```

The typical station information can look like this:

```
[OK0NMA]
NAME=PC/FlexNet Brno
TYPE=FLEXNET
LOC=JN89HE
QRG=144.8125 MHz
SYSNUM=85946
```

There are no mandatory items, the user can add various items depending on what information he wants to store. Current LinPac distribution uses following item names for standard information:

NAME - Text information about the station, or the operator's name. LinPac shows this information when connected to that station. LOC - QRA locator of the station. Shown after connect too.

TYPE - The type of the station. For standard stations the types FLEXNET, THENET, FBB, BAYBOX, DIEBOX, TNOS, JNOS, DXC and TERM for user terminals are recomended, but you can add any other type.

LANG - The language to communicate with the station. This is currently supported by the macros only. When this item is set, LinPac will try to find the macro in the directory `macro/<LANG>/`.

NICKNAME - The nickname of operator.

The standard LinPac configuration also uses this item names:

TERM - What type of terminal is used. If '`ansi`' is set, LinPac switches to the ansi-color mode after connecting this station.
ENC - The character encoding. Used to automaticaly switch to the i/o character conversion.
RCMD - The list of enabled remote commands for this station.
QRG - The user frequency. Used by the logbook.
SYSNUM and **PWD** - Sysop password for the station. See chapter 8.2 for more information.

5.2 The 'Name' command

The `:Name` command is used to modify the station database. Running the command without arguments results printing the name of currently connected station. The arguments are used to modify the data:

`<name>` - modify the NAME item
`-l <locator>` - modify the LOC item
`-t <type>` - TYPE
`-L <language>` - LANG
`-n <nickname>` - NICKNAME
`-s <item>=<value>` - modify other item

The command '`Name -i`' prints all information about the station. When you need to change the information about other than connected station add the argument `-c <callsign>`.

Examples:
```
:Name John
:Name -c OK2JBG -l JN89HF Radek
:Name -i
```

5.3 Using the database

After any connection establishes, LinPac reads the information about the connected station from the database and creates the set of variables with names `STN_<database_item_name>` containing the values of the items. This variables can be used in macros as described below.

6 About macros

6.1 Creating macros

The macro is a LinPac command that is created using the macro language and uses other LinPac commands to perform some action. The macro can be defined by creating the file `macro/<command_name>.mac`. It's possible to define the abbreviated form of the command, this is described in chapter 7. There are two ways to define a macro:

a) Text macros
This way is suitable for commands which are intended to produce a larger text output (for example station information). When executing this macro, each line that doesn't start with ':' is printed (sent out). All commands must start with the colon. This is suitable for modifying the text output using the IF ~ ELSE ~ ENDIF commands or for including some other commands.

b) Command macros
A command macro must start with the line
```
:MACRO <name>
```
Each line of a command macro is interpreted as a command (doesn't start with the colon and doesn't need to start at the begining of line). The text output is provided by the 'echo' command. This way is more synoptical and allows including the comments that must start with the sequence `';;'` and end with the end of line.

The macro is called with its name. When the first arguments starts with the '@' symbol the macro is executed from the specified label. For example the command `:convers @SEND` will execute the macro '**convers.mac**' from the label 'SEND' (see next chapter to see how to define the label).

6.2 Commands used in macros

The macro can contain all the LinPac and user defined commands. There are also some special commands that can be used in macros only:

MACRO [name]
Start of the command script definition (see previous section).

LABEL <label_name>
Creates a label with specified name. In the command scripts the notation :<label_name> can be used.

GOTO <label_name>
Jump to specified label.

IF ~ ELSE ~ ENDIF
Conditional commands. There are two ways to specify a condition:

- normal notation (for more than one command)

  IF <condition>
  .
  .
  (commands to be done when the codition is true)
  .
  .
  ELSE
  .
  .
  (commands to be done when the condition is false)
  .
  .
  ENDIF

  The ELSE part is not necessary - the IF ~ ENDIF notation is possible.

- abreviated notation (for one conditional command)

  IF (<condition>) command

  The parentheses are necessary in this case.

Following example shows how to use conditions and how to use the data from station database. We want to create the macro, that will greet the operator of connected station with his nickname or with his name, if the nickname is not defined.

a) The solution using the text macro (the comments are actually not allowed in the text macros, they are here for explanation only)

```
:if %(exist STN_NICKNAME) == 1 ;; when NICKNAME is defined
Hello %STN_NICKNAME            ;; greet with the nickname
:else                          ;; else (not defined)
Hello %N !                     ;; greet with the name
:endif                         ;;(following commands are always
executed)
You have connected to %Y at %T. ;; Say your callsign and current
time
```

b) The solution using the command macro

```
:macro GREETING ;; start the command macro
if %(exist STN_NICKNAME) == 1 ;; when NICKNAME is defined
echo Hello %STN_NICKNAME     ;; greet with the nickname
else                         ;; else (not defined)
echo Hello %N !              ;; greet with the name
endif                        ;; (following commands are always
executed)
echo You have connected to %Y at %T. ;; Say your callsign and
current time
```

6.3 Special system macros

There are some special macros that are executed automaticaly by LinPac in some cases:

init.mac - This is executed when LinPac is started and its function is to set the callsigns, screen options, and some other parametres.

cinit.mac - This is executed always when some connection establishes. The distribution version of this macro sets the channel parametres in order to station settings in station database (allowed remote commands, i/o encoding, terminal type) and executes the logbook command to sign a start of connection. LinPac always passes two arguments to this macro. The first (%1) argument is the callsign of connected station and the second (%2) argument is the callsign of the previously connected station that provides the connection or it's empty in case of direct connection.

ctext.mac - This macro is executed when some station connects to the terminal. It should print some greeting text. No arguments are passed.

cexit.mac - This is executed always when some connection closes. The distribution version of this macro just executes the logbook command to sign the end of the connection and clears the list of allowed remote commands. There is always one argument passed by LinPac (%1) and contains the callsign of the disconnected station.

7 Creating new commands

New command can be represented by the macro or by the external program (standard linux program or special LinPac application). Macros are placed in the `$LINPACDIR/macro` directory and external programs are placed int the `$LINPACDIR/bin` directory. In both of this directories is the file '`commands`' that contains the list of commands in that directory. You should specify here the name of the file, the name of the command in LinPac (use capital letters to specify the mandatory part of the command). It's not necessary to include the macros here, if you don't want to define the abbreviation.
In case of external programs there is also the possibility to specify the program flags. Currently this flags are supported:

A - Ascii mode program. LinPac provide the CR <-> LF conversion when communicating with this program. This is the default setting.
B - Binary mode. Disables the conversions.
C - Leaves the stdout stream of the program on the console and reads its stderr stream instead.
D - DOS conversion - ignore LF characters.
S - Reads both stdin and stderr streams of the program (shell mode).
L - Local. This program is never available as the remote command.
R - This program is always run as the remote command (its messages are always sent out).
P - LinPac removes the paths from filenames that are passed as the argument of this command when the FIXPATH command is on. This is the security option.

8 Standard applications

8.1 File transfer protocols

At present LinPac supports two protocols for transferring files:

- **Autobin** - a simple protocol suitable for short files

- **YAPP** - very sophisticated transfer protocol that provides better error detection and is able to resume previously interrupted transfer

Usage of this protocols is described in chapter 3.4.

LinPac can also automaticaly save incomming 7+ files. After saving all parts of file LinPac tries to call the '7plus' program to decode the file. Received 7+ files are not removed automaticaly.

8.2 Automatic password generation

8.2.1 Login passwords

LinPac provides automatic replies to the login authorization requests for following systems: F6FBB BBS (VE2BLY C_FILTER), BAYBOX, TNOS (numeric MD5). Each station which requests the login password must have an entry in the station database containing at least following fields:

`TYPE=<`*`station_type`*`>` (FBB, BAYBOX or TNOS)
`LOGINPW=<`*`login_password`*`>`
`PWPROMPT=<`*`BBS_prompt`*`>`

`BBS_prompt` is the text which the BBS sends when requesting the authorization. Usually it looks like 'Password>' or 'OK0XYZ>'.

8.2.2 Sysop and general use passwords

LinPac provides autimatic authorization to following systems: F6FBB BBS (MD2 password), FLEXNET (older versions, the 'magic' numbers system and newer TheNet-like system), THENET and BAYBOX. Each station you want authorize to must have anentry in the station database. For password generation following fields must be set:

`TYPE=<`*`station_type`*`>`
`PWD=<`*`your_password`*`>` or
`SYSNUM=<`*`magic_number`*`>`

Known station types are:

- **FBB** - An F6FBB BBS. The PWD field must contain your password.
- **THENET** - A TheNet node. Again the PWD must contain the password.
- **BAYBOX** - The same system as TheNet.
- **FPAC** - A FPAC node. Password must be stored in the PWD field.

- **FLEXNET** - FlexNet node. If the magic number algorithm is used (older versions of flexdigi) the SYSNUM field must contain your magic number and the PWD field mustn't be set. When the TheNet algorithm is used (newer versions of flexdigi), the PWD field must contain the password and the SYSNUM field mustn't be used.

In case of **FBB** the authorization algorithm can be choosed by setting the **MD** field in the station database:
`MD=5` - this value will select the MD5 algorithm
`MD=2` - this value will select the MD2 algorithm

When no value is set, MD2 algorithm is used.

After connecting to the station you want authorize to the authorization sequence begins with the `:PW` command. LinPac will send the authorization command to the station and tries to answer the authorization request using your password. If the password is correct, authorization should finish succesfuly.

The PW command accepts following parametres:

`:PW [<command> [<system> [<password>]]]`

where
<*command*> is the command to be send to the remote station to start authorization sequence (*sys* by default)
<*system*> is one of the above-mentioned supported systems, this system is used instead of the one specified in station database
<*password*> is the password that will be used instead of the one specified in the station database

It's recommended to create simple macros for frequently used authorizations thad require special arguments to PW command.

8.3 Utilities for mail exchange

LinPac contains some utilities for exchanging mail with the F6FBB BBS. For the proper function of this utilities following variables must be created in channel 0:

HOME_BBS - The AX.25 path to the home BBS including the port name. For example `kiss:OK0PAB OK0NMA` is the BBS `OK0PAB` which can be connected on port `kiss` via the node `OK0NMA`.
HOME_ADDR - The full hierarchical address of the BBS. For example `OK0PAB.#MOR.CZE.EU`.

For setting the variable the `:SET` command can be used. For example:

```
:set HOME_BBS@0 "kiss:OK0PAB OK0NMA"
```

The reccomended place to set this variables is the startup macro `init.mac`. The default version of this macro contains an example of setting this variables. After setting this variables following commands are available:

`:GETMSG <message_number> [<message_number> ...]`
Reads specified messages from the BBS and stores them into `/var/ax25/mail/<BBS_callsign>/<message_number>`. The directory for the BBS must be created before using this command (use capital letters for the BBS callsgin). When the message number starts with the letter **'p'**, the message is considered as a personal one and it's killed automaticaly after download. You can specify the kill command by setting the `KILLPERS` variable in channel 0 using the # character for the number of message (e.g. `:set KILLPERS@0 "kill #"`). When this variable is not set, the default command `K #` is used.

`:SP <address>` or `:SB <address>`
This commands can be used for creating new private message or the bulletin. The usage is the same as at the FBB BBS.

`:FORWARD`
Transfers all new messages to the BBS.

`:DELMSG <message_number>`
Marks the message for delete.

`:PACK`
Deletes all marked messages.

8.4 Mail client

This application allows full screen message editing and browsing. It provides the frontend to mail exchange utilities. Mail client is started by the `:MAIL` command. After the program is started the **H** key shows the operating instructions.

8.5 Logbook

Logbook is a simple application that is started from the cinit.mac and cexit.mac scripts (at the begining and at the end of each connection). It creates the file in the 'log' directory for each callsign and writes here the time when the connections were started and finished and

the QRG. The QRG is taken from the `QRG` field of the station entry in station database. If the station has no QRG defined, the value from the `QRG@0` variable is taken.

9 Command line options

LinPac accepts following command line options:

-m : disable monitor. When this option is used, LinPac doesn't create it's internal monitor objects and saves memory.

-s : disable *ax25spyd*. Linpac normaly tries to connect *ax25spyd* to get monitor data and when the connection fails, the *listen* utility is used instead. When **-s** swicth is used, LinPac doesn't try to connect *ax25spyd* at all.

-d : daemon mode. LinPac doesn't initialize the screen and runs in the background.

-p <string> : specify the arguments for the *listen* program. Default value is `ar8`.

10 Copying

LinPac is Copyright (c) 1998-2002 by Radek Burget, OK2JBG

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation;

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details (contained in file 'license').

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Appendix A - LinPac commands

Built-in commands

a) action commands

ABort [address]
Cancels an action. Some commands are cancelled without specifying any address (e.g. autobin). Addresses for some commands:

Action	Address for abort
autobin RX/TX	**autobin** (or none)
yapp RX/TX	**yapp** (or none)
7plus autosave	**7plus** (or none)
read / RBin	**read** (or none)
write / WBin	**write** (or none)
forward	**forward** (mandatory)

The most of other commands doesn't need any address.

COMPress [on | off | 1 | 0]
Switches the link compression on or off. The arguments 1 and 0 have the some meaning as on or off but the remote station is not synchronized.

Connect [port:]call [digi [digi...]]
Initiates a connection to specified station.

Disconnect
Disconnect actual channel.

Echo <text>
Prints (sends) specified text.

FLUSH
Flush an output buffer. (for example in scripts before disconnect or before requestig input)

SYstem
End of LinPac, cancels all connections.

UNproto <text>
Send specified text in an UI frame.

VERsion
Version information.

b) Commands for variables handling

SET <variable> <value>
Assign a value to the variable. If the variable doesn't exist, it is created.

UNSET <variable>
Removes the variable.

GET <variable>
Returns the value of the variable. Usually is better to use macro %variable (see file macros.txt)

EXISTs <variable>
Rerurns 1 when variable exists, 0 when it doesn't exist.

c) information commands

ENVINFO
Displays the actual information about the variable environment.

ISCONnected
Returns "1" when the channel is connected, "0" otherwise.

MAXCHannels
Returns number of LinPac channels (typically 8).

PCALL
Returns the physical callsign of the connected station (first station connected)

UTCTime
Returns actual UTC time (operating system value).

d) setup commands

CBell [on|off]
When on then LinPac gives an audio signal when any station connects or disconnects.

FIXPath [on|off]
When FIXPath=ON then then the paths to files mentioned in parametres are ignored for external commands marked with a P flag. That means only the default paths are usable.

INFOLEvel [0 | 1 | 2]
Sets the mode of information line:
0 - off (no connection info)
1 - show importatnt informations
2 - show all available informations

KNax [on|off]
Enable/disable sound signal when data is received.

Language [language]
Sets actual languge. At the time supported in scripts only.

LIsten [on|off]
When listen is off, all connect requests are ignored by LinPac. Default value is on.

MBIN [on|off]
When switched on, the monitor shows binary data. When switched off, the binary data is hidden and replaced with the <Binary data> information.

MONitor [on|off]
Enable/disable monitor function. This command is usually followed by STATLINE and CHNLINE commands to adjust the screen layout.

MONPort <port_specification>
Monitor selected port only. Port specification is either a port name (as defined in /etc/ax25/axports) or * for monitoring all the ports.

MYcall <call>
Changes the channel callsign.

Port <port_name>
Sets the default port for the Connect command. This is usually the first port defined in the /etc/ax25/axports file but can be overriden here. This can be overriden for particular channels by setting the `CHN_PORT` variable for the channel (see chapter 3.3).

PRIVate [on|off]
Marks the channel as private. No stations are allowed to connect on this channel.

RCMD [<command list>]
Specifies the list of the available external commands. Only the commands from this list are available to remote user. It's possible to include abbreviated commands. The remote commands can be executed on the channel which provides the connection only. Adding the @ character just after the command name in the list (e.g. GET@) means that the remote user is allowed to specify the channel, where the command should be executed (e.g. //GET@5 NAME).

REMote [on|off]
Enables or disables remote commands.

RXFlow [on|off]
Enables or disables data RX. The data is received only when **RXFlow** is enabled on **all** channels.

TIMEZone [zone]
Set the time zone. Used for information only, doesn't affect time.

UNDest [call]
The destination address for UI frames.

UNPort <port_name>
Sets the default port for the UI frames / unproto traffic command or F10 unproto traffic area. This is usually the first port defined in the /etc/ax25/axports file but can be overriden here. This can be overriden for particular channels by setting the unport variable for the channel. HTML

UNSrc [call]
The source callsign for UI frames.

WAtch <port | 0> <pattern> <command/text>
Starts to watch specified port (0 for all ports). When specified pattern is received then specified command is executed or text is sent. (commands must start with a colon)

e) Screen control commands

STATLINE <n>
Places the status line to the n-th line of the screen.

CHNLINE <n>
Places the channel line to the n-th line of the screen.

SWAPEDit
Replaces the editor window with the QSO window and vice versa.

INFOLine <nm> <text>
Changes the specified info line text. If info line doesn't exist, it's created.

REMOVEINFO <nm>
Removes specified info line.

TRanslate <alias>
Switches I/O character translation (see chapter 3.6). Running this command on channel 0 (unproto channel) switches the translation table in all channels including unproto channel and monitor window.

TErm <type>
Set the terminal type. If 'ansi' is entered then ANSI-color control sequences are interpreted.

SCRLimit <low-limit> <high-limit>
When the size of the window buffer exceeds the high limit, then the size of buffer is truncated to low-limit. The values are in bytes, default is 356352 and 524288 (384 and 512 kB).

DEFColor <color_name> <foreground_color> <background_color>
Changes the color of some part of screen. The color_name parameter specifies which part of screen to change. Following values can be used:
QSO_RX - received text in QSO window
QSO_TX - sent text in QSO window
QSO_INFO - local information in QSO window
QSO_CTRL - control characters in QSO window
ED_TEXT - normal text in editor
ED_INFO - command results in editor
ED_CTRL - control characters in editor
CI_NORM - channel info line - channel numbers
CI_SLCT - selected channel
CI_NCHN - normal channel
CI_PRVT - private channel
IL_TEXT - status lines

For specifying foreground and background colors these values can be used:
BLACK, RED, GREEN, BROWN, BLUE, MAGENTA, CYAN, LIGHTGRAY
These additional colors can be used for foreground only:
DARKGRAY, LIGHTRED, LIGHTGREEN, YELLOW, LIGHTBLUE, LIGHTMAGENTA, LIGHTCYAN, WHITE

f) system commands

RESULT <text>
Returns the text as the result of the script.

g) string commands

STRMid <start> <length> <string>
Returns the substring starting at <start> position and <length> characters long.

STRLeft <length> <string>
Returns the left substring of specified length.

STRRight <length> <string>
Returns the right substring of specified length.

STRLen <string>
Returns the length of the string.

STRPos <substring> <string>
Returns the position of the substring in the string or -1 when the string doesn't contain the substring.

UPCASE <string>
Returns the string converted into capital letters.

External commands

Bell
Calls the operator using an acoustic signal.

Compose [p|b] <address> [<subject>] [<filename>]
Create a private message or a bulletin for the specified address. When no subject is specified, the user is prompted for the subject. When filename is specified, the message is created from the file, otherwise the text is read from the LinPac console.

CATCH -iol <pattern> <command/text>
Catch is the extended version of **WAtch** command. It scans one or more data streams (-i = input stream, -o = output stream, -l = local info stream) for the pattern. The pattern can contain ***** and **?** wildcards. The command can contain string `$1` .. `$n` which are replaced with the string corresponding to the n-th wildcard in the pattern. The `$0` string is replaced with the whole string that matches the pattern. See `catch -h` for extended parametres.

DELMSG <message_number>
Mark the message for delete.

FORWARD
Transmits all new messages to a BBS.

GETMsg <numers>
Reads the messages from BBS.

JOIN <channel_number>
Join specified channel to current channel. All data received on any of these channels is automatically redirected to the other channel. Stop with **:ABort join**.

MAIL
Simple full-screen mail client.

MHeard
List of heard stations.

Name
Stores station name or changes a station database. (see Name -h)

PACK
Deletes all messages marked for delete.

Read <filename>
Sends specified text file.

RPRg <filename>
Transmits the file using Autobin protocol.

RTt
Measures the round trip time.

SENDFile [p|b] <file> <address> [<num_messages>]
This command takes a binary file, splits the file to *num_messages* parts using **7plus** and creates a separate message for each part. When *num_messages* is not specified, the command tries to use
the variable `BLOCK7P@0` which should contain the maximal size of one block. If this variable is not set, blocks of 5 kB are created.

WBin / RBin
The same as Read / Write, but for binary files.

Write <filename>
Starts to write received text to the file.

YPUT <filename>
Transmits the file using the YAPP protocol.

Macros

Activity
Shows the time since the last operator activity.

Conv
Enter the conference.

Info
Local station information.

Help
Brief help.

PW [<command> [<system> [<password>]]]
Starts the authorization to the BBS or the node. See chapter 8.2.2 .

Quit
Sends the quit text and disconnects.

Users / CStatus
List of connected users.

Commands for creating scripts

MACRO [name]
Start of the command script definition (see below).

LABEL <label_name>
Creates a label with specified name. In the command scripts the notation :<label_name> can be used.

GOTO <label_name>
Jump to specified label.

IF ~ ELSE ~ ENDIF
Conditional commands. There are two ways to specify a condition:

- normal notation (for more than one command)

  IF <condition>
  .
  .
  (commands to be done when the codition is true)
  .
  .
  ELSE
  .

.
(commands to be done when the condition is false)
.
.
ENDIF

The ELSE part is not necessary - the IF ~ ENDIF notation is possible.

abreviated notation (for one conditional command)

IF (<condition>) command

The parentheses are necessary in this case.

RETURN [<data>]
Abort the macro execution and return the data as a result.

SLEEP <seconds>
Pause the macro for specified time in seconds.

WAITFOR <condition>
Pause the macro until the condition is true.

Last update: 29.7.2002
Please report any mistakes to radkovo@centrum.cz