A brief guide to TinyMUD

Jennifer Stone (aka Chrysalis)
jennifer@uokmax.ecn.uoknor.edu

Rusty C. Wright
rusty@garnet.berkeley.edu

January 31, 1990

Contents

Much of this is from the TinyMUD source code and the `small.db` example.

1 Ordinary commands

`drop` <*object*>
`throw` <*object*>

Drops the specified object. <*object*> can be either a thing or exit.

`examine` <*name*>
`examine #`<*number*>

Prints a detailed description of object specified by <*name*> or by <*number*> giving name, description, owner, keys, pennies, failure message, success message, others failure message, others success message, and exits. The location will also be displayed if you control the object's location (that is, if it's not being carried by someone else or in a room you don't control).

`get` <*object*>
`take` <*object*>

Gets the specified object. <*object*> can be either a thing or exit.

`give` <*player*> `=` <*amount*>

Gives <*player*> the specified number of pennies.

`go` <*direction*>
`go home`
`move` <*direction*>

`move home`

Moves in the specified direction. `go home` is a special command that returns you to your home (initially Limbo). If the direction is fully specified, the `go` may be omitted.

`gripe` <*message*>

Sends <*message*> to the system maintainer.

`help`

Prints a short help message.

`inventory`

Lists what you are carrying.

`kill` <*player*> [`=` <*cost*>]

Kills the specified player. Killing costs either <*cost*> pennies or 10 pennies, whichever is greater. The probability of success is proportional to the cost.

`look` <*object*>
`read` <*object*>

<*object*> can be a room, thing, player, or direction. Prints a description of <*object*>.

`page` <*player*>

Used to inform an active player that you are looking for them. The targeted player will get a message telling them your name and location.

`rob` <*player*>

Attempt to steal a penny from <*player*>.

`say` <*message*>
`"`<*message*>
`:`<*message*>

The first two forms display the <*message*> with the notification that you said it. For example, if your player's name is Betty the other players in the same room will see

`Betty says` "<*message*>"

The third form *poses* the message, preceded by your name, with no quotes, as in

`Betty` <*message*>

For both the second and third forms, do not put a space after the double quotes or colon as it will be included in the message.

`score`

Prints how many pennies you have.

`whisper` <*player*> `=` <*message*>

<*player*> is presented with <*message*> saying that you whispered it. The other players only see the message

`Betty whispers something to` <*player*>.

2 Commands for modifying the dungeon

`@create` <*name*> [`=` <*cost*>]

Creates a thing with the specified name. Creation costs either <*cost*> pennies or 10 pennies, whichever is greater. The value of a thing is proportional to its cost.

`@describe` <*object*> `=` <*description*>

<*object*> can be a room, thing, player, or direction. Sets the description a player sees when they use the command `look` <*object*>. If <*object*> is `here` it sets the description for the current room that is displayed when the room is entered. If <*object*> is `me` it sets the description for your character.

`@dig` <*name*>

Creates a new room with the specified name, and prints the room's number.

`@fail` <*object*> [`=` <*message*>]

Without a message argument, clears the failure message on <*object*>, otherwise sets it. The failure message is printed when a player unsuccessfully attempts to use the object.

`@find` <*name*>

Prints the name and object number of every room, thing, or player that you control whose name matches <*name*>. Because the `@find` command is computationally expensive, there is a small charge for using it.

`@link` <*direction*> `=` <*room number*>
`@link` <*thing*> `=` <*room number*>
`@link` <*room*> `=` <*room number*>

In the first form links the exit specified by <*direction*> to the room specified by <*room number*>. The exit must be unlinked, and you must own the target room if its `link_ok` attribute is not set. If you don't already own the exit its ownership is transferred to you. The second form sets the home for <*thing*>. If <*thing*> is `me` it sets your home. The third form sets the dropto; see section 5.5 for an explanation of dropto's.

`@lock` <*object*> `=` <*key*>

Sets a key (another object) for an object. If <*key*> starts with `*` then it must be a player's name. <*key*> can contain the Boolean operators `!` (not or negation), `&` (and), and `|` (or), and use parentheses for grouping.

In order to use <*object*> you must satisfy the requirements of <*key*>. In the simplest case you must simply have <*key*>. If <*key*> is preceded by `!` then you must not have <*key*> in order to use <*object*>. See section 6.3 for more complicated examples.

`@name` <*object*> `=` <*name*>
`@name` <*player*> `=` <*name*> <*password*>

Changes the name of the specified object. This can also be used to specify a new direction list for an exit (see for example `@open`). For a player, it requires the player's password.

`@ofail` <*object*> [`=` <*message*>]

Without a message argument, clears the others failure message on <*object*>, otherwise sets it. The others failure message, prefixed by the player's name, is shown to others when the player fails to use <*object*>.

`@open` <*direction*> [`;` <*other-dir*>]* [`=` <*destination*>]

Creates an unlinked exit in the specified direction(s). You can also specify an exit to link the exit to. Once created, you (or any other player) may use the `@link` command to specify the room to which the exit leads. See also `@name`.

`@osuccess` <*object*> [`=` <*message*>]

Without a message argument, clears the others success message on <*object*>, otherwise sets it. The others success message, prefixed by the player's name, is shown to others when the player successfully uses <*object*>.

`@password` <*old*> `=` <*new*>

Sets a new password; you must specify your old password to verify your identity.

`@set` <*object*> `=` <*flag*>
`@set` <*object*> `= !`<*flag*>

Sets (first form) or resets (second form) <*flag*> on <*object*>. The current flags are `dark`, `link_ok`, `sticky`, `temple`, and `wizard`.

`@success` <*object*> `=` <*message*>

Without a message argument, clears the success message on <*object*>, otherwise sets it. The success message is printed when a player successfully uses <*object*>. Without <*message*> it clears the success message.

`@teleport` *<object>* `=` *<destination>*

Teleports the object to the specified destination. You must either control the object or it's current location, and you must be able to link to the destination. You can only teleport objects from room to room, not into or out of someone's hand.

`@unlink` *<direction>*

Removes the link on the exit in the specified *<direction>*. You must own the exit. The exit may then be relinked by any player using the `@link` command and ownership of the exit transfers to that player.

`@unlock` *<object>*

Removes the lock on an object.

3 Miscellaneous commands

`@stats` Shows current total of players, rooms, objects, exits.

4 Money

Some actions have an associated cost:

Action	Cost
`kill`	10 pennies or more
`page`	1 penny
`@open`	1 penny
`@dig`	10 pennies
`@create`	10 pennies (or more)
`@link`	1 penny, plus 1 penny to the previous owner if you didn't already own the exit.

You get money by finding pennies, getting killed, having someone give you money, or by sacrificing a thing. The sacrifice value of a thing is (cost of the thing/5) $-$ 1.

5 TinyMUD concepts

An *object* is either a player, room, thing, or exit.

In addition to the commands listed above there are some built in words in TinyMUD; `me` and `here`. `me` refers to your character or player, and `here` refers to the room you are in. For example, you can use the `@describe` command to give yourself a description; as another example, in order to prevent yourself from being robbed use `@lock me = me`.

5.1 Success and the lack thereof

When you can `take` a thing, use an exit, or rob a player you are successfull in using that object. The converse is true for failure. The `@success`, `@osuccess`, `@fail`, and `@ofail` commands set the success and failure messages on objects.

5.2 Object strings

Every object has six strings:

1. Name. This is what you use with `drop`, `examine`, `get`, and so on. You can also use the object's number (for example, when two objects have the same name).

2. Description. This is what is given when you use the `look` command.

3. Success message. This is what you see when you successfully use the object.

4. Others success message. This is what the other players see when you successfully use the object.

5. Failure message. This is what you see when you fail to use an object.

6. Others failure message. This is what the other players see when you fail to use an object.

The maximum length of each string is 512 characters.

5.3 Object properties

As listed in the @set command, objects can have any of the following properties:

dark When a room has its dark flag set you can't see things in it with the look command. When a thing or player has its dark flag set it can't be seen. Only a wizard can set the dark flag on a thing or player. Setting the dark flag on exits currently has no effect.

link_ok You can link to a room if you control it, or if the room has its link_ok flag set. Being able to link to a room means that you can set the homes of things (or yourself) to that room, and you can set the destination of exits to that room. See also the @link command for additional information on linking and section 5.5 for droptos. Setting the link_ok flag on players, things, and exits currently has no effect.

sticky When an object that has its sticky flag set is dropped it immediately goes home. When a room has its sticky flag set its dropto is delayed until the last person leaves the room. Setting the sticky flag on players and exits currently has no effect.

temple When a room has its temple flag set you can sacrifice things there and receive pennies for your sacrifices. (See section 4 for how many pennies you receive for your sacrifices.) Only a wizard can set this flag. Setting this flag on players, things, and exits currently has no effect.

wizard When a player has its wizard flag set they are a wizard. Only a wizard can set this flag. Setting this flag on things, rooms, and exits currently has no effect.

The flags player, room, and exit are set automatically when a player, room, or exit is created. They cannot be subsequently unset or set with the @set command.

5.4 Control

There are three rules for determining control:

1. You can control anything you own.

2. A wizard can control anything.

3. Anybody can control an unlinked exit (even if it is locked).

Builders should watch out for item 3.

5.5 Dropto's

When the `@link` command is used on a room, it sets a dropto location for that room. Any thing dropped in the room (if it is not `sticky`; see above) will go to that location. If the room has its `sticky` flag set the effect of the dropto will be delayed until the last player leaves the room. The special location `home` may be used as a dropto, as in `@link here = home`; in that case things dropped in the room will go to their homes. To remove the dropto on a room go into that room and use `@unlink here`.

5.6 Homes

Every thing or player has a home. For things, this is the location the thing returns to when sacrificed, when a player carrying it goes home, or when (if its `sticky` flag is set) it is dropped. For players, this is where the player goes when they issue the `home` command. Homes may be set using the `@link` command; for example, `@link donut =` <*room-number*> or `@link me =` <*room-number*>. Exits may also be linked to the special location `home`; for example, `@link north = home`.

5.7 Recycling

Nothing can be destroyed in TinyMUD, but it is possible to recycle just about anything. The `@name` command can be used to rename objects, making it easy to turn a silk purse into a sow's ear or vice versa. Extra exits can be unlinked and picked up by their owner using the `get` command, and dropped like things using the `drop` command in any room controlled by the dropper.

5.8 Being killed

When you are killed you return to your home and any items you were carrying return to their homes. As a consolation you receive 50 pennies from the TinyMUD Total Life Indemnity insurance company.

6 Examples

Here we present examples to demonstrate some of the features of TinyMUD.

6.1 Success and failure messages

Success and failure messages are fairly straightforward. Just remember that for the messages set with `@osuccess` and `@ofail` the player's name is prefixed onto the message when it is printed, while the messages set with `@success` and `@fail` are printed as-is.

Previously we saw that you can use `say`, `"`, and `:` to display messages. An older method for posing non sequiturs is to set the others success message on yourself and then rob yourself. For example, if your character is Betty and you do

```
@osuccess me = starts picking her nose.
rob me
```

on your screen you'll see

```
You stole a penny.
Betty stole one of your pennies!
```

while the other players will see

```
Betty starts picking her nose.
```

An easier way to accomplish this is to simply do

```
:starts picking her nose.
```

then both you and the others see

```
Betty starts picking her nose.
```

When using `"` and `:` don't follow them with a space because it will be included in the output; put the message right up against the quotes or colon.

6.2 Making your home

The minimal steps for making your home are

1. Make the room for your home with the `@dig` command. Write down the room number in case the following step takes a long time.

2. Make or acquire an exit. In order to use the `@open` command you must own the room that you are doing the `@open` in. The alternative is to find a room with an exit that isn't linked and use it.

3. Make a link to your home. Once you've made or found an unlinked exit simply use the `@link` command to link the exit to your room.

4. Find a room to which you can make a link in order to have an exit from your room (this is a room with the `link_ok` flag set). For the sake of example we'll pretend the number of this room is 711; we'll be using it in step 6. Without this you'd be able to go to your home but you wouldn't have any way to get out of it.

5. Set the link from you to your home. Go into your room and do

   ```
   @link me = here
   ```

6. Make the exit and link it to the destination

   ```
   @open out
   @link out = #711
   ```

   (The `#` isn't mandatory.)

Of course there are probably various details that you would want to take care of in addition to the above steps. For example, if you're antisocial and want to prevent other people from using your home room you'd do

```
@lock down = me
```

assuming that `down` is the exit you made in step 2. Along a similar vein you might not want other people linking to your room in which case you'd turn off the `link_ok` flag on your room. You might also set the description of your home room. If you own the exit you could also set the success, others success, fail, and others fail messages on the exit to your home. Without the descriptions places and things are boring and uninteresting.

6.3 Lock key boolean examples

When using the `@lock` command the key is either another object or some boolean combination of other objects. If the key starts with a `*` then that object must be a player.

For example, if a room has a direction `out` and you want to prevent players from carrying the object `xyz` when they go out, you would use

```
@lock out = !xyz
```

or if you want to prevent the player Julia from using the `out` exit you would use

```
@lock out = !*Julia
```

If you want to prevent only Julia from going out with `xyz` you would use

```
@lock out = ( *Julia & !xyz )
```

6.4 Ersatz commands

You can make new commands by making an exit and then locking it to something impossible to have and then assigning the failure and others failure messages to it. For example, assume the following commands have been used

```
@open eat
@link eat = here
@lock eat = something_impossible
@fail eat = You try to eat but only gag
@ofail eat = tries to eat but instead gags
```

Then when you use the command `eat` the others in the room will see `Betty tries to eat but instead gags` and you'll see `You try to eat but only gag`.

Note that this "new command" will only work in the room that you made it in.

Appendix

A Wizard commands

`@chown` <*object*> `=` <*player*>

Changes the ownership of <*object*> to <*player*>.

`@dump`

Forces a dump of the database. This command isn't really necessary since `@shutdown` does one as well as the regular periodic dumps.

`examine` <*name*>
`examine #`<*number*>

Print a detailed description of object specified by <*name*> or by <*number*>. All six strings listed in section 5.2 are printed.

When <*name*> is a room or `here` it lists the owner, key, number of pennies, the description, contents, and exits. When <*name*> is a direction lists the direction number, owner, key, pennies, and destination.

`@force` <*victim*> `=` <*command*>

`@newpassword` <*player*> `=` <*password*>

Changes the password for <*player*>.

`@shutdown`

Shuts down the TinyMUD server.

`@stats`

Lists the total number of objects, which is the sum of the rooms, exits, things, and players, giving the count for each one.

`@teleport` [<*victim*> `=`] <*destination*>

Teleports the object to the specified destination. Object can also be `me`.

`@toad` <*player*>

Turns <*player*> into an object.

`@wall` <*message*>

Send <*message*> to all players.

Index