MUX 2.3 Manual

Stephen Dennis (AKA Brazil)

2003-NOV-03

Solid Vertical Domains, Ltd.

PMB 624
11410 NE 124TH ST
KIRKLAND WA 98034-4305

Introduction

1.1 Overview

This document is broken down first into views or perspectives that cater to a particular type of reader at a particular experience or skill level and look at the subject under discussion from a particular angle. Some redundancy between perspectives is unavoidable.

1.2 Let's Begin

There are perhaps two good ways to begin:

1. find an expert to answer all your questions and prompt you for the questions you would be asking if you even knew where to start.

2. Do some research on your own and learn by patient trial-and-error.

There are advantages and disadvantages to both approaches. I recommend that you rely on yourself and use the experts sparingly in specific areas as a way to accelerate yourself through those specific issues. Always be prepared and ready to find your own way.

There are only so many experts and their expertise may be narrow. Even experts in one area are known to disagree. You have little way of knowing whether someone is an expert in a particular area until you are already all wrapped up in their advice and the political baggage they may carry with them. Solitary reading and trial-and-error has its own problems. The documentation that is available is both limited and out of date. In order to get a good idea of what you are dealing with, you may need to stitch together information from several sources. It is easy to become stuck on a small point and waste a lot of time. In addition, these game servers are large enough and complicated enough that experimentation without the advantage of some experience or history is nearly impossible.

Building

4.1 Basic Building

4.1.1 Overview

The physical world is impossibly complex to represent or model, so any useful model of the physical world is necessarily a vast simplification. Arcade-style video games demonstrate the limits of what is currently possible with their use of polygons and textures to render rooms and everything else in the room. In these games, rooms are described geometrically by first creating a box to separate the inside from the outside, and then by modifying the size, number, and position of the walls until the environment becomes sufficiently interesting and immersive. In a text-oriented game, the model is even simpler. Words take the place of polygons and textures, and we rely on the reader's imagination to add the details for themselves.

The database consists of four different types of objects: *rooms, exits, players,* and *things*. There is a fifth object type called *garbage,* but for the purpose of this discussion, we will ignore it.

The first type of object is a *room*. A *room* is like a stage. *Players* and *things* are located within *rooms,* and *rooms* connect to each other via *exits*. A *room* does not necessarily correspond to a room in the physical world. For example, it may be more interesting to divide a large stadium-sized room into several *rooms*. Likewise, outside areas without walls are still modeled as *rooms*. There is also no requirement that all rooms model the same volume of space. For example, walking through a bazaar at ground level may involve several *rooms,* whereas walking along the rooftops may only demand a single *room*.

Exits act as the doors between rooms. Its name is the doorknob that activates it. While the storyline may benefit from standing in a doorway, there is no such thing from the point of view of the database. A *player* or *thing* is located in only one location at a time. You also may not expect that you will need two *exits* to model a typical door in the physical world – one *exit* for each direction.

This leaves us to describe *players* and *things*. Administratively, a *player* it is an account that allows you to connect to the game and be capable of owning other objects, but otherwise, it is very similar to a *thing*. A *thing* is located inside a *room,* another *thing,* or a *player*. Therefore, *players* and *things* have inventory.

Together, these four objects provide sufficient flexibility to facilitate stories.

New to MUX

2.1 Frequently Asked Questions (FAQ)

2.1 What is a Text-Based Game Server?

It is a program that allows a multitude of people to connect to the same text-based environment and interact with each other and the environment. The administrator runs this server on top of an operating system and the server in turn allows other people to connect.

These layers are written in a particular computer language. Most operating systems are written in C/C++ with some assembly language and some higher-level scripting or macro languages.

The term hardcode usually means the approximately 93,000 lines of C/C++ source code that make up the server itself. All of the servers in the MUSH family are written in C except MUX 2.0 and later which are written in a combination of C and C++.

However, the term can also be used more generically to refer to anything written in C/C++ or even to anything written in a general purpose programming language.

All of the Softcode Global Packages are written in a mostly universal language that does not have a name.

You could call it MUSHcode, but then you are probably referring to the commands and functions of a specific version of a specific server. You could call it softcode, but then you are also probably including non-MUSH family languages further away and more incompatible. For the purpose of this document, we will call it softcode with the understanding that we expect it to run on a particular set of closely related servers and on MUX 2.3 in particular.

2.2 I am using Windows. Why does the log say it cannot find its files?

You are probably running the netmux.exe file directly from the GUI by double clicking on the icon. The server needs the current directory to be the game sub-directory. All the file names are relative to that point, but when you double-click on the netmux.exe program directly, the current directory might be the game/bin or tinymux/release directory, or it might be somewhere else. The current directory is too undefined for the game to be run this way.

You should open a Command Prompt window (that is, an MS-DOS box), change to the drive and 'mux2.3game' sub-directory where you extracted the distribution, make sure the netmux.exe program are in the 'bin' sub-directory, and then run the ./Startmux script.

2.3 How to Begin Running MUX

The first thing you will need is some server hardware with some flavor of Unix or Windows running. It is more than a little helpful if this server has a dedicated Internet or Intranet connection, as without it, you will be using the server all by yourself.

You need to find the latest known-good distribution from www.tinymux.org and download it. If you are using a web browser, it will download it in 'binary' mode for you automatically, however, if you use an FTP client manually, you will need to be sure to put the transfer into binary. Even between Unix machines, non-binary transfers can be problematic.

Here is a typical manual FTP conversation:

```
$ ftp ftp.tinymux.org
Connected to ftp.tinymux.org.
220 sdennis5 Microsoft FTP Service (Version 5.0).
Name (ftp.tinymux.org:sdennis): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password: sdennis@svdltd.com
230 Anonymous user logged in.
ftp
binary
200 Type set to I.
ftp
cd TinyMUX/tinymux-2.3/alpha/7
250 CWD command successful.
ftp
ls *.tar.gz
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
mux-2.3.0.7.unix.tar.gz
mux-2.3.0.7.win32.bin.tar.gz
mux-2.3.0.7.win32.src.tar.gz
226 Transfer complete.
85 bytes received in 0.03Seconds (4.48Kbytes/sec)
ftp
get mux-2.3.0.7.unix.tar.gz
200 PORT command successful.
150 Opening BINARY mode data connection for mux-2.3.0.7.unix.tar.gz(854541 bytes).
226 Transfer complete.
854541 bytes received in 0.0888 secs (9.4e+03 Kbytes/sec)
ftp
quit
221
$
```

The suffixes on the end of the filename tell you which tools were used to produce the file and in which order. Gzip handles '.gz' files, and tar handled '.tar' files. However, tar is able to deal with gzip-ed files itself. So in order to unpack this distribution, we do the following:

```
$ tar xvzf mux-2.3.0.7.unix.tar.gz
```

These two commands will create a sub-directory called 'mux2.3' with all the required files in the right places. So, in order to build the game, continue as follows:

```
$ cd mux2.3/src
$ ./Configure
$ make
```

On some system 'gmake' is required instead of 'make'. At the end of this process, you should have executable programs built in the mux2.3/src directory with links made to the mux2.3/game/bin directory. A link in Unix parlance is another name for the same file contents.

At this point, if you don't have an existing database to convert or import, and if you don't have any configuration changes, you could simply run the server and it would create a minimal database and let you connect to port 2860. This can be done as follows:

```
$ cd ../game
$ ./Startmux
```

The Startmux script will do a few things like index the help files and then return to the $-prompt again. You can verify that the server is running as follows:

```
$ ps u
```

You will see two processes, a netmux game process and a slave process that handled reverse DNS lookups and the identd protocol for hosts that support it. To connect to the game, do the following:

```
$ telnet localhost 2860
```

You will see the connect.txt welcome screen contents and then the server will wait for you to connect or logon. With a minimal database, there is only one player defined: Wizard or #1. Moreover, there is only one room defined: Limbo or #0. The initial password for #1 is something of a standard everywhere. You should change it right away with the @password command. Do the following:

```
Connect Wizard potrzebie
@password potrzebie=XXXXXXXX
```

where XXXXXX is a password of your own chosing.

Use 'help @password' to learn more about the @password command, and 'help topic' if you need help on a command, function, or class of functions.

finally, at some point, you will want to bring the game down in an orderly way. People have stepped over the power cord and other non-orderly ways of bringing the game down, but @shutdown is recommended instead as follows:

```
@shutdown
Goodbye.
Connection Lost.
$
```

The database is quite rugged, but there are no guarantees if you habitually abuse it.

finally, buy a UPS. Backup, off-line. Do it. If you use Windows NT, configure it, and then leave the server alone. WinNT works best if you give it good hardware, good drivers, a set of services to run, and then leave it alone to serve.

Configuration

3.1 Basic Configuration

The game server reads from a configuration file when the game server starts or restarts, but changing a configuration file while the game is running does not change a running game's behavior. To accomplish that, look at the `@admin` or `@restart` commands.

The name of the configuration file is given in the ./Startmux script on the command line and it's netmux.conf for Unix installations and netmux.conf forWin32 installations.

If you look at this file, you will see several lines with 'include' in them. Including a sub-configuration file is the same as if the contents were present in the top-level configuration file.

Everything in a configuration file is handled as if #1 (Wizard) did it.

3.2 Site-Banning

3.2.1 Overview

The following configuration options are used to restrict a player's access based on their IP address:

- `Register_site` allows existing players to connect to their characters, but does not allow players from the specified subnet to create new characters.
- `Forbid_site` disallows any connection and does not even show the welcome screen.
- `allow_guest_from_registered_site <yes|no>`. Default is `yes`. This configuration option allows the `register_site` subnet list to control the ability of guests to connect from those sites as well.
- A future version of MUX will also include the `guest_site` configuration option. That option allows and requires the administrator to manage the guest restrictions separately.

For register-only games, the administrator typically uses `register_site` to encompass the entire Internet (0.0.0.0 0.0.0.0). In this case, it is more convenient to leave `allow_guest_from_registered_site` at its default value of `yes`, and then use `guest_site` or `forbid_site` to control problem guests.

For open games that allow characters to be created at the initial welcome screen, it is more convenient to turn `allow_guest_from_registered_site` off, and then use `register_site` exclusively to control problem players and problem guests. This allows players from that subnet to connect to existing characters, but disallow them from creating new ones or from connecting as guest.

So, the usefulness of the `forbid_site` configuration option is already narrow, and it continues to narrow further.

Also, remember that none of these configuration options are saved between `@shutdown` or `@restart` and they need to be added to the configuration file or appear on #1's STARTUP attribute. However, I would recommend against putting these commands on the #1's STARTUP attribute.

3.2.2 Guessing the Subnet

In order for the above restrictions to be useful, you need to guess the player's subnet, and realize that a single player may be connecting from different subnets and therefore each subnet that they connect from must be dealt with separately.

Each shell account that a player uses to connect to the game will be on it's own subnet. In some ways these are easy to deal with because these servers can be expected to have a single static IP addresses, but it's not guaranteed.

Large Internet Service Providers (ISPs) like AOL have subnets for each service access number., so if someone calls a particular local phone number and logs into AOL, they are assigned an IP dynamically from a large static pool. If they are willing to make a long-distance phone call to another service access number, they may get an IP from a different large static pool.

You must deal with the entire static pool for a particular service access number. If they call a different service access number, then that will represent a yet another subnet that needs to be handled separately as it will probably not be adjacent numerically to their primary one.

3.2.3 Example of Guessing

Not that I want anyone to restrict my subnet, but let us take it as an example. If I connect to a game, you will usually see 207.153.135.132 as the host name if the server is running on Unix, and you will see SDENNIS5 if the server is running on Win32[1].

You might also see me connect from svdltd.com or mirror.svdltd.com, and so, for the sake of this example, you have three IP addresses that are close and therefore they are probably within the same subnet. It is a fair guess.

```
ping -a svdltd.com
Pinging svdltd.com [207.153.135.130] with 32 bytes of data.
...
ping -a 207.135.135.132
Pinging SDENNIS5 [207.153.135.132] with 32 bytes of data.
...
ping -a 207.135.135.134
Pinging 207.153.135.134 with 32 bytes of data.
...
```

And now, we need to convert from decimal to binary as follows:

```
'pack(130,2)' gives '10000010'.
'pack(132,2)' gives '10000100'.
```

---

[1] Unix and NT both do reverse-DNS lookups. However, if no DNS PTR entry exists, Windows does at extra step. It will query the client machine for a NETBIOS name (which is also known as your computer name, and can be found under your network settings). This NETBIOS name is unique across an NT domain and across a NETBEUI subnet, but in a TCP/IP context, it is not necessarily unique, and it is not possible to map from that NETBIOS name back to a numerical IP.

```
'pack(134,2)' gives '10000110'.
```

So, the subnet is at least the lower three binary digits because those three digits are the only ones that are changing. Let's try a different fourth bit and see if the hostname changes. That is, `'unpack(10001001,2)'` gives `'137'`.

```
ping -a 207.153.135.137
...
request timed out.
...
```

And, for my subnet, it's hard to find another computer nearby. It turns out that the lower 4 binary digits are within my subnet, but you could not guess that unless one of my computers had allocated one of the higher IP addresses from the local DHCP service. DHCP tends to assign IP numbers from lower to higher, and it also tends to renew the same IP address to the same device.

Since, there wasn't a computer nearby, you might choose the lower four bits anyway, but for this example, we'll proceed with the assumption that the subnet is defined uniquely by the upper 29 bits and that the lower 3 bits are used for devices within the subnet.

3.2.4 What is the Base IP and Mask?

The Base IP address for the above example is determined by zeroing out lower three bits of an IP address out (e.g., 1000.0000 binary or 128). The Base IP address is 207.153.135.128.

The mask is determined by setting all bits 'one' except for the lower 3 bits which are 'zero' (e.g., the last byte of the IP address is 1111.1000 binary or 248). The mask for our example subnet is 255.255.255.248.

Another way of specifying the mask is to use a slash followed by a number from 0 to 32 ('/N'). N is the number of 'one' bits in the mask, or equivalently, how many bits of the base IP address are significant (i.e., 29 in our example).

3.2.5 MUX commands

You can apply the `register_site` restriction to the desired subnet with either of the following commands, but the second format is the preferred format:

```
@admin register_site=207.153.135.128 255.255.255.248
@admin register_site=207.153.135.128/29
```

The number at the end (i.e., 29) represents the number of significant bits in the base subnet address.