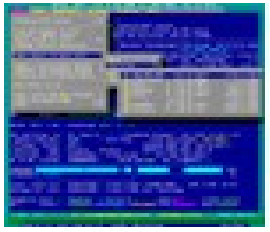
Developing text mode programs with OpenWatcom (OS/2 or Win)

Jan van Wijk

Creating simple or more complex non-GUI applications for multiple platforms from a single development machine running OS/2 (IBM, ArcaOS, eCS) or Windows

FSYS - Software



TxWin

Presentation contents

- Who am I
- Multi-platform, cross-compiling, text mode
- Some OpenWatcom facts, target platforms
- Developing using an IDE versus makefiles
- Platform dependancies and toolkits
- Some example and demo projects ...
- Openwatcom setup for cross compilation

Who am I ?

Jan van Wijk

- Software Engineer, C, Rexx, Assembly
- Founded Fsys Software in 2001
- First OS/2 experience in 1987, developing parts of OS/2 1.0 EE (Query Manager, later DB2)
- Used to be a systems-integration architect at a large bank, 500 servers and 7500 workstations

  - Home page: ***https://www.dfsee.com/***

What is ...

- Text mode (user interface)
  - An application type that uses simple textual screen output, either as a command line with sequential output or using a text windowing system.
    - As opposed to: Graphical User Interface (GUI) apps
- Cross compilation
  - Compiling software for one or more target platforms on a single development platform (ArcaOS for me :-)
    - As opposed to: Multi platform compilation (like GCC ...)

Some OpenWatcom facts

- Is an open-source continuation for the commercial Watcom compiler (Sybase)
- Two forks, openwatcom.org stable at 1.9 and newer sourceforge/github fork at 2.0
- C-compiler, C++ for many platforms ones:
  - DOS, 16-bit and 32-bit (incl DOS extenders)
  - Windows, 16 and 32 bit, text mode and GUI
  - OS2, 16 and 32 bit, text mode and GUI
  - Linux, 32 bit text mode only (in progress)
  - Various UNIX platforms, NOVELL and other niches ...

More OpenWatcom facts

- Downloads available (ZIP) for Windows and OS2 development platforms, Linux doable too, with some extra work and download
- Includes a simple IDE, and a very good debugger, both text mode and GUI
- Original OpenWatcom team, activity declining …
  See: http://www.openwatcom.org
- Alternate fork at sourceforge/github, more active
  See: https://sourceforge.net/projects/openwatcom/
  And: https://github.com/open-watcom

Why OpenWatcom ?

- Advantages:
  - Free, and adaptable when need be
  - True cross-compiler, covering most common platforms
  - Many projects can be done using just this one compiler (I used to need at least 3 :-)
- Disadvantages:
  - C++ support like templates and namespaces not at the latest standards (yet). No problem for me, but crucial for many porting projects like Mozilla and OpenOffice

Why text mode ?

- Advantages:
  - More portable across platforms
  - Works without much effort in minimal environments like boot-diskettes, CDROMs and plain old DOS
  - Appeals to command line junkies like myself :-)
- Disadvantages:
  - Does NOT appeal to typical end-user (Windows, macOS)
  - Windowing environments not readily available

IDE versus makefiles

- IDE is easier to learn for a beginning user since many options are pre-cooked and can be easily selected from a user-interface
- Makefiles are far more flexible, and much more portable to other platforms/compilers
- Automatic building for complete projects is easier to automate with makefiles
- Personally, I use makefiles exclusively :-)

Platform dependancies

- Developing for several platforms using a single source base needs to address different low-level interfaces in the OS
  - Different include files for the OS interface
  - Minor differences in C-library interface or behaviour for 'less-standard' functions
  - Different algorithms may be needed for optimum performance or other platform specific reasons

Platform separation

- There are basically two approaches to having platform dependent code organized:
  - Create specific source files for each platform, and select the correct ones in the build process
    - Useful when huge differences exist for much of the code
  - Use small segments of conditionally compiled code, with modules organized by functionality
    - Useful for smaller differences
    - easier maintenance, simpler build environment

Conditional Compilation

- To limit differences to just a few files
- To make differences easy to spot (grep)

A section for 4-platforms could look like:

```
#if defined    (WIN32)
    ...                     Windows specific stuff
#elif defined (DOS32)
    ...                     Dos specific stuff
#elif defined (LINUX)
    ...                     Linux specific stuff
#else
    ...                     default (OS2 :-)
#endif
```

Project organization

- Based on localized conditional-compilation
- Sources in one directory (small/medium project)
- Deliverables in separate directories per target, per debug/trace version and perhaps others ...

- Simple example, sources and the master makefile are in the 'project' directory:

```
cdev\
    project\
        win32\
        dos32\
        linux\
        os2\
```

- Platform directories contain a (small) makefile, the compiled object files and executables

More complex project

- Using several 'deliverable classes', sources and the master makefile are in the 'complex' directory

```
cdev\
    complex\
        shareware\
            win32\
            linux\
            os2\
        pro\
            win32\
            linux\
            os2\
        oem\
            win32\
            linux\
            os2\
```

Other variations

- You can change this to suit your own needs
  - The TxWin library and DFSee use an added level to separate the retail/debug versions:

```
c\
   dfs\
      oem\
         os2r\
         os2d\
         winr\
         wind\
         ...
```

Building a project

- The project gets build by running WMAKE on each of the platform specific makefiles
- Automated with a simple script (build)
- Each specific makefile sets the relevant definitions for its platform, and then executes the master makefile
- You can build just one, or 'ALL' targets

Sample platform makefile

- A similar file will exist for each possible combination of platform and tracing set.
- These are exactly the same for every project or delivery-class, the differences are in the master makefile (makefile.mif)

```
#OS2 retail version
target_os  = os2
target_sys = os2v2
target_env = retail
!include ..\..\makefile.mif
```

Platform section in .MIF

- Example of the platform specific part in the master makefile (OS2 retail):

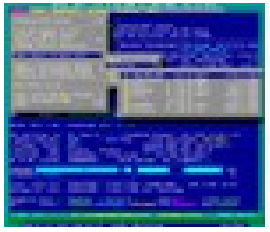
```
!ifeq target_os os2
cflags += -bm
lnkopt += libpath $(%os2tklib)
bintype = os2
comprs  = lxlite
!endif
```

  - There is a similar section for every platform
  - Another section conditionally deals with the tracing selections made (retail/trace/debug)

Developing text mode programs with OpenWatcom (OS/2 or Win)

Questions ?

FSYS - Software



TxWin