A Simple Command Line Argument Parser

Release 1.0

Marius Staring[1] and Stefan Klein[2]

April 12, 2011

[1]Division of Image Processing, Leiden University Medical Center, Leiden, The Netherlands
[2]Biomedical Imaging Group Rotterdam, Departments of Radiology & Medical Informatics, Erasmus MC, Rotterdam, The Netherlands

Abstract

This document describes the implementation of a simple command line argument parser using the Insight Toolkit ITK www.itk.org. Such a parser may be useful for use in the examples of the ITK.
This paper is accompanied with the source code.

Latest version available at the Insight Journal [http://hdl.handle.net/1926/xxxx]

Contents

1 Introduction **1**

2 Example of usage **2**

3 Conclusion **2**

1 Introduction

Command line argument parsing is a common task for many (small) programs. We have created such a parser and have used it extensively for many years now in the toolkit praxix: http://code.google.com/p/praxix/. The parser is extremely simple, and has only three functions. One to set the command line arguments, and two to get them. Arguments are set with:

```
parser->SetCommandLineArguments( argc, argv );
```

Arguments are fetched using

```
parser->GetCommandLineArgument( "-key", argument );
```

or

```
parser->ArgumentExists( "-key" );
```

The tool assumes that arguments are passed in key value pairs, for example -key value1 value2. Keys are identified as a ”-“ followed by a string; subsequent entries that are not keys are the values. One or more values can be specified or even no values.

Arguments can be initialized to default values, which will be left untouched if the key is not provided at the command line. If an argument is initialized with a vector of `size` > 1, and if only one (1) argument is provided in the command line, we create a vector of size `size` and fill it with the single argument.

Internally, the command line arguments are stored in an `std::map` of the argument (key) as an `std::string` together with the index. We make use of the casting functionality of string streams to automatically cast the stored string to the requested type.

2 Example of usage

The command line argument parser can be used as follows:

```
#include "itkCommandLineArgumentParser.h"
...
// Create a command line argument parser.
itk::CommandLineArgumentParser::Pointer parser = itk::CommandLineArgumentParser::New();
parser->SetCommandLineArguments( argc, argv );

// Use it:
std::vector<std::string> inputFileNames; // no default
bool retin = parser->GetCommandLineArgument( "-in", inputFileNames );

std::vector<int> vecA( 3, 1 ); // using default values
bool retpA = parser->GetCommandLineArgument( "-pA", vecA );

float pi = 3.0; // default
bool retpi = parser->GetCommandLineArgument( "-pi", pi );

bool compress = parser->ArgumentExists( "-z" );
```

Arguments are passed to the command as follows:

```
executablename -in input1.mhd input2.mhd -z -out output.png -pi 3.1415926535
```

3 Conclusion

This document describes the implementation of a simple command line argument parser using the Insight Toolkit ITK www.itk.org. Such a parser may be useful for use in the examples of the ITK or on it’s wiki.