Writing a technical book with Manning in 2020

Milan Curcic Follow
Dec 30, 2020 · 20 min read

Two months ago I published *Modern Fortran: Building Efficient Parallel Applications* with Manning Publications. It was a long journey (3 years!) and here I describe my experience: The writing process, tools I used, working with Manning staff, the economics of writing a technical book, and challenges. Though the title of this article mentions the year 2020, my experience reflects the period from mid-2017 to late-2020. A few things may have changed in the meantime, but most of it is relevant. If you're considering writing a technical book or have just started doing so, this article may help you.

How it started

Writing a technical book was the last thing on my mind. In May 2017, I got an email from Mike Stephens, the Acquisitions Editor (AE) from Manning. (An acquisition editor finds prospective authors and works out the idea and scope of the book). Mike wrote something like:

> *We'd like to make a book on Fortran programming. We saw some of your forum posts and GitHub repositories. Would you consider writing a book with Manning?*

"Sure thing", I said. In that moment I didn't believe I was cut out for the job, but the idea excited me so I closed my eyes and took a leap of faith.

Timeline

Here's the timeline of key events:

- **May 2017**: First contact with Manning
- **June 2017**: Sent proposal
- **July 2017**: Contract and ISBN issued
- **August 2017**: Began writing
- **February 2018**: 1/3 of the book done and first external review
- **March 2019**: 2/3 of the book done and second external review
- **July 2019**: Third external review (really, second review part 2)
- **February 2020**: 3/3 of the book done and final external review
- **April 2020**: Copy editing
- **September 2020**: Proofreading and final edits
- **October 2020**: Book is printed and is available from Manning
- **November 2020**: Book is available on Amazon, Target, Barnes and Noble, and other re-sellers.

During the writing of this book, a lot happened in my life. I moved home, changed jobs twice, started a side-business that's still going and growing, won two science grants, published 9 academic papers, braced for two hurricanes, and had my first baby. The writing process seemed just as dynamic.

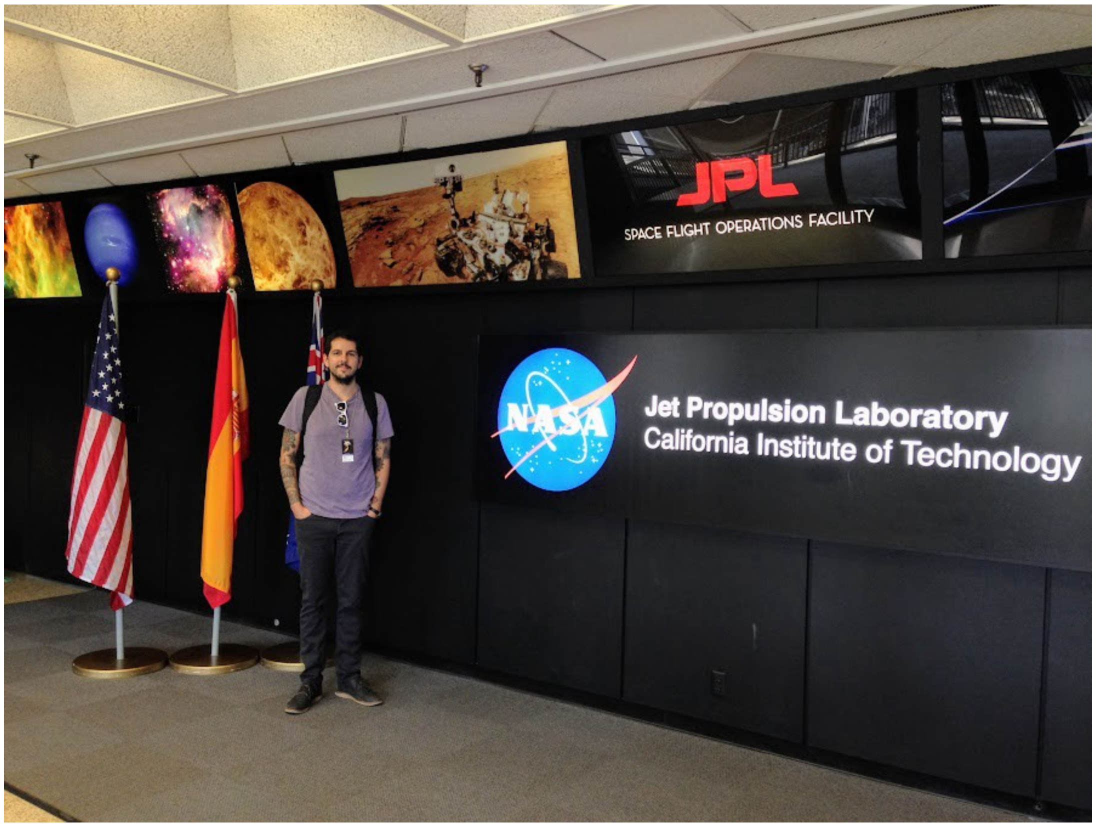
First steps

Brainstorming book ideas

Mike and I got on a call and brainstormed ideas. I wanted to write a Fortran cookbook for intermediate-to-advanced programmers. Mike geared the book toward beginners, teaching the language from scratch and by example. What we ended up with is a Fortran beginners book that teaches the essentials (but not everything) through practical, hands-on examples.

Proposal

The next step was to write a book proposal, which consists of a questionnaire and a sample Table Of Contents (TOC). These are then reviewed by the AE who provides feedback to revise both the questionnaire and the TOC. I made one round of edits to both the TOC and the questionnaire, and sent them to Manning, who sent the proposal to reviewers. In my case, there were six: Four scientists, an HPC sysadmin, and a retired compiler developer. Three of them I knew of by name and their work. To my pleasant surprise, the reviews were strongly encouraging. More exciting to me, in the midst of reviewer response emails, there was an email from Manning with the contract and ISBN number for *Fortran in Action.* The news came while I was visiting the NASA Jet Propulsion Laboratory as an invited speaker. I let my loved ones know and went to celebrate with spicy lamb vindaloo in the highest rated Indian restaurant in Pasadena.


JPL
SPACE FLIGHT OPERATIONS FACILITY
NASA
Jet Propulsion Laboratory
California Institute of Technology

Outside of JPL's mission control center, on the day that Manning decided to go forward with my book. I was shown the inside but taking photos wasn't allowed.

Looking back, the original TOC from the proposal looks nothing like the book we ended up making.

Contract

Manning issued a draft of the contract with what I think are their default terms that they send to prospective authors. They asked me to review and sign. We had a few back and forths negotiating the details, and we were then ready to sign. Here are the key terms that we agreed on:

- I'd get 10% in royalties for the first 5,000 copies, 12% for the next 5,000, and 14% for any sales beyond. (I negotiate these — the original contract offered 10%)
- I'd get $5,000 advance, half at 1/3 of the book delivered, and another half when the

whole book is delivered.

- I'd get 50 author copies (negotiated up from the original 25), and I'd be able to purchase any additional copies at 50% off.
- I wouldn't write another Fortran book with a competing publisher while working on this one. Like writing one book wasn't challenging enough.
- Either party can end the project at any time. In that event I'd return the advance back to Manning and keep the rights to the content written so far.

The contract also stated delivery deadlines and target number of pages, words, figures, and code listings. There were a dozen other legalese items in the contract that aren't interesting enough to mention here. I will touch on this topic again when I discuss the economics of writing.

The writing process

Tools

When we started the work, Manning set me up with:

- A private git repository for the book source files. This is where the raw book files — text, figures, and code — sit.
- A shared Box folder used to share material with editors and other Manning staff.
- A dedicated online book forum for receiving and responding to feedback from readers. Manning expects that you're active on the forum and responsive to readers' queries.

Once the first MEAP (Manning Early Access Program) version is out, you get:

- A webpage for your book. This is where your future readers can learn more about the book, read excerpts, and purchase it while it's still being written.
- An affiliate account with Post Affiliate Pro. This platform allows you to post affiliate links and receive commission on referred sales.

There's also a Slack workspace for Manning authors which I didn't use.

Workflow

Most of the time I worked on a chapter at a time. The workflow for each was:

1. Fill out a chapter plan. This is a two-page document that briefly describes what's the chapter about, why it's needed, what concrete examples will it use, and what's the skill level required of the reader to understand the chapter. This chapter goes to the Development Editor (DE) who reviews it and gives suggestions for changes. At this time we set a tentative deadline for the first draft of the chapter. Usually we

allocated 3 or 4 weeks for this.

2. Over the next few weeks you just write your chapter, at your own pace, without interaction with editors. I made my best effort to make the deadline, but I often needed more time. That was okay. For most chapters I pushed the deadline by one or two weeks. The DE always preferred extending the deadline when needed, rather than getting an incomplete first draft.

3. The DE and the Technical DE (TDE) review the chapter and send back comments and suggestions for improvements. The DE mostly tackles any issues regarding the writing style, organization, whether the examples are appropriate, and similar. The TDE works through both the chapter and code and makes sure that the technical explanations make sense. In my case, the comments came as PDF annotations.

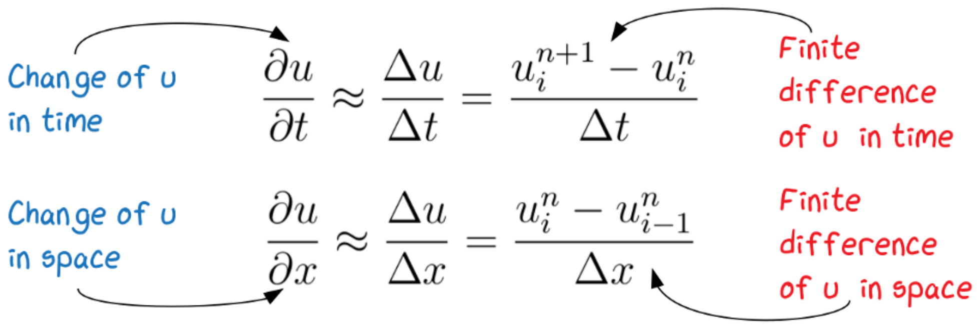
4. Depending on the scope of the requested changes, I took one or two weeks to revise the chapter. On few occasions, the DE asked for a second round of minor revisions. On average, it took me about two months for each chapter, from start to revised and approved.

5. Once finalized, the chapter transitions to MEAP and becomes available to readers in both PDF and LiveBook formats, usually about a few weeks later.

Software I used

In the beginning I asked Mike what software should I use for writing. He said “Use anything you want!”. Then I wrote the proposal in LaTeX and asked if it’s okay to use it for writing the book. Short answer: No, use Asciidoc or Microsoft Word — these are the tools that Manning’s production team works with. So Asciidoc it was. I hadn’t used it until this point, but it was an easy choice for me.

I‘m not a big fan of Asciidoc but I made it work for me. It’s more powerful than Markdown but simpler to use than LaTeX. Writing math requires an external tool, and even when you jump through the hoops to set it up, the result is uglier than any MathJax-enabled Markdown or LaTeX math rendering. This turned out not to be a big deal because Manning strongly discouraged using math in my book. Nevertheless, where I used math I ended up just rendering it in LaTeX and turning it into a figure:



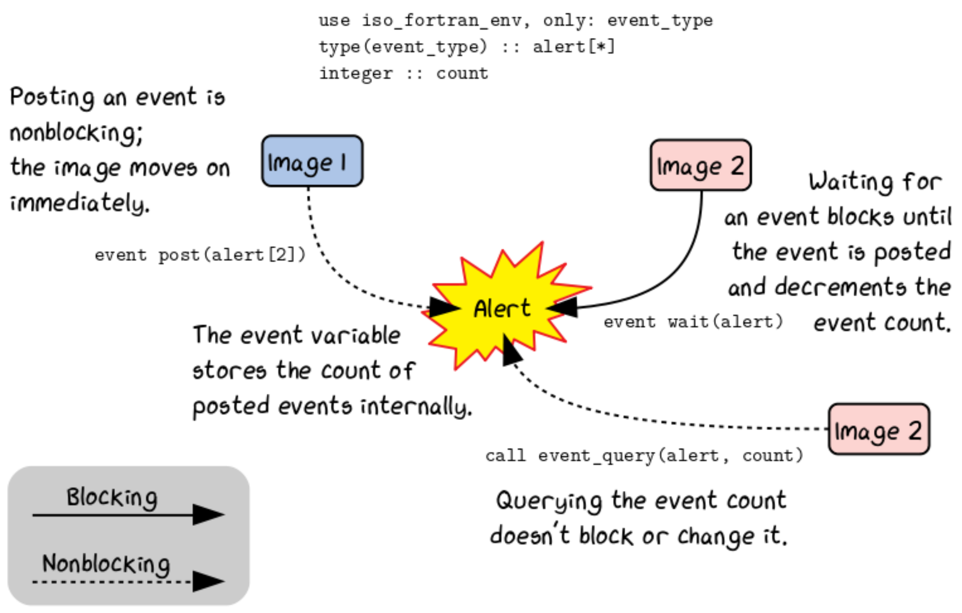
$$\frac{\partial u}{\partial t} \approx \frac{\Delta u}{\Delta t} = \frac{u_i^{n+1} - u_i^n}{\Delta t}$$

$$\frac{\partial u}{\partial x} \approx \frac{\Delta u}{\Delta x} = \frac{u_i^n - u_{i-1}^n}{\Delta x}$$

Change of u in time
Finite difference of u in time
Change of u in space
Finite difference of u in space

In practice, I'd write Asciidoc files using Atom or Vim (interchangeably), then compile them using Asciidoctor into a pdf for preview. In Atom I used the asciidoc-preview plugin, which does exactly what it sounds like.

Most figures — diagrams and illustrations — in the book I made using LibreOffice Draw. This software is simple and a joy to use. Here's one of my favorite figures from the book which illustrates how Fortran 2018 events work:



```
use iso_fortran_env, only: event_type
type(event_type) :: alert[*]
integer :: count
```

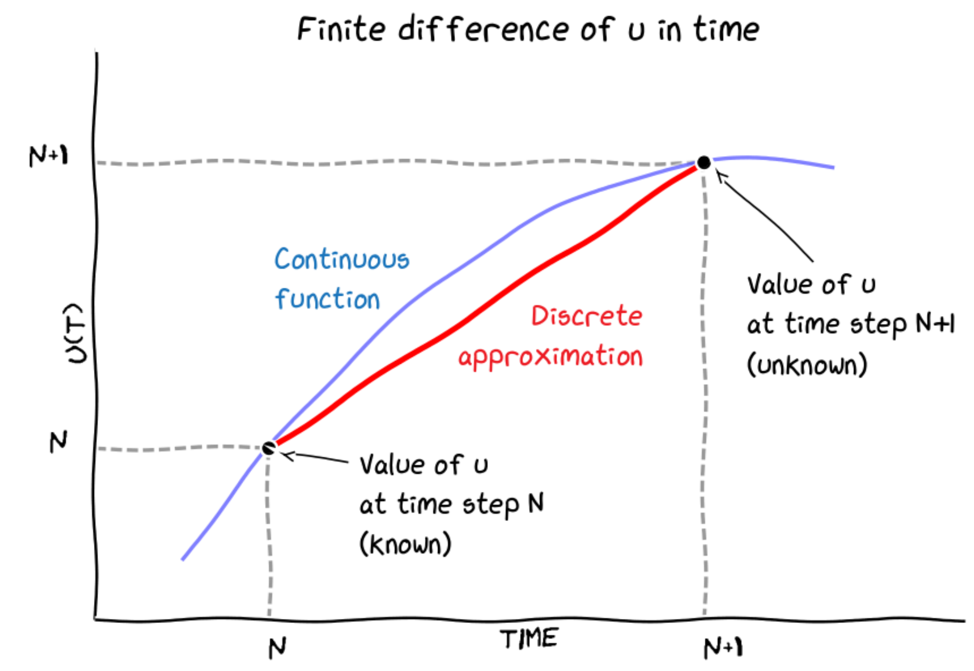
```
event post(alert[2])
```

```
event wait(alert)
```

```
call event_query(alert, count)
```

Posting an event is nonblocking; the image moves on immediately.
Image 1
Image 2
Waiting for an event blocks until the event is posted and decrements the event count.
Alert
The event variable stores the count of posted events internally.
Image 2
Querying the event count doesn't block or change it.
Blocking
Nonblocking

A few others I made with Python and matplotlib in xkcd mode. I used this to illustrate a math concept but still wanted that hand-drawn look. Here's an example:


Finite difference of u in time
N+1
N
u(t)
Continuous function
Discrete approximation
Value of u at time step N+1 (unknown)
Value of u at time step N (known)
N
TIME
N+1

Before this book I had only made figures for academic papers which is a whole different business. Making pretty and clear diagrams and illustrations was something I had to learn. They were universally praised by early readers and reviewers alike, which encouraged me to make them even better.

Getting feedback from readers

Manning provided a dedicated online forum for readers to post questions and suggestions about the book, while it's being written. This was great — to an extent, my readers helped improve the book by providing feedback like "there's a typo here" or "explain this a little more". The forum itself was an old-fashioned, Web 1.0 kind of forum in the spirit of phpBB. This is not a negative — it was clean, simple to use, and *it worked*.

About half-way through my writing, Manning discontinued the forum and moved the posts to their LiveBook part of the site. LiveBook is their modern in-browser book reader which also allows readers to post comments at any place in the book, much like annotating a PDF. However, as an author I found this tool to be frustrating to use to communicate with the readers. Email notifications came the day after, specific conversations were difficult to find, and the UI was overall clunky and sluggish. Although I trust that Manning will only improve it with time, for me this was a low point in the process.

The writing habits

At the time that I started writing, I had transitioned from on-site to fully remote work for my day job. This gave me additional 2 hours every work-day — about 30-minute commute each way, plus about an hour for prepping (bag, laptop, clothes, lunch, etc.) and winding down when I came home. I could then add 1–2 hours of book work each day without taking away from any day-job work or personal time. This was a blessing.

Early on I had committed to work on the book for one hour, and no more, every day. I'd wake up, stretch, have a glass of water, and start writing. Doing it every day and first thing in the morning helped me develop the habit so that writing eventually became its own *raison d'être* — I don't write because I have to but because that's what I do. I also made an effort to not write more than one hour. This has two important effects:

1. When you get in the groove, it's easy to lose track of time and go overtime. This spills over into the other tasks for the day. When this happened, I'd both feel stressed because other tasks could suffer, and I'd finish my workday later than I'd like.

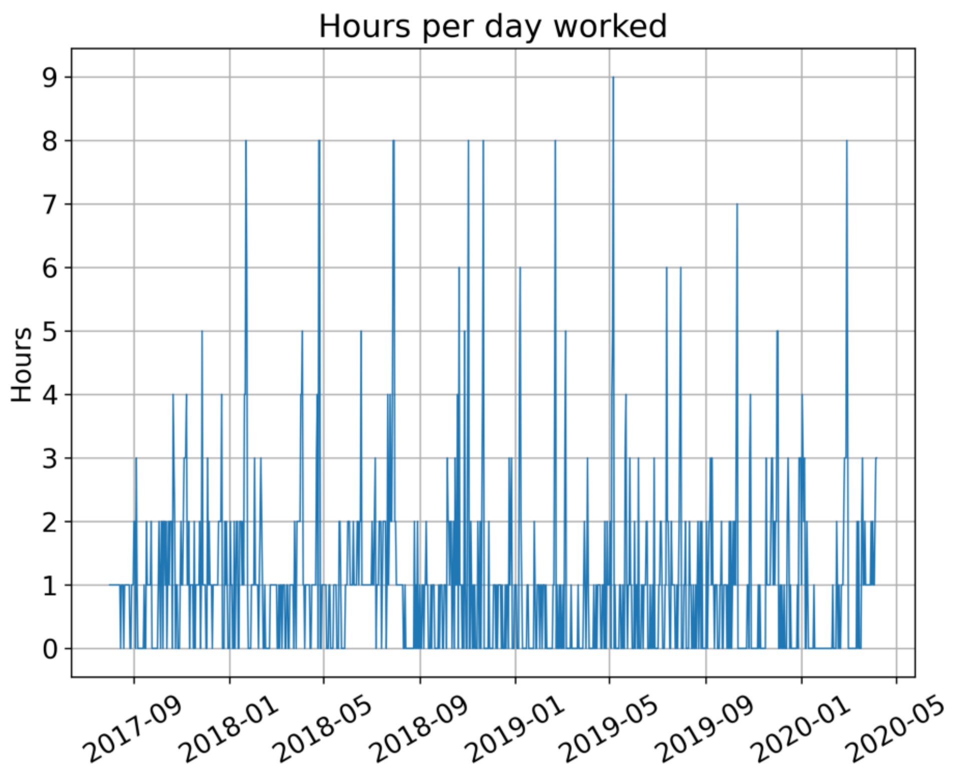
2. When you give yourself a time limit for the task, you don't waste your time getting started. Instead, you get right into it which helps getting into the groove sooner. Setting a hard time limit can help you not procrastinate.

Of course, this was my commitment but not the reality — I often did not follow through. Both writing daily and setting a time limit are easier said than done. There were many days when I didn't write in the morning before the day-job work started, so I'd then write at the end of the day. Everybody's different and I know that many authors work well at night, but this wasn't for me. There were many days when I didn't write at all — during intensive 3-day science meetings, business trips, or on vacation. Once I'd slip with my daily writing habit, it was difficult to get back into it. There were also many days when I worked on it for several hours, usually on weekends, to meet a chapter deadline. The reality for me was different from my original commitment, but that's real life. What matters is that you set the intention and do your best.

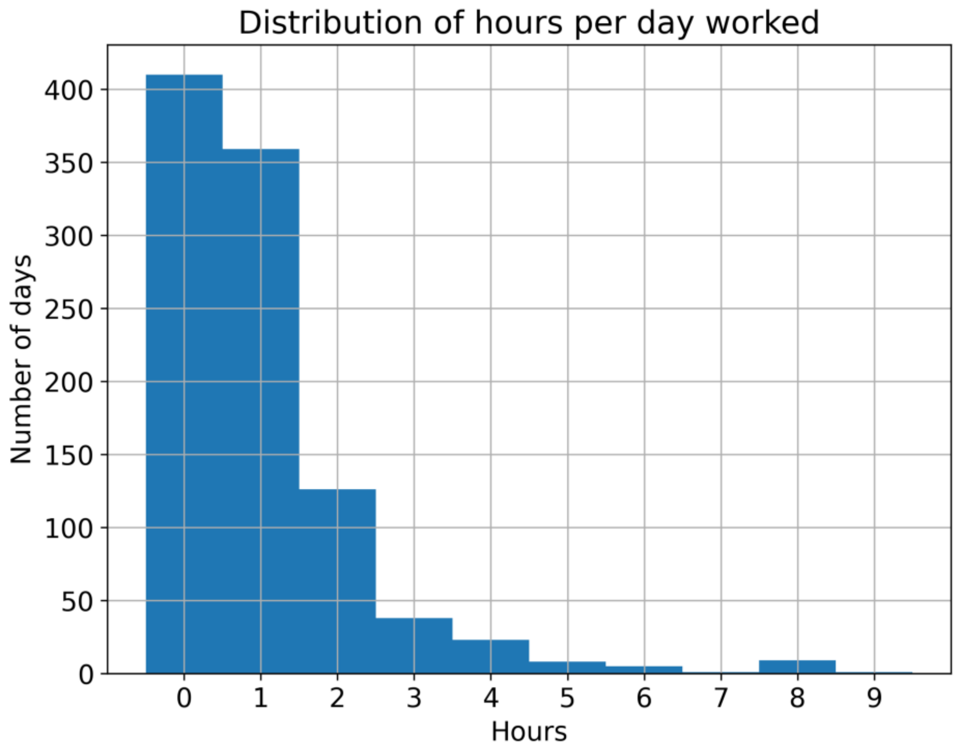
So, what did my writing habits actually look like? Here are some stats:

- I logged a total of 980 days that I worked on the book from start to finish — from 1 August 2017 to 6 April 2020. During that time I put in 975 hours of work. For context, this is about 6 months of full-time work. If you consider the $5000 advance as minimal compensation for the work done, this comes out at $5.12 / hour. For reference, my hourly consulting rate during this period grew from $80 in 2017 to $180 in 2020. Takeaway: Don't do this for the money.

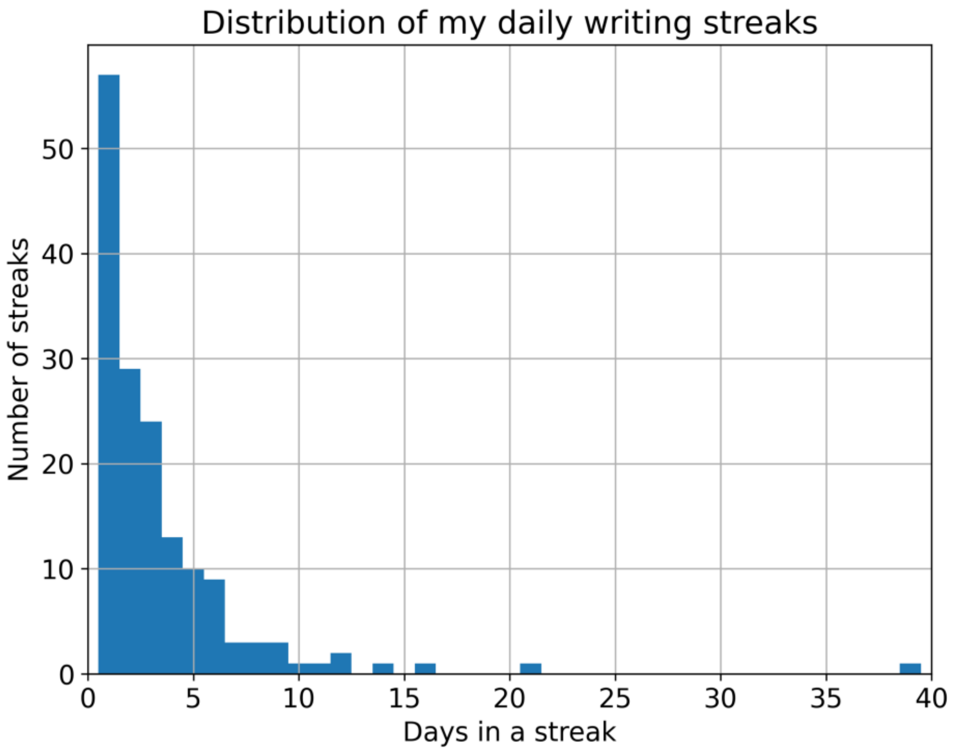
- Despite the initial intention of working exactly one hour per day every day, *and* failing to keep the commitment, I ended up working almost exactly one hour per day on average — 59 minutes and 42 seconds to be exact. The standard deviation is high — 1.31 hours! This means that I was wildly inconsistent with my daily writing habit, but ended up working on average exactly how much I intended to. Here's the full graph:



On most days I either didn't work or I worked the one hour as intended. I had only one day when I worked 9 hours, and 7 days that I worked 8 hours. All of these were to meet a deadline and I didn't enjoy them one bit. Here's the full distribution:

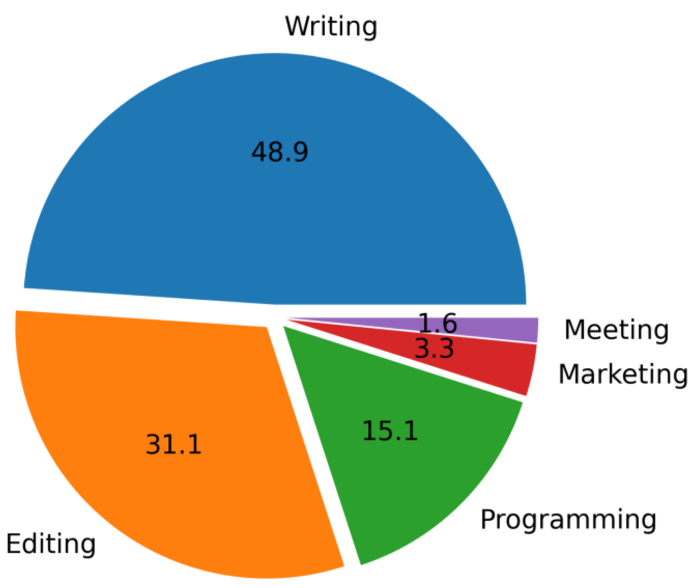


The purpose of my hour-per-day commitment was to maintain consistency. To see how I did, I looked at my writing streaks — periods of time when I wrote daily:



My longest streak was 39 days! I was impressed when I saw this number. However, streaks longer than a few days were extremely rare. The streaks were 3.46±4.18 days-long on average, with a median value of only 2 days. Writing consistently is hard!

Finally, in my writing log I also kept track of what I was working on. Here's the breakdown:



The main takeaway from this figure is that not most of the time goes into writing, as I had previously assumed. I spent a significant chunk of time on editing (31.1%) and programming (15.1%). Of course, the programming part is specific to programming books, although you're likely to spend a similar amount on time on some other kind of research for non-technical books. The time spent on marketing (3.3%) was relatively small, although it didn't feel like it when I was doing it. Marketing consisted of posting on social media, writing blog posts, and emailing people who helped give away e-book copies at events.

Challenges

I faced a few significant challenges while working on this book.

I already mentioned staying consistent with the daily writing habit: Writing is easy; starting to write is hard. On many days I had to simply force myself to sit down and type through the first 5–10 minutes of resistance.

Early on I struggled coming up with concrete examples. Coming from academic science where everything is taught, described, or explained in abstract terms, I was at a loss coming up with specific, concrete scenarios to engage the readers. When I thought of examples, they either seemed too simple to be interesting, or they were too complex to explain well. The editors were patient with me and taught me a lot about this during our coaching calls. In the end, I'm quite happy with the examples that I used and the seem readers enjoy them as well.

I made the work a bit more difficult because I also chose to use a running example — a long project that the reader builds throughout the book (inspired by Miguel Grinberg's Flask Web Development). In my book, that's a parallel tsunami simulator. Getting the running example to develop somewhat evenly as the book progresses, and to keep it both interesting and relevant to chapter topics, was challenging. Not to mention that larger examples are overall more difficult to design and implement well. But it worked out well in the end and the readers seem to enjoy it.

When the editor feedback on Chapter 6 draft came back, it suggested rewriting the chapter from scratch. This was a low point in my writing process and it took me about a month to get back on my feet and start writing again. The problem was that this chapter was meant to teach derived types (classes). Instead, it taught derived types, *and* neural nets, *and* parallel reduction operations. I bit more than I could chew.

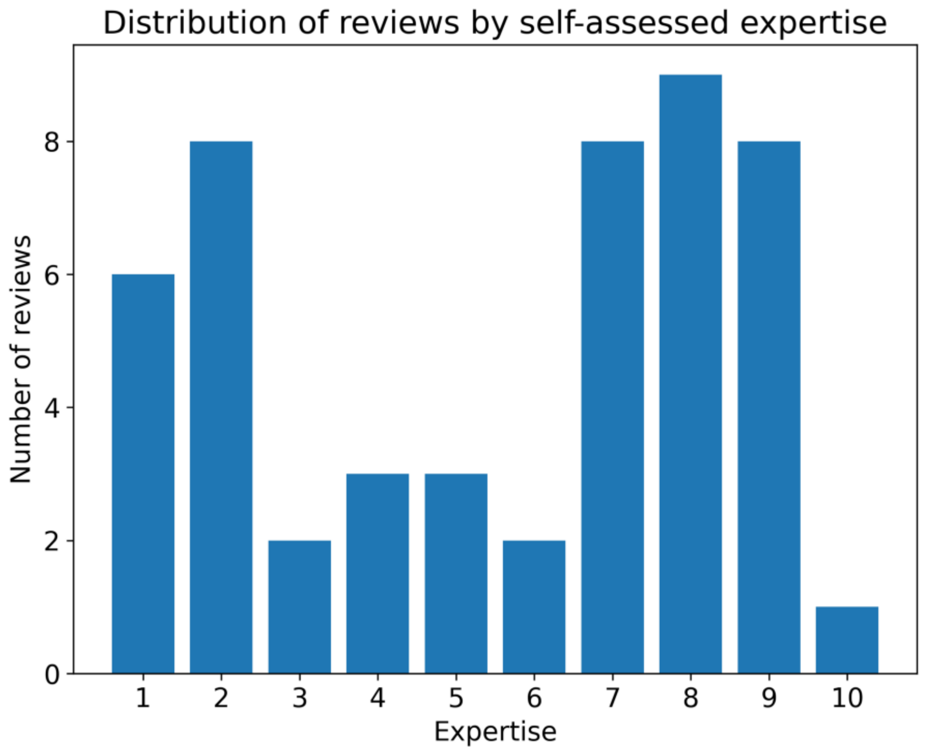
But good things came out of it. The code I wrote for the rejected draft of Chapter 6 I packaged as an open source library, and I even wrote a paper on it. It was since used in by at least two other studies that focused on improving weather and climate models. Not bad for some leftover code from a rejected chapter draft.

Review process

Manning has a comprehensive and diligent review process. Editors review every chapter as it's written and provide feedback and suggestions for edits.

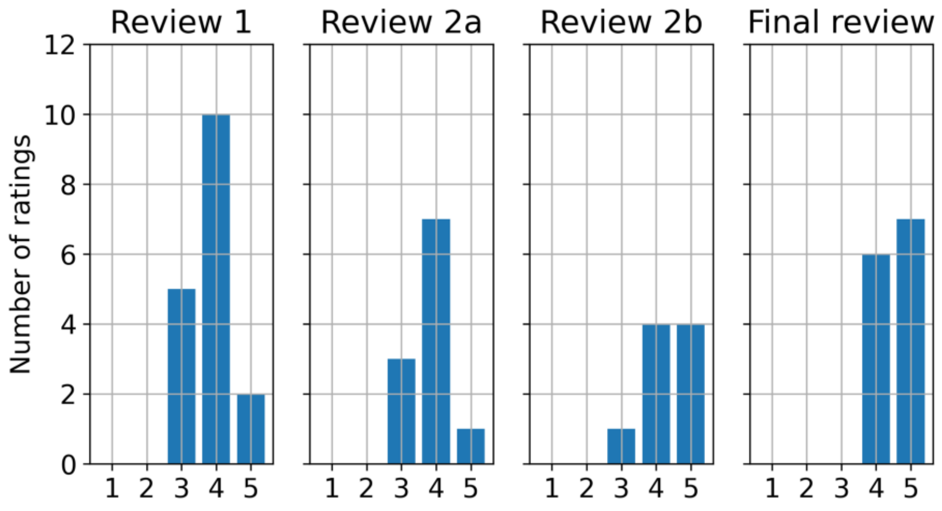
There are also three external reviews, one at each third of the book completed. For these, Manning sends out the book to about a dozen or so technically literate reviewers who are not affiliated with Manning, and who may or may not be familiar with the topic of the book. The reviews are comprehensive — reviewers are asked to assess the quality of the explanations, figures, topics covered, writing and programming style, and more. Many reviewers provided detailed suggestions beyond just answering the questions. They're also asked to give the book, in its current state, an "Amazon" star rating. This was fun — in each review cycle I could see how my average Amazon star rating changed, which was telling me if I was on the right track.

The reviewers also report their self-assessed expertise with the topic (Fortran), on a scale from 1 to 10. Here's what the distribution of that looks like across all reviews:



I don't know if Manning selects reviewers with diverse levels of expertise, or it just happens so. In my case, the distribution is bimodal, with a group of reviewers not too familiar with Fortran, and another who self-reported as advanced Fortran users.

One of the metrics for success was the Amazon ratings from each review. Reading through the reviews I was just hoping for no 1s and 2s, please! 3s I consider okay, and if most ratings were 4s I'd be happy.



First two reviews scored almost equally, 3.82 on average. I was discouraged by the second round of reviews — we didn't seem to have moved the needle. Then Manning re-did the second review, with suspicion that the original pool of reviewers was not representative enough for the topic at hand. They sent the same draft for another review, but this time with considerably more reviewers being familiar with Fortran. The feedback this time was significantly more positive, with the average rating of 4.21. Finally, we were moving in the right direction! By the time the final review feedback came, the average reviewer score was 4.53. Perhaps not suprising, but reviewers who were more familiar with Fortran also thought that the book was higher quality.

As for the feedback, the reviews were somewhat polarized. About half of the reviewers thought that the book was too easy, the other half that it was too hard. Half wanted more math and physics, the other half wanted less physics and more everyday examples. Do you see the pattern? The feedback that was consistent across most reviews was that there were enough many examples, they were interesting, and that the illustrations were high quality and engaging. This was encouraging enough for me to press on.

Working with Manning staff

I've interacted with about a dozen of Manning staff, from editors and marketers to designers and web developers. What stood out to me the most was how every single Manning person was highly professional and owning their work. Sure, occasionally people made minor mistakes or miscommunicated, but any problem that came up was owned and fixed promptly and without resistance.

Overall, I can't recommend enough working with a great publisher — and Manning certainly is one — and having all the support that you need as a first-time author. Mike (AE) told me early on: "We don't expect you to be a great writer when you start. You focus on your topic that you're an expert in. We focus on making great books. We'll make sure that your writing is great in the end."

Economics

You'll spend a lot of time writing your book. Financial rewards are rarely large, and for most technical books they are symbolic.

Royalties

Royalties for technical books tend to be low — about 10%. From my research, I found this figure to be common and expected. With Manning it was no different.

Although I did ask for a higher royalty rate for ebook sales (I got that idea from someone on Hacker News), Manning refused, simply saying "Sorry, we don't do that". I had to try.

We agreed that I would receive 10% of the sales for the first 5000 copies, 12% for the next, and 14% on any sale above 10000. It's quite likely that we won't reach that many sales. John Ressig wrote in 2008 that most technical books don't sell more than 4,000 copies. If that's true for Modern Fortran, I'll only see the 10% royalties. The difference between 10 and 12% is minimal anyway. Moving on.

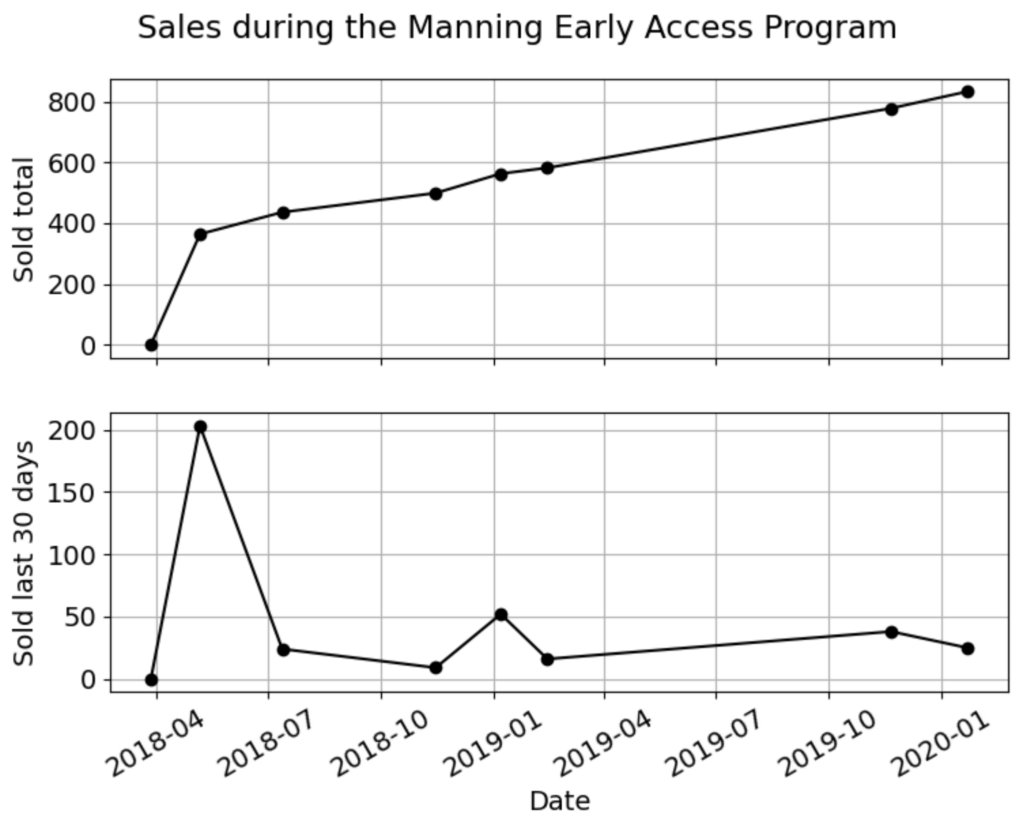
Advance

As I wrote earlier, the advance was $5000, half of it paid when the first third of the book is finished, the other half when the book is done. The advance is the amount in royalties that the author receives ahead of time. If the royalties don't make up for the advance, you keep it. Once they exceed the advance amount, you begin to receive royalty paychecks every quarter.

Manning offers to pay their authors via PayPal or direct deposit. For convenience, I initially chose PayPal. What I didn't realize at the time was that I'd incur a 3% merchant fee for receiving the payment there. For a $2500 payment, this is ~$80. I wasn't happy about this but realized that it was my mistake and moved on. When the time came for the second advance payment, I asked Manning to use direct deposit instead and explained why I wasn't happy getting paid via PayPal. Manning were so gracious and caring that they added the amount to cover the merchant fee from two years before. It's a small gesture, but it convinced again that Manning truly cares for their authors.

Early sales numbers

Manning reported the sales during the MEAP on seemingly irregular intervals. Between May 2018 and January 2020, I received 7 such reports. They details the basic progress metrics — chapters complete, chapters reviewed, number of pages so far, and similar — but also the number of copies sold so far and in the last 30 days. I plotted them here:



The spike in the Spring of 2018 corresponds to the burst of sales when the book was first released in the early access program. Then the sales steadied down at the ~25 copies / month. By January 2020, there were 833 copies sold and that was the last report that I received. Assuming that the linear trend continues, we've surpassed 1000 copies sold sometime in the summer. There should be another burst in sales around the publication time but I haven't received the sales report yet.

These numbers may seem discouraging at first. How my book is doing says nothing about how your book will do. It depends a lot on the topic. Fortran is nowadays a niche topic with a small target audience so we didn't expect that it'd be selling like books on Rust or React would. But Fortran is an extremely mature technology and it's not going away any time soon. So I expect this book to sell slowly and for a long time. However, I do hope that it sells well enough to warrant a second edition, perhaps around the time when the next Fortran standard (202x) is published.

Affiliate sales

When my MEAP was first released (March 2018), Manning also set me up with an affiliate program account. It gives you a unique ID to include in the URL for your or any other Manning book. When a click on my link leads to a book purchase, I get 8% of the sale price.

I didn't at any point actively promote Manning books as an affiliate. I simply placed this URL on my personal book website and on the book's GitHub page, and forgot about it. So far my affiliate account has generated:

- 2878 clicks, 1531 of which are unique
- 176 sales (that's 6.1% conversion rate, or 11.5% for unique clicks)
- $4,686.75 sales revenue for Manning and $374.94 commission revenue for me

It's not much but considering that I barely did any work about it, it's nice to get. Divided by 30 months that I worked on the book, this comes out at about $12 / month from commissions on average. Let's go for ice cream, my treat.

Takeaways

Here are my takeaways and suggestions for anybody embarking on this journey:

- Develop a daily writing habit. It's easier, more enjoyable, and more productive when you write for the sake of writing, than to meet a deadline.
- Find your most productive time of the day to write and weekly schedule, and stick to it. Make it a part of your routine so that you don't have to expend precious willpower on it.

- Writing is easy. Starting to write is hard. Just make yourself sit down and start writing. Even if for just 10 minutes. It doesn’t matter what. Good stuff will come.

- Be clear with your family about what it will take. Have their support and understanding how much time it will take.

Writing Programming Programming Languages Authors Writer

- Working with an established and prolific publisher is rewarding in itself. Just going through the process of writing a book with Manning was a great learning value for me.

About Write Help Legal

- Have a prominent person in the field write the Foreword. This adds credibility to your book. I was lucky to have support and encouragement from Damian Rouson throughout the process.

Get the Medium app

Download on the App Store

GET IT ON Google Play

- [illegible]n’ [illegible]ney. Though there’s a small chance that your book will be a blockbuster, most technical books won’t. The rewards come from the experience, prestige, and opportunities. You’ll have leverage in job interviews, raising your consulting rate, choosing better clients, or an easier time signing the next book deal.

- A few writing tips I picked up along the way: Write like you talk to a friend; don’t overexplain; get to the point; don’t use expensive words; when editing, cut down ruthlessly; The Elements of Style and On Writing Well are great books on this topic.

I hope this helps. Last but not least, here’s the link to my book:

Modern Fortran

Using Fortran, early and accurate forecasts for hurricanes and other major storms have saved thousands of lives. Better…

www.manning.com

If you’re interested in buying the book but want to get a taste for it first, download Chapters 2–4 as a free ebook here:

Exploring Modern Fortran Basics

Fortran has come a long way since IBM first developed it in 1956. Flexible, powerful, and high performing, this general…

www.manning.com

Others on this topic

- *Writing a technical book for Manning* by Barry Pollard (2019)

- *How I wrote my first technical book* by Susan Fowler (2016)

- *Programming book profits* by John Resig (2008)