ARGUS
Style Guide for Command Line Specification and Self-Documentation

Brian R. Gardner

DRAFT of

26 September 1989

1. Argus Argument Types

Specifying the arguments to an Argus program fall into three basic types; they are switches, switch arguments, and objects. All optional arguments to Argus programs are represented by switches, some of which have an accompanying switch argument. Objects are names of items that the program will act upon, such as rule sets, folders, and file names.

All switches are prefixed by a dash ('-') character, and fall into one of three types:

1. Switches that take no arguments,
2. Switches that take an object as an argument,
3. Switches that take one argument chosen from a predetermined set.

Switches with no arguments are a single word long in lower case letters, and can be abbreviated by their first letter. An example is the '-help' switch, which can be abbreviated '-h'. Switches that take an object as an argument are exactly two capitalized letters long, and is immediately followed by its argument (without any space separating them). An example of this is '-RSmyrules', which specifies a rule set named "myrules". Switches whose arguments are chosen from a fixed set are a single word long in lower case letters (which can be abbreviated by using just their first letter), followed by a separating space and the valid switch argument. An example of this is '-data on', which can be abbreviated to '-d on'.

1.1 Argus Self-Documentation and Help

All Argus programs use three strategies for supplying additional information about themselves as limited self-documentation.

1. If the program is executed with insufficient or missing parameters, the program prints out a Usage message displaying all of its optional switches and arguments in the standard Unix format; the program then stops execution normally.
2. If the program is executed with erroneous parameters or specifying non-existing objects, then the program first prints an error message (through 'stderr' output stream), followed by printing a Usage message (through 'stdout'), and finally a "for help:" message; the program terminates execution abnormally, returning an error code.
3. If the program is executed with the '-help' or '-h' switch, then the program prints a Usage, argument description, example, and a "for help:" message; the program then stops execution normally.

1.1.1 Format of -help documentation

The ’-help’ switch lists helpful information in a particular style and format.

Usage — The Usage line begins in column 1 with a capital ’U’, and the program name follows, starting in column 12. The Usage line is generally only one line long; if it does fit on one line, then the second line is indented to align its first argument directly under the first argument after the program name of the first line. If there are multiple usages for an Argus program, then each one is displayed on its own Usage line beginning with "Usage:", in order of most common usage.

Arguments — After the Usage lines, there is a blank line followed by the "arguments:" section. Each line following "arguments" begins with the name of the argument as it appears in the Usage line, starting in column 4, and is followed by an explanation of the object starting in column 15. The arguments section only lists the objects allowed in the command line, not the switches themselves.

Switches — If the program has switches, then a "switches:" section appears after the arguments section. The format for the switches section follows the same guidelines and formats as the arguments section. In the description of each switch, the switch arguments are listed and explained, if the switch as an argument.

Notes — Following the switches section is a "notes:" sections. The notes section describes any additional relevant information the user may need to know, such as which combinations of switches are mutually exclusive. The notes section, if one exists, is indented by 3 spaces so that it starts in the fourth column.

Examples — After the notes section is an "examples:" section. Each of the lines of examples starts in column 4, and give one example per line of a typical invocation of the program. There should be at least one example for each usage line displayed.

Help — The last section is the "for help:" section. It gives the exact command line needed to invoke the ’-help’ option.

An example of a ’-help’ listing is:

```
1  4      12 15   Columns
|  |       |  |

Usage:     rsedit [-help] [-create] ruleset

arguments:
   ruleset    name of the rule set

switches:
   -h
   -help      Lists this short documentation.
   -c
   -create    Specifies that this is a new rule set to be created.

examples:
   rsedit myrules
   rsedit -c mynewrules

for help:  rsedit -help
```

2. List of all Argus objects

```
Object      Description
------      -----------
filename    a file name
folder      the name of a message folder
msgsrc      a message source (ex: 'mh', 'rmail', 'discuss')
msgtype     a message type (ex: 'bug-report', 'meeting', 'bug-fix')
ruleset     the name of a rule set
server      the name of a rule server
```

In addition to these "rulenum" may appear in a usage message, which is not technically an object. It is usually affiliated with the "-r" switch.

```
Item        Description
------      -----------
rulenum      an integer representing the position of
             a rule in a ruleset
```

3. List of all Argus switches

3.1 Switches that don't take arguments

```
Abbrev.  Full     Description
-------  ----     -----------
-c       -create  create something new (eg, flag if a ruleset is new)
-e       -elim    eliminate (delete)
-f       -force   force action (create new ruleset without asking)
-h       -help    User just wants to view help information
-n       -nest    process nested subfolders (recursively)
```

3.2 Switches that take fixed arguments

```
Abbrev.  Full     Set of valid args      Description
 -------  ----     -----------------      -----------
-d       -data    [all | none | msg | nomsg | rl | norl | ra | nora]
                                         specifies data collection info
-s       -state   [on | off]             specify if states are saved
-u       -uprop   [on | off]             specify saving user properties
-w       -warn    [on | off]             if warning messages are viewed
-a       -after   rulenum                insert new rule after rule #
-b       -before  rulenum                insert new rule before rule #
-r       -rule    rulenum                specify which rule # to use
```

3.3 Switches that take objects as arguments

```
Switch & Arg    Description
 ------------    -----------
-FNfilename     specify a filename for I/O
-FOfolder       specify a folder object
-LFfilename     specify a loading file
-MSmsgsrc       specify the message source ('mh', 'rmail', 'discuss')
-MTmsgtype      specify an existing message type
-NMmsgtype      specify a new message type to be created
-RSruleset      specify a ruleset
-SVserver       specifies which rule server to use
```

Table of Contents