The CircleMUD Administrator's Manual

Jeremy Elson
<jelson@circlemud.org>

December 12, 2001

Abstract

This document describes how to configure CircleMUD and how to compile it for the first time. It also discusses how to run the server including documentation of command-line options, a description of system logs and how to use them, and a description of daily and long-term maintenance required by the MUD. The intended audience is implementors new to CircleMUD or MUD administration in general.

More information about CircleMUD, including up-to-date versions of this documentation in ASCII and Postscript, can be found at the CircleMUD Home Page `<http://www.circlemud.org/>` or FTP site `<ftp://ftp.circlemud.org/pub/CircleMUD/>`.

Contents

1 Welcome to CircleMUD!

1.1 Background and Introduction

CircleMUD is a derivative of DikuMud, the Multi-User Dungeon developed at DIKU, the Department of Computer Science at the University of Copenhagen. The original CircleMUD, version 1.0, was run by Jeremy Elson at the Johns Hopkins University's Department of Computer Science with moderate popularity from January until September of 1992. The version 1.0 code, which was never released to the public, was archived and remained inactive for several months after the death of the original CircleMUD. In the summer of 1993, it was taken out of storage and cleaned up with the intention of making it a public code base that anyone could freely download and use to start a MUD. Version 2.0, the first publically available version, was released in June of 1993. Circle has been maintained as a public code base ever since then, although we do not actually run a MUD of our own. The original CircleMUD has not run since it went down in 1992, nor will it ever run again.

Hundreds of carefully spent hours have gone into the development of Circle, but it is not a finished product – it is only a beginning. It has only a small number of spells and skills, a small world with relatively few areas, and only the 4 original DikuMud classes. From a gamer's point of view, it pales in comparison to other MUDs which have dozens of spells spread over a dozen classes, a rich palette of skills, and thousands upon thousands rooms, mobiles, and objects.

Yet from a programmer's point of view, Circle is very highly developed. While the look and feel of the original DikuMud has been maintained, most of the underlying code and many of the structures have been optimized, reworked, or completely redesigned to be efficient, compact, and easily changeable and extensible. Dozens of features which have become standard in the MUDding world over the past few years, although they were not part of the original DikuMud release, were added to Circle as well.

The result is that CircleMUD is a launching pad for your own MUD ideas. Our hope in releasing it is that it will free potential MUD implementors from having to worry about dealing with bug-ridden code or wasting time reinventing the wheel by re-coding standard MUD features, allowing them to devote more effort towards creatively shaping Circle into their own unique vision of how a MUD should look and feel. The reason why Circle has so few specialized spells, skills, classes, and races is to encourage implementors just like you to create your own unique system instead of just another stock MUD that everyone has seen before.

So, it is with this message that we inflict our code on the world: don't just put another generic MUD on the Net – give us something new and exciting, and have as much fun as possible in the process!

1.2 Are you a Player or an Administrator?

If you're playing a game like Zork or Doom, you're both the administrator and the player of the game: you're the one who has to download the software and get it running, and you're also the one who gets to play. MUDding isn't that way at all, because there's usually a very strong division between people who play MUDs and people who administer MUDs. If you've never played a MUD before, jumping right in and trying to run one of your own will probably just get you hopelessly confused. Instead of trying to compile and run the CircleMUD package yourself (which is presumably what you're trying to do if you're reading this document), you should play someone else's copy of CircleMUD. There is a partial list of sites `<http://www.circlemud.org/sites.html>` using CircleMUD on the CircleMUD web server.

1.3 So, you're sure you want to run your own MUD?

If you're already an old hand at playing MUDs and you've decided you want to start one of your own, here is our advice: take a vailum, lie down, and hide in a dark closet until the desire goes away. Just playing MUDs is masochistic enough, isn't it? Or are you trying to shave that extra point off your GPA, jump down that one last notch on your next job evaluation, or get rid of that pesky Significant Other for good? If you think silly distractions like having friends and seeing daylight are preventing you from realizing your full potential in the MUD world, being a MUD Administrator is the job for you.

Don't get me wrong: running a production MUD can be great fun. It can also be overburdened by politics and plagued by spiteful players devoted to making your life difficult, and otherwise be a highly frustrating endeavour. That's why I don't do it any more.

1.4 Giving Credit Where Credit is Due

If I haven't scared you away yet, and you're still sure you want to use CircleMUD, please stop for a moment and look at the CircleMUD license in the file `license.doc`. It outlines the terms under which you must use CircleMUD.

The license is simple. It boils down to the message, "Don't rip off other people's work." Unfortunately, this simple message ended up becoming somewhat long-winded because I am trying to prevent people from abusing DikuMud in the future as they have in the past.

Also, out of courtesy if nothing else, please keep the `credits` file intact. You can add your own credits on top of the existing file, but I'd appreciate it if you would not simply remove it and all references to the word "Circle" everywhere in the MUD. How would *you* feel if someone took *your* code and then took credit for it?

USE OF THIS SOFTWARE IN ANY CAPACITY IMPLIES THAT YOU HAVE READ, UNDERSTOOD, AND AGREED TO ABIDE BY THE TERMS AND CONDITIONS SET DOWN BY THE CIRCLEMUD LICENSE.

2 Getting Started

2.1 Section Requirements

CircleMUD 3.0 was originally written as UNIX software and will automatically configure itself (using GNU autoconf) and compile under most versions of UNIX, both BSD and System V derivatives. With minor adjustments (documented below), the same source should compile under Microsoft Windows 95 and NT, IBM OS/2, and the Amiga. Users have also reported getting Circle to compile and run under MkLinux, the port of Linux to the Power Macintosh. CircleMUD will *not* work under DOS, Windows 3.x, Windows for Workgroups 3.x, or Mac System 7.

Specifically, the following variants of UNIX have been tested and are known to work with Circle:

- SunOS 4.1.4
- Solaris 2.3 and above
- Irix 5.2, 6.2
- AIX 3.2
- Ultrix 4.x
- HP-UX 9.x
- Linux 1.x, 2.x
- BSD/OS 2.1
- Mac OS X (10.0 and above)

If your system is not listed, don't despair; because of the `autoconf` program, Circle will compile under most versions of UNIX on its own. A large effort was made to make Circle v3.0 more portable by converting many of its system calls over to POSIX compliance. Converting Circle to POSIX vastly helps portability on modern operating systems, most of which are POSIX-compliant. Unfortunately, on some older systems that are not (such as NextSTEP 2.x), it may be more difficult to compile 3.0 than it was to compile earlier versions of the code. (POSIX stands for Portable Operating System Interface for UNIX

and is an effort to standardize the way UNIX programs talk to the operating system. For more information, see Stevens' excellent book, *Advanced Programming in the UNIX Environment.*)

For a small, private MUD, or a MUD used only for testing and development, about 10 MB of disk space and 16 MB of memory should be sufficient. For large, public MUDs with a large player base, 30 to 50MB of disk space and at least 32 MB of memory are recommended. Free memory is much more important than CPU speed; CircleMUD uses virtually no CPU time.

Historically, CircleMUD has always been developed under different variants of UNIX. The original CircleMUD was run on a DECstation 3100 running Ultrix 4.0, which remained Circle's development platform until v2.0. Starting with v2.0, Circle was developed under various versions of Linux and Solaris over the years ranging from Linux 0.99.11 through the current Linux 2.2.x and including Solaris x86 5.6.

2.2 Downloading and Unpacking the Source

The first step in setting up CircleMUD is to make sure you have the most recent version of the source; if you downloaded CircleMUD from an unofficial mirror site, it could be out of date. You can always find the most recent version of the source at CircleMUD's official FTP site `<ftp://ftp.circlemud.org/pub/CircleMUD/>`

The archive is offered several formats: one which ends in `.tar.gz`, one which ends in `.zip` and one which ends in `.lha`. The files have the same contents but have been compressed using different programs. You should ensure that you get a copy that you can unpack on your operating system.

Next, unpack the archive. If you have the `.tar.gz` version, unpack it using `gzip` (GNU unzip) and the `tar` archiver. If you don't already have them, both of these utilities can be downloaded from the GNU FTP site `<ftp://ftp.gnu.org/pub/gnu/>`. To unpack the archive on a UNIX system, type:

```
gzip -dc circle30xxx.tar.gz | tar xvf -
```

If you downloaded the `.zip` or `.lha` version, make sure to use an UNZIP program capable of handling long filenames and which preserves the original directory structure of the archive (PKUNZIP 2.04 does *not* do either of these things by default.) The best unzip program is probably Info-Zip's unzip program `<http://www.info-zip.org/>`; it is compatible with all UNIX variants, Windows, OS/2, the Amiga, and pretty much every other computer on the planet. If you have Windows 95/98 or NT, another good choice is WinZip `<http://www.winzip.com/>`.

For the next few sections, please note that the Cygnus Tools (Cygwin) are available for the Windows platform, which allow users to use Unix tools on that operating system.

2.3 Configuring CircleMUD

Note: This section applies *only* to UNIX users. If you're using Windows, OS/2, or the Amiga, read `README.WIN`, `README.OS2` or `README.AMIGA` instead.

Circle must be configured using the `configure` program which attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create Makefiles and a header file called `conf.h`.

From Circle's root directory, simply type

```
./configure
```

If you're using `csh`, it may not execute "configure" properly, giving you an error message like "`Permission denied`". If this occurs, try "`sh ./configure`" instead.

`configure` can take several minutes to run if you're using a slow computer. It will print out various status messages while it works, telling you what characteristics of your computer it has found. It should create two Makefiles (`src/Makefile` and `src/util/Makefile`) and one header file (`src/conf.h`).

You should only need to run `configure` once – right after you unpack CircleMUD from its archive. You will probably compile the source dozens of times as you add new features, but `configure` only needs to be run before the first time you compile. Please note that there is one exception to this rule: if you move your MUD to a different computer, you must run `configure` again before you can recompile the source code. To rerun configure after moving the source, make sure to delete the file called `config.cache` and then run `configure` again.

2.4 Compiling CircleMUD

Note: This section applies *only* to UNIX or Cygwin users. If you're using Windows, OS/2, or the Amiga, read `README.WIN`, `README.OS2`, or `README.AMIGA` instead.

The `src` directory contains the source code for the main MUD server; `src/util` has the source for a dozen or so MUD maintenance utilities. There are two Makefiles, one in each source directory, but all compiling is normally performed from the `src` directory only.

To compile the only CircleMUD server itself, type "`make`". Use "`make utils`" to compile the utilities, or "`make all`" to compile both the server and the utilities. It is also possible to compile individual utilities from the `src/util` directory: from `src/util`, type "`make [utility-name]`". All compiled binaries go to the `bin` directory.

The stock (unmodified) CircleMUD code should compile with no warnings or errors.

Despite my best efforts there's a chance that you'll have problems compiling Circle if you're using some version of UNIX that I've never seen before. It's impossible to give general advice on how to port software, except to say that you should ask a friend who knows a lot about both UNIX and C to help you. Also, if you have problems, you should definitely look at the CircleMUD FAQ (Frequently Asked Questions list with Answers), which you'll find in Circle's home directory.

If you do port Circle to some other platform successfully, please share your hard work with others by sending a description of what you had to do to get Circle to compile on your system to `<cdev@circlemud.org>`. Be sure to include the specific name of your operating system and hardware. Full details on porting to a new system can be found in the Porting CircleMUD document.

The Makefile directs all compiled programs to the `/bin` directory. Although not recommended, you may want to put Circle's `/bin` directory in your $PATH. The reason that this is not recommended is that most of the resources are referenced using relative paths, and thus require that you run the programs from the base CircleMUD directory.

3 Running CircleMUD

Note: This section applies *only* to UNIX or Cygwin users. If you're using Windows, OS/2, or the Amiga, read `README.WIN`, `README.OS2`, or `README.AMIGA` instead.

3.1 Execution and `autorun`

1. type 'autorun &'
2. Wait a few moments for the server to boot.
3. type: `telnet localhost 4000`

Circle should always be run from circle's "root" directory, not the `/bin` directory. You can run it manually by typing `bin/circle` (useful for testing and debugging). For running

the game "for real," it is better to use the `autorun` shell script provided in Circle's root directory.

Autorun lets Circle run itself for long periods of time. It continuously runs the game as well as removing old system logs, moving newer system logs to the `log` directory, and saving certain log entries to permanent files.

Autorun can be controlled by creating files with certain names. You can use the '`touch`' command to create a file, and, of course, the '`rm`' command to remove a file. If a file called '`.fastboot`' exists, the CircleMUD will reboot immediately if it crashes or is shut down instead of waiting 40 seconds as it normally does. A file called '`.killscript`' will cause the script to terminate itself; i.e. if you want to bring the game down. If you want to temporarily prevent the MUD from rebooting, create a file called "`pause`"; the script will go into a wait loop until '`pause`' is removed.

Although you can create these files manually, the**shutdown** command from within the MUD has several options which will create these files for you. See the **shutdown** help entry in `wizhelp.doc` for more information.

3.2 Command-Line Options

Circle recognizes a number of command-line options. You can use them by actually specifying them on the command-line when running Circle manually or, by adding them to the `FLAGS` variable in your autorun script to use the options automatically every time you run the MUD with autorun.

The syntax is:

```
circle [-m] [-q] [-r] [-s] [-d <path>] [port]
```

-m Mini-Mud Mode. Mini-mud will be one of your most powerful debugging tools; it causes Circle to boot with an abridged world, cutting the boot time down to several seconds. It is useful for testing features which are not world-related (i.e, new commands or spells).
CircleMUD uses split world files (in the lib/world hierarchy); each directory (i.e. wld, obj, mob, shp, and zon) has a file called "`index`" which specifies which files should be loaded at boot-time. The file "`index.mini`" specifies which parts of the world should be loaded with the -m option.

-q Quick Boot. Prevents checking of timed out object files. Every time Circle boots, it checks every object file to see if it has timed out; if so, it is deleted. This is done primarily to save disk space. If time is more important to you than space, use the -q option. -q is automatically activated when you use -m.

-r Restricted Game. Allows you to decide at run-time whether or not the game will allow new characters to be created. Using -r is equivalent to typing "`wizlock 1`" (see `wizhelp.doc` for more information).

-s Special Routines Disabled. Allows you to suppress the assigning and calling of all mobile, object, and world special procedures. Historically used as a debugging tool in conjunction with the -d option (see below), but now obsolete because Circle checks to make sure entities exist before attempting to assign a special procedure to them.

-d Data Directory. Useful as a debugging and development tool, if you want to keep one or more sets of game data in addition to the standard set, and choose which set is to be used at run-time. For example, you may wish to make a copy of the entire world in a separate directory, so that you can test additions to the world files without subjecting players to unnecessary hazards. The default data directory is '`lib`'. Any core dumps (may they never happen to you!) will take place in the selected data directory.

port Port Selection. Allows you to choose on which port the game is to listen for connections. The default port is 4000, which is historically the port used by most DikuMuds. You can change the default in `config.c` and the `PORT=` line of the autorun script. (The `PORT=` line of `config.c` simply passes the value as a command-line parameter to Circle.) The port specified on the command line or by the autorun script will take precedence over the port specified in `config.c`.

3.3 Creating an Implementor Character

The first character to log in to the MUD will be made the maximum (Implementor) level. If you log in and aren't made an implementor, then the playerfile probably is not empty – take the MUD down, delete the playerfile (`lib/etc/players`), and start the MUD again. You should fix your stats using the RESTORE command when you first create the character (i.e. type "`RESTORE <your-name>`".

Once you are logged in as an implementor, type WIZHELP for a list of privileged commands. Documentation of those commands is contained in the file `wizhelp.doc`. `wizhelp.doc` is in the standard help file format, so you can simply add it to the help index file if you want online help available for wizard commands. (`wizhelp.doc` is not contained in the online help by default as a security precaution.)

To create additional wizard characters, log them in normally as level 1. Then, advance them using your implementor character, using the `ADVANCE` command. See `wizhelp.doc` for more information.

4 Customizing CircleMUD

Once you get over the initial euphoria of having your own MUD compile and run successfully for the first time, you should be aware of some of the files which control how Circle looks and feels so that you can configure it for your personal taste.

4.1 `config.c`

The most useful file for configuration is the config.c source file. It has dozens of options useful for customizing your MUD. Before you open your MUD to players, you should carefully read through config.c from beginning to end, checking what options are available as well as making sure they're all set the way you want them. Remember, you have to recompile and rerun the MUD before any changes you make to `config.c` take effect, of course.

4.2 Text Files

The place where most of your day-to-day changes will be is in the `lib/text` directory, which contains all of the text files read by players. The most visible file is "`motd`", (Message Of The Day), which is the message that mortals seen (though not necessarily read!) when they log in. Similarly, the "`imotd`" (Immortal MOTD) is seen by immortals when they log in. Other interesting files are "`news`" (for the `NEWS` command), "`help`" (for the `HELP` command with no arguments), "`help_table`" (for `HELP` with an argument), and others.

One file is particularly important: "`policy`". Despite what anyone tells you, and your best efforts to the contrary, your MUD will have political problems. If anyone tells you that you can run a MUD without politics, they're wrong. If you tell your players that you're going to run a revolutionary MUD that doesn't have any politics, you're lying. Every MUD has political problems sooner or later. Unfortunately, this usually occurs "sooner". There are various ways to make political problems easier to handle, but the best way is to decide on some ground rules from the very beginning, and then set those decisions down in writing. That's what the "`policy`" file is for. You probably know what kind of political problems MUDs have (you are not trying to run a MUD without ever having played one, are you!?) – stuff like playerkilling, thieving, kill stealing, client use, multi-char playing, reimbursements, etc. Decide on your policies early and make sure your characters know what those policies are.

It is also important to write down a set of rules dictating what immortals are allowed to do and not allowed to do. That's what the "`handbook`" (Immortal Handbook) is for. Immortals will always try to bend you as far as they can, so it's important for you to decide on answers to questions before those questions come up. Can immortals assist

players? What about giving a single 'cure light' for a player about to die? Can immortals kill mobiles for their own enjoyment? Can they kill players for the same reason? Are they allowed to make policy? Break policy? Moderate disputes? Turn over each others' decisions?

4.3 World Files

The root of the area hierarchy is `lib/world/`. `lib/world/` has 5 subdirectories: wld, mob, obj, shp, and zon, which is where the world, mobile, object, shop, and zone files go, respectively. Each directory has a set of world files in it with the appropriate extension (i.e. the obj subdir will have a bunch of files ending with "`.obj`", such as 30.obj, 31.obj, etc.) plus two special files called "`index`" and "`index.mini`". "`index`" is a list of which world files are to be loaded by Circle. This makes the job of adding new areas easier – just add the new area files to your area directory, and then add the name of the new file to "`index`". "`index.mini`" controls which (smaller) set of world files should be loaded in the debugging mode (Mini-Mud Mode, explained below.)

5 System Logs

CircleMUD writes a wide variety of information to standard output and standard error. If you're using the autorun script, the boot messages are put into a file called "`syslog`". During Circle's boot sequence, the system log keeps a record of everything the MUD is doing to initialize itself; this can be useful to determine what the problem is if the MUD dies while it is booting. Once the game is up and running, the syslog contains player information, recording when players connect, disconnect, rent, unrent, quit, die, hit death traps, etc. The game also records status information about itself, falling generally into two categories: usage information and errors.

5.1 Player Information

The player information recorded by Circle's system logs will serve you very well as your players start to make wild claims about strange bugs resulting in them losing equipment or points. Many mudders prey on the insecurities of a new mud administrator who is terrified that his or her MUD is riddled with bugs and will do anything to satisfy grumpy players – don't let yourself fall into that trap! CircleMUD is bound to contain bugs, but most of the core systems have been well tested, so you should take claims such as "I magically lost all my stuff!" with a grain of salt and check your system logs.

If a player ever asks you for reimbursement of equipment, money, gold, experience

points (XP), or whatever, your gut reaction should always be to check the logs first.

> As a sidebar, let me point out that the value of system logs is twofold: 1) they actually provide you with valuable information, and 2) they make your players paranoid. When I first started mudding and I heard about this mysterious "system log", it made me incredibly paranoid. Now that I've done a good deal of MUD administration, I've seen the same paranoia in many other players.
>
> That paranoia is a very good thing. The system logs become an abstract and shapeless but omnipresent force on the MUD. Players hear about "the System Log" and then get paranoid that everything they do is being recorded, so they tend to behave, lest the evil System Log betray their wrongdoings to the Gods.
>
> For this reason, when you go to check your logs, it's a good idea to say something like "Hold on... let me go check the system logs, OK?" because it reinforces the syslog's presence in the collective psyche of your players.

Back to the point. When someone claims that they've been wronged by the evil system, always check the logs. The logs give you power to say things like "What do you mean your items disappeared in rent? It says right here in the logs 'Rasmussen has quit the game.' You did not rent at all, you just *quit*!"

To diffuse disputes such as, "The game dumped my stuff, but I had enough money!!" or "I logged in and my stuff was gone, there must be a bug!!", two types of log entries are written. First, every time a character rents, the log records the character's per diem rent rate as well as the total amount of money on hand and in the bank. Second, the log records makes a record of all characters' equipment dumped due to insufficient funds.

Remember, rent is specified as a daily rate but is amortized on a per-second basis! In other words, if you rent at the rate of 100 coins per day and come back 36 hours later, you'll be charged 150 coins.

The `autorun` script saves 6 levels of raw system logs. In addition, it greps the logs for certain pieces of extra-juicy information to save indefinitely.

The system logs are your friends. Love them.

5.2 Usage Information

Every 5 minutes, the game counts how many people are playing and records that information in the system log. Optionally, if you define RUSAGE in `comm.c`, it will also record system resource information such as CPU time and memory used. The usage information currently logged by Circle is, as you can see, somewhat sparse; local MUD admins are encouraged to add to this code as is appropriate for their particular site.

Usage information isn't critical, but it is interesting to look at the usage patterns to determine when your peak playing hours are. If you're good at using 'cut' and other Unix utilities, you can even dazzle your friends by graphing your MUD's system usage.

Note: friends not included with the CircleMUD distribution.

5.3 Errors

Just as your first gut instinct should be to look at the logs if a player starts begging you for something, your first gut instinct in the event of a crash or unexpected shutdown should also be to look at the system logs.

A Unix utility called '`tail`' is used to look at the last few lines of a text file; it is very useful for looking at the last entries in the system log to see the last thing that happened before the shutdown. Often, Circle will report an error in the logs just before it crashes. This method is particularly useful if the MUD crashes during its boot sequence, because the logging during boot is intensive.

If Circle shuts down unexpectedly and there is no core dump in the `/lib` directory, the game probably detected an internal error and killed itself. Such shutdowns are always preceded by entries in the system log describing the error.

If there is no error message at the end of the log, then there probably IS a core dump, so you can use '`dbx`', '`gdb`', etc. to examine the core dump and determine the reason for the crash. The file `hacker.doc`, generously provided by Furey of MERC Industries, offers useful insight into the art and science of debugging – you'd be well advised to give it a look-see.

Circle sometimes encounters a serious but non-fatal error; in this case, the error will be written to the system log with the prefix SYSERR, but the MUD will not shut itself down. You should always be aware of any SYSERRs which occur – they are often useful for foreseeing imminent danger or averting problems before they become critical. If a SYSERR does occur, try to determine if a change you've made recently has caused it. Ignoring SYSERRs is like ignoring compiler warnings: you can be tempted to ignore them because the game keeps going even if they exist, but you can easily get yourself into trouble by not listening. The autorun script saves all SYSERRs to the file `log/errors`.

6 MUD Maintenance

6.1 Technical Maintenance

Once you get the MUD up and running, you will surely want to modify it – adding new areas, new features, new code, and new ideas, however these topics are beyond the scope of this document. See `coding.doc`, `areas.doc`, and other creation documents for more information about how to customize your MUD once you get it up and running. This section simply focuses on some of the simple maintenance that'll be necessary to keep your MUD running smoothly. Make sure not to get so caught up in being a God that you forgot you're also supposed to be acting as a software engineer!

First, you should look at the log/errors file regularly to make sure there are no recurring problems, particularly problems that may have been caused by code you've added. If the MUD crashes, it will usually generate what is called a "core dump" – a big file called "`core`" in your lib directory, created by the operating system to record the state the game was in the moment before it crashed. You should look at the core dump using a debugger such as "`gdb`" or "`dbx`" to determine why the MUD crashed. See the file "`hacker.doc`" for more information about debugging.

You probably will want to clean out the playerfile on a regular basis to remove deadweight characters (i.e. people who log in and look around for 10 minutes, and then never come back). You can decide how often to purge the playerfile – every day if disk space is tight, or every month if it isn't. The `purgeplay` utility program included in the `src/util` directory removes deadweight players. Make sure to run the "`purgeobjs`" script (in the `lib/plrobjs` directory) after you purge the playerfile. `purgeobjs` removes the object files of players who no longer exist in the playerfile.

The '`automaint`' script in the main circle directory will automatically purge the playerfile and player objects for you. *DO NOT RUN THIS SCRIPT WHILE THE MUD IS RUNNING!* Doing so will make your life (more) difficult.

6.2 Diplomatic Maintenance

Okay, so now you have your wonderful CircleMUD up and running smoothly and all is right with the world. Right?

Wrong.

Well, technically, right. Circle requires very little day-to-day attention in order to keep the program itself running smoothly. But the MUD itself is just a series of instructions running on a computer, processing data. Never lose sight of the fact that there will be

dozens, hundreds, or maybe even thousands of people connecting to your MUD – and they are not programs. They are people!

From the technical side, there are relatively few things you have to do to keep the game running. But you cannot just dump a MUD on the Internet and then ignore it! Spend time on your MUD. Try to keep up with the boards, and make an effort to respond to the complaints, ideas, and suggestions posted there. Take a look at the '`bug`', '`typo`', and '`idea`' files from time to time – and maybe even respond to some of the ideas using Mudmail. Try to respond to Mudmail you receive from players in a timely manner. Make sure that your '`news`', '`policy`' and other text files are up-to-date and suit the political climate on your MUD.

If you cannot or just do not want to deal with the player politics, make sure that you choose someone who can and will, and make them responsible for dealing with it. If no one does it, your MUD will stagnate and die.

7 Final Thoughts

Try to remember that running a MUD should be fun. It can sometimes be easy to lose sight of the ultimate goal of personal enjoyment that MUDs are supposed to provide, particularly when MUDs start to get crushed under the weight of their own politics or the egos of the administrators. If you find that your MUD is more of a source of frustration than enjoyment, don't be afraid to close it.

Good luck with your MUD! Always feel free to write to us and let us know what you think of the MUD, and the interesting things you've done with it. We cannot promise a timely response, but we love hearing from people who are using our code.

For further information and updates, you can be certain to check out the CircleMUD homepages `<http://www.circlemud.org/>`, the associated pages of Ceramic Mouse `<http://developer.circlemud.org/>`, and the CircleMUD Mailing List `<http://www.circlemud.org/maillist/>`.