PROS Shell Design Document

Richard Stump

# Overview:

The PROS Shell aims to be a shell system for Vex V5 systems running PROS, allowing users to type programs interactively through the terminal or run them from a text file saved on the SD card. However, instead of following the syntax of existing shells, such as bash, the PROS Shell introduces a syntax designed around the requirements of writing autonomous programs for PROS.

## Motivations:

### Autonomous Programming Workflow:

Currently, writing and testing autonomous programs for PROS and Vex V5 systems requires the following steps:

1. Make changes to the code.
2. Compile and upload the code to the robot.
3. Start/Restart the program.
4. Run the autonomous program.

A shell environment would eliminate the need for compiling and uploading the program each time. Users can run the program on their robot, and type commands interactively into the terminal to prototype their autonomous program and/or templates. They can pipe text files into the terminal to run entire programs, or save them to SD cards to run during competitions.

### Movement Controller Parameters:

Work on the rewritten API for the ARMS template showed the number of parameters that can affect robot motion controllers, and designing an API around the number of parameters shows difficulties with figuring out the order and defaults to use in the public functions. For example, the arms::move function could have any of the following parameters passed to it:

* A target distance, point, or pose
* The maximum speed to run the robot at
* The radius around the target to consider the movement completed.
* Proportional constant for linear movement
* Proportional constant for angular movement
* Flags such as thru, async, relative, etc..

C++ function overloading and default parameters were used to provide shorthand forms of this function, but certain parameters had to be prioritized first, and not to mention conflicting overloads that can occur when trying to add new overloads. Parameter order also needed to be looked up occasionally if the programmer could not remember the order of arguments.

The PROS Shell provides a custom syntax that can be used to alleviate some of these concerns.

## Potential Names:

“PROS Shell” will not be the final name of the project, just a working one. Some future names to decide on are:

* Conch
* Nautilus
* Triton
* Muscle

# Template Support:

The PROS Shell will come with support for PROS functionality, as well as popular templates from the community.

# Initial Version Roadmap:

# Public C++ API:

# Shell Language Syntax: