RPGsh User Manual

TheMohawkNinja

March 11, 2024

Ver. 0.7.1

DISCLAIMER: This project is entirely developed independantly. I am not associated in anyway with Wizards of the Coast, Paizo, or any other game development company. To any lawyers eyeing me up, I will not be adding information that would negate the need for players to purchase your products (e.g. spell descriptions).

Contents

1 Introduction and Basic Usage

The Roleplaying Games Shell, `rpgsh`, is an interactive and extensible shell purpose-built for augmenting player and DM gameplay for table-top RPGs like Dungeons and Dragons©, Pathfinder©, and more!

`rpgsh` provides users with capabilities similar to those found in conventional shells like `bash` or `PowerShell` a la command execution and variable assignment/modification.

1.1 The Prompt

When interacting with the shell directly, you will be presented with a prompt that will look similar to the following:

```
┌─[<NO_NAME>]─(0/0 (0))
└─$
```

The prompt contains the currently loaded character's name (*<NO_NAME>*) along with their **current/max** *(temp)* hitpoints.

As with any command line interface, you interact with the prompt by entering in either a variable or a program, along with any operators or parameters. For example, if you want to roll a 20-sided die, you would enter the following:

```
┌─[<NO_NAME>]─(0/0 (0))
└─$ roll d20
```

The maximum size of the input buffer for the prompt is 256 characters. Exceeding this may crash `rpgsh`.

2 Programs

As of version 0.7.1, the following programs are available to the user when interacting with the `rpgsh` prompt:

`banner`

Displays the ASCII art logo for `rpgsh` along with a one-line description of the program and the author's signature.

`list`

Lists all the variables in one or all scopes.

`roll`

Dice-rolling program which supports custom lists and result counting.

`setname`

Sets which variable is used for displaying the character's name.

`variables`

This is ***NOT*** to be explicitly called by the user, but is instead implicitly called when the user enters a variable as the first parameter in the prompt.

`version`

Prints `rpgsh` version.

NOTE: All programs meant for explicit call by the user and which have additional parameters contain `-?` and `--help` flags to assist the user.

3 Variables

`rpgsh` allows the user to set, get, and modify variables. Variables are arranged in a nested hierarchy through both three different scopes, and through a containerization system within each scope.

Variables in `rpgsh` follow the below syntax:

<scope><type>[<character>]/<level 1>/<level 2>/.../<level n>

Below describes each part in detail:

<scope>

A single character (sigil) representing which level of the overall hierarchy is being referenced. These are defined as follows:

`@` Character attributes. This scope encompasses all variables specific to a given character. If the *<character>* attribute is omitted, this references the currently loaded character.

These are stored in the character file for the referenced character.

`#` Campaign variables. This scope encompasses all variables in the current campaign, and are therefore available while any character in the current campaign is loaded.

These are stored in the `~/.rpgsh/campaigns/<campaign>/.vars` file.

`$` Shell variables. This scope encompasses all campaigns and is for all intents and purposes global in-scope.

These files are stored in `~/.rpgsh/.vars` , and therefore on multi-user systems, each Linux user profile will have distinct `rpgsh` shell variables.