Project proposal

Nicholas Riley
Computer Science 323
3 April 2000

1.0 The problem

As desktop operating systems migrate from a single-user, limited-security model (realistically, Windows 9x and Mac OS) to a user-at-a-time model (Windows NT and 2000) or a multiple-simultaneous-user model (Linux, Mac OS X), the organization of a user's files is less definable by the user. This is necessary to support advanced features such as network transparency, roaming, and sophisticated protection, not available earlier in desktop operating systems. The shift is evident at the most visible levels of the user interface. For example, in Windows 2000, 'My Documents' replaces 'My Computer' in the upper-left corner of the desktop. Similarly, disks are no longer present on the Mac OS X desktop; instead, buttons are provided in Finder windows to navigate to "Applications", "Documents" and "People".

Previously, a common usage pattern was to place documents together with the applications that created them. In an environment where applications are often stored on read-only networked volumes, cached locally, or referring to any of a number of servers, this physical layout is not practical. Similarly, some users may want to place related applications together; but the users may disagree on what applications are related. It would seem useful, instead of forcing somewhat unnatural usage patterns on the user, to permit such behavior by separating the appearance of file location from the file's actual location.

Another trend follows the expansion of computers worldwide. With the global nature of the World Wide Web, it is often useful for a computer to be able to display text in languages different from the language of localization of the operating system. Most operating systems already possess support for Unicode encoding, imaging, and multiple input methods hosted under a single installed localization of an operating system. The next logical step is to permit a single operating system installation and applications to display themselves in a language selected by the currently logged-in user. In fact, several current and soon-to-be-shipping operating systems do exactly this. This works well and easily with table lookups for menus, control titles, etc. However, the names of files on disk are still stuck in some language or other (usually English), which may not be the native language of the logged-in user. This is obviously not the best way for things to appear.

Finally, there has been a movement towards putting friendly user interfaces on top of UNIX-style operating systems, such as the GNOME and KDE desktops, and Mac OS X. For UNIX programs to work, the presence of a certain directory structure with directories such as /etc and /usr is required. For users, this organization with strangely named files and directories seems like a step backwards, especially for people used to user-understandable system directory structures such as in classic Mac OS. These directories could be simulated in some kind of compatibility environ-

ment, but inevitably such an approach compromises compatibility and creates porting hassles. Also, while convenient, it can be confusing to have mount points all over the filesystem hierarchy.

2.0 A possible solution

By presenting a view of the file structure that is extremely malleable and bears a resemblance to a single-user system, users will for the most part be able to continue working as they have before. The base view offered to a user will come complete with localization, so application and directory names will reflect the current language where translations are available. It would be possible through the use of visual cues to indicate, for example, that an application was stored on the network versus a local disk. This application could then be "moved" to a location more convenient for the user; it would retain its cue to indicate it was stored on the network, and would appear to move in the user's view, but the action would only affect the user's view of the location, not the application's physical location. Directories such as /usr and /etc would be completely hidden from the display of the logical structure, while still appearing to programs if necessary. Confusing filenames could be mapped to something more understandable for the user, while retaining their usual identities when older programs need to use them. Physical movement of files would still be possible, however it would require a "Send To" style action that did not correspond to simple movement of an icon onscreen.

3.0 Implementation

I would like to implement this as a mountable, browseable filesystem under UNIX, but I realize that this is a major task, so I will most likely instead build a graphical file browser that keeps its own internal state and mappings, and makes standard filesystem calls when a file is referenced.