Typography notes: I suggest a sidebar in several places. The sidebar starts with [!! and ends with !!]. I tried to write the text so that it works even if you don't make a sidebar. Footnote suggestions are [footnote: .... ] Delete if desired/necessary.

I've moved tons of stuff around from any previous version you've seen.

It's completely in time order now and should flow much better.

---

I've been giving away my software since 1983, full-time since 1991. I don't do it for fun, although I enjoy it. I do it because it's a way for a small business to earn money AND it's fun. It started as a hobby, and I turned it into a profession. That started dying out, so I turned to another hobby and turned it, too, into a profession. Not every hobby of mine has turned professional, and I hope to explain why some have and some have not.

Three of my hobby projects have turned profitable. I'll talk about them in depth after I introduce myself. They are Freemacs, Packet Drivers, and qmail[1]. Freemacs is an MS-DOS text editor, styled after Emacs. It's still being used today as the official editor of the FreeDOS project. Packet Drivers hide the difference between Ethernet cards in an MS-DOS system. If you've ever eaten at a McDonald's restaurant, your order was communicated through Crynwr Packet Drivers. Qmail is a mail transfer agent (MTA) for sending and receiving Internet mail. Qmail is the engine behind Rediffmail's 30-million-user, multi-terabyte, hundred node email cluster, and many smaller sites.

## Introduction

I had been doing hardware hacking long before college. Digital electronics was too expensive for me: $1 per TTL quad nand gate at a time when a record only cost $5. So I fiddled around with analog electronics. I invented a trigger sweep for my dual-beam oscilloscope, and an analog computer throttle for my model railroad.

Our high school was a member of LIRICS: Long Island Regional Instructional Computer System. They had a PDP-10 which students could use to learn to program via teletypes operating over modems. I was at Baldwin Sr. High School from 1972 to 1975, but only used it my last year, from 74 to 75. Learned basic, wrote a basic four-banger calculator program. Wrote a word processor in basic. Did all sorts of horrific string manipulation. Took hours to format a two-page social studies paper. Part-way through I got an "IM" from an operator who asked me what I was running, and if it

---

1 qmail is an all-lowercase name, and will only be capitalized here at the beginning of a sentence.

was looping.

Learned PDP-10 assembly language during this period. JRST, HRRLZ, and SKIPNE are all familiar friends to me. Unfortunately, none of my candidate colleges had a PDP-10. MIT almost certainly did, but I was a poor scholar. I was too interested in getting an education than in proving that I had. Of course everything on the PDP-10 was open source. Nobody thought of holding back the source code in those days.

Various colleges will teach you how to become a businessman. I didn't have that desire entering college, and so I sought a degree in in Electrical Engineering. Only later did I conceive of running my own business, but how to learn how? The business world is happy to teach through experience. I started slowly, and didn't take too many risks.

I learned "on the job". I discovered new ways to profit from open source software. Everyone in the business had to teach themselves. There is no Master's of Open Source Business Administration – not yet anyway.

Hewlett-Packard recognized my genius (HAR!) and hired me and my wife to work in their Calculator Division doing integrated circuit design. I missed programming, so I bought a Radio Shack Color Computer (CoCo).

I wrote programs for fun, and sold them to CoCo Magazine for distribution. Just little cute things, but they were in assembly language so they were fast, small, and easy to distribute. The standard distribution was on audio cassette through a paid subscription. It was nice to receive money for writing a program for fun. Without a local user community it was also the only way I could distribute my software.

## Freemacs and Open Source

After I returned to graduate school, I started writing a programmable editor for MS-DOS. Rather than write it de novo, I thought it should be compatible with Emacs. I had used Emacs while working at Hewlett-Packard, had done some hacking on an editor there, and that experience convinced me that I wanted an editor without a distinction between editing and typing. I purchased a copy of the MIT AI Lab memo describing Emacs, written by Richard M. Stallman (RMS). I recognized his name because of his GNU Manifesto, a then-new document. I sent email to RMS to get permission to sell a copy of the memo along with my editor. The document was in the public domain, but I thought asking permission was only polite.

I got a call from RMS (this was back when his wrists hurt so badly he couldn't type). He persuaded me to give away my version of Emacs rather than sell it. He appealed to my sense of fairness. He asked why I should profit a manual which he wrote, not me. I was impressed that so stellar a personage as RMS would take the time to call me. I decided that it would be best to give it away. I didn't really have any idea who I might sell it to. All my previous software sales had been for the Radio Shack Color

Computer, and sold to CoCo Magazine. My editor was for MS-DOS, so this was a completely new situation for me. Once I decided to give it away, I gave it the catchy name of 'Freemacs'.

Being in graduate school, I had access to worldwide networks, and it was actually possible to distribute software "for free". Nobody was kidding themselves; the Department of Defense was paying the bills for the ARPAnet, universities for BITNET, and companies for CSNet. Nothing was free to the institution, but the users perceived the networks to have no incremental cost. This led to allocation by congestion, but I'm not talking about economics yet. Regardless, a software author could give away software for the mere cost of uploading it to a distribution point.

Freemacs was distributed from SIMTEL-20.ARPA. It was an FTP site which had a copy of most useful MS-DOS software. It was run by the Army at the White Sands Missile Base, but nobody cared about it. The point was that they had good stuff, and they were sharing it. I was now a contributing member of the open source community.

## Freemacs and Business.

Freemacs was a hobby, and I had no intention to turn it into a business. My computers (even my home computers) were paid for by my employer, so I had no expenses to cover. As the program gained users, they told other MS-DOS users about it. Those users wanted updates, and were not on any of the worldwide networks of the day. Having no other recourse, they asked me for copies. I knew that they were gaining a benefit from these copies, so I asked for a portion of those benefits in the form of a copying fee. Between 1985 and 1991, most of the activity of Crynwr Software consisted of putting software on floppy disks and mailing it to customers.

## Staying in touch

It's crucially important to stay in touch with your users. The biggest advantage that an open source developer has is close contact with users. If you're the primary user for the software, then of course you know what users want--you just sit and cogitate. Quite a bit of open source software is written to "scratch your own itch".

This is easy and rewarding (most programmers fall into one of the four **NT** classes on the standard Meyer-Briggs personality test) because very little communication is needed. Programmers are typically not great communicators. A programmer who can listen and talk is worth her weight in chips (chips are worth more per ounce than gold).

With Freemacs, I had several mailing lists. Initially I had only one, and that was a mistake. Some people don't need any help and just want to know when new releases come out. Other people want to get or give help but don't want to develop. And still

other people are interested in every miniscule detail of the program. One list cannot serve everyone. I had three lists.

One list carried only announcements of new releases. You really, really want to have an announcements list, and you need to remember to use it. You might only send one or two pieces of email a year, but those are crucial. First, you need to remind people that they've given permission for you to send them email. Second, people need to know that you're in business even if they don't currently need your business. Any user might suddenly find himself needing to become a customer. You need to be the proprietor of the relationship between the software and the user, as you'll use that relationship to make money.

A second list was for user-level help. Some programs are exceptionally powerful (I'm not thinking of the Unix 'cat' or 'tail' here, but more like 'sendmail' or 'qmail') and need a depth of knowledge to properly exploit all that power. Some users will care to acquire that depth. Others will not, but will instead dip their toe into the depths by asking a question on the user mailing list. If there are sufficient users of the program, you will have other businesses competing with you. One of the ways they will compete is by offering to help other people. No need to worry, though! By virtue of your proprietary interest in the program, you will have a built-in advantage over these other businesses. In any case, customers like competition because they perceive it as ensuring fair prices.

The third list is for developers. I am of two minds here. You could have the developers list be open to all comers regardless of their ability to contribute to the project. Or you could have the developers list be open only to those who have actually contributed.

The main trade-off is protecting the time and attention of your contributors. You don't want them signing off the developer mailing list because it has too many user-level questions being asked on it.

You really need their attention to help you make decisions that will affect them. For example, if you change an Application Programming Interface (API), you need to clear that change with your developers.

First, because it keeps them involved in the process, and second because they may be relying on something you're doing.

!!]

Two interesting stories about mailing floppies: One is about a customer in Ireland who had two floppies go bad on him. Guessing that his email was going through some kind of anti-terrorist scanner (as the Irish Republican Army was qutie active at the time), I sent him a third floppy wrapped in 1mm-thick lead foil. That floppy got through okay. Another is of a customer who, although a part of the defense department, had no Internet access, nor did they even have a modem--and this was

when almost everybody had gotten on the Internet. I was surprised that they were even allowed to telephone out, but I sent them a floppy with software on it and they were happy with that.

FreeDOS has adopted Freemacs as its standard text editor, so it still has a user base. I only rarely do any MS-DOS work, and when I do, I'm happy with the state of Freemacs, so it's now frozen in time. There were never any commercial users, so apart from selling copies on floppies, Freemacs only managed to buy me my own computer for home.

## Packet Drivers

Crynwr Software came into its own with packet drivers. A packet driver was needed to share an Ethernet card between two protocol stacks. For about a year's time, the only possible way to get Novell network clients on the Internet was using a packet driver. Also, a packet driver would hide the difference between Ethernet cards.

Unlike video boards, which are at least compatible at the VGA level, Ethernet boards have never been compatible.

Back before packet drivers existed, there were network clients and servers – largely Novell Netware. The manufacturer-written network driver was linked to Novell's code into a single executable. The resultant program provided NO API for an external program to send or receive an Ethernet packet. This was bad, bad, bad, for any competition to Netware. Maybe Novell planned it that way, but I doubt they were that Machiavellan.

Anybody wanting to send packets other than Netware's had a problem.

The 3Com 3C501 was the market leader, and a very insufficient card it was. It had one buffer shared between transmit or receive, so a packet could be lost if it arrived when the buffer had not been emptied, cleaned, and turned around. However, everybody had drivers for it. Novell, in an attempt to improve the state of the art, took National's 8390 demo board and put it into production as the NE1000, and later the NE2000. This board had sufficient memory with separate transmit and receive buffers.

Just about then other Ethernet controllers were coming on the market. People were using the Intel 82586, the AMD LANCE, and the National 8390 (in non-NE2000-compatible ways). Only Netware had a device driver development solution. They had a driver development kit (DDK), and a certification house (Novell Labs).

Other protocol stack vendors were doing the same thing – producing a driver linked into their own product. Nobody had a driver that could be shared. While Novell and Microsoft pondered the problem, little FTP Software (now owned by Netmanage) had the same problem as everyone else. Too much hardware, too few drivers. So they came up with their own specification for a shared Ethernet driver, and, unlike other

vendors, published it as an open standard.

I was working for Clarkson University at the time, and we had the same problem as everyone else: how to support multiple pieces of software and hardware at the same time. I was using Phil Karn (KA9Q)'s NOS, and he had packet driver support. So I wrote packet drivers for the two Ethernet adapters in use at Clarkson (the 3c501 and Racal-Interlan NI5210), and published them as open source software.

A number of fellow Internet users contributed drivers, and before long, we had a considerable portion of the industry covered. This led to more support from TCP/IP vendors, and a group at Brigham-Young University wrote a Netware driver which could use a packet driver. The ball really started rolling then, because anyone with a Netware network could put it on the Internet.

There were some holdouts, notably Microsoft and Novell, both of whom started promoting their own standards: NDIS, and ODI respectively. The NDIS document was published from the start, but there was no sample drivers, no base of code to build from. ODI documentation was only available with purchase of four figures worth of DDK. Packet drivers distinguished themselves by coming with source code, by having a simple, approachable API, and by being small in size. The typical driver was 5K of executable, compared to 20K of ODI, and 40K of NDIS.

About the middle of 1991, I realized that there was money to be made providing packet driver programming and certification services – the latter for drivers not written by Crynwr. So I left Clarkson, and after a five-month placeholder stint at a local PBX company, I started Crynwr Software. Up until 1998, Crynwr's main source of income was from packet drivers.

## Packet Driver Income

Most likely, everybody has heard that the way to profit from open source software is to sell services. That's true, but there are many different types of services. I'll list some of them in the paragraphs below.

The first, and most profitable, is contract programming. Various people needed packet drivers written, or features added, or bugs fixed. I contract with them to fix it, either for a fixed price if I understand the problem, or else at an hourly rate if discovery is needed. Buyers don't like cost uncertainty. They really really like to know what something will cost up-front. On the other hand, whenever you bid a fixed price, you are taking on the risk that the project will be much harder than you thought.

I have had good success in doing what appears, at first sight, to be the worst of both

worlds: charging per-hour with a minimum and maximum price. If you set the minimum and maximum to reasonably sane values, the risk is reasonably shared between the two parties.

[!!

## Business Tutorial

Maybe you're reading this book because you have no experience in business, being a day programmer and evening entrepreneur. Here's a quick tutorial on how people do business which I wish I had when I started. First, customers expect to do business first, and pay you for it later. This is done essentially by the customer issuing a private currency called a purchase order (PO), with a face value and a serial number called the PO Number. Purchase orders owed to you are Accounts Receivable. Purchase orders you have issued and will pay are called Accounts Payable. I call a PO a currency because you can get a loan against good receivables, and you can sell bad receivables (customers who don't pay, or pay very late) at a discount.

Never do work on someone's promise to pay you. If they're really going to pay you, they'll be able to cut you a PO. People change jobs and companies go into bankruptcy. If you have a PO number, that's as good as gold, because a company's ability to purchase things is dependent upon their reputation for paying on terms. If they lose that ability, nobody will accept a PO from them, and then they have to pay cash for everything. Get that PO!

Some companies have intricate purchasing systems, where you have to be a qualified vendor, you have to sign a W-9, and have to sign a nondisclosure just to work with them. Other companies just whip out the credit card, and you're good to go. For any company larger than fifty people, though, you'll be dealing with a buyer. Most buyers are used to purchasing software as a product, and of course they still do.

They are starting to understand that software can be a service, but you still might run into a confused buyer. Sometimes they're told "buy this software" and they don't understand that they're purchasing a CD and a support contract. Take the time to educate them about the difference, and you'll have an easier time working with them.

!!]

I've also sold proprietary packet drivers. This was a special circumstance, and one which was very profitable to me. I had a customer who wanted a new packet driver, but who didn't want to pay the entire price for it. He only wanted to pay half. He persuaded the vendor (SMC Semiconductors, now SmsC) the pay half, since the packet driver would be useful for all of their customers. That seemed fair to me. I had his purchase order and SmsC's promise. Trouble is that he paid up and they

didn't. So now I had a packet driver that the vendor hadn't paid for. If I made it freely copyable on that basis, then no vendor would bother to pay me. I decided to license it to their customers until such time as SmsC paid me. They never did, so I sold it with a clear conscience.

I've also dual-licensed packet drivers. A vendor was going to embed an Ethernet chip into their product, and use an embedded processor.

They wanted to freely copy code from the packet driver without taking a chance that their driver would become a derived work under the provisions of the GPL. They purchased a copy of the code from me, licensed for any use except resale.

I've also sold compatibility certification. Digital Equipment Corporation had written their own proprietary packet drivers. They wanted me to certify to them that they were compatible with the open source packet drivers. I had written a test program which would exercize the edges of a packet driver to try to break it. If that program ran, then they didn't make any stupid mistakes reading the specification. I also ran a stress test for several days. If that didn't run into any problems, then they didn't make any stupid coding mistakes.

I've also done pure consulting. Contracting is different than consulting. A contractor is someone who sells their work output. A consultant is someone who sells their ideas. A customer wanted me to describe how my packet driver worked on their hardware. They sent an engineer to my site for a day to get a debriefing. He took my family out to lunch, and I got paid handsomely for the day. Not as much as I would have been paid to write the improvements myself, but you can't make ALL the money ALL the time.

## Qmail

Qmail[2] is an MTA (Mail Transfer Agent) written by Daniel J. Bernstein, a University of Chicago professor. He needed a fast MTA to run his mailing lists. While I was working on Freemacs and Packet Drivers, I was also running my own mailing lists.

I experienced the same problems he did, so when he published qmail, I was on it right away. First, I ran it on a test machine, because I hate to lose email. Later, I ran it on my email hub after I learned to trust it. Slowly, the qmail community grew.

Working with qmail is a different experience for me, as I did not write the code. However, for the purposes of this book's chapter, it's good, as it gives me something

2 qmail is not open source. It's freely copyable, but you can't redistribute modified executables. Open source MTAs exist, but qmail is close enough to being open source as to give me insufficient reason to switch.

different to write about. It's a trusim that you cannot sell something you don't own. A number of the techniques that I used for Packet Drivers do not work for qmail, as I don't own the copyright on it. Nonetheless, I have done quite well with qmail.

Open source software serves to promote the author. By writing quality software (code AND documentation), the author shows everyone the quality of their work. The software pushes their reputation out into the world. But what if you haven't written the software? I have found the best technique there is to inhabit user forums, and answer questions. You can advance your reputation in this manner.

There are other ways to establish your expertise with a particular program. You can contribute improvements to the code or documentation. You can write your own software which enhances the software, but which otherwise stands alone. You can write extra documentation. You can maintain a website about the software.

I did all of these things for qmail. Dr. Bernstein was not interested in registering qmail.org, so I did it for him and maintained a website. I wrote a pop3 daemon and gave it back to him. I answered questions on the qmail mailing list. I wrote add-on packages, and patches which people have found useful.

Through all of these techniques, I found paying qmail employment. Most often this consisted of people who needed qmail to be installed on their machine, and had extra requirements which needed custom coding. Sometimes it was on-site qmail training. This work has taken me to Stockholm, Mumbai, London, New York City, Oslo, and Istanbul. World travel, paid for by my customers, while earning money for myself as well.

## Open Source Economics

It was my reputation, or "brand name" if you wish, that got me involved in the Open Source Initiative. I had been running the Free Software Business (FSB) mailing list since 1993, and had some success and reputation. When I heard about the Open Source name, I immediately adopted it in describing my software. It's so much easier to explain to customers why they should pay for software when it isn't "Free Software". Some time after that, I heard that Eric Raymond was seeking to create an organization to promote "Open Source". I had been corresponding with him about open source, and we had discussed it on the FSB mailing list, so I volunteered to be on the board of OSI.

I started the Free Software Business mailing list to be more introspective. I wanted to know what other people were doing to make money from open source. It seems, and seems, as if this is a revolutionary way to make money; by adding value to things other people had created. Shakespeare might disagree, as he routinely "recycled" (stole?) plots and storylines written by other people.

Certainly in the software business, a FSB is a new thing. With proprietary software, about the only way to add value is to sell the software for a higher price, or bundle together products and sell them together for a lower price in toto.

Running a Free Software Business has created in me an interest in economics. When you give up a payroll check, and start paying yourself, you also have to pay double Social Security taxes. You also have to start paying estimated income taxes quarterly. The government imposes this dead load on businesses. Why do they do this? Who benefits? Who loses? Economics help you to answer these questions.

Another useful bit of knowledge from economics is how to price your services. There is an economic concept called "price differentiation". It means that you charge different parties different prices. If you charge a single price to everyone, and that price is too high, you miss out on helping some people who cannot afford your services. If you charge a single price that is too low, you create value for some people beyond the amount they paid. It's not exactly fair that they should gain a lot more than you. To maximize your gain, you want to sell things to people based on the value they receive rather than the cost to you. Everyone has seen software sold for less in a third-world country, or to a school.

Another concept is "transaction costs". Every transaction consumes a small amount of value just in the transaction itself. The buyer needs to evaluate whether the cost spent is worth the value received. The seller needs to take the money and provide the value. This cost is not received as value by either the buyer or the seller; it is simply wasted. One way you to sell support to a customer is on a self-renewing yearly support subscription. One month before the subscription expires, you send the customer an invoice. The subscription portion avoids having to bill the customer for each support request. The self-renewing portion avoids having to ask the customer to renew the support contract.

Another concept is "sunk costs". A sunk cost is one which you cannot recover by selling the thing you have bought. A typical example of a sunk cost is a railroad, or a run of fiber optics cable. You can pull up the rails and sell them, but you won't get back anywhere nearthe cost of building it. Well, spending money to create or improve open source software is a sunk cost. Once you've spent that money, you have no way to sell the software (except by exiting the open source system by using a dual license). Just as a railroad needs to recover its sunk cost by selling transportation services, you need to recover your investment in the software by selling related services.

Another concept is "public good". A public good is non-rivalrous (meaning that my use doesn't affect your use) and non-excludable (I can't stop you from using it). Absent copyright or patent protection, all information is a public good. Open Source is typically copyrighted software, but is licensed under terms that make it into a public good. There is currently great debate over how much excludability is necessary to produce the optimal amount of software.

Previously, economics students were taught that public goods were always underproduced, with lighthouses as the canonical example.

Somebody dug up the history of lighthouses, only to find that the early ones were built by voluntary organizations. In a similar manner, the Free Software Foundation wrote the GNU toolset as a public good. Economists can no longer assume that public goods are underproduced.

Another concept is that of "substitution". People don't particularly care about products. People only buy things and own things for the services those things can render to them. People don't want to own a washing machine; they just want clean clothes. Any desire to own a washing machine is secondary to having the clean clothes. Same thing for computers, only computers can provide many different services. The same services that someone can get from a software product can also come from open source software and a support contract.

There are many principles that economists have discovered, and agreed are true. Yes, in spite of all the jokes about ten economists having eleven opinions. This chapter can but touch on them. See the "For Further Reading" section below.

## Where do we go from here?

I've brought you up to the present of my life. Well, not quite. I received an inheritance from my mother, which, when invested in the stock market, generates sufficient income that I no longer need to work for a living. I still take interesting jobs as they come up, and I support long-term customers because they've become friends, not just customers. In everything else, I just write open source and distribute it as I wish. My advice to you is to always pay your retirement fund first. Put 10% of every project's check into a brokerage account and invest it in an Exchange Traded Fund like SPY or QQQQ.

Happy hacking!

## For Further Reading

- For a basic (and enjoyable) introduction to price theory, read _Hidden Order_, by David D. Friedman. In fact, read anything written by any member of the Friedman dynasty: Milton, Rose, David, or Patri.

- For a basic introduction to economics, read _Economics In One Lesson_, by Henry Hazlitt.

- For a very deep exposition on economics, read _Human Action_, by

Ludwig von Mises. It's available at http://www.mises.org/humanaction.asp

o The Free Software Business mailing list is at http://crynwr.com/fsb.html .

o Freemacs is at http://www.freedos.org/jhall/freemacs/

o Packet Drivers are at http://crynwr.com/drivers/

o qmail is at http://qmail.org/