AmiMUD

Gabriele Greco

COLLABORATORS			
	TITLE : AmiMUD		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Gabriele Greco	August 7, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmiMUD

1.1 AmiMUD manual

AmiMUD 3.1 by Gabriele Greco

1.2 Introduction

Introduction

AmiMUD is a MUD client. It runs using socket and can be used to play nearly every text-only mud on the internet or on other TCP/IP based networks.

I wrote this because I saw that there wasn't any decent MUD client for amiga, since I enjoy myself playing muds and I don't like using term or the amitcp telnet to do that I decide that writing a MUD client was a good Idea :)

It was thought for DIKU-type muds because that's the kind of mud I play, but can be used with nearly any type of mud without any problem, obviously some of the features (for instance tickcounter) are not useful on some types of mud, but I don't think this is a big problem, if you have suggestions for features you'll like to see in AmiMUD mail me them.

My initial thought was to make it freeware, but I spent many time (and money in phone bills :)) on it so I decide to make it shareware , so if you like it and you use it regularly please register, you will make future improvements being possible.

History of AmiMUD

1.3 AmiMUD Features...

These are some of the features of AmiMUD:

- Works with any TCP stack (tested with AmiTCP, Miami, IW225, Mlink).
- Runs also without TCP stack if needed. (NEW)
- Multithread. You can run multiple connection within a single AmiMUD. (NEW)
- Input and Output on two windows. From 3.0 the two windows are "joined". (NEW)
- Command history buffer on output window.
- Possibility to log to file what you do.
- MSP, Music Sound Protocol support. (NEW) (REGISTERED)
- Can be opened on Workbench, Public screens or on a custom screen...
- Modular output system, you can write your output module if you don't like
the two available modules. (NEW)
- ANSI color support, with the custom ansi module also 16 colors ANSI
(on public screen needs OS 3.0). (NEW)
- Macros (with or without arguments) and with hotkeys (in the standard C= format).
- Button window to recall via mouse the most used macros. (NEW)
- Aliases.
- Numeric Keypad walking (fully configurable).
- Autologin.
- Triggers.
- Variables.
- Math operation and comparison between variables. (NEW)
- Gags.
- Highlights.
- TAB completion. (REGISTERED)
- Many internal commands to use in triggers and macros.
- Dynamical tick length counter (for DIKU muds).
- Nearly all function may be used by menu.
- AmiMUD is not a unix porting!
- Multiple config capability.
- Possibility to edit / use custom palettes.
- Commodity interface.
- Send a file or clipboard to the mud.
- Speedwalking (also in tintin++ mode). (REGISTERED)
- ARexx, the most powerful script language in ANY mud client! (REGISTERED)
- Multiple commands on a line. (REGISTERED)
- Great configurability.
- Very nice MUI configuration program.

- Speedwalk playing/recording (REGISTERED)

- Graphics board support.

- Partial Telnet compatibility.

- Localization.

...and many other features!

1.4 dist

Distribution

AmiMUD is shareware, the shareware version may be distributed everywere through internet, bbs, cd-rom, cover disk provided that the archive is distributed in his original form and that the price of the cd-rom/cover disk isn't too high.

If you are not registered AmiMUD will open a "reminder" requester when you launch the program.

Starting from 1.10 some features of AmiMUD are only available if you register the program. Actually these features are:

speedwalking (1.5), multiple commands on a line (1.10), AREXX port (2.0) and TAB completion (2.5), MSP support (3.0).

How to register

1.5 Requirements

Requirements

AmiMUD needs to work:

- an Amiga with at least OS 2.0 and 1MB of RAM (some features work only with OS 3.0+).

- reqtools.library (included in the archive)

- a TCP stack installed (tested with Miami, AS225, AmiTCP and MLink) and active if

you want to connect to a MUD.

Not required but useful:

- MUI 3.x to use the preferences program (you can find it on aminet: util/libs/MUI38usr.lha)

- the file amimud.prefs to keep all the configuration settings of the program, must be kept in the program directory (also s: is searched for backward compatibility).

- KingCON if you don't like the new custom ANSI module and you want to have a review buffer

in the output window. (you can find KCON on aminet util/shell/kingcon13.lha)

1.6 Installation

Automatic installation

Simply click on the Install_AmiMUD icon in the AmiMUD drawer.

Manual installation

To install amimud simply copy the file AmiMUD (and optionally the icon AmiMUD.info) on your HD (or on a disk).

To use all the features of the program you'll need also to copy the file AmiMUD.prefs to the same directory where AmiMUD is located.

Make a directory called "output" in the directory where you keep AmiMUD and copy the output modules you need in that directory.

If it does exists a directory <yourlanguage> it means that there is an amimud translation in your language, so copy it to locale:catalogs/<yourlanguage>/ and then (if you have AmigaOS 2.1+) AmiMUD will be in your favourite language.

If you want to use the online help feature of the prefs program you have to copy it in the same directory where you have copied the AmiMUD guide. The prefs program needs MUI to work.

Then you'll need to edit the prefs file as you like (see Configuration section for more info) then you can launch the program from WB or CLI.

If you don't have already installed reqtools.library you'll need to install it:

To install reqtools simply copy amimud/libs/reqtools.library to your libs: assignment. Reqtools has also a prefs program, if you want it get the last version of the library from aminet (util/libs/reqtools*).

Please note that if you launch the program from CLI you can specify some parameters not (yet) available if launched from WB. See the Program usage section for more infos...

1.7 Configuration

Configuration

When you launch AmiMUD the program will look for the file amimud.prefs in the directory where the program is located (or in s: for backward compatibility) to found the settings to use.

AmiMUD has many configuration parameters, so I suggest you to copy the example configuration file and then edit it to your needs.

You can also use AmiMUD_prefs to edit the configuration, it supports every configuration keyword and has short and online help for every gadget.

Configuration keywords

1.8 Configuration keywords

Configuration keywords

From version 2.1 the configuration keywords are NO MORE case sensitive, and be written lowercase or uppercase but without spaces, tabs or other characters before or after the keyword.

Example of correct use:

userprompt=name?

UserPrompt=name?

Examples of WRONG use:

userprompt=name?

userprompt =name?

Remember also that you can place comments and blank lines everywhere, but you can't place a comment on the same line of a command.

The lines beginning with the character ";" are considered comments.

Available keywords:

General keywords

host

port

output

variable

command

history

crmode

fastspeed

input_mode

verbose

speedwalk

separator

rx_path

echo_input

globalkeys

include

max_retries

notelnet

nologo

speed_delay

module

Autologin keywords

user

userprompt

password

passwdprompt

afterpasswd

relogin

Ticks related keywords

ticks

mark

tickseconds

minimum_ticklength

maximum_ticklength

prompt

beforetick

tickmacro

aftertick

Screen menagement keywords

customscreen

displayid

screenwidth

screenheight

screendepth

screenfont

screenfontsize

interleaved

palette

input_width

input_window_x

input_window_y

buttonleft

buttontop

startup_pen

color_reset

Advanced features

alias

gag

gagline

replace

complete

highlight

highline

macro

trigger

crypt_password

Output module related keywords

ansi

ansi16

outfont

outfontsize

review

max_review

1.9 Configuration keywords

Keyword: complete

Usage

This is the keyword used to create a list of words that will be completed if you type a part of a word in this list and then press the TAB key, it works like in tintin++.

Example:

complete=exodus

complete=exterminator

complete=beholder

complete=mistress

If you press tab when you are typing one of these words that will be

completed automatically. The first word is the one with more priority.

For instance:

cast 'heal' e<TAB>

will become:

cast 'heal' exodus

See also:

1.10 Configuration keywords

Keyword: input_mode=(cr|lf|crlf|lfcr)

Usage

Select the way amimud will send the newline to the mud, default is crlf but some mud may need lf only cr only or crlf, actually the large majority of muds work fine with crlf that is the default setting.

Example:

input_mode=lf

; needed for instance by realms (on aminet)

Default value: crlf

See also: crmode

1.11 Configuration keywords

Keyword: notelnet

Usage

This keyword will disable the AmiMUD attempt to connect using telnet protocol if connected to port 23 forcing it to raw mode.

Actually AmiMUD is compatible only with some telnetd so I've included this keyword to avoid problems...

Example:

notelnet

Default value: FALSE

See also:

1.12 Configuration keywords

Keyword: maximum_ticklength

Usage

This keyword let you specify the maximum acceptable number of seconds for a tick, ticks longer than this number will be ignored.

Example:

maximum_ticklength=90

Default value: 120 seconds

See also: Use of the tick counter

1.13 Configuration keywords

Keyword: fastspeed

Usage

If you specify this config keyword the speedwalks will not need anymore an identifier. AmiMUD will parse the string you entered in the input window and will guess if it's a speedwalk or not.

See also: {"speedwalk" link speedwalk} Use of the speedwalk

1.14 Configuration keywords

Keyword: echo_input=(yes|no)

Usage

With this keyword you can disable the input echoing of amimud. By default amimud copies the string you typed in the input window to the output window when you press return, if you don't like this behaviour you can disable it.

Example:

echo_input=no

(disable the input window echoing)

Default value: yes

See also:

1.15 Configuration keywords

Keyword: include

Usage

This keyword introduce the possibility to make complex modular configurations in AmiMUD. You can separate the triggers you use with all your characters and put this stuff in a single file for instance that will be included by the single character configurations.

Example:

include=modules/crypt

will include the alias and macros to use the crypt feature in your configuration.

See also:

1.16 Configuration keywords

Keyword: max_retries

Usage

This keyword let you choose how many times AmiMUD will try to connect to a MUD before the "connection timeup", this is very useful if the mud you are playing crashes or if it is rebooting.

Example:

max_retries=5

AmiMUD will try to connect to the mud 6 times (5 retries), so it will try for about 6 minutes (1 minute per attempt).

Default value: 0

See also:

1.17 Configuration keywords

Keyword: alias

Usage

Alias let you make shortcuts for your macros or also for commands. An alias is recognised ONLY if it's the first word you type in the input gadget, alias can be also nested, so don't make alias as "gt"->"gtell" because this will hang AmiMUD, use for instance "gt "->"gtell ".

Example:

alias=ft

gt

(if it founds ft it will substitute it with gt)

alias=heal

cast 'heal'

(you can substitute simple macros without hotkeys with aliases)

Rememer for very simple hotkeys to put a space after the definition:

alias=h<SPACE>

cast 'heal'<SPACE>

Otherwise all the works beginning with "h" will be substituted...

See also:

1.18 Configuration keywords

Keyword: globalkeys

Usage

Usually the macro hotkeys are local, they works only if the active window is the amimud one. Sometimes (for instance when playing two muds at once) may be useful to be able to send a command to one mud when you are working on the other one window. In these cases you can use this keyword. Your hotkeys will be global, so you have to define different hotkeys for each mud. They will works also if you are using IRC or IBrowse, so be careful when using this option.

Default value: No

Example:

globalkeys

(enable global hotkeys)

See also:

1.19 Configuration keywords

Keyword: variable

Usage

This character identify variables in trigger and macros. You must pay attention using a character you doesn't need to specify in the text of macro/triggers, otherwise you can have strange problems.

Example:

variable=%

(set the variable char to "%", like tinyfugue)

Default value: $

See also: variable character , keyword command , use of variables

1.20 Configuration keywords

Keyword: rx_path

Usage

To execute rexx script AmiMUD needs to know where "rx" is located, it's by default in sys:rexxc/, if you have changed this position in your sistem in order to use #rx command you need to configure properly this keyword.

Example:

rx_path=work:rexxc/

(the rx executable will be searched in work:rexxc/)

Default value: sys:rexxc/

See also: AmiMUD ARexx interface

1.21 Configuration keywords

Keyword: command

Usage

Internal commands and macros to be recognised have to be preceded by a char, the command char. The default value of this character is "#" (so you can for instance call the command cycle with "#cycle"), but this can be changed if you prefer to use another character. The command character is recognised {b}only{/b} if specified at the beginning of a line, so you can give commands with the command character inside without problems (eg. gossip hello!#!#!#?). If you have to use the command character as the first character of the command you can simply substitute it with a double command (eg ##\n will send #\n to the mud).

Example:

command=/

(set the command/macro character to "/", like tinyfugue)

Default value: #

See also: use of command , keyword variable

1.22 Configuration keywords

Keyword: input_window_y

Usage

With this keyword you can specify WHERE the input window will appear on the screen you select for AmiMUD. It works both on public screens and custom screens. By default the input window will be opened on the bottom of the screen, but if you like with this keyword you can also make it open on the top or in the middle.

Example:

input_window_y=10

(this opens the input window on the top of the screen)

Default value: bottom of the screen

See also:

1.23 Configuration keywords

Keyword: separator

Usage

This keyword let you define the character to use to separe different commands to be sent to the mud with a single RETURN keypress. The default is ';' like in tintin++, remember if you want to include this character in you text to use ';;', otherwise it will be interpreted like a separator...

Example:

separator=|

(set the separator character to "|")

Default value: ;

See also: Send commands to the mud

1.24 Configuration keywords

Keyword: speedwalk

Usage

This keyword can be used to change the character used to identify a speedwalk.

Example:

speedwalk=&

(set the speedwalk character to "&")

Default value: %

See also: use of speedwalk

1.25 Configuration keywords

Keyword: screenfont

Usage

Using this keyword and the screenfontsize one you can choose the font amimud will use if opened on a custom screen. If the size is not specified this will be 8 by default, so remember to use always screenfontsize in conjunction with this keyword. Please remember also that the output window is a console window, so if you want to modify the font of this window you need to change the "default system font" in the system preferences.

Example:

screenfont=times.font

(set the screen font to times, use screenfontsize to specify the size of the font to use)

Default value: default workbench screen font

See also: screenfontsize

1.26 Configuration keywords

Keyword: screenfontsize

Usage

Use this to specify the size of the font to use in amimud (in conjunction with screenfont) if you open it on a custom screen.

Example:

screenfontsize=11

(set the font size to 11, if it doesn't exist the font will be scaled)

Default value: default workbench screen font

See also: screenfont

1.27 Configuration keywords

Keyword: outfont

Usage

Using this keyword and the outfontsize one you can choose the font amimud will use in the output window. If the size is not specified this will be 8 by default, so remember to use always outfontsize in conjunction with this keyword. This keyword works only if you use the ANSI output module , so if you are using the console output module you'll need to change the "default system font" in the system preferences.

The AmiMUD output window font may be also a proportional one, but for better visual results I suggest you to use a fixed width font!

Example:

outfont=times.font

(set the output window font to times, use outfontsize to specify the size of the font to use)

Default value: default system font

See also: outfontsize

1.28 Configuration keywords

Keyword: outfontsize

Usage

Use this to specify the size of the font to use in the amimud (in conjunction with outfont) output window, look here for more informations...

Example:

outfontsize=11

(set the font size to 11, if it doesn't exist the font will be scaled)

Default value: default system font

See also: outfont

1.29 Configuration keywords

Keyword: host

Usage

This define the host to automatically call when amimud is launched. This option can be overridden if you specify the host to contact through command line or if you specify the NOCONNECT switch.

Probably you'll need also to specify a valid port because the large majority of the muds doesn't run on the default telnet port (23).

Example:

host=realms.community.net

See also: port

1.30 Configuration keywords

Keyword: port

Usage

The port (tcp service) to connect to.

Usually MUDS don't run on the default telnet port (23), but on higher port numbers. So if you want AmiMUD be able to connect your favourite MUD you'll need to set this keyword to the right value.

This keyword is ignored if you specified a valid port number through command line. If no port is specified the default value of the port AmiMUD will try to connect to is 23.

Example:

port=7777

Default value: 23

See also: host

1.31 Configuration keywords

Keyword: relogin=(yes|no)

Usage

If this keyword is specified in the configuration file the autologin procedure will be attempted each time you connect to a mud, otherwise it will be active only when you connect for the first time. This is useful expecially in lag condition when you may need to disconnect and then reconnect. It's not active by default because often one disconnect himself to reconnect with another character.

Example:

relogin=yes

(do autologin each time you connect to a mud)

Default value: no

See also: user , use of the autologin

1.32 Configuration keywords

Keyword: output

Usage

This option let you specify dimensions and title of the window in which will be displayed the texts coming from the mud. Actually this window is a standard amiga console, you can improve it for instance using KingCON that add a review buffer to the window and some useful menus for things like blocking the output (useful to examine an important text without problems dued to other texts coming from the MUD).

Example:

output=con:5/15/635/420/Output window/CLOSE

(For a 1:1 WB)

output=kcon:5/15/635/200/Output window/CLOSE

(If you use kcon without replacing original handler)

Default:

output=con:5/15/635/200/Output window/CLOSE

You can find KingCON on aminet (util/shell/kingcon13.lha).

See also:

1.33 Configuration keywords

Keyword: history

Usage

This command let you choose how many commands will be remembered.

AmiMUD offers you an history buffer (you can access it with curs up and curs down keys), the default dimension of this buffer is 20 commands, after the 20th command AmiMUD will begin to delete the older ones. You can change this limit using this keyword.

Example:

history=50

(remembers 50 commands)

Default value: 20

See also:

1.34 Use of speedwalking

Speedwalking (REGISTERED)

The speedwalking is a way to send to the mud long path quickly. AmiMUD recognise as speedwalk a line beginning with '%' you can change this character using the speedwalk

The character accepted as valid directions are n,w,s,e,u,d. You can for instance write a speedwalk like this:

%esesuwwn

This will make your character move east, south, east, south, up....

For long path you can also use numbers in a speedwalk. The number you specify refers to the direction that follow the number:

%s13en2u

This will move your character south, then 13 times to east then north and finally 2 times up.

Speedwalk are very useful in macros to move to a place quickly.

From the version 2.6 of AmiMUD speedwalk can be used also without the initial speedwalk character if you specify the fastspeed keyword in the configuration, AmiMUD parse the string and is able to recognise if a string is a speedwalk or not, be careful when you use this mode because for instance the string "news" will be recognised as a speedwalk! (I've put a menu option to disable this for this reason)

Warning: the speedwalk path is sent to the mud after you press return, so there is no way to stop it.

1.35 Configuration keywords

Keyword: crmode=(crlf|lf|cr)

Usage

This command let you choose the type of lines the mud send to you. Actually every mud uses the "crlf" mode (default value), so you probably will never need to use this keyword. Some strange telnet services may require to use "cr" or "lf" mode, I've included this feature for future telnet compatibility...

Example:

crmode=cr

(expects that carriage returns are sent as CR character)

Default value: crlf

See also:

1.36 Configuration keywords

Keyword: verbose=(yes|no)

Usage

This keyword is very important. Many AmiMUD commands can create output messages on the output window, if verbose is disabled none of the not strictly needed output will be displayed. This can be useful for an expert user, but isn't suited for a beginner or if you are testing new trigger/macros...

Example:

verbose=no

(Disable verbose outputs)

Default value: yes

See also:

1.37 Configuration keywords

Keyword: user

Usage

This is the first of the four keyword used to define autologin .

In user you have to specify the name of character you want to login automatically when you start AmiMUD.

IMPORTANT: To enable autologin you MUST specify user, password, userprompt and passwdprompt. If one of these is not specified in the configuration autologin will not work.

Example:

See userprompt

See also: userprompt password passwdprompt afterpasswd relogin

1.38 Configuration keywords

Keyword: userprompt

Usage

This keyword tell AmiMUD when it have to send the character name (user) to the mud. It must be a part of the string received from the mud after the banner.

Example:

userprompt=name?

(for shadowdale and similar muds)

See also: user

1.39 Configuration keywords

Keyword: password

Usage

In password you have to specify the password of the character you specify in user keyword.

Example:

See userprompt

See also: user

1.40 Configuration keywords

Keyword: passwdprompt

Usage

Here you have to specify the prompt received by the mud before typing the password.

Example:

passwdprompt=assword:

(for nearly every mud)

See userprompt

See also: user

1.41 Configuration keywords

Keyword: afterpasswd

Usage

Usually after the password you'll need to send to the mud other text to complete the login operation. For instance on diku muds you'll need to confirm the MOTD (message of the day) with RETURN and to select option 1 from the menu. With afterpasswd you can do this automatically.

This feature is only used if autologin is enabled.

Example:

afterpasswd=1\n1\n

(afterpasswd for a dikumud)

See also: user

1.42 Configuration keywords

Keyword: ticks=(yes|no)

Usage

This keyword is useful if you use AmiMUD to play LPMud or other mud types that don't use ticks.

Example:

ticks=no

(disable tickcounter)

Default value: ticks=yes

See also: Use of tickcounter

1.43 Configuration keywords

Keyword: mark

Usage

If you want to use the tick counter properly and to see your prompt in the title of the input window you need to modify your prompt (or the mark) making AmiMUD able to detect it in the incoming text from the mud. If you can modify the prompt (in many mud you can do it with the command "prompt") you can simply use the default mark and modify your prompt as follow:

prompt ##[your old prompt]>

The prompt must begin with ## (or another mark you can configure with the mark keyword) and end with ">" (this cannot be changed). I made this choice because I see that the way TinyFugue recognises the prompt (a line without the final CR) has many troubles in lag condition...

This way is a bit more complex but works always :)

Example:

mark=**

(identify the prompt as the line that begin with "**" and ends with a ">")

Default value: mark=##

See also: Use of tickcounter

1.44 Configuration keywords

Keyword: tickseconds

Usage

Use this if you know the length of the tick of the mud you are playing. For DIKU type muds it's usually 75seconds, for other mud types 90 and 120, this may not be true if the mud has many players online and then the ticks will be longer.

Example:

tickseconds=90

(set tick length to 90 seconds)

Default value: 75

See also: Use of tickcounter

1.45 Configuration keywords

Keyword: minimum_ticklength

Usage

Set a minimum value to accept as tick length in the automatical tick calculation process, very useful if you have not defined triggers for heals, refresh, etc...

Example:

minimum_ticklength=20

(set the minimum accepted tick length to 20 seconds)

Default value: 60

See also: use of the tickcounter

1.46 Configuration keywords

Keyword: crypt_password

Usage

This password can be set in the configuration or during the use of the program with the #password command. If not defined you will not be able to use the crypt feature of AmiMUD (and Elf for Windows).

Example:

crypt_password=W Genoa

The password is "W Genoa", only the users of AmiMUD or Elf that know the password can decrypt your text.

See also: use of crypt

1.47 Configuration keywords

Keyword: beforetick

Usage

A simple macro to be executed before the tick, useful to recover mana and hp more quickly. Use it in conjunction with aftertick .

Example:

beforetick=rest\nsleep\n

(make your character rest and sleep 3-5 seconds before the tick)

Default value: not enabled

See also: use of the tickcounter

1.48 Configuration keywords

Keyword: tickmacro

Usage

A macro to be executed when the tick occurs.

Example:

Default value: not enabled

See also: use of the tickcounter

1.49 Configuration keywords

Keyword: aftertick

Usage

Simple macro that will be executed after the tick, very useful in conjunction with beforetick .

Example:

aftertick=wake\nstand\n

(make your character wake and stand up right after the tick)

Default value: not enabled

See also: beforetick use of the tickcounter

1.50 Configuration keywords

Keyword: prompt

Usage

Very useful keyword. If you want to use the tick counter with the automatical tick length calculation you'll NEED to specify a correct value for this one. If the ticks of your mud are fixed length then you have to delete this keyword from your config (this will disable automatical tick calculation) and specify the right tick length .

Example:

if the prompt of the mud is:

##HP:3049 MA:2020 MV:111 (------) * -*>

You have to set:

prompt=HP:%ld MA:%ld MV:%ld

(recognise a prompt with 3 variables)

Default value: not enabled

See also: use of the tickcounter

1.51 Configuration keywords

Keyword: customscreen

Usage

AmiMUD will open as default on Workbench screen, if you want to make it opens on a custom screen you need to specify this keyword. Note that ANSI mode is available only if amimud is opened on a custom screen (or on a public screen with ansi colors :)).

Example:

customscreen

(will popup a requester if the other screen parameters are not defined)

Default value: not enabled

See also: AmiMUD on custom screen

1.52 Configuration keywords

Keyword: displayid

Usage

This keyword specify the displayid to use for the screen, if you want to specify it you'll need to specify it as a DECIMAL number. It is considered only if also screenwidth , screenheight and screendepth are correctly specified in the configuration.

Example:

Default value: not enabled

See also: AmiMUD on custom screen

1.53 Configuration keywords

Keyword: screenwidth

Usage

The width of the custom screen to open. It is considered only if also displayid , screenheight and screendepth are correctly specified in the configuration.

Example:

screenwidth=640

Default value: not enabled

See also: AmiMUD on custom screen

1.54 Configuration keywords

Keyword: screenheight

Usage

The height of the custom screen to open. It is considered only if also displayid , screenwidth and screendepth are correctly specified in the configuration.

Example:

screenheight=512

Default value: not enabled

See also: AmiMUD on custom screen

1.55 Configuration keywords

Keyword: screendepth

Usage

The depth of the custom screen to open. It is considered only if also displayid , screenheight and screenwidth are correctly specified in the configuration.

Example:

screendepth=3

(the minimum depth for for ANSI mode)

Default value: not enabled

See also: AmiMUD on custom screen

1.56 Configuration keywords

Keyword: ansi

Usage

If you specify this keyword (and you open AmiMUD on a custom screen) you'll be able to see muds in ANSI colors (if the mud you are playing supports them).

Example:

ansi

(this will enable ansi colors)

Default value: Not enabled

See also: palette

1.57 Configuration keywords

Keyword: palette

Usage

If you want to use a particular palette with AmiMUD you can. You can edit it with the menu option Prefs->Edit palette.. and then save it to a suitable file (for example progdir:amimud.palette) and then add the keyword palette=amimud.palette to your configuration. Please note that the palette keyword works only if AmiMUD is opened on a custom screen.

Important: This feature works only with OS 3.0+

Example:

palette=work:pictures/mypic.iff

(load in the amimud screen the palette of mypic.iff)

Default value: Not enabled

See also: Open AmiMUD on a custom screen

1.58 Configuration keywords

Keyword: macro

Usage

Macro are very useful in muds, expecially if you need to do things very quickly... A macro is a way to send many commands to the mud in a quick way. Macros can be defined in the configuration file, with the command #macro or through the menus. You can also see a list of all the defined macros with the command #list or through menus.

Macro may require arguments or not and the macro text may contain variables. Actually if you want your macro have two parameters you have to add after the macro name $0 $1. Actually macro parameters MUST be in crescent order 0..4.

macro=givefb $0 $1

rem bag\nget $0 bag\nwear bag\ngive $0 to $1\n

This is a correct definition for macro with arguments, the following one instead will NOT work:

macro=givefb $1 $2

rem bag\nget $1 bag\nwear bag\ngive $1 to $2\n

From version 1.1 also macro with parameters can have hotkeys but they works in a different way. When you press the hotkey the macro name will be shown in the input window and the cursor will be placed after the last character of the macro name. Then you need only to type the macro argument(s) and press return. This is useful expecially with macros with only one argument.

Macro can also use variables. The only difference between a macro with parameters and a macro that contains variables is that the second can be launched with an hotkey. You can set the variables with triggers or with the #set command. See below for some examples.

Hotkeys:

The macro hotkeys use the standard Amiga hotkey format (chapter 10-25 of OS 3.1 Workbench manual) here is a short summary:

alt, lalt, ralt - alt keys

shift, lshift, rshift - shift keys

lamiga, ramiga - amiga keys

ctrl, control - control key

leftbutton, middlebutton, rightbutton - mouse buttons

F1..F10 - function keys

Examples of hotkeys:

hotkey=shift F1

hotkey=lamiga ctrl 5

hotkey=leftbutton F1

Important: The macro text must always end with a carriage return (\n) otherwise the macro execution will not be confirmed...

Warning the hotkeys defined in AmiMUD are GLOBALS, this means that if you assign F1 to a macro and you press F1 the macro will be executed also if the amimud window/screen is not the active one! This is dued to the way commodities handles hotkeys, if you need an hotkey you can simply disable AmiMUD commodity interface with Exchange or similar programs.

Buttons:

AmiMUD macros may have also a button shortcut, to add the button simply put the keyword "button" in a line after the macro name and before the macro definition. The button window can be opened/closed with a menu option. You can choose the button window position with the two configuration keywords buttonleft and buttontop .

Examples:

(These examples are thought for diku-like muds)

macro=hungry

hotkey=lamiga h

rem bag\nget bread bag\neat bread\nwear bag\n

(Macro without parameters, useful if your character is hungry (may be activated also with the hotkey amiga+h...)

macro=getbag $0

rem bag\nget $0 bag\nwear bag\n

(Get an object from the bag)

macro=hun&thir

#hungry\n#thirsty\n

(You can also call macros in a macro)

macro=disint

hotkey=alt 3

button

cast 'disintergrate' $7\n

macro=meteor

hotkey=alt 2

button

cast 'meteor swarm' $7\n

macro=dispel

hotkey=alt 1

button

cast 'dispel magic' $7\n

These macros are targeted, using the command #set you can specify what does the variable $7 contents (example: #set 7 dracolich) and then you can call them with hotkey or typing the macro name without the need to specify the target each time.

Examples of macro defined through the command macro:

#macro "hungry" "rem bag\nget bread bag\neat bread\nwear bag\n"

(a macro defined with this command can NOT have hotkeys).

See also: Add a macro , Advanced use

1.59 Configuration keywords

Keyword: trigger

Usage

Triggers are the most powerful feature of a mud client over a normal telnet connection. Triggers may make automatic many actions that have to been performed quickly or that have to be done very often.

Triggers may contains variables , this can be useful in two different ways.

The first (and obvious) way is to use part of the text that activate the trigger in the trigger body, for instance to split some coins through the group members (see examples below). If you put a variable in the trigger definition (obviously with some other text, otherwise ANY line willl match that trigger) if the trigger is found the variable will contain the text between the words that make the trigger match.

trigger=A $1 C

#echo trigger found, $1 is between A and C\n

if the mud sends to the player "A B C" the trigger will be activated and the phrase:

trigger found, B is between A and C

Will be printed in the output window.

The second way to use triggers is an alternative to #set . You can set variables with a trigger without using them. For instance if you need (on a diku-type mud) to know who is the group leader and keep the value in a variable for future use you can make a trigger to get the name from the text the mud sends in some cases:

trigger=now member of $7's group

#echo Put groupleader name ($7) in variable 7\n

Remember that macros and trigger must always end with a carriage return (\n) otherwise the text will be send to the mud, but not confirmed!

Example:

trigger=ou are hungry

#hungry\n

(See the section macro to see how #hungry is defined.)

trigger=as you hear $0m$1's

get all.coins $1\n

trigger=as you hear $1's

get all.coins $1\n

(These two macros make the same thing, the first works if you are playing in ansi mode, the second in plain mode, please note the use of the $0 variable to strip the ansi code)

trigger=here were $1 coins

split $1\n

(This macro split the coins you get from a corpse to the members of your group)

See also: Add a macro , Advanced use

1.60 Configuration keywords

Keyword: gag

Usage

This keyword let you define some text you want to be displayed. It can be useful to make output of some commands cleaner.

Example:

gag=is in excellent condition.

(Will be useful to make the output of command "equip" on diku-mud shorter)

See also: gagline, highlight

1.61 Configuration keywords

Keyword: gagline

Usage

With this command you can tell amimud to not display a line of text. This is useful for instance in fightning with many fighters. You can disable gagline you have defined or add a new gagline in any moment through menus.

Example:

gagline=misses you

(You will see only the hit of enemies if they hit you)

See also: gag, highlight

1.62 Configuration keywords

Keyword: highlight

Usage

Highlight may be used to make some text be highlighted (:-)). This can be useful in many cases. Highlights can be inserted in the configuration file or through the menu option, you can enable/disable them through menus or with the command #high .

The syntax of the command is:

highlight=type

text

The difference between highline and highlight is that the first highlight the whole line where "text" is find (works better if ANSI colors are disabled) and the second only the specified "text".

These are the actually defined highlights:

BOLD 0

UNDERLINE 1

ITALICS 2

REVERSE 3

BOLD_UNDERLINE 4

CLEAR_SCREEN 5

COLOUR_0 10

COLOUR_1 11

COLOUR_2 12

COLOUR_3 13

COLOUR_4 14

COLOUR_5 15

COLOUR_6 16

COLOUR_7 17

Example:

highlight=0

extemely well

highlight=1

massacres

highlight=4

devastates

(these three will highlight in different ways different hits)

Default value:

See also: highline

1.63 Configuration keywords

Keyword: highline

Usage

Highline may be used to make some text be highlighted (:-)). This can be useful in many cases. Highlights can be inserted in the configuration file or through the menu option, you can enable/disable them through menus or with the command #highline .

The syntax of the command is:

highline=type

text

The difference between highline and highlight is that the first highlight the whole line where "text" is find (works better if ANSI colors are disabled) and the second only the specified "text".

These are the actually defined types of highlight:

BOLD 0

UNDERLINE 1

ITALICS 2

REVERSE 3

BOLD_UNDERLINE 4

CLEAR_SCREEN 5

COLOUR_0 10

COLOUR_1 11

COLOUR_2 12

COLOUR_3 13

COLOUR_4 14

COLOUR_5 15

COLOUR_6 16

COLOUR_7 17

Example:

highline=0

extemely well

highline=1

massacres

highline=4

devastates

(these three will highlight the whole lines where the text "extremely well", "massacres" and "devastates" appear in different ways)

Default value:

See also: highlight

1.64 AmiMUD on custom screen

Open AmiMUD on a custom screen

There are two ways to make AmiMUD open a custom screen.

1 - Using the CS or CUSTOMSCREEN command line switch if you launch AmiMUD from a shell.

2 - Specifying customscreen keyword in the configuration file.

These two option will pop up a screenmode requester, if you use AmiMUD always on the same screen you may like to select the mode once for all. You can do this in three ways:

1 - (Easy way) Select the mode as usual in the screenmode requester then save the preferences of AmiMUD with the "Save Prefs" menu option.

2 - (Tricky way) Edit displayid , screenwidth , screenheight , screendepth . These FOUR parameters (and customscreen) have to be specified if you want to skip the screenmode requester. Optionally you can also specify ansi (that init the screen with the default ANSI palette), palette that let you load an IFF palette (also from a picture) in the screen and screenfont/screenfontsize to specify the font to use in the custom screen.

3 - Use AmiMUD_Prefs

1.65 Send Commands to the MUD

Send Command to the MUD

To send commands to the mud you simply need to write in the input window gadget and then press return. The text you have typed will be sent to the MUD. Please note that the input gadget will be activated also if you type a character on the output window, if you want to avoid this (for instance because you want to copy to the clipboard some text from the output window) you need to disable the commodity interface of AmiMUD through the program exchange (distributed with WB since 2.0), the side effect is that also hotkeys will be disabled.

From V1.10 AmiMUD have the possibility to send multiple commands on a single line. The commands have to been separated by ';' (if you want to change the separator character you can use the configuration keyword separator), I have to use this instead of "\n" as separator because some internal commands support \n in the arguments... Anyway tintin users will be happy with this choice (and the others can change it through the config keyword) :)

Example:

open door;n;backstab drow

Will performs these operations in sequence avoiding the risk of type error in a time critical situation.

The multiple commands feature is available only in the registered version.

1.66 Using the program

Using the program

Start AmiMUD

Output Modules

Connect to a MUD

Send commands to the MUD

1.67 Iconify the program

Iconify

From version 2.1 AmiMUD may be iconified, you can do it with the menu option or with the "Hide Interface" command of commodities exchange.

To uniconify it you have to use commodities exchange "Show Interface" command.

1.68 Use of the autologin

Autologin

To connect to your favourite mud in a quicker way you can use the autologin facility of AmiMUD. To active this feature you need to specify in the configuration file the keywords: userprompt , user , password and passwdprompt . If one of them isn't found in the configuration file autologin will NOT be available.

There are also two other autologin optional keywords, afterpasswd and relogin . With the first you can specify which characters send to the mud to bypass the initial menus and banners usually shown after the login. The second give you the possibility to do autologin not only the first time you connect.

1.69 Multiple connections...

Multiple connections?

AmiMUD 3.0 is fully multithread, you can run up to 9 session at once. To open a new session you just need to select the menu option "New session" that will open a new session with a default configuration or with "Open new session..." that will ask you with a file requester for an AmiMUD configuration file.

You can close a session with the menu option "Close this session" or with the close gadget of the input window. When you close the last session AmiMUD quits. The "Sessions" menu of AmiMUD allows you to see how many sessions are actually active and let you active a session (the input window of that session will be actived and bring in front of the other windows).

The only limitation of the new structure of AmiMUD is that all the sessions have to be in the same screen (Workbench or public). To avoid this you can run multiple copies of AmiMUD, each copy will open his screen. The TOTAL limit of the sessions is still 9.

1.70 Connecting...

Connect to a MUD

To connect a mud you usually need to specify TWO parameters.

Address:

May be an IP (192.106.166.6) or a mnemonic name (mclmud.mclink.it). The first one works also if you don't configure properly the DNS (name server) on your tcp stack. You have to write the address calling the menu item "Set Address..." in the menu "File" if you have not already defined them in the configuration file.

Port:

Usually muds don't run on the default telnet port (23). So if you want to connect a mud generally you have to specify also the port number the mud runs on. To do this use the "Set Port..." menu item in the menu "File". The port may also be specified in the configuration file.

Once you have specified the address and the port and obviously you have already the TCP stack running you can connect to the mud. Use the "Connect" menu item or the shortcut RightAMIGA+C.

You can also connect to a MUD with the command connect .

The "tick" near the "Connect" menu item means you are connected.

1.71 Use of the keymap

Keymap Movement

One intresting feature of AmiMUD is the "keymap movement", AmiMUD remaps the classic cardinal directions on the numeric keypad, so you can move using it, without need of hitting return for each keystroke...

This is how the directions are remapped:

7 8 9 -

n d

4 5 6 +

w exits e u

1 2 3

s

0 .

This is the default configuration, these settings may be changed with the configuration keywords key_0, key_1, ... , key_9, key_dot, key_plus, key_minus.

You can edit the setting of any of these keys as you want but you have to respect some limits:

- Maximum 10 chars.

- Newlines (\n) are not permitted.

- You can't use internal commands or macros .

I suggest everyone to use keypad to move it's very quickly and handy if you use it often.

1.72 Variable character

Variable character

The variable char is a character that identify a variable inside a trigger or a macro, so you have to use a character you don't need in the text, otherwise you will get an error if you will use it.

Dafault value: $

Example:

(in the configuration file)

variable=%

macro=hello %0

gossip Hello %0!\n

(This will change the variable char from '$' to '%', like tinyfugue)

1.73 Commmand character

Command character

Actually all AmiMUD internal commands and all macros are detected through a special character at the beginning of the line. The default command character is '#' but may be changed through the config keyword command .

1.74 AmiMUD internal commands

Command: beep

Usage

If you call beep the screen will blink and/or a sample will be sound (it depends from your system preferences). This command may be very useful to make a trigger that call you if you are away from keyboard and someone wants to talk to you.

Example:

(in the configuration file if your name is Exodus)

trigger=xodus

beep\n

(will perform a display beep if someone on the mud gossips says or tell "xodus" the first letter of the name is not specified to avoid lower/upper case problems).

1.75 AmiMUD internal commands

Command: echo

Usage

---

echo is a small and very useful command for complex triggers . It simply prints to the output window the contents of the line after his invocation. This sometimes may not be very useful, but often can be very handy (expecially if verbose =no in your configuration file!).

Example:

echo hello

(will print hello in the output window)

echo $ 1

(will print the contents of variable "1" to the output window, a bit more useful than the previous example)

1.76 AmiMUD internal commands

Command: cycle

Usage

The cycle command let you repeat <times> times a macro or some commands. It can be used also into a trigger.

Syntax: #cycle <times> <macro or text>

Examples:

#cycle 5 buy bread\n

Buys 5 peaces of bread

#cycle 20 kick $9\n

Kicks 20 times the guy in variable $9

#cycle 2 #eat\n

Executes 2 times the macro #eat.

1.77 AmiMUD variables

Variables

Actually AmiMUD supports 2 types of variables.

The simpler ones are the variables from 0 to 9, these are statically defined so they are very quick, you can reference them with $0,..,$9 provided that '$' is the variable character . These can be used by triggers/macros and all the other commands. A variable is identified by the variable char (default '$') followed by a number from 0 to 9. The maximum length of the text in a variable is 120 characters. A variable is keep in memory until it's replaced by another value, this can be done through macros , triggers or the #set command.

Custom variables are dynamically created if needed, for instance with:

#set tank mongo

You create a variable $tank that you can use in macros or trigger. The limitation of the custom variables are two:

- Custom variables may NOT be the arguments of a macro.

This macro don't work:

macro=givebag $object $person

get $object bag\ngive $object $person\nwear bag\n

Use:

macro=givebag $0 $1

get $0 bag\ngive $0 $1\nwear bag\n

- Custom variables may not be triggered:

This trigger don't work:

trigger=$mob's death cry

get coins $mob\n

Use:

trigger=$2's death cry

get coins $2\n

See also: Advanced use of variables , Variable char , variable keyword

1.78 AmiMUD internal commands

Commands

All these command may be used directly through the input window or in triggers and macros. All the commands must be used with the following syntax:

commandname <arguments>

action

alias

beep

clip

complete

connect

crypt

cycle

decrypt

echo

file

gag

gagline

help

high

highline

if

list

log

macro

math

music (REG)

password

play

record

remmacro

remaction

replace

rx

send

set

skiptick

separator

sound (REG)

stoplog

suspend

sync

system

trigger

unaction

wait

zap

1.79 AmiMUD internal commands

Command: separator

Usage

Syntax: #separator (on/off)

This command is used to disable temporary the separator feature, it's useful only in the registered version and in certain situations (when you need the separator character in a sentence for instance).

1.80 AmiMUD internal commands

Command: crypt

Usage

Syntax: #crypt <variable> <sentence>

This command is used to crypt the sentence given and place the compressed sentence in the variable number <variable>. You have not to use it directly, you can include the module crypt that contains macros and aliases that make the access to this feature very easy.

To be able to use #crypt and #decrypt the crypt password have to be defined through the command #password or through the config keyword crypt_password .

The text crypted with AmiMUD can be decrypted also with Elf for Windows (elfw218.zip, search this with archie).

Examples:

look at "modules/crypt"

See also: decrypt Use of crypt

1.81 AmiMUD internal commands

Command: decrypt

Usage

Syntax: #decrypt <variable> <sentence>

This command decrypt a phrase and place the result in the variable. You have not to use it directly, you can include the module crypt that contains a trigger that make the access to this feature very easy.

To be able to use #crypt and #decrypt the crypt password have to be defined through the command #password or through the config keyword crypt_password .

AmiMUD can decrypt the text crypted with Elf for Windows (elfw218.zip, search this with archie).

Examples:

look at "modules/crypt"

See also: crypt Use of crypt

1.82 AmiMUD internal commands

Command: password

Usage

Syntax: #password <crypt password>

1.83 AmiMUD internal commands

Command: rx

Usage

Syntax: #rx <script>

This command launch the rexx interpreter (rx), it works only if you configure properly rx_path. You can send directly arexx commands to amimud or execute a previously written script. See examples for major info. You can execute an arexx script also through the menu option "Execute ARexx script..." or executing rx in a shell.

1.84 AmiMUD internal commands

Command: wait

Usage

Syntax: #wait <seconds*50>

The wait command can be used in triggers to delay a command, it accept as parameter a number identifying the 1/50 of seconds to wait, it practically passes this number to the amigados function Delay(). While waiting both output and input will be freezed, so don't wait for very long periods :)

1.85 AmiMUD internal commands

Command: action

Usage

A trigger can be added through the config keyword trigger , through menus or through this command. If you use #action to define a trigger you CAN'T define hotkeys for triggers...

Example:

#action "ou feel less protected." "cast 'armor' exodus\n"

If the magic shield of your character fades cast on yourself another one.

1.86 AmiMUD internal commands

Command: help

Usage

This command list all available internal commands and provides a brief description of their usage. Call it without arguments.

1.87 AmiMUD internal commands

Command: set

Usage

Syntax: #set <var num> <value>

This command assign the text in <value> (until the end of the line) to the variable <var num>. It can be very useful for targeted macros .

See also: AmiMUD variables , Macros

Example:

#set 7 beholder

1.88 AmiMUD internal commands

Command: list

Usage

This command lists all defined macros. You can also see them through the menu option "Macro List...". It doesn't require arguments.

1.89 Log to a file...

Log to file

If you are exploring an unknown area or you are trying an heroic mission you probably want to store what you are doing on disk for future use.

AmiMUD let you log to file through two simple commands or a menu item.

log <filename>

Will start to log all the output and everything you send to the mud to the file <filename>. You can stop the operation with the command: #stoplog.

Otherwise you can use the menu item "Log to file..." in the menu "File". When you select it for the first time you will be prompted with a file requester to select the file to write to. To end the log operation you have to select the "Log to file..." item another time.

If there is a "tick" near to the "Log to file..." text it means you are logging. You can also activate/deactivate logging with the shortcut rightAMIGA + L.

Note: If you are using ANSI your log will contain ANSI codes, I suggest using stripansi.lzh (on aminet) to remove ansi codes from log files.

1.90 Add a macro or a trigger

Macros & Triggers

Macro and triggers can be added in three ways. The first (and simpler) is in the configuration file, with the keyword macro and trigger , otherwise you can define them online with the #macro and #action commands or also through the menu options "New Macro..." and "New Trigger...", remember anyway that with the macro command you can't define macro hotkeys while with the config keyword or through menu you can. Usually is better to define macro and trigger in the configuration, so that you don't need to save the configuration to keep them for future use.

1.91 Advanced use of macros and triggers

Advanced use of macros and triggers

Macros and triggers are very powerful if used properly. Actually the only limitation they have is the not yet supported command #if (probably it will be in the next version) and the maximum size of 159 characters. This second problem can be easly avoided making a very long macro as the concatenation of different macros:

macro=verylong

#part1\n#part2\n#part3\n

macro=part1

[160 chars available...]

macro=part2

[160 chars available...]

macro=part3

[160 chars available...]

Note that if you put more than 160 characters on a macro definition it will be cutted, but amimud will not complain about it...

One use of macros very intresting expecially for combat oriented muds is the targeting. This let you perform the commands very quickly, without the risk of typing errors caused by a too fast typing on the keyboard. Follow an example of targeted macros for diku-type muds:

;This macro set the target and the tank and tell it to the group

macro=target $0 $1

#set 8 $0\n#set 9 $1\ngtell Ok, the tank is $0 and the target is $1\n

; Attack macros (target is $9) note that all macros are available with

; the single keypress of a function key. This can be done also defining these macros

; as the first ten in the config file WITHOUT specifying an hotkey.

macro=disint

hotkey=F1

cast ’disint’ $9\n

macro=meteor

hotkey=F2

cast ’meteor’ $9\n

macro=dispel

hotkey=F3

cast ’dispel’ $9\n

macro=slow

hotkey=F4

cast ’slow’ $9\n

; Defence macros (target is $8)

macro=heal

hotkey=F6

cast ’heal’ $8\n

macro=sanc

hotkey=F7

cast ’sanc’ $8\n

macro=armor

hotkey=F8

cast ’armor’ $8\n

1.92 Add a gag or an highlight

Add a gag or an highlight

A gag (word or line) and an highlight (word or line) may be added through the config keyword gag , gagline and highlight , highline link highline} or through the menu options Prefs->New Gag->(Line...|Word...) and Prefs->New HighLight->(Line..|Word...). A gag simply cut the specified part of text from the incoming text from the mud, a gagline cut the entire line if it found the specified text on it (so a documentation to see what styles/colors are actually available.

1.93 AmiMUD internal commands

Command: send

Usage

This command is very useful to test triggers / highlights / gags without need to be connected (you only need to start amitcp/inet-225, I’m not sure it can be done with other TCP stacks). If you type the command "#send hello world!" the program will think it has received "hello world!" from the mud, then if you have some trigger/highlight/gag on these words they will be performed.

Note: the #send command to be effective must be performed in verbose mode .

Example:

#action "hello world!" "shout I hate the people shouting hello world!"

#send "Ehi, hello world!"

this will make the previously defined trigger being activated, obviously if you are connected the shout command will be sent to the mud, otherwise only to the output window (if you are in verbose mode).

See Also: Triggers

1.94 Send text to the mud...

Send text to the mud

Sometimes may be useful to send some text previously written to the mud. This can be done in two ways in AmiMUD.

1) Clipboard - You can edit the text in a text editor or word processor or also a shell then copy it to the clipboard usualy through the combination right amiga+c, then you can use the #clip command (without parameters) or the "Send Clipboard" menu option (in Actions).

2) File - You can edit the text with your favourite text editor/wp and then save it to a file in plain ASCII text mode, then with the command #file <filename> or the "Send File..." menu option (in Actions) you can send this file to the mud.

Note that in both cases the text will be send also to the output window ONLY if the verbose mode is active.

1.95 Quit the program...

Quit the program

Clicking on the input window close gadget or on the quit menu option will make the program quit (after a requester asking for confirm the operation) and close the socket. Remember that this don't perform the mud quit procedure (you may need to go to an inn or at least to perform a save command) then your character will be classified as link dead. So to be sure all it's ok you have to wait the message "connection closed by foreign host" before quitting the client.

1.96 Disabling Trigger/Highlight/Gags...

Disabling Trigger/Highlight/Gags...

In some cases may be useful disabling trigger/highlights or gags (expecially trigger during fightings...) so AmiMUD has some commands to do it and also some menu options...

All these commands are enabled by default.

Commands:

#trigger on

Enable triggers

#trigger off

Disable triggers

You can also enable/disable a single trigger with the command unaction

#high on

Enable text highlights

#high off

Disable text highlights

#highline on

Enable line highlights

#highline off

Disable line highlights

#gag on

Enable gags

#gag off

Disable gags

#gagline on

Enable gaglines

#gagline off

Disable gaglines

#replace on

Enable gag replaces

#replace off

Disable gag replaces

There are also in the preferences menu options to do the same thing.

See also: disable a single trigger

1.97 Use of the tick counter

Tick Counter

The tick counter is one of the more impressive (and complex) features of the client. It's very simple to use if your mud as a fixed length tick. In this case you have only to configure the tickseconds keyword to the right value (default for diku is 75 seconds). Probably you'll need also to perform a #sync call to syncronize the tick counter with the real ticks. You can also make a simple trigger that catch the messages that mud sometimes send at the tick (The day has begun, The moon is going down...) to perform the #sync command automatically.

If your mud as a very high load (many users) or is of another type than diku the ticks may have a NON fixed length. With non fixed length the use of the tick counter is a little more difficult:

You'll need to configure your prompt with a mark at the beginning (default ##, but may be changed with the keyword mark) and the final character '>' (this is fixed). In a diku type mud you can do this with the following command:

prompt ##H:%h M:%m V:%v XP:%x %c -%C>

This example prompt uses the standard mark so you don't need to use the keyword in the configuration.

You'll need also to tell the program where there are movement, hitpoint and mana in the prompt string. This is done with the prompt keyword.

This because AmiMUD recognise the tick making a comparison of the values of the first TWO or THREE numeric values of the prompt, this detection is automatical, it does depends on how many %ld the program finds in the prompt definition)

It is important that the two or three number you want to use are the first two or three you specify in the prompt. I don't think that it's a big limitation :)

Example:

##H:559 M:200 V:100 bla bla bla >

This is how your prompt should look with the previous configuration, the important is that the first two or three parameters are the ones that RAISE at ticks. With this example prompt you should configure the prompt keyword as follow:

prompt=H:%ld M:%ld

If you want to use only mana and hp for detecting ticks or:

prompt=H:%ld M:%ld V:%ld

If you want to use also movement.

The tick length can be controlled also with minimum_ticklength (default value 60 seconds) and maximum_ticklength (default value 120 seconds). Ticks longer than maximum_ticklength or shorter than mimimum_ticklength will be discarded. Please note than 60 and 120 are the tipical dikumud limits, if you play for instance medievia the ticks are a lot shorter (about 30 seconds) so you will have to change these values.

Obviously this will make the tick counter fail if a character is healed or refreshed, in the example configuration there are some triggers that using the #skiptick command let you avoid this. Basically the #skiptick command tell the mud that the next increase of HP or Mana or Movement as to be ignored.

Please note that if you play Medievia you can use the tick counter ONLY if you disable ANSI, this because of the fact that medievia prompt contains a lot of ansi codes that may change...

1.98 Launch the program

Start AmiMUD

The program can be launched from CLI or from workbench if you launch it from WB you loose the possibilty to specify some option available only through command line.

Command line options

This is the AmiMUD argument pattern:

HOST,PORT,CFG=CONFIG/K,SC=SCREEN/K,NOC=NOCONNECT/S,NC=NOCONFIG/S,

CS=CUSTOMSCREEN/S,REQ=REQUESTER/S

HOST, PORT - These are the host and port to connect to. If you specify these you override the prefs settings (if present).

Example: AmiMUD realms.community.net 7777

CONFIG filename - This potion make AmiMUD load "filename" as configuration instead of the default "s:amimud.prefs", useful if you use multiple characters with different configs.

SCREEN - Name of the public screen where to open AmigaMUD

NOCONNECT - If autologin is actived in the prefs file this option will disable it.

NOCONFIG - Doesn't load the prefs file. As if amimud.prefs doesn't exists (nor in progdir: nor in s:).

CUSTOMSCREEN - Force AmiMUD on a custom screen, if this option is specified AmiMUD will pop up a screen mode requester to choose the screenmode you want to use.

REQUESTER - Open a file requester to choose the name of the prefs file to load, useful if you own many different configurations :)

1.99 AmiMUD Icon Tooltypes

Tooltypes

From version 2.1 of AmiMUD you can specify some options in the icon tooltypes, to change them click one time on the program icon and select "Informations..." from the "Icon" menu.

Here are the AmiMUD supported tooltypes (for descriptions look at the start section).

HOST=hostname

PORT=portname

CONFIG=filename

SCREEN=screenname

NOCONNECT

NOCONFIG

CUSTOMSCREEN

REQUESTER

1.100 Crypt some text...

Crypt

AmiMUD offers a nice crypt feature compatible with the one of Elf for windows. This feature will let you talk with other character without the risk that immortals or people that have access to the mud log can understand what you are saying.

The way it works it's very simple:

First you have to exchange a password with the one you want to talk, for instance through E-Mail. Then you can use provided aliases tellc, sayc, gtc, ftc, gossc (include modules/crypt in your configuration file) to talk with other people that know the password.

The algorithm is based on a crc32, so it's quite difficult to decrypt.

Look at modules/crypt to see the details of the working method of this code...

See also: crypt, decrypt , #password , crypt_password

1.101 History

History

---

v 1.0 (29-7-96)

- First public release

v 2.0 (5-10-96)

- New keyword screenfont/screenfontsize to change the custom screen font.

- Speedwalk for quicker movements and simpler macros. (REGISTERED)

- Localization.

- Highlines: now you can highlight also a whole line.

- Aliases: now you can use shortcut to command or macros.

- Rewritten macro hotkey handling code, now works also with macro with parameters.

- Rewritten input window opening code, now it opens with the correct font and with the correct size and position.

- New keywords separator, globalkeys, input_window_y, highline, relogin, rx_path, echo_input, alias, see manual for more info.

- Now you can send many commands in a single line (like tintin++) (REGISTERED)

- ARexx port! (REGISTERED)

- Increased output window scroll speed!

- Included in the archive compiled version for faster processors.

- Macro menu.

- Many bug fixed.

v 2.7 (9-3-97)

- New preferences program that let you configure easily all the AmiMUD features, uses MUI but you can still configure AmiMUD with your favourite text editor if you don't like it.

- Includes, now an AmiMUD configuration may be splitted in various modules (for instance a screen module).

- Aliases!

- Speedwalk playing/recording that let you record on a file your movements and use them again if you need it. (REGISTERED)

- Complete, an easy way to type less text via TAB completion. (REGISTERED)

- Fast speedwalk, now you can handle speedwalk like tt++ does (without speedwalk char) (REGISTERED)

- Partial telnet support, actually works only with some telnetd...

- Improved the tick computation process (maximum_ticklength).

- A lot of new configuration keywords (all available also in the prefs program).

- Improved scrolling speed on 3.0+ amigas.

- Crypt code, now you can talk with other players using AmiMUD or Elf for Windows being sure that no one can understand what you are saying! (except who knows the crypt password)

- Now you can merge configurations...

- Fixed a lot of bug :)

v 3.0 (4-9-97)

- AmiMUD is multisession, this means that you don't need anymore to run multiple AmiMUD to use multiple MUDs at once.

- New modular output system.

- New custom output window module that supports ANSI 16 colors very fast with AGA.

- Now AmiMUD can be executed also without a TCP stack running.

- Gag replaces, display a text every time you receive another text.

- MSP support, Music and Sound with the MUDs that support it. (REGISTERED)

- Button window, a shortcut to the most useful macros.

- AmiTCP version, quicker if you use AmiTCP, if you use Miami you should use the INet225 version.

- Even better configurability.

- New USEFUL commands: #if, #math, #system, #suspend, #zap...

- Rewritten and optimized command parser.

- Added a small logo window that is opened at the startup (if the screen has more

than 8 colors).

- Custom variables.

- Many new configuration keywords.

- A lot of bug fixed.

v 3.1

- Now it's possible to change the output window font (if you use the ANSI output module).

- New commands: #unaction, #remmacro, #remaction

- Now you can disable/remove trigger and macros through the "Show Trigger/Macros..." menu options.

- Now you can open/close/use the button window also if you are offline.

- A lot of internal code cleanup.

- New registration method with EURO.

- Fixed a possible cause of crash if you disconnect and reconnect a lot of times.

- Fixed some problems with the ansi output module.

- Updated preferences program to cope with all the new 3.x features.

1.102 Author

Author

You can get always the last beta versions of AmiMUD in my home page:

http://members.tripod.com/~GabrieleGreco/amimud.html

For bug fix, hints, suggestion and REGISTRATIONS mail me at:

gabry@promix.it

If the address is down (it's rare but may occur) use:

ggreco@freenet.hut.fi

ggreco@iol.it

Please note that ggreco@iol.it will not be valid anymore after 30-2-99.

You can contact me also at the following addresses:

Internet: gabry@promix.it

Address:

Gabriele Greco

Via Banchi 12

16030 Uscio (GE)

Italy

You can also found me on the following italian muds:

Tempora Sanguinis: ts.roxybar.it 4000

as Gabriel(IMP)

Lumen et Umbra: mclmud.mclink.it 6000

as Colosso/Mongo

1.103 Registering AmiMUD...

Registering AmiMUD

AmiMUD is shareware, if you use the program after 1 month of testing you HAVE to register. If you register the program you will receive through e-mail or snail mail a keyfile that will remove the opening requester, you will also receive an e-mail notifying if new versions are available.

Registration fee:

EURO 10

US dollars 15

DM 20

Italian lire 20000

UK pounds 10

French Francs 60

You can send me the money enclosed in an envelope or you can use an international money order. For italians the best choice is a "vaglia postale".

If you send the money in the envelope please send me that in one of the currencies I listed...

Send the cash/postal money order to my address , I will respond as soon as possible.

In the envelope enclose at least you e-mail address or your postal address (e-mail preferred).

I accept also as "registation fee" the registered version of a shareware program YOU write.

Since 1-1-99 it's also possible, and safe, to register making a cheque in EURO (10 euro) and sending it to me in a closed envelope, if the cheque is intested to Gabriele Greco and cannot be trasferred to other people it's a safe and quick way to register!

Thank you for supporting shareware concept!

1.104 Things to do...

Future features

- Possibility to use AmiMUD as a terminal program with serial.device and similar devices.

- Automapping

If you have an idea you'll like to see implemented in AmiMUD mail it to me .

1.105 Thanks to...

Thanks

Translators:

- Marnix van den Bergh (Dutch)

- Thoresten Ernst (German)

- Martin Strandh (Swedish)

- Roger Kristiansen (Norwegian)

I would like to thank the following persons for suggestions/bugfix/betatesting:

- Riccardo Zironi

- Stuart Logan

- Agostino Fanti

- Gianluca Porta

- Francesco Munda

I would like to thank for the source code of his Elf's crypt code:

- Alfredo Milani-Comparetti

And also:

Emmanuele Benedetti

Giuseppe Caggese

Because they create Lumen et Umbra the mud that make me decide to write an Amiga MUD client...

1.106 AmiMUD ARexx interface

AmiMUD ARexx interface

Please note that the Arexx port is available only in the registered version of AmiMUD.

Actually the command's arguments have to be included in double quotes if they contains spaces, single quotes if the case is important and may be passed also without quote if the case is not important.

Examples:

am_connect medievia.com 4000

The arguments that AmiMUD will receive will be: MEDIEVIA.COM, 4000 and this will work because addresses aren't case sensitive.

am_addtrigger 'you are hungry' 'rem bag\nget bread bag\neat bread\nwear bag\n'

This will not work. The correct use is:

am_addtrigger '"you are hungry"' '"rem bag\nget bread bag\neat bread\nwear bag\n"'

This because the arexx interpreter will send you, are, hungry, rem, bag\nget.... as different arguments.

You can also use:

'am_addtrigger "you are hungry" "rem bag\nget bread bag\neat bread\nwear bag\n"'

Also this syntax is correct.

You can use single quotes when you need to use only a word. Look at the example script for more examples.

These are the commands actually supported by AmiMUD, if you need a particular command for a script send me a mail, and (if useful) I'll implement it.

am_addmacro

am_addtrigger

am_connect

am_disconnect

am_echo

am_getscreen

am_getvar

am_log

am_load

am_include

am_quit

am_send

am_setvar

am_waitfor

am_waittriggers

See example.amud and example2.amud for more infos.

Remeber to use:

address 'AmiMUD.1'

...in your arexx script to tell ARexx interpreter to send the commands to amimud! (Obviously you can send the scripts also to AmiMUD.2, .3, .4...)

Hint: if you want to assign an hotkey to an ARexx script just insert the #rx command in a macro:

macro=myscript

hotkey=F3

#rx myscript.amud\n

1.107 AmiMUD ARexx commands

Command: am_addmacro

Syntax: am_addmacro NAME/A,MACRO/A,HOTKEY

Use

This command will add a macro to the AmiMUD session you have adressed the command to, like the macro configuration keyword.

NAME will be the name of the macro, MACRO the text (use double quotes) and HOTKEY the hotkey the macro is assigned to (not required).

Example:

am_addmacro '"getbag $0"' '"rem bag\nget $0 bag\nwear bag\n"'

1.108 AmiMUD ARexx commands

Command: am_load

Syntax: am_load FILENAME/A

Use

This command allows you to load a configuration (and overwriting the existing one).

Example:

am_load s:med.prefs

/* Loads s:med.prefs and REPLACES the current settings */

See also: am_include

1.109 AmiMUD ARexx commands

Command: am_include

Syntax: am_include FILENAME/A

Use

This commands include a file containing some configuration keywords in the

actual configuration, overlapping options will be replaced but triggers,

macros... will be kept.

See also: am_load

1.110 AmiMUD ARexx commands

Command: am_addtrigger

Syntax: am_addtrigger TEXT/A,ACTION/A,REXXTRIGGER/S

Use

This command will add a trigger to the amimud session you have addressed the command to, like the trigger configuration keyword.

TEXT is the text that active the trigger, ACTION is the action to perform, REXXTRIGGER is a switch to use with the REXXONLY switch of am_waittriggers .

Example:

am_addtrigger '"you are hungry"' '"rem bag\nget bread bag\neat bread\nwear bag\n"'

1.111 AmiMUD ARexx commands

Command: am_connect

Syntax: am_connect HOST/A,PORT/N

Use

If not connected this command will let you connect to the specified HOST, PORT will be the port to connect to, it's optional, if not specified the default value is 23.

Example:

am_connect medievia.com 4000

1.112 AmiMUD ARexx commands

Command: am_disconnect

Syntax: am_disconnect

Use

This command will disconnect you from the mud you are connected to.

Example:

am_disconnect

1.113 AmiMUD ARexx commands

Command: am_echo

Syntax: am_echo TEXT/A

Use

This command is very useful for complex script (look at example.amud to see how it's used). It print the TEXT specified to the output window without sending it to the mud.

Example:

am_echo '"This text will be printed only on the output window!"'

1.114 AmiMUD ARexx commands

Command: am_getscreen

Syntax: am_getscreen

Use

This command is very useful if you want to know the screenname of AmiMUD. It will return in the ARexx variable screenname the value.

Example:

am_getscreen

say 'AmiMUD is opened on:' screenname

1.115 AmiMUD ARexx commands

Command: am_getvar

Syntax: am_getvar NUMBER/A/N,VARIABLE/A

Use

With this command you will copy the contents of variable $NUMBER to the ARexx variable VARIABLE. NUMBER must be 0..9.

Example:

am_getvar 0 myvar

say 'Variable $0 contains:' myvar

1.116 AmiMUD ARexx commands

Command: am_log

Syntax: am_log LINES/A/N,VARIABLE/A,WAITFOR/K

Use

This command let you put in a variable the contents of the next LINES lines. The contents are stored in a stem, the base variable is the one named VARIABLE. If you use the WAITFOR keyword the log will start from the line after the one where <keyword> is found, otherwise it will start at the first line received from the mud after the command is performed. Please note that the command will return the control to the script only when it receives the specified number of lines or if amimud is terminated. Please note that if a script is logging already with the command am_log another call to this command will cause an error (the command will return with code 5). Please note that sometimes due to the division in TCP packets a line may be splitted it two parts.

Example:

am_log 5 BUFFER

do i=0 to 4

say "line" i "is" BUFFER.i

end

1.117 AmiMUD ARexx commands

Command: am_quit

Syntax: am_quit

Use

This command will quit AmiMUD.

Example:

(from a shell)

rx "address 'AmiMUD.1' am_quit"

1.118 AmiMUD ARexx commands

Command: am_send

Syntax: am_send TEXT/A

Use

This will send the text TEXT to the mud. It can contains also multiple lines use separator character to divide them.

Example:

am_send '"rem bag;get sword bag;wear bag;wield sword"'

1.119 AmiMUD ARexx commands

Command: am_setvar

Syntax: am_setvar NUMBER/A/N,TEXT/A

Use

This command will put TEXT in the variable $NUMBER. NUMBER must be 0..9.

Example:

am_setvar 1 '"Hello world!"'

1.120 AmiMUD ARexx commands

Command: am_waitfor

Syntax: am_waitfor STRING/A

Use

This command is very useful in scripts. It HANGS the script until the text STRING is received from the mud or via the #send command. If the text TEXT is found the command will also return the line containing TEXT in the variable LINEBUFFER.

If AmiMUD is terminated all waitfor commands will be replied.

Example:

am_waitfor 'death'

say 'Someone is dead, this is the line:' linebuffer

1.121 AmiMUD ARexx commands

Command: am_waittriggers

Syntax: am_waittriggers REXXONLY/S

Use

This command will HANG the arexx script until one of the trigger defined is actived by the mud or via the command #send . AmiMUD will put the actived trigger number in the ARexx variable TRIGGER and the line that actived the trigger in the variable LINEBUFFER.

If the REXXONLY switch is specified the command will be satisfacted ONLY if the trigger actived has been defined through the rexx command am_addtrigger with the switch REXXTRIGGER. This can be useful if you want that your script reacts only to SOME triggers.

If AmiMUD is terminated all am_waittriggers commands will be replied.

Example:

am_waittriggers

say 'The trigger actived is N.' TRIGGER ' and the string is ' LINEBUFFER

For an example of REXXTRIGGER/REXXONLY look at the example.amud example script.

1.122 Configuration keywords

Keyword: nologo

Usage

This keyword disables the logo window that is open at amimud startup.

1.123 Configuration keywords

Keyword: startup_pen

Usage

This keyword is useful if you are using AmiMUD in ANSI mode with output/console, to set the right pen at the startup. It's also useful in conjunction with color_reset .

Example:

startup_pen=7

; Set default pen to white (with ANSI colors)

See also: color_reset , module

1.124 Configuration keywords

Keyword: color_reset

Usage

This keyword can be used to reset the color of the text to the value of the startup_pen after an highlight .

See also: startup_pen , highlights

1.125 Configuration keywords

Keyword: input_width

Usage

With this keyword you can set the width of the input window, by default the window will be as large as the screen where AmiMUD is opened. This may be not good with high resolution such 1024x768 or better.

Example:

input_width=500

See also: input_window_x , input_window_y

1.126 Configuration keywords

Keyword: input_window_x

Usage

This keyword set the left edge of the input window, if you set this keyword the input window repositioning will be disabled, so if you don't like the input window repositioning just specify: input_window_x=0.

Example:

input_window_x=20

See also: input_window_y , input_width

1.127 Configuration keywords

Keyword: buttonleft

Usage

This keyword let you set the left edge of the button window.

Example:

buttonleft=10

See also: buttontop , macros

1.128 Configuration keywords

Keyword: buttontop

Usage

This keyword let you set the top edge of the button window.

Example:

buttontop=100

See also: buttonleft , macros

1.129 Configuration keywords

Keyword: speed_delay

Usage

This keyword is useful in MUDs like Medievia that don't allow the pretyping of lot of directions, since they accept only a certain amount of commands per second. speed_delay let you specify a delay between the single directions in seconds so that you'll can use AmiMUD speedwalks ALSO with MUDs like Medievia.

Default value: 0

Example:

speed_delay=1

; Every speedwalk direction will be delayed by a second.

See also:

1.130 Configuration keywords

Keyword: interleaved

Usage

The AmiMUD 3.0 custom ANSI output module uses bitmap masking to move less datas when you scroll the screen, so interleaved is now disabled by default (to gain scrolling speed). To enable it (useful if you use output/console as output module) use this keyword.

Example:

interleaved

See also: module

1.131 Configuration keywords

Keyword: replace

Usage

Replaces let you gag some text and display another text.

Example:

replace=Adept

41-50

Adept Mongo the troll destroyer

will became...

41-50 Mongo the troll destroyer

1.132 Configuration keywords

Keyword: ansi16

Usage

This keyword enable 16 colors ansi mode if available. Ansi 16 actually needs output/ansi, AmiMUD opened in a screen with at least 16 FREE colors (with AmigaOS 3.0) or in a custom screen (with AmigaOS 2.0 Ansi 16 works only in a custom screen). Anyway the new output module will fallback to previous modes if the screen isn't enough depth. I suggest AGA users to open a 32 color screen for ANSI 16 mode, graphic cards users should have no problem with 256 chunky modes (tested with CyberGraphX and Picasso96).

Example:

ansi16

See also:

1.133 Configuration keywords

Keyword: review

Usage

This keyword is used to disable the review buffer if you are using output/ansi as output module. This speed up a lot the scrolling speed and my be useful in certain circumstances.

Example:

review=no

See also: max_review

1.134 Configuration keywords

Keyword: max_review

Usage

This keyword is useful if you use the new custom output module or if you are using kingcon as console module. It sets the maximum size of the review buffer.

Default value: 300

Example:

max_review=1000

;Set the review buffer to 1000 characters.

See also: review

1.135 AmiMUD internal commands

Command: alias

Usage

Syntax: #alias "name" "definition"

This command let you define a new alias without using the mouse.

Example:

#alias "ct" "clantalk"

Is equal to:

alias=ct

clantalk

See also: aliases

1.136 Configuration keywords

Keyword: module

Usage

AmiMUD 3.0 is modular, for instance the output window is a shared library with a defined API, if you don't like the available output modules you can build own (obviously you need to be a programmer) using the FD and includes distributed in the AmiMUD archive. Then you simply have to use "module" in your configuration file to make amimud load the correct loader.

Default value: output/ansi

Example:

module=output/console

;Use the console as output Module (As in AmiMUD 2.x).

See also: ansi16 , max_review

1.137 AmiMUD internal commands

Command: connect

Usage

Syntax: #connect host [port]

This command is equal to the "Connect" menu item, you have specify the host to connect to and the port (if the port isn't specified the connection will be by default to the port 23). The host may be both an IP or a normal internet address.

Example:

#connect medievia.com 4000

1.138 AmiMUD internal commands

Command: zap

Usage

Syntax: #zap

This command close the connection without asking for confirmation.

1.139 AmiMUD internal commands

Command: suspend

Usage

Syntax: #suspend

This command is very useful when you need to read something scrolled up using the review buffer. If AmiMUD receive new output text it normally scrolls to the bottom of the review buffer, with #suspend you can suspend the output until another #suspend. Obviously this is useful only if you are using an output module that supports the review buffer.

Example:

macro=sus

hotkey=F10

#suspend\n

(With F10 you can suspend/resume output)

1.140 AmiMUD internal commands

Command: music

Usage

Syntax: #music

To use this command you need to register AmiMUD and also GMPlay 1.2+ installed, gmplay: must point to the directory where the GMPlay executable is placed. This commands plays MIDI files, it's used in the MSP support code. You can find gmplay in any aminet mirror in the directory mus/midi.

Example:

#music dh0:midi/starwars.mid

See also: MSP

1.141 AmiMUD internal commands

Command: sound

Usage

Syntax: #sound

This command is available only if you are a registered user of AmiMUD and if you have WB 3.0+, and plays sound through datatypes. So with this command you can play ANY sound you have a datatype for. It's used also by the MSP module.

Example:

trigger=death cry

#sound sounds/death.wav\n

See also: MSP

1.142 AmiMUD internal commands

Command: if

Usage

Syntax: #if <expression> then <statement> [else <statement>]

This command makes AmiMUD triggers and Macros very powerful, using #if you can do macros that do complex tasks that previously were only available with arexx.

Expression must be in the form:

variable<comparison>variable

variable<comparison>value

<comparison> may be:

=, <, >, <=, >=, <>

variable must be a variable of AmiMUD (custom or simple) and value can be a number or a string.

Examples:

#if $myhit<200 then #healmyself

#if $tank=Exodus then #healmyself else #heal $tank

1.143 AmiMUD internal commands

Command: system

Usage

Syntax: #system <command string>

This command let you execute an external AmigaDOS command, it will open a shell for the output if necessary.

Example:

#system echo >ram:file $0

Prints in the file "ram:file" the contents of the variable $0, very useful in macros.

1.144 AmiMUD internal commands

Command: system

Usage

Syntax: #system <command string>

This command let you execute an external AmigaDOS command, it will open a shell for the output if necessary.

Example:

#system echo >ram:file $0

Prints in the file "ram:file" the contents of the variable $0, very useful in macros.

1.145 AmiMUD internal commands

Command: reemmacro

Usage

Syntax: #remmacro <macroname>

This command disable a single trigger, executing it again with the same parameter will enable the trigger again, the parameter must be the COMPLETE trigger activation string, you can do the same thing also with the menu option "Show triggers...".

Example:

#remmacro "hungry"

Remove a macro defined for instance as:

#macro "hungry" "get bread bag\neat bread\n"

See also: macro creation

1.146 AmiMUD internal commands

Command: unaction

Usage

Syntax: #unaction <trigger activation>

This command disable a single trigger, executing it again with the same parameter will enable the trigger again, the parameter must be the COMPLETE trigger activation string, you can do the same thing also with the menu option "Show triggers...".

Example:

#unaction {ou are hungry}

#unaction "ou are hungry"

Two ways to disable the "hungry" trigger

1.147 AmiMUD internal commands

Command: rematction

Usage

Syntax: #rematction <trigger activation>

This command REMOVE a single trigger from the trigger list, the parameter must be the COMPLETE trigger activation string, you can do the same thing also with the menu option "Show triggers...".

Example:

#rematction {ou are hungry}

#rematction "ou are hungry"

Two ways to REMOVE the "hungry" trigger.

See also: unaction to see how disable a trigger.

1.148 AmiMUD internal commands

Command: math

Usage

Syntax: #math var=<expression>

Not yet implemented.

1.149 Music Sound Protocol

Music Sound Protocol

AmiMUD supports the Music Sound Protocol, this protocol is also supported by MudMaster and ZMud and by a growing number of muds, it has been developed from ZMud's author and Medievia implementors.

To use MSP on Amiga you need:

AmigaOS 3.0+

AmiMUD 2.10+ REGISTERED

wav.datatype installed

GMPlay 1.2+ installed

gmplay: correctly installed

modules/msp included in your AmiMUD configuration.

Last but non the least: the sound/music sets of the mud you are playing that supports MSP.

Practically when you are playing a MSP-aware mud with MSP enabled the MUD will send to you some extra datas to tell the client what sounds/musics to play in certain situations.

Obviously you need to have these sounds ALREADY on your hard disk, often you can find them in the MUD web page. The archive with the sounds have to be dearchived with paths in the AmiMUD directory.

If some sounds you need are not on your HD AmiMUD will not complain about it, only when you close the program it will show you the list of the missing wav/midi.

See also: #sound , #music

1.150 Output Modules

Output modules

AmiMUD 3.0 has a modular output system, in the "output/" directory there are actually two modules, console and ansi. The first is the old system, the second is a new output module coded to support ANSI 16, OS 3.0 palette handling and to enhance the scroll speed in ANSI modes. Maybe in future I'll write other modules, you can write your own if you have enough programming skills using the includes provided in this archive.

Please note that the new AmiMUD output module is quicker than kcon and supports 16 colors.

On my A4000 (040/25) this is the speed result of a comparison:

#cycle 50 #help

(on my 1024x768 WB with a window of 530x400 on PicassoII with CGFX):

AmiMUD ANSI: 1:30

KingCON: 2:05

ViNCEd: 1:40

CON: 1:35 (Without review buffer)

(on a custom DOUBLE-NTSC screen 640x400 32 colors, full screen window):

AmiMUD ANSI: 4:00

KingCON: 4:50

ViNCEd: 5:10

CON: 5:00 (without review)

See also: ansi16 , module , interleaved , review