Wux: Unix Tools under Windows

Diomidis Spinellis
Department of Computing
Imperial College, London

Abstract

Wux is a port of Unix tools to the Microsoft Windows environment. It is based on a library providing a Unix-compatible set of system calls on top of Windows. Unix-derived tools run in parallel, communicating using the Unix pipe abstraction. All processes are run within an application template that gives them basic Windows compatibility such as input and output windows and an icon. The performance of the system is comparable to that of Unix ports to the PC architecture.

1 Motivation

The Unix operating system offers a wide variety of tools that can be used as building blocks or autonomous units for application development. Although these and similar tools can be ported to other environments [KP76, Gor93], their usability is often impaired by the lack of the glue elements available under Unix: multitasking and interprocess communication using pipes. Specifically, tool ports to the widely used MS-DOS system have to rely on serialised process execution and pipes implemented using inefficient intermediate files. The advent of the Microsoft Windows[1] system provided an accessible and widely used platform on which a more serious, complete, and efficient porting effort could be based.

The main objectives of our porting project were the following:

- Port a number of useful Unix-derived tools to the widely used Windows platform. As Windows applications have to be structured around a message loop, this exercise is far from trivial.
- Utilise the features provided by the Windows system to provide an environment that would be closer to Unix. Windows runs (in its 3.1 incarnation) on top of MS-DOS offering a number of advantages. The features important for our work were:
    - non-preemptive (co-operative) multitasking,
    - dynamically linked libraries with a data segment shared between processes,
    - memory management including virtual memory, and
    - a graphical user interface (GUI).

  We hoped that the first three capabilities could be used to provide Unix-like functionality, while the last one would enhance the usability of the resulting system.
- Experiment with the integration of these utilities in the Windows environment. We wanted to test the possible synergy between the traditional Unix tools and Windows concepts like the mandatory message loop structure, the clipboard, and OLE (object linking and embedding).

2 Functional Description

Wux consists of a set of native Windows executable files, each one of them corresponding to a Unix tool (Figure 1). The programs can be used in two different ways:

---

[1] From here on "Windows" is used to refer to Microsoft Windows.

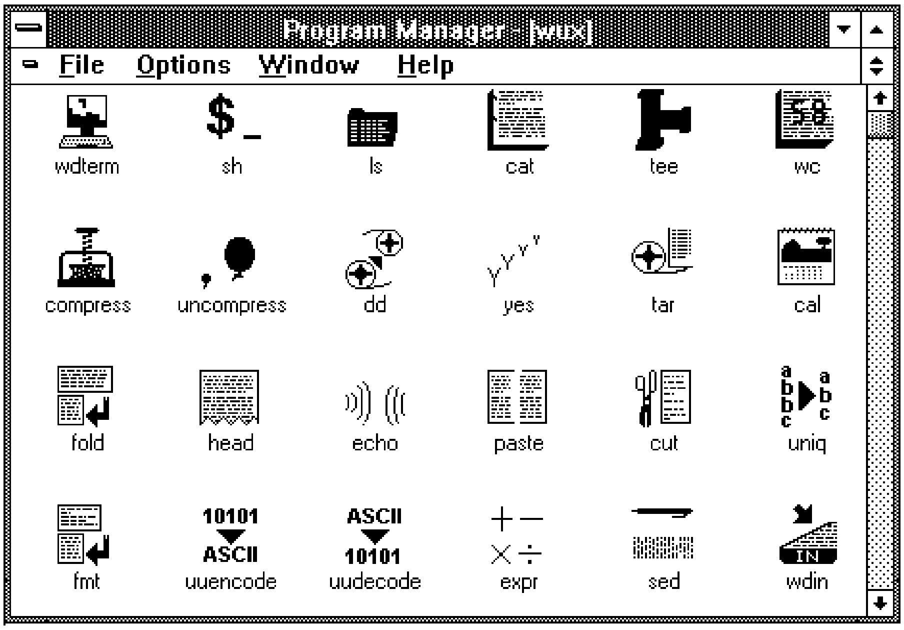

Figure 1: Program manager group of *wux* applications.

1. they can be executed from a shell-like window using the traditional Unix argument parsing and redirection functionality including pipes, or
2. they can be executed as stand-alone processes within the Windows environment using the Unix argument parsing and redirection conventions including redirection to file descriptors. To this second mode of execution we plan to add a graphical interface for argument specification and input/output file specification.

The programs run as native Windows processes, and can therefore utilise all the facilities provided by Windows, such as the ability to allocate extended and virtual memory, inter-process communication, and non-preemptive multitasking. Running processes are represented by their icon displayed at the bottom of the Windows screen (Figure 2). Whenever a process requests input from the standard input device or produces output to the standard output device and no redirection has been specified a suitable data source or sink window is created to provide that service. Data can be typed-in on a line-by-line basis in the *wdin* window which also provides an editable history of previous lines. Similarly, generated data can be displayed in the *wdout* window which provides a scrollbar to examine previous output. The standard error output is also redirected in a *wdout* window. Applications that are of interactive nature (such as editors and shells) can be executed with both input and output connected to *wdterm*, a terminal emulation window, akin to the X-Windows [SG86] *xterm* application.

3 *Wux* Implementation

In the following sections we describe the *wux* implementation. We first provide a short technical description of the Windows environment, then analyse the difficulties of porting Unix applications to Windows, proceed with an overview of the *wux* architecture and its components, and end with a section detailing how Unix applications are ported to *wux*, and some performance figures.

3.1 Technical Description of the Windows Environment

Windows 3.1 runs on top of the Microsoft MS-DOS operating system. In addition to a windowing graphical user interface, it provides many functions (see section 1) that are normally provided by an operating

[redacted]