Kermit Oral History Panel
Jeffrey Altman, Bill Catchings, and Frank da Cruz

Moderated by:
Alex Bochannek

Filmed by:
Gardner Hendrie

Recorded: April 6, 2012
Watson Laboratory, Columbia University, New York

CHM Reference number: X6479.2012

Editor's Note: Extensive commentary and additional notes were supplied by Frank da Cruz after this interview took place. These instances are denoted by square brackets with the initials "FDC" followed by his commentary in italics.

Alex Bochannek: Welcome. Today is Friday, April 6, 2012. We are recording this oral history panel on the file transfer protocol, Kermit, for the Computer History Museum's archive. On the panel we have Frank da Cruz, Bill Catchings, and Jeffrey Altman. I'm Alex Bochannek, a curator at the Computer History Museum [CHM]. Also present is CHM Trustee Gardner Hendrie. Observing the conversation in the room are Bill Catchings' wife Susie and my daughter Cecily. Now, I would like to start with people just offering a brief biography, their full names, when and where they were born, how they came to the Kermit project and what they have done since. You can certainly include any sort of pivotal moments or influential people that you would like to mention. Why don't we start with Frank?

Frank da Cruz: Okay. I was born in 1944 in Washington D.C. That makes me pretty old. I walked into this building when I was in my twenties and now look. I was a country boy. I lived in rural Virginia. I grew up in the segregated south. The place where I lived isn't there anymore. It's all glass and steel now. Then I was an Army brat and I lived in Germany on Army bases. Then, I was in the Army myself for three years and I lived in Germany on Army bases again. Then [after six months in Washington DC as a musician] I came to Columbia. I was at Columbia from 1966 until last year, 2011, so that's what? 45 years? I started working here at what was called the Computer Center in 1974. Before that, I was a student and I worked full time and went to school full time. I actually paid my own way through Columbia. That's inconceivable now. What else is interesting? When I graduated from Columbia with a BS in Sociology, I had a little trouble finding a job and so I was a taxi driver. After that, a friend of mine whose mother worked here found a job for me in the engineering school. I was working in the engineering school just doing office work, but the professors took an interest in me. One of them, Lee (Leon J.) Lidofsky, said, "How would you like to write a computer program?" I said, "Okay." Actually, I had had some training in the Army (Basic Machine Operations and Wiring Course, US Army Communications Zone, Europe, UDPRO School, Orleans, France), so my first exposure to computer programming was in 1965. It wasn't really a computer. It was an IBM [International Business Machines, Inc.] 407, an accounting machine you program with wires but it was a kind of programming. You read the input and do things to it and print out the results. In the Columbia engineering school, in the department where I worked, they had a minicomputer, a SEL-810B that was about the size of this room with 16K of core memory. They had a PDP-8 that was about this big [tabletop size] whose only I/O [input/output] device was a 9-track magnetic tape drive. You had to program it from the switches. They were running experiments all of the time. They were getting nuclear cross sections of all different radioisotopes. They would put some sample behind a pile of lead bricks. We used to handle lead bricks all of the time or paraffin because they were both good shields. Then put a counter and register all of the emissions and collect all of the data on the PDP-8's tape drive. Then [we would] take the tape to the minicomputer and write big FORTAN [IBM's Mathematical Formula Translating System] programs to analyze it. I enjoyed it, I enjoyed programming. One of the professors I worked for, Lee Lidofsky asked me how I'd like to have a full time job programming. I said, "Okay," and it was at Mount Sinai Hospital in Manhattan, where he did some consulting. They had something called the Laboratory of Computer Science. I worked there for a while and I wrote some programs that were real interesting. One of them was a database on a magnetic tape about anal fissures with a program that would do reports from these magnetic tapes. It was cool, actually, because I treated them like DECtapes [Digital Equipment Corporation random-access tapes]. Remember, DECtapes, the little tapes that spin back and forth? These big nine-track magnetic tapes were not designed for that.

Bill Catchings: You wore them out?

da Cruz: Yes. The most interesting thing that I worked on was a program that had to do with cervical cancer. The problem was that the surgeons would treat cervical cancer by inserting radioactive needles into the tumor. They did this with their fingers. They were very exacting about the insertion to get it just exactly right. Then, they started to get disease in their fingers. The professor that I worked for had this brilliant idea; what if the doctor stuck the needles in real fast any old way just so long as they got into the tumor? Then, [they] took a stereo X-ray of the tumor, ran back to the laboratory—which was in another building—scanned the X-rays to figure out where the needles are in the tumor, calculate the dose delivered by each needle, and their overlapping doses and so forth. Then [they printed out] a list of times to pull out each needle to maximize the dosage to the tumor and minimize the dosage to the surrounding tissue. [They would then] run back to the operating room with the printout and follow the directions. This is on a PDP-11/20, which was also about the size of this room. We had an oscilloscope so you could look at the picture of the tumor. It was like an oscilloscope. It was a lot of fun and I liked it. Except, I didn't like the doctors much because they didn't treat people very well. Now, I was taking graduate computer science courses in the Columbia engineering school. [*FDC: I suppose I should mention that eventually I wound up with a Master of Science degree from Columbia in Electrical Engineering and Computer Science.*] One of my professors, Howard Eskin, [who was also the Systems manager at the Computer Center], offered me a job and so that's how I got here. At first I was a systems programmer for OS/360. Since I had a background with Digital Equipment computers, I was chosen to be the one who brings timesharing to Columbia. We started with the PDP-11. Then, we worked our way up to the DECSYSTEM-20 which is considered a mainframe. It was a huge computer that cost a million dollars and for four of them it cost $10,000 a month just for the electricity. We hired Bill and some other programmers. We were the systems group on the DEC-20. Everybody who ever worked on the DEC-20 will tell you it was just the most fun computer to work with. It was very popular in universities. All of the universities had them, the universities that you've heard of in connection with computing; CMU [Carnegie Mellon University], Stanford, MIT [Massachusetts Institute of Technology], and Rutgers and so forth. Although, some of them ran TOPS-10 [Timesharing/ Total Operating System] and not TOPS-20 [two different operating systems for the same architecture]. They formed a community so we got software from them that we used on our computers to do our job, to serve our users, and we would reciprocate by giving them software we wrote here. We would meet these people in person at DECUS [Digital Equipment Computer Users' Society] conventions. It was kind of like a club.

Bochannek: Let's talk about the user community aspect a little bit later. Let's switch to Bill. Can you introduce yourself and give us some background in how you came to the Kermit project? Frank just mentioned he had hired you?

Catchings: I'll start from before he hired me. I always go by Bill Catchings, but actually my full name is William Baird Catchings III which is probably why I go by Bill Catchings. I grew up in New Jersey. I was born in 1958. I came to Columbia University in 1976 as a student. I always knew that I wanted to study computers. I'm really not exactly sure why but that's what I always knew I wanted to study. It was just what was in my head. I came here to study computers. First introduced to, I think, it was the PDP-11. Somehow in those days, you could sneak into the lab and actually just go play with the computers. I worked as a student. They had positions for students working here helping other students with their programs. Then, when I graduated in 1980, I came to work with a small group—three of us got all hired at the same time to work here at the university. Crucially, I often say, Columbia University but this was not the computer science department. You called it the computer center. I mean I think back in those days it was CUCCA: Columbia University Center for Computing Activities.

da Cruz: It's gone through many names.

Catchings: When I was here, it was CUCCA.

da Cruz: Everybody called it the computer center.

Catchings: Yes. As Frank said, we were here to maintain the computers. I know I worked on a PDP-11 running RSTS [Resource Sharing Timesharing System] which was one of the stranger computer operating systems since it was all interpreted rather than compiled and it was all written in BASIC [Beginner's All-purpose Symbolic Instruction Code]; hard to believe a computer operating system written in BASIC. Then we got the first DEC-20 and each of the three of us got different projects to do. I think mine was to do a bulletin board program. The other two people—I don't know that they ever actually finished their projects but I finished mine first. I got to do the next project and that project was what turned out to be Kermit. We basically, at that point, just knew we needed to figure out a way for students—student accounts would disappear between semesters—to move their data from one semester to another.

Bochannek: Right, so we'll get into that in a second. When did you leave the project and what have you done since?

Catchings: I left twice. I kept trying to get out of here. I left first, I believe, in 1982-'83. I left for a year and worked on Wall Street at Lehman Brothers, which later died. It's also died multiple times. After it died the first time, they got bought out by Shearson/American Express. I came back to work for Columbia for another year, worked on the Kermit project as well as some other things. Since then, I have been working in software development and in computer magazines like PC Magazine and PC Week. I worked for years as a writer for those magazines. Then, I worked developing benchmarks like Winbench, Winstone, many of the main benchmarks in the PC [personal computer] industry were defined by our group, Ziff-Davis Benchmark Operation. For the last ten years, I've been one of the co-founders of my company, Principled Technologies, where we do what we call fact-based marketing. It's basically helping computer companies to show what's good or better about their products than their competitors and that's where I am today.

da Cruz: Just one interruption. There's a missing chair here. I've been here the whole time. Bill was here at the beginning. Jeff was here sort of towards the end. The middle is Joe Doupnik who is not here. There's a gap.

Bochannek: We hope to do an oral history interview with him.

Catchings: Can you just Photoshop him in or something?

Bochannek: Jeff.

Jeffrey Altman: My name is Jeffrey Eric Altman. I need to specify that at times because, going back to my days in elementary school, there's a famous comedian/actor named Jeffrey Altman, a producer named Jeffrey Altman and there are two very wealthy financial individuals here in New York City named Jeffrey Altman. I often frequently get their mail, their invitation to their White House Christmas dinners, things of that nature. One of the more hysterical confusions was in fifth grade, coming back from winter break, when my social studies teacher congratulated me and apologized for missing me on some TV movie that she apparently thought I had been starring in. My exposure to computers started from an extremely early age. My father was an electrical engineer, CS [computer science] faculty member first at Princeton University and then at Stony Brook University. We had modems in the house. We had printers in the house, teletype terminals. I think I was given my first computer somewhere around the age of 10 or 11. I worked through TI-99/4As [Texas Instruments], Timex Sinclair's, Apple Is, Apple IIs, original IBM PCs and so on all in the house. My first paid programming gig was at the age of 14 or 15 which, given that I was born in 1966, makes it very early for my generation. My exposure to Kermit came from when I attended Stony Brook. Stony Brook was distributing, at the time in 1984, what they called Stony Brook Kermit which was essentially Joe Doupnik's MS-DOS [Microsoft Disk Operating System] Kermit wrapped with custom scripts for dialing into the Stony Brook University communication system I think to attain access to their VMS [Virtual Memory System] DEC systems. I was an undergraduate student. I was trying to avoid being a computer programmer. I wanted very, very hard to be anything but that except that programming came very easy to me. I went off to study astrophysics instead and that's what I was attempting to do.

da Cruz: Like Joe Doupnik.

Altman: Like Joe. [I was] attempting to do that at Stony Brook. I also was not a very good student. I was much more interested in the social aspects of college campuses; very active in student government, student polity association at Stony Brook at that time, the Stony Brook concerts. I came to run one of the big night clubs on campus that were student run as well as some of the fall festivals and spring festivals. I always made time for the social activities. I needed to avoid staying in the computer labs at two o'clock in the morning waiting for one of the three terminals to become available. Stony Brook had installed a ROLM telephone system across the campus. I figured out a way of being able to use an IBM PC portable with a modem, the ROLMphone, Kermit to be able to gain access to the VMS systems. I could sit in my room, doing my homework, ordering Domino's pizza—not drinking beer yet because I didn't start drinking alcohol until after I left college—listening to my music and hanging out with my friends, while all of the other classmates were staying up all night waiting desperately for a terminal to become free to submit their homework on. I started off sending in bug reports to Joe and making script additions. By the time I graduated in '89, I was already in the process of hacking on code and trying to work on the OS/2 version. I joined the Kermit project full time in 1995 [1994] in order to take the OS/2 version and produce a Windows implementation which we'll go into more detail later. I stayed with the project until 2002, when Columbia had a financial fallout from the Internet bubble collapse. There was a significant downturn here in the city after the Towers came down. Since then, I've founded two companies; Secure Endpoints Inc. and Your File System Inc. I'm sort of a serial entrepreneur at this point, very heavily devoted to open source technologies. I'm continuing to expand upon the things I've brought to Kermit in terms of secure Internet transfer of information.

Bochannek: Before we go into the origin of Kermit, and Bill already started talking a little bit about that, maybe Frank, you can tell us a little bit about the significance of the building we're in. Where are we? How does this relate to the Kermit project?

da Cruz: We'll start with the building that you were waiting in front of. That building had been a fraternity house; that's 612 West 116th Street. As World War II approached, Wallace Eckert, who was a Columbia professor, was drafted by the government to run the Almanac Office of the Naval Observatory for the war and produce all of the almanacs that were used by all of the ships and airplanes and some of the ground transportation to find where they were and to plot their courses, based on his work at Columbia in automating scientific calculations. He was, in my estimation, an important computer pioneer because he did a lot of things before anybody else did them. IBM asked him towards the end of the war to come back to Columbia and found what amounted to a computer science laboratory, the very first computer science laboratory. IBM bought the fraternity building on 116th Street. It was empty because all of the fraternity boys were in the war. He gave Eckert carte blanche to hire whoever he wanted and to get whatever equipment struck his fancy. He hired some of the best physicists and mathematicians that were to be hired. He had all kinds of IBM punch card equipment, but some of it was special custom souped-up versions. He had two relay calculators that were faster and more powerful than anything else on earth. He had two of the four or five that were built and the other ones were used in the military for ballistic calculations. It's what is now IBM Watson Laboratory, which is now in Yorktown Heights. From its very first years, it gave computer science courses, which as far as I know were the first computer science courses, beginning in 1946 or 1947. By 1953, they had run out of space and they needed another building. They bought this building, which previously had been a women's residence for students and faculty of Barnard and Teachers College and Julliard, which was right down the street but has moved since. IBM occupied this building from 1953 to 1970. It had machine rooms, laboratories, and the other kind of machine room where you build machines: lathes, die cutting. Everything was in here. They had huge amounts of electricity coming in, giant ducts for cooling. They built some very significant computers here; among them was the NORC, the Naval Ordinance Research Calculator, in 1954 which at its time and for the next 10 years was the most powerful computer on earth. It was built one floor below us in the machine room in the back. They also built the SSEC [IBM Selective Sequence Electronic Calculator] which was probably one of the first computers, if not the first, that entered the public imagination because IBM put it in their headquarters building on Madison Avenue where I'm pretty sure it could be seen from the street. People would walk by and they'd see these giant blinking lights and spinning things. It was an incredible machine. They said, “Oh, we need to store stuff in some way so we can read it back.” They had these rolls of paper tape that weighed something like 2 tons each and they had 20 of them [*FDC: these are not necessarily precise numbers*]. All of this stuff spinning and bouncing and jumping up and down and blinking at the same time. People would see it and then you'd see cartoons in the newspapers of this giant computer and a little card would come out with the answer. They built that here. John Backus worked here. He was the chief programmer for the SSEC. I asked him if he thought that the SSEC was the first stored-program computer. He said, "Sort of." It mostly wasn't but it was possible to load a program into some registers, execute it in the registers, to modify them at the same time and mix data with the program.

Bochannek: At what point did IBM move out of this building? Or was it a joint IBM Columbia facility?

da Cruz: Well, in a sense it was joint IBM-Columbia because all the scientists were on the faculty. On the other hand, they were collecting patents for IBM like crazy. It was mixed. They stayed here until 1970 [and turned the building over to Columbia, along with the 116th Street building]. I don't really know why they left. I actually remember when they were here. I was here when they were here but I didn't know it. I came into this building just a few years later and it was still littered with plug-boards and little plug-board wires, all kinds of card-oriented paraphernalia, but over the years all that disappeared.

Bochannek: Okay. Let's get back to the start of Kermit. Bill started to talk about how he had the issue with users on the DEC-20 timesharing systems. Either of you just tell the story of how Kermit got started and how Kermit got its name. If you could try, what is Kermit? How can you define it?

da Cruz: We'll start with how it started. The DEC-20s were not only well-beloved by us, the programmers, but also by the students and the faculty. These are the first interactive, well aside from RSTS, which was kind of a toy, these were the first general-purpose multi-language do-everything big, fast computers that were available to students and faculty.

Catchings: No punch cards.

da Cruz: Yes. You didn't have to stand in line waiting for a card punch and then stand at another line waiting to feed your cards into the card reader and then wait until the next day to get the printout which was this thick because it was a dump because there was a bug in your program and you had 20 pounds of hexadecimal numbers [to dig through]. It was friendly. It was very easy to use. It was intuitive. It was helpful. It had a sense of humor. It was the first instance of e-mail. No one [at Columbia] ever used e-mail before. *[FDC: In fact, the very first e-mail software at Columbia was written for our PDP-11 RSTS system by Andy Koenig, but it was not widely used. On the DEC-20, everybody used e-mail.]* Pretty soon, it had a bulletin board that Bill wrote which was a big deal. In the ARPANET [Advanced Research Projects Agency Network], they had these things but we weren't on the ARPANET, partly my fault. In 1968, we made such a ruckus that the defense contractors didn't want to even come near Columbia. Columbia didn't get on the ARPANET until fifteen years later. The DEC-20 was very popular to the extent that we got our first DEC-20 in 1977 and by 1979 or '80, like Jeff was describing, there were huge lines waiting to get in the terminal room. There was even a large demonstration on campus for the administration to fund another DEC-20. We were always—every budget proposal, “Oh look at the usage figures, they're through the roof. It's so congested that blah, blah, blah.” The central administration would say, “Should I buy another one of these computers for a million dollars or should I buy whatever other thing?” and they always did the other thing.

Catchings: Which wasn't the football team.

da Cruz: Right. By this time, it had come to such a pass that it was so painful to use these computers that when you pressed a key it would take over a minute to get the echo back. That just increased the waiting time. They had to keep buildings open all night. We had terminal rooms and that was a whole other story just getting the space from different departments in schools to build terminal rooms in different buildings that were accessible to students. When we first got the DEC-20, we said IDs were free and anybody could get an ID, a student, faculty or staff and they would have it forever. A mere two years later, not being able to keep up with demand because we couldn't get funding for it, we had to find ways to control the access. It was an awful thing to do but we said, “Well, sorry, we're going to have to wipe your account. Instead of having a perpetual free personal ID, you get an ID that's associated with a particular class that you're taking.” At the end of the class, we wipe the ID. Then they said, “What do we do with our stuff?”

Especially if you're in engineering, you go through the years of school creating little tools that you bring with you and you use them and you keep building on them term after term. If you lose it all every term, what are you going to do? *[FDC: This was actually a step down from the punch-card days, because you could keep your card decks forever. Cards were free to everybody.]* That's where the demand for Kermit came from. Students needed a way to archive their work. We began by looking at how we could do this. Is there some device we could put out there for them to use? We thought actually DECtapes would be the perfect thing, but the DEC-20 didn't support them. Maybe nine-track tapes. We'll put nine-track tapes in a terminal room. Those were—

Catchings: They were finicky.

da Cruz: Yes, they're very finicky. They're broken more often than they work and it takes days or weeks to fix them. Obviously, they're very expensive. Plus, a nine-track tape drive is bigger than a refrigerator. To get your stuff off it, you'd have to have it dedicated to yourself for half an hour. It wasn't practical. We considered some other things also, but kind of like that. DEC had some product that was a remote terminal controller that was a large minicomputer with an eight-inch floppy disk. They actually gave us one, but it never came to anything. What we finally decided was, “Well, why can't they just use their terminals to get their files on and off the computer?” In those days, microcomputers were just coming on the market and they had floppy disks. They had a screen, a keyboard, and a serial port just like a terminal. All you had to do was write a program for the microcomputer that acted like a terminal so you could replace the terminal with the microcomputer. This program would also have the ability to transfer a file, when used in conjunction with another program that we would write for the DEC-20. Then you could have a way to send files back and forth over your terminal connection and put them on the floppy disk or take them off the floppy disk and put them on the DEC-20. Obviously, when you do that you want the files to be correct and there's a lot that goes into that. Then also figuring into the equation, we had not only DEC-20 mainframes, we had IBM mainframes which are completely different. There's nothing in common between them whatsoever. This one is uses ASCII [American Standard Code for Information Interchange] to represent text. The other one uses EBCDIC [Extended Binary Coded Decimal Interchange Code]. This one has a stream-oriented file system. That one has a block-oriented file system. This one uses asynchronous communication. That one uses synchronous. We had to design a protocol that took all of the differences into account. Also, that when a terminal is talking to a computer, it's not a transparent connection. When I'm sitting at a terminal and I say, “Oh, woops, I made a mistake, delete,” and then the computer says, “I have to go delete all of those characters and put spaces.” Or if you type on the DEC-20; for example, if you type Control-T, it would print a little report saying what's happening. Or if you type control-C, it would interrupt the current process, make it stop. Therefore, you couldn't put all conceivable byte patterns into whatever kind of messaging we were going to use for file transfer.

Bochannek: The question was about the choice of the [Intertec] Superbrain; why was it chosen? Then I'm also looking at the Kermit DEC-20 Superbrain file interchange document that both Bill and Frank wrote in '81. The follow-on question is; what was the level of expertise in the student population with these types of machines? What a floppy disk is had to be explained in this document.

da Cruz: I don't think so. You know how kids now can use any kind of contraption? Well, they could then too. These floppy disks weren't a mystery.

Bochannek: Okay. Just being conservative by having definitions.

Catchings: On the other hand, not everybody—now everybody has a whatever in your house. Most students wouldn't come from a home—they were nowhere. Microcomputers, at that point in time, the reason we picked a company you've never heard of, was because there were no companies you had heard of other than maybe Apple that existed then.

da Cruz: Apples were really scarce.

Catchings: Yes, you mentioned Apple I. I'm actually skeptical because Apple I's—there were not many of them.

Altman: We had an Apple I. In my home, our closet was a museum similar to what we have across the hall [*FDC: at this point all of my old equipment and artifacts were piled up in the hall outside the conference room, most of this was eventually discarded*]. We had an Apple I which had, I believe, 8K of memory in that machine. After about a year-and-a-half, my father bought an Apple II and then we had an Apple II plus. We always had one of the first Apple five megabyte hard drives, which cost about $6,000 at the time, because it was just too annoying—talking about tapes you would save your program off on a cassette tape. We would have to fiddle around with the tone control because you had to record at one tone setting and play back at a different one in order to get the Apple II to actually read the data back off of it.

da Cruz: That was the other thing we looked at. IBM had some personal computers before they had the PC and those had cassette tapes [IBM 5100 “SCAMP”]. We had some of those computers. They were just too obscure and expensive.

Catchings: They would have needed the real manual.

<*break in audio*>

da Cruz: We wanted something that was familiar and intuitive. The command language of CP/M [Control Program for Microcomputers] is really like a DEC user interface. It's like RT-11 [Real Time] or TOPS-10, which is kind of familiar to people who used the DEC-20 because the DEC-20 is part PDP-10—half of its software is from the PDP-10. It has things like PIP [Peripheral Interchange Program] and whatnot just like CP/M. Joel was the resident hobbyist and he knew all about these new microcomputers. We said, “Which one do you think we should get?” We looked at the other ones and some of them had detached keyboards or other separate pieces.

Catchings: Which is not good with students; pieces that can disappear need to be discouraged.

da Cruz: Exactly. This was a one solid chunk that weighed 65 pounds and you couldn't walk out with it. You have it now. It's a tank.

Bochannek: I'm sorry, what is Joel's full name?

da Cruz: Joel Rosenblatt. He's actually been here longer—he was here when I came and he's still here. There's a couple of people like that. Everybody else—I walk around the building and I don't know who the heck they are. [*FDC: At this point I went upstairs to look for Joel but he wasn’t in*.]

Bochannek: Was the decision made to buy microcomputers for the student pools first? Or was it at the same time as the file transfer issue became?

da Cruz: It was all together.

Catchings: Without the file transfer, it was more expensive than a terminal so there was no point.

Bochannek: Right. But it wasn't part of a microcomputer programming class or anything like that?

da Cruz: No. It was only for Kermit. That's all it was for.

Catchings: How many did we get originally? Two?

da Cruz: Just two.

Catchings: One to write the code on and one to put into use.

da Cruz: We eventually had two of them out in the field. It was enough. If they had wanted more we would have bought more.

Bochannek: Who were the primary users for this?

da Cruz: Students.

Bochannek: If there are only two, were there long lines? Did people have to wait?

da Cruz: Honestly, I don't know. We worked in this building but they were on campus which is a ways away.

Bochannek: Only two machines, you said.

Catchings: Yes, at least initially. I'm trying to think the biggest computer room had maybe 16 stations in it or something. For all of the—the DEC-20 supported 60 people at a time, maybe 100. When it was 100, is when it was really slow. The community of people who were current at that point using it at a time, a couple seemed to be enough at least initially.

da Cruz: Yes. This is 1981. You know what else happened in 1981, the IBM PC was announced. No sooner had we made Kermit for—well, backing up—so the first Kermit, like we were saying in the e-mail—First, I was meeting with my boss Howard Eskin and some of his doctoral students. For two years, we were talking about this problem. Then, finally, we decided that we should have microcomputers and do it that way. Then, it was left to me and my group to come up with how to do it. It was Bill and I~~.~~ The best way to characterize it was that I did the research, went out to the other institutions and talked to different people to find out what they were doing. There were some other places that were confronting the same problem. Also, searching the literature so I basically had a stack of literature and I had some software source code listings from different places.

Catchings: Searching the literature was a lot harder then.

da Cruz: Yes, right. I turned it over to Bill. Bill; he was a very energetic young man.

Catchings: It was a long time ago.

da Cruz: I wanted the thing to follow the layered model of data link layer and transport layer, not session layer, not network layer as well, but the layers that you need.

Catchings: Yes, we skipped from layer three to seven. I think there was some missing in there. [*FDC: There was no network layer because it was point-to-point protocol. There was no session layer because the whole concept of a session wasn't well understood and as far as I know still isn't.*]

da Cruz: Following that, it came up with a packet where the outermost parts are the datalink layer. Then, the next ones are the—

Catchings: Yes, they were layers.

da Cruz: You strip them off until you come to the data as you progress up the protocol stack. That way you get framing, error detection, and sequencing, so the packet receiver can tell if it was damaged and request retransmission. It could also tell if it got the wrong packet and take action to request retransmission of the packet that it needs.

Bochannek: You mentioned earlier the challenges dealing with the DEC-20 and the IBM mainframe side of things. One of the key features of the current protocol is the use of simple ASCII characters in the protocol stream. One of the references that I've seen and I believe that was part of the e-mail conversation we had before this oral history, is that TTY [UNIX terminal] FTP [File Transfer Protocol] that was done at Stanford Medical School by Bill Yeager but also DIALNET that was done at SAIL [Stanford Artificial Intelligence Lab] and then Utah Small FTP were three of the protocols that you had considered. Those are the three that you mentioned in the Byte magazine article. Do you recall specific influences there?

da Cruz: I might have the names mixed up. I know that the one that influenced me most was the one from the University of Utah. That was strictly lines of text. In fact, not just text, I think it was only digits and upper case letters. That's all they used. The code space was 36 symbols, and 256 symbols had to mapped to 36, which was pretty inefficient, obviously. Safe, but inefficient. The design that Bill came up with was more efficient. It used all of the printable characters of ASCII; that's 96 or 95, plus two control characters. With that you could encode any eight-bit bytes. If you're transferring an ASCII text file, it's almost 100 percent efficient. If you're transferring a binary file that has a uniform distribution of byte values about one third of it would go through as clear text and the rest would be doubled. In those days, I think it was more common to transfer text files. I made a point in the e-mail about the difference between our world and the BBS [bulletin board system] world. In the BBS world, PC-to-PC they were sending mostly binary files and they didn't concern themselves with any kind of conversions or encoding because they didn't need to. Whereas, we had to because we were going between unlike platforms, microcomputers, DEC mainframes and IBM mainframes and any two of them had to be able to interchange both text and binary data over connections that were not transparent to 8-bit bytes or even to control characters. [*FDC: To clarify, we did not study the other protocols in depth or look at the code at all, we mainly were just encouraged that similar things were being done at sites we respected, but we had a different set of requirements and constraints—such as having IBM mainframes to consider—so none of the other protocols or software would have been of use to us.*]

Bochannek: Let me interject one other question about the DEC-20 environment. You've commented on how well liked it was. The Kermit user interface is very much like a TOPS-20 EXEC style.

Catchings: Pretty much exactly like it.

Bochannek: Right, there's a clear—

Catchings: That would probably be my fault.

[*FDC: The TOPS-20 command parser was a revelation to all those who had suffered with unhelpful, cryptic text-mode user interfaces (that you couldn't use without a manual at your side), as well as to those who had to grovel through endless verbose text-mode menus, especially on slow serial connections. It was a combination of both, that penalized neither the novice user, nor the experienced user, incorporating two novel concepts: keyword completion (so you didn't have type every command word out in full) and "menu on demand" so you could get a list of possibilities at any point in a command if you did not know what to do, without losing what you had typed so far, but without having menus forced upon you. We have Dan Murphy to thank for all that. The post-DEC-20 world still hasn't caught up with it.*]

Bochannek: —connection there. Even in the Kermit book, the appendices looked very much like a DEC handbook where you have the powers of two tables, and so forth. Is that a direct reflection of that environment? Do you feel that you were really stepped into the DEC world and that's just really the influence?

Catchings: Yes.

da Cruz: Yes.

Catchings: That's what we did. That's what we loved. That was the thing. It was the computing milieu we were in and we really liked it. Our user community really liked it. The weird thing is the first version of Kermit as best I recall was not between—there was no microcomputer at all. The strange thing is we had two DEC-20s, these two big massive blocky stuff.

da Cruz: No, no, one DEC-20.

Catchings: We actually used one DEC-20 with two versions of Kermit to communicate across a null-modem cable that connected two of its serial ports. So we did it first on one machine talking back to itself, but this was a way to transfer even between mainframes. It was pretty strange because networks were—well they weren't. [*FDC: It's worth nothing that even in the earliest days, it was possible transfer files, as Bill suggests, between the DEC-20 and the IBM mainframe with Kermit and I'm pretty sure I remember doing just that.*]

da Cruz: Well, there was DECnet. [*FDC: DECnet was the proprietary Digital Equipment Corporation networking method that allowed most (but not all) DEC computers to communicate with each other*]

Bochannek: Right. That was actually the next question. Did you think of Kermit as a long-term solution to a problem? Was this a stopgap until networks became more affordable, more widely spread?

da Cruz: [*FDC: I should have said: "If you're asking if I foresaw the ubiquity of the Internet, the answer is no."*] I didn't think of it either way. I mean we were just having so much fun. First, we did it and it was received. Let me back up a little. 1981, we had just done the version between the Superbrain and the DEC-20. Then, almost immediately, a guy at DEC took the Superbrain version and modified it to run on the DEC VT-180 which was a—

Catchings: Bernie Eiben.

da Cruz: Yes, Bernie Eiben. It was a VT-100 terminal—

Catchings: With a floppy drive nailed to the top.

da Cruz: Exactly. They wrote it up in all of their literature. They used to have newspapers that they sent everybody. Then, everybody found out about Kermit and they started bugging us about it. They said, "Oh, that's so cool, but we have a VAX [a DEC 32-bit minicomputer, VAX stands for Virtual Address Extension]; do you have it for the VAX?" We said, "No, but you could write it yourself." We had written the specification for the protocol so they did that. Then, they sent it back to us and they said, "Look, we did it for the VAX." Then, somebody said, "Well, can you do it for Data General?" "Well, we don't have a Data General." Pretty soon, people were sending us different Kermit versions from all over the world, for all different kinds of computers. The CP/M version started out, as I recall, as a monolithic program.

Catchings: It would only run on the Superbrain. That was the time it was where you didn't have—it wasn't like the Unix version that will run on all of these different computers. No, it ran only on the Superbrain. Then, somebody had an OSI [Ohio Scientific Inc.]. Somebody had this oddball CP/M machine because I remember having to go over to his lab, one of the labs here and recode it to make it work and there was no manual and so you're just pushing bytes out and seeing if they're showing up on the particular port.

da Cruz: That blue box that's in the picture, it was right there across the hall until a few months ago. That was Bill's line analyzer he carried with him down to the laboratories and watched the bytes go by.

[FDC: An Atlantic Research Corporation Interview 30A Data Analyzer from 1980 or so, about the size of a suitcase.]

Catchings: You talk about the long-term stuff and I don't want to claim that we were so naïve that we didn't think about that. The point is, we defined this protocol and then it was a coding exercise. For me, a coding exercise was cool. You run it on this one and you run it on that one and you figure things out. We didn't have plans of how this was going to take over the world. We had the specific need. Then, there were a lot of needs in different departments inside the university. They had some level of priority. They had a VAX over in the chemistry department. There were different people. It seemed like all I was ever doing was running over to one lab or another trying to help them get their version of Kermit.

da Cruz: That's right. A little later, when you wrote the CP/M-86 version and then in the linguistics department there was the professor, Marvin Herzog, who wascompiling the Language and Culture Atlas of Ashkenazi Jewry (LCAAJ). **Catchings:** Who knew?

da Cruz: He was using Victor 9000 computers; do you remember this?

Catchings: Vaguely.

da Cruz: You did it. We went over and we spent time with him. These Victor 9000s were the only computers in those days that you could see Hebrew writing (which is used also for Yiddish) or Russian writing on the screen. In this country, anyway. I think they were from Sweden or some place.

Bochannek: I noticed that Jeff seemed kind of eager to jump in.

Catchings: I'm sorry. You were pretty involved at this time.

Altman: You're still predating me. At what point did Joe get involved with MS-DOS Kermit? My first interaction with it was in 1985.

da Cruz: Yes, I'll come to that. All of this stuff was happening. Then the IBM PC was announced. Columbia, in those days, was still pretty chummy with IBM. I think there were some years where Columbia was IBM's biggest customer. The day that we heard that PC was announced, the computer center business manager here on this floor ordered 20 PCs sight unseen. He gave several of them to some high-profile professors. The professors said, “What are we going to do with these? All of our data is on the mainframe. That Kermit thing, that Kermit thing, we need that Kermit thing for these PCs.” It came down from on high, “Get that Kermit thing out now.” It came down the food chain until it landed on Bill.

Catchings: I actually think I might have refused to do the IBM PC version. I'm not sure.

da Cruz: He basically did it in one sitting. He just put the monolithic Superbrain version into EMACS [Editing Macros, the text editor from Richard Stallman at MIT, now known as GNU EMACS] and said, “Change this to that, change this to that,” and then you gave it to Daphne, right?

[*FDC: Everything was written in assembly language in those days, so he was changing 8080 instructions into 8088 instructions. Here we are discussing the original program for the IBM PC, which was called PC Kermit, that eventually became the MS-DOS Kermit program that was licensed to IBM and AT&T other companies, published in books, etc. It was originally called PC Kermit because it was for the IBM PC, but when variations were produced for other, incompatible MS-DOS PCs, the name was changed to MS-DOS Kermit.*]

Catchings: Yes. Daphne Tzoar.

da Cruz: Then she was the public face of MS-DOS Kermit for the first year or two. It was a big deal because not even IBM had a way for their PC to communicate with their other products. They actually licensed Kermit from us to provide to their own customers. I think I sent you an IBM packaging.

Bochannek: Yes, you did.

da Cruz: Somewhere I have it written down. There's a book that three of these professors collaborated on where they were using their PC's at home and they were sending their chapters back and forth to the DEC-20 where they had a common space for the working copy of the book.

[FDC: It was "The Scientific Experience" by Herbert Goldstein, Robert Pollack, and Jonathan Gross. This marked a turning point in computing at Columbia University with desktop computers being used to do the real work, and the central computers used only for communication, sharing, and archiving, pretty much the opposite of what we had designed Kermit for, and yet Kermit was perfectly suited for it. Anyway, this book was the reason for the rapid development of MS-DOS Kermit for the IBM PC.]

Bochannek: Now, when the IBM PC's came in, it sounds like initially it was just yet another type of machine. It didn't seem from your description to be a real sort of cultural shift within the computing environment here at Columbia. Maybe the shift away from the DEC-20s to the UNIX systems was more that type of a shift. Did that effect Kermit development when the DEC-20s got shut down?

Catchings: Well, that was much later though.

da Cruz: Yes, that was 1988. We're still talking early eighties here.

Catchings: I mean the IBM PC the introduction of that made the big change that it was now legitimate. As you say, Superbrain suddenly went from being a leading producer to how long did they even last after that? Once, the big people like IBM entered and then Digital was doing their own thing with CP/M, I mean CP/M-86 [*FDC: Strictly speaking, the DEC Rainbow desktop computer covered all the bases: it could run CP/M-80, CP/M-86, MS-DOS, or no operating system at all, in which case it was a VT220 terminal*]. It was all of these big players suddenly entered. I think in that way it changed because now I don't know how many students would have had but suddenly there isn't just two Superbrains, or ten Superbrains or whatever it is. It's now PCs. We've gone from microcomputers to PCs. [*FDC*: It was significant that not only were the new PCs made by big companies such as IBM, DEC, and HP, but also they had 16-bit processors, rather than the 8-bit processors that limited the earlier CP/M machines to 64KB of memory.]

da Cruz: They have it in their dorm rooms.

Catchings: Yes.

da Cruz: It's like Jeff said, you didn't have to stand in line anymore. You could be in your dorm room. Not only could you have a terminal but you could ship files back and forth. Meanwhile, I don't remember the timing exactly if the IBM PC was the first MS-DOS PC? Because HP, DEC?

Catchings: It was the first one that mattered because we had—I'm trying to think who else was first, but they were the big ones because it was Seattle DOS and then they sold it.

da Cruz: Right. There was NEC [formerly Nippon Electric Company, Ltd.]. Victor was CP/M-86. We had to adapt MS-DOS Kermit to each different make and model of PC.

Catchings: Every one of them is different. That's the key thing, they're all different. [*FDC: After a few years all PCs became "IBM compatible" and the noncompatible ones fell into disuse; eventually IBM itself stopped making PCs. MS-DOS Kermit continued to support the odd ducks, more as a matter of pride than of demand.*]

da Cruz: Yes, they're incompatible. Since Columbia is this big chaotic place with no central—the center for computing activities supposedly was in charge of computing at Columbia. But Columbia is like a bunch of feudal kingdoms and nobody can tell anybody else what to do.

Catchings: These are cheap enough that anybody can buy—or starting to be that people can buy them, PCs.

da Cruz: Yes. Everybody wanted to have them in their departments. We would say, "You should get one of these kinds because then you'll be able to communicate with everybody else. But if you get something else you're on your own." They'd get something else. Then they say, "Now, you make it communicate because that's your job." Depending on how big a cheese they are, we would actually get orders to make it work. We did a lot of adaptations in house to different MS-DOS machines, different CP/M machines, and different CP/M-86 machines. The CP/M version sort of took on a life of its own. Somebody, I don't remember who, chopped it up into modules and said, "Here's the system independent part and here's the system dependent part with the I/O routines." All you have to do to port it is to just fill in these little routines and they were called overlays.

Catchings: Right, modular programming, what a development.

da Cruz: The object files output by the assembler were text files, lines of hexadecimal characters. There was a hex file for the main program which never changed. Then there was a separate hex file for the I/O routines for each computer and all you had to do was tell the loader to put the main one together with the one that you wanted and then you had an executable program for your computer. That was the CP/M way of distributing software. Pretty soon we had a huge list of CP/M computers that had Kermit because building these system-specific I/O overlays was like a cottage industry.

Bochannek: You mentioned the porting efforts of Kermit to different systems here at Columbia. You mentioned DEC did a bunch of versions. You mentioned the IBM deal. Maybe now is a good time to talk about licenses, the copyright issue and the funding of the project?

[*FDC: We got sidetracked and never returned to this question. It's a long story but I'll try tell it briefly. At first we wanted the software to be free to everybody. But then we found ourselves spending all our time making tapes instead of managing the systems and developing software. So we started the Kermit distribution business and hired 3 full timers and some part-time students. But then as demand for tapes dwindled their work turned more towards tech support which brought in no revenue. Not wanting to fire them, I had to find a way to generate income to pay their salaries. At this point our two most popular Kermit programs were MS-DOS Kermit (for DOS) and C-Kermit (for Unix, VMS, and about six other operating systems). A lot of companies were "monetizing" our software, making it an integral part of their products or services; i.e. making (or saving) money from our work without paying us back in any way. Furthermore, end users of the products or services of these companies would call us for tech support – in other words, we do the work and the companies get the money. So I wrote a license that I thought was eminently fair which said, in essence, that companies that wished to bundle MS-DOS Kermit or C-Kermit with a commercial product or service would have to supply each user with the published manual so that (a) we'd get some revenue and (b) we would not have such a severe tech support burden. This was fairly successful, we shipped truckloads of books directly to big companies like AT&T and Lucent, and even more were drop-shipped directly from Digital Press. But our biggest customers either disappeared or downsized in the late 1990s, and by that time we had released Kermit 95, which was purely commercial, so we dropped the book requirement and came out with uniform "mix and match" bulk licensing terms for Kermit 95, C-Kermit, and MS-DOS Kermit. Meanwhile, C-Kermit and MS-DOS Kermit remained, as* *always, free for "own use" or use within a company. This scheme worked quite well until the financial crash of 2008, and this plus the fact that we were forbidden from issuing a new release of Kermit 95, resulted in the ultimate cancelation of the Kermit Project.*]

da Cruz: That's later. Now, it was just—

Catchings: It was a free for all. Things like Mac; one totally different thing coming out in '84: the Mac. At the same time this was happening, DECUS, the Digital Equipment Corporation User Society—I remember going and speaking at a couple of those telling people about this. That was part of that thing where it was starting to get outside of the university much more.

da Cruz: Bill has a really good voice.

Catchings: Yes, I do voiceovers now. That was one of the big things. There was this whole thing happening inside the university with this machine, that machine making it work but then it started happening much more outside.

da Cruz: Through DECUS, everybody found out about it and we started getting people sending versions not only for VMS but for every PDP-11 operating system. There were PDP-8, PDP-12, and I don't know whatever DEC made somebody came up with a version. [*FDC: DECUS also distributed the many and varied Kermit programs for DEC operating systems on native media for each kind of computer.*] DEC publicized Kermit heavily. Ultimately, DEC was also the publisher of the Kermit books. Every time you got DEC literature, you saw Kermit. How did we name it? I'll tell my version and then Bill can tell his.

Catchings: We have to say ahead of time we have not compared notes and I don't believe our stories will be exactly the same. The truth lies somewhere in between, over top or around it.

da Cruz: In those days, when there was some programming project, there was always a naming theme. The popular naming themes, obviously, were Star Wars, Star Trek and CS Lewis. All of the programs in the world already had those names. At first, we didn't call it anything. It was the thing that we were doing. When we finally started to have to give it a name; when we deployed it, it had to have a name.

Catchings: You have to type in something to start the execution; what do you type in?

da Cruz: Exactly. As I recall, it was Bill and Jeff Damens and Bill Schilit and I standing in a room on the seventh floor. We were on the seventh floor—this is the sixth floor—standing around saying, "Well, I guess we've got to think of a name for it. What should we call it?" Everybody goes, "I don't know. How about Enterprise or—"

Catchings: Sulu.

da Cruz: Where I was standing was a calendar on the wall, a Muppets calendar. I said, "How about Kermit? It's short. You can say it and not be embarrassed and you can spell it and so forth." We said, "Okay, all right, fine we'll call it Kermit." It wasn't a big deal. It wasn't like "ta-da."

Catchings: Yes, if we thought it was going to be a big deal we probably would have named it something better.

da Cruz: We probably would in retrospect.

Catchings: Picking a copyrighted trademark name wouldn't have been probably our first choice.

da Cruz: Not just that, but also I think it contributed to the common perception that it was a toy, that it wasn't serious.

Catchings: The only things I would add is that I'm pretty sure we must have been standing in my office because that calendar was on my wall. It was either Bill Schilit's or mine; we shared an office then.

da Cruz: It was 715, right.

Catchings: I think so, yes. That sounds about right. In my memory, of course, I was the one that suggested it but that's just a slight interpretation difference. I also recall coming up with an acronym for it pretty much instantly, though you recall that as after the fact.

Altman: What was the acronym?

Catchings: The acronym was the KL10 Error-Free Reciprocal Micro Interface Transfer which comes trippingly off the tongue. The real problem is that there're no good words that start with K. In retrospect, I wish I had used the word “kernel” because at least that would have made some sense but the KL10 was the processor in DEC 20 and nobody knew or cared and it didn't make a lot of sense.

da Cruz: Right. Then later you came back when you were about to have a child?

Catchings: Yes, which was about a year later.

da Cruz: You were looking through the baby book.

Catchings: It said that Kermit is derived from a Celtic word for “free.” I have no clue if that's true or not but that's what the baby book said.

da Cruz: Right. We were adamant that this had to be free software and it seemed perfect. But really it was named after the Muppets.

Bochannek: At what point did you get in contact with the Henson people?

da Cruz: Well, Bill—again, it was his idea to write an article for Byte. I said, “Wow, do you think they'll print that?” He said, “Look at all of the crap they print,” so we did. It was a big deal. They were serious. There were page proofs and copyediting and sending the things back and forth. By this time, Bill was working at the bank. He had a Xerox something or other. Was it a Xerox workstation?

Catchings: Yes, I think so. It might have been a Star—I don't remember.

da Cruz: Something like that where you could do graphics and you could print them. He was doing the diagrams, I remember. Then it was kind of frustrating because they got the article how they wanted it, but then they wouldn't print it because it didn't fit with any—

Catchings: They had a theme every issue. They had something pretty on the cover by Tinney or whoever drew all of those pictures.

da Cruz: We had to wait for a year before they came up with a theme where they thought Kermit would fit. The theme for the issue was—

Catchings: Which made no sense.

da Cruz: —University computing.

Bochannek: It was called the “Computers in Education” issue in June '84.

da Cruz: Right. They said~~,~~ "Kermit, A File Transfer Protocol for Universities".

Catchings: Whatever.

da Cruz: They changed the title of the article. By doing that, they misrepresented—they made it seem like this thing for pointy-headed, egg-headed people or something.

Catchings: As opposed to fun guys who liked to code and name things after frogs.

da Cruz: Yes, right, exactly. It was good that we published it and more people found out about it that way, I guess. In the course of all the page proofs and stuff, their legal department said, “You should contact Henson to get permission to use the name.” Probably only because that we said that it was named after Kermit the Frog. If we didn't say that, we could have said that we named it after Kermit Roosevelt. Not that I would want to name anything after Kermit Roosevelt.

Catchings: Kermit Shaefer. There are other Kermits.

da Cruz: I wrote them and they wrote me back—this is not clear to me anymore. I don't remember if there was one letter or two letters or three letters. I think you have one letter. They wrote back a letter saying, “Yes, you can use the name in the article that you publish or you can use the name in things that you publish as long as you say Kermit is named after Kermit the Frog, star of this show.” They gave the exact wording. We always used that wording. Later on, when the software became very popular and there were newspaper articles about it, and then this is the part I don't remember. When I published the first Kermit book, did I contact Henson again or not? I can't find any record of it but the thing that I do remember is that when I got the book back from the publisher, I sent a copy to Jim Henson. I said, “Look at this, isn't this cool?” It's named after your frog and it says right here, “Named after.” I don't think Henson ever actually received the letter or the book. I think it went straight to the lawyer and the lawyer said, “Who said you can do that?” I said, “Well I have this letter.” That was the last I heard from them. [*FDC: I recently discovered the pertinent letter and sent a copy to the Computer History Museum.*]

Bochannek: How did the book deal with Digital Press come about?

da Cruz: Bernie Eiben, I think.

Catchings: Could be. After my time.

da Cruz: Yes, I asked you to write it with me, remember.

Catchings: Probably, but I was busy raising kids and moving to North Carolina or something.

da Cruz: Me, too. I got in a lot of trouble for that. I guess I had the idea. It was my idea to publish the book, but I didn't know how to do it. I was asking around and our main Kermit contact at DEC—because we had a lot of contacts at DEC because we were a huge customer.

Catchings: Yes, because we didn't just buy two DEC 20s. We kept buying them.

da Cruz: Yes. We ourselves bought four and there were two other ones at Columbia besides. That's six million dollars.

Catchings: In 1980s dollars.

da Cruz: Yes. We'd go up to Boston and get the helicopter ride to Maynard or Marlborough. One time, they put us up in this crazy hotel. I don't think you were with us. I think it was me and Howard. It was like a medieval castle. Everybody was wearing Elizabethan costumes. I said, "This is weird." They'd make us ride in limousines. It was creepy. Aside from that, being there was really cool. Visiting the Mill; what a cool place to work. [*FDC: it was a mid-19th-century textile mill powered by a water wheel, with all its original wood flooring, interior brick walls, etc*.]

Bochannek: It was through a contact at DEC that you got to do the book?

da Cruz: Well, Bernie Eiben came out of nowhere. He contacted me when CP/M Kermit came out and he said he wanted to adapt it to the VT-180. After that, Bernie and I worked together very closely for years and years and years. E-mail flying back and forth constantly. I only met him once in person at a DECUS. He spoke half German, half English.

Bochannek: The graphics in the book are actually interesting, too. They're very whimsical. I was wondering if you could tell the story of that briefly; how the illustrations came to be?

[*FDC:* The illustrations were by George Ulrich, a well-known illustrator (and sometimes also writer) of children's books.]

da Cruz: They went crazy with the production of this book. It has features that you wish it didn't have, but they wanted it to be special. For example, they have words in green. I think whenever they write "Kermit," they put it in green. I said, "That might be a little crossing the line, a little too froggish." They started the chapter on the verso instead of the recto. All kinds of crazy things. They were always, "Oh, this is the latest in Swiss design." They would fly us up just to have conferences about the book. Did you know that DEC had an airline? I was going to say an air force. [*FDC: I forgot to mention that they also changed the title of the book from "The Kermit File Transfer Protocol" to "Kermit, A File Transfer Protocol" for "design reasons".*]

Catchings: I believe at one time it was supposedly the largest private airline or whatever in the country.

da Cruz: Yes, we flew on a Digital Airplane.

Catchings: I flew out of Teterboro up to Logan one time.

Bochannek: The book was part of the license agreement, though, right?

da Cruz: Not that book. That book was still written in the days when everything is free, except the book. I think there was even wording in the book about, "This will be open and free forever." I was very naïve in those days.

Catchings: He was very young, very idealistic.

da Cruz: It never occurred to me that Columbia could be so mean.

Catchings: I think that might be in that little thing.

Bochannek: How many people were working at—we're at about 1986 now, roughly?

da Cruz: Yes.

Bochannek: How many people were actively working on Kermit in your department?

da Cruz: I guess Bill was gone by then.

Catchings: I left in May of '85. You still hired some of my relatives but I was gone.

da Cruz: Right. Four or five. It wasn't the Kermit group. It was the DEC-20 systems group but we were mainly working on Kermit.

Bochannek: Can you just give us the names of the people?

da Cruz: Bill Schilit, Jeff Damens. I don't know if they were all at the same time. Daphne Tzoar.

Catchings: Ken Rossman.

da Cruz: Ken never worked on Kermit.

Bochannek: When did Christine come in?

da Cruz: Eighty-six. By this time, it was such a big deal that people were—we said it was free. But how do you get it? You have to write to Columbia. We sent you a mag tape. We said, “You send us a blank mag tape and we'll put Kermit on it.”

Catchings: They'd send all different size mag tapes and my job devolved into making tapes. I like to code, not make tapes.

da Cruz: Me, too. I remember a time when we had a party and the whole DEC-20 systems group and half the people from the other group were over in the machine room making tapes because we had such a big backlog. We had to clear out 200 tapes. We had all of the tape drives going for 24 hours.

Catchings: As we said earlier, they were finicky.

da Cruz: Somebody said, “This is ridiculous.”

Bochannek: We're roughly in 1986. Kermit has become quite popular. There are many different implementations. The first edition of the book has been published. There's more and more staff busy copying tapes. [*FDC: We mailed tapes to most of the countries in the world. This is where I developed my “international postal addressing” skills, which are the topic of a fairly famous website, “Frank’s Compulsive Guide to Postal Addresses.”*]

da Cruz: We've reached the point now where we're going to have an actual Kermit project, instead of a bunch of people doing this, stealing time from their real job to do it. At first it was our real job. After a while we had other responsibilities.

Catchings: Install new DEC-20s.

da Cruz: Right. We had a user community of 6,000 people on the DEC-20s alone, always having to find some way to get more juice out of the computers. It's not relevant to Kermit especially, but we would make trips to Marlborough to talk to the engineers like Peter Hurley. We used to talk to him all of the time about things we could do to get more control over access because all of the resources were being swamped. They actually—there were a lot of things that went into TOPS-20 that came from us; either that we wrote the code for, or pushed the idea so hard that they accepted it. The management here was not crazy about seeing all of these system programmers packing tapes all day. We agreed that we would charge $100 to send a tape and we would use the money to fund a production staff. We hired Christine as the business manager and she hired, I forget, one or two production people. We had the whole operation in the back room on the seventh floor [*FDC: which still had its power and raised floor from the days when it was an IBM Watson Lab machine room*]. We had a VAX-750 and a VAX-730 and some micro VAX's and tape drives and all of the stuff we needed. We moved the operation off the DEC-20s and we brought it—it wasn't totally on the DEC-20. We also cut tapes on the IBM mainframes because we had a lot of customers who had IBM shops. We could make OS tapes or CMS [IBM's VM/CMS operating system: Virtual Machine Conversational Monitor System] tapes on the mainframes. We still made DEC-20 tapes. We could also make TOPS-10 tapes on the DEC-20. We could make ANSI-[American National Standards Institute] tapes on the DEC-20 or on the VAX [*FDC: we wrote a number of tape writing, reading, and conversion programs, allowing us, for example, to write IBM mainframe tapes on the VAX*]. They [the Kermit production and business staff] took orders. They kept the database. They did the customer relations and all like that. [*FDC: They also did first-level tech support, and passed along any cases they couldn't solve to the developers.*] It was really successful. Like I said in the e-mail, I thought that charging $100 per tape—

Catchings: Would make them go away.

da Cruz: Yes, right. Exactly. Then we could stop bothering with this so much. It was really nice that so many people wanted it but it was kind of a lot of work, also. [*FDC: To state it more clearly, I never thought anybody would pay $100 for us to make them a tape. I was very wrong. By August 2003 we had fulfilled 15307 tape orders, 92752 diskette orders, 5524 Kermit 95 shrinkwrap orders, and 28260 book orders. All this is mail order. After 2003 most of revenue came from Kermit 95 and C-Kermit bulk licenses.*]

Bochannek: There were four or five different tapes, too, depending on what exactly you wanted.

da Cruz: We were using 1600 BPI 9-track tapes which hold, I forget how much, 50 megabytes, something like that. At first, it all fit on one tape. As we got more and more Kermit programs coming in and more documentation, we had to go to a second tape and then a third tape and then a fourth tape and then a fifth tape. By the time we finished with tapes, we had five tapes. To this day the Kermit distribution is—the file naming conventions and the groupings were by tape. First, there was a tape for everything. All of the file names had to be not only unique, but when a Kermit version had multiple files, they had to start with a unique prefix so they would be grouped together. Then, when we went to multiple tapes—and also file names had to be no longer than a certain length because some operating systems could not read long file names. Furthermore, some tape labels couldn't have long file names. We had to conform to all of those restrictions. All that started tapering off around 1990, I guess. Of course, when the books came out, we started selling the books as well as the tapes. We also sold disks, diskettes, different kinds of weird media tape like TK-50 tape cartridges, whatever that we could make that people wanted. In 1985, I started working on C-Kermit. I was telling Bill about how when he wrote CP/M-86 Kermit that he wanted to make a new paradigm for writing Kermit software so that it would be modular and it could be easily adapted to different platforms but he didn't remember. I did the same thing with C-Kermit. I wanted to make it modular so you could plug-and-play the different pieces and have it on different platforms. You could even put different applications on top of it and use the low-level I/O. I don't think anybody did that, but they could have. Or to use the command processor for some other application, things like that, which, again is another DEC-20 command processor. I wrote it from scratch so I wasn't stealing code from anybody. [*FDC: Ironically, I managed a project in the mid-1980s where we implemented the DEC-20 command parser for UNIX, mainly for use in the MM (mail manager) program, but I wound up not using it in Kermit because it was too complicated.*] I don't even see what the point is in stealing code because you never can really understand somebody else's code. It's just easier to write it yourself because then you understand it. In my experience, there're very few people who have a talent for working with somebody else's code. Some people are very good at it; like Jeff, for example. There was a guy who used to work for me named Chris Ryland. He was a really interesting guy. You remember him?

Catchings: Yes, the Imagen printers.

da Cruz: He came from Harvard. He was the first person who came here that knew all about this world of the ARPANET and all of the cool stuff that was happening at all of these other universities. He's the one who introduced us to all of that. He could take any software that anybody wrote and kind of understand it completely in a very short time and then do whatever he wanted to with it. But I'm not very good at that.

Bochannek: C-Kermit was primarily targeting which platforms originally?

da Cruz: Unix and VMS.

Bochannek: Unix and VMS. There was an earlier Unix Kermit as well, correct?

da Cruz: Yes. The earlier Unix Kermit was—how Kermit works is there's one on the local computer and one on the remote computer. The one on the local computer has to be not just the file transfer program. It has to be a terminal emulator. It has to do all of the I/O. Whereas, the one on the far end only has to do file transfer and it doesn't have to do I/O at all because it just uses standard input and standard output. That's what the original Unix Kermit was. It was the far end and it basically just implemented the file transfer protocol. It was written by Bill, I think.

Catchings: I know I wrote some sort of Kermit in C.

da Cruz: Two guys from the computer science department and three other guys from here. It was just the minimal Unix command line program where you just say Kermit dash this dash that dash that, no interaction, nothing at all, just the bare minimum. C-Kermit was supposed to pick up where DEC-20 Kermit left off. For many years, we said the DEC-20 Kermit was the definitive Kermit program. The test of any other Kermit program was, “Did it interoperate with the DEC-20 Kermit?” But then DEC-20s disappeared. Unix was the next thing and to some extent VMS.

Bochannek: How did you guys feel about that change since the DEC-20s were so well loved?

da Cruz: Well, we were bummed out. To this day, every time I am doing something I try to do it the DEC-20 way that was so perfect and, “Oops, it's not the DEC-20 anymore.” You have to go through some annoying other way of doing it. Some people were very bitter about it, that DEC just cancelled the DEC-20 and tried to get everybody to swallow VMS.

Catchings: Which was totally different as far as?

da Cruz: Yes.

Catchings: They may have seen it as a similar, but it was totally different.

da Cruz: Yes. We weren't like that. We just ignored it. We went straight to Unix.

Bochannek: But on DEC hardware.

da Cruz: On DEC hardware, because all we had to do was take out the DEC-20 and put in a VAX 8650 or VAX 8700 and plug all of the same peripherals [and networks] into it [even DECnet].

Bochannek: Now, at this point, we tried to talk about that a little bit earlier and you said that maybe this is the better time frame to discuss the whole licensing aspect. You have to somehow fund the staff that is doing the tape copying and so forth. What was the agreement within the university? What other funding sources were there?

da Cruz: There was no funding source except for income that we generate from outside.

Bochannek: What about any sort of corporate funding?

da Cruz: The only corporate funding we ever had was—it wasn't funding per se—it was a C-Kermit corporate-wide license to Hewlett-Packard from 1994 until the end. It was $10,000 a year. It wasn't really enough to pay for even a small fraction of one person.

Bochannek: You mentioned earlier that maybe ideology didn't necessarily play a huge role but you also talked about how you just wanted to share the software because it was just what you guys did. You wrote the code. You shared it.

da Cruz: We were paid. We were paid to write software. No skin off our nose. Even better if other people take it and use it and help to improve it.

Catchings: I think in that era the thought that the university would make money off of—it's so different now where the universities are all patenting things and they've got half of this company. I think back in that era, I don't remember—how could a university generate—maybe I didn't think hard enough. At that time, it just seemed like an odd thing that a university would try to make money off code.

da Cruz: Oddly enough, Columbia did have an office that was in charge of such things, patenting things.

Catchings: I didn't know where they were.

da Cruz: Neither did I. I heard about them and every once in a while somebody would mention it to me; they'd say, “They want to talk to you.” I'd just hope they would forget and they did. I never saw them until the end. I'm not allowed to talk about that, but it wasn't pleasant.

Bochannek: With whom did the peaceful use clause originate? Was that you, Frank?

da Cruz: Me.

Catchings: That would be Frank.

Bochannek: Okay.

Catchings: No argument there.

Bochannek: Can you talk a little bit about that?

da Cruz: Nobody really complained to me about it. Then, I told you in the e-mail of the episode where a company turned it to their own advantage.

Bochannek: This is the Pentagon contract where Kermit was actually specified and a subcontractor was trying to use Kermit in the bid. I believe that's the story.

da Cruz: Right. No, Kermit wasn't specified in the bid. They just wanted something that did what Kermit did. Kermit was the obvious choice because it was free. Then, some company could come and—I don't know—package it or something. I'm not sure what would have happened. They would have had to come to us and we would have made some arrangement.

Bochannek: What about commercial data communication software packages that did include a Kermit file transfer option? There was a license agreement with you? Or did they just specify what it said in the book as the protocol?

da Cruz: Even before there was a book, we had the protocol specified online. Anybody could read the protocol and they could implement it. I don't know if that was a mistake or not. In a way, it made it more popular but in another way it hurt us because the third party implementations were almost without exception awful. [*FDC: To be more precise, there is a basic protocol specification, the minimum which MUST be implemented, and then there are a lot of performance enhancements that are optional. Third parties would implement only the bare minimum. Which, if implemented correctly, was exceptionally robust but also quite slow. Many people saw only these minimal implementations and concluded that Kermit protocol was unbearably slow, even though the Kermit Project implementations are as fast as or faster than anything else and at the same time more robust and "featureful."*]

Catchings: It was mostly a checkbox. They wanted to say supports Kermit.

da Cruz: Exactly. Since so many people used those, especially the BBS people who bought these packages—whatever they are called?

Altman: Procomm, Softronics.

Catchings: Telix.

da Cruz: Right. They would use the Kermit protocol in some places because it was the only one that worked but it was so slow or buggy that they just said, "This is horrible, why would anybody ever use this?"

Bochannek: We talked about more and more people using it and more and more different platforms. I believe in one of the *Kermit News*, I was reading about a collaboration between some of the local universities and how it sounded like you worked fairly closely together with some of them. Let me ask you about the perception of the user community of themselves; was there a sense of, for lack of a better term, a "Kermit brotherhood?" Did everybody sort of feel like they belonged to this group?

da Cruz: You mean end users or developers?

Bochannek: Either way.

Catchings: This could be where he gets a chance as somebody who was one of those Kermit groupies.

Bochannek: Yes, let's ask Jeff.

Altman: Yes, I think there are two perspectives. When I first was exposed to Kermit, I wasn't exposed to it as Columbia at all. Stony Brook had completely rebranded MS-DOS Kermit. If you wanted to get a version of Kermit that worked with the Stony Brook modem pool or on campus communications, there was a particular distribution that inevitably was years out of date, had very bad, poorly written scripts. From a direct end user perspective, you probably didn't—unless you were part of DECUS you probably weren't interacting with Columbia on most campuses that were using Kermit. You were interacting with the local campus technology group and help desk, which certainly hurt the Columbia effort in many regards because none of those users ever bought a book. [*FDC: By this point Kermit book sales were a major source of the revenue that supported the Kermit Project*]. None of those users ever bought software. The campuses weren't paying for licensing for the software that they were redistributing.

da Cruz: Of course, we didn't ask them to.

Altman: Which was all fine. From that perspective, I think there was a significant disconnect between the user community and the people who are developing the software. I think there was a huge transition between the early eighties and the late eighties from the perspective that in the early eighties everybody was interested in getting Kermit on to new platforms. There was a lot of direct communication with Columbia. The lack of networks between campuses meant that if you wanted it you had to pick up a phone or send a letter?

Catchings: Or send out an e-mail.

Altman: You could have sent e-mail through BITNET or what have you eventually. But a lot of the international communications were done through postal services. There was a broader connectivity between the end users or at least the system administrators and Kermit central.

da Cruz: Yes, you have to look at the book, The Matrix by John Quarterman, the crazy ways that people had of getting to some resource that they needed somewhere on this multiplicity of networks that had all kind of hokey little gateways like the one we had here at Columbia between BITNET and DECnet.

Catchings: Well, whatever our local university DECnet we called it.

da Cruz: CCNET. You could send mail from anywhere on BITNET to any of the DECnet network that we had with a bunch of different universities like CMU and Stevens Institue of Technology.

Catchings: Somebody had an ARPANET on theirs [*FDC: It was Carnegie-Melon*]. You had to have these weird e-mail addresses with colons in them, I think.

da Cruz: Right. Source routing in e-mail addresses. [*FDC: This is a whole topic in itself, covered pretty well in Quarterman's book.*]

Bochannek: It seems like even though Kermit is such a nice handy name for the tool, the protocol, the software—the community that I'm not sure other than a very small core of developers—people who did the work on other platforms or end users really thought of themselves as a community in this sense? From the Kermit digest and the Info Kermit mailing list, it seems like there's definitely some sense of that going on but maybe not in the large.

da Cruz: It's more developers than end users.

Catchings: Yes, and that would have been a community of 100 people on the order of as opposed to—I mean, there was thousands—I don't know if there were millions but there were thousands and thousands and thousands of people who were using it, but the developers was a much smaller—?

da Cruz: Somewhere between 100 and 500. I think in the first edition of the C-Kermit book, I tried to list everybody who contributed to just the C-Kermit code and it was three pages.

Altman: By '93, the OS/2 version had 50,000 downloads on what was the Internet at that point which was a ridiculous number for this program. It's a utility program that provides terminal emulator and a scripting language and a file transfer over networks. How many people could there be that actually would want terminal emulation in the age of AOL and CompuServe.

da Cruz: Well, it's really hard to wrap a young mind around the fact that up until about 1995, everybody who used computers was completely comfortable with text. Secretaries. I'm not denigrating anybody, but every single person who worked at Columbia or was a student or faculty or supporting staff who used computers, they dealt with command line processors. So how hard can it be? Then, suddenly, like a switch being flipped, it's like you can't even mention them without hoots of derision. People refused to deal with them. Microsoft goes out of its way to hide command.com so nobody can find it.

Altman: I think that's an important point; at Stony Brook, for example, from '86 on the entire mail system was on the VAX. If you wanted to interact on campus mail, you didn't use Kermit for file transfer. You used Kermit because Kermit was the terminal emulator that got you from your PC on your desktop to all of the campus computing services.

Bochannek: This is a good opportunity to get back to a question I asked in the very, very beginning and I don't think we answered which is the definition of what is Kermit? We talked about the protocol. We talked about a file transfer program. Now, we mentioned the terminal emulation. We haven't talked about the scripting language yet. I think you, Jeff, in the e-mail mentioned the Swiss Army knife.

Altman: Yes, well, from my perspective, it was not only a Swiss Army knife it was also a lab. By the time we were done with Kermit 95, obviously we had serial support. We had Named Pipe support. We had LAT support. We had raw NetBEUI [Network BIOS (Basic Input/Output System) Extended User Interface] implementation. All of the TCP-based protocols that you would use to connect, whether it be R [remote] shell, R [remote] login, Telnet, SSH [Secure Shell], FTP [File Transfer Protocol], a variety of raw SAFI [ph?] connections. We implemented our own—tried to get brought to the IETF [Internet Engineering Task Force] a Kermit daemon service. [*FDC: Note that Internet application-level protocols such as Telnet, FTP, and HTTP were also defined in secure versions, using Kerberos IV, Kerberos V, SSL/TLS, and other security methods and these were supported by C-Kermit and Kermit 95.]*

Bochannek: The IKSD [Internet Kermit Service daemon].

Altman: The IKSD provided a more secure and, in fact, a more efficient file transfer protocol than FTP. It's scriptable on both ends. [*FDC: IKSD is recognized as an Internet protocol by the IETF, specified in RFCs 2839 and 2840.*]

da Cruz: And more flexible and it does a hundred things that FTP doesn't do. People won't even use FTP any more because it's "too complicated". Now they only use HTTP [Hypertext Transfer Protocol] to transfer files, which doesn't do anything except send the raw bytes back and forth. [*FDC: Ditto for SSH-based protocols like SCP and SFTP—in all these cases there is no character-set or record-format conversion like Kermit does for text files, and very little, if anything, in the way of selection criteria, collision options, etc.*]

Altman: Well, a lot of that has to do with the fact that HTTP ports are the only ports that are open on it. Now, in fact, the Internet is in many regards—you can flag it down to one port number. Everything you want to do is tunneled over HTTP and then proxied on the back end.

da Cruz: It's really ironic. We started out with this huge diversity of computers. We had to really think hard about how to make them communicate effectively with each other by taking all of their characteristics into account—

Catchings: Characteristics slash quirks.

da Cruz: Yes. Arriving at what we call common intermediate representations for data. Formulating actual standards so that you have this well-defined representation for data. Then you have everything that's different. If it knows how to translate between its own formats and the common one, then anything can communicate with anything else. Now we just don't do it anymore. It's like, "Okay, I'm on a Mac and you're on a PC, so I'm going to send you this thing but I have to put it in this kind of archive but you don't even know about that kind of archive." If you can unpack the archive, all of the files are in Mac format instead of PC format. Somehow you have to find some other utility that knows how to decode the Mac files and put them in PC format. It's crazy. Kermit does all of that in the act of transferring the file, automatically.

Bochannek: The file attribute extension that wasn't in the original Kermit; is it correct that that came from Brian Nelson's PDP-11 implementation of Kermit with the different file format attributes?

da Cruz: Well, I designed it. I think he implemented it first.

Bochannek: That was one of the things I thought was interesting is in the original Kermit documentation, for example, in the Byte article there's the explicit point made that Kermit will not convert different file formats for you. Then the later versions you can negotiate file attributes and different formats based on what platform you're on.

da Cruz: Right.

[*FDC: Actually, Kermit always did convert text-file formats between unlike platforms; e.g. between record-oriented EBCDIC and stream-oriented ASCII. A common intermediate representation for text files was part of the protocol specification from Day One. What was meant by not converting different file formats applied to application-specific file formats like Word, Excel, DB2, etc.*]

Bochannek: That seemed to me as sort of a shift in how you thought of Kermit.

da Cruz: Well, I think we always thought of it as being an extensible protocol.

Catchings: Yes, of all of the things in terms of you, I, whoever did it, that we did right was to make it a negotiated protocol up front as opposed to most of the other things you talked about. FTP works the way FTP works. You decide binary whatever, that's it. Kermit was always a negotiated protocol which would work with the lowest common denominator. If the other guy knew how to do it you could make it work as well. To me, that's the big design choice we made that was right. [*FDC: Just to clarify by an example, the file sender might say at the very beginning, "I would like to compress the data during transfer" and the receiver can say "Yes, please do", or "No, please don't" or "What??? I never heard of that", thus ensuring that the receiver will understand and properly decode the encoding method used by the sender, and then in the general case, advanced options—e.g. for speed—will be used if both sides support them.*]

[*FDC: Interesting sidelight: Later versions of FTP protocol have incorporated feature negotiation.*]

da Cruz: Yes.

<coughing; question starts at 0:46:16>

Bochannek: Speaking of the extension and extensibility of Kermit, not the protocol level but at the application level, it seems like the work Joe Doupnik did for MS-DOS Kermit really advanced.

da Cruz: Yes, I want to spend some time on Joe Doupnik. He moved Kermit to a whole new level. We wrote MS-DOS Kermit here and we developed it up until version 2.27 as I recall. It was through a succession of people that I mentioned already. Then, I forget exactly what happened. Maybe it was Jeff Damens who was the main guy at that point and then he left.

Catchings: Yes, he left some time around there.

da Cruz: Probably we had let it lie for a while. All of a sudden, I get e-mail from this guy. He's a professor of astrophysics at Utah State University. He says he has these suggestions. He said, "I wrote this code and I made these changes." That began a relationship. The missing ten years between Bill and Jeff, I worked on a daily basis with Joe to design new protocol. We did Sliding Windows, for example, which is really complicated, more complicated for Kermit than it would be for something like X25 because for all of its layering Kermit isn't really a network. It's a point-to-point protocol that has to take a lot of quirky things into account about the transparency and so forth. It took us a couple of years to get it right. I was working on C-Kermit and he was working on MS-DOS Kermit. We'd send versions back and forth and we'd beat them to death and find something or think of some way to make it go faster. That was just one thing. Yes, he contributed a lot to the protocol. More than anything, he made MS-DOS Kermit into an amazing piece of software that fit into this little teeny computer. The thing that I always say to everybody, and I've already said to you lots of times, is he implemented an entire TCP/IP stack with Telnet protocol and put it into MS-DOS Kermit so it could be a Telnet client as well as a serial port program. He put in all of those other PC networking things, as well, that were important to him, Novell things mainly. It still fit on a floppy disk. Can you believe that? Nowadays, the average computer program is 30 megabytes.

Catchings: I was going to say a floppy disk in those days was 440 kb or something.

da Cruz: Well, a 3.5-inch rigid disk, so it was 1.1 megabyte. The whole thing fit on that [complete with documentation, code pages, key maps, scripts, and all sorts of other things]. It was enormously popular. It was really popular. Christine wrote the manual. We must have sold 100,000 of those manuals. [*[FDC: Interesting sidelight: Later versions of FTP protocol have incorporated feature negotiation.*]

Bochannek: Could you say Christine's full name, real quick.

da Cruz: Gianone. The book was also picked up by AT&T [American Telephone and Telegraph] and IBM. The book had the disk in the back.

Bochannek: Translated quite widely, too.

da Cruz: It was published in German and French, as well.

Bochannek: I believe there's a Japanese version.

da Cruz: And Japanese. That's not a translation. That's a rewriting—it was written from scratch but about the same program [*FDC: by Dr. Hirofumi Fujii of the Japan National Laboratory of High Energy Physics in Tokyo, and his co-author Fukuko Yuasa. "MS-Kermit Nyumon". It was about MS-DOS Kermit on the NEC PC9801. A truly remarkable computer; Hiro showed it to me once. It's just MS-DOS, which, face it, is pretty primitive. But its keyboard driver! It allowed Japanese Kanji and well as Katakana to be entered from a Roman (QWERTY) keyboard, learning from the typist and building up a knowledge base over time. This was in 1987.*]

Bochannek: Now, is it the MS-DOS Kermit where Kermit really acquired terminal emulation as a central feature? Were there other Kermits where it had?

da Cruz: No, from day one. CP/M Kermit was a VT-52 which is the easiest terminal on earth to emulate. [*FDC: PC Kermit, later to be MS-DOS Kermit, started out with VT52 emulation but had VT100 and Heath terminal emulation added, before Joe took over.*]

Altman: What I would say about MS-DOS Kermit is that it was the first Kermit application that had the VT-220 implementation and the VT-320 implementation. It was so fully feature functional that you could replace a real DEC terminal for VMS applications with it.

da Cruz: Which are very demanding. [*FDC: DEC's VMS operating system took full advantage of every feature of each DEC video terminal: VT52, VT100, VT220, VT320, etc. Very few VT100, 220, or 320 terminal emulators worked well enough to be used with VMS.I*

Altman: From my days of working on the emulator in Kermit 95 and OS/2 C-Kermit, the only way that you would be able to identify some of these applications is to figure out what the escape sequences really did in the real terminal. Whereas, to sit through and watch your program step by step, pipe a character through your program and transparently through the real terminal at the same time, watching exactly what was happening to the curser at each point, watching the state machine change.

Bochannek: That was MS-DOS Kermit, I believe. Also, did Tek [Tektronix graphics terminal] emulation and even some others?

da Cruz: It was the only Kermit program to have a graphics terminal emulator. Besides Tektronix 4010/4014 it did Data General Dasher emulation which is incredibly difficult; it was a color graphics terminal as well as a text terminal. Maybe I wrote one line of code in MS-DOS Kermit. I did a lot of the other stuff. For example, we put a distribution together so that there are code pages that you can load for Cyrillic and Hebrew and Arabic and so forth, Latin 1, Latin 2. Then, Kermit itself knew how to switch code pages. This was the first terminal emulator that people could use in most of the countries where they don't speak English or Dutch. Even in Holland, [although Dutch doesn't use accents], they have the ij digraph. Even they would be a little picky.

Bochannek: Now, the scripting language for Kermit we mentioned all of the different features that Kermit acquired over the years. Did that also originate primarily in DOS Kermit when it started to get control flow structures?

da Cruz: No. I have to say that's me. It really started in DEC-20 Kermit but it was very limited in DEC-20 Kermit. Kermit's command structure is a prompt and a command and then you get another prompt and then another command. You can take the commands and put them in a file. Then you can tell Kermit to execute the file. At some point—I don't know if it was Bill or me but we added macros to DEC-20 Kermit. We had defined a macro as a series of commands to execute. That was the extent of it. The macros didn't have parameters or variables. It was just like a shortcut. The first actual scripting language was in C-Kermit and it was all me. My real interest has always been programming languages. I programmed in lots and lots of programming languages since the 1960s and I appreciated the design features of different ones which is kind of ironic when you look at how the Kermit scripting language turned out because it's a real mess from a design point of view.

Bochannek: I was going to ask you about what inspirations you had for the syntax because it is a bit idiosyncratic at times.

da Cruz: We started out with command files so the scripting language was Kermit commands. Then the first thing you want to do that is add variables so you can pass parameters to a procedure. How can you distinguish a variable name from text in a command? Kermit's script language is executed on the spot, left-to-right so to speak; it's not compiled; we can't build complex structures and then unwind them at the end as compilers do because it's real time; each "word" is parsed and verified as it's encountered. It's a command language like the Unix shell. Like the Unix shell, it needed an escape character to introduce a variable. The first thing I said to myself was, "Well, gee, what if we want to have functions or different kinds of variables?" I said, "Okay, we need an escape character and then a subsequent character that says which kind of thing this is that we're introducing. Already it's getting ugly. The first variables we had were backslash percent one, backslash percent two which were like DOS variables with a backslash in front of them or like a Unix shell variables but with backslash percent instead of dollar sign. [*FDC: One conscious design decision was that, unlike the Unix shell, the Kermit scripting language would have only one escape character; you only need to look at some Unix shell scripts to see why this was a good idea. However, the choice of backslash was unfortunate because when C-Kermit was adapted to Windows (as Kermit 95), the escape character was the same as the Windows directory separator, which caused no end of heartache.*]

Bochannek: Then you even acquired S-expressions…?

[*FDC: We got sidetracked and I never answered this question. Yes, I added a mini-Lisp subsystem to C-Kermit and Kermit 95 in 2000-2001. Somebody dared me to do it. It's actually quite handy.*]

da Cruz: Then you want control structures, you want FOR loops and WHILE loops and SWITCH statements and GOTOs. GOTOs are not always bad. Figuring out how to implement them… everything had to be done in such a way that it didn't break everything else that had been done before because all throughout the years people were writing Kermit command files and then little by little they started calling them scripts because they had programming features. It just turns out that to add something to the language it's not always possible to do it in the nicest way, the most pleasing way because that would break other things. So we wind up with what you see today.

Catchings: I think that's a general thing, not just as a scripting language but of all parts of Kermit is we always cared about the legacy. It wasn't just well as of January 1 all of that stuff is trash and doesn't work anymore. I don't know for a fact that the first version of Kermit would operate with last version of Kermit but that certainly would have been our goal. I think we've always wanted to make sure of that.

da Cruz: I think it would.

Catchings: I would hope so, but I won't promise anybody.

da Cruz: The main problem is that the compilers changed out from underneath us. The code that we wrote in the seventies or eighties, you just can't compile it anymore. C is a language you love to hate because—the first time I saw C I said, "Wow, isn't that cool? It's so terse and concise."

Catchings: Self-documenting.

da Cruz: Yes. Right, it looks like PDP-11 assembly language.

Catchings: I wonder why.

da Cruz: Yes, I wonder why. Then, I learned how to program in C in the late seventies or early eighties but you have to know what you're doing. Well, the powers that be started saying, "People don't know what they're doing and so you can't make them have to know what they're doing. We have to protect them from themselves." C starts getting more and more and more bureaucratic until now it's almost impossible to deal with. You have to comply with this and comply with that. Oh, that's dangerous, you can't do that. I said, "But I know what I'm doing." Jeff has a clause somewhere where he's allocating an array without initializing it and reading from it to get a random-number generator seed. The compilers scream bloody murder when he tries to compile the code. "But I'm doing it on purpose! I want to do that!" [*FDC: The real problem with C is that it keeps changing and there are too many varieties of it. In earlier times, programming languages were rigidly defined and stable. I'll name PL/I and SNOBOL as two good examples. You could write programs in those languages and they would work on different kinds of computers and different operating system releases and across vast spans of time.*]

Bochannek: I actually want to get back to the user community real quick and just talk a little bit about how you ended up having Kermit users really all over the world. Going through the archives, I was fascinated by finding postcards from individuals sending you a message from Norway that they had just found a copy of Kermit and were excited about it, to letters from universities in China to really all over the world.

da Cruz: In the days of Chairman Mao.

Bochannek: Absolutely and you could preserve those things and add them to our archive. How did that come to be? It seems obvious how in the university environment or in certain hardware environments like the DEC-20 community how it would have become popular. But how did that spread so widely throughout the world?

da Cruz: One way is whenever we sent a tape to anybody, we kept a record of it. Then, when we started publishing newsletters in 1986 we sent them to everybody on the list. Maybe that had some affect.

Catchings: I wouldn't be surprised if Byte magazine for all of it may have had other consequences Byte was very widespread.

da Cruz: That's true. Did they have other language editions?

Catchings: I don't know. I would think so. In the 1980s, I know *PC Magazine* and *PC Week*, because I had my articles translated into—

da Cruz: I saw that.

Catchings: —all different kinds of languages that I had no—

da Cruz: Swedish.

Catchings: I would assume "Byte," in that hobbyist era especially, was everywhere. We would get—I remember, even when I was there up to '85, we'd get letters from places which we weren't allowed to get letters from.

da Cruz: Yes.

Catchings: We would get stuff from Cuba, or places that you weren't allowed to have contact from. They heard that somewhere.

da Cruz: Remember the time that the National Security Agency sent us a tape, so that we could put Kermit on it and send it back? It was wrapped in, like 20 feet of lead foil. Anyway.

Catchings: Did you send that one?

da Cruz: I think I threw it away.

Catchings: I was going to say—yes, we did have some differing opinions about who was allowed to get it and who was not.

da Cruz: I used to throw away any orders that we got from South Africa, for example.

Bochannek: Now, you mentioned the newsletter. One of the other things that I saw in the archive, was that it seems like producing, printing and mailing the newsletter was quite expensive.

da Cruz: At first, it wasn't so bad. It's just being in my brain here, but the first newsletter was probably eight or ten thousand copies, and the last newsletter was maybe 100 or 200 thousand. Plus, it was a lot more pages, and it was stapled, it had a slick cover and everything. The first one was more like a flyer; that last newsletter was the first one that lost money. All the other ones generated more income. You could tell, because they would tear off the order form from the newsletter and send it back with an order. Then we kept a record of that.

Bochannek: Now in the *Kermit News*, Volume 1, Number 1, original that you added to the archives, there is still a frog with a crown.

da Cruz: Right, and the one online doesn't have a frog. I just did that, because at the time, I didn't want Columbia to be sued, but now I don't care. I don't have access to the archive anymore, so I can't replace it.

Bochannek: Could Kermit have benefited from another icon, rather than the frog?

da Cruz: Kermit didn't really have an icon, except for Macintosh Kermit.

Catchings: Yes, that looks like a face on the Mac.

da Cruz: That is the Macintosh Kermit.

Catchings: I <*inaudible*> that icon. I'm just kidding, sorry.

Altman: There was the Windows 3.1 port that wasn't done at Columbia, had—

da Cruz: Oh, that one, right.

Altman: —an actual Kermit the Frog, that was being distributed on CompuServe, and I'm sure that was part—some of the angst that was going on, as Henson family was trying to sell Kermit branded educational software. [*FDC: That was one of few Kermit programs we never distributed.*]

Bochannek: Yes, but there really wasn't—even in, sort of general conversation on the mailings or anything—there really wasn't a symbol like a logo, type, a font or anything like that.

Altman: No, and even for OS/2-C Kermit and Kermit 95, we used the Columbia Logo.

da Cruz: The first version of Kermit 95, had a hokey little square icon that said, K-95. The second one has the Columbia crown, which we go to pains to point out; it's not because we're royalists, but because it's the crown of King Edward II, who founded Columbia University.

Bochannek: What's the origin of the coffee mug that you sent?

da Cruz: My college roommate's wife, Judith Bryant—we were all friends in college, and then they moved to Vermont and she became a potter. I asked her to make some cups, a Kermit cup.

Bochannek: Okay. Were they made in large quantities, or—?

da Cruz: No. She only made ten.

Bochannek: Okay. Now we have—

da Cruz: We gave them to a few people.

Catchings: I was going to say; I didn't get one, let's put it that way.

Bochannek: There's one in the Computer History Museum now; very excited about that.

Catchings: That must be mine.

da Cruz: Yes, right.

Bochannek: How would you describe, if you could even describe such a person, the typical Kermit user? Is there such a thing?

Altman: I think it changed over the decades. I mean, early on at Columbia, you would use Kermit to be able to transfer your file.

da Cruz: You had to do it, and so you had to learn how to do it, right?

Catchings: They were just students; they had no special—other than, maybe taking a computer class. I don't think it would have been all classes initially, it would have been computer classes.

da Cruz: But then, even by 1982 or so, when we had MS-DOS Kermit for the first time, I spent a lot of time with, for example, Herb Goldstein, who was a famous physicist, who was writing that book. He had to use Kermit to collaborate on the book that I was telling you about. He was totally computer shy. He's written the definitive book on earth on branch of mathematics [Newtonian physics]. He was at a department that had some of the first computers in the university, but he wouldn't have anything to do with them. Yet he dealt with Kermit. Everybody did.

[*FDC: Herbert Goldstein, 1922–2005. The book I was trying to think of is "Classical Mechanics."*]

Altman: Later on, you would have users, as we discussed, who were predominantly using it for terminal emulation. They had a PC or a Mac somewhere. Their goal was to connect to the mail system or the HR [human resources] system or whatever it was, that was running on the mainframe or a minicomputer or Unix system, in order to do their job. They weren't thinking of Kermit as anything. In fact, in many cases, they didn't even know they were running it. They had a script on their PC that said HR App, and that kicked off Kermit running a script that logged them in, connected them, brought up the right terminal, and put them into console mode, and from that point forward, they were working.

da Cruz: Right, they typed HR, and they see the screen of the application on there, and that's all they know.

Altman: In '91 or '92 at Stony Brook, we had macros that were built into WordPerfect on DOS, that you'd press I forget what combination of keys, and it would kick off a script to take the file, or save the file that you were working on, take the name, [and] call Kermit. Basically, you'd log into your mail system, upload the file, attach it to a mail message that you could then send. It went from being something the end users consciously knew about, to something that was embedded. It was really like an embedded technology. Certainly, by the time Kermit 95 came around, I would say there were two classes of users. There were System Administrators, who were using it as a terminal and they wanted to know all about the detail of some scripting languages and SSH, Kerberos, or SRP [Secure Remote Password] logins, and exactly what the inner details were, because they wanted things to be done in a particular way. We went to great extremes to add every possible knob that you could think of, to allow customization of the behaviors. They were predominantly using it, for the most part, for terminal emulation. You then had applications, Costco's Pharmacy, where there—it's a desktop cash register terminal that they're working off of, and it's just built into the system. You place an order, and the order would be logged to a file, transferred using Kermit, to over a dialup line, and the back end computers would do all the processing and send back a file that contained the results.

da Cruz: The operator had no idea that there—that Kermit even existed. The same thing with Best Buy. This might be after your time, but Best Buy—

Altman: Burger King was using it to deliver all their nightly results.

da Cruz: Yes, exactly, and franchise things like Burger King; it was a very common Kermit application. They had a VAX or something, and then all the cash registers would say, well we sold this many of these things, and send the report, then the VAX would order whatever supplies were running low. Best Buy wanted to buy a huge Kermit 95 license, but they didn't at first; they found some flaw whereby if the operator did a certain thing, they could get at the command prompts. By this time, Jeff didn't work here anymore, and I said, "You could pay Jeff to fix it." They couldn't. Somehow, they would buy a million [*FDC: "million" is hyperbole, but yes, tens of thousands of*] licenses, but they couldn't spend $1,000 to fix the thing that was bothering them. I was going to lose the sale. I said, "All right, here's what you can do. For the command screen, you set the text color to be the same as the background color, so if they come to the command prompt, they won't see anything." That was good enough for them.

Bochannek: How did these applications come about, like the experiment support on the ISS [International Space Station], or like, the one with the Brazilian elections that was featured on *Kermit News*?

da Cruz: Some of them came out of the blue. I knew nothing about the space station until it happened, and it was in the newspaper.

Bochannek: This was just a user somehow at—

da Cruz: NASA [National Aeronautics and Space Administration].

Bochannek: At NASA, who knew how to use it, and they just made it work?

da Cruz: Yes.

Bochannek: What about the Kermit implementation on the HP 48 SX Calculator?

[*FDC: We never came back to this question, but yes, we worked with Hewlett-Packard on the design of the HP-48 calculator, which uses Kermit protocol to communicate with the outside world. When the product shipped, they recommended MS-DOS Kermit and Macintosh Kermit for the other end of the connection; there was a flyer in the box, and also in the user manual. Incidentally HP also published a user manual for Kermit on HP-UX, their Unix operating system, which came with C-Kermit pre-installed.*]

da Cruz: Let's go back to the Brazilian election; that was more interesting, because we worked with them on that through the whole process, and it actually needed a customized version of MS-DOS Kermit that had a Portuguese user interface. It was just the minimal amount that they needed done, and we did it. It worked, it was just—it might have been the most significant thing we'd ever done. This was the biggest election that had ever been held in the history of the earth up to that time, 1994. Previous elections in Brazil had been disasters, both from the standpoint of corruption, and from the standpoint of how long it takes to get the results reported, or even boxes being lost on the way, or sinking in boats. Now, for the first time, they had electronic transmission of results on the same day, and it just turned out perfectly. I was pretty proud of it, and Joe was pretty proud of it, and that's—

<*inaudible background conversation*>

Bochannek: Yes, the question is, were you approached by the Brazilian government, or—?

da Cruz: No, the Brazilian government hired a consulting firm, Padrão iX Sistemas Abertos, Brasília, to design how to do it, and the guy who owned the consulting firm, Fernando Cabral, was a Kermit friend. He—that was how he would plan to do it from the beginning.

Bochannek: The reason I'm asking about this, the space station, the Brazilian election, or even the HP calculator, is because these are not the typical applications, necessarily. It's not the, loading a file from a mainframe onto your CP/M machine, or—

da Cruz: Once you can transfer files—

Bochannek: It's anything—right.

da Cruz: You can think of all kinds of things. My favorite example is a cardiac pacemaker. Modern pacemakers have computers in them and collect data, and furthermore, they can be adjusted by remote control. It's a little scary, right, but in this scenario, you have your pacemaker, you go to visit the doctor, the doctor has a special laptop from the pacemaker company, he [or she] points it at your chest, downloads the history from the pacemaker with Kermit, he analyzes it, or he has some other software that analyzes it, and then says, “We need to change some adjustment,” and types in commands, and it adjusts your pacemaker. Kermit 95 is on the laptop, and there's an embedded implementation of Kermit in the pacemaker, which, actually, I wrote myself.

Bochannek: How has the termination of the project affected those types of deeply embedded applications?

da Cruz: It doesn't affect them because people who bought embedded versions of Kermit; it's a one shot deal. It's a perpetual license, and an unlimited license that we negotiate. However we negotiate it, you could say, “Okay, if you pay this much, you can distribute 100,000 copies.” They said, “I want an unlimited license.” Then we talk back and forth and arrive at a figure. Once that's agreed, that's the end of it. They never ask for tech support. Maybe, they don't need it. It's a pretty solid program. It's very simple, it's very small and compact, and they just have to adapt it to their particular microprocessor. [*FDC: Embedded Kermit was purely commercial, written only to make money (and, hopefully, do a little bit of good in the world); there was no fixed fee schedule, every case was unique and negotiated according to the situation. Now of course, it’s free and Open Source like all Kermit software.*]

Bochannek: The typical data communication line in 1981, compared to today, has changed quite a bit. Things like when error correcting modems came in, I'm sure that had an effect on the Kermit protocol, using Kermit over packet switched networks has probably affected some of the protocol specifications as well. Can you talk about that a little bit; about the evolution?

da Cruz: I'll let Jeff talk about it, because Jeff is responsible for adapting Kermit to basically reliable error-free connections.

Altman: I think there were changes even before that. As speed improved, as reliability of the underlying connections improved—

da Cruz: All right, I'm sorry, wait, wait, let me back up. There's an earlier part.

Altman: Here's an earlier part.

da Cruz: The earlier part was that, not only were communication lines noisy and poor quality, but the devices that were communicating had serious limitations. The short packet of the original Kermit protocol, is because—

Catchings: People can only type so fast.

Altman: That's part of it.

da Cruz: The DEC-20, where the first Kermit program was written, had a little PDP-11 minicomputer as its communication front end, and the little PDP-11, as Bill says, was designed to accept input at about the rate that people can type. The funny story about this is that when the VT-100 terminal came out, DEC was all excited, and they came to show it to us. The guy hooked it up to a port on a DEC-20 and so we were using it, and he said, "And now, watch this." He puts it in smooth scrolling mode, and the DEC-20 crashes.

Catchings: X-on, X-on, X-on. [*FDC: To explain…The DEC-20 is sending text to the VT100 faster than its scrolling rate, so the VT100 is sending back a constant stream of XOFF and XON flow-control characters, faster than a person could type, so many that the PDP-11 crashes, taking the DEC-20 down with it.*]

da Cruz: That's a story in itself, but then later on, the same thing happened. We had a famous programmer who once worked here. I won't mention his name, but he was kind of a fanatical programmer. He'd stay up all night, programming, and eating cookies and smoking cigarettes and drinking huge coffees. One night, he dozed off. His head landed on the keyboard. The keyboard started auto-repeating, and the DEC-20 crashed. When it came back up, it crashed again. It came back up, it crashed again, and finally, they found him with his head on the keyboard, and the keyboard was auto-repeating. [For reasons like these] Kermit protocol could not exceed the capacity of the DEC-20 front end. That's why we made these short little packets that we were excoriated for over the next 20 years. [*FDC: The original maximum length was 94, which was just under the largest burst of serial-line input the DEC-20 could tolerate.*] As Jeff said, even before the work that he did, we made it possible to get very efficient file transfers on better connections, by having longer packets and also with sliding windows [which resulted in huge improvements when transferring files over connections with big delays; for example, through a satellite link]. Then, we noticed that, even with all that, that in Windows Kermit, the transfer rates could be really awful on an internet connection.

Altman: Oh, I don't remember. Go on, go on.

da Cruz: I think it was the Nagle algorithm?

Altman: Oh, right. Yes, so we end up at a completely different layer of problems. One is, as you're talking over TCP or a TCP based protocol, it's like Rlogin, or in Telnet, we have much more transparency in the protocol, which you want to be able to take advantage of because the fewer characters you need to encode, the more actual data you can get into the packets that we're sending. When we're sending Kermit over Internet Protocol, we are also able to make much broader, better use of sliding windows. We can get a great deal of data onto the wire at once. It turns out that the Microsoft implementation of TCP/IP, in fact all the implementations of TCP/IP at that point in time, were struggling with congestion control in the internet networks. The Nagle Algorithm was developed to permit you—or to improve the congestion control by not sending non-full packets.

da Cruz: TCP packets.

Altman: TCP packets. Essentially, what you would do, is until you sent a full packet, you would not allow a single byte to be sent until the acknowledgment came back from the other side. You would keep gathering the data that's being sent by the application in the TCP stack until the acknowledgment is received from the peer, then you would allow, if it was a full packet, you would send it immediately, otherwise, you would then send it when the acknowledgment is received. This, when you're dealing with something that's going over a terminal interface, we end up with very strange interactions between timing of the Kermit packet through the TTY interface on the peer, and network stack.

da Cruz: Basically, the Kermit acknowledgement packet would be a runt in the terminology of TCP. The data packet is huge, and the acknowledgment is just a little teeny packet, and you send it, and it doesn't go anywhere.

Catchings: It's waiting for its friends to come—

Altman: It's waiting for additional packets to arrive, for until either a time or the packet fillers, or the other peer sends the acknowledgment to the previous response. You could disable that, which we did. It took a long time for us to diagnose that that was in fact the interaction problem that we were seeing. It wasn't strictly Kermit that would have this problem, but any protocol that they—acknowledgment on top of TCP based, and not just relying strictly upon TCP to do congestion control and encapsulation of the data.

da Cruz: Yes, but what about streaming?

Altman: Then we decided at some point, to remove the need for acknowledgment of the packets, because of course—

da Cruz: If it's—

Altman: We knew we were on a reliable connection; if we were running on TCP on both ends, then we could trust the TCP to handle that for us. We just simply did encapsulation of the Kermit packets with the error checking, the peer would receive them, but we didn't wait for the acknowledgments to come back before sending off the next one in the sequence, essentially allowing for a sliding window of infinite size. The peers would be able to say, “Oh, something went wrong, you need to backtrack,” up to some number of packets that the client would hold onto.

Bochannek: This could also address situations where you may be on TCP on both ends, but maybe go through another non-TCP connection section.

<*overlapping conversation*>

da Cruz: No, that's the danger. You can't do streaming over a connection like that.

Bochannek: No, I thought he just said; it would actually fall back in that case.

Altman: We actually had these interesting algorithms that we had to come up with, because—what's the name of the gentleman that had the connection that was going TCP to X-25, to X-25, to TCP?

da Cruz: Was he in Brazil?

Altman: No, he was in Eastern Europe; to Russia?

da Cruz: Oh, they're the convention guys. Wait, no that was before your time. I can't remember.

Catchings: We'll call him Mr. X.

Altman: We can go through the support e-mail archives. Even in the days of Kermit 95, we—Kermit was being used over these very complex networks, where simply the fact that the initial hop was TCP based, did not mean that the destination—because the intermediary—we would often have a Kermit connection from a PC to a Unix box, and then there would be another C-Kermit running on that box to connect to the next hop that might be going over—it could be going over anything. It could be going over a serial connection, it could be going over another TCP connection, it could be going over some alternate protocol, and if any of those connections is requiring the use of control characters for flow control, then the Kermit connection would fail.

da Cruz: Under normal circumstances, the two Kermit partners could say, “I'm on TCP/IP.” “Oh, me, too. Oh, well let's, you know, we'll just skip all the formalities and blast the data through.” That usually works, except when you have one of these weird things in the middle.

Catchings: There's no way for them to know.

da Cruz: No way for the end points to know, right.

Altman: Right, the C-Kermit and the Kermit 95 applications basically allow you to turn this functionality off, so if you know that you disable streaming over TCP, if you know that you're going to be constructing these multi-hop connections.

da Cruz: This brings up the other thing that every software developer knows, —”Why doesn't your software do this? Oh, okay, we'll add that feature.” Then, “Why does your software have so many commands? Why don't you have, like, a stripped down simple version?” It repeats infinitely, and we do, too. We have C-Kermit which has everything, and then Kermit 95, and then we have G-Kermit, which is a teensy-weensy little program that only transfers files and doesn't have any commands. Then they say, but why don't you add this to G-Kermit? Well, it's not my job anymore.

Altman: Right, in some regards, the powers that be drove—in terms of Kermit 95, Kermit 95 was a transition, because it was really about trying to make the application easy to use for those used to GUI—the Graphical User Interfaces—and point to click. All the work that was put into the dialer application, which is really a separate app from Kermit, is about allowing a user to dive in as deep as they need to go into the Kermit functionality to configure a connection, a terminal, a set of operations that they want to be performed upon connection, and then having the dialer based upon that database generate a Kermit script on the fly, to actually drive the underlying—

da Cruz: In other words, it gives the commands that they should have given.

Altman: Without requiring that they know anything about the parsing—the scripting language, and then, but still, we always allowed the ability for users to go in and learn the dirty details of a scripting language to embed text commands into their dialer database entry configurations—

da Cruz: It's a constant push-pull, “Give us a GUI. But let us put commands in the GUI.” No, the GUI is too complicated.

Catchings: Could you please make it simpler in a more complex way?

Bochannek: Let me actually ask about some of this adding more features sort of things, and what I'm specifically curious about, and something that you mentioned earlier, Jeff, were a lot of the security features. How did they come to be? I think it's quite interesting how comprehensive the support was for, for example, secure FTP, those kinds of features.

Altman: That was all my hobby.

da Cruz: It came about as orders from above. We were told to “Kerberize” Kermit, remember?

Altman: Well, were we—

da Cruz: We were told to Kerberize it, or else Columbia would not use it.

Altman: Right, we had these interesting problems. Columbia hired me to produce an application, and throughout the entire time I was at Columbia, the biggest battle that we fought was getting Columbia to actually use it at Columbia. I only know about what you would describe as the turning point from where early on, every student used it. By the time I came on board, Columbia didn't want to have anything to do with it, but everybody else outside of Columbia loved us. We had this underlying problem, which was [that] the communication was no longer over serial, and it was over internet protocol, and the internet protocols required a different level of protection than what you have if you have a dedicated dialup line.

[*FDC: Clarification... Before Windows 95 came out, everybody at Columbia used MS-DOS Kermit or, if they had a Macintosh, Mac Kermit. Students, faculty, secretaries, administrative staff, deans, provosts, and the President. The Computer Center distributed these as the official supported communications software for the University for about 12 or 14 years. We had Columbia handouts, we taught classes, we published newsletter articles, we did tech support, everybody was happy. This continued with Kermit 95 for the first couple years, but then for no apparent reason somebody, I never knew who or why, decided to switch to some other package that was simply inadequate, and which soon proved itself to be. Then they switched to another. Finally when they decided to mandate secure connections, they turned back to us because the packages they had switched to did not support them. And Kermit 95 was the official desktop communications once again, for a year or two, until Columbia suddenly abandoned Kerberos for user security and switched to SSH, over our strong objections because Kerberos was far more secure and, more important, more manageable. Once Columbia switched to SSH, it abandoned Kermit 95 for good. All this was as mystifying to Jeff as it was to me.*]

da Cruz: First of all, we had clear text, Telnet and so forth.

Altman: Sure.

da Cruz: Then Columbia itself, the computer center management decided that that all the connections had to be secure, in 1997, I think. [*FDC: Columbia had just implemented MIT's Kerberos as its primary authentication and security method on the Unix timesharing hosts, and was about to require its use by all applications that wanted access, including to shell sessions via terminal emulators.*]

Altman: That's probably about the right time, end of '96, '97. I started becoming involved. The first problem was, we needed a Kerberos implementation on Windows, and MIT had one, but it wasn't very good, and did not really work very well, for what we were trying to build. I became involved with MIT in building their Windows port of Kerberos, but we were looking at Kermit, not just from the perspective of Columbia, but what other sites required. It wasn't just Telnet, it was Rlogin, you'd want to Kerberize Rlogin, you'd want to have, for FTP, when we finally implemented FTP, a GSS [Generic Security Services] based authentication for FTP. There were other authentication protocols. Telnet by itself didn't provide very strong or actually, by itself, didn't provide any privacy protection. The Kerberos authentication and the Telnet encryption option were both in themselves very, very weak. We were looking for alternatives. Doing Telnet over TLS [Transport Layer Security] was something that originated in the 3270 working group within the IETF, but basically died a very slow painful death there. The Kermit project is the entity that actually brought it to the implementation. It was for the Kermit project and the fact that we were doing Kerberos authentication over TLS that really brought out the need to have channel bindings between security layers on internet protocol. The first channel binding work in the IETF was actually the binding of Kerberos and open S/SL [Syntax/Semantic Language] implementation of TLS, while using the TLS handshake messages as the binding for GSS or for Kerberos authenticators. Today, you can't pass any protocols through the IETF as security layer that doesn't have channel bindings. That was driven by both a user need and by what I perceived was just a requirement. It just wasn't safe to use the software over unprotected connections. We wanted to make it as flexible as possible, so our implementation wasn't just Kerberos, even though that was what we used at Columbia. Stanford Secure Remote Password [SRP] was better for password based authentication as opposed to using a zero knowledge algorithm would be much better than using a central—if you didn't have centralized Kerberos, you could deploy that on your systems, and so we implemented that. We implemented X.509 certificate client support, we implemented server support, we have a complete GSS stack. The Kerberos libraries from MIT were ported and modified and became part of the base distribution, although Kermit was implemented to be layered, so you could actually deploy the Kermit 95, strip out the binaries that we shipped and got approved. We had all the export control issues as well, so we had specific binaries that were approved by BIS [Bureau of Industry and Security] for export, and so we weren't able to ship arbitrary things from MIT, but you could actually rip our code up and still use binaries that you built yourself.

Bochannek: Do you know specific sites where those features were really critical?

Altman: University of Illinois, the Urbana-Champaign campus. All their student registration system for example, was our student—you either went and you stood in queue to register for classes or you downloaded a Kermit 95 application, with a strip that logged you in securely using Kerberos over VTLS [Virginia Tech Library Systems] to an online registration system to-- pre, the web. The web wasn't secure enough for registration.

Bochannek: We touched on another area several times already, and that's the whole internationalization aspect and the character sets, and the terminal support with different character sets and so forth. I was just wondering if you could—either Frank or Jeff—talk a little bit about the connection between different character set support, Kermit support, all those character sets in files, in the terminal emulators and so forth, and how those things related to each other.

da Cruz: We started having contact by e-mail with people all over the world, as early as 1982, '83, with Bitnet. The way we were able to do that was on our DEC-20s, we made a mail gateway that pretended to be a card reader and punch, and then we connected to the IBM mainframe [*FDC: BITNET (“Because It’s There” Network) was an academic network of IBM mainframes based on Remote Job Entry protocols started, I believe, by Ira Fuchs of CCNY*]. Anyway, you couldn't send more than 80 characters. People needed to be able to transfer files in their own language, and they needed to be able to do terminal emulation, see the characters used by their own language. We became sensitive to the issues pretty quickly. Doing it in the terminal emulator was actually—it wasn't easy, but it was straightforward, because “all” you had to do was implement ISO [International Organization for Standardization] 2022. In fact, that is what the DEC VT-220 and higher terminals did, so that's what we did. Jeff did a prodigious amount of work on implementing that on Kermit 95, and Joe did the same in MS-DOS Kermit.

Bochannek: You, Frank, were—

da Cruz: I was doing C-Kermit and C-Kermit does not have a terminal emulator. It makes a terminal connection, but you're seeing it through your own console screen and that's where the terminal emulation occurs; the part where it places the cursor and formats the screen and so forth, but C-Kermit does convert between character sets on the terminal connection. Actually, I think I wrote the first VT-220 emulator for OS/2 C-Kermit, right? After that, Jeff did all the terminal emulators. For file transfer, you needed protocol. Again, we had to negotiate—we had to add features to the protocol where you would negotiate— it’s the problem where you need a common intermediate representation, because for example, in the early to mid 1980s, people were making PCs and selling them all over the world, and every company that made a PC to sell it in Argentina, or Korea, or Poland. They would say, “Oh, we've got to put Polish characters in here, or they won't buy it.” They just invent some character set and put it in the PC. IBM had a de facto standard in their PC code pages, and in their mainframe Country-Encoded Code Pages, which were, of course, different from the PC code pages. IBM had a huge repertoire of international and national code pages. Incidentally, you've probably been contacted by somebody trying to get to that stuff that I sent you [*FDC: Internal IBM documents that I donated to the Computer History Museum*]. They were different from everybody else, because DEC had other ones, HP had other ones, Apple was completely different.

Altman: And Microsoft.

da Cruz: Microsoft, right. Windows was different from DOS. This was before Unicode, so we said, “You have to break it down by the category of character set.” If it's west European languages, east European languages, languages that use Cyrillic, Japanese, Hebrew, Arabic, those were the main ones that we did.

We did the ones that people were screaming for, not necessarily just, “Wouldn't it be cool if we did Linear B.”

Catchings: Klingon.

Altman: I'm sure somebody did.

Somebody: Swahili...

da Cruz: Swahili is just Roman characters. We go to the standards. ISO has a whole series of standard character sets. For west European character sets, it's 8859-1, Latin Alphabet 1. Then, we make mappings between every western European private code page to Latin 1, and from Latin 1 to every western European private code page, and same thing for eastern European languages and Latin 2, same thing for Cyrillic, same thing for Arabic, same thing for Hebrew, for Japanese—Japanese being much harder, because I can't even begin to—well I could, but it would take a long time. It's a very complex writing system. Again, I think the computer representations of it are even more complex. Those days, we did it. We actually were able to convert between Japanese EUC [Extended Unix Code], Shift JIS [a PC code page based on Japanese Industrial Standards], and all these different representations that they had for mixing Roman letters, Katakana, and Kanji, and, for that matter, Cyrillic, because for some reason, they always include Cyrillic in all their character sets. I never really understood why.

Bochannek: You've become involved in the Unicode effort as well, during that time.

da Cruz: Tape time?

Bochannek: Actually, you can answer this one real quick, your involvement in Unicode standardization.

da Cruz: When we were doing Kermit 95, and Windows; I won't say the one good thing, but one of the few good things about Windows '95 and later is that it's based on Unicode, but Unicode did not include many of the characters that we needed to do terminal emulation. Mostly; not letters so much as box corners, and slanted things and whatnot, and so I wrote a series of proposals to have them added to Unicode, which were approved, along with something else I thought would be very useful; hex byte pictures, FF, or AB or B0, so that when you were displaying a character that was not in the font, or if you wanted to dump a character screen—like, a character screen in a Unicode based, say, terminal emulator, you could look at the hex values of the characters that were coming across, like the data analyzer that data communications technicians use. This proposal was turned down, but people keep asking for it. It keeps coming back up. Someday, they're going to put it in.

Altman: In terms of Unicode and the file transfer perspective, what Unicode provided to Kermit was a uniform intermediary character set that you could use to go between any of the groupings that—

da Cruz: I didn't come to the point where I explained that the user would actually have to say which character set family they were using, so that Kermit could set up the proper translations.

Altman: Right, and Kermit was limited, in that you couldn't go from a Latin 1 family to a Latin 2 family or a Latin 7—

da Cruz: Well, you sort of could.

Altman: If you were using characters were being translated, you would get—

da Cruz: It would actually turn them into ASCII.

Altman: Well yes. But Unicode solved that in that there was now a uniform intermediary that the user—

da Cruz: It didn't solve it, because if you're trying to transfer a Polish file to, like, a computer that only had CP437; even if Unicode was the intermediary, you'd still get the same problem.

Altman: You would get the problem on the far end. It would be a local translation issue, not—

da Cruz: Right. But the good thing about it was that from now on, the user wouldn't actually have to know the details of setting up the translation. The thing is, the people who were eager to do this, they didn't mind having to know the details. They craved just having the ability to do it.

Altman: Another thing that got added to C-Kermit around the same time as the Unicode work, was auto-detection of file types.

da Cruz: Right.

Altman: Was—

da Cruz: Yes, we're entering the days now where users don't want to know anything. They just want it to work.

Altman: Also, you're saying, “Here, translate this entire directory tree of 15,000 files across 5,000 directories,” and they're a mish-mosh of everything. You can't say, “Have a user pick out which one's a binary file and which one is a text file, and which one is a PDF [Portable Document Formant], which has a binary file that looks like a text file.” You have to provide mechanisms to automate this on a file by file basis. [*FDC: And you can’t base on the file name or “extension” either because users can name files anything they want.*]

da Cruz: Right, you actually have to look, do the kind of thing that Microsoft does, and look inside the file first, and try to figure out what's in it; it's an atrocity, because—but it works, it's really nice, but what if you screw up?

Altman: We had algorithms that Frank developed for detecting whether a file was a seven bit ASCII, an eight bit single-byte character set, or Unicode, etc, by scanning the first X number of bytes of a file, whether it would be PDF or not PDF, and we did a pretty decent job.

da Cruz: I haven't had any complaints yet.

<*overlapping conversation*>

da Cruz: Now maybe people don't—maybe people don't complain.

Bochannek: One of the questions I wanted to ask you: We mentioned communities earlier, we mentioned sort of the global reach of Kermit, and the conference in Moscow in '89, it was called the first international Kermit users conference.

da Cruz: First and last.

Bochannek: The only one, right? How did that happen, how did that come to be?

da Cruz: First in 1987, we were invited to Japan to a DECUS conference. DECUS was a big deal in those days. Thousands and thousands of people come, and they spent a lot of money; the airfare alone, they put us up in first class hotels. Word gets around. The next year, we said, “Well, gee, I wonder who's going to invite us to go someplace this year.” That time, we were invited to Swiss DECUS. When we go to one of these places, we also go around—when we went to Japan, we said, “As long as we're in the neighborhood,” [and] we went to Hong Kong and Macau, and some other places, and met people. Then we went to Swiss DECUS; we went to Germany and met Gisbert Selke who is the translator of the two books that were translated into German. He's a good friend, and also a very funny guy. Since I had lived in Germany—I'd lived more north—I wanted to see the old neighborhood but it was too far. We didn't have that much time. We met in Ludwigsburg, and then we went to Paris, where they had something called Club Kermit. The French people who like Kermit, they get together and enjoy it.

Catchings: It would have been Ker-mee.

da Cruz: Yes, right, I—

Bochannek: Was there another Kermit club anywhere else that you're familiar with, or was it a uniquely French thing?

da Cruz: I can't think of another one, but who knows? They actually had a club, and they had a newsletter and logo—did I send you that little frog toy?

Bochannek: You did.

da Cruz: Yes, the petit grenouille.

Bochannek: I believe so, I'm not sure.

da Cruz: It's a little frog with a cowboy hat. That was their emblem or something. They bought truckloads of them and gave them to everybody. They translated the MS-DOS Kermit book into French.

Bochannek: We're trying to get to the Moscow conference.

da Cruz: By that time, we had a reputation like a road show. Now around this time, we were working on the international character sets, and so the people who were most interested—the Japanese people were interested. The guy who wrote the MS-DOS Kermit book is also the one who worked with us on implementing the Japanese part of the software. And the Russians. The Russians have the same problem everybody else has. They want to have Cyrillic writing in their computers, but every different computer—this one comes from Bulgaria, that one comes from East Germany, this one comes from Poland—they use different representations. Plus, they have IBM mainframe clones that use EBCDIC versions of Cyrillic. They had those BESM [Bolshaya Elektronno-Schetnaya Mashina, “Large Electronically Computing Machine”] computers. Basically, I worked the protocol out, the details of the protocol, with a guy named Konstantin Vinogradov, Kostya, of the International Centre for Scientific and Technical Computing in Moscow, the organization that hosted the conference. Well, I don't know if I actually can say that he was the only one who worked on the details, but he was the one I worked with directly. They were on BITNET by that time, surreptitiously. You couldn't do it openly. We were able, with e-mail, to correspond and to work it out. Several of the people there spoke pretty good English. At one time I spoke Russian, but that was 20 years before. We could understand each other. I have to say, those people were really smart, and something that I came to appreciate when I went there, is that when they have to work with such restricted resources, they have to be extra smart to get anything done. Americans, they were so used to - saying, “We'll just buy it, spend more money, get more power, buy more memory, get a newer one. The Russians couldn't do that. They were able to do amazing things with their brains. Anyway, we worked out the whole protocol; it's all written down with their names and everything. There was the Level One protocol, which is the one where you announce—the one I just described, where you translate from your local encoding to one of the group of standard encodings, and then back to the other local encoding. A Level Two protocol that was never implemented was based on ISO 2022, where you could actually switch character sets in midstream, and transfer any mixture of scripts in a single file. If you had a document that's written in Russian, that's describing some Greek poetry and passages from the Bible in Hebrew, you could send that. This is only two years before Unicode. By the time Unicode came out, we didn't bother to implement it.

Bochannek: The conference was established based on that personal connection with the gentleman who helped with the Cyrillic character set work?

da Cruz: Yes. Well, his boss. The boss' name was Yuri Gornostaev. He wasn't the big boss, but he was the boss that we dealt with, and he was a really nice guy. They're all really nice and fun loving, I might add. They really know how to put on a show. They had a huge conference hall, and the equipment wasn't as fancy or anything; they just had an overhead projector, but they brought in so many people to attend the conference, from Bulgaria, East Germany, Cuba, all parts of the Soviet Union—you have the list. We spent several days because they all viewed the protocol, we gave demonstrations. We had hands on sessions and then we spent a whole day talking about the character set extension to the protocol. Some people made good suggestions that we incorporated. It was very lively and people would jump up and ask questions. We had this simultaneous translator.

Bochannek: Who all went from the project?

da Cruz: Christine and I.

Bochannek: Okay. The conference was how long?

da Cruz: The conference itself was three days, I think.

Bochannek: Then you had a chance to travel?

da Cruz: Yes. Then, they took us all over the place. They paid for everything except the airfare. They couldn't pay for that.

Bochannek: Right.

da Cruz: I actually got Columbia to pay for the airfare, on Aeroflot.

Bochannek: Now the Kermit project got terminated last year, 2011, and what is the current status of the particular versions of Kermit and the project in general?

da Cruz: When they decided to shut it down—they wanted to shut it down I think on January first. -They wanted me to wrap up the business. All they were concerned with—I had 1,000 Kermit 95 bulk licenses that have a service contract. Many of them were still active, in which they paid an annual fee for continued support, which I was becoming increasingly embarrassed about. We hadn't released a new version in some years, since Jeff left. I can't tell you how hard I tried, and how much effort I put into getting Columbia to let me spend the money that I earned, to pay Jeff to make a new version. You'll have to keep this part for 50 years, and then unseal it or something. They just took the money and spent it on whatever. They wouldn't let me spend the money that I worked 80 hours a week to come out that far ahead so that I could have money to invest. Anyway, they just wanted to wrap up all these contracts. This is when they turned—they turned me over to the enterprise whatever office and it was this long painful process of contacting all the licensees, saying, “It's the end of that.” They still are selling Kermit 95 with no support. The thing that distinguished Kermit 95 from everything else on the planet, I believe, was the support. Anybody could call us up or write us e-mail and we talked to them or answered their e-mail. No call processing system, no tickets, no nothing.

Altman: Pretty close to 24-7.

da Cruz: Yes.

Catchings: And no credit card.

da Cruz: Yes, right.

Bochannek: Now, there is not a GPL-ed [General Public License] version of Kermit as well, correct?

[*FDC: I think Alex was actually asking about the status of Kermit 95 and C-Kermit after the shutdown of the Kermit Project. These programs, and by implication all the others, were re-released with the "Modified 3-clause BSD License", which is an Open Source license. We did not use the GPL. Kermit 95 could not be released in usable form because it included a lot of proprietary code from other companies that licensed us to use it, but not to give it away or publish it. So we put all the source modules we could on the new Open Source ftp site in hopes that some day, somebody would start working on it again and, as of late October 2013, somebody is. Check the kermitproject.org website for news. The following was in response to Alex saying "GPL".*]

da Cruz: In 1990, Richard Stallman said that he was going to write open source versions of Kermit and put us out of business, because our software wasn't free.

Altman: I would have liked to see him try.

da Cruz: Well, I know.

Catchings: He and reality weren't always that closely related.

<*overlapping conversation*>

Catchings: He doesn't know me, it's okay.

da Cruz: He's all right.

Altman: He knows me.

Catchings: I didn't say he wasn't all right. An interesting character.

da Cruz: Anyway, we went back and forth, and what I finally agreed to do was to write a minimal, absolutely minimal, implementation of the Kermit protocol with a GPL, which I did. I said, "This is going to be an example of eternal software. It's not going to be upgraded and it won't need to be upgraded unless they totally change the C language so it won't compile anymore." It's completely ANSI Standard. It follows every rule, complies with everything, and it doesn't call any APIs [application programming interface] that are not universal, except, there's a couple of conditional things. Anyway, it turned out that way. Here we are, 12 years later, and there hasn't been a new release, and there hasn't needed to be. Anyway, I don't know if anybody ever used it.

Altman: Well, it's distributed with Debian and other Linux packages.

da Cruz: Now C-Kermit is distributed with Debian and with Red Hat.

Catchings: It's open source.

Bochannek: A related question to that; the longevity of Kermit. What do you think were the successful features of Kermit that contributed to its longevity?

Altman: It worked. More so than anything else, it worked.

da Cruz: I mean, we stood behind it. We helped people when they had trouble with it. We explained it, we documented the heck out of it. When I came to work here, I had the regular kind of System Administration jobs or programming jobs, and I enjoyed it, it was fun. Then Kermit was something different, because it was like a creation. We weren't just modifying some accounting program, we were creating something new. It was something that people needed and appreciated. It was fun, and then we got all this recognition. People from all over the world came to us for this software. Bill left too early, and Jeff came too late, but we got invited all over the world. It was a quite a ride, it was really a lot of fun. Until 1994, I had a real job here. First I was the manager of the academic timesharing systems. Well, first I was a systems programmer. Then I was a manager of the academic timesharing system. Then I had a job; something called the manager of systems integration and that meant that anybody who bought a computer at Columbia had to come to me to get my permission and I would try to talk them into buying something that would mesh with the other things that were on campus. I'd warn them about what the consequences would be if they bought something else. Then I became the first manager of the networks. I [my group] put in the first Ethernet, and later the first campus network to every room of every building over the new telephone wiring plant. The last thing the group did before I left it was wiring all of the dormitories for Ethernet. Then in 1994—now this is just about when Jeff was coming to develop Kermit 95—I went to the management and I said, "I want to just do Kermit. We'll earn enough money to pay for it." They said, "Okay."

Bochannek: A question that ties directly to that is major turning points on the project. That sounds like that was a major turning point. You were dedicated to just doing Kermit.

da Cruz: Yes, 1986 was a turning point because we hired a production staff. 1994 was a turning point because the Kermit budget became an actual department of the computer center with people dedicated to doing it and not stealing time from their real job.

Bochannek: The next question you can take to be a technical one and that's about choices you made that you regret as it pertains to the project. Is there anything that comes to mind? There in the Kermit book there's a checksum issue. There's a—?

Catchings: It would probably be my fault.

da Cruz: No, no, you fixed it.

Bochannek: How about in a larger sense, any particular choices that pertain to Kermit that you?

da Cruz: There's a lot of things in the protocol that I would fix if I could do it over. For example, the checksum is not identified in the packet. It's stateful. You say, "Oh I'm in this phase so the checksum must be the two bytes on the end." Well, the protocol would be so much simpler if the checksum were clearly marked in each packet.

Catchings: On the other hand, the fact that it evolved over time was I think one of its strong—that's the tradeoff. On the one hand, because we kept being backward compatible that meant that we had to live with the mistakes that happened in the past. I think that's the tradeoff.

da Cruz: Anyway, there was never a problem except just—this is interesting that you don't know about it. The ocean floats—at the University of Washington, they deploy these floats in places where there's hurricanes in the ocean; buoys that can submerge and come back up repeatedly. During the hurricane, they go up and down and up and down and they measure the temperature and salinity and current at different depths and then they pop back up and then they use Kermit to send the data to a satellite that sends it back to phone home. They use E-Kermit for that, the embedded version. They've been doing that for a long time. Now they're expanding the project and they're getting-- they're going someplace that's farther away from the satellite. I think they're going to Thailand or something. They discovered that it's just too slow. It's a 300 baud connection. It's extremely noisy. They needed one thing that E-Kermit didn't have which was true Sliding Windows. E-Kermit is unique in that it fakes that it's doing Sliding Windows but it isn't really doing them. The other side thinks that it's doing them but if there's an error it's just, "Sorry, game's over." He implemented true Sliding Windows. I had the outline sketched in there and he filled it in.

Hendrie: Who's the he?

da Cruz: His name John Dunlap. He's a physicist at University of Washington. I would have invited him but I lost touch. Anyway, for this new project he wanted to do serious testing in advance and he did something that I had always wanted to do myself and never had time. Suppose you had two computers with Kermit on them and you're transferring files between them. You want to put a box in between them which simulates all kinds of different delays and noise so it can buffer up the packets and send them out of order [corrupt them, drop them] and do all kind of crazy things. It would just completely stress test the protocol to its limit. Well, he made that box and he ran tests for weeks and weeks. He found that there was one place where Kermit would consistently fail if a packet was corrupted by noise in a certain way then the checksum would still agree if you calculate from the rest of the packet. That's on the first packet because the first packet always has the six-bit checks on them. The last thing I did in Kermit protocol before I left Columbia was to define a new protocol whereby the user could say to put 16-bit checksum on every packet including the first one. That's what it was. If the first packet was corrupted but appeared to be correct and it was corrupted in one of the negotiation parameters?

Catchings: Oh, right, right. Yes.

[*FDC: For example, a compression or quoting negotiation was mangled without being caught. The file transfer would proceed but the file receiver would misinterpret the coding used by the file sender and produce a corrupt file. The reason the first packet always has a six-bit checksum is that Kermit protocol was designed so it could be implemented even on computers that could do only 8-bit arithmetic.*]

da Cruz: Yes. Then I even figured out a way to make it automatic at least to the extent that you only had to do it on one end and the other end would recognize that it had been done.

Bochannek: Let me ask you a related question that I meant to ask earlier; in what scenarios would you not use Kermit? We mentioned this Army knife aspect of it earlier and we had so many stories about how flexible it is, but have you run into situations where that type of a problem that seemed to be solvable with Kermit just simply wasn't?

da Cruz: You're talking to somebody who uses Kermit all of the time, every day for everything.

Catchings: Right.

da Cruz: If I need to write a program to do something I write it as a Kermit script.

Altman: Yes, I'm still using Kermit and I haven't touched it in eight years. It's still my terminal emulator of choice on modern day Windows, even though the basic code dates back to 2003 for the revisions that we're using. The challenge of Kermit is that there's a huge selection of things which Kermit does where there's an alternative, where the alternative only does a small subset or a small overlapping intersection with the Kermit feature set. In many cases, the tools that you choose are based upon the things you're comfortable with. If you have a set of problems which Kermit is good at across the board, then you're going to use Kermit for many solutions based upon the fact that Kermit implements Telnet and Rlogin and terminal emulation and FTP and—

da Cruz: It's programmable.

Altman: —and has the scripting language and has other things. If all you need is a Linux terminal emulation over SSH, you're probably going to choose PuTTY over Kermit 95. What you're limiting yourself to is the fact that it's going to be a very low end terminal emulation which is good for most editing in EMACS but isn't going to be good enough to run a VMS application.

Catchings: But no one cares about VMS any more.

Altman: Right. But for the 99.9 percent of users, they don't care about that. It's always the edge cases. It's the edge cases that always made Kermit special is that we paid attention to the exacting details of terminal emulation. We paid attention to the fact that the Kermit character sets implementing ISO 2022 properly, handling the mapping of ISO character sets to Unicode in the terminal. Even going so far as to contract or to work with Everson to have a—

da Cruz: Oh, Michael Everson.

Altman: Michael Everson to produce a monospace Unicode font that actually included all of the character sets that Kermit supported so that we could actually display most of Unicode on Windows in a terminal session. There was nothing at that point in time.

da Cruz: 20 people in the world would care about that.

Altman: But we cared.

da Cruz: You could have a session of Mongolian terminal emulation on Kermit.

Altman: It's the fact that we handle all of the edge cases which really made Kermit very flexible and powerful and why people—and like I said earlier, it worked. In general, it worked. I mean are there bugs in the software? There are bugs in all of our software.

Catchings: I was going to say; it's software.

Altman: For the most part it works. Users they deploy it and it still runs.

da Cruz: Jeff did an amazing job because most applications will break when a new version of the operating system comes out. Since Jeff wrote Kermit 95 for Windows XP and it's okay in Vista. It's okay in Windows 7. As far as I know, it's okay in Window 8.

Altman: Well, we could try it.

Bochannek: That's actually a great dovetail; what accomplishment are you most proud of? There's actually two of the three—

da Cruz: Actually, I wanted to expand on the previous thing a bit. Like I said, I'm sitting in front of Kermit all day, every day and what I do now to keep myself busy and supplement my retirement income is a website. The website is based in Unix. I have a terminal emulation to the Unix system. The website is heavy on images so I'm always editing images on the PC in Photoshop. I'm transferring the images back and forth with Kermit file transfer protocol, in the same session where I edit the web pages. All of the scripts that maintain the website and get statistics about it reported are all written in Kermit. Every single thing that I do to maintain and develop and account for a pretty big comprehensive website is completely done in Kermit. I don't use any other tool except Photoshop. Kermit even knows something about images that—for example, if I have a bunch of images that are landscape and portrait Kermit can detect which ones are landscape and which ones are portrait. You can write a script to format a webpage that's full of images to mix landscape and portrait and have them all be the same height.

Bochannek: That's interesting. So accomplishments; what are you most proud of?

Hendrie: Can you describe this website?

da Cruz: I'm not telling.

Hendrie: You don't have to tell me the name of it. I don't want to go to it. But what it's for?

da Cruz: It's an Amazon Associate website.

Bochannek: An e-commerce site.

Hendrie: An e-commerce site for an Amazon Associate.

Altman: I'd like to go back and answer your question regarding the things that we wish we could have done or done differently. The biggest problem that I always felt was Kermit's weakness in moving forward was that we're still in, around 2000, a monolithic application. The scripting language, the terminal emulation, the file transfer, all of the underlying transports, the security modules, it was all one big package and you either got all of it or none of it. In the period from 98 through 2003, there was an ongoing transition to modularity, especially on Windows development of DCOM [Distributed Component Object Model]. The desire to be able to embed—it would have been nice to have been able to embed a terminal emulator into a web browser, relying on that feed for proxy services. We had to implement HTTP. We had to implement a variety of other things to allow us to go do transfers for the Web proxies. The scripting language given the way it's implemented when integrating into the terminal emulator are synchronous. They can't be running asynchronously. We can't run multiple instances of the scripting language engine at the same time. We weren't able to detach that and replace it with—I loved REXX [Restructured Extended Executor]. We had an OS/2 C-Kermit. We had a way to call out to REXX and call back into Kermit from REXX. We had a callback mechanism. On Windows we never really were able to get to a point where we could replace—today, you'd want to use PowerShell, for example. Our desire to maintain backward compatibility across all platforms placed constraints. Our need to ensure there was one code base across all platforms because we didn't have the staff to support independent builds. We needed it to be possible for us to implement, maintain one implementation of the protocol module one?

da Cruz: Kermit 95 about half of it is C-Kermit.

Altman: Kermit 95 provides a GUI layer on top of it. It provides a terminal emulator. It provides all of the I/O networking and security infrastructure and all of the plumbing for the operating system. It relies heavily upon C-Kermit for most of the logic. C-Kermit is implemented as a single threaded monolithic application. That provided constraints on what we were able to do.

da Cruz: Again, that's because it has to run on all the Stratuses and VMS and AOS/VS and Plan 9 from Outer Space, not that all those things still exist in large numbers or at all. I never wanted to release a version of Kermit that could not be built on all of the platforms where it was built before. Even if I can't prove that it can be, I never wanted to deliberately say, “Sorry, we're cutting you guys off.” Because it turned out years after I thought AOS/VS had disappeared from the face of the earth I learned that the US

Forest Service was completely dependent on it. They couldn't have lived without my support of AOS/VS C-Kermit.

Bochannek: What was the most fun about the project? What was the most enjoyable?

Altman: For me it was the lab. It was an educational lab. Anything I wanted to learn I could learn by implementing in Kermit and somebody would make use of it for something. Seriously, that is not an exaggeration. The Named Pipe support that we implemented, that I implemented in OS/2 C-Kermit and the NetBEUI support which ended up being in one of the OS/2 journals.

~~Catchings:~~ da Cruz: It's in the museum (the OS/2 journal). I saved all of that stuff.

Altman: I saved a lot of it much more than my wife was happy with me saving. All of that work was done not because there was anybody asking for it. It was because I was trying to learn as much as I could about OS/2 and file systems. I implemented as many of the techniques and things as I could in Kermit to solve a problem. Once it was there in the toolkit you could make use of it. In the company I was working at immediately prior to coming to Columbia to work on Kermit 95 fulltime, having built the Named Pipe support as something to do we found the problem that allowed us to communicate information from Raleigh to New York over the file system network that was built between the two pipes. We didn't have public Internet. They had a private network for the file system that was dedicated for that purpose. We were able to do name pipe communication to transfer information across that.

Bochannek: How about you, Bill? What was the most fun?

Catchings: For me, it's always easy. The most fun was coding. To actually get to code stuff that would be used. Any people who have done software development know that sometimes your best code never gets used by anybody but you. To actually be able to write something that really mattered and did something was very cool. I mean you asked earlier about what I'm most proud of or whatever and the thing I really liked about Kermit was that I think it somehow managed to meld together two ideas that almost make no sense together. On the one hand, this was something that was freely distributed but was treated as a real product. People cared. It was supported. It was a real pride and ownership. On the other hand, and this is probably its downfall too, was it was not meant to be a for-profit thing. It was not meant to make a ton of money. On the other hand, that meant it's continued to work even 30 years later because it wasn't like well, it's not making enough revenue it's cancelled. That somehow it's a real product but it was something we were real proud of. I was only there for a while but I was really proud to have been a part of it. I think that sums it up for me.

Bochannek: What about you Frank? Is there anything either of you want to add in closing, I think?

da Cruz: I'm proud of Kermit. I'm proud that it got all over the world and people knew about it. In fact, in some ways that had a different side to it in that; for example, big cheeses from Columbia would go to high-level conferences and they'd come back and they would say, "What is this Kermit everybody is talking about?" "You're from Columbia, that's where Kermit is from." It kind of made them angry because they didn't even know what it was [and everybody was asking them about it as if it was the only thing they knew about Columbia]. I think in the end the thing that I'm most proud of is that I was able to make—help a lot of people get real stuff done and provided jobs for—I haven't calculated it, but I think it's something like fifty [sixty] person-years of fulltime work for people in a non-oppressive environment. We had a lot of fun. We used our creativity. We wrote the code. Nobody was getting bossed or yelled at. This is the room where the meetings were held. I don't think you guys were ever in this room in a meeting about Kermit?

Catchings: I don't think so.

da Cruz: Right. We never had not one single meeting. We didn't need them. We all worked in the same place. If you want to talk to somebody you walk down the hall to their office.

Altman: You didn't walk down the hall and talk to me. I was never here. I think a simple anecdote one of my criteria for taking the job was Columbia was going to buy me a best of line laptop at the time which was, I think, a ThinkPad 701. I used it to work from wherever, even from South Africa, when I was traveling around South Africa for three weeks. I would literally stop at bars or inns and find somebody that would let me plug into their phone network to dial up through IBM's global net to come back to Columbia to exchange mail and upload code and download patch changes for the day. Answer tech support e-mails. Do builds and upload them. That was a ridiculous phone bill metric.

da Cruz: Yes, we built a tool that was not only used by all kinds of people for all kinds of stuff including kind of boring stuff like I can't tell you how many cash registers that are just PC's running Kermit. We built a tool that was really handy for us to use ourselves. We put a lot of stuff into it just for us because we needed to have that thing in it. I'm glad we did because now I use it all day, every day and I use those things.

Altman: I think, Frank, it would be worthwhile talking about the writings that you did after the Code Red scare regarding using Kermit in terminal emulation as an access way to e-mail.

da Cruz: It's funny. Yes, Code Red was 2000, right?

Altman: It was 2000.

da Cruz: Everything I do all day in front of a computer is exactly the same way I did it in 1985. I use Kermit. I have terminal access to a bigger computer some place where I actually do my real work on a Unix server, where I do my text editing and e-mail and everything. When people walk into my office and they see me doing that, they say, "What is that? What's that blue screen with the little white squiggles on it? Where's all of the spinning things and things popping up?" Around the year 2000 everybody was going into a tizzy because there was another virus every week that would knock the whole network off and wreck all of the PCs. It infected everybody except me because even though I had Windows on my desktop the only thing I used it for was Kermit and Netscape. My e-mail was a text-based e-mail program on the host. When somebody sent me a virus, I *saw* the virus. It didn't execute on me. Whose idea was that I could send you e-mail that contains a program that will execute on your computer? This was MIME [Multipurpose Internet Mail Extensions]. If you go back to the archives of the MIME discussions, you'll see me in there making these rants. Basically all of my arguments, I thought, made sense. The end of the discussion was, "Well, this is really just a pro forma discussion. It's already decided." Big companies, they want to have all of these cute things that happen by magic and astound the users. That's what they're going to get and that's what they got. To this day I mean every single—updates, updates, new version, quick update, don't turn off your computer. When I first came to work here we were running OS/360 21-point-something on our mammoth IBM 360/91 [whose control console is in the Computer History Museum collection]. We had been running it for 10 years and we were thinking about upgrading to 21.8. When we finally got around to it, it was completely ready and debugged. Then we ran that for another ten years before the 360 went out. When we had the DEC-20, we had source code for everything. They would send patches. The patches were source code. It was DDT [dynamic debugging technique]. We'd read them and we'd look at the source code. We'd say, "Does this make sense?" We wouldn't just install things on our computer without knowing what they were. Now we're running—Windows XP was 80 megabytes of kernel or something, some crazy thing like that. Nobody even knows what's in it. Nobody.

Altman: Sure you do. The source code is posted on the Web.

da Cruz: Is it?

Altman: It was stolen, the 2000 code; it's up there.

da Cruz: All of it?

Altman: Large portions of it.

da Cruz: Yes, but there're so many things that nobody even knows what they all are; even people who work at Microsoft. There can't be any one person who knows what it all is, right. It's too much.

Altman: I wouldn't even say for a project the size of AFS [Andrew File System] that one person really knows all of the code. There're nine million lines.

da Cruz: What can I say. I'm a dinosaur. I come from the days where you could understand the computer. You could even understand the electronics of it. You could understand the gates, and the logic and the memory and everything. Now, we're helpless. We have no idea what these things do, how they work, what they're doing to us, what they're telling other people about us. Anyway, so I wrote this thing where I said here's how I do my work. I use Kermit. I go to the Unix server and I use MM [Columbia Mail Manager] for e-mail. MM from the DEC-20. We rewrote it in C and I'm still using it. I'm probably the last one who still uses it. It doesn't have any MIME support whatsoever. I see the actual base 64 on the screen. Usually, it's stuff I don't want to read so I just delete it. If it's a message that maybe it's from somebody that I know that sent me a picture or something I download it because MM has Kermit. It will download the message to my PC and automatically pops up in whatever—?

Altman: Photoshop.

da Cruz: —then I see the picture. Only after I look at it first to make sure it's not a booby trap or something. What else? In those days, it was just too dangerous to even conceive of using Internet Explorer on your desktop, so I basically said, “Don't use it. You can stick with Netscape because it might not be any better but all of the attacks are concentrated on the Microsoft products.” Columbia said, “Put a big disclaimer on it, that's your opinion, this is not the opinion of Columbia University. We do modern things and we're cutting edge and we're the leading edge of enterprise computing. And here you are playing with these ancient tools.” I said, “Yeah, but look. I get more work done in a day than everybody else gets done in a week.” In those days, the computer centers in the different universities used to publish newsletters. One day, the University of Oregon newsletter came and there was my article. In other places they would say, “This is the best thing I ever read.” It's sad because Columbia was on the very forefront of computing in the 1940s and all through the '60s. Then it just lost it. It completely lost it. Other things in life are more important than computing, but it's not like Columbia turned towards those things. It turned towards Wall Street basically. Money, money, money and real estate, real estate, real estate and snapping everything up and evicting people and raising rents and raising tuition. Are people learning anything more? Are people learning better stuff? I don't think so. In the 1960s all of the students here knew about the world. They knew what was going on. For one reason because it was a male college and it wasn't co-ed yet. Everybody had the draft hanging over their head and there was a war going on. Now, we have six wars going on at a time and nobody even knows what they are. Columbia students don't care. As long as they have their iPhones and their Starbucks, that's their whole life. You better delete this whole part.

Catchings: Redacted.

Bochannek: That's okay. Any other thoughts? We're in the open discussion.

Altman: I think that working on Kermit was a blast. I don't know how many years I worked on it.

da Cruz: Jeff worked even more hours a day than I did and that's saying something. He would be online any time day or night.

Altman: Well, that's true and it's not true. What I loved about my time at Columbia, working on Kermit more so than at Columbia specifically, was that our user base was scattered around the world. There wasn't a good time of day to work. There was a good time for the people in Australia and there was a good time for people in Japan. There was a good time for the people in Eastern Europe and in Western Europe and here in the US. My work schedule was work a few hours in the morning, go off to go do something. Come back, work a few more hours during the day, go out and take a dance class. Come back, work a few hours go off and play volleyball. Go out drinking, come back work until a few more hours over night, go to sleep and start the routine over in the morning which might include going horseback riding in Central Park or playing golf or Frisbee. During the time that I worked on Kermit fulltime, it was a very—even though I worked long hours and basically seven days a week for the entire time that I was working on it, I never felt burned out. It was well-integrated into everything.

da Cruz: You get a lot of energy out of creating things instead of just doing something that somebody tells you to do.

Altman: I didn't have to come to a specific office every day. It didn't matter where I was as long as I could get online to do the job. It was very rewarding because people would send the e-mail and they would be having a problem. We had built into Kermit 95 and into C-Kermit all of the things you needed to do from a debugging perspective to capture data, to send to us, the things that we would need to be able to recreate exactly what they were experiencing. I wish I could do that in AFS.

da Cruz: Yes. If there's a problem, we would fix it and they would be just so flabbergasted [because most other software providers would not even talk to them, let alone fix problems in their software].

Altman: Usually, there would be a fix within a day for a terminal emulation bug. If I was able to come in, I would come to the office because I had to go grab a server off the rack more often than any other reason or on days when we were doing builds across just about everything. I mean we had Kermit 95 building on a version of Windows running on SPARC which nobody knew about. We had OS/2 running on MIPS [Microprocessor without Interlocked Pipeline Stages]. We had a version of OS/2—was it the 64-bit version of OS/2 that we had. It was something IBM never ever got released. We had Kermit running it.

da Cruz: Oh, yes. We had tons and tons of equipment that manufacturers would give to us so that we could develop Kermit for it.

Catchings: It would be ready on day one when they went public.

Altman: They probably couldn't even use it until?

da Cruz: Or even the opposite. For instance, Data General, when they started to go downhill, people all wanted Kermit on their Data General boxes so they could get files off of it. [*FDC: Data General paid us a lot of money and gave us a lot of equipment so we could deliver up-to-date C-Kermit for AOS/VS and MS-DOS Kermit to with Data General terminal emulation so their customers could migrate to PCs (I did C-Kermit for AOS/VS and Joe Doupnik did the DG emulation in MS-DOS Kermit, and we had plenty of engineering support from DG, not to mention a wall of manuals.*]

Catchings: Right, for the transition.

da Cruz: Yes. In the fairly recent years, I had this antiquated Data General Mini with a nine-track tape drive. They make some strange machines. Did you ever see the back of a Data General computer? It's a backplane.

Hendrie: I worked there for five years.

da Cruz: It's just a huge matrix of thousands of pins sticking out. To connect something to it, you have to find the right pins to plug the plug into. Well, anyway.

Hendrie: Very inexpensive to make that way.

da Cruz: Yes.

Hendrie: That's how it got that way.

Bochannek: What about you Bill?

Catchings: I was going to say, there seems like there's a certain irony here since as best I can tell all three of us have established that we may not work well with others. I know I've spent years not being able to work for other people so I work for myself.

Altman: I really think you should qualify that. It's not that you don't work well with others. I think that “you may not take direction from others” might be a better qualification.

Catchings: All right. There we go. A group of people who may well not take direction from others, worked on a product that allows multiple disparate different computers to communicate with each other. I don't know if that's irony or maybe that's somehow the whole point since that's how in order to work together you had to figure out ways of negotiating protocols or whatever. I don't know. It's an interesting contrast.

Altman: I've always thought about computing as a tool. Computing in and of itself is never the end goal in life. We need tools and you need to allow other people to communicate and share information. I'm doing the same thing today with AFS that in a sense that I was doing with Kermit except I'm trying to do it now on a global scale where you're connected all of the time and anybody can share data with anybody else from anywhere in the world, any device that can be on the network. I'm just not supporting networks that are made up of two tin cans and a piece of string.

da Cruz: Unlike Kermit.

Catchings: Did you ever implement that one?

da Cruz: It works automatically by itself.

Bochannek: Any good user application war stories that come to mind that you want to share? Any crazy environments that aren't maybe already written up in the *Kermit News*. I mean the Antarctica story is written up pretty well.

da Cruz: Yes, that was a good one. These guys in Antarctica had a PDP-11 which managed all of their inventory and supplies and it broke. I'm forgetting the details but somehow—they had a disk crash that's what it was. It wiped out the part of the disk that had their communication software on it which was Kermit, of course. They needed a new copy of Kermit. There wasn't going to be another flight for six months. If they didn't have the computer working to order the supplies, well, I suppose they could have found some other way to get them like calling someone on the phone or something. I don't know. They had to make the computer work. I actually put a new copy of Kermit on their computer without Kermit being there by getting RSX-11, by getting the binary turning it into a hex file, chopping it up into little pieces. Then dialing up through the satellite and just pushing the little pieces out into a process to copy them on to the disk. Eventually concatenated them all together and de-hexified them and they had a Kermit program, again. It wasn't easy because more often than not there would be a noise on the line or something like that so I had to carefully check every single piece. It took all day but they were real happy. Looking back, I don't think they would have died if I hadn't done that but still. They depended on the thing for—oh, wait a minute. I think I remember. They needed to have Kermit on there so they could send back all of their observation results because they couldn't just store them all up for six months because they didn't have the storage capacity. Besides, there was all of these people waiting in Florida to work on them every day.

I think that's the story. So there was that. Instead of telling all of the stories like that, I just want to remark that one of the biggest frustrations of the last ten or fifteen years is that I could not get stories like this from anybody because unless it's a non-classified government agency or a nonprofit, certain kinds of nonprofits, they're just not going to talk about what they're doing. I would beg them. I would say this would be such a great story. People would tell me vaguely what they were doing and I said, “Gosh, could you give me permission, could we write that up?” They'd say, “No. No.” There're tens of thousands stories out there that I never heard. It's too bad because everything is all homogenized and bland and there's this one way to do everything and it's not very good, but everybody lives with it and it could have been much better.

Hendrie: I have one question. I was just curious about how the code, what you wrote the original code in especially and then how it grew.

da Cruz: On the website we have an archive of that broken down.

Catchings: Originally, of course, you only wrote things in assembler. It was the assembler of whatever machine.

da Cruz: 8080 assembler, DEC-20 assembler.

Catchings: Then, finally, we smartened up and went with C.

da Cruz: I'm not so sure it was smartening up.

Hendrie: Up until the C version you wrote everything in assembler up until the C version pretty much.

da Cruz: Well, we wrote everything in assembler up until the C version. I think the first person who sent us a tape with a new Kermit program on it was a VMS version written in Pascal. Then somebody else sent us—I should mention for these tapes, the Stevens Institute of Technology [in Hoboken] was a huge contributor to Kermit.

Catchings: Early on, yes.

da Cruz: We worked with them for years, they were right across the river, we could take the PATH train. We were networked with them on our little DECnet network with CMU, Stevens and NYU [New York University] and University of Toledo. All of them were Kermit contributors. Stevens was a big DEC installation. It was 100 percent DEC through and through. They had a big PDP-10 at the center. They had VAX's with VMS. They had Rainbows. They had Pro 350s and Pro 380s. They wanted all of these things to communicate. They must not have had a network at that time so they needed Kermit. They had the idea that they would write Kermit and common BLISS [Basic Language for Implementation of System Software] which was the DEC implementation language that runs on all of their platforms. They wrote the Kermit program in common BLISS and then with a minimal amount of filling in system dependent stuff they had it on the PDP-10 and TOPS-10 on VMS and on P/OS [Professional Operating System]. That was a big deal. In doing that we talked with them a lot. We also developed a lot of features in the protocol, a lot of the option negotiations and things are thanks to the discussions we had with them. It was Nick Bush and Bob McQueen. [*FDC: Leslie Maltz was the director of computing. Years later she came to work with us at the Columbia computer center.*]

Bochannek: Did you use software version control on the C-Kermit system?

da Cruz: No. I'm proud to say we programmed the old fashioned way. We just hacked the code. Basically, whatever I did during the day I put it into a place where Jeff could find it [and then the next morning I’d pick up what he did.]

Altman: Yes, I could never convince him to adopt CVS [Concurrent Versions Systems] or anything else.

da Cruz: Well, that's because the next day, "CVS??? No don't use CVS, nobody uses THAT any more!" Besides, I was swapping code back and forth with people, including myself, on platforms that did not have CVS or any other version control system. But at bottom, I just don't like bureaucracy.

Altman: More importantly, we didn't use any type of autoconf or other configuration language.

Bochannek: Right.

Altman: The make file for Unix is the largest make file on the planet.

Bochannek: It's making very extensive use of pre-processor directives in the C-files as well in the compilation system.

da Cruz: Right. Not taking advantage of more modern features of the C language because you still wanted to compile on some of the older platforms. I just released C-Kermit 9.0 a few months ago and I built it on 4.2 BSD [Berkeley Software Distribution]. How many people can build their applications on 4.2 BSD? I'm the only one. It's a big feature-full program. It does millions of things and it's only about two megabytes.

Altman: Info-ZIP on their website had the most—their byline was the most portable application in the world after Kermit.

da Cruz: No, actually it said—what was it? It was something like that. Hello World! was in there somewhere too. They decided that Hello World! was not that portable any more because to do it on modern platforms you have to have all of these prologues and epilogues. But it's true. You can build C-Kermit on hundreds of combinations of hardware, operating system, and operating system version.

Bochannek: I think that's it. Thank you very, very much.

Hendrie: Thank you for taking the time and doing an oral history for the Computer History Museum.

END OF PANEL SESSION

For Further Reader and Background (as supplied by da Cruz):

History of Computing at Columbia University:
http://www.columbia.edu/cu/computinghistory/

IBM 407 (my first "computer"):
http://www.columbia.edu/cu/computinghistory/407.html

Wallace Eckert:
http://www.columbia.edu/cu/computinghistory/eckert.html

Columbia 1968:
http://www.columbia.edu/cu/computinghistory/1968/

What was so special about the DEC-20:
http://www.columbia.edu/cu/computinghistory/dec20.html

What is a terminal?
http://en.wikipedia.org/wiki/Computer_terminal

Overview of Kermit protocol, software, and issues:
http://www.kermitproject.org/kermit.html

The Superbrain microcomputer:
http://www.columbia.edu/cu/computinghistory/superbrain.html

My old artifacts (p.8):
http://www.columbia.edu/cu/computinghistory/books/

Marvin Herzog, Yiddish Encyclopedia:
http://forward.com/articles/179799/mikhl-herzog-columbia-professor-and-yiddishist-die/

International Postal Addressing:
http://www.columbia.edu/~fdc/postal/

Kermit Bibliography:
http://www.kermitproject.org/biblio.html

The historical Kermit software archive:
http://www.columbia.edu/kermit/archive/

1994 Brazilian Election:
http://www.kermitproject.org/newsn6.html
http://www.kermitproject.org/kn6_cover.html (photo)

Character set conversions in Kermit:
http://www.kermitproject.org/csetnames.html

Moscow Kermit Conference:
http://www.columbia.edu/cu/computinghistory/ussr/

Ocean floats for studying hurricanes:
http://www.kermitproject.org/em-apex.html

MS-DOS Kermit:
http://www.kermitproject.org/mskermit.html

Kermit 95:
http://www.kermitproject.org/k95.html

C-Kermit:
http://www.kermitproject.org/ckermit.html

E-Kermit:
http://www.kermitproject.org/ek.html

IBM Mainframe Kermit:
http://www.kermitproject.org/k370.html

Bill Catchings' company:
http://www.principledtechnologies.com/

Jeff Altman's companies:
http://www.secure-endpoints.com/
http://www.your-file-system.com/

Joe Doupnik:
http://support.novell.com/community/volunteers/1joed.html