Building and Installing GNU `units` on Microsoft Windows with Microsoft Visual Studio

Edition 2 for `units` Version 2.16

Jeff Conrad

This manual is for building GNU `units` (version 2.16) with Microsoft Visual Studio on Microsoft Windows.

Table of Contents

Preface

This manual covers building and installing GNU units on Windows, using Microsoft Visual Studio from the Windows command prompt. You may be able to import Makefile.Win into the Visual Studio IDE, but that is beyond the scope of this document.

If you have Unix-like utilities, you may be able to build and install in much the same manner as on most Unix-like systems, perhaps with a few minor adjustments. Versions 2.12 and earlier were built using Microsoft Visual C/C++ 6.0, Visual Studio Express 9.0 and 10.0, and the MKS Toolkit version 9.6 under Windows XP, SP3. Version 2.16 was built using Microsoft Visual Studio 2015 and the MKS Toolkit version 10.0 on Windows 10—see *UnitsMKS* for the details.

A Windows binary distribution is available on the project website; the resulting installation is essentially the same as that using Makefile.Win, and usually can be achieved with less effort.

The most recent build was for units version 2.16, using Microsoft Visual Studio 2015 on Microsoft Windows Professional 10 on 19 October 2017.

— Jeff Conrad (jeff_conrad@msn.com) 19 October 2017

Building from the Windows Command Prompt

If you have Microsoft Visual Studio but don't have Unix-like utilities, you should be able to build and install units from the Windows command prompt using Makefile.Win:

```
nmake /f Makefile.Win
nmake /f Makefile.Win install
```

The build requires that many environment variables be properly set; the easiest way to do this is to select Developer Command Prompt in the Visual Studio folder on the Start menu, and then change to the units source directory.

If you install in the default location, you'll probably require elevated privileges; the easiest way to do this is to right-click on Developer Command Prompt in the Visual Studio folder on the Start menu, and select Run as administrator.

By default, the units executable and data files are placed in the directory given by %ProgramFiles(x86)%\GNU\units; in most cases, this is C:\Program Files (x86)\GNU\units. On a 32-bit Windows system, the directory should be changed to %ProgramFiles%\GNU\units.

You can preview the installation directories with

```
nmake /f Makefile.Win showdest
```

If the destination directories don't exist, they will be created during installation. You can change these locations by editing Makefile.Win.

If you want to run units from a command prompt or from the Start Menu Run box, you can add the installation directory to the PATH environment variable. Alternatively, you can create a shortcut to the program and place it in a convenient location.

Icons and File Association

The installation process associates `units` data files with the `notepad` editor; double-clicking on the file icon opens the file for editing. The installation process makes `unitsfile.ico` the default icon for these files. An additional icon file, `unitsprog.ico`, is embedded in the executable file as part of the build process; this icon also may be useful if you wish to create a shortcut to the `units` program. Both icons are copied to the `units` installation directory.

Currency Definitions Updater

The script `units_cur.py` can be used to update currency definitions (if your system hides file extensions, this script will display as `units_cur`). The script requires Python (available from `https://www.python.org/`).

Installing Python

If you want to use the currency updater, install Python if it is not already installed. If you need to install Python, unless you have (or anticipate having) applications that depend on Python 2, the best choice is probably to install Python 3.

After installing Python, you should be able to run `units_cur.py` using the shortcut on the Start Menu, or if you have added the units installation directory to your `PATH`, from a command-prompt window.

When you first run `units_cur.py`, you may get a complaint about a missing module; for example

```
ModuleNotFoundError: No module named 'requests'
```

If so, you will need to install the missing module. The easiest way to do this is with the `pip` command; for example

```
pip install requests
```

If you have Python 2.7.9 or later or Python 3.4 or later, you should have `pip`, though you may need to upgrade to the latest version. If you do not have `pip`, you will need to install it manually; see the Python documentation or the Python website for instructions on how to do this.

Configuring `units_cur.py`

If you want to run the currency-update script from the command prompt without changing to the program installation directory, you will need to modify `units_cur.py` to give the full pathname of the output file currency.units, i.e., change

```
outfile = 'currency.units'
```

to

```
outfile = 'installation_directory/currency.units'
```

For the default installation directory on a 64-bit system, this would be

```
outfile = 'C:/Program Files (x86)/GNU/units/currency.units'
```

The safest approach is to run

```
nmake /f Makefile.Win showdest
```

to get the destination directory. Be sure to use forward slashes in the pathname to avoid confusing Python. The best approach is to modify units_cur.py before installation.

If you add .py to the PATHEXT environment variable, you can simply type units_cur to run the updater from a command-prompt window. You can do this from the command prompt by typing

```
set PATHEXT=%PATHEXT%;.py
```

but you'll need to do this with every new instance. You can make a permanent change by adding ;.py to PATHEXT from the Advanced tab of the System dialog: click the 'Environment Variables' button, find PATHEXT in either the list of User variables or the list of System variables; click the 'Edit' button, make the change, and click 'OK'.

Example

If you are installing units in the default location of C:/Program Files (x86)/GNU/units on a 64-bit system, the process would be to

1. Build the executable by running

   ```
   nmake /f Makefile.Win
   ```

2. Confirm the installation location by running

   ```
   nmake /f Makefile.Win showdest
   ```

   It is assumed that the program will be installed in a subdirectory of the standard location for executables (typically, C:\Program Files (x86) on a 64-bit system or C:\Program Files on a 32-bit system), and a warning is given if this directory does not exist. Ignore the warning if you are intentionally installing in another location.

3. If necessary, modify units_cur.py so that the output file is given by

   ```
   outfile = 'installation_directory/currency.units'
   ```

   By default, this will usually be

   ```
   outfile = 'C:/Program Files (x86)/GNU/units/currency.units'
   ```

4. Install the files by running

   ```
   nmake /f Makefile.Win install
   ```

5. Ensure that currency.units is writable by ordinary users. The installation should do this automatically, but if for some reason it does not, set permissions manually by adding 'Modify' permission for the appropriate groups (typically 'Power Users' and 'Users')

Running the Updater

Updating from a Command Prompt

If you have modified the currency-update script to give the full pathname of the output file currency.units, you can update the file by running units_cur.py from any instance of the Windows command prompt.

Automatic Updates

The easiest way to keep currency values up to date is by having the Windows Task Scheduler run `units_cur.py` on a regular basis. The Task Scheduler is fussy about the format for the action, which must be an executable file; an entry might look something like

```
C:\Windows\py.exe "C:\Program Files (x86)\GNU\units\units_cur.py"
```

if the Python launcher is in `C:\Windows` and the script is in `C:\Program Files (x86)\GNU\units`. The program must start in the `units` installation directory; the starting directory must be specified *without* quotes.