Streamlining Source Code Generation for a Computer Program

Pat LaVarre

Pure Storage, Inc.
Streamlining Source Code Generation for a Computer Program

Pat LaVarre
Summary

A programming system that turns a simple script provided by a user to perform one or more tasks into the underlying programming language that executes the tasks.

This white paper illustrates...

Motivation

When writing a computer program, a large portion of the source code includes various scripts that need to be written out by a programmer to perform various tasks, which is tedious and time consuming. A majority of these scripts could be written by processing logic if a user was able to simply describe which tasks are to be executed by the computer program.

Description

The improved programming system minimizes the amount of time a developer has to spend writing out programming script for a computer program. The programming system allows a user to enter a simple script that describes one or more tasks that the computer program is to perform. Processing logic of the programming system can then use the tasks included in the simple script to generate the underlying source code for the computer program. Accordingly, the developer no longer has to spend a majority of their time entering the complex script for the source code of the computer program. Rather, the developer can quickly provide a few lines in a simple script and allow the processing logic to generate the source code based on the script.

Below is an example of an embodiment of the present invention:

```bash
% argdoc.py /dev/null

#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
usage: null [-h]

do good stuff

optional arguments:
  -h, --help    show this help message and exit

examples:
  Oh no! No examples disclosed!! 🌟❤️🌟
"""
```
import argparse
import sys

def main(argv):
    """Run a command line"""

    main_module = sys.modules['__main__']
doc = main_module.__doc__

    prog = doc.strip().splitlines()[0].split()[1]
description = list(_ for _ in doc.strip().splitlines() if _)[1]
epilog_at = doc.index('examples:')
epilog = doc[epilog_at:]

    parser = argparse.ArgumentParser(
        prog=prog,
        description=description,
        add_help=True,
        formatter_class=argparse.RawTextHelpFormatter,
        epilog=epilog,
    )

    args = parser.parse_args(argv[1:])
sys.stderr.write('{}
'.format(args))

    main.args = args
    main.parser = parser

    sys.stderr.write('{}
'.format(parser.format_usage().rstrip()))
sys.stderr.write('null: error: not implemented
')
sys.exit(2)  # exit 2 from rejecting usage

if __name__ == '__main__':
sys.exit(main(sys.argv))

% argdoc.py /dev/null -- -v, --verbose say more

"""optional arguments:
- -h, --help    show this help message and exit
- -h, --help    show this help message and exit
+ -v, --verbose  say more

    parser.add_argument(
        "-v", "--verbose", action="count", default=0,
        help="say more"
    )
About the Author

Software & Firmware & ASIC/SoC design & development since 1987 with catalytic power, all the while leaning in to be "delighted by people: their phrases, their frailties, and their fleeting moments of grandeur" : -)