My Adventures with UNIX

Why am I writing this? At least in the case of Unix, it seems that I unwittingly participated in History. Recently, some of what I've read online or seen on YouTube about the early days of Unix has irritated me. So, I'm writing my version. Caviat: this is my personal account — not backed with research.
Dick Haight — 9/21/2023

Me, circa 1971
(now hard to believe)

I transferred to AT&T Bell Telephone Laboratories (hereafter Bell Labs or BTL) in 1967 as one of several hundred AT&T Bell System programmers (and managers) who were dumped into Bell Labs to form BISP – the Business Information Systems Project. We were supposed to solve computer problems that were common to the 23 Bell System Operating Companies. These projects included computerizing customer records and billing, inventorying equipment, the white and yellow pages directories, etc. Judge Green and "The United States v. AT&T" were still in the future. To accommodate this effort a whole new Bell Labs organization was created.

So how did the existing BTL organizations feel about this? Irritated and slightly degraded, probably. For one thing the ratio of PhDs to total employees certainly took a hit. I, myself, had a mere BA in English Lit. After my "lifetime" career in the Michigan Bell Traffic Department was automated away (due to the introduction of customer direct dialing) in just six years in 1961, I became a programmer simply by passing the IBM "Programmer's Aptitude Test" – just a bit easier than the not-yet-invented university degree in Computer Science.

Bell Labs Holmdel
Now abandoned, weeds in the parking lot
Corporate planning at its finest

By the time I arrived at Bell Labs Holmdel, NJ, I'd had an exposure to the first three computer generations: a bit of IBM EAM equipment's logic board wiring (first generation) followed by extensive assembly language programming on second generation IBM 1401s and 7074s. And by 1967, I'd also written assembler, Cobol and PL/1 on IBM 360s and a bit of Algol for the wonderful Burroughs 5500. Early on in my programming career I'd decided that writing software for other programmers was more rewarding (and interesting) than the usual drudge of commercial applications. And that, of course, had caused me to be drafted into BISP and Bell Labs.

IBM 1401

So slow that instruction timing was given in fractions of a millisecond
60 years on I remember all the op-codes

I started at Bell Labs, Holmdel, NJ but I wasn't at that building very long. Just long enough to get to know a couple interesting programmers: Robert Wilson (as in the yet-to-be Nobel Prize winners Penzias and Wilson) and Ralph Griswold (main author of SNOBOL4, my all-time favorite programming language but now sadly forgotten). Holmdel had only recently swapped their dual maxed-out IBM 7094s for System 360s. At that time the program development world was still 99% keypunched cards in, printouts back. At Holmdel Labs there was an "expectant programmer's" waiting room. And if you hung out there all day you might get a half dozen batch program test shots per day. Twice as many as if you returned to your office. Denizens of the waiting room soon sorted themselves out. The more you were there the closer you got to the window where listings/dumps/card-decks were returned. The center of the room was taken up with a half dozen keypunch machines. Wilson and Griswold were top of the pecking order but I was close.

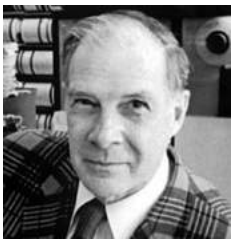
Penzias and Wilson with their microwave horn
aimed at the beginning of the Universe

Some time that winter BISP outgrew the space at Holmdel and we moved around through temporary quarters before ending up in an eleven-story new building in Piscataway, NJ where I worked until I escaped BISP. By 1971, some of the Bell Company people had started moving back to their home organizations. Another Bell System Traffic Department retread called me one day. My little group had picked up a reputation as serious hackers — i.e., as programmers who got things done *regardless*. So, this Jersey Bell Traffic guy asked me if I could change a Digital Equipment Corporation PDP-11 BASIC compiler so that in place of printing results it punched out paper tape (so telephone circuit usage results could be collected and summarized later). Apparently, this was OK by his DEC salesman, but he couldn't do it himself. I called that sales guy. He gave me the assembler source for BASIC on magnetic tape and the executable program on paper tape. I asked, "Where can I compile it?" The DEC guy said,

"You could try Ken Thompson. He just got a big new PDP-11." Ken was in the *real* Bell Labs – "Research," organization number 1200-something. So, I called Ken and he told me that I couldn't assemble the DEC code on his machine because he wasn't using byte-one of DEC's official software. Wow. Shocking! I'd never considered totally cutting loose from vendor software. This could have been the end of this story but Thompson next said, "However, if you like to play around on our new UNIX system I'll give you a login." Why not. I was familiar with IBM's ugly and inefficient TSO (time-sharing option) and had used the UNIVAC-1108 Demand offering extensively. In fact, I'd written a little UNIVAC utility that simplified batch job submission. It was more popular with the 200 or so programmers than with the comp center people who complained that was executed more often than any other program. BTW: we did fix the Traffic guy's problem: by physically splicing in the paper tape binary code for PUNCH-TAPE in place of the code for PRINT.

My next day's company mail contained a Unix V1.0? manual – a half inch thick, Xeroxed output printed by a Teletype Model 37 terminal (upper and lower-case, all the special characters eventually used in C Language, 15 CPS). I went looking for my own TTY-37. Found one, seemed unused, midnight requisitioned it. It was so heavy I needed a dolly to move it. Read the Unix manual (20 minutes?). I dialed up, logged in and was hooked in another 20 minutes—à la St. Paul's vision (fit?) on the road to Damascus. I'm not exaggerating. I knew plenty about how not to do the time-sharing interface – so I could appreciate something way superior when I tried it.

Thompson's "boss"[1] in Research was Doug McIlroy. I'd known McIlroy for a few years. I'd been nominated as the BISP (sacrificial) interface person to Research – in case they produced something practical that BISP could use. During that time, I had done Doug a small favor. He knew that I ran short seminars/talks for BISP programmers when I had a topic. They were scheduled Fridays at 2:00PM. McIlroy suggested that he would appreciate having his co-worker, Richard Hamming (Turing Award, everything short of the Nobel), invited to talk at a few of my seminars. Hamming was rightly famous for inventing the computer error correcting codes without which our current cell phones and the Internet could not exist[2]. However, these accomplishments were in the past.



Richard Hamming — *with signature loud sport coat*

In *that* present, Hamming liked to drop in on other still-productive researchers and spend a few hours describing his past glories. My suggested assignment from McIlroy was to distract Hamming occasionally. To be fair to Hamming, his stories were very interesting if you were hearing them for the first time. So, I got to know Richard. He came to one of my talks. He suggested topics. From then on, my plan went like this: around 11AM I'd drive to Murray Hill (home of the real, sincere Bell Labs), pick

[1] Ken could be steered to a certain extent but bossed or supervised — unlikely.

[2] It was the *moment* for error correction. *Someone* was going to invent it. Note: this was the age of the Vietnam War, the draft lottery, anti-war sentiment and new BTL hires with long hair and beards, t-shirts and jeans. Richard was an unhappy hawk.

up Hamming and take him for a long, two-martini lunch, then on to my seminar room for his talk. My minions made sure that he had a decent audience. After that I'd fill another hour about what my group was working on (E.g., at that time we were developing a programming language called Snoflake – a compiled-out subset of SNOBOL4 that could be linked to Cobol programs). Then I took Richard back to Murray Hill just too late to do anything but drive himself home. I did this three or four times. I forget. Anyway, this may have had something to do with me being welcomed into the Unix world. Or not.

Background on the birth of Unix. Again, just my recollection.
Bell Labs had been in a 3-way 1960s development effort between the U.S. Government (military?), General Electric[3] (the presumed computer vendor) and BTL. The project was called Multix (implying multiple processors). Development being done on a GE-635 (an IBM-7094 wannabe) for the mythical GE-645 (never completed, I think). After a few unproductive (frustrating?) years BTL pulled out[4]. At which point (1969) a frustrated BTL programmer, Ken Thompson, wrote Unix V0.0 on an unused Digital Equipment PDP-9 – sort of. He then convinced the BTL Patent Department[5] that he and friends could provide them with a multi-user computer word-processing system customized for patent application preparation. Somehow that worked and Thompson got funding to buy a PDP-11/20 to provide that service. In that version everything was written in assembly language. Thompson seems to have written the assembler, the Unix kernel, the shell, various utilities, and the **`ed(1)`** text editor[6]. (Ken must have used the DEC assembler at least long enough to write *his* assembler.) Joe Ossana wrote the word processing program **`roff`** ("run off", later nroff and troff). This was Unix V1.0. That's about when I got on board. I was warned (by email! — which I'd never heard of before) not to test programs during Patent Office working hours because system memory was not protected from user program errors.

Ritchie and Thompson at the original PDP-11/45
posed, neater than usual, in the un-airconditioned attic at Murray Hill

Pretty soon after (late '71?) the computer was upgraded to a PDP-11/45 with system/user memory protection! Unix V1.1, I think. Still all assembler. But Denis Ritchie was working on language **B**. I think the naming went something like this: Algol60 morphed into CPL (Cambridge [University] Programming Language), then came a scaled down version: BCPL (Basic CPL, I had tried an IBM-360

[3] In those days the U.S. computer business was referred to as "Snow White and the Seven Dwarves." IBM was Snow White, the rest were Univac, General Electric, Honeywell, RCA, Burroughs, Digital Equipment and Control Data (maybe NCR: 8th dwarf?).
[4] Again, Anti-Vietnam War sentiment probably played a part in the decision.
[5] Bell Labs applied for thousands of patents per year.
[6] Note on the short command names: typing on a 1970s Teletype was slow work. Each key had to be depressed, firmly about a half inch.

implementation. The only data-type was the machine word) and this became **B** (for short), sort of. For a while **B** changed every day. No backward compatibility. Eventually someone (Doug McIlroy?) suggested it should have a new name and Language **C** was born[7].

During this early period, I got into the habit of dropping by the Unix Lab about every week (BTL-speak: a "Lab" = any room not an office where work is done). I was tolerated. I was never introduced to anyone; it took a while to work out the cast of characters. McIlroy and Dennis Ritchie (a kindly character) might say hi. Otherwise, I was invisible. A note about work habits: these people generally worked (à la left-over second-gen university culture) at night – the only time a programmer could lay hands (figuratively) on million+ dollar second generation mainframes. Thompson tended to arrive at work about 2:00PM. He and friends would have late lunch in the very nice white-tablecloth part of the Murray Hill dining room. The real work at the Unix Lab didn't start until around 3:00 and might run all night.

A memorable drop-in at Unix Lab:
On that particular day, as I arrived, Robert Morris was the center of attention. He was crying tears and snot (I think) which was dripping through his beard into the type box of his TTY-37. The problem was the then-current Unix maximum file size: 64k bytes (amazing, eh?). Morris was trying to complete his desk-calculator program, **`dc`**. It turned out that even after segmenting the program as much as possible the **`as`** (assembler) program tried to create a temporary file longer than the magic 64k. At that point Thompson said something like, "Oh, all right – I'll add pipes." Seems to me that the PDP-11 was rebooted within 30 minutes, pipes worked and the assembler was modified to use them (more importantly, also the shell). Pipe data was passed between assembler program phases without creating a file. Problem solved. I assumed that this feature was Ken's idea, but I happened to say this in front of McIlroy once and he quickly set me straight that pipes were *his* idea, not Ken's. Morris (who did the encryption for the **`passwd`** command) was not amused at me being there. More about him later.

1974: BISP had hundreds of programmers. It took me a while, but I eventually convinced my BISP management that using Unix systems as cheap time-sharing front ends to the IBM and UNIVAC mainframes made sense[8]. My group got the then world's biggest Unix system: a PDP-11/45, 48k 16-bit words of core memory, 2 40mb disk drives, 32 serial ports (16 dial up) and a phototypesetter. BISP programmers edited their programs, submitted mainframe jobs over data links and got their result back as a file instead of a pile of paper (max file size had been increased). This was instantly popular. I was soon administering three PDP-11/70s (and growing) – which led me (out of self-preservation) to write the **`find(1)`** and **`cpio(1)`** commands to facilitate file back-up. I believe that my *find* command was

[7] The PDP-11 had a lovely instruction set (especially next to the IBM-360). The following instructions copied a character string from the address pointed to by R1 to that pointed to by R2, incremented both registers and set the condition code to zero if the character value was zero. All in one 16-bit instruction. Think that affected the **C "++"** operator?

```
COPYCH MVC R1+, R2+
       BNZ COPYCH
```

[8] Ok, it wasn't just about cheap effective timesharing: There was the "source code control system". Managers of programmers have a never-ending urge to control programmers — not that any of them actually ***read*** code. But **`sccs`** was much of why these systems were funded. The resulting Unix *package* was called "The Programmer's Workbench".

the only code in the first *official* Research Unix software distribution that was not written by someone in the Research group[9].

Beyond BISP: by 1977, there were over 200 distinct Unix-based development projects within BTL and Western Electric. To serve them a Unix Support Group (USG) of about 20 people (department head, two supervisors, a secretary plus Members of Technical Staff) had been created within the BTL computer center organization. USG had 200 systems to maintain and I had four. But a few of those "200" noticed that I kept my computers more up to date with Research Unix. I had a half-dozen MTS[10] working for me but I mostly did the Unix stuff myself. I had a BTL off-premises phone line and a terminal in my basement. During evening TV commercials, I ran up and down the stairs doing system "admin". And about once a week I copied anything new (all source code of op-system and commands) from Research Unix to my master PDP-11. This was done between then "fast" 2000-Baud half-duplex modems. I, of course, used my `find` and `cpio` commands to select what had been updated. If worthwhile, I then compiled everything and rebooted my system. Note, it wasn't up to Hamming's standards, but I did a checksum on the data transfer. When that worked, and I don't remember it ever failing, the new Unix would be installed in my other systems over the weekend.

Anyway, after a bit some of the "Supported" USG projects jumped ship and installed my version of the system. Big surprise, as mentioned, I kept my system up to date as an off-hours hobby – something the official group couldn't manage. Of course, they had fractious customers pulling in several directions and had wounded themselves with paperwork.

Other things I had a hand in:
The day that the **`struct`** composite datatype was added to **C** (1973?), I asked Dennis Ritchie whether self-referencing `struct` pointers were allowed. He said yes and I wrote a recursive tree sort[11] example that afternoon. My code is in the first edition Kernighan and Ritchie C language book. I also helped talk Dennis into adding named bit fields and an escape into assembler to the language. I probably wasn't the only one asking. In addition to `find` and `cpio` I also added true variables to Thompson's original shell (it became known as the Mashey shell — Research went with the Borne shell, instead). I got named pipes added (called ***fifo*** files) and I wrote `popen(3)` subroutine – to use fifos. Note: Research wrote `tar(1)` (short for **t**ape **ar**chive), a better (in some ways) version of **`cpio`**.

I noticed how quickly **`tar`** followed **`cpio.`** This suggested to me how I might fix another admin problem: a power failure or disk drive failure generally meant the loss of open files. I was doing a full file backup once a week and ran cumulative incremental backups of "important" files every night. I needed a way to make this automatic, i.e., to arrange for the execution of shell commands to be run off-hours and on a repeating schedule. I couldn't get Research interested so I assigned a summer intern to write such a program. He made a mess of it but by the end of the summer he turned in something that

[9] Over the years, I've received a lot of complaints about **find**'s command line syntax. The day I wrote the first version of the code I emailed it to McIlroy for comment. He liked it but *he* chose the syntax. BTW: I think that if I got 1¢ for each time **find** was executed on all of the Unix and Linux systems for just one week I'd be rich as Bezos.
[10] Member of Technical Staff – top grade tech person.
[11] I got the idea for this from a film Ken Knowlton made about his L6 programming language. Knowlton was BTL Computing Research/**Holmdel** branch rather than **Murray Hill** Branch (like Penzias, Wilson, Griswold, and London and Reiser—who added demand paging to VAX Unix which became the basis for Berkley versions). I guess there was *history* between these groups. My C comment mentioning Knowlton didn't make it into the K & R book.

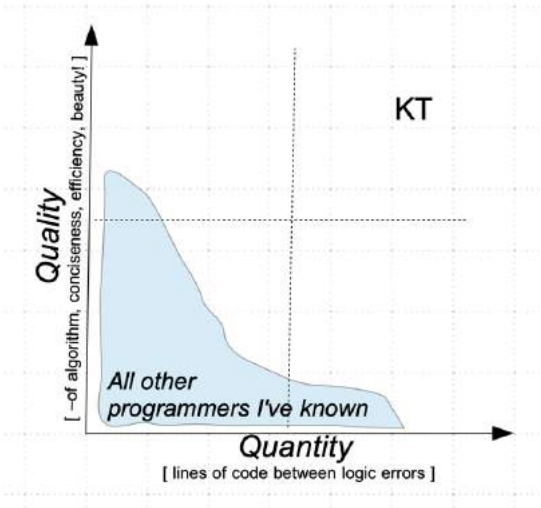
sort of worked. I immediately sent a description and the source code to Ken Thompson. The next day we had the **`cron`** (a.k.a. **crontab**) program — which was 100 times better and it is part of all of the millions of Unix and Linux systems to this day. That time I *clearly* got Ken to do something!

What else?
Big surprise, I transferred to the USG. Stayed there until 1981. By then the official Unix work had become political rather than technical and I was encouraged to move on. This is a whole different story (and USG became part of the so-called "Unix Wars").

Back to Robert Morris: In 1977 (plus or minus) I was the first to install a Unix system in a high school. I used the regulation Research "Academic License" version. Morris complained bitterly about this, suggesting that it would lead to nothing but high-school-aged hackers breaking into Unix systems. Didn't happen, but (*schadenfreude* was invented just for this) in 1988, Morris's son, Robert T. Morris, Jr., created the "Morris Worm" which infected thousands of Unix systems[12]. Junior was convicted under the "Computer Fraud and Abuse Act" — got off with a fine, community service and suspended sentence. Morris senior was by then working on "security" for NASA.

Ken Thompson: Programmer. (not really off the topic)
Toward the end of the '70s, in preparation for moving Unix to other machines, all of the Unix manual Section-1 commands had to be converted from assembly language to **C**. My group in USG got most of that chore. I assigned a brand new MS/CS hire to rewrite Ken's tick-tack-toe (**`ttt`**) game. As I recall Ken's version was 87 lines of assembler. In an unkind moment, I suggested that the **C** version should be shorter, **C** being a higher level language. I think it took over 200 lines in **C**.

Here's my view of Ken's ability:



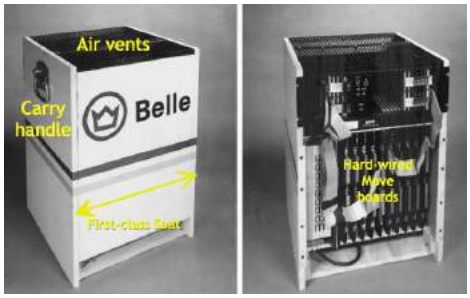
And if your ego needed eviscerating, just send him some source code for comment.

[12] By then, these were mostly licensed BSD (Berkley [University] Software Distribution) Unix versions running on DEC VAX or Sun Micro hardware. BSD was the best version of Unix — USG's unfortunately, lagged behind.

Ken

"One of my most productive days was
Throwing away 1000 lines of code."

Ken Thompson and computer chess: Thompson occasionally joked that he helped create Unix so he'd have a productive environment for writing his chess-playing program[13]. Ultimately, his avocation led to the creation of the "Belle" chess-playing computer. Thompson wrote the software and his hardware buddy, Joe Condon, created a chess-move-evaluating hard-wired device that calculated the value of 8 potential chess moves simultaneously.



BTL let Thompson and Belle travel to chess tournaments. Belle won everything for a few years (until IBM squashed him with "Big Blue"). Ken and Belle flew first class because Belle's enclosure was designed to just fit next to him in a first-class seat. This worked well until he was invited to play in Moscow. This was late USSR times. The only direct flight was by Aeroflot and true to Soviet egalitarian principals, it offered no first class. So, the only way to get Belle to Russia was as checked luggage. Ken ok'd that. Someone alert at JFK Customs decided that was a bad idea and confiscated Belle. Thompson had a fit. He claimed that the only way Belle could hurt anyone would be to drop it on them. I think he was being naïve. I suspect Belle would have been "lost luggage" for a few days, at least. Further, I suspect that today's Nvidia graphics processors owe a bit to Belle's design. Belle itself is now in the Smithsonian's computer collection.

By the late'70s Ken considered Unix finished and moved on to Plan 9. In 1992, according to Rob Pike, Ken (writing on a diner placemat) developed the UTF-8 multi-byte character encoding scheme. UTF-8 makes the Internet work for the world's text. IBM generally gets the credit but I knew a committee

[13] BTW: one should be careful about things Thompson has said about his motivation. When he said he did the first-try PDP-9 Unix so he could play Spacewar — or that he created the later Unix to have an efficient environment for writing his chess program: I think those were his responses to repeated, irritating questions from the media (or AT&T management). The phrase "doesn't suffer fools gladly" was a nice fit for Ken. Perhaps his worse quip was by naming the Plan 9 operating system after a really dumb 1959 scifi movie. INMHO, the world would be in a better place if the Linux/Unix server farms were running Plan 9.

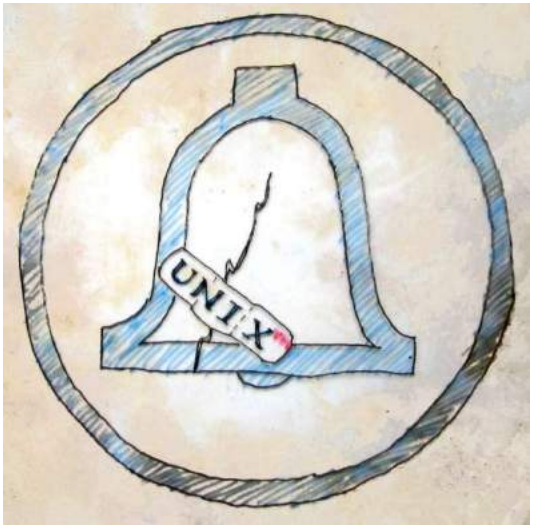
could never design anything this clever and efficient. Last I checked, Ken was at Google and had a hand in developing Go language.

Last gasp:
About programming productivity: The second generation/early third generation routine (keypunch, card deck [don't drop the cards], listings and core dumps) was bad but the computers were slow and very expensive. The Unix timesharing model (edit, compile, run; 30 second turn around) was way superior but not quite as great as I had hoped. The old-old times 2-tests-per-day did eventually produce working programs. Slow turn-around forced you to fix more than one problem at a time and a fresh paper source listing was helpful (if wasteful).

Programming tools that sort of worked when most programs were input->compute->output don't work as well for programs designed to run forever. And then there are the "exceptions": the 90% of the code is about errors that seldom, if ever happen. Languages like Python (where bugs may only be found when source code is executed) aren't part of the solution.

Ah well.

This is probably enough. About me? I went on to work on "interactive video" — much of what's everybody's smart phone now but then the size of a fridge and costing thousands. Lately, I'm into IoT — using Raspberry Pi, Arduino, etc. Oddly, nearly all of the programmers I've known instantly stopped programming when they were promoted or when they retired. Not me. 'Can't leave it alone.



From a viewgraph transparency (remember those?) that I drew for a talk in 1984,
i.e.,shortly before Judge Green dismembered the Bell System
[*no wonder it's yellowed*]