Part I

Fundamentals

Chapter 1 – The Basics

Getting Started

Learning is active. No amount of reading about MOOs and MOOing can convey what it's really like as well as jumping in and doing it! The goal of this section is to present the basics of connecting to a MOO and starting to get your bearings.

First, you must connect to the MOO. The most primitive way to do so is using telnet. The syntax, from a command line host computer, is

```
telnet <MOO name> <port number>
```

For example:

```
telnet yib.moo.mud.org 7777
```

or

```
telnet lambda.moo.mud.org 8888
```

Some telnet applications may have you type the MOO name and port number into a dialog box, then select "Connect" or "OK".

This is sometimes referred to as *raw telnet*, because there is no additional software between you and the MOO. In some cases, the backspace key doesn't work while using raw telnet. Sometimes, even the text you type doesn't display! (Some telnet programs have menus to adjust these things.) Also with telnet, if text is sent from the MOO to your screen while you are typing (and it often is), the text will appear right in the middle of the line you are typing, and things get very confusing, very quickly. For this reason, most people try to obtain a kind of program known as a MOO (or MUD) *client*. There are a variety of client programs available, and your choice of a client will depend partly on what kind of machine or operating system you are using to connect to the MOO. Client programs are not part of the MOO per se, but greatly enhance one's MOOing experience. I strongly recommend them.

The most important thing that a client does is separate, in some way, the text that you are in the middle of typing from text that the MOO is sending to your screen. Some clients have a small window at the bottom of a main window; others keep shifting your line down and inserting received text above it. The rest is frosting: Some show the text that you type in bold face, for example. Most let you use a scroll bar or other command to review previously-displayed text; some provide fancy editing capability, and so forth, but the main thing is to be able to see straight (so to speak). Some well-known client programs include Pueblo, tkMOO, MUDDweller, and emacs with mud.el. To find a client that's compatible with your computer, I recommend using an Internet search engine such as `www.google.com` and searching for `MOO clients`.

The first time you visit a MOO, you will connect as a guest. Guests have limited privileges (they can't receive MOOmail, for example), but are equipped to do all the basics of communicating and getting around. It's usually a good idea to visit a given

MOO a few times as a guest before requesting a character, to get a feel for the place and the people and then decide if you actually *want* to get a character there. Characters can select their own names, and usually have at least limited building privileges.

Please read the welcome screen.

After doing so, type:

```
connect guest
```
[1]

If you do use a client, you will probably have to specify the MOO's address in a set-up window. The specifics of how to do this are different for each client and beyond the scope of this book.

A good place to find out about many of the MOOs that are available is Rachel's Super MOO List, at:

```
http://cinemaspace.berkeley.edu/~rachel/moolist/index.html.
```

On most MOOs there will be some information that you are invited and requested to read when you first connect. First, you may be instructed to type `help` at any time for assistance. You may then be informed that there is new news, and instructed to type `news` or `news new` to read it.[2] After that (on LambdaMOO), you

[1] Since you are logging in anonymously, you may be asked a few additional questions. Take the time to read and answer them.

Most MOOs permit more than one guest to be logged on simultaneously. Your actual guest name might be something like "Green_Guest", or "Blue_Guest".

[2] If you *don't* use a client program, there are certain commands that you will want to be aware of from the very start.

First, you will need to specify the size (height in lines and width in characters) of the screen or window you are using. To specify the height of your screen, type

```
@pagelen <number>
```

Many screens have 24 lines.

To specify the width of your screen, type

```
@linelen <number>
```

Many screens have a line length of 80 characters; if yours does but things still look funny, specify one fewer characters than are actually displayed on a line.

Finally, type

```
@wrap on
```

This will prevent words from being broken in the middle across line breaks.

If you do these things, then when your screen gets full, you will be prompted to type `@more` to cause more text to be displayed.

If, after getting a character (see the section on requesting a character and getting settled in, page 14), you *switch* from using raw telnet to using a client, you will probably need to type the following:

```
@pagelen 0
@wrap off
```

Doing this will prevent the MOO from prompting you to type `@more` every time your screen gets full (because now the client lets you scroll back). Most clients separate lines at word boundaries, so typing are invited to type `@tutorial` for an introduction to basic MOOing, and requested to type `help manners` and read the text presented if you have not already done so. This is an awful lot of stuff to read. I recommend you proceed in the following sequence:

Skim `help manners` or `help rules` or any indicated text about the way you are expected to behave. There will likely be several things in it that may not make sense at first, but MOOers will often expect you to have read it anyway. On the first pass, your goal should be to get what you can from it, note the various *kinds* of things that are in it, for future reference, and to realize that the community *does* have formal expectations of its members and guests.

If a tutorial is indicated, do that next.

Then, type `help` just to see the basics of the online help system. If a topic grabs your interest, go ahead and read about it by typing:

```
help <topic>
```

If there is a newspaper, read it, by typing `news`. If an obvious exit is indicated, try it. Welcome!

When you are ready to disconnect from the MOO, type:

```
@quit
```

Basic Communications

This section explains various ways of interacting with other people on a MOO.

Say

The most fundamental communication command on a MOO is the `say` command, and this is what to use if you want to say something to someone who is in the same room as you. If Yib types:

```
say Hello.
```

Then Yib will see on her screen:

```
You say, "Hello."
```

And everyone else in the room with Yib will see on eir screen:

```
Yib says, "Hello."
```

`@wrap off` relieves the MOO itself of that task and may speed up the MOO’s response time slightly, which is desirable.

The `say` command is so basic, and used so much, that it has an abbreviation, which is the double-quote character ("). If Yib wants to say that she likes to tap dance, she can type:

```
"I like to tap dance.
```

and she will see on her screen:

```
You say, "I like to tap dance."
```

and everyone else in room when Yib says this will see on eir screen:

```
Yib says, "I like to tap dance."
```

Notice that when using this form of the command, you do not type a close-quote character at the end. The system appends one automatically.

Emote

Sometimes you might want to express something non-verbal, such as a feeling or a gesture. It might seem awkward to say, “I feel happy and am smiling.” For this situation, use the emote command. This command will prepend your name to whatever you type after it. For example, if Yib types:

```
emote feels happy.
```

Everyone in the room (including Yib) will see on eir screen the line:

```
Yib feels happy.
```

Notice that Yib typed her sentence in the third person. If she had typed:

```
emote feel happy.
```

Then everyone would have seen the line:

```
Yib feel happy.
```

The emote command, too, is used so much that it has a single-character abbreviation, which is the colon (`:`). So Yib could type:

```
:feels like dancing.
```

and everyone in the room (including Yib) would see on eir screen:

```
Yib feels like dancing.
```

There is a difference between emoting things and actually doing things, i.e. actually interacting with objects on the MOO. Suppose Yib is holding an object called “linen handkerchief”. If Yib emotes the line:

```
:drops linen handkerchief.
```

Everyone in the room would see the text line:

```
Yib drops linen handkerchief.
```

But Yib would *not* in fact have dropped it. If one were to look at the room Yib was in at the time, one would not see the handkerchief there. If Yib had instead typed:

```
drop linen handkerchief
```

then the object, `linen handkerchief`, would actually move from Yib to the room that Yib was in. This is a fundamental concept (though perhaps a subtle one, especially at first): whether one "merely generates text", or whether one causes a change in the database, e.g. changes the location of an object. Yib could emote dropping an elephant, or lifting one, and the text would print out, regardless of whether or not there actually was an elephant in the vicinity.

A special form of `emote` is the double colon (`::`) It is just like a regular `emote` except that a space doesn't appear after your name. A typical usage would be:

```
::'s new hat is of truly astonishing dimensions!

Yib's new hat is of truly astonishing dimensions!
```

Many MOOs have a collection of short cut commands such as `wave <person>` or `hug <person>`, which can be used for frequently-depicted actions such as waving, hugging and so forth. People who get spoiled with these short cuts often neglect emote, but it remains one of the single most flexible and expressive commands on the MOO.

Directed Say

Typically, when a room has a large number of people in it, the conversation tends to break up into several conversations going on at once. It's impractical to listen to everyone talking at the same time, and people don't. On a MOO, every utterance and gesture appears on a line by itself, so, in theory, one could keep up with everything, even in a crowded, noisy room. In practice, however, MOO conversations also tend to break up in crowded rooms, and it's typical to follow what one or a few people are saying and tune out the rest. To facilitate this, there is a command that is generally referred to as *directed say*.[3] To direct a remark to a particular person, begin your line of text with a dash (`-`), then, without typing a space, type the name of the person you wish to address, then a space, then your remark. For example, Barth might direct a remark to Yib by typing:

```
-Yib My, what a lovely hat you have on!
```

Everyone in the room (including Barth and Yib) would see the line:

```
Barth [to Yib]: My, what a lovely hat you have on!
```

[3] Different MOOs may or may not provide this automatically to brand new players. LambdaMOO does, using a kind of object called a *feature object* (FO). (Feature objects are explained in the section beginning on page 41.)

Whisper

You might want to communicate something to one person only, without others in the room being aware of it. For this there is the *whisper* command. The syntax of this command is slightly different than the other communications commands, in that you have to put quotes around the text you wish to transmit. Here's an example. Nim types:

```
whisper "Meet me in the hot tub in two minutes." to Yib
```

Yib sees:

```
Nim whispers, "Meet me in the hot tub in two minutes."
```

Nim sees:

```
You whisper, "Meet me in the hot tub in two minutes." to
Yib.
```

No one else in the room sees anything of this exchange.

Page

It's often the case that one wishes to communicate with someone who is logged on but not in the same room. For this we have the `page` command. The syntax is `page <person> <message>`. If Boo were in a room called `A Quiet Place` and wanted to greet Yib from afar, she might type:

```
page Yib Hi!  Been up to any mischief lately?
```

Yib would see:

```
You sense that Boo is looking for you in A Quiet Place.
She pages, "Hi!  Been up to any mischief lately?"
```

Boo would see:

```
Your message has been sent.
```

Because this is virtual reality, and because the players themselves can change the way it works, even simple commands like `page` can give fun and interesting results. For example, Boo can adjust the text that appears in the first line that a person sees when she pages em. For example, she might set it to say, "Boo tosses you a poison dart with a message attached." Yib can adjust the text that someone paging her sees when the message is received. For example, she might set it to say, "Yib adjusts her glasses and reads your message." (How to do this is explained in the segment on setting messages, beginning on page 46.)

Remote-Emote

Sometimes, instead of paging, one wants to gesture from afar, and the name given to this is `remote-emote`. The syntax for this command is `+<player> <text>`. If Yib were on top of the observatory dome and wanted to wave to Tartan_Guest from there, she would type:

```
+Tartan_Guest waves.
```

Tartan_Guest would see:

```
 (from On Top of the Observatory Dome) Yib waves.
```

Yib would see:

```
Tartan_Guest has received your emote.
```

Like emote, this command is very flexible, but doesn't actually *do* anything, i.e. it doesn't change the database. If I were to remote-emote to Klaatu:

```
+klaatu hooks you with her fishing pole and reels you in!
```

Klaatu would see the text, but would not in fact be reeled in anywhere, and would remain wherever he was. This distinction is important, because one *could* make a fishing pole object, and endow it with verbs that moved other objects (e.g. fish, or other players).

Channels

Channels on a MOO are kind of like channels on a CB radio. The idea is, you tune to a given channel, and you and everyone else tuned to that channel can communicate with one another even though you may all be in different locations. One difference between a channel and a real CB radio is that with a real CB radio, everyone present in the room can hear it. On a MOO, only those connected to a channel can listen to it, even though others may be present in the room. There are public channels that anyone can join, and private channels that have restricted access. Channels are quite popular and many MOOs have them, though they are not included as part of LambdaCore. Details on how to use channels on LambdaMOO are given in the section on LambdaMOO Feature objects (see page 217).

Recording Communication

All of these commands (say, emote, directed-say, whisper, page, remote-emote and communicating on channels) are examples of communication that occurs in real time, i.e., the information is received at the time of transmission rather than stored for retrieval at a later time. In general, communication of this sort is of a transient nature, although it is stored in the computer's Random Access Memory (RAM) and is protected by copyright[4] . Players can issue a command (`@paranoid`) to keep the most recent several lines available for closer inspection. Furthermore, many players' client programs save arbitrary numbers of lines which can be viewed in the client window by scrolling back. It is also possible for players to save logs (transcripts) of MOO sessions to a file on disk. In theory, only the guests and players in a particular room are supposed to be privy to conversation that occurs in that room at that time, but such is rarely the case in practice. It is technically possible for conversations to be bugged. Occasional overt "bugging" is tolerated, (for example a player might participate in a conversation remotely via a surrogate called a "puppet"), but this practice is generally discouraged. Covert bugging, also called "spying", is generally discouraged, and is rare. The command `@sweep` is provided for those who wish to check to see whether a listening device is present.

Requesting a Character and Getting Settled In

The Initial Request

Different MOOs have different registration policies. LambdaMOO requires that each player provide a valid email address and its administrators take pains to ensure that there is only one character per typist, or else that multiple characters are duly cross-referenced to one another. Other MOOs may have less restrictive policies.

The usual syntax for requesting a character on any MOO is:

```
@request <name> for <email-address>
```

For example:

```
@request Cinderella for cindy@fireplace.com
```

Read the questions carefully, and answer straightforwardly. Depending on which MOO you're on, the system may check your connection site and compare it against the email address you give. If they don't match, it may ask you for a brief explanation. If you already have a character and are requesting a second (this is permitted on some MOOs, not on others), it may ask you to verify this fact, and/or explain how it might be that there is already a character with the email address you've specified. Some MOOs disallow certain email addresses, specifically those from providers that are known for giving free, anonymous email accounts. If you have multiple email addresses, you may be asked to provide them.

The questions that are asked of you in the `@request` process come in three forms: YES/NO questions, single line answers, or multiple line answers. If you change your mind about answering any question at all, type:

```
@abort
```

(Note that if you are in the middle of a multi-line answer, `@abort` must be on a line by itself.) For the multiline answers, you may type in as many lines as you need to. A line, for the purpose of this discussion, is an arbitrary amount of text

[4] David Jacobson, "Doing Research in Cyberspace," Field Methods, 1999, 11:2:;127-145.

terminated by the `<enter>` key. To end your answer, type a period (`.`) on a line all by itself (and then press the `<enter>` key again). Sometimes further explanation is needed. In such cases, you will be given a regular email address to use for providing it.

Soon you should receive email at the address you provided, containing a password and verification of the player name you requested, or notification that the name you requested was already in use.[5] In either case, the character you requested will also have an alias of the form `New-Player-<number>` e.g. `New-Player-58337`. The password is case-sensitive, i.e. you must type it in exactly as given. You are not stuck with the name `New-Player-<number>`. After you connect initially (and at any time thereafter, as often as you like), you have the option of changing your name to anything you like. Suppose the name "Cinderella" was already in use, and our intrepid typist received email saying that e had been given a character with the name "New-Player-58337" and password "OgkM2".

The first order of business is to log on. Our typist would connect to the MOO, see the welcome screen, then type:

```
connect New-Player-58337 OgkM2
```

There are a variety of things to do from this point, and the sequence isn't essential. Two obvious things are to decide on a different name, and to change the password to something that's easier to remember.

Changing Your Password, Changing Your Name, Adding Aliases

Passwords have to be more than four characters long, and are not permitted to be common English words. One scheme is to select two English words and run them together, for example, "RubySlippers". As always, passwords are case-sensitive. To change your password, type `@password <old-password> <new-password>`. So our typist would enter:

```
@password OgkM2 RubySlippers
```

Now let's consider a name change, and perhaps some aliases. First, you'll want to cast about and see if the name you're considering is already in use, since names have to be unique. To do this, you can use the `@who` command. Suppose we want to see if the names "Drusilla" and "Prunella" are taken.

You type:

[5] LambdaMOO is under certain population-growth restrictions, and so it is possible that your request will be added to a queue, there. At the end of the `@request` process, you will be told what your place in the queue is. Most times, the queue moves along fairly briskly. You can check your status as often as you like: To do so, log on as a guest, then type:

```
@go registrar
```

Then, `check status on <email-address>` using the email address you gave when you requested your character. The person requesting the character Cinderella, for example, would type

```
check status on cindy@fireplace.com
```

```
  @who Drusilla
```

And you see on your screen:

```
Disconnected
Player             Last Disconnect              Location
-----------------  ------------------------     -----------
Werebull (#58806)  Sun Sep 19 02:30:42 1999 EDT The Pasture
```

This indicates that the character Werebull already has “Drusilla” as an alias, and therefore that name is not available to you. Let’s try Prunella:

You type:

```
  @who Prunella
```

And you see on your screen:

```
  "Prunella" is not the name of any player.
```

Success! To change your name, use the `@rename` command, as follows:

```
  @rename me to Prunella
```

Names of players are not case-sensitive. Your name will appear as you type it in when you use the `@rename` command, but someone typing `@who pRuneLLa` would still find *you*.

You are not limited to one name. You might, for example, want to use “Prunie” as a nickname. As before, the first thing to do is to see if that name (alias) is already in use. You type:

```
  @who Prunie
```

And if you’re lucky, you’ll see:

```
  "Prunie" is not the name of any player.
```

To add a name (as opposed to changing your name), use the `@addalias` command:

```
  @addalias Prunie to me
```

If you later decide you don’t like that nickname so much after all, you can remove it as follows:

```
  @rmalias Prunie from me
```

Describing Yourself

The description you give yourself is what people will see when they look at you. Use the `@describe` command as in the following example:

```
  @describe me as "One of Cinderella's two wicked step
  sisters.  Her beauty, such as it is, is as cold as her
  scheming heart."
```

Type in the whole description as one long line (even though it shows as multiple lines in the example).

Notice that the description is enclosed in double-quote marks. This is optional, except that it happens to be the case that if you omit them, there will only be one space between sentences, even if you type two spaces between each sentence. If you use the double quotes, there will be as many spaces between sentences as you type.

If you find that you've made a typographical error in describing yourself, you have two options. One (easiest if your description isn't too long) is to `@describe` yourself all over again, as above. You can revise and re-enter your description as many times as you like. The other alternative is to learn to use the note editor (see the section that begins on page 66) and *edit* your description

As you meet other players, you will notice that some of them have multi-line descriptions. You can't get a multi-line description with `@describe`. This is another incentive to learn to use the editor, as it is the best way to create a description that has more than one line (paragraph).

Sooner or later you will undoubtedly encounter someone who reacts or responds to the fact that you looked at em. Certain player classes provide the ability to be notified when someone looks at you; this is generally referred to as *look detection*, and it can be disconcerting the first few times you encounter it. In general, it is considered poor form to give someone grief for looking; some people assert that it is rude even to mention that one knows someone looked. In contrast, some players not only notice when someone else looks at em, but in addition broadcast a message to the room, such as, "Oliver notices Prunella's glance and smiles at her," or worse, "Prunella looks at Oliver and smiles." The second form is worse because it should be up to Prunella to decide whether she smiles at the sight of Oliver or not. Perhaps it is Prunella's nature to sneer, instead, and she should have the right to specify that. MOOers are not united in their opinions on this issue, although most would probably concede that broadcasting a message to the room every time someone looks at you becomes boring fairly quickly.

Setting Your Gender

MOOs offer a variety of gender options. You can use the `@gender` command in two ways. If you type it on a line by itself:

```
@gender
```

The system will tell you your current gender setting, the pronouns it associates with that gender, and will show you a list of available genders. To set your gender, use the `@gender` command with an *argument* (see the glossary for an explanation of what an argument is), e.g,

```
@gender female
```

In this case, the system would display:

```
Gender set to female.
Your pronouns:
  she,her,her,hers,herself,She,Her,Her,Hers,Herself
```

Since there is no way to tell, online, whether someone is telling the truth about eir gender, we sometimes speak of a player as “presenting as male” or “presenting as female”.

Setting a Home

When you disconnect from the MOO (using the command `@quit`), your player object is returned to its `home` on the MOO. Players who have not otherwise specified a home are returned to a default location. (On LambdaMOO it’s the Linen Closet.) There is nothing wrong with keeping the default player start as your home. Many players do this.

Another option is to find a room that will permit you to set your home there (to do so, go to the room and type `@sethome`). There is no penalty for trying to set your home to a location where that is not permitted. You will simply be given a message instructing you to ask the owner to make you a resident of that room. If you are the owner of a room and someone else wishes to set eir home there, and you wish to let em do so, the command is:

```
@resident <player or object>
```

To see a list of your room’s residents, type:

```
@residents
```

To remove someone from the list of residents type:

```
@resident !<player or object>
```

A third option is to create a home for yourself. Most players do this eventually. To do it, you would use the `@dig` command. For an in-depth discussion of `@dig` and related commands, see the section on building, starting on page 79.

Suppose Prunella wanted to create a home for herself named “The Crystal Palace”. She could type:

```
@dig The Crystal Palace
```

The system would then create a new room object, would set Prunella to be the owner of this room object, would set the name of this room object to “The Crystal Palace”, and would print a message to Prunella saying that the room had been created and informing her of its object number. A detailed discussion of objects begins on page 37.

```
The Crystal Palace (<object-number>) created.
```

To get to her new room, Prunella will have to teleport there, using the object number that the system just printed out for her. If the object number were #29370, for example, she would type:

```
@go #29370
```

After arriving, Prunella could type:

```
@sethome
```

and then that location would be her new home. She would be in that room when she connected, and would be returned to that room when she logged off. She could also go there at any time by typing the `home` command:

```
home
```

The room would start off with no description. Prunella could give it one with the `@describe` command. First she would `@go` there, then she could type:

```
@describe here as "You are in a magnificent palace, as warm
and inviting as its name suggests."
```

If Prunella ever forgot the object number of her room, she could type:

```
@audit
```

And that would show her a list of objects she owns, including herself and the room she had just created.

Unexpected Greetings

Sooner or later, someone will greet you (typically with a page or a remote-emote) within moments of your logging on. How do they do this? There is a *feature object* (see page 41) that lets one designate interesting players, and which will notify one when an interesting player connects or disconnects. These are called *login watchers.* Most MOOs have some version of a login watcher.[6]

Another thing that may seem disconcerting is that people may know that you are new without your having told them. It's more than just being an unfamiliar face. The `@age` command tells how old a person is in terms of the MOO.[7]

Eventually, you will become familiar with these commands, and take such spontaneous greetings in stride, or even start making them yourself!

[6] On LambdaMOO, it's feature object #24222.

[7] On LambdaMOO, other commands to find out about players' ages are on Carrot's Social Interaction Feature (#36714).