Unix — a software marketing phenomenon

Many readers will be familiar with Unix or will at least have heard the name. Unix is a time-sharing operating system written at Bell Laboratories. The most well-known version runs on the larger PDP-11 computers, but there are also versions for other machines, either in existence (Interdata 8/32) or rumored (System/370). The reader who would like to know more about it should get the July-August 1978 issue of Bell System Technical Journal or an earlier paper in the July 1974 CACM. Without going into technical details here, let me just say that it is a very well-done, compact, and powerful system.

Although it was developed originally in a research environment, Unix has become a production system on hundreds of computers within the Bell System. It is also widely used outside Bell, both in universities and commercially. It has been available free of charge for educational use and at a stiff license fee for commercial use. Whether or not you have to pay for it, there is no vendor support; you get a tape and some manuals, and that is the last you hear from the vendor. What is worthy of special note, in my opinion, is the remarkable position of this product in the software marketplace.

First, there is the phenomenon of an operating system furnished by other than the hardware manufacturer. True, there have been lots of operating systems written by various groups for various machines; but I can’t think of that many that have found really wide appeal and success in the marketplace. Equally phenomenal is the fact that the developers of Unix did not plan to develop a commercial product. In fact, they weren’t even trying to produce a product for widespread use within Bell; they were just trying to make a nice system for research programmers to use.

Perhaps one reason for the success of Unix in the educational market is that it is free, in contrast to the vendor-supplied systems for PDP-11s. Of course the cost to the user of having no support for Unix must be weighed against the saving in initial cost. For an educational institution this balance probably weighs in favor of the free software, since there is almost always a group of users hanging around who will hack software for little or no pay. And Unix does lend itself to in-house maintenance, being small enough to be comprehensible to one person and being furnished mostly in high-level-language source form.

Another undoubted factor in the popularity of Unix is the number of programming languages, utility programs, and application programs that are available with it at little or no extra cost. This is something that happens naturally in a research or educational environment. Users come together from time to time to swap programs, operating system modifications, and rumors of the work of others in these areas. Unix facilitates the development of new software because nothing is hidden. With some popular manufacturer-supplied systems it is next to impossible to write your own compiler because the interfaces to the operating system are more or less deliberately kept secret.

With so many bright young people being exposed to Unix in universities, the lack of vendor support is not the problem it might be. Companies and individual consultants offering support for a fee are springing up like weeds. A few of these are licensed to furnish the system, with support, to their customers; the going price is not a lot higher than the price from Bell without support. Thus, Unix has created a niche for enterprise. Also, it’s not unusual these days to see employers advertising for people with Unix experience. The fact that the using organization has to become self-sufficient in software maintenance is perhaps not a great disadvantage; for with self-sufficiency comes the ability to modify the system at will to meet the special needs of the organization.

In the case of commercial users who have to pay for Unix, it must be that the perceived value is worth the price. A former student who now works for a computer company once told me why she much prefers dealing with commercial accounts.

"The university types," she said, "try to get all the service they can without paying, and if they don’t like something about company policy they take out their anger on me. But the commercial customers decide what they need and pay the asking price without complaint. And, if they are dissatisfied, it is all worked out between their lawyers and our lawyers; they always treat me nicely."

Unix, even for a price, is turning out to be a formidable competitor to the manufacturer’s operating system offerings. This can only mean that the buyers find it to have some truly superior features.

As noted, versions of Unix exist for at least two dissimilar machines, with additional versions undoubtedly in the works. While Unix is not the first or only portable operating system, it is probably the only widely used system to arouse so much interest and activity in portability. Hence, there is a definite possibility that Unix will become a de facto standard operating system. One move already in this direction is a Defense Department project to develop a militarily-secure version of Unix. Also, a representative for a computer company told me recently that his company was considering in-house development of a new operating system that would resemble Unix.

We can see at least two levels of possible resemblance here. One is a system that to the user behaves like Unix and has the same general concepts and facilities, but which would not in general accept users’ source code originally written for a different Unix system. The second level would have a high degree of compatibility with canonical Unix, such
that only portions of the operating system are specially tailored to the different hardware; all user programs and libraries could be recompiled for the target system without changes to the source code. This is a very real possibility, in that the high-level language in which Unix is written (“C”) is already implemented on several machines with a considerable degree of compatibility.

One would expect compatibility between PDP-11 and System/370 versions, given that both machines are based on 8-bit bytes and 32-bit floating point; but there is also a version of C for the Honeywell Level 66 machines which have 9-bit bytes and 36-bit floating point.

Clearly Unix is good; people are choosing it over other offerings in spite of the limited promotion given it by Bell. But is it the greatest? As with most other things, that’s a matter of opinion and circumstance. If one wants only bare Basic or beginners and nothing more, there are other systems that are easier to learn to use and are perhaps more efficient in such a limited context. In the other direction, Unix does not presently provide facilities for certain real-time system needs, such as the ability to lock a process into memory. In the world of general-purpose multi-user systems, Unix doesn’t have absolutely everything, but it may well be the best system around. Mostly it needs to outgrow a couple of its juvenile weaknesses: the limitations built into it because of its minicomputer background, and the lack of features needed in the hostile world of general public use.

If I could, today, buy any system I wanted, regardless of price, I might be more inclined to select something like the Dartmouth Time Sharing System, which can support hundreds of terminals on the big Honeywell hardware. (Given that the need is for hundreds of terminals at a single site.) But for a more realistic situation in which there are fewer users or in which funding is not available at all once, Unix would be my choice unless someone can show me something better. It will be very interesting to see what becomes of Unix as it is adapted to larger machines than the PDP-11s. It will need to become a multiprocessor system, such that a big processor that can run big programs is not constantly being interrupted by the trivia of terminal input and output.

In summary, Unix is going to shake up some conventional thinking about software marketing. It has the following aspects which, if not unique, are at least interesting and important:

(1) You can now buy bare hardware from one vendor and get the software elsewhere. This has some further implications for mixed-vendor hardware; it is not strictly necessary for the peripherals to be absolutely compatible with those offered by the CPU vendor.

(2) You can run the same application programs and have the same operating procedures on computers from different vendors, as well as on machines of different sizes from one vendor.

(3) By making a software system available free to universities, the developers enhance the system’s marketability by virtue of the additional public-domain software that will be written for it.

(4) By not being excessively secretive about the source code for the system, the developers ensure the creation of a large body of people qualified to maintain it in the field.

(5) It’s possible to develop and sell software without having a field maintenance organization—indeed, without making any commitments to maintenance at all. If the product is good enough, third parties will come forth to perform this function.

(6) It’s possible to go into the hardware business without committing to an everlasting, full-blown software development effort. (Those who make CPUs plug-compatible with IBM are already doing this; but it’s a little different when you don’t have to worry about your software supplier being your hardware competitor.)

(7) The world might be about ready for a standard operating system, and one not imposed on it by a dominant equipment vendor.

(8) It is really possible to have a full-featured software system that is flexible and powerful, yet simple, modular, and comprehensible.

Footnote: I’ve been wondering how Unix got its name. Not having access to any inside information, I’m now ready to propose a theory, in hopes that someone who really knows will come forward to confirm or deny. The very earliest version of Unix was a single-user system, done quite a few years ago when Multics was a big project of MIT, Bell Labs, and GE. Multics was/is a very large multi-user system. Seeing that Multics rhymes with Unix, it’s tempting to believe that Unix is derived from Uni- for one user versus Multi- for multiple users. Perhaps the ending in “x” instead of “cs” simply reflects that the Multics is an acronym and Unix is not; and the shorter word Unix is preferred because it runs on a machine with a shorter word length.

J. H.

COMPUTER