Oral History of Paul Laughton

Interviewed by:
Bruce Damer,
with Len Shustek

Recorded: August 14, 2013
Mountain View, California

CHM Reference number: X6895.2014

Bruce Damer: I'm going to read oral history questions for Paul Laughton on and around the history of the creation of the Apple II DOS in 1978, at the Computer History Museum on August 14, 2013. Thank you, Paul, for agreeing to do this.

Paul Laughton: Thank you for inviting me.

Damer: It's an honor to be a part of capturing this wonderful history. I've got some questions prepared. And we can diverge from the questions; we'll probably end up answering many of them in the previous ones.

Laughton: Okay.

Damer: I hope they'll help joggle old memories and capture this wonderful period of time.

Paul's Background

Damer: The first question I have is: Paul, could you give us a summary of your early history? Where you were born and raised, schooling, what were the formative influences in your early life, and how did you make your way into the field of computing and come to be in Silicon Valley at such an auspicious time in its history?

Laughton: Well, I was born during World War II. My father was an engineer at the MIT Radiation Laboratories, and he is what we would call today a geek. He was very much into engineering. A very, very smart guy who didn't have a high school degree, but he did work as an engineer. He worked for the Navy after that. I got interested in computers in, I guess, the early sixties. We went to the Naval Research Laboratory, and they had a computer on display. I thought that was really kind of cool. Later, at Virginia Tech, I took the only computer course they had available. It was Fortran. Also, in the electrical engineering department we had an analog computer class. So those were the two courses that I had. Around, I think it was, 1968,[actually, September, 1966. I still have this issue] there was a *Scientific American* magazine that came out about computing [the issue title was "Information"], and it was just wonderful. I mean, it totally fascinated me—the things they were talking about, and the possibilities, and real time displays, and Star Wars games they were playing on CRT's. It, just like I say, fascinated me.

I took my first job at the Applied Physics Laboratory at Johns Hopkins University. I was called a system engineer, but I was really a glorified computer operator. There I learned IBM/360—360 assembly language, 360 operating system—and it just got permanently in my blood at that point. Follow on, there was an ad in the paper for people to work on the IBM/360 for the Air Force Manned Orbital Laboratory.

This was the Space Station that was going to be run by the Air Force for spying. I got hired by Lockheed to come out here and work on that. I was here a month when it was cancelled, which I guess is typical of contract work. After that I went to work for IBM in their CALL/360, CALL/370 [timesharing] operating system. So I worked continuously on the operating system for eight years. That, and on languages. I did a BASIC compiler essentially, but mostly operating system work. All aspects of it, the Disk Operating System, timesharing, networking—every last part of it. And that's how I came to be able to do Apple DOS.

Damer: That answers the next question. You had all the fundamentals down for real operating systems, so when it came to do a tiny operating system you had all the components you needed: all the training and the background you needed to fit something into such a small machine.

Laughton: Yes. Actually, it was what I would call a piece of cake. The way of doing it was obvious. Everything that needed to be done was obvious. I had done most of the stuff in one way or another before, and so I was able to just sit down and do it.

Shepardson Microsystems

Damer: How did you meet and come to work with Bob Shepardson? What projects was Shepardson Microsystems working on in the mid-seventies, and how did he come to work with Apple on the BASIC interpreter earlier—for example when you joined; I guess it was about that time.

Laughton: Well, Bob had worked on an assembler for the IMP-16 by National Semiconductor. He wrote the assembler under contract to National Semiconductor. He then modified that assembler to write an assembler for the [Motorola] 6800. There was a character involved with Apple, a guy named Marty Spergel, who tried [Marty sold these TV interfaces at the request of Apple. Apple did this because they did not want to deal with FCC. Marty sold tens of thousands of these interfaces.] to make money selling TV interfaces for the Apple II, and he was working with somebody else on a 6800 computer. He had contracted with Bob to write a BASIC for the 6800. Marty, I guess, got scared off before that, sold it off to somebody else, and the BASIC never got written. Shep then proceeded to have a contract with Cromemco for a BASIC for, I think it was, a Z80 BASIC.

While he was just finishing that, Apple was a few doors down. A very, very small company: 10, 15 people. They were working on a new computer called "Apple Annie", and they wanted a BASIC for that. The idea of the BASIC was it would be in a cartridge. They knew Bob had written a BASIC, so they came to him and wanted him to work on it. Well, Bob was busy doing other things, so he put a small ad in the paper and I responded. His small ad was for somebody to come in and do the BASIC for Apple. I had experience with BASIC, I had experience with many different compilers and assemblers, and I also had experience with the [MOS Technology] 6502 [microprocessor used in the Apple II]. So Bob hired me right away, and right away I started working on BASIC for Apple Annie.

Damer: So, tell us more about Annie, because Annie never saw the light of day. Was it a follow-on to the Apple II?

Laughton: It was supposed to be a follow-on. A lot of the circuitry that was in the Apple II was going to be put in larger-scale integrated circuits. Even at the time, they had large scale integrated circuits. And, like I say, they were going to have cartridges. The BASIC was going to be in a cartridge, and other things were going to be in a cartridge. The disk interface was going to be part of the main motherboard. I really didn't know a lot about Apple Annie, other than I was writing a BASIC for it.

About the same time we were writing it, there was a guy named Jef Raskin who showed up and wanted more and more and more and more in this simple BASIC. We contracted for simple BASIC. We ended up naming it, "Notzo BASIC," N-O-T-Z-O, because of Jef Raskin continually changing it. That led to a need for another programmer, so Shep hired my wife [Kathleen] O'Brien to help work on it. [Kathleen] and I ended up working together at Shep's, and we continued to work on this Notzo BASIC.

Writing Apple DOS

Laughton: One day Steve Wozniak came in the office showing off his new disk drive. He was showing it off to Shep. He had talked to, I think his name was, Gary Kildall of Digital Research about a DOS for Apple. Apparently Woz didn't really like Gary's approach. He just happened to mention to Shep that he was looking for a DOS, and I said "I know how to do that." Within 24 hours, that led to an agreement that I would do the DOS for the Apple II disk drive.

Damer: We see, in the documents you provided, that somewhere in April of '78 was that contract between Shep and Steve Jobs. Was that happening around March, April time frame? Do you remember?

Laughton: The contract was signed pretty quickly after we made the agreement. Shep liked to get things in contract without the aid of lawyers. It was just quickly written up and Jobs signed it, and I was on my way.

Damer: Gosh, you started as soon as that document was signed? The delivery was for early June of '78, so we're talking, sort of, middle of April '78 through to early June of '78?

Laughton: Yeah, I don't remember those details; I'm sure you have better access to them than I do right now. But, yeah, it was pretty quick. I knew what needed to be done, and I knew how to do it. I was confident that I could do it pretty quickly.

Damer: You were provided a set of specs from Woz, sort of handwritten notes. There were also circuit diagrams, all in one packet, for the beginning of your work. You had access to a minicomputer with a 6502 emulator, is that how it was?

Laughton: No, it was a 6502 assembler.

Damer: Assembler.

Laughton: DOS was written on 80 column punch cards. I would actually write out the code on 80 column punch card sheets – hand write it. We had a guy at Shepardson named Mike Peters who would take those sheets and punch the cards. The punch cards would then be read into a National Semiconductor IMP-16 and assembled, and paper tape produced. The paper tape was read into the Apple II by a card made by Wozniak. It was a plug-in card to the Apple that would read the paper tape in, and I would proceed to debug it.

Damer: So that whole process—how long would it take? Did you have that setup next to your desk, like an ASR-33 attached to an Apple II with the prototype disk drives? What did the setup look like?

Laughton: The setup was: I had an Apple II with a single Apple disk drive at the time. It was an Apple II given to me by Apple. Steve Wozniak [it was Steve Jobs, not Steve Wozniak] came over and put in the extra 32k of memory. Steve Wozniak? Steve Jobs. At first it was for the development of the BASIC, and then when I got the disk drive, it had the disk drive on it. My process usually was to write some code on these sheets of paper and Mike Peters would [punch] them up and then he would eventually give me a paper tape and I would run the paper tape into the Apple II and proceed to debug. As the project got further along and the code was all written, and it was debugging and updating, I would mark up a listing and give it to Mike Peters who would then change whatever was necessary and deliver me a paper tape and I'd start again.

Damer: So you had this wonderful team; it sounded like a pipeline process.

Laughton: Yes.

Damer: Did this hearken back to your days at IBM when you were doing the timesharing and the operating system work, that kind of teamwork?

Laughton: Yes, sort of. I think at IBM I actually did all my [own] punch cards, and went to the 360 and did my own assemblies, and did my own testing. During periods of the day I'd have my very own 360. It was like having a PC except it was this huge mainframe.

Damer: That was at home through a teletype terminal?

Laughton: Well, once the operating system was built, we needed to test. We would have testing for all the features. I could call in at home from my Selectric terminal. We had little modems in big wooden boxes where you put the phone in. I would actually play games and do all sorts of things from home. It was like having a home computer.

Damer: In 1968?

Laughton: In 1968.

Damer: So in a sense, that prepared you for the whole movement to the home computer revolution. You kind of had lived there already to some degree.

Laughton: I had. One of the first things I wrote was the "eight queens problem": how do you put eight queens on a chess board such that no queen can take another queen? I wrote a program, a [PL/1] program, to solve that problem. It was just fun.

Damer: This spirit of fun, which certainly infused the Homebrew Club and that whole milieu, you were clearly already into that. Did you ever attend a Homebrew meeting?

Laughton: Yeah. Actually there was a book called *Fire in the Valley,* and there's a picture in there that has myself and my wife sitting in the background. We went several times, just part of the crowd. My wife is a geek also. Today she teaches computer science at San Jose State, mainly Java.

Damer: You may have run into Len [Shustek] at the meetings. It was a long time ago. But as you were working on BASICs, I know that Bill Gates talked about all the machines that they wrote BASICs for, and one of them was Cromemco. Did you encounter Microsoft or Bill at that time period? Or Paul Allen? Was there a kind of a competition—in terms of producing BASICs for these machines?

Laughton: We never actually crossed paths. After Apple DOS, I did a BASIC for Atari. Atari had bought Microsoft BASIC, but they couldn't get it up and running in time. So they came to Shepardson for a

BASIC for their Atari home computer. Later I worked for Atari. I met Bill Gates several times while working at Atari, but we never crossed paths in terms of BASICs.

Damer: In that whole period of Apple's history, were you working physically at the Apple offices, or were you working at Shep's offices, or were you working at home?

Laughton: I was mostly… on the DOS I would work in Shep's office. My typical day would start about one o'clock. I'd come in from one until about five or six and go out to dinner with my wife and maybe some of the people from Shep, and then come back to work about ten or eleven o'clock and stay until about three a.m.

Damer: The real productive period.

Laughton: Exactly. Then I would a leave marked-up listing. Mike Peters would have a new tape for me when I came in the next day, and start the process all over again.

Damer: Did some of the other early people—like Mike Scott, the first president of Apple, or Daniel Kottke—did you encounter some of those early Apple people while you were there?

Laughton: I encountered Mike Scott briefly; he never really got too involved with me. My main contacts at Apple were Woz and Jobs and Randy Wigginton,

Damer: Randy.

Laughton: Yeah. I've nothing but admiration for Randy. He must have been 17, 18 years old at the time. Just a pure genius.

Damer: We did an event years ago, about Apple in the garage. We had Randy there, and Woz, and a panel of people who actually worked in the garage. Randy rode his bike from school every day to be part of the Apple team then. It's just fascinating to see his staying power. He was the one, I think, who commented about, "Do not cross page boundary" in the code, so he has a good solid memory of working on that.

Laughton: Yeah, I think he and Woz actually developed that routine in a hotel at the Consumer Electronics Show.

Damer: Which routine was it?

Laughton: The read/write track/sector [(RWTS)]. That's kind of the very core. It's a thing that reads and writes the disk, and everything evolves out from that.

Damer: So where we see Woz's hand-drawn notes in that red felt pen, it's him drawing out the read/write sector [routines]. Then there's some other documents which look like they're photocopies, but they're marked up, which show more details, or block version of the read/write sector. That was done over Christmas of '77-78?

Laughton: I think so, yeah. It was for the Consumer Electronics Fair, which is in early January. This was before I met them. But the read/write track/sector—Woz and Wigginton actually sat down and wrote it out by hand, and then they compiled it or assembled it by hand, and then they took the opcodes and entered them into the Apple II by hand. They didn't use an assembler or anything else; it was all hand done.

Damer: Hand done.

Laughton: Actually the Integer BASIC in the Apple II was done the same way. Woz did that whole thing by hand.

Damer: In some of the other documents we see handwritten assembler. That's done by Woz or Wigginton?

Laughton: By Woz.

Damer: By Woz.

Laughton: Yeah. And then when I did the DOS, I actually took that handwritten read/write track/sector [routine] and put it on punch cards and then included it as part of the DOS.

Damer: So that was the code that was delivered to you from them, but you had to key it in?

Laughton: Right, it was just a piece of paper.

Damer: Just a piece of paper.

Laughton: Yeah, it's a small piece of code. I mean, it's probably no more than a page long.

Damer: We hear the stories about the NorthStar disk controller, which I have in my collection at the DigiBarn; an enormous board with at least 24 chips on it. And then there's Woz's design with…

Laughton: Seven.

Damer: …seven chips.

Laughton: Right.

Damer: My understanding, maybe you can clarify, is that Woz didn't understand the design of a disk controller. He'd never really seen one, so he made something that just was elegant and simple and low chip count. He used software to do a lot of things that these other companies were doing with hardware. Can you comment on that?

Laughton: Well, I was familiar with the other disk controllers, and when I saw Woz's design I was just blown away. I mean, it was so simple, and so elegant. And the fact that he was doing it in software—a lot of the work that was done by those massive amounts of chips was done in a few lines of software. It was just pure genius.

Damer: Why would no other company have come up with such a…

Laughton: Because no other company had Woz.

Damer: Right.

Laughton: Like you say, he didn't know how to do it, and he created it in a way that is uniquely Woz.

Damer: So like the whole stepper motor control, everything was done through software?

Laughton: Everything was done through software, just amazing.

Damer: The design that he and Randy delivered to you as code, as assembled code—you put that in as a block into your project?

Laughton: Right.

Damer: Did that design stay pretty much the way it was?

Laughton: Exactly. It never changed during the time I worked on it.

Damer: Your first delivery, we note from the listings, was in early June of '78, pretty much within a week of the target of that one-page contract. Which to my mind is why I've made the statement that yours is the "programmer's greatest generation." We talk about the World War II generation being the greatest generation; I think Tom Brokaw may have coined that term. But of course, it's based on all your prior work and your preparation, that you can sit down and just stream that out and make that delivery.

Apple DOS: What It Was, and Wasn't

Damer: The first delivery in June of '78—was that pretty much running and doing what they had planned to do? Following that, I know there's the whole summer of '78, all the way to the October final delivery. What characterized those two phases of the project?

Laughton: When I began the project, what I had in mind was a file management system. This creates a structure whereby you can open, read, write and close files and that's what my goal was and that's what I created to start with. In order to use it you had to write some code. If you had an assembly language program, you could write a little piece of code to fill in a control block and call my code and you would get a file opened. You would get data from the file. You could put data into the file. But there were no commands; you had to do it yourself. That's what I did, and that's what I delivered.

Everything about the BASIC interface came after that. When we talk about changing the specifications, my idea of the specification was a file management system that somebody else would write code to interface. It developed that I ended up writing the code to interface BASIC to that file management system. One of the things that always disturbed me was I had a very clean interface to the file management system. There were control blocks, and places you could call to do things. And it never really got published. Instead, what you saw was the BASIC interface.

Damer: So really, in some sense the full power and efficiencies of the Apple II DOS could have been greater if they had published in what we now today call an Application Programming Interface and allowed the developer to work that way, rather than just through the BASIC interpreter.

Laughton: Exactly. I have encountered people in my life since then who disassembled the code and figured it out. They ended up writing interfaces to it the way that I thought it should be done. These were mainly game developers.

Damer: The game developers always push those limits. They want to get as close to the hardware as they can.

Laughton: Exactly. I just met a guy about a month ago and we were reminiscing over it. He was explaining how he could do 6502 assembly language in his head. So I popped a question: “What’s an RTS?” And he said “Six-zero." Those guys, and me—we just knew this stuff by heart.

Damer: That’s why I think of you guys as the greatest generation.

Laughton: Actually, I don’t think that the engineers today are much different. I think they work just as hard, just as long hours, and have just as much drive. I know some people that work at Google, and these guys are no different.

Damer: Especially if they’re building fundamental operating system, which they’ve had to. We’ve built and rebuilt operating systems now for 50 years, I guess, 55 years. Having built an operating system that got so widely used—the numbers of copies of Apple II DOS out there were extraordinary, perhaps only exceeded by MS-DOS later—have you seen or do you sense any sort of a trend in operating system development? Have we come farther or are we still just building file systems?

Laughton: Oh, we’ve come a long, long way. The disks are no longer in terms of kilobytes, they’re in terms of terabytes. The structure needed to create file systems under that is immensely much more complicated. What I created was a file management system. A true operating system has a lot of other components to it, interfacing to other pieces of hardware, USB, the networks, screens, keyboards. A true operating system is much, much, more than what I did for Apple.

Damer: In the Apple II design it had the ten slots that Woz had insisted on; therefore you had all these add-in boards in those slots. Did you have to build facilities to deal with those slots?

Laughton: Actually, the design that Woz created had a pretty much set protocol for using those slots. When you plugged a card in, it would essentially know where it was and you could address it that way. During some of my follow-on work with other people at Shep with other companies, we used Apple IIs. For example I worked with a company that created a high speed tape cassette, and they created their own plug-in board. I worked with another company that created a printer, and they had their own plug-in

board. Another company developed a speech recognition thing, and they had their own board. So it was pretty easy for people to design and plugin boards.

Damer: In those projects were you tempted or able to use your own direct interface to the file system in the DOS, or were you using BASIC?

Laughton: I didn't really use any. For example, the cassette tape—it was an interface to drive that cassette tape. What the company wanted was something equivalent of read/write track/sector for their cassette. They were going to do everything else. For the printer, it was mainly taking characters and pushing them out to the printer. So it was really not much involved in the DOS.

Recap: Starting the Apple DOS Project

Damer: Another question: when did you start working with Apple? What was your first day with Shep working on Apple projects, do you recall? Was it in '77 or '78?

Laughton: I don't remember the exact day. It was probably no more than a month or two before the DOS contract.

Damer: Probably March of '78?

Laughton: Yeah, probably.

Damer: When you would go in—it was just down the hall—it was in the same building at some point?

Laughton: Yeah. It really wasn't a hall, but the way the offices were arranged, each one had doors facing out into a mall. So it was down the mall.

Damer: Down the mall, not down the hall.

Laughton: Yeah. The mall is still there.

Damer: That's how Shep met Jobs and Woz, because they were just physically co-located?

Laughton: Exactly. And Shep had a reputation for having done microprocessor type stuff.

Damer: The right place at the right time.

Laughton: Yeah, exactly.

Damer: This was before they moved to Bandley Drive [in Cupertino][Yes, in Cupertino]?

Laughton: Yes.

Damer: What was the address of that location?

Laughton: I don't remember.

Damer: Somewhere in Cupertino?

Laughton: Right now there is a Panera Bread and a Starbucks where Good Earth used to be. If you go behind that, those same buildings are still there.

Damer: Gosh. When you would go into the offices—you said there were 20 or so people there—what was the sense of it? What was the feel of the energy level, the commitment, the "buzz", as we might call it?

Laughton: I actually never went in to the Apple offices. They were right over there, but I was probably two to three weeks into the DOS when they moved to Bandley Drive. Most of the time when I met with somebody, it was over at Bandley Drive.

Damer: Another question that I have is: without a real operating system, a disk operating system, and without a real disk controller—perhaps you maybe could correct this—the Apple II sales would have been impacted by only having a cassette interface. Was that the case? Did you sense and was there urgency behind the DOS project?

Laughton: Well, I got a real sense of urgency, but I really didn't get a lot of sense of what all else was going on at Apple. I was highly focused. When I would come over it would be to talk about the DOS or something, or BASIC. There were incidental contacts with people, but I really didn't get a reading of their morale.

The Influence of Steve Jobs

Damer: Andy Herzfeld has written about the creation of the operating system for the Mac, and how Steve [Jobs] would look at every feature, and he would be concerned about the boot times. He would ask Andy to shave a second off of here or there, etc. Was there any of that kind of interaction with Steve Jobs as a designer, as a project leader with Woz?

Laughton: Well, in the BASIC. "Nutso BASIC", not so BASIC.

Damer: Nutso BASIC?

Laughton: Yeah. At Shephardson we called it "nutso", because it was growing so out of proportion. My wife, Kathleen O'Brien, Kathy O'Brien, was working on it with me. One time we were over at Apple and we were complaining, whining how big it was getting. Steve made some comment about, "It shouldn't be a problem," and my wife grabbed him by the collar, the shirt, pushed him against the wall and tried to say, "Steve, this is a problem. You need to listen." And he listened. At least he seemed to listen.

Damer: My gosh. Shades of Alvy Ray Smith's confrontations with Steve at Pixar years later.

Laughton: I hadn't heard about that.

Damer: Jef [Raskin] had come from the University of California at San Diego, into the company at that time. What were his reasons for adding so much to the "nutso" BASIC?

Laughton: My understanding was he was supposed to write the manual for nutso BASIC. So as he was writing the manual, he was creating specs and he was just adding all sorts of stuff. But his job was to write the manual, and that was driving the project then.

Damer: There must've been some conflict emerging between Jef and Steve Jobs at that point.

Laughton: If there was, I didn't see it. And Jef's involvement in the DOS was fairly minor. He would suggest something here or there. I think he ended up being the guy that wrote the brochure, if you will, that came with the DOS.

Releasing DOS

Damer: This is a good segue into the delivery in October of ’78, when it became an Apple product. A couple of common questions that I always hear, and you’ve heard is, “Why was it a DOS 3.1?” And "How did the handover go?" Who did it get handed over to? What was the whole process?

Laughton: My understanding of how it became 3.1 was: When I would do a new version of the code, which was almost daily, I had a version number, and I would increment it. I think it started out at 0.01, and for each new increment it would go up. By the time I delivered, I think the listing had something like 3-something on it. So somebody just decided to call it 3.1 or whatever. That’s my understanding.

The delivery: In one of the pieces of paper I gave you there are notes about a meeting. This was coming to the completion date; they were wanting to ship, and there was some discussion about bugs, and a discussion about how it would be delivered. We made a general agreement. I ended up having the cards read in in pure text and put on an Apple II disk—the cards for the program. That was how it was delivered.

Damer: So that’s how the actual textual 6502 assembly language version was delivered?

Laughton: Exactly.

Damer: The listings that you provided, these two listings. They were printed out from the National Semiconductor machine?

Laughton: Well, there was a line printer. I’m not sure what the brand of it was. A typical line printer of the day. Big box of paper and paper flowing through it.

Damer: So Apple wouldn’t have an interest in getting such a listing if they could get it on a floppy?

Laughton: Well, the floppy made it easy for them to then use it with a 6502 assembler.

Damer: Did they reassemble? I guess at that point they were using the assemblers on Apple IIs.

Laughton: Yes. I delivered it to Randy [Wigginton] and—I forget his name now.

Damer: Is it Dick Huston?

Laughton: Yeah, Dick Huston. Dick Huston took it over. Shortly after the meeting I delivered a marked-up listing with some bug fixes, and the disk with the source on it. I was done. I really didn't have any contact with them after that. There had been discussion of my doing a verbal documentation. I think once Dick Houston got into the listing, he understood pretty quickly how it was organized and took off running with it.

Damer: It sounds like it was a very, very clean project, from your perspective.

Len Shustek: Who had designed the structures on the disk for directories and for free-space management? Was that you?

Laughton: Yes.

Shustek Was there iteration and negotiation about that?

Laughton: No. It was all designed by me. There was no discussion, no specs. I just did it. Woz, in his read-write track/sector, determined the interleaving of the sectors on the disk so that there were optimal selection and rotation, but everything else was done by me.

Shustek: In designing those structures, how much did you try to anticipate that disks would get bigger, and track sizes would be different, and so forth?

Laughton: Not very much at all. I was highly focused on the job ahead. I had some focus, but I didn't really plan.

After Apple DOS

Laughton: What quickly followed the Apple DOS was Atari coming in and wanting BASIC and a DOS for their home computer. So as soon as I turned the stuff over to Apple, I was off on Atari. That was in October, and I delivered Atari a DOS and a BASIC in January.

Damer: Gosh. What machine was that for, Paul?

Laughton: This was the Atari 400/800.

Shustek: With the peanut butter keyboard?

Laughton: The 400 had the peanut butter keyboard. The 800 had a regular keyboard.

Shustek: So that was a cartridge-based machine.

Laughton: Yes, and the BASIC went into a cartridge. And, of course, the disk operating system was delivered on a disk.

Damer: So that was, in a sense, the Apple Annie, but from another company.

Laughton: Exactly. Actually, parts of the BASIC were what I had been writing for Apple when they canceled Apple Annie.

Damer: These are the days before non-compete [agreements] and all these sorts of things.

Laughton: Right. There were no lawyers involved in the contract. I didn't really use a whole lot of the code. As far as I had gotten on the BASIC was the text-editor portion of it, and part of that is what went into the Atari BASIC.

Damer: That's an amazing thing. In a single year, actually, you wrote two operating systems and delivered a full BASIC.

Laughton: Well, actually there was more than that. I wrote a C compiler for Cromemco in the space of the same year.

Damer: An actual C compiler?

Laughton: An actual C compiler.

Shustek: Kernighan and Ritchie?

Laughton: Yes. Took Whitesmith C on the PDP-11, wrote it in Whitesmith C, and it ran on the Z-80.

Damer: So you wrote the compiler in C?

Laughton: I wrote the compiler in C, yes.

Damer: And was that for the System One?

Laughton: It was for the Cromemco, and I don't remember the model number.

Damer: The one that had the hard drive?

Laughton: Yes.

Damer: Incredibly heavy units.

Laughton: Exactly. It was the eight-inch hard drives.

Damer: That's all in 1978?

Laughton: '78, maybe, '79 at the latest, because I went to work for Atari in '79.

Damer: You're no longer with Shep at that point?

Laughton: Shep kind of lost interest in his business. We had a really good period of time with the Atari contract, and got huge bonuses because we delivered early. It was something like $1000 for every week early, and we were something like 15 weeks early. So, we got a huge bonus from them, and Shep got all enthusiastic about his business. He was going to sell hardware components. And then something happened. He just lost all interest. Really wasn't interested in having employees anymore. Atari made me an offer I couldn't refuse, so I went to work for Atari. And Kathy went back to work for SBC.

Shustek: Did you or Shep ever think about taking stock in these companies, rather than monetary payment?

Laughton: There's an apocryphal story. Before Apple hired a new vice president of software engineering, Steve Jobs came over and offered Shep to become Apple software engineering [VP] for 25 percent of Apple. Shep turned it down. He thought there was more money to be made independently. Like I say, an apocryphal story.

Damer: How long were you with Atari?

Laughton: About two, three years.

Damer: So you were there during that entire incredible growth period when “Space Invaders” had come out and things like that.

Laughton: Exactly. I worked in the home computer division. I was the manager of the home computer division software. I left Atari to go to work for Fox Video Games. I was director of engineering there and worked on a lot of games, mainly from a management standpoint. At Fox we did games for all sorts of home computers at the time: the TI, the Commodore, Apple. We would turn out the same game for each machine.

Damer: That’s an interesting challenge, given different CPUs and different cartridge formats or disk formats. Complex.

Laughton: It was an interesting time at Fox. Fox considered this yet another project, like they have a movie that’s a project. Well, this is one of their projects, and it wasn’t making money. So they shut it down.

Damer: Kind of in a similar sense to Lucasfilm and Lucas Games, etc. That would’ve been about ’82, ’83?

Laughton: Yeah, about ’82. It was around 1984, 1985 that I joined a startup company doing voicemail and created an entire voicemail system with myself and another guy.

Damer: Amazing. Digital voicemail?

Laughton: Digital voicemail, yes.

Damer: How did that company go?

Laughton: The company went on for a number of years into the early ’90s. Finally went under. A company called VOYSYS I went from VOYSYS to Logitech, and was director of software engineering there. Then in the mid-’90s I got enthusiasm for digital cameras, and persuaded Logitech to come out with the first consumer digital camera. This was ’94, ’95.

Damer: Which model was that?

Laughton: It was called the Logitech Fotoman. Then Logitech and Kodak worked together, and we created the Kodak DC40, which was Kodak's first commercial consumer digital camera. I was really enthused about digital cameras at the time.

Damer: You weren't wrong.

Laughton: No, I wasn't. Our dream at the time was a camera in every pocket, and that's what we got.

Damer: That's what we have.

Laughton: In the late '90s I worked for a company that had a digital camera operating system, and we were interfacing cell phones to cameras. It was kind of the lead-in to what we have today.

Damer: Your at work at IBM, your work at Apple and Atari, digital voicemail, and the cameras was all about small operating systems doing efficient things with files in robust ways that interface with a few devices. [This common approach] led to a widespread propagation of these very, very small but effective operating systems.

Laughton: In a sense, they were all operating systems. Even voicemail was an operating system, because we had to organize a disk for the various users and their messages and so forth.

Damer: This is a through-line theme through all of your work.

Laughton: I never thought about it that way, but that's true.

Damer: And since 2000?

Laughton: I retired in 2000. About three years ago I got interested in doing software engineering again, and I created a BASIC that runs on Android. This BASIC now has about 50,000 downloads. I released it under a GPL/GNU license, and now I have a team of developers working on it, and also have a volunteer manual editor working on the 120-page manual. So once again into the computer world.

Damer: Does it implement all the features that Jef wanted in nutso BASIC?

Laughton: It implements all those and much, much more. It touches just about every API in the Android. You can manipulate pictures. You can send and receive pictures. You can do FTP, all sorts of networking stuff. Whatever you can think of that you can do on an Android, you can write a program in BASIC to do.

Damer: So 35 years ago, sitting there down the hall from Apple, building this file system OS that connected with a BASIC—if you could've seen yourself in 2010 or '11, whenever this project started, doing a BASIC for another device that has really good access to its file system and all of its services, but it's held in your hand and it has gigabytes of memory and it's connected to a worldwide wireless network and can play video and all these sorts of things—did you ever cast your mind forward, ever dream that things would evolve so incredibly in that 35 years?

Laughton: Well, I certainly had an idea that computing, home computing, would evolve quickly. I don't think I ever really forecasted the smartphone or the tablet, but if the me then came forward to today, I wouldn't be surprised, if you will. Android's an operating system. I really haven't delved into it much because I use all their APIs, but I certainly have a lot of empathy for it and the issues that the Google engineers have.

Damer: I'm hoping that this interview at some point will be online and available, and that some engineer —like a younger Google engineer—will hear your story and say, "Hey, I'm working along the same lines, the same lineage, the same thinking", and feel connected to computer history. Which, of course, is why we're all so passionate about this field.

Laughton: There was recently a show on TV; I think it was called "Silicon Valley." It was kind of a reality show with some guys and girls working in Silicon Valley. My watching the show and the way those guys worked, it was really no different than the way we worked back in the '70s.

Shustek: There's a new generation of very small computers built out of Arduinos and Raspberry Pis. Have you considered working with them and porting some of your software to them?

Laughton: Actually, my current interest is in amateur radio. There's something called software-defined radio, where all the capacitors and coils and all the stuff that used to be in a big radio—tuned circuits and so forth—are now eliminated. You essentially capture the entire RF spectrum and digitize it, and then send it to a computer to work on. So my interest is in software-defined radio, digital signal processing.

Damer: I think there was such an incredible crossover between the Ham community and the early personal computer community. There's a note in the second issue of Byte in October of 1975, an angry letter sent to the editor—I guess it was then Carl Helmers—saying, "Why did you turn a perfectly good amateur radio magazine into this computer thing?" And then in the second edition of the Altair Notes there is someone who connected an Altair with a Ham radio system using an A-to-D converter and

everything, to be able to send mail wirelessly in New Mexico. He created a mailbox application in the summer of ’76...

Laughton: Amazing.

Damer: …from S100 machines. So that Ham-computer connection was always strong. The same kind of thinking, and the same kind of people.

What Apple II DOS Was, and Wasn't

Damer: This leads to another technical question I wanted to ask. What was specifically done in the DOS to support access to what we call now a frame buffer or screen buffer, to allow gamers to write to the color display of the Apple II?

Laughton: There was nothing in the DOS that did that. That was all in Steve Wozniak’s [Apple II ROM code]… He had an operating system of sorts for interfacing the screen memory, and for the plug-in cards, and so forth. I was saying that what I did wasn’t really a DOS; it was a file-management system. The other part that would make a DOS is something that Woz did in the Apple II ROM.

Damer: So really, in some sense, the Apple DOS is a collaboration between yourself, Randy, and Woz. Put it all together and it was the DOS.

Laughton: Yeah. Like I say, my goal was to create a file-management system. Then on top of that we added the command interface to BASIC. You could type in "Load" in BASIC and it would end up going to load a file. Those were all add-ons. The specs for that were mainly done between Randy and I, and with a bit of Jef Raskin, and a bit of Woz.

Damer: There’s a letter in June of ’78 that talks about a couple of changes that are needed, and that they’ll be done by tomorrow, and it’s for $500 <laughs> which is an interesting dollars-per-line-of-code kind of thing. That was happening after your first delivery, the first listing. Those kind of add-ons of Randy, and adding that stuff to finish..

Laughton: Somebody would want things. I would usually write it down on a piece of paper and then take it to Shep, and he would put a price on it. That’s the way it worked.

Damer: That's interesting. I would gather there would be a number of Apple IIs over at Apple on Bandley Drive that would be running the DOS, a number of people kind of cranking away on it while you were writing it?

Laughton: I really don't know. I would take it over and deliver it to them, and they would come back with something. There were not many people working on DOS at Apple during the time I was working on it. Randy was probably the only one that was working on it full time. Dick Houston hadn't been hired yet, and Jef Raskin kind of dabbled in it. But that was about it.

Damer: It's interesting, because you would think it'd be a core focus of the company.

Laughton: It was, but there wasn't much of a company there. Like I say, Jobs had offered Shep to become the software-engineering department of Apple. We would've been it.

Other Apple II Applications

Damer: This is when you were no longer really in contact with Apple but I wanted to ask about the VisiCalc team building VisiCalc using the shipped DOS. I would assume they built it in an assembler?

Laughton: I really don't know. I didn't have contact with any of those people

Damer: VisiCalc led to the taking off of the Apple II as a true tool for the businessman and the small business, etc.

Laughton: I think it led to the taking off of personal computers, home computers.

Damer: This'll be a follow-up interview, where we try to figure out how the VisiCalc team did it and we ask them "how did you fit your 2K spreadsheet application into this framework efficiently, because was it built in BASIC?"

Laughton: I assume it was done in assembly language. If I were doing it, I would do it in assembly language.

Damer: If it was built in assembly language, how would they interface with your file system without a documented API?

Laughton: Like the game developers, they probably figured it out.

Damer: They reverse-engineered it.

Laughton: Well, there is a book called “Inside Apple DOS,” and I think “Inside Apple DOS” figured it out and explained it.

Shustek: Why do you think Apple didn’t release that API documentation?

Laughton: I really don’t know. I asked them to. I volunteered to write it up, and their response was something of, “We’re busy.” Like I say, people figured it out.

Damer: Years later, when people would talk about how they started [in computing], that their first love was an Apple II, how they built businesses on it, and whatnot. Would you mention your role in the history? Or would you just stay quiet and have a smile on your face?

Laughton: Well, I usually mention my role in it and usually get an “Oh, wow.”

Damer: You collected “oh, wows.”

Laughton: Exactly. I recently sponsored a block party for my neighborhood, and as part of it, I sent around fliers. Some of the people googled me, and after googling me they’d say, “Are you the guy?” It’s been fun.

Damer: Did you have an Apple II sitting at home and throughout the ’80s?

Laughton: No, actually. I moved on. I loaned my Apple II to somebody who was starting a company. I actually had an Apple II and two disk drives, serial number 2 and serial number 3. The person that I loaned them to sent them back to Apple for refurbishing, and they came back generic disk drives, different serial numbers.

Damer: So they melted into wherever.

Laughton: Either somebody at Apple—some engineer’s got it in their personal collection—or it went into the junkyard.

The Impact of DOS on Apple

Damer: I've been calling this little project of ours "How Apple Booted Up" as a company, in the sense that it needed a software platform to allow a developer community to emerge. Does that seem fair? That whole period and this whole project really helped Apple to boot up as a company?

Laughton: I really think so. I think the Apple II was a wonderful machine, but without a mass storage device it wasn't going to go anywhere. Having disks on it meant a world of difference. It was the catalyst that created VisiCalc, and VisiCalc created the revolution, I think. So it didn't just boot up Apple, I think it booted up the home computer industry.

Damer: Interesting. Certainly working on Apple IIs back in the day, I always used to appreciate that little sound that would occur, and the red light would go on, and my Apple II would come alive. Then, of course, the DOS would be there and then it would go to the other drive for my applications. That subtle stepper motor sound, or whatever it was, plus that red light, always gave me a sense of reassurance that my computing world was coming alive. I never imagined I would meet the man that worked on that little sound.

Laughton: You're talking about the 1000-cycle note, the beep?

Damer: Ok, that's right. There was a beep as well.

Laughton: That was in Wozniak's ROM. Whenever Apple would beep [boot?], it would make that 1000-cycle note.

Damer: There was the beep, followed by a sector-read operation. It was very distinctive.

Laughton: Yeah. You could hear the stepper motors going.

Steve Wozniak

Damer: We recently had Steve Wozniak over to my collection at the DigiBarn Computer Museum and we put in original—I think it was Serial Number 400—machine; early ones were built by Dan Kottke at Apple at Bandley Drive. We recorded everything. We took video of the machine so we could get that sound specifically recorded. That floppy drive painted a picture of Woz's face on the screen as an invitation to the [1982] US Festival. The little signature of Woz came out. Woz was delighted to see that it all was working. He had forgotten about his floppy-based invitation that he had sent out to all these people.

Laughton: Woz was quite a guy. He really was.

Damer: As a human being we all know and love him. What was some of your impressions of him?

Laughton: He was interesting. My interaction to him was mainly technical and on the project. We went to lunch together a couple times at the Good Earth. I remember we had a discussion about salary, and I was getting paid—I think it was something like $35,000. He was getting $25,000. Sometimes people ask me why I didn't go to work for Apple. I was making a lot more money as a consultant.

Damer: Interesting. Did Woz talk about his life at Apple, or was it mainly technical discussions that you would have with him?

Laughton: Mainly technical. There was some discussion about his wife, or ex-wife, as I recall. She was known as the condo queen. But not a whole lot of personal stuff, no. We were busy people.

Damer: You were busy people. Would you look over listings when you went to lunch?

Laughton: No. It was mainly discussing what was coming, and so forth. I don't think anybody ever looked at my listing at Apple until I gave that final one to them. We were just moving too fast, and they didn't have personnel who had the time.

Software at Apple

Damer: So, in truth, Shep and you guys were the software division.

Laughton: In truth, yeah. Very much so. Cannot discount Randy. Randy was the one that made Microsoft BASIC work on the Apple II, and that was no small task. If anybody was software engineering at Apple, it was Randy.

Damer: I understand Randy was a completely self-taught man, a very young man. You and Randy would go to lunch and have technical conversations?

Laughton: Not very often, no. Maybe once or twice it was Randy and Steve Jobs and Woz who went to lunch at Good Earth.

Damer: That must have been an interesting table to sit around, in retrospect.

Laughton: Yeah. I don't know if you're familiar with the Good Earth; it's kind of like a Whole Foods.

Damer: Early kind of almost organic.

Laughton: Yeah, exactly. A lot of vegetarian stuff, and vegetable drinks.

Damer: Famous for their tea.

Laughton: Tea, yes. My wife still gets their tea.

Damer: Do you ever recall sitting around the table with most of Apple at one point? Do you recall any of those conversations?

Laughton: It never really happened. Like I say, once or twice coincidentally, and I don't even remember much of the conversation. We were in there for the business.

Damer: You were mostly head down working, you had deliverables to make.

Laughton: Right. Exactly. The same thing when I was working with Atari. It was very little discussion about—there was a lot of politics going on at Atari at the time when I was working as a consultant for them, and I just didn't get involved. I was heads down, working.

Damer: That sounds like a life strategy that actually is fairly sensible advice to anyone.

Laughton: Well, I did contracting for many, many years. Among us contractors, one of the things we like is that we didn't get involved in politics in the company. We did the job that was asked for us, and if there were any other issues, that wasn't our problem.

Damer: I think sometimes the brackets around all of us is simplicity, KISS: keep it simple stupid, or whatever we call it.

Laughton: That was diametrically opposed to my being an engineering manager, where I was the politics.

Damer: So you ended up having to carry the load of the politics to, in a sense, protect the people who were doing the real engineering down the road.

Laughton: Exactly, very much so.

Damer: These are just the last few questions, and Len, you may have some, too. Did you ever in the valley visit places like Xerox PARC, or SRI, or go to talks, for examples Stanford talks?

Laughton: Not really. I've been to a few talks here at the Computer History Museum, but most of the time I've just been busy.

Damer: When you worked at Shep, were you living in Cupertino?

Laughton: Actually my wife and I bought a house in West San Jose near De Anza Boulevard, and we still live there.

Damer: Same house?

Laughton: Same house.

Damer: From the mid-'70s?

Laughton: '79 is when we bought the house.

Damer: Simplicity again.

Laughton: Yes. We bought it because it was close to Shep, close to Apple. Then I ended up spending almost ten years at Logitech, which is in Fremont.

Damer: So you could just go up that way.

Laughton: Go up that way on my motorcycle.

Damer: On your motorcycle?

Laughton: Yeah.

Damer: Did you use your motorcycle going to Shep?

Laughton: No, I didn't start riding until the mid-'80s. Then I quit riding in the late '90s.

Advice for the Next Generation

Shustek: What's your advice for young people today who want to do the kind of successful projects that you did? What languages should they learn, what experiences should they have?

Laughton: Well, one of the interesting things is… for example, my son-in-law's brother is the vice president of engineering at a company called Etsy, and he only has a high school education.

Damer: In New York, in Brooklyn?

Laughton: In New York, yes.

Damer: My nephew works there.

Laughton: Okay. I think there are a number of very brilliant kids who have never been to college, or who have done very little college. There are also the opposite; you've got the founders of Google, who were very much academics. I would say to kids who have a dream: just go for it. You don't necessarily need that college degree.

Shustek: Do you think young engineers need to understand digital logic and machine language programming? Or can do they do everything with high functioning boards and C++?

Laughton: I think today they do everything with high-level programming. Even things like digital signal processing is done with a high level language now. The only people that write low level language—I suppose if you had a very small ROM you might do some assembly language. But I doubt if much of anything is done in it.

Damer: Maybe embedded systems?

Laughton: I think even embedded systems are compiled code now.

Shustek: Do you think there's a danger that people who work only at high levels will not understand how the machine really works?

Laughton: I think you have to understand it in order to drive it. For something like an embedded controller, you may work at a high level language, but you have to understand the electrical interfaces because you're driving them, and using them. There's object-oriented programming, and that has led to a lot of reusing of code. For example, today if you want to do a sort, almost nobody writes a sort [program], they just reuse somebody else's sort. My experience working on Android is there are lots of API calls there, and we use those. I think the same thing is true even for Arduino development; there's a library of routines. If you want to talk to RS-232, there's an RS-232 routine that you call. You don't actually go out and drive Data Terminal Ready and RTS [Request To Send], and so forth.

The Future of Apple DOS

Damer: An interesting thought just occurred. As you know, we've obtained permission from Apple to release the [Apple II DOS] source publicly, and it's now been keyed in, at least the October '78 version, and we'll be attempting an assembly of the source. Any advice for the future assembler and people working on that source code?

Laughton: As I said, the assembler used to create this was an adaptation of an assembler that Shep wrote for the IMP-16. You're going to find opcodes that are different, and various directives that are different. It's not going to be that you just load a 6502 assembler and run it. There's going to have to be some changes made.

Damer: So they'll have to write some macros to represent those?

Laughton: I don't think they'll have to be macros, it's just doing it in standard 6502 assembly language is different. It's just finding the different opcodes or directives that do the same thing.

Damer: A follow-on question to that is: if someone out there manages to assemble it and run it in emulation, we would also obviously try to run it on an actual Apple II too , and then possibly group will emerge around extending it somehow, because you can emulate anything on the net. So there could potentially be a growth path for this piece of code. What do you think that you would have wanted to do with the Apple II DOS that never got done? Where do you think it could go from here?

Laughton: If I look at what is available today for storage, just looking at it from a file management standpoint, a file management of today is just so totally different, so complex. You have an immense amount of data sectors available, and organizing that and so forth is just a vastly different job. It's like a lawn mower engine as opposed to a V12 Ferrari. It's just so totally different.

Damer: So somebody extending this DOS, and giving it a new name, because if it's released it could have a future life, what on earth could it be used for?

Laughton: At Shepardson we had this crazy idea of taking the DOS and Apple disk drives and making a generic product for 6502 devices. We were going to do it for the KIM-1, the Sim-1, the Commodore PET. I actually designed a board that would interface the KIM-1 to the Apple II disk drives. I wrote a front end to the file management system that we called CP/A. It was a command line thing just like CP/M was. We actually published a brochure, and Digital Research told us to cease and desist.

Damer: An early example of a legal action…

Laughton: Yeah. We were going to do that, but then Shepardson, like I said, lost interest.

The Origin of Command Names

Damer: Next are sort of tail-end questions. The command line interface—deciding that the word "catalog" would list the files, for example—did you write all that? The command line interpreter and what the commands would be?

Laughton: For BASIC, yes.

Damer: For BASIC.

Laughton: Yeah. I mean, not for BASIC, for Apple DOS.

Damer: For Apple DOS. So the word "catalog"… CP/M I guess used “dir” to list files, and DOS also used “dir” but with different drive letters. When you chose those command names, what was your inspiration for using a name like "catalog", for example? "Run" is obvious.

Laughton: I think IBM. I recall we used the term CAT at IBM, in the CALL/ 370 timesharing system. If you wanted a catalog of your disk, you type in CAT. So it was pretty much lifted from Call/370.

Damer: So you could put in the word catalog, or just the shorter CAT?

Laughton: Right. Yeah.

Damer: Are there any other commands you remember that came from that world?

Laughton: I'd have to look over it. They're in the listing. In the Apple DOS listing you can find all the commands that went through BASIC. I'm sure there were some. A lot of the terminology used in the listing came from IBM operating system. Things like the VTOC, the Volume Table of Contents. This is what you'd call the "directory" today. The sector allocation map, things like that, all came from terminology that I used at IBM.

Damer: So in a way this is a very early IBM/Apple connection.

Laughton: Yes, very early.

Damer: And MS-DOS being on the later IBM PC, when IBM got into the hardware game.

Laughton: MS-DOS was really a derivative of CP/M. It was a derivative for the 16-bit processor that was done not by Microsoft, but by somebody else who Microsoft bought it from.

Damer: I think by Tim Paterson for the Seattle Computer Products Gazelle machine.

Laughton: Yeah, okay. So it really had a different evolutionary path than Apple DOS.

Damer: IBM was building the 5110s, and 5120s in the period, but I guess the IBM PC was a couple of years in the future.

Laughton: I remember wanting a 5120, but they were very expensive at the time. I think it was what, $5,000—something like that.

Damer: Did you look at other languages like APL that would have been available for a machine like that?

Laughton: I actually wrote a Prolog interpreter that ran on the Atari. In my various jobs I have written many different compilers that do things other than compile computer languages; network connections, things like that.

What Came Later for Paul

Damer: So the other side of your career was compiling all kinds of codes to other codes.

Laughton: Yeah. I was fascinated in the compiling, assembly language translation kind of thing, and so I did a lot of that.

Damer: Of course it became protocol conversions later with the Internet, converting to and from TCP/IP and all that, that sort of real-time translation. One of the things that amazes me is the amount of programming that's done through just interpreted scripts today. For script-based programs, like JavaScript, perhaps more code is run that way than with compiled code.

Laughton: Well, Java is compiled into a byte code. My wife teaches Java, JavaScript, so I get to hear a lot about it.

Shustek: The follow-on machine to the Apple II was the ill-fated Apple III, for which Apple was designing their own "SOS" sophisticated operating system. Did you or Shepardson have any involvement in that?

Laughton: No, and I actually didn't know about the Apple III until just last weekend. I was at a ham flea market, and there was an Apple III there. That was the first time, in my memory, that I ever heard of it. I was chatting with the guy that was telling me about it, and he was saying they had this thing called Apple SOS. That was the first I heard of it.

Damer: There's another lost-to-history project in Apple. When Lisa was being developed there was an effort to make sort of a visual operating system, and they called it Mona.

Laughton: I've heard of it.

Damer: I've only heard one reference to this thing that was being built, because there was this effort on the Apple II community to bring themselves into the graphical world. I guess there was that machine in the mid-'80s which could boot into Apple II DOS, or into some kind of a graphical mouse-based interface. It was very klugy, but that was the only entry of the Apple II into the future bitmapped graphic world.

Laughton: Didn't Bill Atkinson work on Mona?

Damer: Oh, interesting.

Laughton: I seem to recall hearing the word from him.

Damer: From Bill?

Laughton: Yeah.

Damer: Interesting. Mona and Lisa.

Laughton: Yeah. Like I said, our offices were right next to each other, and we would both go outside to smoke.

Damer: There's another artifact in my collection [at the DigiBarn Computer Museum], which is a sleeve of chips that say "Brooklyn". They are 16-bit 6502 prototypes—16-bit! There was a proposed machine that never saw the light of day, called the Brooklyn, which was meant to bridge between the two worlds of the Apple II and the Macintosh.

Laughton: Did you say 16-bit?

Damer: 16-bit.

Laughton: Yeah, okay. I think I'd heard of the 16-bit 6502. The 6502 was what we would call a "reduced architecture" today. It was a very simple. A simple architecture as opposed to the [Intel] 8080, or the [Motorola] 68000.

Damer: I believe that Woz chose the [6502] chip just because of cost.

Laughton: Yeah. It was quite a remarkable chip. Commodore chose it. Apple chose it.

Damer: The whole history of MOS [Technology] and all of that—Commodore taking it over. There's another piece of code that Woz provided, called SWEET16. Did you know about SWEET16?

Laughton: Oh, yes. I'm trying to think. I had done something with SWEET16, but I don't recall. I think I may have ported it to the Apple or to the Atari.

Damer: Interesting. So it was a 16-bit assembler?

Laughton: It was 16-bit math.

Damer: Wow. Okay. We actually had very faded copies of handwritten listings of this, and the same fellow who may assemble your source had hand-drawn and filled in and completed [a restoration of Woz' SWEET16 code] so you could actually see it. I presented it to Woz in this building about seven years ago. We had an event, Apple's 30th birthday party, and I presented [the restored "Woz Wonderbook"] to him, and looked and his eyes teared up. He said, "SWEET16, I haven't seen this in 30 years." This is the kind of emotional connection that someone like him has to [computer] code. I don't suppose you have such an emotional connection to your code? Or do you?

Remembering the Process

Laughton: Well, just a little while ago we were looking over this listing. It's the first time I had looked at it in decades. I started getting flooded with memories of the time, and I could actually smell the environment that I was in when I was creating this in the lab at Shepardson's, just picturing this whole thing.

Damer: So these artifacts are like memory palaces.

Laughton: Yeah. It took me back when my wife and I were both working at Shepardson, and we would work for days on end. Then in the middle of the week we'd take Tuesday or Wednesday, or Wednesday and Thursday off, and go skiing when the ski slopes were empty. It was the kind of thing that we could do.

Damer: A new freedom, a new lifestyle actually, was being invented then. Instead of the 9-to-5 lifestyle. It must have been appealing to the contractor that you had flex time. You could work any 100-hour week you chose <laughs>.

Laughton: Exactly, and we did, and we took great advantage of it. You were asking about Shep earlier. One of the things that I do remember about him is he was very generous. There was a time when he and his wife took Kathy and I to Las Vegas for the CES [Consumer Electronics] show. He just paid our entire way, took us out to dinner, took us to shows. And at the completion of the Atari BASIC project he took us skiing to that ski place in Colorado. We went skiing for an entire week. He paid for the entire trip, bought all the meals, bought our lift tickets—Aspen, Colorado. He often would take us out to very expensive

dinners. There was a place called the Town Oak over in Sunnyvale; it was a bar. I had all my reviews in the Town Oak. We'd go over there about noon, and we'd start drinking. I would get drunk, and then sober, and then drunk and then sober. About midnight I'd have to call my wife and say come pick me up, because I didn't want to ride home with Shep who was drunk. But we spent quite a bit of time at Town Oak, and met a lot of people in Silicon Valley. I met Chuck Simonyi. I don't know why he was at the Town Oak. But just a lot of people.

Damer: Chuck was at Xerox PARC, at least until '82 or so.

Laughton: Yeah, I think—I don't know. Okay.

Damer: Is there anything we've left out, or you've left out, that we should put on tape?

Laughton: Well, I think I have very much enjoyed my life as a software engineer in Silicon Valley, and all the different companies I've worked on, and all the different things I've worked on. I'm proud to be part of Apple DOS, but I'd be just as happy if I had worked on other things that were less well known. I'm just glad to have been a part of it.

Damer: In some sense this is a spin of a roulette wheel. Our careers and chips fall in different slots. You've put chips in so many slots in all the projects you've done, which are impressive and wonderful. Of course one of your chips fell in the Apple slot, and Apple went on through the lottery of life to become well known. But certainly all of your other projects, for us computer historians, are equally valuable in terms of the history and understanding how it all fits together.

Laughton: My own personal evaluation is that the work I did for Atari—Atari BASIC and DOS —was much more monumental than the work I did for Apple, because it was so much in such a short time.

Damer: And it got launched on all these different machines, and different cartridge formats.

Laughton: Yeah, exactly.

Damer: Then of course the work on digital cameras. That's pretty monumental.

Laughton: Exactly. It's been a fun life.

Damer: Well, thank you, Paul. This has been a total delight.

Laughton: Thank you, Bruce.

Damer: I'm glad that you one day decided to ship a box of documents to me at the DigiBarn. And apologies for not getting back to you sooner, I was then finishing a Ph.D. and whatnot; I think it was back in November of 2012 that I finally opened the box and realized how important your documents were, and decided to make that the focus of my computer history work in 2013. This is the big project of this year. Thank you for being so willing to help us understand these documents, and then unfold this whole history. It's been an incredible delight.

Laughton: Well, I'm very pleased with what you've done with it. I'm glad I was able to get it to you.

Damer: Now this story and digitized versions of the documents have a home here at the Computer History Museum, and it will have a life out there in the world. Perhaps at some point in the next year or two we might end up doing some kind of an event for the public; I haven't really thought about it much.

Laughton: That would be fun.

Damer: It would be fun to do. There are several people who have expressed an interest; the VisiCalc guys have, certainly Woz has, and Randy was interested. Just to come together and talk about the software—the booting up of Apple.

Laughton: I'd be really interested in talking to Dick Houston, because he took on my work and had to maintain it for many years. I'd like to know what his experiences were.

Damer: That might be a good follow-on interview to put a bookend on this one, as we chart this history.

Epilogue: Examining the Source Code Listings

Laughton: What I have here is a listing, a line printer listing, of the Apple DOS as it was delivered to Apple on October of 1978. One of the first things I notice here, this code starts off with relocation of the DOS. DOS was actually written to be relocatable. It would load into the lower 16K of the Apple II. Then it would look around and find out how much memory was actually there, and then relocate up to the highest memory location. This was the first thing it did every time it booted. This small section of code here is the code that figures out where it is, where it needs to be, and then does the relocation. The assembly code in 6502 had hard-coded addresses in them. So if you were branching or loading from a location, it would branch or load from a location. So a relocator had to change all those addresses as it relocated. It took a tricky bit of coding to write the relocator. I'm just looking through it and remembering doing that, and being

shocked when Apple said that I had to do it. This was not part of the spec, but "You want me to relocate the code?" That's pretty heavy stuff.

Damer: Well, there's a comment in there that I like that says, "If it ain't there…." It's right in the first few pages.

Laughton: If it ain't there?

Damer: It's looking something that's…

Laughton: It's looking for…

Damer: I wanted to ask you about it. It's actually right down here. "Get how long. If it ain't 3, then don't relocate."

Laughton: That's 3-byte operators; it was an operator and address. If it wasn't a 3-byte type operator code, then the following two bytes didn't need to be relocated. That's what that was all about, figuring out it had an address. One of the interesting things, for example, when you typed in a CA command in Apple you'd type in CAT or LOAD. The code would have a look-up table, and it would find in this table the word like CAT, and then it would take the index of that word. There was a vector table, so it would take that index, look into the vector table, and then go to the location where the code for that was. There was no such way of doing that in 6502, but there was a tricky way of doing it. If you had an address and a register you could do what's called "return from subroutine," RTS. If you did the RTS it would go to the address in the register. So the vector table consisted of a bunch of addresses that I'd load into a register. Load that address, and then do an RTS, and that would a vector to the routine. I'm just remembering creating that and how clever I thought it was. Then it turns out Woz had done that.

Damer: Woz and Wiggington noted that you shouldn't go over page boundaries on their handwritten listing. What was that about?

Laughton: Well, this was just in the read/write track/sector. Timing was very critical. They knew exactly the clock speed of the Apple II 6502, and you have this spinning thing that you have to be there exactly at the right time for. If an instruction went over a page boundary it would add an extra clock cycle, and that would mess up the timing. A page is a 256-byte section. So the instructions had to stay within a page in these particular routines, or else it wouldn't read the disk drive.

Damer: At the end you have a symbol table.

Laughton: This is part of the assembler. So for example, there's a variable called TSR, and this tells me everywhere—every line—where TSR is used. It's mainly if I'm debugging and want to know where a variable is used, I can come here and find it rather than having to look through the whole listing. It's an artifact, a good artifact, of the assembler.

Damer: Were you using a line editor to write the code?

Laughton: No, we had punch cards.

Damer: Oh, it's all punch cards, of course.

Laughton: It was all on punch cards. The way things would get changed is, I would mark up a listing and hand it to a guy named Mike Peters, who would go to the keypunch and do all the stuff and put it in the deck, and then run it through the computer that did the assembly. It was all keypunch. We could only dream of text editors at that point.

Damer: No paper tape survived?

Laughton: I would assume not, no. There was no reason to keep them around. Shepardson actually had a business of duplicating paper tapes. At the time a lot of paper tapes were used in the valley, so he would essentially provide a service duplicating your tapes. Or copying paper tapes onto Mylar tapes so they were more durable.

Damer: Perhaps Bill Gates' famous angry letter to hobbyists, where he complains about people duplicating his BASIC—maybe Shep did a few of those.

Laughton: That would be possible. I certainly remember that letter very much, Bill Gates complaining about not getting enough money for his BASIC. It's kind of interesting, in retrospect.

Shustek: Is there anything else you think is worth mentioning? The disk controller?

Damer: Oh, the disk controller, of course. The before and after. Before we used to have to do this…

Laughton: Cassette tapes.

Damer: …and then after we get to a disk.

Laughton: This controller, this is what I call the famous disk controller with seven integrated circuits. Disk controllers at the time had probably over 100 circuits on them. What those 100 circuits were doing is now being done by Wozniak in software. One of these, this one here, is a state machine. Basically it did state transitions. You would put in a state and it would give you as output another state. His code would step through the state machine. It was genius, just genius.

Damer: In fact, the actual state machine itself is described in the documents. There it is. I'm fortunate though because I'm on the right page. So there we go.

Laughton: Yeah, the state machine was a 256-byte ROM. You would put an address in and get some data out. This was his state machine. This was what was in those 256 bytes. It was quite amazing. I still don't know how he came up with the idea.

Damer: Now here's something with your handwriting, and some of Woz's handwriting.

Laughton: This page was my handwritten instructions that I got verbally from Woz on how to do a boot. When you put the disk in and wanted to boot, this is the steps I needed to go through to boot up. Very interesting. I don't quite understand my notes now. I'm sure I did at the time.

Damer: This is Woz's handwriting on the right section?

Laughton: I guess this is just a short map of the code in his read/write track/sector; the various portions of it. There's read, there's write. Write a sector, read a sector. This is getting the address of a sector. You've got this disk spinning and you want to know the address of what's under the head. This gives you that address. This is read/write track/sector. The disk drive had stepper motors and this routine here controlled the stepper motors. Usually this was done in hardware. He did it in software. He was actually driving the stepper motors step by step by step. A stepper motor, you don't just drive it. You have to accelerate it because there's momentum involved. So you start off and you accelerate it until it gets to the location you want. This was all in this little routine.

Damer: These are the actual opcodes.

Laughton: This is just part of the thing. He has a full listing in here done by hand. This is a block diagram, essentially, of the thing that's on his card. These are the seven ICs, and how they are wired together. I guess you call it a schematic. Very simple, very elegant.

Damer: Bill Fernandez—I wrote to him about this. He had created this one.

Laughton: Was Bill Fernandez the hardware engineer?

Damer: Yeah. Here's another piece from him that was in your collection. A schematic from December '77! So this is before Woz produced the floppy drive, but I'm not quite sure what these refer to.

Laughton: I'd like to say a word or two about Bill, because he was an analog engineer who was just as much of a genius as Wozniak. If you open up a disk drive, inside there is a board, and there are ICs on the board. If you look at the standard disk drive of the era, it was just chock full of stuff: resistors and transistors and capacitors and transistors. It was just end-to-end filled. Fernandez did the analog part that went on the drive. If you look at it, there's almost nothing there. It's sparsely populated. It's as sparsely populated as Woz's control card. In my mind it took a genius to create an analog circuit that could be done in so few parts. Of course what this did for Apple was: everybody else, when they made a disk drive with a controller, had to spend a huge amount of money to get these things into production. Apple could do it very, very cheaply, and sell it for the same thing that they [the others] were selling it for. So it was a huge profit margin for Apple. Just a genius.

When I was working I told you we were making disk drives for other 6502 machines. When I was working on the project I needed to convert plus five volts to minus five volts. I called several electronic engineers and asked them, and they recommended off-the-shelf parts, and they had very complex solution. I walked over and talked to Bill Fernandez, and he sat there at his desk with a cigarette hanging out of his mouth and scratched out a couple of CMOS ICs and some capacitors and a very, very simple circuit that did the job. Just off the top of his head. Amazing guy.

Damer: He would've worked with the head step timing you see here?

Laughton: I think the head step timing would have come from Wozniak. This is the stepper motor driver.

Damer: Here's Woz's more official sector format.

Laughton: The sectors on the disk are not laid out in sequence. Sector two does not follow sector one. They are laid out because there's a latency between the time you read a sector and the time you want to read the next one. It might be sector one, and then three sectors later, sector two, so that you can do continuous reading. If you laid them out one after the other, you have to wait a whole revolution to get to it. So he developed a sector map—the way the sectors on the disk should be laid out—for the fastest reading.

Damer: Interesting. So not only efficient chip design, but very efficient software operations.

Laughton: Yes. And this is a problem faced by file management systems today. Laying out the disk, and how are we going to do it. As you get faster disks, you have to change it.

Damer: There's a couple of other things here. It's all these crazy notes. Are these Woz's?

Laughton: Yeah, these would be Woz's. What it looks like here is he is essentially brainstorming how to write the code for the read/write track/sector [routine]. So these are his thoughts verbalized on paper, which later all got put together in this listing right here.

Damer: This is what Wigginton and Woz wrote by hand?

Laughton: Yeah. This is the full read/write track/sector after it was all developed and tested. They had taken the opcodes that they generated by hand and entered them into the machine, and actually got the read/write track/sector working. This is how they delivered to me the code for read/write track/sector. I later put these in punch cards and put them into the DOS. Amazing stuff. A lot of memories here. Here's the note, "Must not cross page boundary."

Damer: I'll think of maybe one or two other things and then we'll look through. If you see something jumping out. This is more of the read/write sector DOS boot.

Laughton: Can I see the DOS boot?

Damer: DOS boot PROM [Programmable Read-Only Memory].

Laughton: One of the circuits here is a PROM read-only memory that has 6502 code in it. In the boot-up process you would read that code into the 6502 and then that had the read/write track/sector enough to boot the DOS. So the bootstrap process was really a bootstrap process. You would end up reading some code off of here which would read it in, and then that code would read in the boot sectors off of the disk. We call it bootstrap because it's lifting yourself up by your bootstraps.

Damer: So that's the bootstrap. That's the device address assignment. Track assignments. This looks much more official here, and Woz's handwritten "Company Confidential."

Laughton: Yes, as he was handing me the documents, he was writing "Company Confidential" on them. I guess he had talked to a lawyer.

These cards, when you plugged them in, they would go into an address space in the 6502. Those address spaces could either contain memory, or they could contain electrical signals for things. This [code] gives the electrical signals the addresses, and the electrical signals that control this card. So if you wanted to move the head, you'd send out a number to this address. If you were going through the state machine, these are the addresses that you would read or write to.

Damer: We looked at the state machine. How about the pin assignments. None of this changed during the development?

Laughton: No. Well, actually these pin assignments are the pin assignment of the state machine, the 256 byte ROM. These are the various pin numbers on the chip and what they were used for.

Damer: That's it for our state machines and technical documents. The next document that certainly is fascinating, is this letter dated April 10, 1978 and addressed "Dear Steve," to Steve Jobs. It seems to be the contract for the DOS system. He signs it twice here, and to me it's almost like he tried to sign with a fountain pen there and it made a mess, so he signed there again with a ballpoint pen. I don't know, but he was into calligraphy. But this is obviously an old photocopy. How did you come to have these Shep contractual docs?

Laughton: Well, usually, like I was saying, I would create the content and then Shep would type them up—actually his wife would type them up—and then once they were signed, he'd give me a copy. So this is the original contract.

Damer: That's the contract.

Laughton: Note that this says "will not have a sysgen". Sysgen was an IBM term. It was [the process] used for generating the system. My anticipation was that a sysgen would be needed, but it was not included in this. Also, I was saying I was going to have a backup—a way to back up a disk—a way of recovering a disk, and a file copy routine. This was all in that original contract, none of which I delivered.

Damer: But they paid you anyway?

Laughton: They paid me anyway. I actually did a lot more than what was called for, but none of those utilities were delivered. The file manager was delivered, the BASIC interface was delivered, and none of the other stuff.

Damer: In the rest of these letters, you've got the system contract of May 10 of '78, so that's about midway through.

Laughton: I recognize the words, but I don't think this produced anything like what was being said here. The functions got created, but they didn't get created in the way specified here. This calls for a boot, and a format of a disk. It all got done; it just didn't get done the way it's specified here. This looks like another contract. This was a very vague contract. "Paul is going to make the changes agreed upon". <laughs> "Price $500." Nothing specifying what they are!

Damer: It might've been in some later document handwritten by you.

Laughton: There's notes of meeting minutes. This is a document that was produced as a result of them saying that I would fix certain things. These are the things that would be fixed.

Damer: That is your handwriting?

Laughton: Yeah, this is my handwriting. And the time it would take. So maybe this is the "agreed-upon" stuff for $500.

Damer: Informal in those days.

Laughton: Yeah. Everything was just done. It was mostly gentlemen's agreement. Everybody was moving fast and nobody had time. If we had had lawyers involved, it wouldn't' have gotten done.

Damer: Which is what John von Neumann told people at the Institute for Advanced Study. He said something like "I don't want any lawyers involved in building the machine. We'll spend more time and money on them than building the machine. Therefore we're going to give away our plans."

Laughton: I remember trying to ship products at Logitech—things like digital cameras—and enormous amount of time spent with lawyers. Things that had to be… warnings about inserting batteries. The design made it impossible to insert the batteries wrong, but their lawyers insisted.

<crew talk>

Laughton: Here we have—this says "DOS 3.3 system master." When you booted DOS it would relocate in memory. You could then create a master with it already relocated, so it didn't have to relocate. So

calling it a "system master" would be a relocated version. If you notice over here there's a little piece of tape around here. That's "write protect;" if I put this in and try to write on it, it won't write. It would be nice if this was an actual machine and we could watch it boot up.

Damer: We can if you want.

Laughton: Oh really? Can we do that?

Damer: Yeah, I got it all. Where's the power?

<crew talk>

<break in recording>

Damer: This one seems to be lighting up. Wait a minute. Here we go. There's that sound.

Laughton: That sound you hear is the stepper motor pulling the head all the way back so it knows where it is. That head could've been left in any position, so it's pulling it from the farthest position back. Then what you're hearing is it's beating its head against the wall, but it ends up knowing exactly where it is.

Damer: Interesting.

Laughton: We're all booted up and we can type in CAT. It says syntax error.

Damer: Oh CATALOG. You need to write the whole thing out.

Laughton: Where's the backspace? A lot of stuff on there!

Damer: There are some BASIC programs there.

Laughton: Looks like Applesoft [BASIC] is there, floating point basic, integer basic, a renumber program.

Damer: Might be a game in there too somewhere.

Laughton: Very good. I remember that old 40-column screen. That brings back a lot of memories. I don't know how many man-hours I spent on a keyboard working on those things.

Damer: Great. I'm really glad we got to boot it up.

<crew talk>

<break in recording>

Code Review of the DOS Relocation Phase

Shustek: [For the relocation phase, where is the code] to find out how long a particular opcode is? Is that in the ROM somewhere? Is that F88E?

Laughton: Well, if we want to know where it is, we can go to this thing at the back.

Shustek: I think it's defined as a constant. I think saw it actually—INSDS2. I'm assuming that that's some address in the Apple II ROM. So was there some routine in the Apple II ROM that knew about 6502 opcodes, and based on what was the opcode in the A register would tell you, probably stored down in location 2F, how long that instruction is? In order to do the relocation you have to know a lot about the opcode structure of the 6502—to know how long instructions are—and I didn't see anything in here that did that.

Laughton: Somewhere in here I thought I had a comment saying Apple code, but I'm assuming you're right.

Shustek: Why would that have been a routine in the ROM?

Laughton: There was a lot of the little stuff in the ROM that Woz put there. Sweet 16 was in the ROM.

Shustek: Obviously this is fragile, because if a new version of the ROM comes out, that changes and this code is broken.

Laughton: Well, there wouldn't be a new version of ROM for any reason.

Shustek: Okay. It's ROM.

Laughton: Woz did write things right the first time. There is a madness to the value of an operator—a method. On the 6502 operators you could look at them and classify them into things that they did.

Shustek: But I don't see any of that code in here.

Laughton: No, you'd have to go to the Apple II ROM.

Shustek: Right. So you're assuming that each of these was one block of just pure code, with no data, no address constants, stuff like that.

Laughton: Right. No, there were also those jump tables.

Shustek: I guess that's what this is, right? This section up here is relocating address tables. And then this is relocating the code. These are lists of the address ranges that have to be relocated, and this is a list of the address code ranges that have to be relocated.

Laughton: I guess I built in some space—

Shustek: —for additional ones. Other than that mysterious call, this all makes sense. You're looking to see if it's a range that needs to be relocated, otherwise you ignore it. You look to see if it's an opcode that is too small to have an address in it in, which case you don't have to do anything. Then you look for the end of the code in the segment, and when you get to the end of the segment you go to the next segment. So you're stepping down this list. It's clever code.

Laughton: Some of these things… I was talking about different opcodes. I think opcodes like DB are not official.

Shustek: DB is not an opcode; it's just an assembler pseudo-op.

Laughton: No, they're directives, but I don't think it's a 6502 directive, an official one.

Shustek: Right, it could be. But I think most of the instruction opcodes are the same as the standard ones.

Laughton: Yeah, they are.

Shustek: It looks like you may be going through this table backwards, right? Comparing to the beginning of the table to see if it's done. Let's see where you started.

Laughton: This is adding…

Shustek: It's starting at code table plus one comma X, and X starts out as zero. But then why is this comparing to the beginning of the table? This was a little confusing. What's that DB six times four?

Laughton: I think it's the size. The value at CDETAB is the size of the table. Does that make sense?

Shustek: Yes, okay. I see what you're saying.

Laughton: There's six elements in the table and they're four bytes long. So this is checking to see if I'm at that value.

Shustek: Got it, okay. So X starts at zero and goes to the value of the CDETAB in increments of four. Got it. I thought CDETAB was the address.

Laughton: CDETAB is an address, I think. CDETAB plus one. CDETAB is located at this address, and CDETAB would be this byte here, and then plus one takes you to the next place.

Shustek: Oh, I get it. CDETAB is the address of this byte, which contains the size of the table.

Laughton: Right.

Damer: Got it. So this is not compare immediate. This is a compare with 1B.

Laughton: And then up here—

Shustek: You start at the next location.

Laughton: Yeah.

Shustek: Right, because that's the beginning of the first four-byte quantity. All right, I get it. That's interesting. Were there no compare immediate instructions?

Laughton: Compare immediate?

Shustek: Where rather than having the address of a byte that contains the value to be compared, the value to compare is in the instruction. I don't know, maybe not.

Laughton: I think… I'm trying to remember. If you had values in zero page you had such things that you could do. You understand zero page in 6502?

Shustek: Yeah.

Laughton: I think you could do compare immediates sometimes.

Shustek: It's a pretty strange instruction set, but it was fun. You could do a lot with very little.

Laughton: These branch relatives were nice too. These are the vector tables, and you had "load," "save," "run." So pick these out, and then do a JSR or RTS.

Shustek: Because there was no indirect jump.

Laughton: Yeah. I don't know what that's about.

Shustek: It's a typo, I think. You're trying to get to here. It's INIT2A versus INITA2. I fixed that.

Damer: Thanks, Len.

Laughton: This is essentially getting a command. A lot of work just to read the keyboard.

Shustek: I'm surprised there weren't routines in ROM that you could have used for that, that were already there for other purposes.

Laughton: Well, part of it was I think because we were going through BASIC. I don't know. The bug was to determine whether BASIC was in execution mode or not.

Damer: How big was the total code when you delivered it? Do you know? It wouldn't run in a 4K Apple II. It needed like an 8K.

Laughton: I think Apple II only came in 16K chunks. It was within the 16—

Damer: So it's just really the Volume Table of Contents [VTOC] search where you're doing the file name comparison.

Laughton: It looks like deleted files—

Shustek: Are just marked as deleted. You didn't reorganize the Volume Table of Contents, which makes sense. Just mark it as a deleted entry. That's the way most file systems work. You can't reorganize the directory each time.

Laughton: The VTOC and the sector allocation stuff was what I would call on a central cylinder. In my IBM days you had the cylinders in a disk, and you wanted to put that which is going to be accessed a lot in the central cylinder. I probably carried that over and it was in an inner ring. Then as I recall, sectors were allocated in and out from that point, which is the way we did it in IBM.

Shustek: It looks like you keep track of—pardon the pun—you keep track of a track that is entirely composed of free sectors, so this frees the sector and then this frees a track of sectors. I'm a little bit confused by that. It has two levels. Why would you bother to keep track of tracks at all?

Laughton: The tracks being the cylinders.

Shustek: Right, right.

Laughton: I think I was trying to manage the allocation of, like I say, getting a new track to use it to the left or the right of the VTOC.

Shustek: Okay, so here's the VTOC for the volume. Things like DOS type, and where the directory sector is. DOS release number, volume number, maximum sectors in a file directory. Allocation algorithm bytes. Number of tracks on the volume, number of sectors per track, number of bytes per sector. That's

fairly configurable. Sector allocation area. Sectors allocated by bitmap, four bytes of bits per track. So there were 16 bits saying which of those sectors on the track were empty?

Laughton: Yeah, bits were on if they were available.

Shustek: Bits were on if they were available. Okay.

Damer: Did you ever think of doing a hierarchical directory structure or this is just all flat? I know that got added in the ‘80s in MS-DOS and other operating systems like that.

Laughton: Hierarchical directory. I’m not sure what you mean.

Damer: Like subdirectories, where a file can be a directory of other files like subdirectories.

Laughton: No, no.

Damer: Too soon, too early, too complicated?

Laughton: Yup. Not enough disk space to even think about.

END OF INTERVIEW

