HISTORY OF THE C PROGRAMMING LANGUAGE

This material was written by Don Anselmo, an AT&T / Bell Laboratories Engineer and Manager (1958 to 1986). As Director of Computer Development at the Labs, Don was intimately familiar with the development of UNIX, C, C^{++}, and the ATT PC Computer line. The paragraphs below are taken from the listed References.

In 1969, after cancellation of the Multics project, a collaborative effort to build a multi-user operating system (between GE, Bell Labs and Project MAC at MIT), participating members of the Bell Labs staff were still intent on creating such a facility. So an effort was started at the laboratories by Ken Thompson, Dennis Ritchie, and others to design the file system for a new operating system.

In his spare time, Thompson had developed a computer game called "Space Travel" that simulated the motion of planets in the solar system. A player could cruise between planets, attempting to land the ship. First written on Multics and then rewritten in FORTRAN for the GECOS operating system on a GE 635 computer, the game's display was jerky and hard to control because the player had to type commands to control the ship. Also, it cost about $75 in CPU time on the GE 635. Thompson soon found a little-used PDP-7 computer with an excellent display terminal, and with help from Dennis Ritchie, Space Travel was ported to this machine. This was written in assembly language for a cross-assembler that ran under GECOS and produced paper tapes to be carried to the PDP-7. Though it made a very attractive game, it was difficult to prepare the programs for the PDP-7. Thus, the PDP-7 became a natural candidate to develop what was to become the UNIX operating system, and development proceeded in the PDP-7's bare assembler - with no loader or link-editor.

One day Thompson decided that Unix needed a FORTRAN compiler, so he started to design it. After a day or so, he decided not to do it. Instead, he wrote a very simple language he called B, which was based on the word-oriented BCPL. It was also influenced by Thompson's taste for spartan syntax, and the very small space in which the compiler had to fit to run on the PDP-7. It worked, but there were problems. First, the implementation was interpreted, so it was slow. Second, the word-orientation of B did not match the new byte-oriented machines, like the new PDP-11.

In 1971, UNIX was moved to a PDP-11. Although the first version of UNIX was written in assembler, Thompson's intention was that it be written in a high-level language. Ritchie used the PDP-11 to add types to B, which for a while was called NB for "New B." Then he started to write a compiler for it. The first phase of C included some language changes from B, i.e., adding the type structure without much change in the syntax, and the compiler. This version of C did not have structures to support tables. After adding this and some other facilities, a concerted effort was made to redo the operating system in C.

In their book, **The C Programming Language**, Kernighan and Ritchie, [5], C is described in the first sentence of the Preface: "C is a general-purpose programming language which features economy of expression, ..." In Chapter 0, it is described as a relatively "low level" language, and that "A compiler for C can be simple and compact. Compilers are also easily written; using current technology, one can expect to prepare a compiler for a new machine in a couple of months, ... "

Quotes From Dennis Ritchie, in the UNIX Oral History Project [4].

C was an adaptation of B; that was pretty much Ken's. B actually started out as system Fortran. Ken one day said the PDP-7 Unix system needed a Fortran compiler in order to be a serious system, and so he actually sat down and started to do the Fortran grammar. This was before *yacc*; he actually started in TMG. ... Anyway, it took him about a day to realize that he didn't want to do a Fortran compiler at all. So he did this very simple language called B and got it going on the PDP-7. B was actually moved to the PDP-11. A few system programs were written in it, not the operating system itself, but the utilities. It was fairly slow, because it was an interpreter. And there were sort of two realizations about the problems of B. One was that, because the implementation was interpreted it was always going to be slow. And the second was that, unlike all the machines we had used before, which were word-oriented, we now had a machine that was byte-oriented and that the basic notions that were built into B, which was in turn based on BCPL, were just not really right for this byte-oriented machine. In particular, B and BCPL had notions of pointers, which were names of storage cells, but on a byte oriented machine in particular and also one in which the -- had 16-bit words and -- I don't think it did originally, but they were clearly intending to have 32-bit and 64-bit floating point numbers. So that there all these different sizes of objects, and B and BCPL were really only oriented toward a single size of object. From a linguistic point of view that was the biggest limitation of B; not only the fact that all objects were the same size but also that just the whole notion of pointer to object didn't fit well with .... So, more or less simultaneously, I started trying to add types to the language B, and fairly soon afterwards tried to write a compiler for it. Language changes came first. For a while it was called NB for "New B"; it was also an interpreter, and I actually started with the B compiler. ... because C was written in a language very much like itself, at every stage of the game, so, yes, it must have started with the B compiler and sort of merged it into the C compiler and added the various, the type structure. And then tried to convert that into a compiler.

The original version of C did not have structures. So to make tables, e.g., process tables and file tables, was really fairly painful. One of the techniques was to define names who were actually small constants and then use these essentially as subscripts to pointers -- basically a pointer offset by a small constant that was named; the name was really the equivalent of the name of a field of a structure. It was clumsy; I guess people still do the same sort of thing in Fortran.

Over the year, I added structures and probably made the compiler somewhat better, so over the next summer, we made the concerted effort to redo the whole operating system in C.

References

1. **The C Programming Language**, Kernighan and Ritchie, Prentice Hall, 1978.
2. *The Unix Time-sharing System*, BSTJ, Vol. 57 No.6, July- August 1978.
3. *Evolution of the Unix Time-sharing System,* Ritchie, D.M., ATT/BLTJ, Vol. 63 No.8, Oct. 1984.
4. *The Unix Oral History Project,* Michael S. Mahoney, http://www.princeton.edu/~mike/expotape.htm
5. *Development of the C Language*, Dennis M. Ritchie, Second History of Programming Conf., 1993, http://cm.bell-labs.com/cm/cs/who/dmr/chist.html

BRIEF RESUME OF DON ANSELMO (1936 - 2006)

EDUCATION

BS, Electrical Engineering - VPI
MS, Electrical Engineering - NYU
CDT Program, Bell Telephone Laboratories
Post Grad Mathematics - Stevens Institute

HONORS

ETA KAPPA NU
PHI KAPPA PHI
TAU BETA PI

GENERAL BACKGROUND

Director Interconnection Technology, ATT Bell Labs
Director of Computer Development, ATT Bell Labs †
Director of Product Management, ATT Technologies ††
President and COO, Open Connect Systems Corp.
Vice President, Technical Systems Division - Motorola Computer Group
Director, Telephony Products Office - Satellite Communications, Motorola
Director, Strategic Program Development, Network Solutions Sector, Motorola
Consultant/Director Prediction Systems, Inc.

† As Director of Computer Development, Bell Labs: Responsible for the development of 3B computer line and the UNIX operating system, contributing to the commercialization of UNIX as an ATT product.

†† As Director of Product Management, ATT Technologies: Oversaw product management for the 3B computer line, ATT PC, and Olivetti relationship for 3B distribution in Europe. Controlled management of corporate investments by ATT in several third party strategic relationships. Part of the board membership representing ATT in strategic investments.