

The `act()` Function

Alex Fletcher

Based on text from the DikuMud authors

December 14, 2001

Abstract

This document is intended to give an overview of the `act()` function and give a summary of the various flags and the associated descriptive control characters. The `act()` function is one of the primary functions for sending a message from the game to a character or number of characters in a room.

Contents

1	The <code>act()</code> Function	2
1.1	Overview	2
1.2	The Function	2
1.3	The Parameters	2
1.4	Control Characters	3
1.5	Examples	4
A	<code>act()</code> Reference Sheet	7

1 The `act()` Function

1.1 Overview

The `act()` function is used to process and send strings of text to characters in a room. It can be used to send the same basic string to a number of characters filling in certain segments – designated by control characters – in different ways, dependant on what each character can see and who each character is. Once the text string passed to the function has been parsed, it is capitalized and a newline is added to its tail.

1.2 The Function

The `act()` function is found in the `comm.c` source file and is described as:

```
void act(const char *str, int hide_invisible, struct char_data *ch,
         struct obj_data *obj, const void *vict_obj, int type)
```

These pieces are used as follows:

- str:** This is the basic string, a null terminated character array, including control characters (see section 1.4 on the following page ‘Control Characters’), to be sent to characters designated by the targets.
- hide_invisible:** A TRUE or FALSE value indicating whether or not to hide the entire output from any characters that cannot see the “performing character”.
- ch:** The “performing character”. This is the character that the output string is associated with. The character is used to determine the room for the output of the action in question.
- obj:** An object (an actual item – `obj_data`) used in the course of the action.
- vict_obj:** This can be either a character involved in the action, another object, or even a predefined string of text.
- type:** One of `TO_VICT`, `TO_CHAR`, `TO_NOTVICT`, or `TO_ROOM`. This indicates who it is to be targeted at.

1.3 The Parameters

Of the various parameters passed to the `act()` function, the `str` is the most important, as it is the basis for the actual final output. If this parameter is a null-pointer or points to a null-character,

then the function returns immediately. The next important parameter is the `ch` parameter. This, as mentioned, points to the central character associated with the output string and action.

`obj` is an object of type `struct obj_data *` that is passed to the function. If there is no object to pass to the function, then a `NULL` or `0` should be passed in this location.

The next parameter `vict_obj` can be a number of things ranging from a game object (`struct obj_data *`), through to a character (`struct char_data *`), and even a null terminated character array (`char *`).

Do note, however, that `obj` and `vict_obj` are both ignored if there is no control character reference (see section 1.4 ‘Control Characters’) to them and the `type` is set to `TO_ROOM` or `TO_CHAR`. In these cases, `NULL` should be supplied as the input to the function.

The `hide_invisible` flag dictates whether or not the action output should be hidden from characters that cannot see `ch`. If the flag is set to `TRUE` (non-zero), then this is the case.

The `type` determines who the output is to be sent to. There are four options for this (all defined in `comm.h`) described below:

- `TO_ROOM`: This sends the output to everybody in the room, except `ch`.
- `TO_VICT`: This option sends the output to the character pointed to by `vict_obj`. Obviously, in this case, `vict_obj` must point at a character rather than an object.
- `TO_NOTVICT`: In another case where `vict_obj` must point to a character. This sends the output to everybody in the room except `ch` and `vict_obj`.
- `TO_CHAR`: Finally, this option sends the output to the `ch`.
- `TO_SLEEP`: This is a special option that must be combined with one of the above options. It tells `act()` that the output is to be sent even to characters that are sleeping. It is combined with a bitwise ‘or’. For example, `TO_VICT | TO_SLEEP`.

When the string has been parsed, it is capitalized and a newline is added.

1.4 Control Characters

In a manner similar to the `printf()` family of functions, `act()` uses control characters. However, instead of using the `%` symbol, `act()` uses the `$` character to indicate control characters.

\$n Write name, short description, or “*someone*”, for `ch`, depending on whether `ch` is a PC, a NPC, or an invisible PC/NPC.

\$N Like **\$n**, except insert the text for `vict_obj`. NOTE: `vict_obj` must point to an object of type `struct char_data *`.

\$m “*him*,” “*her*,” or “*it*,” depending on the gender of `ch`.

\$M Like **\$m**, for `vict_obj`. NOTE: `vict_obj` must be a pointer of type `struct char_data *`.

\$s “*his*,” “*her*,” or “*it*,” depending on the gender of `ch`.

\$S Like **\$s**, for `vict_obj`. NOTE: `vict_obj` must be a pointer of type `struct char_data *`.

\$e “*he*,” “*she*,” “*it*,” depending on the gender of `ch`.

\$E Like **\$e**, for `vict_obj`. NOTE: `vict_obj` must be a pointer of type `struct char_data *`.

\$o Name or “*something*” for `obj`, depending on visibility.

\$O Like **\$o**, for `vict_obj`. NOTE: `vict_obj` must be a pointer of type `struct obj_data *`.

\$p Short description or “*something*” for `obj`.

\$P Like **\$p** for `vict_obj`. NOTE: `vict_obj` must be a pointer of type `struct obj_data *`.

\$a “*an*” or “*a*”, depending on the first character of `obj`’s name.

\$A Like **\$a**, for `vict_obj`. NOTE: `vict_obj` must be a pointer of type `struct obj_data *`.

\$T Prints the string pointed to by `vict_obj`. NOTE: `vict_obj` must be a pointer of type `char *`.

\$F Processes the string pointed to by `vict_obj` with the `fname()` function prior to printing. NOTE: `vict_obj` must be a pointer of type `char *`.

\$u Processes the buffer and uppercases the first letter of the previous word (the word immediately prior to the control code). If there is no previous word, no action is taken.

\$U Processes the buffer and uppercases the first letter of the following word (the word immediately after to the control code). If there is no following word, no action is taken.

\$\$ Print the character ‘\$’.

1.5 Examples

In all of the following examples, `ch` points to male character **Ras**, `vict` always points to the female character **Anna**. `obj1` is *a small sword*, and `obj2` is *a small sack*.

```
act("$n smiles happily.", TRUE, ch, NULL, NULL, TO_ROOM);
```

This is sent to the room that **Ras** is currently in, and the string that they see if they can see him is:

Ras smiles happily.

If a character cannot see Ras, then they will not see the action at all.

```
act("You kiss $M.", FALSE, ch, NULL, vict, TO_CHAR);
```

In this action, **Ras** is kissing **Anna**, and Ras will see:

You kiss her.

```
act("$n gives $p to $N.", TRUE, ch, obj1, vict, TO_NOTVICT);
```

The output from this string is sent to everyone in the room except for **Ras** and **Anna**. Of course, if they cannot see Ras, then they will not see any output at all. The string that each character in the room will see is:

Ras gives a small sword to Anna.

If a character cannot see Anna, then *someone* will be used in place of her name, and if they cannot see the small sword, then *something* will be used in its place.

```
act("$n gives you $p.", FALSE, ch, obj1, vict, TO_VICT);
```

Similar to the prior example, this is the output for **Anna**. She will see this even if she cannot see **Ras**, and the output that she will get is:

Ras gives you a small sword.

Just as per the last example, if she cannot see Ras, *someone* will be used in place of his name, and if she cannot see the sword, then *something* will be used in its place.

```
act("$n puts $p in $s $O.", TRUE, ch, obj1, obj2, TO_ROOM);
```

This action uses two objects rather than two characters, and is displayed to the entire room (with the exception of **Ras** of course). If the character can see Ras, they will see:

Ras puts a small sword in his small sack.

Otherwise, they will see nothing. Again, as per the prior two examples, *something* will be used in place of any objects that the viewing character cannot see.

```
act("The $F opens quietly.", FALSE, ch, NULL, EXIT(ch, door)->keyword, TO_ROOM)
```

If the keywords for the door were *gate wooden*, then this would send the output string of:

The gate opens quietly.

to all of the characters in the room with the exception of Ras.

In addition to these examples, a multitude of other examples can be found scattered throughout the CircleMUD source code.

A act () Reference Sheet

```
void act(const char *str, int hide_invisible, struct char_data *ch,
        struct obj_data *obj, const void *vict_obj, int type)
```

str: String to be parsed.

hide_invisible: If TRUE, hide from characters that cannot see the “performer”.

ch: The “performer”. Also determines the room for the output.

obj: struct obj_data *

vict_obj: Predefined string of text, or second character or object.

type: TO_VICT, TO_CHAR, TO_NOTVICT, or TO_ROOM.

\$a “ <i>an</i> ” or “ <i>a</i> ”, depending on the first character of obj’s name.	\$o Name or “ <i>something</i> ” for obj, depending on visibility.
\$A Like \$a , for vict_obj which is of type struct obj_data *.	\$O Like \$o , for vict_obj which is of type struct obj_data *.
\$e “ <i>he</i> ,” “ <i>she</i> ,” “ <i>it</i> ,” depending on the gender of ch.	\$p Short description or “ <i>something</i> ” for obj.
\$E Like \$e , for vict_obj which is of type struct char_data *.	\$P Like \$p for vict_obj which is of type struct obj_data *.
\$F Processes the string pointed to by vict_obj (pointer of type char *) with the fname () function prior to printing.	\$s “ <i>his</i> ,” “ <i>her</i> ,” or “ <i>it</i> ,” depending on the gender of ch.
\$n Write name, short description, or “ <i>someone</i> ”, for ch, depending on whether ch is a PC, a NPC, or an invisible PC/NPC.	\$S Like \$s , for vict_obj which is of type struct char_data *.
\$N Like \$n , except insert the text for vict_obj which is of type struct char_data *.	\$T Prints the string pointed to by vict_obj which is of type char *.
\$m “ <i>him</i> ,” “ <i>her</i> ,” or “ <i>it</i> ,” depending on the gender of ch.	\$u Processes the buffer and uppercases the first letter of the previous word (the word immediately prior to the control code).
\$M Like \$m , for vict_obj which is of type struct char_data *.	\$U Processes the buffer and uppercases the first letter of the following word (the word immediately after to the control code).
	\$\$ Print the character ‘\$’.