Test Specification (TSPE) for Gpredict

Jakub Zarzyka TKIS 162632
08.12.2019

1. Introduction

This document describes test specifications for Gpredict. Gpredict is a real-time satellite tracking and orbit prediction application. It can track a large number of satellites and display their position and other data in lists, tables, maps, and polar plots (radar view). Gpredict can also predict the time of future passes for a satellite, and provide with detailed information about each pass.

Even though Gpredict consists of many modules and functionalities, this specification will cover only those in technical reach of the author.

2. Applicable and reference documents

The following are applicable:

- http://gpredict.oz9aec.net
- ECSS-E-ST-10-03 – Space Engineering Testing, ECSS, June 2012
- Gpredict User Manual

3. Definitions and abbreviations

- GPS - Global Positioning System
- ISS – International Space Station

4. Requirements to be verified

Following tests will cover various functionalities such as Module Creation and it's management. The application must be able to provide fundamental data on satellite tracking and orbit prediction based on app's design. Here is a list of some functionalities:

- Module Creation
- Module Edition
- Data update
- Log Browser functionalities
- Layout view change
- Metrical units change
- Time format and zone change

5. Test approach and test requirements

There are 4 main types of test:

- Installation test – based on build manual
- Black-box tests (functional tests) – tests for application's functionalities as mentioned in Point 4 of this specification.
- White-box tests (structural tests) – tests for application's source code. These will cover function designed to change angle format from Degrees Minutes Seconds to Decimal Degrees.
- Performance tests – tests for application's reaction time under different conditions and usage plans.

6. Test description

Description for white-box test of *dms2dec* function which is resposible for angle format change from Degrees Minutes Seconds to Decimal Degrees, e.g.:

$$30° \, 15' 30" = 30.2638889°$$

First of all, install CUnit package

```
CUnit-2.1-3.tar.bz2
```

Then, make sure you have development tools installed on your OS, then build CUnit using:

```
tar -xvf CUnit-2.1-3.tar.bz2
cd CUnit-2.1-3
libtoolize --force
aclocal
autoheader
automake --force-missing --add-missing
autoconf
chmod u+x configure
./configure --prefix `readlink -f ../CUnit`
make
make install
cd ..
rm -rf CUnit-2.1-3
```

The function itself is located in file locator.c in application's source folder. The author of this specification has prepared another file *test_locator.c* allowing for testing of mentioned function Run test_locator.c using:

```
gcc test_locator.c -ICUnit/include -LCUnit/lib -lcunit -lm -o
test_locator.out
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:CUnit/lib
./test_locator.out
```

Then, check console for results. Example results:

```
[jakubz@localhost test1]$ ./test_locator.out
-------------
dms2dec(30,15,50,0) = 30.263889 //round to 6 decimal points
-------------
dms2dec(40,20,30,0) = 40.341667 //round to 6 decimal points
-------------


     CUnit - A unit testing framework for C - Version 2.1-3
     http://cunit.sourceforge.net/



Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      1      1    n/a      0        0
               tests      2      2      2      0        0
             asserts      4      4      4      0      n/a

Elapsed time =    0.000 seconds
```

7. Test facility

Every test is performed on 64-bit Fedora built on Oracle Virtual Machine on a Windows laptop. Every additional package or library required to build application or run tests in mentioned in test descriptions on Bugzilla Project page.

8. Test sequence

The author recommends this order of tests:

- Installation test
- Structural tests
- Functional tests
- Performance tests

Proposed order is based on a typical application usage, which begins by installation and goes through functions usage up to setting up individual performance and display settings.

9. Pass/fail criteria

Pass and fail criterias are described in test plans on Bugzilla Project page and depend on test type. These may be e.g. successful button usage or function run in time under the specified amount in test plan.

10. Test documentation

Every test is thoroughly described on https://multus23.eti.pg.gda.pl/bugzilla/. For application reference see it's homepage http://gpredict.oz9aec.net/, web forum or GitHub repository.