

college edition

# chuck

the  
**down & dirty**  
of  
personal  
computer  
evolution



tales of a coder 1982 - 2019

# **Chuck**

The down and dirty of personal computer evolution

Written by  
Charles Bowen

Cjbowen.com

Chuckthebook.com

Revision 1.2

Copyright © 2019

## Table of Contents

Chapter 1	End of the Stone Age	Page 4
Chapter 2	A Coder is Born	Page 15
Chapter 3	This is LANtastic	Page 25
Chapter 4	Growing Pains	Page 35
Chapter 5	And then there was HTML	Page 48
Chapter 6	Entertaining Bandwidth	Page 59
Chapter 7	Cloudy days forever	Page 77

Text in **blue** corresponds with images. Text in **red** are in reference to Wikipedia articles regarding the subject at hand.

My story is the history of the personal computer as seen through the eyes of a coder 1982 – 2019. My experiences will vary from others who rode this incredible journey through the growth of PC technology. I honor all who have dedicated their lives to the cause of technology and our future.

I dedicate this book to my daughter Mary, for all I do is for her.

I thank my sister Charlene Matthews and Andrea Macon for surviving the editing of my book.

## Chapter 1

### End of the Stone Age

The year is 1982. Michael Jackson's "**Thriller**" sells 20 million albums to become the largest selling record ever. Ozzy bites the head off a live bat thrown at him at a January 20th performance. The Vietnam Memorial is erected in Washington D.C. Italy wins the Soccer World Cup in Spain, and **John Belushi** dies March 5th from cocaine and heroin. Fast Times at Ridgemont High, Blade Runner, E.T. and Poltergeist were box office hits.

Modern day historians say planet Earth has been around for about 4 billion years, one third the age of the Universe. They also have evidence of human civilization dating back about six thousand years, when suddenly humans begin building huge stone temples, cities, ships, statues and other arts. All this technology basically levels off until the turn of the Twentieth Century, the year 1900, at the start of the **industrial age**.

Within 120 years, modern-day technology has been developed. Cars, electricity, telephones, airplanes, high rise buildings, high powered weapons, microchips, and cellular/wireless technologies. Rapid enhancements of these technologies are happening faster than we can grasp, blending into our day-to-day lives unnoticed by many.

The personal computer has been available to the general public for about forty years, the Internet and cellular telephones for about twenty. The latest generation, let's say people in their twenties, are the first to use these technologies their entire lives and don't realize how new this all is to the world.

Advancements in technology are accelerating at such speeds, we can clearly determine that today, we are in the infant stages of the next generation of technology beyond the personal computer.

My story starts as a 20-year-old college-kid born and raised in Los Angeles California. If you could stand me up next to college kids today, I wouldn't look out of place, yet my life was so different without today's technology. I was a police explorer in high school and had planned on spending the rest of my life, fighting crime and the whole college thing seemed much too boring. School was tough for me because I was dyslectic which made it hard to read or learn math. These types of disabilities were not identifiable at the time and kids like me, just dealt with our shortcomings. Later in life I'm tested with an above average IQ that I contribute to the thought process required to overcome my disability.



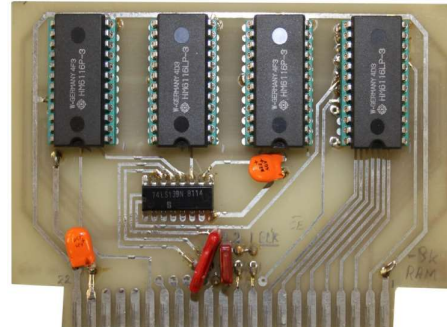
One day, while standing in the front lobby of my father's office building, I gazed is my future. A small office desk that hosts a grey colored, 12" black and white television sitting on a grey, plastic, molded box the size of a kitchen cabinet drawer. There were air slots on the front face and in front of the box sits a keyboard the size of a shoe box.

Sitting next to one side of the TV were two boxes with large slots and to the other side, a wide cartridge printer loaded with green and white lined paper. The Tandy TRS-80 personal computer. The best computer you could buy. Well, the only computer you could buy in 1982.

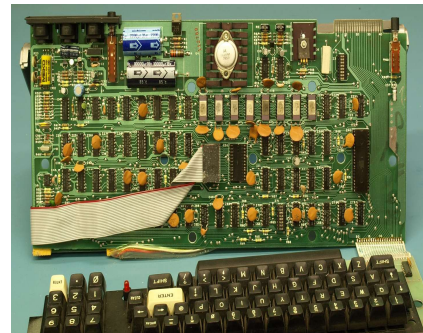




This \$1000 business machine was equipped with an [8K ram motherboard](#) and 2 5 ¼" external floppy disks, each with 84k of memory and access to a printer. The ability to type letters without having to use a typewriter and the automatic tallying of accounting ledger columns was worth every penny of its cost.



The TRS-80 had no hard drive, no built-in modem, nor a network card because none of these devices had been invented yet. Software was available on 5 ¼" floppy sets, usually requiring users to load multiple floppies as applications loaded into memory. Users would use the other floppy disk drive, if you had a second one, to store spreadsheets and word processing documents.



If you're thinking that this first personal computer was anything less than amazing, then you must don't see 1982 in its true perspective. There were no cell phones and [dial telephones](#) were wired to wall jacks, mobile only by the length of their cord from the wall they were permanently anchored.



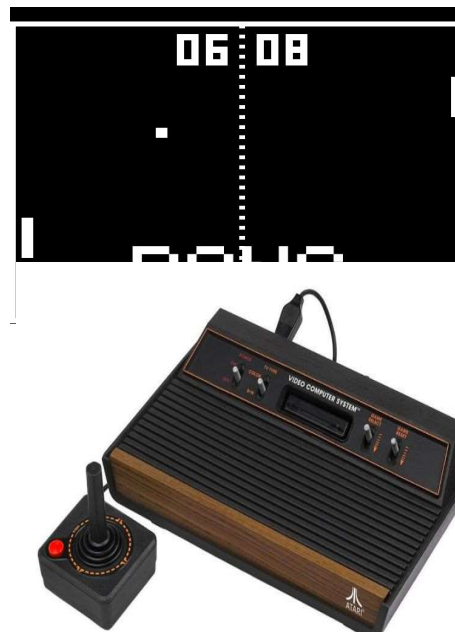
There was no texting. If you wanted to gather your friends to meet after work for a few drinks, you had to call their telephones numbers and leave messages on their answering machines that they might listen to when they got home. Mailing letters and notes left on car windshields were common.

Even better, you could page them on their “beepers” with a single numeric line of information. Everyone that was anyone had a beeper, sometimes multiple beepers. These little, battery operated devices, half the size of a pack of cigarettes, would allow people to call an assigned telephone number paired to your beeper and leave a numeric message that displayed on the top of the device. Work or home would leave predetermined codes, such as “911”, meaning to call NOW! Oh, and being able to call 911 was new to us and remembering important telephone numbers could be life-saving. If you forgot a telephone number, you could call 411 and a live person would help you. If you needed assistance you could call the operator by dialing ‘0’.



Beeper, or pagers, were a great enhancement to the business world but an annoyance to the free willed. My brother was known to have thrown several out the window of his car or into half full pitchers of beer as his wife made attempts to keep tabs on him. Someone could page you over and over as many times as they wished and the device would make an annoying “beep, beep, beep” noise until you pressed the acknowledge button.

There was no Internet, no Facebook, Twitter, Email, online shopping or porn. No personal computers, notebooks or tablets. NOTHING!!! No video games but we had [pong](#) that was played on the television attached to an [Atari](#) gaming console. Gaming [arcades](#) were everywhere. A high score or being deemed the neighborhood [pinball](#) wizard was every kids goal in life. If you weren't physically standing next to someone there was a social disconnect. You never missed upcoming gatherings, for doing so would remove you from the social loop.







It gets worse! No digital cameras. In fact, there was nothing digital. No music CD's or movie DVDs. No Netflix or Amazon Prime. If you wanted a picture for your memories, you had to use a [35mm film](#) roll camera which required using a developing service to make your prints that took days. The [Polaroid](#) camera with its instant prints, was around but it was too expensive for everyday use, plus the prints faded away after time.



There were no iPads or music sites. If you wanted to listen to music, you had the radio, albums, [8 track](#) or [cassette tapes](#). Cassette tapes were recordable, so everyone made their favorite song mixes. There was always a live concert, which you never missed if your favorite band was in town. You had to rely on the daily newspaper for news, movie times, shopping specials and upcoming concerts. Everyone looked forward to the Sunday newspaper with its entertainment guides, cartoons and extras.



Cable television was fresh to the entertainment market and slow to make its way across America. A coaxial cable network needed to be installed down every street and to every home. There were no Satellites or wireless networks. Cable was great and [MTV](#) was in its heyday, yet the "[pet rock](#)" got more attention during dinner table conversation.



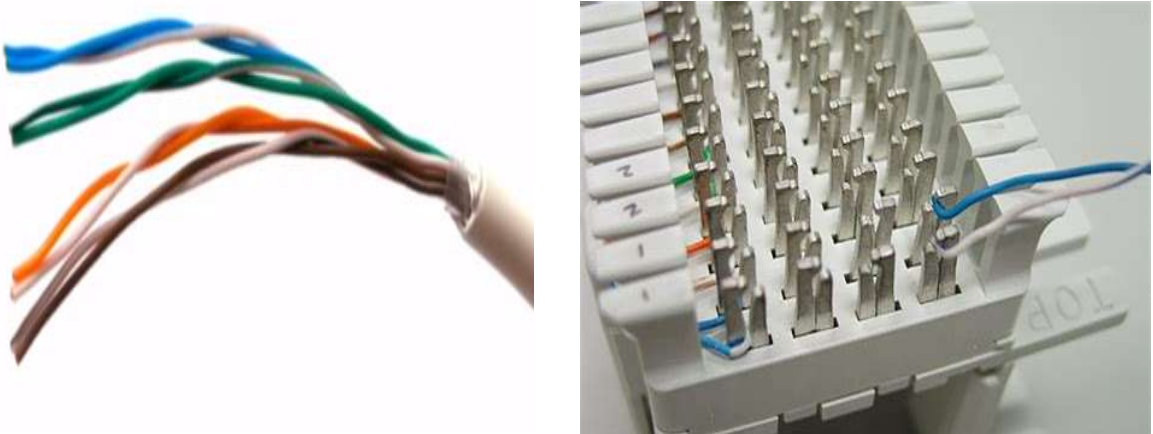
If you wanted to type a letter, you used a typewriter. In high school and college, you could take typing classes. There was a technique for typing fast by placing your fingers on selected keys of the typewriter and learning all the keys around each finger position. After lots of practice, you would be able to type without ever looking at the keyboard. If you made a mistake, you would have to use [whiteout](#) to erase the error, then back space the type head over the whiteout and type the correct text or start your document over from the beginning.



Before the PC, the world had mainframe computers, such as the [IBM AS400](#). These systems and associated software applications cost in the hundreds of thousands of dollars and incurred expensive networking costs utilizing dedicated data lines through the telephone companies. The AS400 was a metal box about the size of a desk. The box contained its brain, data storage, software, and a networking interface. All user instances were hosted within the box and user [workstations](#) were [dumb terminals](#) that emulated the user experience hosted by the mainframe computer. Workstations consisted of a monitor, keyboard, and modem.



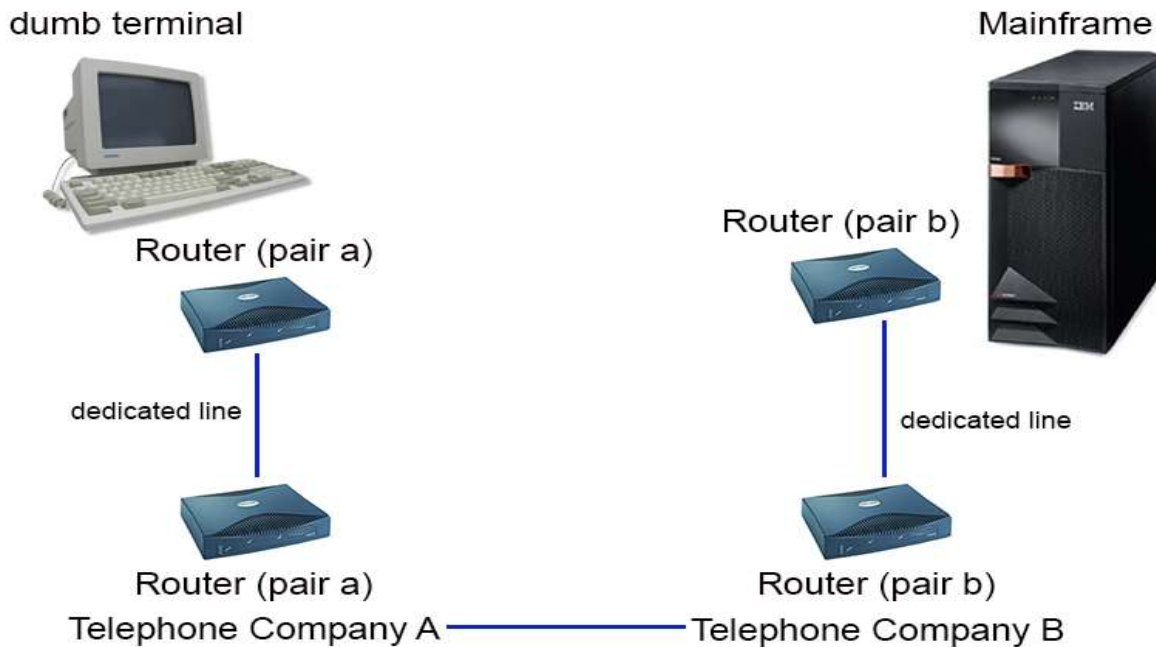
Most if not all workstations were connected to the mainframe through a WAN (**wide area network**), unlike the **local area networks** available today. A workstation's modem would connect to a wall jack using 2 pair telephone wire. The wire connected to the wall jack would run through the building to a telephone jack terminal, usually located in the building's telephone room. The telephone jack terminal would be connected to a data router via a data cable. The data router would be connected to a dedicated data line that would connect to its paired data router located at the local telephone company.



The telephone company would then bounce the data across its network until the data found the telephone number associated with data router located at the local telephone company nearest the location of the mainframe computer. That data router would be connected to another dedicated data line that would connect to its paired data router in the building hosting the mainframe computer. The data router in the building would then be connected to the network interface of the mainframe computer. The monthly cost of these dedicated data lines would be in the thousands of dollars, plus the cost of moving data. [WAN-before-interne diagram](#)



## Wide Area Network before the Internet



Software was limited to data entry and search applications, which is still the core power of computers, data collection, and analysis. As a workstation's modem is turned on, a data string is sent through the telephone line to the building's router. The router sends the data to the local telephone company, which bounces the data across the telephone company's network until it reaches the router of the mainframe computer.

The mainframe accepts the data and determines what to do with it, in this case, it authenticates the modem's ID and sends back a data string for a login screen. The workstation receives the data string and displaying a screen. A lot like how **HTML** pages are loaded into browsers one line after another. The workstation user then enters their login information and sends that data back to the mainframe. The mainframe authenticates the user and creates a user session, returning to the workstation a data string for a menu screen. The workstation then sends requests, which the mainframe returns either data forms or data lists.

Technical advancements, such as the fax or [teletype](#), weren't used extensively outside large corporations and government due to expense of equipment and data fees.



My brother's friend worked as an assistant for my dad's company and the car he used during work had a [car phone](#). Car phones were a very expensive technology prior to cellular technology used by business executives or the rich. While working, he used the car phone as he drove around Los Angeles to talk to his girlfriend. Not realizing he had spent 40 minutes on the device, he ran up a two-hundred-dollar data bill and lost his job.

Back to the start of my story, standing in front of that [Tandy computer](#), I watched it think for itself without a lifeline connecting it to a computer system somewhere else, was amazing. The phrase "An entire mainframe the size of a room, all in the one little box", was true. I had no idea what was to come, that there would be an Internet, digital media, or any of the other future advances. Every new technology thereafter would hit me by surprise without warning. Everyone would say, "What will they come up with next!", over, and over again.



I traded a revolver (gun) for a friend's [Commodore 64](#) gaming console. You could write small code routines, I believe in Basic, for stupid things like calculating sales tax for a dollar amount of a purchased item. Its hard to explain why that was so amazing to me, see, before then the best we had were [adding machines](#), or pencil and paper.



## Chapter 2

### A Coder is Born

The year is 1985. “**We are the World**” is recorded by USA for Africa, LIVE AID in London and Philadelphia, beamed around the world, Crack cocaine starts to appear. The Rock 'n' Roll Hall of fame is opened, New Coke is introduced in April and quickly replaced with original Coke. The hole in the ozone layer, first detected in 1977, is now indisputable. Back to the Future, The Goonies, The Breakfast Club, The Color Purple, Fright Night and Weird Science were box office hits. Dire Straits “Money for Nothing” is the best rock song of 1985.



I'm now 23 and a **policeman** working at a state college in Orange County, California. Since I hated writing parking tickets, I would spend most of my free time in the college's **computer lab**, hanging out with the lab technicians with the goal of learning everything I could. The lab consisted of a dozen or so **IBM 8086** machines and maybe twenty **Apple 2** computers that were networked, well ahead of the IBM world. I became friends with one of the technicians, Roger Cossaboom, who became a lifelong friend.





Roger was finishing his degree in computer sciences at Ca. State Fullerton and was responsible for maintaining the systems in the computer lab. He later became a guru of PC technology and by my side throughout all my software ventures for thirty years. It was like this: Every time I would get stuck on a project, Roger would get a phone call, “Hey Roger, what’s up?” He always knew what was next, and always gave me the best advice, writing a lot of code along the way, without asking a penny, ever.

If you wanted to write code for the personal computer (IBM), your choices in programming language were BASIC, C, (programming language), or **dbase**. Most coders programed in basic but dbase, which I chose to use, was very popular in the 1980s. All the experienced coders wrote in **Fortran** or **Cobol**, which were mainframe languages, a world controlled by IBM.

We coders lived as if we were the Starship Enterprise, boldly going where no man has gone before. There wasn’t any software available for personal computers beyond the basics, such as word processors and spreadsheet applications.

There was no end to the development of new applications, however the complexity of design for such applications were for stand-alone access on low powered machines. This problem quickly cured itself through the rapid growth of personal computer technology in both hardware and operating systems.

Speaking of operating systems, at this time all personal computers were driven by **DOS**, or by **Apple Inc. Windows** did not exist. There were no graphics or color monitors, so the personal computer experience had the look and feel of a dumb terminal.

Small and specialized police departments were not computerized at this time. Most had access to a PC or two for word processing and spreadsheets, but there were no pc-based police software systems on the market. I saw the need for law enforcement use and was authorized to write code for the police department in which I was employed.

During the first few weeks of each semester we were required to issue warning tickets for illegally parked vehicles rather than citations. Copies of such warning tickets were available in the police cars and we had to sift through the stack of warnings before issuing a citation. I wrote a simple input screen for the warning tickets and a routine that would print license plate indexed reports for the police cars. WOW! that got me a commendation and the desire to build an entire police records keeping system. This ultimately developed into a **computer aided dispatch** system.



The first code I wrote was on an IBM 8086 after hours in the computer lab. The computer had a **monochrome** monitor, no hard drive, 512k of ram, and two 5 1/4" floppy drives. When I would sit down to write code, I would have to load two different dBase II application disks into the first floppy drive. Disk one first, after a few minutes of grinding disk drive sounds disk two would be prompted to load. Once the dBase II application was successfully loading into the machine's ram, a dot prompt was displayed



for writing code. I would place a 5 1/4" disk in the "B" drive that would contain my code and tables. From the [dot prompt](#), I would type "modi comm b:\program.prg" and a text editor would display for the writing of code.



Code was written in text files that had a maximum 64k size limit. There was no indenting of code and every byte mattered.

Because floppy disks only had 512k of disk space, data table design to the smallest footprint was a must. You would never think about wasting a byte of data space, attempting to get as much functionality as possible onto a single disk. Check out this [sample code \(set 1\)](#).

Programming languages use IF and DO WHILE type of statements to move through data and business logic. In the 1980's and before Foxbase/DBase had access to SQL statements, creating a data list looked a lot like this:

```
Mlastname = "bowen"
Use customers.dbf index lastname
Seek Mlastname
If .not. eof()
Do while lastname = Mlastname

    ? lastname

If .not. eof()
    Skip 1
Else
    exit
Endif
Enddo
Endif
Close data
```

IBM was slow to introduce new technology so other companies created the [PC clone](#). All clone components were manufactured overseas, mostly in Taiwan. Small shops that built PC clones started popping up everywhere. They would purchase all the components in bulk then build machines to order. It didn't take long for the first [5-megabyte hard drive](#) to hit the street. I needed such a device for the police software project, I purchased my first PC. With a hard drive, I was able to install dBase III and all my code libraries onto a single fixed disk.



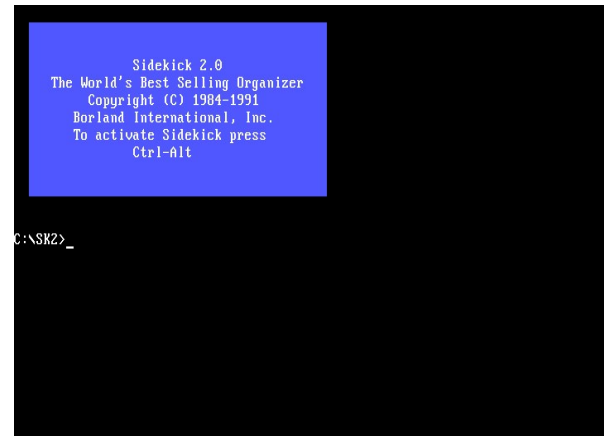
Within several months I was able to release the first version of my police software that included thousands of pages of code. Report generators had not been invented yet, I had to build reports line by line. I used [ASCII](#) codes to print lines and other basic symbols to [dot matrix](#) printers.



When I think back to 1986, I realize that programmers were programmers. Modern day programmers with studios, code libraries and graphics, what us old guys call "plug and play" programming, rely on pre-existing logic and code. We didn't even have copy-paste utilities, so we had to write code line by line and many times wrote the same routines over and over from scratch.



When [Sidekick](#), one the first robust text editors, became available, we were able to write code much faster. Because there were no code checkers, you would spend hours finding an “IF” statement that was out of place. This often required printing the code on continuous tracker fed paper and plotting out by hand all your “IF” and “DO WHILE” statements through non-indented pages of code to locate the mistake or missing end statement.



I would write code all night long, it was addicting. I was inventing not only new applications but a new way to manage police records that had never seen by most cops. This was all I thought about, day and night. Born was the first generation of the computer geek, and with a gun no less!

By the end of the first year, I had sold and installed my software to several of the college police departments in the local area. I was often visiting my clients with upgrades and training, plus having to work my regular job, so my installs needed to remain geographically close. There was no way to upgrade software versions via [modem](#) or the Internet. Upgrades were available via disk sets and often clumsy for the end-user to install. I had to sell my software cheap as small agencies who didn't have budget for new technology. I was young and inexperienced in business, and had no idea what I was doing, nor where I was going.



The biggest obstacle was that colleges had mainframes and per their contract with the vendor, usually IBM or [Digital Equipment Corporation](#) (DEC), the police departments were required to purchase the PC version of the vendor's equipment. IBM had the [OS/2](#) personal computer and DEC

had the [Rainbow](#) personal computer. Both operating systems required proprietary software and they could not run dBase, which was required to run my software.



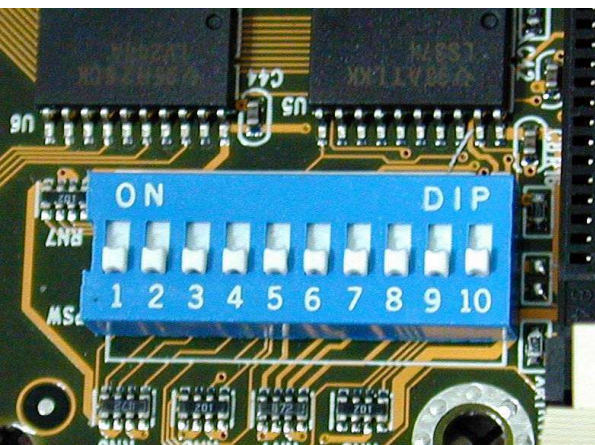
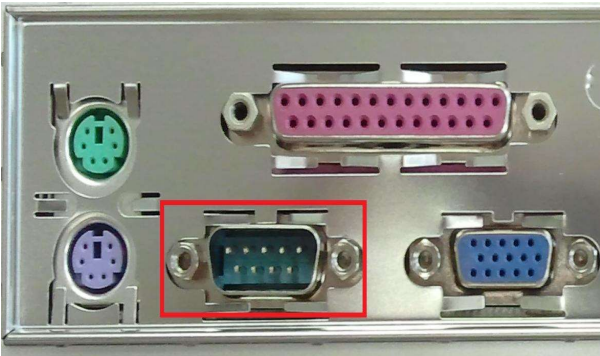
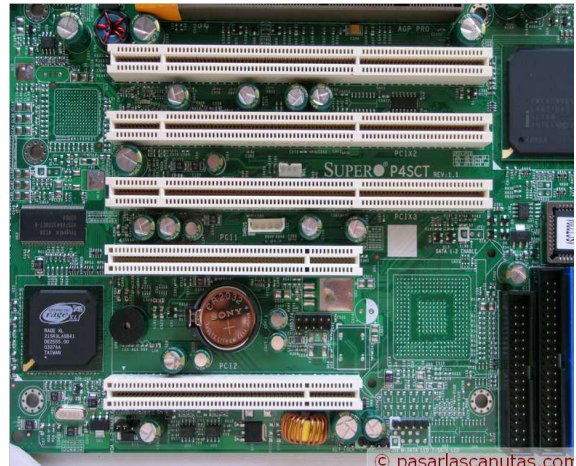
I have always felt that IBM made serious mistakes during this time period. By not using DOS as part of their attempt to force companies into their proprietary solutions. For years to come, the phrase “IBM compatible” was used less and less, until it was not used at all. Thus the phrase “PC clone” was born.

Clones were cheaper and compatible with all the software available for personal computers. However, the small clone builders were not established enough to capture government contracts. It was slow moving to convince government agencies to purchase “third party” clone machines. Eventually, though, the desire for my software would force

agencies to purchase the clones through third-party vendors, outside the control of their IT departments.

Clients often purchased the clones along with my software as a single purchase, this meant that I had to build relationships with clone computer builders and somewhat support the hardware. Hardware was not my forte. I had to learn as much as I could as technology changed, and there was no going to school for this. I knew more about personal computer hardware than most people on Earth yet had no idea what I was doing.

In 1985 personal computer **Motherboards** consisted of the **CPU**, ram, and **slots for peripherals**, unlike today's Motherboards that have most standard peripheral hardware sets built into the board. You needed to purchase and install a monitor, printer & **serial port**, and modem **device-cards** along with your personal computer box. Each peripheral card had **dip switches** that allowed you to set the card with a unique **interrupt request (IRQ)** address.





If two cards had the same IRQ address, the computer would fail to boot to the operating system. Hardware dip switches were a complete pain in the ass because you never knew what other peripheral IRQs were. In later years, operating systems, such as Windows, automatically assigned IRQ addresses to devices upon boot up and the dip switches were retired to a deep hole in space. If you happened to know enough to get a personal computer setup and bootable, you were the default expert others relied on to get through the daily routine of having a personal computer.

I used a computer builder in Anaheim, Ca, named MagicTek Computers, to supply my clients with hardware. William, the Asian owner had contacts overseas, as seen by his warehouse full of computer components. These guys are completely responsible for the launching of the personal computer and they are remembered by none. Every time we had the need to purchase a computer, we would pile into the car to visit William's shop. It was the toys-r-us of computer equipment.

#### PC Technical Snap-Shot – 1985

IBM and Apple go strong at releasing their first models of the personal computer. Limited to word processing and spread sheet applications, the cost of these new devices was prohibitive for home use.

The “hard drive” becomes available and allows applications for almost any purpose to become available, lots as [shareware](#). IBM compatible computers entered the market, soon to be called PC clones. Personal computers were considered business machines and education tools, and the ‘Computer Lab’ was born, found at every college in the country. Basically, typewriters without whiteout.

PC clone builders and software developers create an entire new industry made up of self-learned entrepreneurs.

Disk storage consisted of [floppy disks](#). 5 ¼” and 3 1/2” are now both available, with a maximum 720 byte capacity.



### Chapter 3

#### This is LANtastic

In the year 1986: January 28th, the Challenger explodes. The worst nuclear disaster ever occurs in **Chernobyl**, USSR. In April, Argentina wins Soccer World Cup in Mexico. Haley's Comet returns. America celebrates the national holiday of Dr. Martin Luther King Jr. day for the first time. The Statue of Liberty celebrates its 100th Anniversary and Bill Buckner lets the World Series squirt through his legs. The Transformers, Platoon, Aliens, Stand by Me, Top Gun are box office hits, and Howard The Duck is released. Bon Jovi "Livin' On A Prayer" is the best rock song of 1986.

My parents divorced when I was eight, in 1970. Of the five kids in my family, two chose to live with our dad, while two chose to live with our mom, including myself. The oldest, my brother Stan, just took off. Divorce was kind of unheard of back in 1970, we were the only family in South Pasadena with divorced parents, which totally sucked.

My dad refused to pay child support and my mom had been a housewife for eighteen years, just out of high school. She was forced into the job market for the first time in her life while supporting two children. We ended up living in South San Gabriel, a complete and total dumb, that taught me true survival. We often moved and I learned to be the loner, a trait almost necessary to be a career programmer. Fate was in the works for my future.

My mother became a dental office manager and married a dentist. Together they built and managed several dental offices that were perfect candidates for patient records and insurance billing software. They have me purchase a PC clone for each of the offices, then we evaluated demonstrations for the few dental office software systems available at the time. None of them were written by office managers, rather by dentists themselves, and did not meet the requirements for successful office management.

My mother asked if I would write a software system for their offices that would handle patient appointments, patient and insurance billing, and



office accounting. Another opportunity to write code and I was incredibly excited. To learn everything I needed to know about the business, which is necessary for writing great software, I quit the police department and worked for my parents full time as their general manager. One the side, I continued the police software development and installations.

The dental offices were large nine chair clinics that would require large data files for a pc-based system. The clones we purchased were probably 1 meg ram, 10 meg hard drive, monochrome monitor computers. I had to write code that would allow for fast data responses. If the software ran too slow, the staff would refuse to use the computers.

I remember solving problems overnight, so they didn't have a reason to stop using the computers the next day. Coders often got the best work done in the middle of the night, with no interruptions and full attention to the project at hand. Looking at a photograph of a current day gamer resembles the image of a 1986 programmer.

I built a patient table to handle 60,000 patients. A table this size for the time would have been very slow while fetching data. My solution was to create six tables pat001.dbf - pat006.dbf. Patient numbers 1 - 10000 would populate pat001.dbf, patient numbers 10001 - 20000 would populate pat002.dbf, so on through pat006.dbf.

Every time my code needed to access patient records, the staff would enter a patient number and I'd use a cluster of IF statements to route the request to the proper table. The difference in speed versus a single table was substantially faster. When searching for other fields, such as last and first names, I would have to search through all six tables to complete the request, which was slow but acceptable to the staff, as such a search through manual chart systems was impossible before computers.

The staff would enter patient treatments throughout the day that would tally accounting totals automatically and more importantly allow for quick insurance form printing that previously required a well-trained typist that understood the business rules for preparing medical insurance forms. Forms were completed faster, mailed without delay and resulted in much

faster payments from the insurance companies. Lower error rates prevented delays in payments as well.

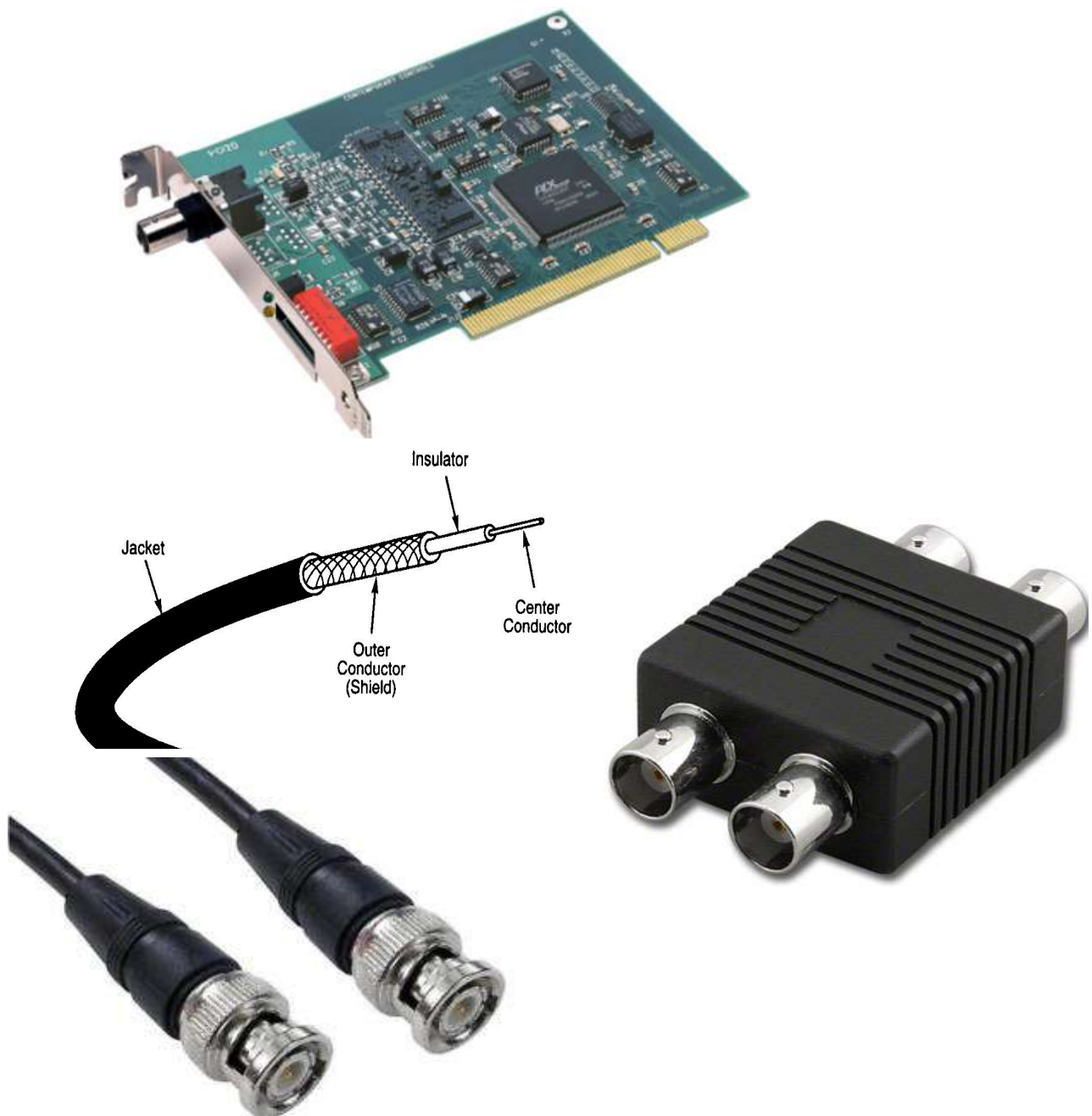
My software enabled management to audit patient payments eliminating front office staff theft. In fact, one of the office managers was so against installing the new computers, which would eliminate the manual accounting system, we suspected she was stealing money. Over a long weekend, we audited all the patient records against the manual payment ledgers and discovered over \$50,000 in theft.

One major problem we encountered, which was a deal killer if I couldn't solve it, was that one computer wasn't enough for three employees. There was no way to link multiple computers together, but chatter among friends lead me to believe such technology did exist but wasn't available to the general public.

I had recently met a computer consultant, Frank Goodyear, at the college I worked as a cop, who was experienced and knowledgeable in personal computer hardware, local area networking and was the "Shareware" king. Frank had an interest in what I was doing and invited me to share office space with him for my police software business. I learned a lot from Goodyear and I'll always be in his debt, along with Roger, for my success as a software developer.

Frank's suggestion was that the dental offices purchase two additional computers for each office, then use a newly released networking application called **Lantastic** to create a LAN or local area network, that should handle the requirements at hand. I took Frank's advice and a few weeks later installed my first LAN, again one of the first ever in the world, and again, I had no clue what I was doing.

Each computer was equipped with a [network card](#), and yes with a unique IRQ address. [Coaxial cable](#), I believe it was rg58, was used to connect the computers together via a non-powered [4 port hub](#). The cable ends were [A&B connectors](#) that twist and locked into place on the network cards and hub. Coaxial cable was single shielded copper wire. The shielding was made of metal mesh and connecting the A&B connectors to the cable ends was not only complicated but not very reliable, because the cables would easily pull out from the connectors and cause network failures.



The host version of the Lantastic software was installed on the computer designated to be the file server/workstation combination. The other computers were installed with the workstation version of Lantastic. The software included the necessary network card drivers and the host version used the computer's c drive (hard drive) for file sharing.

After several days of trial and error, reinstalling everything several times, reconnecting A&B connectors, trying everything possible to get the LAN to work, it suddenly connected, and the LAN was online. I could see the computer monitor of one of the workstations list files from the host computer and I stood up tall. A glow of light must have shot up from my head to the heavens and with an ear-to-ear grin I said "Hell Ya!" Who needed sex, marriage, children, homes or cars when you can make a Lantastic LAN work!

This was a great solution for the dental office, but not for government. These new technologies were unfamiliar to the IT departments and were unable to install or support, so they would not allow such purchases.

What made **dbase** so powerful was the ease to write, bundled commands plus native data tables. Native data tables allowed us dbase programmers to work with data from within our own code. There was no need for handshaking with outside tables, such as **SQL**. When networking became possible, we had to learn to record\file lock data, so multiple user access would not crash the tables. In 1987, dbase and soon to be FoxBASE, did not have auto-record locking, we had to do it by hand. Check out this [sample code \(set 2\)](#) illustrating record locking.

In later versions of FoxBASE and Foxpro, record locking became automatic, but until then, it was an art of code to prevent users from being blocked from updating data due to others accessing the data at the same time.

**Code accessing data before multi-user environments:**

```
Mid=12
Use customers.dbf
Locate for id = Mid
If .not. eof()
    Replace lastname with "bowen"
Endif
Close data
```

**Code accessing data in the multi-user environment:**

```
Mid=12
Use customers.dbf
Locate for id = Mid
If .not. eof()

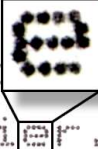
    Do while .t.
        If rlock()
            Replace lastname with "bowen"
            Unlock
            exit
        Else
            @ 01,01 say "Stand by, attempting to lock database....."
        Endif
    Enddo

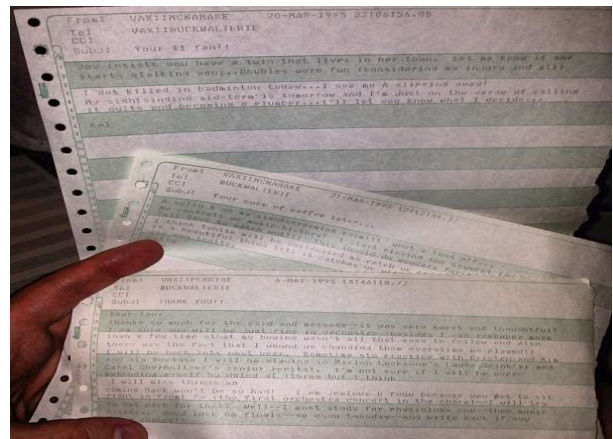
Endif
Close data
```

There wasn't an Internet yet, and desktop modems weren't fast enough, nor available to the general public. There was no electronic moving of data or email. This meant that insurance forms, collection letters, and other correspondences had to be printed and mailed. Even though the first generation of laser too expensive and not easily interfaced. The only option for printing from personal computers was the [dot matrix](#) printer.

Printing was limited to one font, the ugly square **computer font**, and **continuous feed** paper. There were no print spoolers or software that built printable forms from objects like in today's world. You loaded continuous feed paper, setting the top of the first page to the physical position on the printer that moved the print head from side to side. Your code would send a print command, then data, line by line to the printer. The print head would move from one side to the other as a metal head would imprint text through an ink ribbon onto the paper. After each line of text, your code would send a carriage return command to move to the next line.



system where a  could allow us to get commercial supplier.



Creating reports was a lot like building an **HTML** page. You start at the top and work line by line down the page. Below is a very simple example of the code needed to send data to a printer;

Set printer on

? "-----"

? "this is a line of text that goes between the lines"

? '-----'

Eject

Set printer off



Creating boxes around text to build forms was a true art of code, which was very time consuming and tedious. To be successful in software development, you really had to dedicate yourself to the finish work. The last 10% of the work felt like 50% of the code. When you got to the finish work, and code cleanup, the fun is over.

The dental industry moved fast on supplying dentists with pre-printed continuous feed insurance forms. You could even use carbon copy forms because dot matrix printers imprinted text just like typewriters, physically stamping a print head onto the paper, which was strong enough to imprint the text images on multiple sheets of carbon copy paper. It was great for the dental office, press a button and all the day's insurance forms would print in a single batch, saving hours and hours of hand typed processing.

The police industry didn't have it so easy. Form companies didn't see the demand to supply pre-printed continuous feed crime report forms. So, we had to build the forms line by line as they printed. We used **ASCII** codes to send "--" lines and corners to build boxes around text. A report form would take maybe twenty pages of code just for the print routine, and days of code writing to make it all look good. The different make and models of printers would interpret ASCII codes differently, so coders had to add IF statements to handle the different printer styles in their code.

Later versions of Foxpro, and other languages, included report form generators which at first were clumsy to use but well worth the learning curve. All that manual report writing went to a dark hole in space forever, saving coders one half of their work.

The first time I got to see an **HP laser printer** with **PostScript** in action was again one of those amazing "Oh My Gosh, sign me up!" experiences.



One of, and maybe the best-selling point of my police software, was that I included the source code. That meant I provided the un-compiled editable version of the software, in the event something would happen to me or I would go out of business. This was an unheard-of philosophy in the vertical market software industry, for the biggest fear of developers was software theft.

Software license protections were a complete nightmare to software users. Developers would use methods that would force software use to a single piece of hardware and if the user purchased a new computer or had to install a new hard drive, their software licenses would fail.

Many software developers realized they were spending more money on user support than the loss of potential software theft and changed to protection methods less aggressive to users.

## PC Technical Snap-Shot – 1987

Every business had to have a PC clone. You were able to network your personal computers and have access to shared data. We were all saying, “The day that personal computers can talk over a WAN, that will be the day personal computers would take over the world!”

Software for most industries was being developed and the need for hardware reached levels that required super stores to supply the demands of the general public. As a result, the “Computer Consultant” is born.

Printer manufacturers flooded the market with tons of printer models in hopes of selling profitable ink cartridges. HP becomes a player in the PC clone market and years later survives a highly competitive industry, successful to this day.

## Chapter 4

### Growing Pains

The year is 1989. The Berlin wall falls on November 9<sup>th</sup>. The Exxon Valdez oil disaster occurs in Alaska in March. Rob Lowe is spotted in a soft porn video with an underage girl. An October 17<sup>th</sup> quake disrupts the third game of the world series between the San Francisco Giants and Oakland Athletics. The parents of the Menendez brothers are found murdered. Pete Rose is banned from baseball for betting on games. Students protest on Tiananmen Square, Beijing, China - the army intervenes; 3000-7000 killed on June 3<sup>rd</sup>. Batman, Road House, and Dead Poets Society were box office hits. The B-52's "Love Shack" is the best rock song of 1989.

Living in the PC world was like riding a rollercoaster. Not only was new technology hitting the streets every month, but existing hardware was advancing so quickly that the purchase of a new computer was obsolete within the year. For example, the first [hard drive](#) was [5-megabyte](#) in storage size, and heavy enough to be a door stop. Then there was the 10-meg model, then the 15 meg, then the 30 meg, then the 60 meg, then the 120 meg and so on and so on. Each model smaller in size with faster access speeds. Upgrades were often required for clients and getting the data from the old drive to the new drive was time consuming and tedious work.



The first noticeable enhancement was the [color monitor](#). They were big, heavy and expensive. But everyone had to have one, which also included the need for a new graphics card, graphics were not built into the Motherboard yet.

For software developers, the color monitor required code to display text and basic graphics in color rather than **monochrome** or black and white. At the time, it was hard to programmatically determine which type of graphics card was installed on a computer, so coders had to build workstation, or login tables that stored the hardware properties of each workstation.



When a user would log into our software, they would be prompted to enter the workstation ID they were using. Coders would create global variables, one being the type of monitor and set color schemes for the user experience. Attempting to view monochrome color schemes on a color monitor was a waste of color technology and viewing a color scheme on a monochrome monitor made it impossible to see anything.

Because of the rapid advancements in PC technology, we had clients with a very wide range of computer configurations. Government and small businesses would purchase thousands of dollars of PC hardware and not have the financial resources to continuously upgrade.

This was a burden when it came to code enhancements that relied on newer computers and a nightmare for software upgrades. To allow clients to have access to newer code, if their hardware was powerful enough to run it, we built a data table storing client hardware configurations. The software would look up this information before running newer code. If a client's hardware configuration was too old, we would run earlier versions of the code for them.



We would also use the client configuration table to run custom versions of code as needed. As client hardware configurations changed, we would update the master configuration table within the next version release, that the client had access to in the next software upgrade.

I had a rule to write and test code on the oldest computers we owned to ensure overall software performance would be satisfactory to all our clients. Through the years, I have seen newer coders use the newest and best code libraries available to them. This was unfair to users where software was still on older, slower computers.

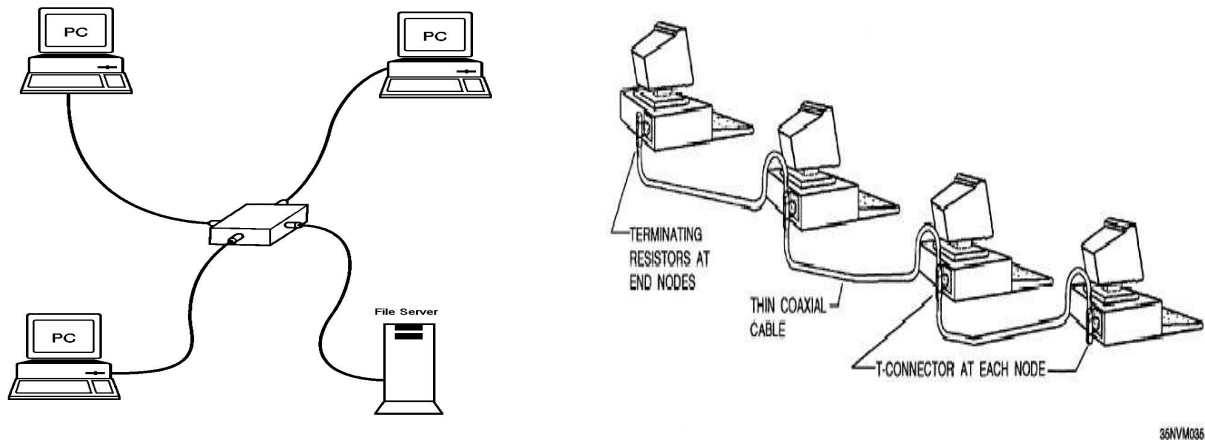
**Novell**, a Utah based networking protocol company, launched their pc-based file server and networking drivers to the general public. These were sold through authorized resellers, in which Frank Goodyear was involved, Novell's dedicated file server software and their **NE2000** network driver, that managed network cards in **DOS** and Windows environments, was in my opinion, the single most important enhancement to the growth of personal computer.

Windows 2.1 was on the market, but we DOS software developers wanted nothing to do with it. DOS programs could not run within the Windows OS yet. Also, in the business and government markets, Apple computers were nonexistent.

Along with faster motherboards and larger hard drives, larger police departments were able to use my software. This led to over fifty (50) software installations throughout the United States. I was on the road most of the time and living out of hotel rooms. I would spend a few weeks at any given location, then move on to the next. Being on the road required me to hire another programmer, Omid Haghani. Omid was a great coder and I didn't need to pay him much just as he was fresh out of college.

Novell **NetWare** allowed for two different network architectures. **ARCNet**, which was the old "spider" design, connecting workstations together using data hubs. The second was **Ethernet**, which was a newer daisy chain design. Workstations were connected one after the other in a daisy chain throughout the building. Each port of a file server's network card, they

could have up to four, could handle its own daisy chain. Ethernet quickly became the industry standard as it was faster than ARCNet.



I disagreed with using Ethernet over ARCNet because the 'spider' design was more reliable than the daisy chain design. If a workstation in the daisy chain lost network capabilities, such as a faulty cable or network card, the rest of the computers on that daisy chain lost connection to the LAN. ARCNet didn't have this problem and I couldn't have a computer somewhere in the building of a police department cause the dispatch computers and other mission critical workstations to stop working.

There were many times the IT department of a city would begin the process of installing an Ethernet network for a new police department client of mine, where I had to stop them and demand the switch to ARCNet. They would refuse and I would fight to the death, refusing to install my software on an Ethernet network. I won most of the time, and when I lost the fight with IT, I just installed the cabling myself with the blessings of the police chief.

A few years later, Ethernet incorporated the ARCNet architecture and the daisy chain design was retired. Soon, powered networking switches were available and [cat5](#) networking cabling replaced the coaxial. Novell LANs were now extremely reliable and the benchmark for all future networking platforms.

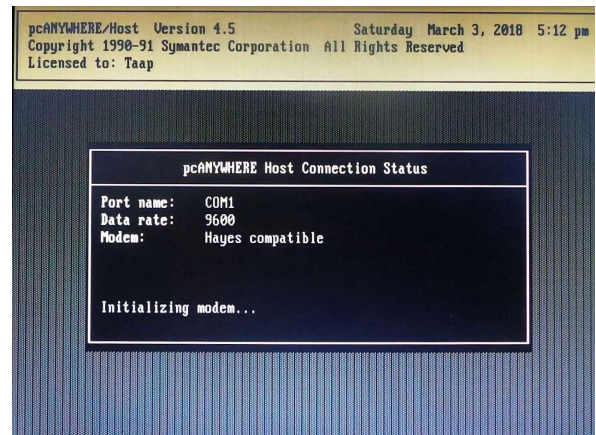


Novell was spending a lot of money promoting their pc LAN/WAN solutions. I recall attending a free invite party boat cruise in Newport Harbor that included dinner and drinks. Novell was showcasing new interfaces with Apple workstations that evening and making sure we all were enjoying the experience. Novell solutions weren't cheap and anyone that wanted to be someone was a CNE (**Certified Novell Engineer**) which was granted by Novell upon completion of their paid courses.

The reality of the pc WAN rumors came true when news hit the streets that Southern California Edison, a large utility company in California, had abandoned their mainframe systems and built a pc server room with over one hundred servers, networked using Novell technology. That was all we needed to hear. If they could do it, everyone could do it!

The first-generation pc **data center** was born. **Network bandwidth** was expensive and hard to get to your building, so large companies took their pc servers to the bandwidth. Telephone company facilities hosted all the data lines. PacBell; Pacific Telephone saw the opportunity and started renting data space and selling large amounts of bandwidth at high tech, well secure data centers. I was able to tour a PacBell data center in 1998, which I will describe in a later chapter. It was truly amazing.

As modems became faster, up to 1200 baud, and less expensive, we were able to remotely upgrade and maintain software using a monitor emulating software called [pcAnywhere](#). The client host would plug a telephone line into the modem of a workstation then tell the pcAnywhere host software to wait for a call. Then we would plug a telephone line into our computer modem and tell the remote version of the software to call the host via the telephone number associated with the modem.



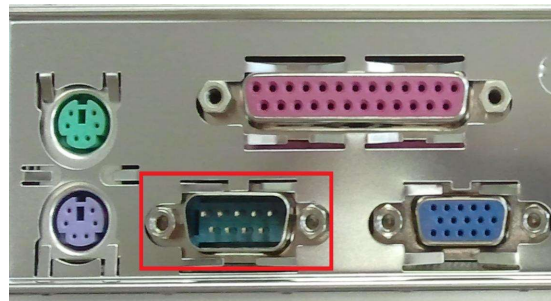
The host would accept the remote call and allow for the remote user to login with a username and password. After authentication, the remote computer would see what was displayed on the host computer, with the ability to control the host mouse and keyboard. We could also send files between the host and remote computers. This was what remote desktop is now, or even services such as [LogMeIn](#).

Remote desktop capabilities allowed us to upgrade and support our clients without having to travel. This was amazing. I can remember the days where I would be woken up in the middle of the night, with an incoming call from a police dispatch center needing help with their system. I was able to connect my computer to their host workstation and access their system from my bedroom. Half asleep, and usually half-drunk from the night before, doing what I did best - writing code in the middle of the night.

Problems were often fixed with lucky guesses and the all mighty “let’s try this” technique. Everything we were doing was new and oddities were often first occurrences with no documented solutions. In one case, I remember having a new police department that switched from an old, expensive mainframe system to my pc-based system and needed to export their data from old to new.

The mainframe software vendor was not happy with the department’s decision to switch and as a result cut off their support. The department

needed to export all their data from the mainframe to a media that I could import into my software tables. Without the help of the mainframe vendor, the department had no clue what to do. I utilized the “let’s try this” technique in which I used a [null modem cable](#), connecting it to the end of the mainframe’s printer cable and to the [serial port](#) of a personal computer. The null modem cable faked out the mainframe to think my pc was a printer, so we printed data reports from the mainframe, and I captured the data coming into the PC as text files. It worked! And we exported all their data through the available reports on the mainframe software interface.



I then parsed the text files and imported data into my tables. This was obviously harder than it sounds, and it took several days to complete. Properly designed data tables and their relationships was essential for fast and powerful applications, which I learned early in my career and made a specialty. What I had to do for successful installations was sometimes crazy in nature and never compensated for, but man was it fun!

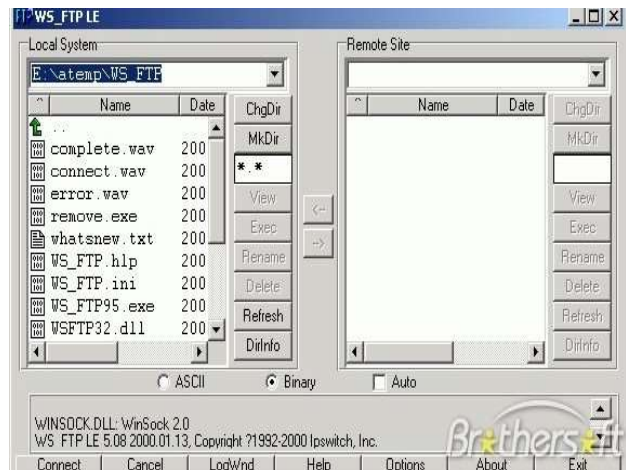
The year is now 1990. Saddam Hussein orders the Iraq invasion of Neighboring Kuwait. “The Simpsons ” is aired on Fox for the first time. West Germany Wins 1990 World Cup in Italy. Boris N. Yeltsin becomes president of the Russian republic. East and West Germany reunite. Home Alone, Ghost, Edward Scissorhands, and Total Recall were box office hits.

Laptops were now available but expensive and bulky compared to today’s models. I needed one for my police department installations and chose the [Acer Anywhere](#) because it had an internal modem. I remember paying



over \$2000 for it and it was worth every penny. I was able to take the laptop with me to installations and transfer all the software code from the Acer to a client's workstation via **file transfer protocol** (FTP), serial port to serial port.

**WS FTP** Light was a free file transfer utility that allowed people to transfer files from computer to computer, sitting side-by-side of each other. This was huge as the only other option was to burn floppy disk sets and copy them one-by-one onto the new computer. Computer consultants would use this utility to setup new computers with software. This was also the preferred solution for hard drive upgrades.



Windows finally enhanced their OS to allow DOS applications to run within a DOS window. Windows users were now able to run my police software from within a single OS. Windows now had their own LAN networking solution that worked a lot like the old **Lantastic** platform, utilizing a workstation as the network's file server. Novell was still the preferred vendor for larger networks because dedicated file servers were faster and more reliable.

Windows soon become compatible to the Novell ne2000 network driver which now allowed facilities to have Windows workstations, running DOS applications, on a Novell LAN. This network configuration quickly becomes the industry standard.

PC LAN solutions, Larger hard drives and faster **CPU**'s enabled more and more police departments to purchase my software. Then came the need for more screen display for the dispatch centers. At this time, personal computers could only support one monitor, and dispatchers needed a second stand-alone monitor to display patrol cars, their location, and status that could be viewed at any time by the dispatcher and other staff,

without interfering with other dispatch screens, such as 911 input and incident updating.

I could have just added another computer in the room that displayed officer status information, but it wouldn't update in real time, and many departments didn't have budgets to purchase additional workstations. My solution was to use a dumb terminal. The [Wyse terminal](#) was cheap and able to connect to the dispatcher computer's serial port. Each time an officer's status was changed from the dispatcher's screen, we sent a data string through the pc serial port to the dumb terminal, which would refresh the screen and display two columns of officer status and locations.



It worked great and only cost the police departments \$300 for the Wyse terminal. Even as dual monitors became supported in Windows, the Wyse solution was still desired because of the immediate update from the dispatch console, and the Wyse terminal did not interfere with other applications.

College police departments were having problems getting all the parking permit data into my software system at the beginning of each semester, data greatly needed the first day of school. The issue was that admissions would process 20,000 – 40,000 students a few weeks before the start of school and the permits were hand-written. Police department staff were responsible for the data entry and they didn't have enough employees or

You know those [bubble fill-in test forms](#)? A company named [Scantron](#) made those forms and the machines that scanned them. Well, they had just come out with a scanner that captured, then transferred the data to a personal computer via the serial port.



Matching needed enhancements with new technology was probably the most fun we had back in the day and it was happening non-stop.

45

patterns of vehicle burglaries occurring in my own neighborhood and was able to predict when and where these crimes were to occur in the future twice. I actually stayed up waiting on the two different nights and was able to watch the crimes unfold, calling the people into action both times.

If car burglar visits his mother on the north side of town every Tuesday night, he will pick that part of town to commit crimes on Tuesdays. He may have to wait until everyone else in the house falls sleep or after their shift is over at the factory. They don't plan the crimes, they just commit them as they need money and at their own convenience. Such behavior develops patterns. This is how Texas Ranger Frank Hamer was able to ambush Bonnie and Clyde in 1934 with 167 bullets to their deaths.

I wrote algorithm that would track crime categories compared to map sectors, day-of-week and time-of-day event details. All my police department clients were geographically too small to collect enough data for the code to find patterns, so my clients would have to seek out co-operation with other neighboring agencies, for example, all the cities within a small county, for data collection. At this time, agencies that were computerized were unable to share data, so such data collection would have to be done manually. Also, agencies didn't like sharing data of their on-going investigations for some reason. So, code that predicted crimes never got used.

I thought it would be a good idea to start reselling other software products. I found a hotel management software developed out of Texas for smaller hotels. It was in the right price range and I that thought my crew could sell it. I created ISMG (International Software Marketing Group).

After our first mailing to hotels in California, we received a lead from the owner of three hotels in Mammoth Lakes, a popular ski resort, and sold him on all three properties. I spent two weeks at the hotels installing hardware and the software at the front desk and back office of each of the three properties, then started the setup and training.

After a couple days of setting up the software, we realized the software totally sucked. It wasn't flexible and encountered fatal errors, for reasons

the software developer could not resolve. The hotel owner wasn't happy and decided to abort the installation with the request for a refund.

I had invested too much time and money on the project and had a better solution for him. I offered to write his custom hotel management system at no additional cost, except to provide me a room when I was there working. He agreed to my offer and off I went learning all I could about the hotel's reservation and back office needs. Then I started writing code.

By this time in my life, I was a code ninja. I was able to complete the first usable version in two weeks, then spent the next few months cleaning up code and adding new features as needed. I didn't want to spend the rest of my life maintaining this code, so I found a local **dbase** programmer that the hotel could hire from time-to-time for updates and modifications. I supplied all the source code for the project and handed it off.

I was so pissed at the hotel software developer that I stopped selling other-people's code. There was this mad rush for software and lots of developers were releasing what we called vertical market software, code for a specific industry, and most of this software wasn't written by people within their industries and sucked.

I always had the ability to learn and listen, developing code in my head during the learning curve and creating applications just the way a client expected it to work. It's easy for a coder to build an application the way they think it should be designed, ignoring the details outlined by the client. As Bill Murray says in **Caddyshack**, a movie you must love to say you're a coder, "Be the Ball". Get into the heads of your clients, see what they see, and build their application the way they envisioned it to be.

PC Technical Snap-Shot – 1990

More power, more data storage, more devices, and more applications. The personal computer was here to stay, and in more homes every day. Novell sets the standard for LAN technology and companies embrace the PC LAN and abandon the mainframe world.



Laptops were now available, but expensive and old-of-date a few months after purchase. Practical for the traveling businessman, but no so for the general public.

Modems were now cheaper and faster, which allowed for pc-to-pc data transfers. Software was soon available for pc-based faxing. The pc serial port was well used for moving data to and from other devices.

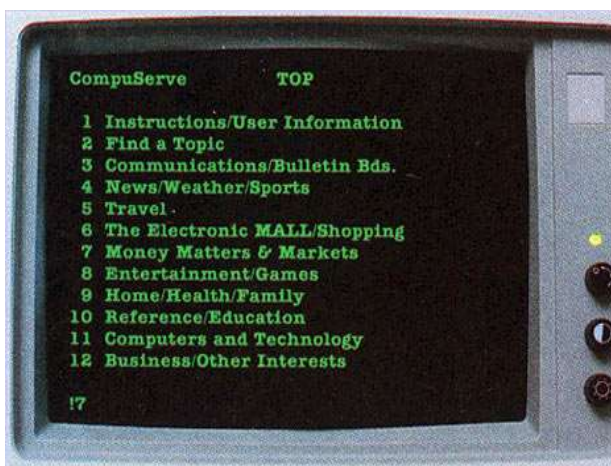
## Chapter 5

### And Then There Was HTML

The year is 1992. Ross Perot announces he will run as an independent in the upcoming presidential race against Bill Clinton and President Bush, Clinton Wins. Rioting breaks out in Los Angeles following the acquittal of four white police officers accused of beating black motorist Rodney King. Prince Charles and Princess Diana separate, and The North American Free Trade Agreement between U.S.A., Canada and Mexico is signed. Wayne's World, Basic Instinct, A Few Good Men and Reservoir Dogs were box office hits. Peter Gabriel "Steam" is the best rock song of 1992.

Finally, the Internet is born, and it just appears with no big hype. At first, we had access to [CompuServe](#), which was a [bulletin board system](#) (BBS), accessed by [modem](#) and command line driven, with a look and feel of a dumb terminal. There were no real graphics available and modem speeds were very slow, under 1200 baud, so text and very slow loading images is all we had.

Next was [AOL](#), "You Got Mail", and yes EMAIL. The AOL user experience was AOL hosted web pages and very limited. An email had to be retrieved from within the AOL website. Users dialed into the AOL network via a modem. AOL deserves credit for being the first web-based user experience, of course for a monthly fee.



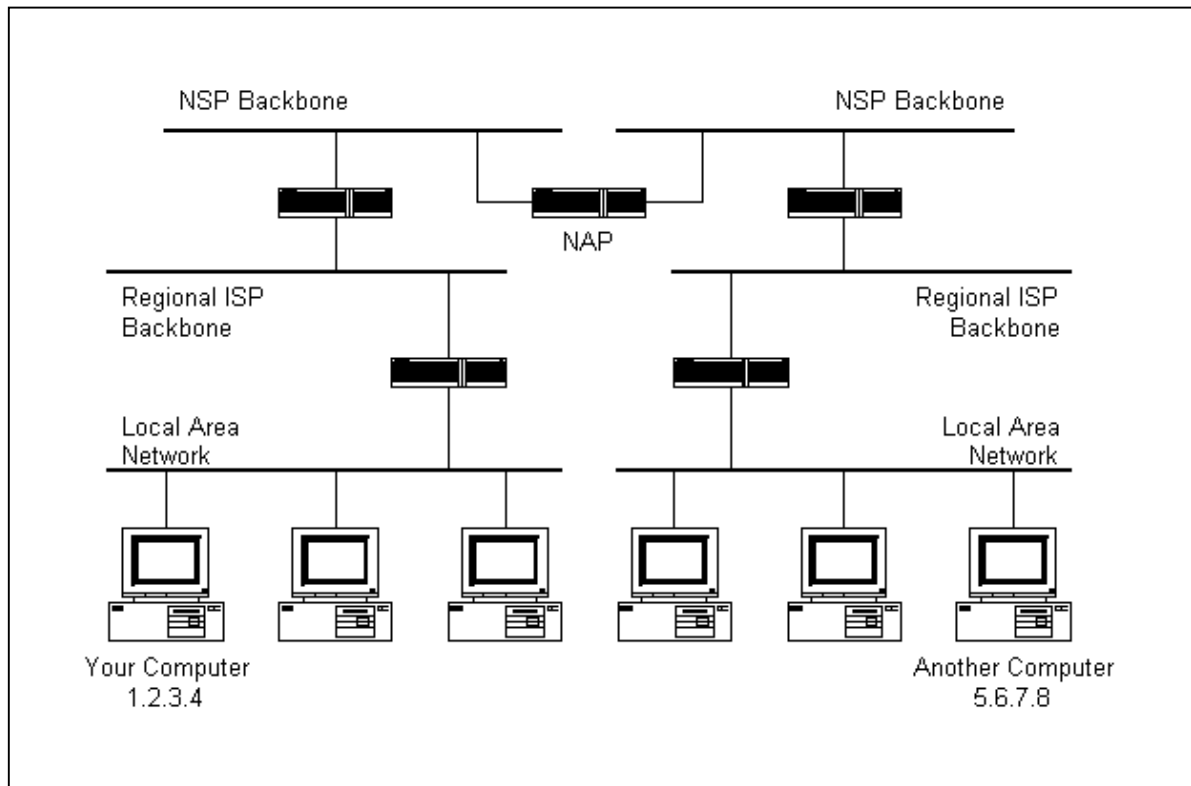
Companies such as **Netscape** entered the market which allowed users to dial into their network and browse public websites, plus they provided email services. So, you would dial into the Netscape network, then load the Netscape browser and surf a very limited Internet. The **Internet Service Provider** (ISP) was in its infancy.

The Internet was kind of just “THERE”. It was hard to learn much about **HTML** because there wasn’t any information available on the web, yet. No one knew about registering **domain names** or where to host websites. Information was slow to reach the public, as anyone that had such knowledge, kept it to themselves. ISP’s, such as AOL, controlled access to webpages through their portals. To be honest with you, no one was really taking it seriously at first.

So, remember back in Chapter 1 and the discussion regarding the mainframes and how we used dedicated lines and routers to move data through the telephone companies, what we called wide area networking WAN? Well, that’s kind of how the Internet works. A public data network controlled by many, many routers.

Every web server and browser accessing the Internet are assigned an **Internet Protocol** (IP) Address. IP addresses are like telephone numbers and domain names are easy to remember labels for website IP addresses. For example, the IP address for my web server that hosts my website may be 204.250.24.14, which no one would ever remember, so I purchased the domain name “cjbowen.com” and assigned it to my IP address. When you type ‘cjbowen.com” in a browser’s address bar, a **Domain Name System** (DNS) lookup is made to find my IP address. The browser is then routed

to my IP address, which is a web server that hosts my website. IP address routing works kind of like this: to find the IP address 204.250.24.14, the network finds the router location of the “204”, then finds the “250”, then finds the “24”, then finds the “14”, which would be the Internet connection associated with my web server. [WAN-Internet-diagram](#)



The year is now 1995. United Nations intervenes in Bosnian Civil War. A magnitude 7.3 earthquake occurs near Kobe, Japan, killing 6,433 people, OJ Simpson is found innocent. Major League Baseball players end a 232-day strike and the 55 MPH limit imposed during the Middle East Crisis in 1973 / 1974 is ended. Ace Ventura, Goldeneye, Braveheart and Apollo 13 were box office hits.

Sadly, the police software business died. The current economy hurt small cities to a point that the police departments had purchase freezes. I gave the company away to a friend that wanted to handle support for the existing clients, and I moved on. I was excited to find a new project to work on. I was young and that made all the sense in the world. Looking back now, what I did makes no sense at all!

I started learning HTML from tutorials that became available and by viewing existing web page code. Web pages were dumb, and HTML loaded like the old mainframe terminals, displaying text line by line starting from the top of the page. There was no **JavaScript** and page widgets were written in **Perl** and **Common Gateway Interface** (CGI) scripts. Also, if you were building web pages, you built them from scratch using a text editor, there were no “visual studio” products available for visual design. I think Frontpage was the first, but any respectable programmer wouldn’t use a tool written for beginners.

Soon, free CGI script web sites started appearing on the Internet. You could download most any html feature from these sites and you really didn’t need to learn Perl. Any of the scripts that stored data, such as a shopping cart, used delimited text files that were located on the website folders of the hosting server.

Websites were hosted on **Unix** web servers that were usually hosted by companies with access to bandwidth or in a **data center**. You could rent a Unix server for about \$300 a month and you would remotely connect to maintain your websites. The Unix operating system was command line driven and if you wanted to maintain websites you had to learn many of the Unix commands, this is all I needed, something else to remember!

Because website data was stored on the Unix servers and at a different location from your business, there was a disconnect between your web data and your day-to-day business applications and data. Moving data between different locations and different operating systems was pretty much out of the question for the time.

To solve this problem, developers began building browser-based applications that were used on the web and in the office, sharing data. This was the first real WAN (wide area networking) solution for personal computers.

Moving from desktop applications with small data sets to web applications in WAN environments, we were now exposed to applications that could



require millions and millions of records in shared tables, which allowed SQL to take off and in a huge way. Interfacing with SQL for programmers such as myself, was impossible at the time. Until solutions such as API access to data was fully developed, we had to somehow figure it all out, such as manually converting and merging data after hours. Migrating data was clumsy and time consuming, which often delayed access to real-time data.

Network Solutions was really the only domain name registrant available, so there I went and domain names I bought. Everyone I knew wanted a website, and I aimed to please. I was writing so much HTML I could see it in my sleep.

It was hard bouncing between program languages and the different server operating systems, some days I would use them all. I was writing code in Foxpro, HTML and Perl, and maintaining Novell and Unix servers. JavaScript routines were starting to become available and to survive in this new world, coders had to learn every language out there, and on his own.

In the desktop application world, coders learned to utilize large code libraries, such as Java includes, to allow access to features on-the-fly. Upon an application start-up, all the libraries would load into memory and use of library features were fast and easy to migrate into the application.

As JavaScript becomes popular for front-end, HTML design, coders used the same style of adding large libraries at website start-up. Developers slowly learned that loading large libraries into web browsers at website start-up, on slow bandwidth connections would cause website pages to load slowly and often fail, for the libraries would never finish loading into the browser's memory. A good example of this, is the ATT.COM website. I live in a small town with poor access to the Internet and this website takes forever to load.

This style also caused problems on the web server side of delivering business logic and data back to the front-end. Every time a webpage calls a web server, the web server's IIS service initiates a worker process to run the application's DLL. If the DLL is overloaded with code libraries, each

worker process is required to use excessive server resources to process the webpage request, thus limiting the number of web request threads the server can handle at any given time.

Yep! Another project, another application.

My father invented a dental drill sterilizer. Dentists didn't sterilize their drills between patients until the AIDS scare of the 1980's, which forced them to do so. He talked my brother into forming a shareholders group of doctors to fund the start-up manufacturing of this well-needed medical device. I was asked to join the management team for my technology experience.

Manufacturing and customer support software that were US CFR 800 **FDA** compliant really didn't exist yet, so I put my FDA hat on and went to work. I studied the CFR 800 learning everything I needed to know about FDA quality assurance compliance and built a custom software application for the plant. I ended up knowing so much about FDA guidelines that I wrote a 120-page manual on FDA compliance for small manufacturers that I sold on Amazon for years,

My life was driven by the code I wrote. I wasn't thinking of my career, I was thinking of the next application I could build. I saw myself living these cool adventures through the code I wrote, hoping to have a problem to solve or an interface to write. As projects were completing, I wanted to move on to the next one, with endless opportunities.

The company issued me a cell phone. They were just now available to the general public, but most were purchased by companies. There wasn't the ability for texting or browsing the Internet and the cell calls were expensive. Cell phones were basically conversation pieces that would impress your friends.

By this time drivers were available for devices such as electronic time clocks and bar code scanners. The driver software interfaces were often clumsy, but I was able to capture data as needed to make the QA software seamless. If I could get a port listener to work, I could capture and parse text from most any device to my data tables.

A Novell dedicated file server with Windows workstations was an obvious must that I installed along with the telephone system. Printers, scanners, telephone interfaces, anything with a logic board seemed to attract my interest and I was usually the only person in the building that could even attempt to install and maintain such devices.

Pacific Telephone began a sales campaign to sell [rack space](#) in their new data centers, one close in Irvine. While the PacBell representatives were visiting the manufacturing facility, I was able to talk the talk and convince them our company would be a perfect candidate for rack space in the data center, I was given the invite for a tour, which I gladly accepted.



As I approached the address on the invitation, all I could see was a huge single-story industrial building with no business markings. All I could access was a small public parking lot to the side of the building. Access to any other property was fenced off with video surveillance. Through the front doors was a lobby with a security window to one side. Next to the security window was a thick glass door that entered a small security pocket between the first door and another thick glass door. It felt like I was in some sort of military facility. It was kind of spooky.

I wasn't sure I was in the right building, then in walks my sales representative. He escorts me through the first glass door. After security authenticated my identity, I was issued a visitor's badge and granted access through the second door. I was now standing in front of a large glass wall overlooking a control room with a huge map of the United States the size of the entire back wall. The map had colored lines moving all over the country that represented the US bandwidth channels. Apparently, people in this control room could reroute bandwidth as needed throughout the country.

The building had several conference rooms that companies in the data center could use, equipped with video conferencing, which was unheard of in the real world. Clients even had access to sleeping bunks, showers and break rooms. You could basically have staff in the data center 24/7 365.

They walked me into the [server room](#). OMG, I'll never forget this moment. Standing inside a [huge white room](#), I mean like the size of a football field. The entire room had [raised white flooring](#) for the cooling system and for running cables throughout the room.



The room was cool and temperature controlled. The room, or should I say the compound, was filled with [large metal cages](#) with a maze of walkways. In the cages were racks and racks of servers, I mean servers were everywhere. The [white noise](#) of thousands of server fans running was so loud you could barely hear yourself talk.



As I looked into the metal cages I could see technicians wearing headsets sitting at desks, surrounded by racks of servers working away. They all looked grungy as if they hadn't showered for days, staring at their monitors as if they were in the middle of an online game.

Outside the building, in a well-protected area, there were several huge generators, which could handle power outages for weeks with an amount of fuel on site to do so. They also had an agreement with an oil company to receive priority fuel during emergencies.

I wanted to have servers in that building, even though I didn't have a need for them. The desire for wanting to participate in the data center dwindled as I reviewed the quote for rack space. It was extremely expensive, yet they were full and currently constructing a second phase of the building.

Large companies were moving their data to pc servers and placing them in data centers to have access to bandwidth for dedicated networks and access to the web at a cost much less than the cost of maintaining their abandoned mainframe systems. This was obviously the data center's market, not for the little-guys-like-Chuck.



## PC Technical Snap-Shot – 1999

The Internet was on full throttle and the PC world was now accessible in a WAN environment. Larger amounts of data were being stored to handle the Web environment and there was a developing need to communicate between servers at different locations.

Bandwidth was expensive and hard to get, so the telephone companies built data centers to rent rack space and sell bandwidth at a premium. Companies could now rent physical servers located at data centers with remote access.

Cell technology is now available to the public and the micro-chip finds a whole new market. The concept of texting was new and costly, especially as our kids started using cell phones.

The first portable high capacity disk drive was released in 1995, the [Iomega Zip drive](#). This drive was external and the disks were larger and thicker than the 3 1/2" disk. The first release had a storage capacity of 100 Meg, which was insane for the time and everyone had to have one. People would carry around stacks of these disks with all their data in hand, it was crazy. This technology was short-lived as writable DVDs and USB flash sticks became available.



The band **Steely Dan** becomes the official music of the International Coder's Association – just kidding, but if I were the President of such an organization, I would have made Steely Dan the official music of all coders. Steely Dan was comprised of professional studio musicians who would sit in a circle playing music to each other on stage during their concerts. They looked and acted like coders and produced some of the world's best music. Sit back and listen to Steely Dan's '**Dirty Work**' and all that I'm saying will become the reality of your own lives.

## Chapter 6

### Entertaining Bandwidth

The year is 2000. Mad Cow Disease causes alarm in Europe due to its growth. Tiger Woods becomes the youngest player to win a Grand Slam in Golf, U.S. presidential election, 2000 Republican challenger George W. Bush defeats Democrat Vice President Al Gore, but the final outcome is not known for over a month because of disputed votes in Florida. The Concern over Y2K passes without the serious, widespread computer failures and malfunctions that had been predicted and The Lifespan in the US is now 77.5 years. Meet the Parents, X-Men, Scary Movie, Cast Away and How the Grinch Stole Christmas! were box office hits. 3 Doors Down "Loser" is the best rock song of 2000.

The release of Windows Server 2000 allowed companies to now have Windows-based dedicated servers. The Windows server OS was powerful, extremely stable, and allowed companies to host their own websites via the Internet Information Service (IIS). The Novell file server was soon doomed, and Windows became the only real competitor to Unix, which currently ruled the web server market.

The gambling and porn industries jumped into the online market fast and furious. Porn at your fingertips. It was so cool. Porn sites had images only, no video yet, and because modem speeds were at minimum baud rates, such images were slow loading, but worth the wait. I remember winning a \$4000 royal flush on the gambling site "English Harbor" located on the Island of Antigua in the Caribbean. They mailed me a cashier's check from a bank in New York.

The whole Internet experience was a fresh, exciting way to get known to people around the world. Everyone was working to get there first, buying the right domain names and getting listed at the top of search pages.

Receiving top ranked search results for your website was simple at first. You made sure the top-of-page meta tags described your website properly with appropriate keywords, and that the text on your web pages explained

the purpose of your website. Search engine robots (code routines that scanned websites) would use this information for search result placement. The more your profile matched the search keywords, the higher you would rank on the search results page.

This was amazing for small businesses because if they offered what someone was looking for, they would be listed even sometimes higher than the large companies on the Internet. For example, if all you provided as a service was “antique book repair”, you would have a high ranking for the search keyword “antique book repair”, more so than a website promoting “book repair”.

This all came to a stop when the search engine providers introduced pay-to-play and “the bigger you are, the more you get” search ranking protocols. Many fake websites figured out how to take advantage of this protocol and now most top-ranked websites on a search result are garbage, and the real, honest hardworking businesses are buried on back pages.

My brother Stan thought it would be cool to have a **T1 line**, a 1.5 meg dedicated data line to the telephone company at a cost of \$1200 USD per month, and he let me host some web servers at his business in return for writing all his web pages. I also kept quiet the fact that he loved watching the voyager sex sites.

One night while hanging out with a few of the other drunks at the local moose lodge, I came up with the idea of all ideas. “Hey, let’s start an Internet radio station!”. Someone says back, “Why? people can listen to music on the jukebox.” I meant video, audio streaming entertainment, where people can interface with the DJ, watch him live, request music and chat with other listeners, like an online community.

So, I set up a pc, camera and two microphone headsets at the corner of the bar and got different drunks to commit to doing shows each night of the week there at the bar. I purchased the domain [surfcityradio.com](http://surfcityradio.com) and built a website embedding the Yahoo chat engine for the DJ viewing and chat page. We streamed at 130 kbps every night at 7 pm. In the summer

of 2001, I believe I began the world's first video/audio streaming independent radio station, with live interaction with the online DJ.

Each one of us had a different theme for our shows. Mine being a shock rock theme. My brother became my best fan and offered for me to stream the shows through his T1 line. I setup a Windows media server at his business, increased the stream bandwidth to 256 kbps, 96 kbps for the music and 160 for the video. We webcast from the bar for about a year, then I felt the need to build a [webcast-studio](#) in my house.



Bandwidth receives and sends data. The download speed is what people use for watching and downloading Internet content. The ISP's, Internet service providers, basically give download speed away so they can drive up the demand for downloading content. The upload speed is very controlled and costly, as its what businesses need to deliver the content you download. For example, when you download a 1 meg image from a website, the website owner pays for the bandwidth it takes to deliver the image to your computer.

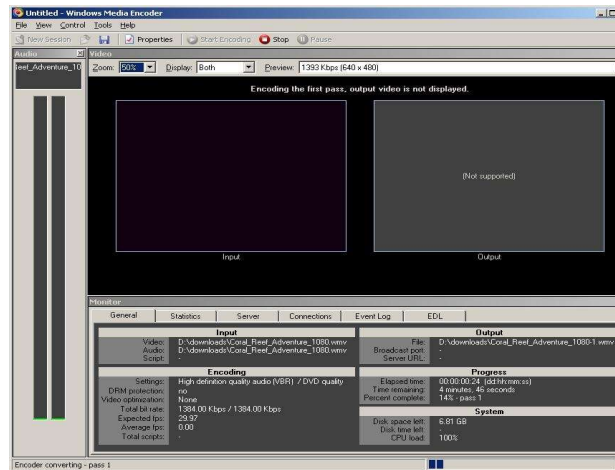
So, to webcast live video/audio content to viewers, I would need as much upload speed as I could find and afford. When streaming through my brother's T1 line, I had no cost, but the number of viewers was limited to the 1.5 meg pipe. I purchased an Internet business account with 2 meg of upload speed for my house for the webcast studio with a cost of \$200 per month. That was too expensive per viewer and I had to figure a way to get lots of upload speed for as little as possible.



My solution was to build Windows **Media Server** boxes and distribute them among family and friends' businesses, with the agreement I could use their bandwidth at night, when they were closed for business. At one point I had over twenty of these servers set up all over the southern California area. I created a viewer interface that would allow viewers to bounce between media servers when experiencing video delays or buffering. The servers ran without needing to be maintained and the viewers loved the whole experience.

So, how the streaming thing all worked? You would build a Windows Media Server, which is a service of the Microsoft Server. Then create an end-point that the video/audio source would steam to using the **Microsoft Media Encoder 9**. Viewers would access the associated server URL from within their browser's media player object. For every viewer, the server would need

bandwidth to stream it out to the Internet. So, if you were streaming a 200 Kbps video and 5 people were connected to the server, viewing the video, your server box would need at least 1 meg of upload speed bandwidth.



This is what I did every night, six days a week for the next eight years. I'd work my day job, then do a 7 pm - 10 pm show every night. My brother introduced me to a DJ located in Toronto Canada, who was hosting his own show in a video community website. His name was Mark Brewer and he did his shows long side his wife "Summer". Mark and I had the same goals of making our mark in the webcasting industry and soon agreed to work together on building a listener base.

I changed the name of the radio station to “5150FM” and registered the station with the United State Copyright Office as a radio station and Sound Exchange for complying with artist royalty laws set in place by the United States Congress. I thought 5150fm was a great name for the station, because law enforcement refers to crazy people as “5150”. I would do the 7 pm PST spot and Mark and Summy would do the 10 pm PST spot every night of the week. The viewing numbers grew, and we added more DJ’s until we had shows webcasting 24 hours a day, 7 days each week.



The radio station developed a cult following with an online community of thousands of people from [around the World](#). All my media servers could not handle the viewer levels we were receiving during our shows and I had to figure out a solution or fail. While we were able to technically make it happen, with great on-air talent, everyone was doing it for the fun. Making money was the last thing on our minds, and it should have been the first.



The year is now 2006. Twelve coal miners die in the Sago Mine Disaster near Buckhannon, West Virginia. Saddam Hussein is charged on May 15th and following the trial is found guilty of crimes against humanity and sentenced to death by hanging, Italy wins the 2006 world cup in Germany

defeating France. Barry Bonds of the San Francisco Giants Breaks the record held by Babe Ruth and hits his 715th home run on May 28th to pass Babe Ruth on the all-time greatest list and Congress passes the Communications Opportunity, Promotion and Enhancement Act (Cope Act) which ends current Network Neutrality. Monster House, Underworld: Evolution, V for Vendetta and The Pink Panther were box office hits.

**Http Request-response** was the ultimate solution to personal computer wide-area-networking (WAN). Web pages and servers could now make http communication calls to servers for data in which the connection would stay open, allowing the host server to return the data as the response back to the calling browser or server. As faster and cheaper bandwidth became available, this method of communication between servers allowed the PC world to move data acceptable to any industry, anywhere in the world.

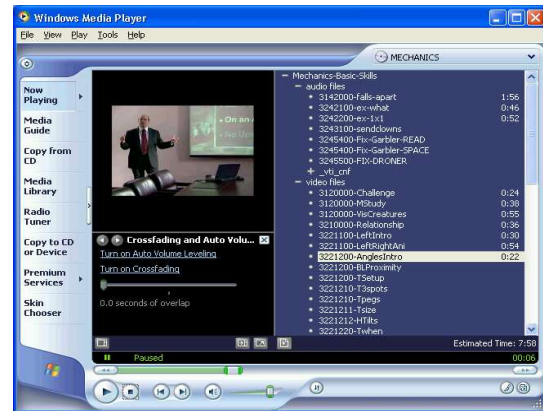
This solution would allow websites to communicate with database backends. A few different .NET bridges moved HTTP requests through IIS to Foxpro code. We were now able to utilize our own data tables, rather than the text files stored on the web server and allowed us to handle complex business logic for webpage request. We were now able to share data between our different applications, and from different locations.

Several new programming languages were now available, that new coders were learning in school. I was advanced in Foxpro and HTML and had good knowledge of **JavaScript** and **Java**, which allowed me to build most any application thrown my way, so learning new languages wasn't on my plate. Foxpro was and still is very powerful, allowing for fast deployment. Use of its own tables and now with the .NET bridge capabilities, I felt I was working with the best language platform available, and I still feel that way today.

While the radio station continued to grow, my buddy Roger and I came up with another idea. Yes, another idea! We developed a pay-per-view platform for streaming live concerts, we named the company Webcastia. I worked full time managing development of the website, backend and the actual broadcast system needed to film the concerts. Roger invested \$80,000 and we were able to get it all going for about \$30,000, most of which were the costs of the cameras and Audio/video mixing equipment.

A requirement for capturing live concert gigs was that the stream would have to be protected. The only encryption service available was Microsoft **Digital Rights Management**. We became an authorized DRM provider and developed a pay-per-view DRM platform. The Microsoft Media Encoder 9 would encrypt a stream with a data header that included the URL to our DRM authentication server.

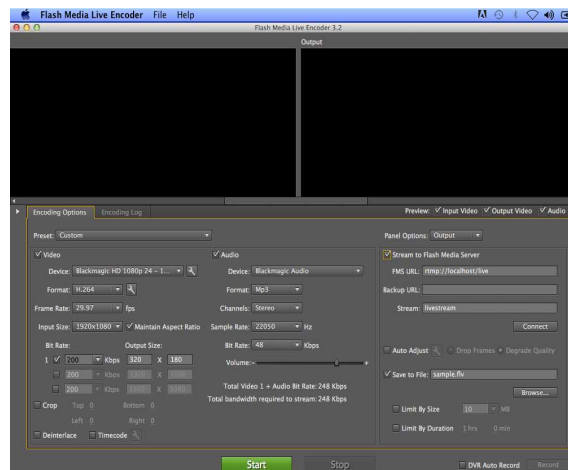
When someone pressed the play button on the **Microsoft Media Player**, the embedded header blocked the stream until authentication. The viewer was redirected to a web page on our DRM authentication server in which they were asked for their Pay-Per-View credentials. Upon successful authentication, the media player was given the ok to stream.



Microsoft was the only company to ever successfully provide a DRM protection service. Adobe Flash made several attempts to create a DRM platform but failed. Microsoft did have a bug in their design that caused issues accessing our live performances.

Once a viewer was authenticated from our DRM server, the media player would set up local credentials on the viewer's computer. The process required read-write access to the folder the player needed to store such information. If the viewer was not logged into the Windows machine as the administrator, the OS would not grant the user read-write access to build the DRM profile. This became a nightmare while supporting viewers during live events, so much so that DRM became less important than viewers having access to the content.

Our solution was to use flash streaming technology and develop a powerful authentication interface blocking viewers from the player when we didn't want them to see the content. It worked great. We were able to block the embedded player from unauthorized access, and even display messages to all viewers when needed. The solution kept the honest, honest with very low authentication failures during live events. We also began using the [Flash Live Encoder](#) to upload the content to the media server.



This solution didn't encrypt the stream, but our "keep the honest, honest" protocol was accepted because hackers were able to crack the encryption anyway. Microsoft's DRM later become their **Silverlight** platform which is still used today by content providers, such as Netflix, to comply with the wishes of the movie studios.

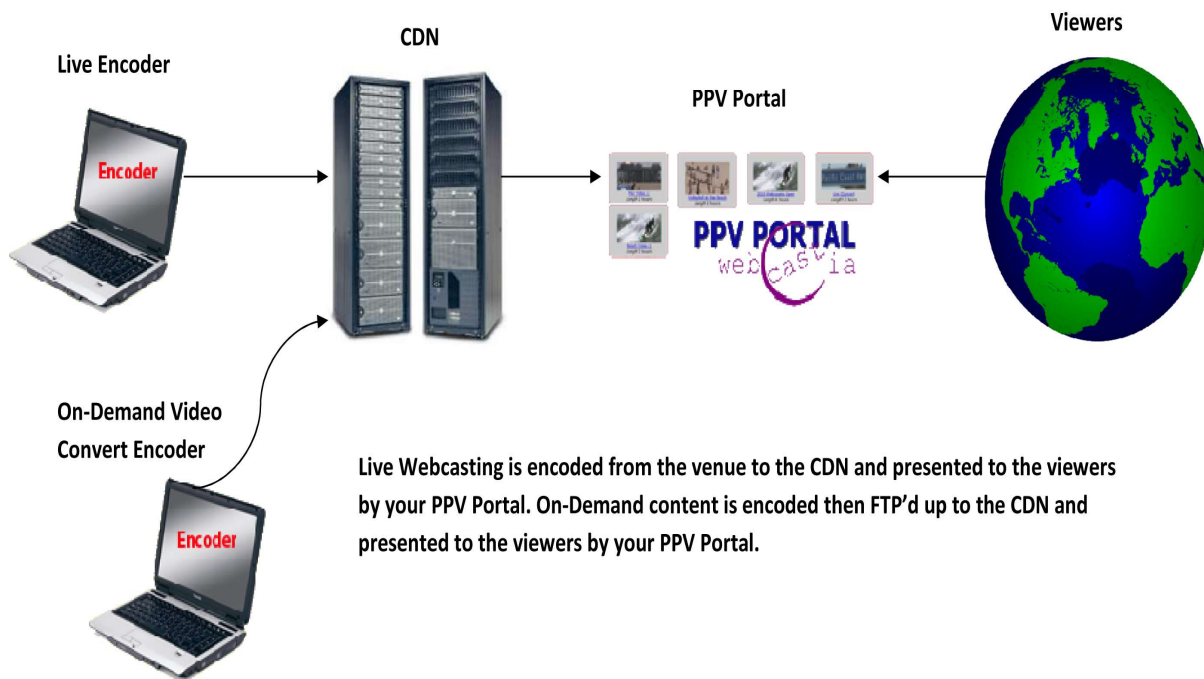
The next hurdle was that streaming 1 meg live content to large numbers of viewers could not be handled by any single server or even **data center**. The Content Delivery Network (CDN) industry was created to handle content streaming, which is still used today. So CDN companies would place media servers in data centers throughout the world. We would stream our live content to their endpoints, and they handled the load balancing between all the data centers to allow enough bandwidth to handle all the viewers. We had a .10 cent per gig cost to stream our live content through our CDN vendor, which was Limelight, so it was important to us to have a strong authentication platform, for we had a hard cost per viewer.



## Live Webcast Encoding Diagram



## PPV Webcasting



Then there was the browser compatibility issue for the embedded player object and the video stream type, which was flash (FLV). Every browser was different in some way or another, and Apple didn't support flash video streaming. Our solution was to start using the **JWPlayer**, which was a cross-platform media player object for web pages.

The JWPlayer was magic. The player would evaluate the viewer's browser environment then convert the stream to a format allowed by the browser. For the day, we had the best pay-per-view platform on the market.

Prior to an event, people would be sent to our website to purchase a ticket to an event. We would process the payment and issue a ticket code, such as 5Y4R-S21W. At the time of the event, authenticated viewers were directed to our viewing page, that included the JW player object script. We could remotely mask and deny access to the stream to all or any viewer and display messages as people watched the show. We had full control of the viewing experience from our live support control console.

At the **venue**, we would send a 5-man **camcorder** crew with a portable broadcast system we designed to fit in the back of my suburban. We would set up the **AV control table** connected to the venue's main **mixing board**. Then three or four camcorders were set up at locations in the venue to best capture the show. Cables were thrown from each of the camera locations to the AV control table.





The camera cable sets included an [XLR](#) audio cable for our communications system and a coaxial cable for the camera feed. All the cables had to be taped down and out of reach from the public attending the concert. Then we had to find the Internet router in the building and run an [ethernet-cable](#) it to from our AV control table. These routers were often located in the office area of the venue which required us to throw hundreds of feet of cable, up staircases, down hallways, through offices to connect our stream to the router.



Fronto

Wireless WIFI was not reliable, for the signals were too weak to stream the live content without losing data packets, causing buffering issues. Also, none of the venues had much upload speed and we couldn't control venue employees from using bandwidth while we were broadcasting. Our rule was to only use one half of the available upload speed which obviously determined the quality of the viewer experience.

There was a list of thirty things that could go wrong, any one of them would cause an interruption of the event. We were good at what we did and fast on our feet, which made us successful at all the gigs we shot, which was well over 50 shows.

We had to deal with Windows and their blunders. Remember Vista! Vista had new security protocols which caused most third-party drivers not to

work. That meant all of the AV devices stopped working and new drivers would not be available for some time. We had to stay on the XP platform for our AV control computers which was fine, because XP was stable and lightweight.

Then we learned the hard way that Windows started running automatic updates as you were using your computer. The updates would use all the computer's CPU and take forever to complete. This happened to us during a test cycle prior to a live event which shut down the stream. Note to yourself, TURN OFF AUTOMATIC UPDATES. At this time in my life, I swore that if I ever met Bill Gates, that I'd kick him right in the junk!

I have to say that filming all those live concerts around Los Angeles was great fun. We had full access to venues, including backstage areas and everyone loves the camera crew.

Myspace, the first social media platform, was used by all the bands to post upcoming events and songs, which could be played online. Everyone that wanted an online presence had a Myspace account, until Facebook became popular with a more robust interface. By this time, everybody had a cell phone and texting was the preferred method of communication with other humans. The Motorola Razr and the Blackberry Pearl were the cell phones to own, and if you signed a two-year contract with your cell phone provider, the phone was free!



The year is 2008, Shuttle Endeavour goes for a 16-day mission to the International Space Station, Governments around the world face the continuing problems of increased oil prices causing inflation and unemployment increases. Oil hits an all-time high of \$147 a barrel, Fidel Castro steps down as president of Cuba after almost 50 years in power. The Governor of Illinois Rod Blagojevich is arrested on federal corruption charges, United States Presidential Election 2008 Barack Obama (Democratic) Defeats John McCain (Republican). Iron Man, Wall-E, Step Brothers, Mamma Mia, and Hancock were box office hits.

Data center companies were renting physical server boxes with Unix or Windows operating systems in the \$200 range per month. **Virtual machine** (VM) instances were now available. You could now install a VM manager, such as **VMware**, and emulate operating systems on your server. This means one physical server box (hardware) could now host many operating systems, virtual servers sharing the physical box's resources.

This was huge! your \$200 per month server could now be three servers. All fully functional with remote access. This was the beginning of cloud computing technology. In a few years, you'd be able to create VM servers in the cloud, never to purchase server operating systems or deal with hardware problems ever again.



A friend asked me if I wanted to stream the live [runway](#) shows for LA Fashion Week, and I responded with a “HELL YA!”. I began working with [Box8 Studios](#) in downtown Los Angeles and [Stickam](#), an up and coming online video community service, for the stream distribution. We were given special backend access using the FME flash encoder to stream at any bandwidth we desired, and all the runway shows were featured on the main Stickam website. This gave us hundreds of thousands of viewers at no cost to us, Stickam paid for all the streaming costs.



I was given the title of Video Director by Box8 and oversaw the photographer platform at the end of the runway. Our video feed was used backstage by the show director to time the models onto the runway. My camera crew agreed to work the runway shows for free in exchange for free cocktails.

I was responsible for making sure video, music, lighting and the backstage movement of models flowed smoothly and followed the show script. During one of the shows, I was asked to come backstage to fix a monitor issue. I rushed backstage, then I suddenly stopped, realizing I was standing in the middle of a room surrounded by a dozen half-naked models, getting ready for the next show. They didn't even notice I was there. I felt as if I was invisible and I remember saying to myself.....well, maybe I shouldn't repeat what I was saying just then, but you could probably guess for yourselves.

The runway webcasts were so successful that Stickam allowed us to have feature programming whenever we wanted. I moved the radio station

broadcasts to Stickam and soon became the network's number one entertainer, with over a million viewers per month.

I [personally hosted](#) over 1200 live shows with over 4000 hours of live entertainment, and between the radio station and Webcastia, produced and/or directed over 25,000 hours of live content. We interviewed major artists such as [Mickey Thomas of Starship](#), [Terry Nunn of Berlin](#), Mike Reno of Loverboy, and showcased tons of local LA bands and national tour bands such as Shiny Toy Guns, The Maine, and LA Guns.



The best experience of all was having dinner with [Dom Deluise](#) at his home in Brentwood. A crazy story I must tell! We were showcasing a local LA band during a Stickam feature show, in which one of the band members brought a ton of home cooked Italian food, and it was amazing.

During the band interview I made comment to the great food they brought. One of the band members explained that he was one of Dom Deluise's personal assistants. Dom loved to cook and prepared food for us to eat

during the show. I couldn't stop praising Dom for how great the food was, while continuously eating during the interview. I was being a complete jackass, which was my best show quality.

A few days later, the band member who worked for Dom, calls me to say that Dom has invited me to his house for dinner. So later that week I show up at Dom's home in Brentwood, and we sat at the kitchen table for a few

hours talking. We spent the first 20 minutes swapping lines from the movie "Blazing Saddles", my all-time favorite movie.

I was hanging out with one of my favorite movie actors talking behind the scenes of my all-time favorite movie. Then his wife walks through the room and, OMG, she was the school teacher chick in the movie. I couldn't stop talking and Dom was loving it. He asked me if I wanted anything from him and I responded, "Just your company, my friend!" Dom was nonstop one-liners and he made me feel so comfortable, a truly great guy.

When we meet, Dom was in a wheelchair and not well. A month after our kitchen hangout, he passed away. I was blessed to meet him.

The Webcastia project never got its big gig, the one that would put us on the map. Again, we were before our time. The music industry couldn't understand the financial benefits of streaming live concerts for money. One of the major rock bands, as I remember Led Zeppelin, held their farewell concert at the O2 in London. They had a 20 million person waiting list for the sold-out show. We made several attempts to contact the promoters and the band's agents with not a single response. If they could have sold 20 million \$20 webcast tickets, they would have made an additional \$400,000,000.

This would have been a first, and as of today, would have still been a first. The reality of success would have been an endeavor worth every penny of the 10% payout. First, all four the of the major CDN providers would have had to participate to provide enough bandwidth to serve the 20 million streams throughout the world. Second, we would have had to build

dozens of web servers to attempt to handle all the page hits at the same time.

Keeping the page hits to a minimum and getting the viewers to the media player's CDN stream URL as quickly as possible would have been the plan. The band would have also needed to agree to stream the concert archive for a week to allow webcast ticket holders to view the concert later if they experienced technical difficulties during the live webcast, to eliminate credit card charge backs. If anyone could have made it happen, it was us!

Truly an industry managed by knuckle heads. If a promoter wanted it, a manager didn't want it. We'd get a concert set up then an attorney would disappear and/or forget to sign the final contract. Through the radio and Webcastia experiences, I truly gained the upmost respect for musicians and their dedication to the arts. I'm so glad to see artists progress through social media, rather than the traditional music industry channels, giving all talented artists their chance for success.

There was a need for online radio stations to have **video chatrooms** that allowed a host to invite several viewers, or participants, to join a group video session. I started playing with the Flash Media Player JavaScript code and was able to **API** call my Foxpro tables for user authentication. I then found the **Red 5** Media Server, and my life goes on hold for the next three months.

Red 5 was an open source Java application for a media server that handled video, audio and persistent user chatrooms. The application also required Apache Ant and a few other utilities to run. Getting all the setup files configured correctly was a nightmare. You would run the application then review the error log, which would have hundreds of errors listed. Try a fix, re-run, try a fix, re-run, try a fix, re-run – over and over, day and night.

After about three months of trial and error, I remember restarting the application and upon reviewing the error log, BINGO, none! I sat there for a few minutes in dis-belief, a successful Red 5 installation. People in the

Red 5 bulletin boards would say, “If you can get Red 5 to work, you are a God!” Yes, and I did feel kind of God-like for a few days.

Once the Red 5 Media Server was burned in and stable, I began writing tons of code for the video chat. The Flash player would call the Red 5 server for video streams and the persistent chat interface. If you want to learn **Java (programming language)** the old-fashion way, make Red 5 work and build some video chat interfaces. It was tons of fun.

## PC Technical Snap-Shot – 2010

The Internet providers saw the demand for video and audio streaming and developed new technologies to move larger amounts of data, faster and more affordable.

Cloud computing was developed and quickly becoming the way of the future. Virtual machines and virtual networks were now accessible to most any business, but still expensive. Amazon is the first to provide VM products with Microsoft Azure right behind. Owners of both these companies become two of the richest men in the World.

Smaller notebooks, tablets and smart cell phones were now available, and these devices had access to the Internet. The mobile app market explodes, Apple and Google racing to be on top, Windows losing the battle with limited and poorly written applications for their new OS.

## Chapter 7

### Cloudy Days Forever

The year is 2011. On December 18th the last US Troops to leave Iraq entered Kuwait in a convoy of vehicles nearly 9 years after the initial invasion. On March 11th an underwater earthquake with a magnitude of 9.0 hits off the coast of Japan, causing a tsunami that hit the Iwate prefecture with waves over 130 feet high. On May 2nd, it was announced by US President Obama that Osama Bin Laden had been found and killed by US Navy Seals in Abbottabad, Pakistan. NASA launches the Juno spacecraft during August of 2011 and North Korea's Kim Jong Il dies and is replaced by his son Kim Jong Un. No movies were box office hits! Linkin Park "New Divide" is the best rock song of 2009.

Microsoft Azure and Amazon AWS were fighting to control the cloud computing market. You could now replicate the server environment you normally would have in a cage at a data center, on the cloud, at compatible costs. No hardware, auto backups of your data, and unlimited data storage.

Data centers losing clients to the cloud, became vendors to the cloud providers for load balancing cloud storage and bandwidth. Supercomputers and mass storage devices now hosted virtual LANs with virtual machines, including the server OS, for about \$60 per virtual server machine instance per month.

Virtual server machine instances can be fired up quickly when activity overloads existing servers, and they can be shut down when not needed. The cloud charges by the minute for running instances, so starting and stopping VM instances have a minimum cost effect on large companies. Some of the largest cloud customers, the credit card processors, fire up tens of thousands of VM workers during busy times of the day. The days of overloading web servers are over.

Beyond the world moving to a virtual server environment, access to virtually unlimited data storage with fast and easy access to and from



virtual servers was not only amazing to developers but safer. VM servers and data could now be replicated to other physical locations in the event of a cloud data center going down or the loss of data.

Newer technologies will have greater need for data and the World does not have enough storage space as is. A technology firm here in the US has developed a new brain scan that tracks brain waves in real time. The firm claims that a single brain scan would need more data space than available throughout the World.

At this time of my life, I was a coder without a cause, which could be dangerous to the World. I was coming up with great ideas for new web applications without planning out marketing strategies. I wrote an organization collaboration portal, a way cool information search portal and a hotel movie pay-per-view platform for small motels.

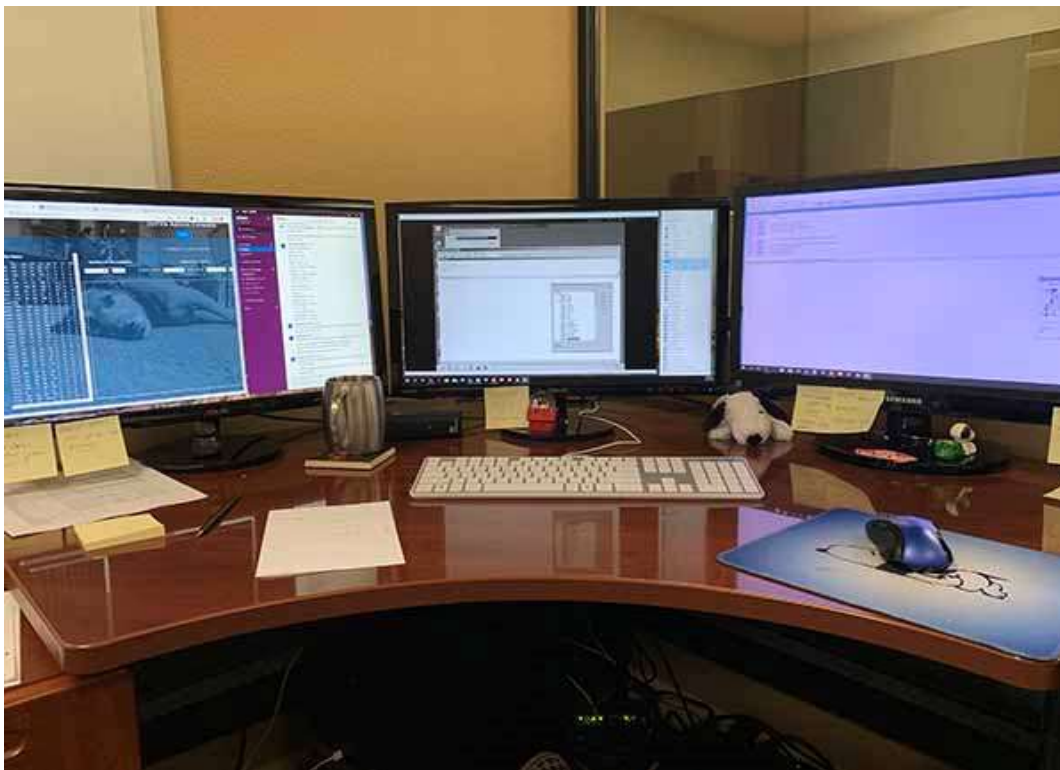
I write solid code and the applications were tight, but they went nowhere fast. We really didn't have the social media exposure that we have today and I'm the worst salesman. I was in a rut and all I knew was to write code to fix my worries. I lost everything, and when this happens, it happens fast with no time to bounce back. I actually slept in my storage unit for few nights and sold everything I owned to stay off the streets.

My daughter had just graduated from high school and I'll always be bothered by the fact that I wasn't able to provide for her in the manner I should have that year. I was always in the code and wanting the next challenge, rather than being financially stable for my family. With the amount of code I've written in my life-time, you'd think I'd be rich, wrong! My skill set will always make me more than most, but I never concerned myself with my retirement, college funds or savings.

The year is now 2014. North Korea continues missile and nuclear tests despite international pressure. Sony releases the PlayStation 4 gaming system while Microsoft releases the Xbox One. Twitter goes public and Popular movies released in 2013 include Iron Man 3, Star Trek into Darkness, The Great Gatsby, Man of Steel, World War Z, Anchorman 2, American Hustle and the Wolf of Wall Street.

One of the people I trained back in the 90's, ended up leaving police work and started a software company with a successful niche, which was somewhat unique to us Foxpro developers. He needed a Foxpro developer and I was without work at the time, so I moved up to Northern California to do what I do best, write code.

Modern age PC configurations, CPU speeds and access to high speed data access required we coders to enhance our workstations....



Our servers were still in a data center cage and Foxpro had limits to table size, 1 billion records or 2 gig file size. I developed a solution using **XML** and **JSON** text files to store heavy pieces of data rather than bloating tables. I would store header data that was searchable in the Foxpro tables, then store the data set for the record in a JSON package to disk. The file name of the JSON package would be the PK, or unique ID of the record in the table.

Access to the JSON data was fast. I would file to string the JSON package then create a cursor of the data, using it anyway needed. The only real concern with this method of storing data, is that Windows OS file indexing becomes slow when there are large number of files in folders. So, I created a data farm, a folder tree that limited the total number of files in any given folder.

I wanted to create a data farm that could retrieve a json package through the folder structure by its PK (unique record ID) alone. Let's say the PK of the json package I was seeking is "123456789". I would parse out each digit of the PK in order, then create a folder structure as follows; f:\data\1\2\3\4\5\6\7\8\9, then store the json package as f:\data\1\2\3\4\5\6\7\8\9\123456789.json. This would allow me to retrieve data just by knowing the PK, and kept the number of files in the data farm folders to a minimum.

Better yet, I'm now storing these json files, along with most all our digital files, on [Amazon AWS](#), at minimum costs. Data buckets don't have the folder file limits and you can unlimited access to storage space. Real-time access these files are extremely fast, providing optimum backend performance for web applications.

Bandwidth for businesses was much cheaper now. 20 meg down and 20 meg up speed for under two hundred dollars a month, and on fiber to boot. Making HTTP requests to other servers for data was a normal routine. Web pages were able to make multiple [API](#) calls at the same time as well. The PC based WAN environment was finally achieved and now the IT World bounces data everywhere, 24/7.

Back in the day, websites were driven by the backend logic. In today's world, with access to complex [JavaScript](#) libraries, front-end developers are capable of building responsive pages that can include business rules and validation. This provides dynamic user experiences but makes validation and the job of quality assurance more complex, for such requirements must be maintained both on the front-end and back-end.

Code libraries such as **NODE.js**, solves this problem for coders can use the same libraries for both front-end and back-end development, allowing validation code to remain in the same piece of code.

The major code platforms, such as Java, JavaScript and **PHP**, now use much smaller code libraries in which coders can include as needed instead of having to load large libraries at website or web server application start-up.

### **The death of the personal computer**

As the need for new hardware, servers and OS updates stacked up for our rack in the data center, the more we pondered a move to the cloud. Turning off our server boxes, entrusting our data to the cloud, and having a physical disconnect from our technology would be a psychological barrier to overcome. The big question was, why?



Converting a dozen servers to a virtual environment would mean no more hardware downtime, hard drive failures and required OS upgrades. The virtual LAN and access from the IIS servers to data would be 4 times faster than in our current data center rack. Our virtual hard drives would be backed up automatically to remote locations. A full suite of virtual services and the ability to fire up new servers within minutes, from most any beach in the world, at one half the cost of just the rack rental at the data center answered the question without a doubt.

It took a couple weeks to replicate our data center rack to an Amazon cloud environment and a few days to move the data. A few rounds of testing, then some DNS routing and our entire business model was cloud-based.

## A virtual-server-rack on the Amazon cloud environment













<div> <div>Launch Instance</div> <div>Connect</div> <div>Actions</div> </div>							
<div> <div>Filter by tags and attributes or search by keyword</div> </div>							
<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
<input type="checkbox"/>	Test1	i-0e2bd9a72c8e3fa8d	t3.small	us-east-1a	stopped		None
<input type="checkbox"/>	WebtoJax	i-0042796fd4cda4ffb	t3.medium	us-east-1a	running	2/2 checks ...	None
<input type="checkbox"/>	Web1	i-0091e0b0cf3860ce2	t3.medium	us-east-1a	running	2/2 checks ...	None
<input type="checkbox"/>	chuck	i-0424adff1a779d023	t3.small	us-east-1a	running	2/2 checks ...	None
<input type="checkbox"/>	jason	i-0842cddc5f87dd39d	t3a.small	us-east-1a	running	2/2 checks ...	None
<input type="checkbox"/>	Appserver1	i-089d07a66561b737c	t3.medium	us-east-1a	running	2/2 checks ...	None
<input type="checkbox"/>	wcweb1	i-09e1c5ca19f6f10dc	t3.medium	us-east-1a	running	2/2 checks ...	None
<input type="checkbox"/>	chris	i-0a520f3d2a8a05bf4	t3.small	us-east-1a	running	2/2 checks ...	None
<input type="checkbox"/>	DataServer	i-0d6a575fd0f16fe96	t3.medium	us-east-1a	running	2/2 checks ...	None
<input type="checkbox"/>	Lbbroken	i-0de0ade49ff33629e	t2.micro	us-east-1a	running	2/2 checks ...	None
<input type="checkbox"/>	JTest	i-0dea4621b3ac984b9	t2.nano	us-east-1a	running	2/2 checks ...	None

## Virtual-hard-disk management on the Amazon cloud environment

<div> <div>Create Volume</div> <div>Actions</div> </div>									
<div> <div>Filter by tags and attributes or search by keyword</div> </div>									
<input type="checkbox"/>	Name	Volume ID	Size	Volume Type	IOPS	Snapshot	Created	Availability Zone	State
<input type="checkbox"/>	JTest	vol-0133ff29...	500 GiB	sc1	-	snap-032ecba...	August 27, 2019 at ...	us-east-1a	in-use
<input type="checkbox"/>	Dataserver	vol-00e91dc...	60 GiB	gp2	180	snap-07c67e1...	August 6, 2019 at 3:...	us-east-1a	in-use
<input type="checkbox"/>	test1	vol-02091bb...	60 GiB	gp2	180	snap-0cce470...	August 7, 2019 at 8:...	us-east-1a	in-use
<input type="checkbox"/>	wcweb1	vol-02c3294...	60 GiB	gp2	180	snap-0d73b7e...	August 12, 2019 at ...	us-east-1a	in-use
<input type="checkbox"/>	Appserver1	vol-0507fd72...	60 GiB	gp2	180	snap-004babe...	August 7, 2019 at 2:...	us-east-1a	in-use
<input type="checkbox"/>		vol-088b693...	8 GiB	gp2	100	snap-05c47a3...	August 27, 2019 at ...	us-east-1a	in-use
<input type="checkbox"/>		vol-0904442...	8 GiB	gp2	100	snap-05c47a3...	August 6, 2019 at 1...	us-east-1a	in-use
<input type="checkbox"/>	Dataserver_...	vol-0c4f36cb...	500 GiB	gp2	1500		August 13, 2019 at ...	us-east-1a	in-use
<input type="checkbox"/>	jason	vol-0c61964...	60 GiB	gp2	180	snap-0d73b7e...	August 13, 2019 at ...	us-east-1a	in-use
<input type="checkbox"/>	chris	vol-0c9c02d...	60 GiB	gp2	180	snap-0d73b7e...	August 12, 2019 at ...	us-east-1a	in-use
<input type="checkbox"/>	Web1	vol-0d2b3b7...	60 GiB	gp2	180	snap-03e30e2...	August 8, 2019 at 1...	us-east-1a	in-use
<input type="checkbox"/>	WebtoJax	vol-0dc8a80...	60 GiB	gp2	180	snap-03e30e2...	August 8, 2019 at 2:...	us-east-1a	in-use
<input type="checkbox"/>	chuck	vol-0f8d549f...	60 GiB	gp2	180	snap-0d73b7e...	August 12, 2019 at ...	us-east-1a	in-use



## The Amazon suite of virtual-services

<b>Compute</b> EC2 Lightsail <a href="#">↗</a> ECR ECS EKS Lambda Batch Elastic Beanstalk Serverless Application Repository	 <b>Robotics</b> AWS RoboMaker	 <b>Analytics</b> Athena EMR CloudSearch Elasticsearch Service Kinesis QuickSight <a href="#">↗</a> Data Pipeline AWS Glue AWS Lake Formation MSK	 <b>Business Applications</b> Alexa for Business Amazon Chime <a href="#">↗</a> WorkMail
	 <b>Blockchain</b> Amazon Managed Blockchain		 <b>End User Computing</b> WorkSpaces AppStream 2.0 WorkDocs WorkLink
	 <b>Satellite</b> Ground Station		
<b>Storage</b> S3 EFS FSx S3 Glacier Storage Gateway AWS Backup	 <b>Management &amp; Governance</b> AWS Organizations CloudWatch AWS Auto Scaling CloudFormation CloudTrail Config OpsWorks Service Catalog Systems Manager Trusted Advisor Managed Services Control Tower AWS License Manager AWS Well-Architected Tool Personal Health Dashboard <a href="#">↗</a> AWS Chatbot	 <b>Security, Identity, &amp; Compliance</b> IAM Resource Access Manager Cognito Secrets Manager GuardDuty Inspector Amazon Macie <a href="#">↗</a> AWS Single Sign-On Certificate Manager Key Management Service CloudHSM Directory Service WAF & Shield Artifact Security Hub	 <b>Internet Of Things</b> IoT Core Amazon FreeRTOS IoT 1-Click IoT Analytics IoT Device Defender IoT Device Management IoT Events IoT Greengrass IoT SiteWise IoT Things Graph
<b>Database</b> RDS DynamoDB ElastiCache Neptune Amazon Redshift Amazon QLDB Amazon DocumentDB			
<b>Migration &amp; Transfer</b> AWS Migration Hub Application Discovery Service Database Migration Service	 <b>Media Services</b> Elastic Transcoder Kinesis Video Streams MediaConnect	 <b>Mobile</b> AWS Amplify Mobile Hub AWS AppSync	 <b>Game Development</b> Amazon GameLift

The thought of never touching hardware again made me realize that the evolution of the personal computer was close to over, the death of the PC was soon to come. Desktop and hand-held devices will soon communicate, then emulate cloud-based desktop sessions. All logic will be processed in the cloud. Full circle back to WAN environments used by the main-frame world prior to the introduction of the personal computer, with amazing power, and capacities soon beyond our imaginations.



## **My Story Concluded**

So, I was at the right time and at the right place, with some crazy experiences, to have a story to tell. A story that I couldn't write into a novel and would require a substantial number of illustrations. The chance of getting a publisher to publish my full-color story would be nearly impossible.

My solution; to develop a web service that would allow authors to create multimedia e-books with ease, with a pay-to-read viewing portal. Technology books and manuals require access to the market much faster than publishers can release such works to the public. My final and best work, utilizing all my knowledge and experience to offer such a needed application for education. I couldn't think if a better way to end a career in programming.

While building the [cjb Bowen.com](http://cjb Bowen.com) web service, I realized not only amazing technology was available to the general public, but also affordable. I built an entire application that operates for under one hundred dollars a month, which would have cost thousands not that long ago.

Anyone can build websites and have them hosted for \$10 a month. Domain name ownership can be as low as \$7 a month. You can have a powerful VM server hosted at Azure for only \$40 a month. You can decrease power and pay as little as \$7 a month or increase power as needed.

There are now credit card merchant services with complete API gateways, so you can integrate the payment process with ease, for no monthly costs at all. SSL https encryption for your website is free in today's world.

Have access to company telephone systems, which can be used from anywhere, for \$20 a month, and telephone equipment that costs under \$50 per device. Company email services are as low as \$6 a month per user and access to Microsoft Office is as low as \$8 a month per user.

Cell phones are now mini personal computers with access to applications to manage all your business resources on the go. I can access any of my servers via the **LogMeln** app on my iPhone. I can basically be anywhere in the world and manage my online business.

The past 36 years has been a blast! I was a cop, ran dental offices, built sterilizers, owned a radio station, become famous in my own mind, filmed concerts, wrote books and wrote millions of lines of code, enhancing the day-to-day lives of people using my software applications.

For the past twenty years the world has been archiving data, images, videos and stories regarding everything on earth and its history, on the Internet. It's going to be important for us to tie all this information together, such as I have done in this e-book, before we lose the current generation, or all this data will begin to have less and less meaning to our future generations.

I use the "Don't Do This at Home" disclaimer, for I feel I should have stuck with a project and secured my future financially, rather than driving in the fast lane of code development.

I see tons of new technology just over the horizon. One side of me can't wait and the other, older and wiser side of me, just wants to slow down and enjoy the ride, like the rest of the world. The goal of writing this book was to give the younger generation a better understanding of how they got what they have at their fingertips today. Understanding the past, helps for a successful future.

The End

1bed4all  
homeless rehabilitation project

**1BED4ALL.ORG**

