PYRUSH USER GUIDE

Written By: Andrew Knickman
COSC 439 Project
Pyrush Team: Andrew Knickman and Adeniyi Ajayi

Pyrush Set Up

Setting up Pyrush is only a matter of pulling the code from the Git repository (https://github.com/andrew-knickman/pyrush/blob/master/pyrush.c) and then compiling and running the program.

1. Compile the Pyrush shell (pyrush.c) in a Linux operating system with the following command:

   ```
   gcc -o pyrush pyrush.c -lreadline
   ```

2. Then, run Pyrush in the system console with the following command:

   ```
   ./pyrush
   ```

Built-In Commands

Pyrush features a number of built-in commands that are passed with Papyrus syntax. The following is a list of functions, their associated commands in the command console and their formatting, the variants of those commands, and a short description of their effect.

1. **pyrushCOC**
   Command:
   `Debug.CenterOnCell() /directory`
   Changes the user's current working directory

2. **pyrushHelp**
   Command:
   `Help`
   Returns a list of Pyrush commands and links to a Creation Engine reference page.

3. **pyrushQQQ**
   Command:
   `QuitGame`
   Exits the Pyrush shell.

4. **pyrushMoveTo**
   Command:
   `Ref.MoveTo() file.txt /dir`
   Moves the file to a new directory

   Command:
   `Ref.MoveTo() file.txt filecopy.txt`
   Moves the file to a new directory or copies that file in the current directory.

5. **pyrushTime**
   Command:
   `GetCurrentTime`
   Returns the current system date and time.

6. **pyrushGetPlayer**
   Command:
   `GetPlayer()`
   Returns the system user name.
7. **pyrushGetLoc**
   Command:
   `GetCurrentLocation()`
   Returns a list of files and directories in the current working directory.

8. **pyrushGetCell**
   Command:
   `GetParentCell()`
   Returns the current working directory of the user.

9. **pyrushEquip**
   Command:
   `EquipItem() r file.txt`
   Reads the contents of a file

   Command:
   `EquipItem() w file.txt text`
   Writes user text to a file

Running User Programs

Pyrush is also capable of running user programs as one would in a normal Linux system.

1. Compile user programs as one typically would in a Linux system:

   ```
   gcc -o program program.c
   ```

2. Run programs in the Linux system style as well:

   ```
   ./program
   ```