Unix Shell Admin Guide

Created by Jamdog for tbaMUD Users

Contents

Installing your tbaMUD

The first step on your journey to MUD Administration must begin with downloading the latest version of the tbaMUD codebase. tbaMUD is constantly changing and evolving to provide MUD administrators with a codebase that is fully featured while being as adaptable as possible. This means that a new version is released regularly, with patches to update older versions. The latest version is always available from the tbaMUD.com homepage.

Note that for the purposes of this installation guide, the filename tbamud will be used when the filename will usually have a version number after it. So, if you downloaded version 3.58, the filename would be tbamud-3.58.tgz, rather than tbamud.tgz.

Once you have downloaded tbaMUD, you will need to put it onto your server. If you aren't running the MUD on your own PC, then you should be provided with FTP details by your server provider. Use these details in an FTP program to connect to your server. You can then upload the tbamud.tgz file onto your server. If you have your own Unix PC, then save the file where you would like you MUD to be stored. If you are using Cygwin on a Windows-based PC, then you should save the file into the **C:/cygwin/home/<*username*>/** folder (where C: and <username> are dependant on your system).

You should then log into your shell. If you are using a remote server, you will need to use a program like PuTTY or telnet to do this. If you are using Cygwin, then just start the Cygwin program. Once in the shell, navigate to the location of your new file, and decompress it using the following command (type the yellow text only, because the grey text is the prompt):

```
bash $ gunzip < tbamud.tgz    | tar xvf -
```

If you are using cygwin, or your own Linux PC as a server, you may be able to use some other software to decompress the file, such as WinRAR.

You will need to adjust the .tgz filename depending on your version of tbaMUD. You can then verify that the new folder has been created using the **ls** command:

```
bash $ ls
tbamud.tgz        tbamud
bash $
```

Once your new MUD has been decompressed, you will need to configure it to run on your server. The configure script does this automatically for you by detecting your servers software versions and capabilities. To run the script, enter your new tbamud folder using the **cd** command, then run the script:

```
bash $ cd tbamud
bash $ ./configure
```

The configure script will check your computer or server for various settings and you will see the list of checks. You do not need to do anything and seeing several No's is normal. When the configure script has completed, it is time to compile your new MUD for the first time. Change to the src folder, and simple type make:

```
bash $ cd src
bash $ make
```

Your new MUD should compile error-free and warning-free. Once it is compiled, you are ready to start up the MUD.

Running tbaMUD for the first time

If you have installed, configured and compiled your MUD correctly, then it is time to boot up the MUD and test that it works. If you are still in the src folder, then move back up to the tbaMUD main folder, and run the MUD using the autorun script:

```
bash $ cd ..
bash $ ./autorun &
```

The autorun script not only runs the MUD for you, but sets the port number (which is usually 4000 by default), and restarts the MUD if it crashes, creating a crash log for you.

If you wish to change the port number for your MUD, you can edit the autorun script in any text editor.

You should now be able to log into your MUD. On a remote server, just use the supplied address and port 4000 (unless you changed the port). If you are using your own PC, then the server name should be **localhost**.

If you cannot connect to your MUD, then your first step should be to check the end of the syslog file, and look at running processes to ensure that your MUD is still running. Checking the syslog is done with the **tail** command, and running processes can be checked with the **ps** command:

```
bash $ ps uwx
      PID    PPID    PGID     WINPID  TTY  UID    STIME COMMAND
     1460       1    1460       1460  con 1006   Jan 17 bash
     6080    1460    6080       5564  con 1006   Jan 20 sh
     2584    5916    5916       5980  con 1006 10:56:41 circle
     4996    1460    4996       3532  con 1006 18:32:27 ps
bash $ tail syslog
Jan 23 10:56:44 :: Boot db -- DONE.
Jan 23 10:56:44 :: Signal trapping.
Jan 23 10:56:44 :: Entering game loop.
Jan 23 10:56:44 :: No connections.  Going to sleep.
bash $
```

In the above example, the MUD is running normally. You can see the circle process is running, and looking at end of the syslog file shows that the MUD is in 'sleep' mode, which happens when no players are connected.

If you cannot see the MUD process running, or your syslog shows a SYSERR message, then your MUD may have crashed. In this case, you should refer to the section of this guide called “How to deal with and debug MUD crashes”.

Logging into your MUD for the first time

It is important that the MUD owner be the first person to log into the MUD. This is because the first character to be created is automatically given Implementor status, and therefore has total control of the MUD. All characters created after the first one begin at level 1.

```
> who
Immortals
---------
[34 Mu] Jamdog the Implementor

One lonely character displayed.
>
```

Your first job should be to configure your MUD as you want it. This is done by using the **cedit** OLC. Simply type **cedit** to enter the configuration editor menu.

```
> cedit
OasisOLC MUD Configuration Editor
G) Game Play Options
C) Crashsave/Rent Options
R) Room Numbers
O) Operation Options
A) Autowiz Options
Q) Quit
Enter your choice :
```

You should explore each of the option available. A full description of each option can be found in Appendix C at the end of this guide.

Typing **wizhelp** will show you a list of available Imm commands:

```
> wizhelp
;          dig        invis      olist      roomflags  tcopy      vstat
advance    echo       last       pardon     saveall    tedit      wiznet
aedit      export     links      peace      scopy      teleport   wizhelp
at         file       load       plist      sedit      thaw       wizlock
attach     force      mcopy      purge      send       tlist      wizupdate
ban        freeze     medit      qecho      set        transfer   zcheck
buildwalk  gecho      mlist      qedit      show       trigedit   zedit
cedit      goto       mute       qlist      shutdow    tstat      zlist
changelog  handbook   nohassle   rcopy      shutdown   unaffect   zpurge
checkload  hcontrol   notitle    redit      skillset   unban      zreset
copyover   hedit      nowiz      reload     slist      uptime
date       helpcheck  ocopy      reroll     snoop      users
dc         holylight  oedit      restore    stat       vdelete
detach     imotd      olc        rlist      switch     vnum
>
```

You should familiarise yourself with all of these commands by typing **help <*command*>**. Once you are comfortable with what all of these commands do, you are ready to start accepting players.

Advertise your new MUD by adding it to MUD lists, such as TMC, Top MUD Sites, MUDGamers, MUD Lists, FindMUD, MUD Pages, MUD Bytes and any other MUD listing site you can find. This will bring you new players, who you can then encourage to stay with promises of new ideas and enhancements to your MUD. Many of these sites have a voting system – add a link to the voting pages to your new MUDs website (if you have one), as more votes usually means more players.

How to deal with and debug MUD crashes

If your MUD crashes, a file will be created by the autorun script that shows the last few lines of the syslog file at the time of the crash. This file is called **syslog.CRASH**, and will usually give some indication of what caused the crash.

```
bash $ tail syslog.CRASH
Jan 23 10:51:28 :: Loading triggers and generating index.
Jan 23 10:51:28 :: SYSERR: File 'world/trg/1.trg' listed in
'world/trg//index': No such file or directory
Jan 23 10:51:28 :: SYSERR: world/trg/1.trg: No such file or directory
Quick boot mode -- rent check supressed.
Using file descriptor for logging.
autoscript killed Fri Jan 23 10:51:28 GMTST 2009
bash $
```

The example above clearly shows that a necessary world file is missing, but the error could be almost anything. Unfortunately, there are times when a MUD will crash unexpectedly, and the crash log will not show a reason. In this case, it's time to use the **gdb** debug tool.

The gdb debug tool is highly flexible, but generally, you will only use a small portion of it's capabilities. It runs software in a debug mode, and pauses the software when it crashes, allowing you to look at the code, and also the status of any variables at the crash-time.

To start debugging an error, type **gdb bin/circle** which starts gdb, and tells it to use your MUD executable. You will now be given the gdb prompt instead of your normal unix shell prompt.

Next, type **run *<port>*** to run your mud (obviously entering a port number to use, not typing "*<port>*"). You should see all your normal syslog info appear on your screen, and if your MUD isn't crashing during startup, you will eventually see a message saying "No connections, going to sleep."

You should then be able to connect to your MUD on the specified port in the normal way. Your next step is to crash the MUD. When you manage this, the MUD will appear to pause, and you will get a message in gdb, and your gdb prompt will return. You can now start debugging.

```
bash $ gdb bin/circle
GNU gdb 6.8.0.20080328-cvs
Copyright (C) 2008 Free Software Foundation, Inc.
(gdb) run 4000
...
Jan 23 10:56:44 :: Boot db - DONE.
Jan 23 10:56:44 :: Signal trapping.
Jan 23 10:56:44 :: Entering game loop.
Jan 23 10:56:44 :: No connections.  Going to sleep.
Program received signal SIGSEGV, Segmentation fault.
0x00002ab047f6eb30 in strlen () from /lib/libc.so.6
(gdb)
```

Normally, the first command you type will be **bt** (for 'backtrace') which will show you the function tree, with the lowest level at the top. This shows the files each function call is in, and the line number it's called from.

The **bt** command should give you a rough idea of where the crash happened (usually the highest listed MUD source file). You should then use the **up** command to move to within that program location, as shown below in this typical output example:

```
Program received signal SIGSEGV, Segmentation fault.
0x443b8089 in free () from /lib/libc.so.6
(gdb) bt
#0  0x443b8089 in free () from /lib/libc.so.6
#1  0x080e88e9 in free_var_el (var=0x44473d24) at dg_handler.c:35
#2  0x080e890e in free_varlist (vd=0x8584740) at dg_handler.c:49
#3  0x080f2ee4 in script_driver (go_adress=0xb0e6b768, trig=0x89028b8,
type=0, mode=143665336) at dg_scripts.c:2763
#4  0x080eeac6 in trig_wait_event (event_obj=0x8584740) at
dg_scripts.c:740
#5  0x080e8561 in event_process () at dg_event.c:111
#6  0x0808cd8d in heartbeat (heart_pulse=240) at comm.c:1045
#7  0x0808ca4a in game_loop (mother_desc=5) at comm.c:1017
#8  0x0808c245 in init_game (port=1235) at comm.c:587
#9  0x0808ba8c in main (argc=135847552, argv=0x81507e5) at comm.c:410
#10 0x44356d06 in __libc_start_main () from /lib/libc.so.6
(gdb) up
#1  0x080e88e9 in free_var_el (var=0x44473d24) at dg_handler.c:35
35      dg_handler.c: No such file or directory.
        in dg_handler.c
(gdb) up
#2  0x080e890e in free_varlist (vd=0x8584740) at dg_handler.c:49
49      in dg_handler.c
(gdb)
```

Once you are in the function that you think is causing the problem, you can examine all the local variables by typing **info local**.

```
(gdb) info local
i = <value optimized out>
rznum = 0
qm = 0
tempfunc = (int (*)(struct char_data *, void *, int, char *)) 0xa
quest = (struct aq_data *) 0x81b2170
quest_flags = "NOBITS ", '\0' <repeats 16376 times>
(gdb)
```

The list command can be used to look at the source code at the current location (or in fact, any location in the source). To check the current location, you can simply type list to see the current line, and a few lines before and after it:

```
Program received signal SIGSEGV, Segmentation fault.
times_message (obj=0x0, name=0xbfb28f48 "boom", num=1) at shop.c:448

448           ptr = obj->short_description;
(gdb) list
444       if (obj)
445         len = strlcpy(buf, obj->short_description, sizeof(buf));
446       else {
447         if ((ptr = strchr(name, '.')) == NULL)
448           ptr = obj->short_description;
449         else
450           ptr++;
451         len = snprintf(buf, sizeof(buf), "%s %s", AN(ptr), ptr);
(gdb)
```

And you can more closely examine a specific variable using the **print** command:

```
(gdb) print ch->desc
$1 = (struct descriptor_data *) 0x8fdfbd0
(gdb) print ch->desc->original
$2 = (struct char_data *) 0x0
(gdb)
```

Using the up, down, list, info local and print commands, it should be possible to find the cause of your bug. If you are still unable to find the cause, you can copy/paste your gdb output, and post it onto the tbaMUD forum, where several coding experts would be happy to examine your debugging output and help you to find a solution.

When you have finished debugging, you can either kill the program, so that you can run it again, or you can quit from gdb. If you try to quit when the MUD is still active, then it will ask you if you want to kill the process before quitting.

A handy overview of the gdb commands can be found in Appendix B of this document.

Making a backup of your MUD

MUD backups are very important, and should be done regularly. How regularly depends entirely on how much work you would be willing to lose, and how much work you believe your players would be willing to lose.

There are three areas to consider when backing up files:

Description	**Who makes changes**	**Backup**
MUD Source Code	MUD Coders	Hourly-Monthly
Player Files	Players	Daily-Weekly
The MUD world	MUD Builders	Daily-Monthly

For example, if you are constantly coding on your MUD, you may wish to backup the source code every hour. If you very rarely make code changes, then you might backup the source code once per month, or before and following every code change.

Backing up player files should really be done at least every week, if not every day. If your MUD is a very busy one, with lots of players that play all day, every day, then you may even consider a more frequent backup.

You may wish to backup the whole MUD together, or just one of it's parts, but it is recommended that you keep a record of the date of the backup, usually in the filename.

To create a compressed backup, you use the **tar** and **gzip** commands. Although these look quite scary if you look at the help files for them, they are quite simple to use.

To create a backup of your src folder, for example, you would do the following:

```
bash $ tar -cvf src_backup_09-01-25.tar src/

bash $ gzip src_backup_09-01-25.tar

bash $
```

The above example uses **tar -cvf** to *create* the file **src_backup_09-01-25.tar** and fill it with the contents of the **src/** folder. The **-cvf** is used to *create* a new file, but if you wished to *add* to an existing tar file (commonly known as a tarball) you would use **-Avf** instead.

You can clearly see that the filename contains the date in YY-MM-DD format, so sorting in alphabetical order would also sort in date order. This means that when you are looking through old backups, you can quickly see which backups are the most recent.

The gzip line compresses the tarball, and changes the filename by adding .gz on the end to make **src_backup_09-01-25.tar.gz**. This is your backup file.

You can now transfer this backup file to your home PC using your normal FTP software, where you will be able to store it safely. When you have a few backups, you may wish to save a few to CD or DVD as a permanent backup.

Appendix A – Shell command Reference

ls – list directory contents
The ls command simply lists directory contents.
ls -l can be used to show permissions, filesize and date

cd – change directory
Used to enter a subdirectory or path. **cd ..** will go back to the parent directory and **cd ~** returns you to your 'home' directory.

mkdir – make a directory
Create a new, empty directory for putting your files into. Just **mkdir <name>**

rm – remove a file or directory
The *permanently* delete a file, simply **rm filename**
Note that the file cannot be recovered later!

mv – move a file or rename a file or directory
This moves a file or directory from one location to another. Unlike copying, the original location will not contain the file after moving.
To rename a file or directory, simply **mv oldname newname**

cp – copy a file or directory
Copying a file results in two identical files.
To copy a file or directory, simply **cp oldname newname**

ps – process show, displays running processes
Usually the command **ps uwx** is used to list processes as this provides the most useful information. The PID column shows the process ID, that is required by the **kill** command

kill – kill a running process
Simply type **kill <pid>** to stop a process running. The process id (pid) can be found using the **ps** command. Occasionally, a process cannot be stopped, in which case you can use **kill -9 <pid>** force an end to the process.

grep – search inside files
An amazingly useful tool for any programmer, grep allows you to search through text files, looking for a particular sequence of characters.
Use **grep text *.c** to look through all the C files for 'text'.
Use **grep -n 'two words' *.c** to look for two words with a space between, and **-n** makes it show line numbers in the results.

man – manual (help) pages for shell commands
This works for most shell commands. Just type **man <command>** for help.

more – show output one page at a time
Usually, the output from another command is 'piped' through more. This is done by adding **| more** to the end of the command. For example, you might use the cat command to look at the changelog, and wish to page the output, you would use the command **cat changelog | more**
It is also possible to send a text file to more for viewing. Using changelog as an example again, you would type **more < changelog**
When using more, you can press the space-bar to jump a page at a time, the enter key to jump one line at a time, or Ctrl-Z to exit from more

cat – show the contents of a file

The cat command simply lists the entire contents of a file.
It can also be used to concatenate two files, like **cat file1 file2 > file3**

tail – show last few lines of a file

Useful for checking logs, especially the syslog and crash log for your MUD. Just type **tail <file>** to show the last few lines of the file, or if you wanted to specify the number of lines, just use **tail -<lines> <file>** (for example, **tail -20 syslog** lists the last 20 lines of the syslog file)

pwd – print working directory

If you have used the **cd** command a lot, and got a little lost, you can type **pwd** to just show your current directory.

quota – display your disk/cpu/memory quota and remaining filespace

The **quota** command does not work on all system, but if you have a remote server, it should work. It simply shows you how much server resources you have been allocated, and how much you have left available to you.

du – disk usage

Provides a simply list of disk space used by the files and subdirectories within the current directory.

df – disk filesystem

Lists each of the available disks on your system, and how much free space there is available on each. This can be useful for determining if a drive has become full, so that something can be done about it.

Appendix B - GDB Command Reference

The following list of commands are used within the gdb debugging tool. This is not a complete list of every command that can be used in gdb, but just a few of the basic commands that should allow anyone to figure out where and why a crash is happening.

bt	Displays a backtrace. This shows the 'main' function at the bottom of the list, then every sub-function that has been entered into, shown above main. The function at the top of the list is therefore the 'current' location at crash-time.
info local	Displays the status of 'local' variables. Local variables are those created within the current function.
list	List the source code. This shows 3 or 4 lines above and below the current location. You can also list another location using **list <file>:<line>** for example, **list structs.h:20** might list structs.h from line 16 to 24.
print <var>	Displays the status of the specified variable. The variable should be local to the current function, or a global variable. It is possible to view variables within structures, for example: **print ch->name** is valid.
up	Moves up the function tree displayed by the bt command, making the parent function into the current function.
kill	Stop the current program from running.
quit	Exit from gdb (prompts to kill a running program).

Appendix C – MUD Configuration Options

The following describes each of the options within the **cedit** OLC menus. There are in the order found in **cedit**, and separated into sections named as the **cedit** main menu. In each case, the default option is shown, so you can return to default options if you wish.

Game Play Options

Player Killing Allowed : No
Are players allowed to kill other players? If this is set to yes, then player killing becomes enabled. Players killing other players will be flagged with the KILLER flag.

Player Thieving Allowed : No
Are players allowed to steal from other players, if they have the steal ability? If this is set to yes, then players will be able to steal from other players, but will have their THIEF flag set.

Minimum Level To Shout : 1
This provides an option to not allow new players to shout. Shouts can be heard throughout the zone that the shouter is in. If this is set to 1, then all players are able to shout.

Holler Move Cost : 20
Using the holler command has a movement cost. Holler is exactly the same as shout in that it can be seen by anyone in the same zone. There is no reason to use the holler command if the player can use shout instead, so you only need to check this value if you have set a minimum level for shout.

Tunnel Size : 2
This value dictates how many players are allowed into a TUNNEL flagged room.

Maximum Experience Gain : 100000
This is the maximum amount of experience that any player can gain at once. If something tries to give more than this amount, then the amount specified here will be given instead.

Maximum Experience Loss : 500000
This is the maximum amount of experience that can be lost (usually upon the death of a character) that any player can lose at once. If something tries to take more than this amount off the player, then the amount specified here will be taken instead.

Max Time for NPC Corpse : 5
How many game hours (ticks) that a mobile's corpse should stay in the MUD before it decomposes and vanishes.

Max Time for PC Corpse : 10
How many game hours (ticks) that a player's corpse should stay in the MUD before it decomposes and vanishes.

Tics before PC sent to void : 8
How many game hours (ticks) that should elapse before an idle character should be sent to the void. When a player is in the void, they are moved from their current location to a 'safe zone' where they cannot be attacked. When the player enters any command, they are instantly returned to their previous location. If

Tics before PC is autosaved : 48
Online players are automatically saved at regular intervals, so that in the event of a MUD crash, they don't lose a lot of hard-earned experience points. The default value of 48 ticks is equal to one hour real-time.

Level Immune To IDLE : 32
It is convenient for immortals not to be subjected to a disconnection due to idling. This is the minimum level at which this immunity is given.

Death Traps Junk Items : Yes
Death trap rooms should be rare, and well signposted, to prevent the average player from stumbling into them. If, however, several players become victims, their corpses and everything they carried and wore will be abandoned in the trap. If these are automatically junked, then there is no chance of a player regaining their lost equipment (for example, by a corpse retrieval script system). They will be gone forever.

Objects Load Into Inventory : Yes
When an immortal loads an object, should that object load in the Imm's inventory, or on the ground in the room. If this is set to 'No', then it would be possible for players to grab items before an Imm can pick it up. With it set to 'Yes', Imms are able to load non-carryable items, such as bulletin boards. This is an old throwback to the CircleMUD codebase, and may be removed in future releases.

Track Through Doors : Yes
When using the track command, should players be able to track mobiles or other players if the path passes through a closed or locked door.

Display Closed Doors : Yes
Should closed doors be shown on the autoexits, and when players type the exits command. If this is set to No, then players will only find the exit by looking in the correct direction, or by trying to walk in a direction that isn't shown.

Mortals Level To Immortal : Yes
Once a player has reached the highest possible level, should they be able to get some more experience, and level again to automatically become a low-level immortal? Although the default setting is Yes, many MUDs will change this to No.

OK Message Text : Okay.
Several commands simply return an OK message when they have completed successfully. Although the default message is fine, you may wish to personalise your MUD by changing it, or even part of changing your MUDs language.

NOPERSON Message Text : No one by that name here.
This is the message given to players when they attempt to do something (such as cast a spell, or perform an action) to another player or mobile, but the target cannot be found.

NOEFFECT Message Text : Nothing seems to happen.
This is the generic message that is used when an action (such as casting a spell) has no effect on the target.

Map/Automap Option : Imm-Only
The automap is shown to the right of each room description, and the map command shows a more detailed map of the player's surroundings. It is possible to allow anyone to use the map, or to restrict use of this feature to Immortals-Only, or to turn it off completely.

Default map size : 6
This is the size of the map shown when the player simply uses the map command. The map size must be in the range 1-12. This default value can be overridden by anyone, by entering a different number after the map command.

Default minimap size : 2
The minimap is shown to the right of room descriptions for any player that has the automap flag turned on. The default size of 2 shows the current room, and the rooms immediately surrounding the current room.

Scripts on PC's : No
Should the MUD allow mobile triggers to be attached to players. If you keep a tight control on what your MUDs builders are doing with their scripts, and you check each new script or script change personally, then it should be safe to allow scripts to be attached to players. This is still a relatively new addition to tbaMUD, so unforeseen problems may arise if this is turned on.

Exit To The Main Menu
Simply returns you to the main cedit menu.

Crashsave/Rent Options

Free Rent : Yes
Should players be charged gold when they leave the MUD for each item that they have. If free rent is disabled, then when a player logs back in, some of their gold will be taken as a rent payment. If the player cannot afford the rent payment, they will lose their entire inventory and equipment.

Max Objects Saved : 30
This limits the number of items that a player can keep when they quit. The players' equipment is saved first, so this value should always be greater than the number of eq slots. If this is set too low, then players will complain about lost items. If it is set too high, then players might rent with too many items, which may cause lag when they log in or out.

Minimum Rent Cost : 100
This is the minimum amount, in gold coins, that will be charged as rent when a player logs back in (if 'Free Rent' is disabled, see above).

Auto Save : Yes
Determines whether the auto-save feature should be enabled or disabled. If it is turned on, then the progress of every player in the MUD will be saved at regular intervals. This may cause some lag on a MUD with hundreds of players logged in.

Auto Save Time : 5 minute(s)
How often should each player be saved if the autosave feature is enabled? After a player has been logged in for the specified time, they are added to the 'save queue', and are saved as soon as possible. This also determines the frequency at which player's houses are saved.

Crash File Timeout : 10 day(s)
This is the lifetime of crashfiles and forced-rent (idlesave) files in days. After this time, the MUD may remove a crash file to preserve space on the server.

Rent File Timeout : 30 day(s)
This is the lifetime of normal rent files in days. After this time, the MUD may remove a rent file to preserve space on the server.

Exit To The Main Menu
Simply returns you to the main cedit menu.

Room Numbers

Mortal Start Room : 3001
This is the VNUM of the room that new players should start in. Usually, when a player quits, their start room will become the room they logged out at, but following a crash, it will generally revert to this default room again.

Immortal Start Room : 1204
This is the VNUM of the room where Immortals should start when they log in. This is usually set to a central room in the 'God Zone' that provides quick and easy access to most of the Immortal facilities.

Frozen Start Room : 1202
The 'ice box' is a prison room, where players can be placed from which they can't escape or communicate with other players. It allows immortals to speak to and discipline errant players who break the MUD rules. A player is often 'frozen' with the **freeze** command, which disables most commands, then transferred here. The player should be unfrozen with the **thaw** command once punishment has been given.

Donation Room #1 : 3063
Donation Room #2 : 5510
Donation Room #3 : 235
Your MUD can have up to three donation rooms. If the VNUMs of valid rooms are entered here, then when a player 'donates' an item, it will select one of the donation rooms (at random) and place the item in that room. Donation rooms are usually located in large cities that are accessible to inexperienced players.

Exit To The Main Menu
Simply returns you to the main cedit menu.

Operation Options

Default Port : 4000
The default port number is used if the MUD is run manually from the shell without using the autorun script, and without specifying a port on the command line. The autorun script will over-ride this value.

Default IP : <None>
IP address to which the MUD should bind. This is only useful if you're running Circle on a host that host more than one IP interface, and you only want to bind to *one* of them instead of all of them. Setting this to NULL (the default) causes Circle to bind to all interfaces on the host. Otherwise, specify a numeric IP address in dotted quad format, and Circle will only bind to that IP address. (Of course, that IP address must be one of your host's interfaces, or it won't work.)

Default Directory : lib
The MUD generates a large number of files, and stores all of these in the 'lib' folder by default. If you wish to rename the MUDs data folder, or move the files to another location, then you can set the new location here.

Logfile Name : <None>
The logfile is handled by the autorun script, and your MUD just outputs text which is sent to the syslog file by the script. The MUD program therefore has no way of knowing the name of the log file. If you change the log filename in the autorun script, then you should also set the log filename here to prevent complications.

Max Players : 300
This is the maximum number of players that can be logged into the MUD at once. It is quite rare that this limit will be reached, but if you are getting a lot of players, and finding that it causes lag, you may wish to consider reducing this value. If this limit is reached, then anyone trying to log into your MUD will be turned away.

Max Filesize : 50000
Maximum size of bug, typo and idea files in bytes (to prevent bombing).

Max Bad Pws : 3
Maximum number of password attempts before disconnection. After disconnection, the player can reconnect and try again to login.

Site Ok Everyone : YES
By flagging every new user as SITE-OK, it allows them to log when a URL has been banned. This means that if a player is banned, then new players that happen to use the same ISP will also be able to log in. If a banned player repeatedly creates new characters, you might consider turning this option to No.

Name Server Is Slow : NO
With modern internet speeds, it is highly unlikely that you will need to disable the DNS check that is done when a player logs in. This converts the player's IP address to a readbale URL, to give a possible indication of the player's location. If this is set to 'Yes', then only four-number IP addresses will be shown for each player.

Use new socials file: YES
If you wish to use the aedit socials (actions) editor, then this must be set to yes.

OLC autosave to disk: YES
Should OasisOLC automatically save all changes to disk. If this is turned off, then builders can see which zones are unsaved with the **olc** command, and all pending changes can be saved by typing **olc save**. If this is set to Yes, then when a builder saves the changes 'internally' on exit from any OLC menu, the changes are also saved to the data files on the server.

Main Menu:

```
               Welcome to tbaMUD!
               0) Exit from tbaMUD.
               1) Enter the game.
               2) Enter description.
               3) Read the background story.
               4) Change password.
               5) Delete this character.

                 Make your choice:
```

This allows you to rewrite the login menu without making any code changes. If you do re-arrange the options, or add new options, then you will need to make corresponding changes in the source code.

Welcome Message   :

```
               Welcome to tbaMUD!  May your visit here be... Enlightening
```

This is the message seen by players when they select option 1 from the above menu to log into the MUD.

Start Message :

Welcome. This is your new tbaMUD character! You can now earn gold, gain experience, find weapons and equipment, and much more -- while meeting people from around the world!

This message is seen by new players when they log in for the first time.

Exit To The Main Menu

Simply returns you to the main cedit menu.

Autowiz Options

Use the autowiz : Yes

The autowiz automatically generates a new wizlist and immlist whenever a player is promoted, or an Imm is demoted. Both of these text files are in the **lib/text** folder, but if you choose to manually edit the files, you should turn off the autowiz, so that your changes aren't lost. Note that autowiz only works under UNIX.

Minimum wizlist level : 32

The wizlist shows higher-level immortals, and here, you can set the level at which immortals are moved from the immlist to the wizlist. Any immortal below this level will be shown on the immlist.

Exit To The Main Menu

Simply returns you to the main cedit menu.

Appendix D – Glossary of Terms

cedit	Configuration editor. This allows MUD owners to configure the MUD.
debug	The process of finding and fixing bugs in the source code.
gdb	GDB is a UNIX debugging tool. It allows programmers to track down bugs in their source code.
olc	OnLine Creation. A method of editing and creating aspects of the MUD, such as zones, rooms, mobiles, quests, etc... In tbaMUD, OasisOLC is used, which has an easy-to-use menu interface
tick	Also known as tics. A tick is one game 'hour', which can be seen by using the **time** command. By default in tbaMUD, one tick lasts 75 seconds.