TiMRS

Timers, Made Readable and Simple

Spring 2021

Jeff Kline	jk4209	-	Systems Architect
Faisal Rahman	fr2422	-	Language Guru
Daniel Rindone	dcr2165	-	Project Manager
Eric Webb	edw2139	-	Tester

1. Introduction

Using devices to measure an elapsed duration of time has proven useful for various purposes across human history. From the earliest Babylonian water clock to smartphones, the basics of these devices have essentially remained the same: mark some starting point, mark some ending point, then “start” the clock. However, in our modern world, timers have become increasingly more complex, often incorporating a number of different procedures when facilitating a task.

TiMRS is a programming language designed to give users a new way to easily design and script complex timers for any task. This method allows one to code complex timing procedures using TiMRS’ built-in timer types with customizable durations and functionalities. These include accounting for repetition, intervals, multiple processes, saving sessions, and other techniques to track and perform any task where it may be useful.

Pending success of the basic components outlined above, we may seek to further incorporate other possibilities within TiMRS that may be more specifically useful for programmers. Examples can include enabling the execution of other programs or scripts within the framework of the TiMRS language.

2. Language Overview:

TiMRS is a mixture of Python and C orientated syntax. The use of Python keeps the programming of timers simplistic, while incorporating C allows the compiler to parse easier and have more direct program memory management, should it be needed. A major component of running a TiMRS script is providing the user with a graphical clock and some indication of stage of/progress through the timing routine.

2.1 Data Types:

Primitive data types:

Type	Description
`int`	Integer Example: 10
`string`	String: Example: “hello”
`bool`	Boolean Example: True

Complex data types:

Type	Description
`Timer`	string label hr min sec bool start bool stop
`hr`	int value, 3600 seconds
`min`	int value, 60 seconds
`sec`	int value, 1 second
`list`	array to store Timer(s)

2.2 Language Components:

Control Flow:

`//`	Single line comments
`/* */`	Multiple line comments
`if, elif, else`	Conditional statements
`for/while`	Conditional loop statements

TiMRS-specific commands:

`# rounds`	Looping condition based on '#' value
`start`	Starts a timing event
`increase`	Increments the timer by a specified amount
`decrease`	Decrements the timer by a specified amount
`del`	Removes a timer object

Logical Operators:

`||`	Standard 'or'
`&&`	Standard 'and'
`!`	Standard 'not'

Comparison Operators:

`!=, ==, <, >, <=, >=`	Standard operations for numerical relation

Mathematical Operators:

`+, -, *, /, %, ++, --`	Standard operations for mathematical arithmetic
`=`	General assignment
`!=, ==, <, >, <=, >=`	Standard operations for numerical relation

3. Example Programs:

Simple scripting:

```
start timer 5 min                       // start a timer for 300 secs

start timer 1 hr 5 min 3 sec            // starts a timer for 3903 secs

start 10 rounds:
       timer 1 min                      // loops a 60 sec timer 10 times

start 2 rounds:                         // run everything indented twice
       timer 10 sec
       start 2 rounds:                         // nested rounds
              timer 30 sec
       timer 1 min 10 sec                      //  total timer ran for 280 seconds
```

Use of the 'Timer' object:

```
Timer cook_egg = 1 min 10 sec           // creates a timer structure labeled
                                        // "cook_egg"
                                        /*
                                               Timer {
                                                   str label = cook_egg
                                                   int hr = 0
                                                   int min = 1
                                                   int sec = 10
                                                   bool start = false
                                                   bool stop = false
                                               }
                                        */


start timer cook_egg                    // runs timer x for 70 seconds

start 10 rounds:                        // loops cook_egg 10 times
       timer cook_egg

start 2 rounds:                         /*
       timer cook_egg
       start 2 rounds:                     A complex round running timers in sequence
              timer 30 sec
       timer cook_egg                   */

increase timer cook_egg 30 sec          // changes timer to 1 min 40 sec

decrease timer cook_egg 1 min           // changes timer to 40 sec

del timer cook_egg                      // deletes the timer object
```