Are File **Names** Enough?

Walter G. Piotrowski
Department of Computer Science
State University of New York
Binghamton, NY 13901
(waltp@bingvaxu.cc.binghamton.edu)

Software systems are probably the only complex system structures in which locally assigned names serve as the sole identifiers for system components.

Take, as an example, the hardware in a computer system. First, unlike software components, hardware items have distinctive shapes which give an indication of their function but, more importantly, all (or almost all) have markings indicating the manufacturer and the manufacturer's part number. These markings globally and unambiguously define the components characteristics. Someone who has a general familiarity with computer systems can locate items of importance and perform useful operations on the system (perhaps maintenance or repair) without understanding the total system structure.

By contrast, when a software component (a program or data file) becomes part of a system, it acquires a name that is locally determined, partly from its position in the local file tree and partly from the impulse of the person adding it to the system. Once named, only the person who named it knows, with any degree of certainty, what the file "really" is. A visitor[1] to the system can never be sure.

Historically, this has not been a major problem. Our systems have served user communities that have been essentially static with few visitors. In this situation, a locally assigned name is analogous to a globally unique identifier. Our world, however, is changing and a machine's user community is not apt to be either closed or static.

From a human user's point of view, moving from one system[2] to another usually requires some adaptation. The basic system commands are usually invoked the same way and have the same effect but additional software, which he may be actively using, will be different. The same software object may have different names or a familiar name may invoke a different object.

Most human users make the required adaptation through experimentation, combined with assistance from outside sources. Making this adaptation has its cost, heaviest at the time of first use of the "foreign" system and recurring, with lower cost, with each revisit to the system.

Non-human users can not be expected to have the same adaptive skills. Consider a networked system in which lightly loaded machines in the system act as computation servers for other, more heavily loaded machines. In a system in which every machine's file naming structure is administered independently, the only reasonable choice is that a remotely executing process will access all of its required files from its home, over the network. Unfortunately, a large volume of data transfer over the network, along with the burden placed on the home computer system, will make remote execution a less viable alternative.

Globally Unique Names?

A solution which seems technically simple but is politically difficult is to force a set of uniform naming rules on all of the machines within an "administrative domain". The political difficulties seem obvious and will not be

[1] A visitor may be a person or a program.

[2] We are assuming that the same operating system is in use.

belabored here. Note that, even technically, the solution has its drawbacks since it ignores the user who crosses domain boundaries and also would prevent the interconnection of existing systems in separate administrative domains.

Local Names *and* Universal Identifiers?

The naming problem discussed here comes about because we do not follow the lead of other "manufacturers" of components. We do not provide component identifiers that are independent of the system in which the components are installed.

The purpose of this discussion has been to sensitize the reader to the problem and not to solve it. A modest starter suggestion, however, is the inclusion of a file, in the root directory of every system, which would list universally assigned identifiers for the file components[3] of the system along with the locally assigned names for these components. This file could be viewed as the equivalent of a parts list. Visitors (human or otherwise) to the system could then ask if a particular standard product was available. Human visitors might also simply browse the list.

Not all files need be included in the list. Files of purely local value or those not intended to be shared could be omitted. Files intended to be shared might have prefix sections which contained their universal identifiers and system utility programs (copy, delete..) could automatically update the system's parts list as the system configuration changed.

There are many possibilities for the assignment and administration of these universal component identifiers. One is to follow the example of the Universal Product Code that is found on nearly every product in grocery stores as well as an increasing number of non-grocery products. With care, it might be possible to construct our universal identifiers so that portions would have generic meaning. That is, a few characters identifying the manufacturer, a few others the manufacturer (C compiler, Pascal compiler, etc.), others the specific version, etc. Visiting humans or programs could determine, with an appropriate degree of precision, if a needed component was available for their use.

Software components are expensive to produce. Aren't they worth the same degree of care as the components in other endeavors (integrated circuits, tires, machine screws...)?

[3] Not all files need be included in this list. Files of purely local value could be omitted.