

# Using the MiPal Whiteboard Classgenerator

Mick Hawkins

5 October 2015

## Contents

<b>1 Introduction .....</b>	<b>2</b>
1.1 Supported Operating Systems .....	2
<b>2 Creating an input file .....</b>	<b>2</b>
2.1 File type and filename .....	2
2.2 Specifying your name .....	2
2.3 Specifying an alias .....	3
2.4 Specifying the variables .....	3
<b>3 Installing the classgenerator executable file.....</b>	<b>3</b>
<b>4 Running the program .....</b>	<b>4</b>
<b>5 Supported data types .....</b>	<b>4</b>

# 1 Introduction

The Classgenerator is a command line tool used to generate classes for use with the MiPal Whiteboard. It reads input from a text file and generates Whiteboard .c and .h class files, and a C++ wrapper.

It is assumed the user has general skills in use of the `bash` shell.

## 1.1 Supported Operating Systems

The Classgenerator requires MacOS X 10.9 and later.

# 2 Creating an input file

An input file must be created before using the Classgenerator. The input file specifies the variables types used in the generated classes.

## 2.1 File type and filename

The input file must be a plain-text `.txt` file. The `.txt` file extension must be used.

To correctly generate C and C++ class names:

- The filename should use lowercase letters
- The filename must begin with a lowercase letter
- The filename should use underscores between words
- Numbers may be used
- Other than in the `.txt` file extension, periods/fullstops must not be used

If the input file's name includes uppercase letters:

- These capital letters will be kept and used in the C++ filename/class (which is in camel case)
- These capital letters will be converted to lowercase for the `wb_` files

These are some examples of *suitable* filenames:

```
ball_colour.txt
oculus_prime_interface.txt
vision_goals.txt
point2D.txt
point_2D.txt
```

These are examples of *unsuitable* filenames:

```
BallColour.txt
goal.doc
WALK.txt
vision_goals
```

A sample text file `MY_test.txt` can be found in the `GUNao/posix/classgenerator/classgenerator` folder.

## 2.2 Specifying your name

As the author you may, as an option, specify your name in the input file. Your name is used in the comment at the top of each file:

- As the creator of the file
- In the copyright clause
- In the GNU license

**If you not specify your name in the input file, the system username will be used.**

Specify your name in the first line of the input file using the following format:

```
author /tab Your Name
```

- author must be in lowercase
- There must be a single tab between author and your name
- Your name must be written exactly how you want it to appear (as a suggestion, capitalised with a space between parts of the name)

Hyphenated names, and multi-word names will work as expected.

Examples of how to specify names:

```
author Captain Spaulding
```

```
author Otis B. Driftwood
```

```
author Billy-Ray Snapper
```

## 2.3 Specifying an alias

To allow compatibility with existing, older code, you may specify an alias class using the following format:

```
alias /tab alias_filename
```

## 2.4 Specifying the variables

To specify variables, use the following format:

```
datatype /tab variable_name /tab default
```

- The data type must be written as specified in section 5 *Supported Data Types*
- Specifying a default value is optional
- There must be a single tab between the datatype and the variable name, and the variable name and the default value (if specified)
- Variable names should be written exactly how you want them to appear

Currently supported data types are listed in section 5 *Supported Data Types*. Strings, Arrays and objects to be added shortly.

Examples of specifying variables:

```
int16_t pointX 5
```

```
int16_t pointY
```

```
bool is_aware false
```

```
long long bigNumber
```

Note: depending on the tab setting of your text file editor, things may not line up perfectly as above.

If default values are not specified, the following defaults will be used:

- Boolean: false
- Numerical types: 0

## 3 Installing the classgenerator executable file

The classgenerator executable is located in the GUNao/posix/classgenerator/classgenerator folder. It is called classgenerator.

To allow the executable to be run from any directory, copy it to the usr/local/bin directory under MacintoshHD. This directory is hidden. To open it, go to the Finder and, under the “Go” menu, use “Go to folder”.

If you do not have a usr/local/bin directory, enter the following in the Terminal:

```
sudo mkdir -p /usr/local/bin
cd /usr/local/bin
open .
```

...this will create and open the directory. Copy the executable into this folder.

## 4 Running the program

With the program installed in the `usr/local/bin` directory, it can be run from any location.

In the Terminal, change to the directory that you would like your generated files to be located. Put your input file in this directory also.

The name of the input file must be entered as a command line argument. For example:

```
classgenerator ball_colour.txt
```

This will run the generator using the file `ball_colour.txt` as input and will generate the Whiteboard classes:

```
wb_ball_colour.h
wb_ball_colour.c
```

To also generate a C++ wrapper for these files, use the command line argument `c` or `-c`

```
classgenerator ball_colour.txt c
```

This will generate the Whiteboard classes and a C++ wrapper:

```
wb_ball_colour.h
wb_ball_colour.c
BallColour.h
```

The command line arguments may be entered in any order. These variations will produce the same result:

```
classgenerator ball_colour.txt -c
classgenerator c ball_colour.txt
classgenerator -c ball_colour.txt c
```

Note: Command line options for a Swift wrapper and usage information will be added shortly.

## 5 Supported data types

Strings, arrays and object types to be supported shortly. The currently supported data types are:

```
bool

char
signed char
unsigned char

int
signed int
unsigned
unsigned int

int8_t
uint8_t
```

int16\_t  
uint16\_t  
int32\_t  
uint32\_t  
int64\_t  
uint64\_t

short  
short int  
signed short  
signed short int  
unsigned short  
unsigned short int

long  
long int  
signed long  
signed long int  
unsigned long  
unsigned long int

long long  
long long int  
signed long long  
signed long long int  
unsigned long long  
unsigned long long int  
long64\_t

float  
float\_t

double  
double\_t

long double  
double double