

# Ample Power Company



## Technical Documents, Manuals, App notes

---

This Ample Power website first went online in the spring of 1997. It was a struggle getting the Ample Power Primer online for a couple of reasons. First, there was our own ignorance about HTML, the mark-up language of Web documents. Secondly, the tools for generating HTML from standard text editors were weak at best. And complicating matters was the fact that the Primer, our catalogs and books are in a different mark-up language most will not recognize. To get up our first Web offering meant extra labor stripping out mark-up commands by hand and then running the plain text through a Windows<sup>1</sup> based program to generate HTML. This brute force method was not satisfactory in the long term because it was too labor intensive, and meant that two sets of documents would have to be maintained, one for printed output, and one for Web content. Keeping two copies of the same material violates a basic principle of document management

### In the Beginning

In the beginning all we ever wrote were programs and simple letters, so a programmers editor was all we needed. We began writing the book *Living on 12 Volts with Ample Power* while cruising in 1983–84. (Yes, we had a personal computer on board, much more powerful than the original PC that followed.)

In 1985 were ready to take our plain text files for the book and convert them to a format for publishing. By now we were using a PC because despite it lack of horsepower and clunky operating system, it had become a standard. A friend who was a technical writer recommended a program that he was familiar with. We ended up calling it *word scum*. Not just because it crashed a lot. The program wanted us to think about fonts and sizes for all the headings, worry about placement of figures, and construct tables with a great amount of labor. What good is a computer if it's no more than a glorified typewriter?

At the time we were writing some schematic editing software. Although we were using PCs, they were tied to a *back room* computer running Unix. And like most Unix systems of the era, it was connected to the forerunner of the Internet. It didn't take long to discover the publishing program I wanted, called T<sub>E</sub>X. It was authored by the well known Donald Knuth

at Stanford University.

Running as a set of 'macros', or super instructions on top of T<sub>E</sub>X is another program called L<sup>A</sup>T<sub>E</sub>X. Written by Leslie Lamport, L<sup>A</sup>T<sub>E</sub>X allows an author to write without worrying about structural details ...L<sup>A</sup>T<sub>E</sub>X can build a letter, article or book with tables of contents, list of figures, etc. That meant I could take my plain text files and with minimal markup, produce formatted output!

While L<sup>A</sup>T<sub>E</sub>X was freely available on Unix systems, we couldn't afford such a system at the time and therefore had to buy an expensive version and run L<sup>A</sup>T<sub>E</sub>X on our lowly PC. Our 1985 Compaq computer, with a 10 Mbyte hard drive ground for 4.5 hours making the final output for *Living on 12 Volts with Ample Power*. While each year brought faster computers, and software upgrades for our PC version of L<sup>A</sup>T<sub>E</sub>X, we knew from following the T<sub>E</sub>X community that the real way to run the program was under the Unix operating system.

### Part of a Tangled Web

In 1991, a new operating system started growing over the Internet. Linus Torvalds, a student at the University of Helsinki in Finland made an Internet posting that captured the imagination of programming wizards all around the world. That seed flowered into Linux, an operating system that is faster, and more stable than the typical Unix. While Linux is available at no cost to anyone with Internet access, it's chief attraction isn't the price, but the tools that are available for it ...also available by simply downloading them.

Running Linux meant we could use T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X in it's native environment. All the packages that have been added to the T<sub>E</sub>X community over the years would be available to us immediately, without waiting for support to be added by the vendor of the PC version.

We began exploring Linux, and by 1994 had tossed out the PC for good.

### So Where Does this Go?

So what does all this have to do with this Web Site? The Web runs on HTML, or Hyper Text Markup Language. We have years of work in L<sup>A</sup>T<sub>E</sub>X, another type of markup language. But we're not alone. While the average computer consumer has been on a diet of Apples and Windows, universities and

---

<sup>1</sup>Windows may be a trademark of Microsoft, Inc.

large organizations have used some variant of Unix. Thousands, (millions?) of books and articles have been published using  $\LaTeX$ . It had to happen, and it did ...a converter that takes  $\LaTeX$ input and pumps out HTML. As you'll see, we have used that converter to power our Web site. We were able to find the source code in the UK, download it, install it, and run our first test case in less than an hour!

The converter to make HTML out of  $\LaTeX$  is written in a language called Perl. Perl, written by Larry Wall, is another tool available on the Internet at no cost. At one time, we read that Perl was being used by over 90% of the Web servers on the Internet. Perl excels at text interpretation and translation into computer actions.

### **We're not in the Clear.**

There was one other major hurdle we had to clear before we could really move our  $\LaTeX$  material to the Web. Early photos were scanned into a PC and saved as Windows .bmp files. Schematics and block diagrams can be output as PostScript, (.ps), or Encapsulated PostScript Files, (.eps). Such material on the Web is usually in Graphics Interchange Format, (.gif), or image compressed format, JPEG, (.jpg), now .png.

The first version of the Ample Power Primer Online, which may still be in use as you read this, managed to get system wiring diagrams into .gif files by first displaying the original PostScript files on the screen, and then doing a screen capture. It was done using a lot of patience and brute force work.

You can imagine our excitement to find on our Linux system a program, with a modest name of xv, which can translate between just about any format! Not only could we use the Postscript wiring diagram files, we can also translate the Windows .bmp photo files into .gif, or even .ps to use for printed pages. We won't bore you with details about make utilities, but with a *single source document* and a make file, we can produce one output destined for the printer, and another output readable on the Internet.

### **One more Stretch**

One hurdle remained. How do you transfer files from a Windows 3.1 machine to a Linux machine. Yes, Linux can read DOS files on the floppy drive, but many of our photo files were greater than the capacity of a floppy disk. The answer lies in TCP/IP, and SAMBA. TCP/IP is the underlying protocol of the Internet, and despite a slow start supporting the Internet, Windows software became available to run TCP/IP.

SAMBA is a program that allows a Linux and Windows machine to communicate using a conventional Ethernet interface. We been using such a network for our internal network. First we experimented with hooking two Linux machines together on a small subnet ...a piece of cake. Getting SAMBA to work between Linux and Windows wasn't quite so easy,

but eventually the pieces meshed.

If all this seems like a lot of effort to re-use old material, and to continue publishing printed documents with the familiar  $\LaTeX$  tools, consider this ...one source file of text and graphic file names can be 'made' into a printable document, or a Web document without changes! But yes, it was a lot of effort, and we couldn't have done it without the contributing efforts of thousands of programmers around the world.

### **The Bottom Line**

It seems the bottom line is always cost, so why should it be different here. How much did it cost? All the software except xv is available at no cost if you have Internet access. Because we planned on using xv for commercial purposes, a registration fee of \$25.00 applied ...a bargain if there ever was one.

This doesn't tell the whole story, of course. To be fair, we'd have to include the cost of a mighty stack of books on  $\TeX$ ,  $\LaTeX$ , Perl, Unix, and Linux. We have every issue of Linux Journal, so the cost of that subscription over the years should be included. And then there were a few CD distributions of Linux programs we 'played' with that cost from \$15.00 to \$49.00. Maybe this all adds up to \$600.00.

And what about the time to learn how to use all these tools? There's a learning curve for any program, so learning how to use  $\LaTeX$  or xv is not much different than learning how to use any commercial program. Getting Linux to install on a standard PC is not terribly difficult with the latest distributions. We complicated our life by first trying to install it on a laptop in 1994 before there was full Linux support for laptops.

### **It's 2008 Already?**

Linux has become a powerhouse. We now use a distribution called Ubuntu. See also the meaning of Ubuntu.

Installation of Ubuntu is easier than Windows . . . plug the CD in and answer a few questions. The standard Ubuntu installation comes with  $\TeX$  and  $\LaTeX$ . It also includes compatible programs to use Windows files for documents and spread sheets. Other free programs can be installed by selecting from a menu. We recently downloaded pdflatex that converts  $\LaTeX$  files to PDF format. We used that to generate projector pages for a seminar.

However, the website using pure output from the  $\LaTeX$  to HTML converter was looking dated and was difficult to navigate. Yes, there are publication frameworks available, but they expect data to be entered within their context. We were not going to regenerate our website if it meant converting existing material to another format. And we don't want give up the convenience of having one document that can translate to HTML, or PDF. (In fact we routinely produce both and give the viewer the option of downloading the PDF version.)

A set of navigating links along the top that were the same for every page would allow the viewer to jump categories with a single mouse click. Links along the left or each page would allow a viewer to quickly access links relevant to the present page . . . to click between the data sheet and manual.

### What's a Programmer to Do

Not finding any tweaks for the  $\text{\LaTeX}$ to HTML converter that would allow placement of top and side links there were two options. One, we could hack on the source code from the converter. But that's written in Perl, and we know just enough about Perl to think we'd hate having to program with it.

The second option, and the one we chose, was to do post processing on the HTML file after it has been converted from  $\text{\LaTeX}$ . That turned out to be a fairly easy exercise using Python, another programming language that we like a lot.

But there was another *mess* that begged to be cleaned up. All programmers are familiar with *make* and *Makefile*. The contents of a Makefile describe what actions are necessary to convert a source file to one or more output files. The verbage to do that is arcane, at best, and very easy to make mistakes and even harder to find those mistakes.

So I tossed it out and wrote a new program called Pmake, where the P is shorthand for Python. Instead of a Makefile that is interpreted, I named my description of what to do simply *config.py*. No, Pmake is not a general purpose replacement for *make*. It is designed to make PDF and HTML documents from a  $\text{\LaTeX}$ input according to the specifications in *config.py*. And there's an option to *sync* the local machine with one or more remote web servers.

After editing this source file I'll first type the command *Pmake*. Once *Pmake* tells me it's done, all I have to do is hit the up arrow to bring back the *Pmake* command and type *sync* after it and hit ENTER. Shortly this will appear on the webserver.

Our office is in Seattle, while the webserver is located in Pittsburgh. We could run an in-house server and use the same programs that an Internet service provider does. But, we're at the end of a dedicated, but slow DSL circuit. Without a very fast connection, hosting our own web server is not realistic.

### Adding an Internet Store . . . 2009

Besides bandwidth, an ISP can provide security. After the security breaches of the last couple of years where millions of credit card details have been hoisted from websites, the credit card companies have gotten downright persnickety about e-commerce sites. Even if we had bandwidth for an in-house server, meeting rules of the credit card companies would be difficult.

Besides putting up a web store, we wanted to improve customer support. There's a lot of troubleshooting information on our website, but it isn't as focused as it could be. The result is emails to support from various users with the same of nearly the same questions. What is needed is the latest version of the old bulletin board.

Our website provider must have been reading our mind, because along came a notice one day that they were now offering *virtual servers*. A virtual server operates as if it is an actual hardware device, although one machine is running the virtual servers. It's a lot like owning your own machine but running within their network. They have to maintain it, provide backup, security, and handle software updates.

So we signed up! They installed the storefront and it's being populated by us. I installed the bulletin board support forum and it's being used now. It was a busy week.

### Buying from our Static Pages

If you're reading the tech data on one of our products and you want to buy one, should you have to go to another URL and search for that product in the store? Of course not.

Adding *add to cart* buttons into our tech pages was plain simple.

### A Great Read

In the Beginning was the Command Line is too good not to link here. If your computer experience is limited to pointing and clicking at icons provided by someone else's software, the command line is foreign to you. And, you've been brainwashed to think that the graphical user interface, GUI, is simpler than the command line.

If the task is graphical in nature, such as generating some artwork then the GUI is a perfect fit. Otherwise, the command line is often quicker.

But read the article, and don't miss the ending.