NAME

akfavatar-term − module to execute terminal based programs inside of AKFAvatar

SYNOPSIS

term = require "akfavatar-term"

Note: You have to use a global variable if you want to use the APC interface.

DESCRIPTION

This module can be used to execute common terminal based programs inside of AKFAvatar. The terminal emulation emulates a Linux textconsole (not only on Linux based systems).

It can also be used to write specific programs for AKFAvatar in otherwise unsupported languages. There are some facilities with which the hosting script and the guest program can communicate with each other.

ATTENTION: This module is not available for all platforms! Especially not for Windows.

term.startdir(*[directory]***)**

Sets the start directory for the next **term.execute()** command. If no directory is given, it clears the previous setting.

Note: The working directory of the host and the guest program are independent!

term.homedir()

Set the start directory for the next **term.execute()** command to the users home directory.

Note: The working directory of the host and the guest program are independent!

term.color(*true*|*false***)**

Switches the color support on or off.
The terminal type with color is "linux", without color it is "linux-m".

term.setenv(*variable, value***)**

Sets the given environment *variable* to the given *value*.
This also affects programs started with **os.execute()** or **io.popen()**.

term.unsetenv(*variable***)**

Unsets (clears) the given environment *variable*.
This also affects programs started with **os.execute()** or **io.popen()**.

term.execute(*[program [,arguments]]***)**

Executes the given *program* with the given *arguments* in the terminal emulation. If no *program* is given it starts the default shell.

APC interface

The "Application Program Command" (APC) interface is a method with which the guest program can communicate with the Lua interpreter of the host program. To use this, the guest program sends Escape sequences to the standard output, like follows: ESC + "_", followed by the command, closed with ESC + "\" to end the sequence. The output has eventually to be flushed to get an immediate effect.

For example in C:

```
static void
APC (const char *s)
{
  fprintf (stdout, "\033_%s\033\\", s);
  fflush (stdout);
}
```

The commands can be Lua function calls, especially the "avt." commands defined by "lua-akfavatar" (see the "lua-akfavatar-ref").
Up to 1024 characters can be sent. So it is not suited to send complicated code from the guest program. However the hosting script, that is the Lua script that calls the guest program, can define its own global functions, which are then also accessible with the APC interface.

As a convention the host script should define an environment variable named *"APC"*, which decribes the accessible commands.

For example:

```
term.setenv("APC", _VERSION .. ", lua-akfavatar")
```

term.send(*string***)**

Send a string to the guest program, as if it is typed from the keyboard. This can only be used from the APC interface (see above).
If the guest program is line-oriented, the string should be closed with a "\r" (for return).

term.decide(*string1 [,string2]***)**

Present plus/minus buttons and depending on that choice sends *string1* or *string2* to the guest program, as if it is typed from the keyboard. This can only be used from the APC interface (see above). If the guest program is line-oriented, both strings should be given and they should be closed with "\r" (for return).

SEE ALSO

lua-akfavatar(1) **lua**(1) **lua-akfavatar-ref**(3) **akfavatar-graphic**(3) **akfavatar.utf8**(3)