2018 Week 37: The Week Ending Saturday, September 15th

Paul R. Potts

Sunday

Iron Monkey (1993 Film)

It's been a difficult day. We stayed up later last night than we probably should have, and slept late.

First, we watched a movie in the basement, called *Iron Monkey*. This is a Kung Fu movie from 1993 and it's a lot of fun, with the over-the-top flying moves and fight scenes that I love. Like a lot of Kung Fu movies, it also features characters with a strong moral code, fighting corrupt state officials and monks. there are only a few moments that make me a bit embarrassed to be showing it to my kids. There are a few painfully sexist gags where women get thrown around and used as punching bags. I'm not talking about the fight scenes that actually feature Jean Wang as Miss Orchid. She's a great character. I'm talking about a late fight scene where a woman winds up thrown around as collateral damage and literally goes richocheting off the bad guy. But it could have been worse — for the most part I thought this movie was fine for the kids. There are some terrific scenes of people cooking Chinese dishes, and scenes showing how traditional Chinese medicine was practiced. There are even some bits of music and calligraphy as well. I'm not sure how accurate these scenes were, but they were interesting. It wasn't all ridiculously over-the-top flying fistfights, although there were plenty of those as well.

After that we sent the kids to bed but Grace and I stayed up a while longer. We got up late. I read some more of *Icehenge*. The second novella is really quite good, and continues to be better than I hoped.

The St. Francis of Assisi Parish Picnic

We decided to take the kids to the St. Francis parish picnic, which was actually held indoors because it has been cool and rainy. That was a big potluck in the parish activities center. Grace made a terrific potato salad. It took us forever to get the kids ready to go, though; no one seemed to be able to find and put on decent clothes, a couple of kids couldn't find their shoes, and then there was last-minute confusion with a car seat which had been swapped with a

housemate's car seat, and which was in her boyfriend's car. But we made it and managed to get there in time for everyone to eat and for the kids to make it to some of the activities in the gym.

Afterwards we gassed up Grace's truck and came on home, and started to work on putting away laundry, sorting laundry, and washing more laundry. It quickly became evident that the situation upstairs in the laundry room was a lot worse than we thought, with blankets and a forgotten load of wet items piled up there because Veronica apparently lost track of it. It was starting to get on towards evening. I hadn't really prepared enough for a podcast; I had selected some articles to read, but hadn't printed them out, hadn't marked them up, and hadn't even finished reading them. I realized I just didn't have it in me to record and produce and upload a podcast tonight. So I wrote some notes on Facebook and Twitter apologizing for that. No show tonight. We will at leat start out the week with some clean laundry ready to go, though.

Monday

Last night was disjointed as we were off our regular schedule. We didn't record a podcast. We spent a few hours working on laundry: folding it, putting it away, and trying to figure out what happened to some of our laundry that was left upstairs. Grace then left for a while to run a couple of errands. While she was gone I threw together one of our "such as it is" dinners — I heated up the rest of our tortillas, got out some salsa and leftover meatless taco filling made with walnuts, and a half-empty can of refried beans that was in the refrigerator. I scrambled a half-dozen eggs. I put that on the table and let the kids eat what they wanted. Elanor got a torn-up tortilla and bits of egg. When Grace got back home with more eggs and dishwasher packs, I scrambled another three eggs for her. I got the dishwasher loaded and going, and asked Veronica to do some hand-washing, and it was time to get ready for bed. Four of the kids had taken a nap late in the day, so bedtime was very unevenly distributed; Benjamin and Pippin were very restive.

In the fridge we've still got leftover roast chicken, another whole chicken, a bag of kale salad, and some top round we haven't eaten yet. There's a package of salmon burgers in the freezer, and we have a bag of little dinner rolls. We just stocked up on eggs. There's a bag of brussels sprouts to shred or roast. There are potatoes in the pantry, and another bag of bananas that aren't ripe yet but will be soon. There's rice and oatmeal and chicken stock. So there's plenty of food to get through the next couple of days. It's just mostly not of the ready-to-eat kind of food. I will run to Costco again on Tuesday night after work to pick up a few things.

Vague, Unspecified Housemate Troubles

We are having some difficulties with our housemate, and I'm scratching my head as to what I want to write in this journal, which I've been making public. I'm

also wondering if I should write about it in separate files that I keep private. I guess it depends on what I'm hoping to *do*, eventually, if anything, with this public text, and that private text.

In this public piece of writing I'll just say that one big difficulty stems from conflicts over how we procure, prepare, and consume food.

Grace and I have long hoped to live in community, have tried to establish community, and *have* lived in community, both with our families of origin and with other people, at different points in our lives. We believe that all efforts to live in community must center around, and pivot around, preparing and eating food together. This is why co-ops of all kinds, extended families, religious orders, etc. all find ways to center and organize their lives around food. And that simple word, "food," extends outwards in all directions — into our culture, our finances, our food choices, our desires to garden and grow our own food and to support food production in our community.

Trying to be in community with people who have fundamentally different values around food is *hard*.

Fourth-hand Smoke

There are other issues that have come up. One of them is the simple fact that our housemate's boyfriend smokes, and that's an ongoing problem. He doesn't smoke in our house, but even the residue that spreads to everything he comes in contact with, and the people and things that are transported in his car, is a problem. It's not even second-hand smoke; we're talking *third-hand* smoke, or even *fourth-hand* smoke.

This past week, due to a situation involving transferring babies from their car to our car, and a sleeping baby no one wanted to wake up, Elanor's car seat wound up in his car for a few days, and his baby's car seat wound up in our car. We had to leave for the parish picnic with Elanor in his baby's car seat in our car. Even our short exposure to that car seat — literally, to a *thing* that had been in his car while he he smoked cigarettes — left my throat raw and my voice hoarse, although I wasn't coughing, and seems like it also left Elanor miserably coughing half the night. Elanor can get a nap today, but Grace and I can't; I've got work, and she's got a whole string of appointments.

Yesterday evening we got back Elanor's car seat, and just a couple of days in his car left it reeking of smoke, too. So we had to disassemble it and take the cover off and run it through the washer, twice. The first time, Veronica forgot to add laundry detergent. It still smelled like smoke when it came out of the dryer. The second time, it seemed improved, so we reassembled it. But we're wondering if actually might need to throw it out and buy a new car seat. Cigarette smoke residue is *persistent*. I can smell cigarette smoke on my clothes today, because I spent an hour in a car that also contained a car seat that had spent time in a smoker's car. And smokers *cannot* smell it.

When we took a class on infant CPR at Mott, the respiratory therapist who taught the class described an incident in which an infant heart patient's parent came into the room. There's no smoking in the building, obviously, but there was smoke on the parent's clothes. That cigarette smoke residue was enough of a respiratory challenge that the child went into cardiac arrest. She had to administer CPR and rescue the child. The parent in question apparently didn't believe that this "third-hand" smoke could *possibly* be that harmful to the child, and so the next day it happened *again*.

I don't think Elanor is actually *that* fragile — she's been off all her medication for many months, and her vitals look good. It's been over a year since her open-heart surgery and for the most part she seems to be quite robust, active, and healthy. But it's an interesting correlation that just in the last week, she's started to show health problems; we've been worried about pneumonia and infections. Her chest x-ray showed nothing worrying except a slightly enlarged heart, which I believe is normal for a child who used to have her particular congenital heart defect, before it was repaired; while waiting for infants with her condition to grow large enough to get through the the surgery safely, her doctors *allowed* her to develop a slight heart enlargement, on the grounds that it was safer than doing the surgery immediately after birth. She should "grow into it" as she gets bigger. And her blood tests showed nothing worrying, except a slight anemia. We'll work on that with diet. So my only remaining explanation for her recent coughing is an environmental irritant. She could have seasonal allergies, I suppose, but I really don't think that is the actual culprit.

I don't think there's any way we can convince our housemate to take seriously the idea that her boyfriend's smoking could put our fragile infant with Down Syndrome and a repaired congenital heart defect into the E. R., or into the ground. She's allowed to make her own choices about the environments she puts *her* children into. But Grace and I insist that *we* be able to control the environments *our* children are exposed to.

That's all I'm going to say about this matter for the time being.

Icehenge by Kim Stanley Robinson

Last night and this morning I finished most of *Icehenge*. This book, which didn't seem like it was all that promising, has continued to impress me more and more, and I now regard it as a sort of warm-up for Robinson's amazing Mars Trilogy, and later works. The premise seemed dumb, but it's not a dumb book at all. It touches on many of the same themes. One of the most prominent theme is Robinson's attempt to answer this question: "What it would be like, given our limited human brains, to actually live to be several hundred years old?" He takes this question on quite seriously, and *Icehenge* explores how family relationships would change, how careers would change, how cultures would change, and how our own perception of our own lives would change.

I think this book would serve as a great introduction to several of Robinson's

later works, including the Mars Trilogy, *Aurora*, and *2312*. I don't think there's as much of a clear through-line to his Science in the Capital series, which I have to admit I never finished, or his Three Californias trilogy, which I have to admit I never started. I don't think I will ever finish the Science in the Capital series; it just seemed a little too much like other simplistic disaster potboilers that I've read. And I haven't picked up *New York 2140*. You'd think, as someone very interested in anthropogenic global warming, I'd be interested in climate fiction. What I've read of this particular sub-genre hasn't impressed me, though. This book is pushing me to dig deeper into Robinson's older work.

Over lunch, I finished *Icehenge*. The ending is pretty satisfying, and succeeded in surprising me slightly; I thought that I might have the ending sussed out, but I was wrong. Along the way I think I noticed two subtle references to other works: one, to Robert Silverberg's *Dying Inside*, and another, to the short story by James Tiptree Junior called "And I Awoke and Found Me Here on the Cold Hill's Side." I think there are also some scenes that are stylistic shout-outs to William Gibson's Villa Straylight. There may be other scenes that Robinson also had in mind, and I feel like there are, but I can't quite put my finger on them.

I don't know anything about it yet, but I'm very excited to hear that there's a new novel from Kim Stanley Robinson, called *Red Moon*. It's due to arrive in late October. When it comes out, I probably won't be able to read anything else until I've finished it!

I just got a string of text messages from our realtor. So it's time to close this file, take a deep breath and ask Dorothy Parker's old question, "what fresh hell is this?"

The fresh hell was nothing too unexpected; the showings of our old house this weekend resulted in no interest in a purchase.

Tuesday

Calculating High-Precision Wavelength Values

I made pretty good progress at work and left late. The particular problem I've been trying to solve involves calculating wavelength given frequency. Given a frequency in Hertz, we can calculate the wavelength in meters by dividing the speed of light in meters per second by the frequency.

Things can get a little tricky when trying to do this by computer with very large or very small numbers. The speed of light is already a pretty big number, 299,792,458m/sec. But when working with frequencies of light that are more conveniently measured in *THz* (terahertz, or a trillion Hertz) than Hz, and wavelengths that are more conveniently measured in *nm* (nanometers, or a billionth of a meter) or *fm* (femtometers, or a quadrillionth of a meter), we can wind up having to do math on big numbers.

If we have a number like 191,500,000MHz, to get Hz we have to scale it up by 10^6. The result will be in m, and to get nm we have to scale it up by 10^9. Like so:

```
wavelength = 299,792,458m/sec / 191,500,000,000Hz = 0.00000156549586m = 1565.49586nm
```

Instead of scaling up the divisor from MHz to Hz, and the quotient from meters back down to nm, we can just scale up the speed of light by 10^3 and get the same results — that is, divide 299,792,458,000mm/sec by 191,500,000MHz and get the result in nm:

```
wavelength = 299,792,458,000mm/sec / 191,500,000MHz = 1565.49586nm
```

Because our laser device can be tuned in increments of 1Mhz, we want to display the wavelength in nanometers with 5 fractional digits. We want a precise 9-digit value. But here's where we start to run into a problem: the ARM microcontrollers in the device only support 32-bit integers and 32-bit single-precision floating-point. Single-precision floating-point doesn't really provide that many digits of precision. If we do the math for 191.487767THz, we get 1565.59594. A more precise calculation using 64-bit floating-point tells us that the value should be more like 1565.59587. So that result is off by by 0.00007nm. The calculation is less precise than our laser is.

In addition, some adjacent frequency values will produce identical wavelength values. Values from 191.487768THz to 191.487784THz will all produce 1565.59582. So if the user is changing the frequency in 1MHz steps, the displayed wavelength in nm will appear to get "stuck"

We can't fix this by, say, scaling up by 100 so that more of the digits are to the left of the decimal point. The problem is in the way floating-point numbers are implemented. They are stored as an exponent and a significand, and this gives us a very wide range of values that can be represented, but only 6 to 9 significant digits of precision. That's because the significand has to be a binary number with a fixed width. So in this case, there just aren't enough possible values. We're getting values that are as precise as possible, but they aren't precise enough for our needs. We want five significant digits to the right of the decimal point.

We ought to be able to do this using integers. Our frequency in MHz is already effectively an integer since we can only tune the laser in 1MHz increments. Working with integers, to get a wavelength value with five digits to the right of the decimal point, I want to start with six digits and round by adding 5 and dividing by ten. So we want our result to be a ten-digit integer. To get this we can scale up our speed of light value still more, and get a whole number result in femtometers:

```
wavelength = 299,792,458,000,000,000nm/sec / 191,500,000MHz = 1565495856fm
```

After adding 5 and dividing by ten, we'd have 156549586 in units of fm * 10. The LCD GUI could display this as 1565.549 86 (formatted with the space to

separate the digit for readability).

There's only one problem: we can't work with integers that big.

Well, we can't *easily* work with integers that big.

64-bit Division on a 32-bit Microcontroller

The Atmel SAM4 series has 32-bit integers. The compiler I'm using, for the Keil ARM-MDK, actually lies to me. The online help shows that it supports **long long** and claims these are 64 bit values. It will compile code written using **unsigned long long**. (I didn't think the chip would magically grow 64-bit register, but I thought that maybe the compiler would do 64-bit math using a math library to provide 64-bit operations even though the hardware can really only work with 32-bit integers; it's slower, but it certainly can be done.) But no; it just lies to me. The code using **unsigned long long** compiles just fine, with no errors or warnings. I am able to specify 64-bit constants, and even 64-bit specifiers for formatting numbers with **printf**. But the generated code uses 32-bit values and so I get completely wrong results.

Finding that the compiler would accept **unsigned long long** produced a momentary feeling of victory, followed by a feeling of defeat as I determined that, no, it really was just pretending.

This is an absolutely *egregious* failure to correctly implement the C standard. I'm using the compiler **armcc.exe** version 5.06 update 5 (build 528), for anyone curious, part of the Keil toolchain MDK-ARM Essential Version 5.24.1. I could complain and file bug reports. But I don't really have time to litigate this with my vendor; I'm just trying to solve a problem.

So this was discouraging, but I wasn't sunk. There are algorithms that will let me do 64-bit division. The problem was finding one that works, isn't too slow, and wasn't difficult to port.

Hacker's Delight, Second Edition by Henry S. Warren

I messed around with some unattributed code I found, but couldn't get it produce correct results. So then I turned to one of the thousands of books that are still packed in boxes in my basement. Fortunately because they are all catalogued in Delicious Library, it is easy to find the right box. The book is *Hacker's Delight*, by Henry S. Warren. This is a terrific book; I used to own the first edition and liked it so much I bought the second one. You can't go wrong with either one, but the second edition has a little more cool stuff in it.

Warren describes many, many algorithms in this book. The thing that makes it useful is that these algorithms aren't theoretical, written for some theoretical computer. They are designed and tested to run on real machines, and tested and optimized for modern RISC instruction sets. Some of them are in pseudo-code, but he has a lot of sample code in C as well, and he's tested this code.

Of course, "modern" is always relative, but Warren lays out his reasoning for why the algorithms are written the way they are. He's done the math and worked out the timings, at the instruction level. He explains that if your microprocessor supports *this* kind of instruction, you can make the algorithm run faster this way, or if you are willing to use the space for a lookup table, it can run a little faster like *so*, etc.

In fact, part of what the book does is to make a case for what a future computer architecture's instruction set *ought to* contain, if we want that computer to be able to run certain common operations efficiently. And the book also suggests what kinds of optimizations *compilers* should be able to provide, given certain hardware support. It's really a deep dive into the relationship between algorithms, compilers, optimization, instruction sets, and hardware, wrapped up and presented as a cookbook of tricks to help the programmer do tricky things as efficiently as possible.

Warren actually maintains a web site with the code from the book here. You can use this code even if you didn't buy his book. But I recommend buying the book; it's a great book.

The specific **divlu2** function I'm using is available here. That function takes four parameters:

- Two 32-bit unsigned integers, representing the high and low parts of a 64-bit unsigned integer dividend
- A 32-bit unsigned integer divisor
- A pointer to a 32-bit unsigned integer remainder

It returns a 32-bit unsigned integer.

Warren has other algorithms that will work with bigger numbers, but I chose this one because it actually does what I need. My divisor and quotient will both fit into 32 bits. I made some minor tweaks to his code. I removed the remainder logic, since I don't need it. I modified his C implementation to follow my "house style."

Go To Instruction Considered (Mostly) Unnecessary

My general policy, having been writing C code for a living on and off for almost 30 years, is to avoid **goto**. Some programmers use **goto** routinely in error-handling. I don't. I use the structured programming concepts I learned way back in school. I just believe these constructs make code more readable in most cases.

But I'm not entirely dogmatic about it; I've used **goto** in C code on occasion, to break out of nested loops. The need for **goto** always suggests to me that I should consider refactoring my code to avoid it, but sometimes there just isn't a simpler way to do things, so I use it. I think I've probably actually come across fewer than a dozen cases where I felt that **goto** was justified, in *decades* of writing code in C and related languages.

I guess my rule could be stated "don't use **goto** in C code unless you have a very strong justification for doing so."

This is not all that different than Edsger W. Dijkstra's advice in his famous letter to the editors of CACM, "Go To Statement Considered Harmful," in which he writes:

> The go to statement as it stands is just too primitive; it is too much an invitation to make a mess of one's program. One can regard and appreciate the clauses considered as bridling its use. I do not claim that the clauses mentioned are exhaustive in the sense that they will satisfy all needs, but whatever clauses are suggested (e.g. abortion clauses) they should satisfy the requirement that a programmer independent coordinate system can be maintained to describe the process in a helpful and manageable way.

David Tribble has annotated Dijkstra's famous letter, which might help us understand it:

> Here, finally, we get to the crux of Dijkstra's argument concerning the lowly goto statement. Essentially, Dijkstra argues that the "unbridled use" of goto statements in a program obscures the execution state and history of the program, so that at any given moment the values of the call stack and loop iteration stack are no longer sufficient to determine the value of the program variables.
>
> This obfuscation is a consequence of the fact that an unconstrained goto statement can transfer control out of a loop before it is completed, and likewise can transfer control into the middle of a loop that is already being iterated. Both cases complicate the way in which the counters in the loop iteration stack are modified.

Tribble makes the point that Dijkstra is talking about the use of *unstructured* **goto**:

> What Dijkstra means by the goto statement *as it stands* is otherwise known as an *unstructured* goto. That is, a **goto** statement with no restrictions about how it may be used in an otherwise structured language.

In my own code I'll occasionally justify **goto** in cases where I want to break out of deeply nested code, jumping *forward*, and I can't find a clearer way to express the code.

Of course, at the machine language level, most flow control involves jumps that are the equivalent of **goto**, either conditional or unconditional, and of course high-level code *turns into* code with this kind of **goto** in it. I say "most flow control" because some architectures do provide instructions that implement loops without explicit **goto**.

So this suggests another possible justification: a programmer might want to use **goto** because he or she knows that using **goto** will result in a compiler generating a certain desired sequence of instructions, for performance reasons. I'm not a big fan of this approach, using C as more readable assembly language, because it tends to be very fragile. A minor compiler update or change to an optimizer can and likely will break the programmer's assumptions. In these cases, I think it might be better to write a reference implementation in C without **goto**, and a platform- and compiler-specific implementation in assembly language, for maximum control of the implementation.

Warren uses **goto** in a couple of places like so:

```
again1:
    if (q1 >= b || q1*vn0 > b*rhat + un1) {
        q1 = q1 - 1;
        rhat = rhat + vn1;
        if (rhat < b) goto again1;}
```

He's not using it to escape from nested loops; he's using it to create a **while** loop, albeit a **while** loop with an extra exit condition at the bottom. So essentially he's combining a **while** loop and a **do** loop. In C; there's no **continue_if** keyword; I can't 'write:

```
loop_if ( condition 1 )
{
    /* Loop body */

} continue_if ( condition 2 );
```

But what if I could?

Alternative Looping Constructs for a C-like Language

C was designed when computers were much smaller and slower and offers a fairly minimal set of keywords and looping constructs. What if we were willing to commplicate it slightly for modern computers and modern programmers? For example, we could have **break_if_not**:

```
loop_if_not ( condition 1 )
{
    /* Loop body */

} break_if_not ( condition 2 );
```

What if we allowed the statements **break**, **continue**, **break_if**, **break_if_not**, **continue_if**, and **continue_if_not** to be used in the body of the loop, as well as the end? Would those constructs help people better express their programs?

Then, we could turn plain infinite loops, defined using a **loop** keyword at the top, and break or continue in different ways at different points inside the loop

body:

```
loop
{
    /* Loop body */
    /* optional break or continue statements or their negative versions can go anywhere in t

};
```

We might consider the keywords; **loop**, **loop_if**, **loop_if_not**, **continue**, **continue_if**, **continue_if_not**, **break**, **break_if**, and **break_if_not**. But we could express a huge variety of loops, and we'd avoid keywords from Common Lisp, Dylan, and Ruby that I find hard to read, like **unless**.

We've already come up with a replacement for the standard **while** loop in C, using **loop_if** or **loop_if_not**. The equivalent of a C **do** … **while** loop could look like so:

```
loop
{
    /* Loop body */

} continue_if( condition_1 );
```

If we banned single-statement loops without blocks (and those are considered bad practice anyway, along with single-statement **if** and **for** without blocks), we could retire **while** and **do** (although **do** might have to stick around to continue to support a macro expansion trick).

And what if we had a variant of **break** which used a label? That would pretty much cover all the loops I ever use, and the only conditions under which I would typically use **goto**. Although I'm still chewing over how to stick the label to a set of braces; I might like to put the label after the top brace, and I might like to use a Perl-like sigil as a cue to the compiler, and a label namespace. But I haven't thought about the problem very hard yet.

Anyway, this is why they don't let me design computer languages… but I am still harboring a secret plan to design and implement a language.

Let's look at Warren's code again:

```
again1:
    if (q1 >= b || q1*vn0 > b*rhat + un1) {
        q1 = q1 - 1;
        rhat = rhat + vn1;
        if (rhat < b) goto again1;}
```

There are some funny things in this code. Both conditions depend on **rhat** and **b**, and two "branches" of the first condition depend on **b**, so that's three comparisons that depend on **b**, and two that depend on **rhat**. This suggests to me that there might be a way to combine these conditions. But for now I'm going

to assume that if there really was a good way to optimize those comparisons, Warren would have done it, and so I'm not going to go down that rabbit hole, at least not today. (Did you ever wish you could split yourself into multiple clones, like Michael Keaton does in *Multiplicity*?)

Normally I'd write a loop like this, which executes zero or more times, as a **while** loop, and I'd use **break** to end the loop early. But this logic is backwards: the code goes back to re-evaluate the loop condition if **rhat** $<$ **b**. We could do that like this:

```
while (q1 >= b || q1*vn0 > b*rhat + un1) {
    q1 = q1 - 1;
    rhat = rhat + vn1;
    if (rhat < b) continue; else break;}
```

But if we're willing to reverse the comparison to **rhat** $>=$ **b**, and I don't see why we shouldn't be willing to do that, we can just write it like this:

```
while (q1 >= b || q1*vn0 > b*rhat + un1) {
    q1 = q1 - 1;
    rhat = rhat + vn1;
    if (rhat >= b) break };
```

I took a peek at the generated assembly, and it didn't seem to be inflated by this change, and I tested the function, and got the same results, so I'll stick with this change to get rid of the **goto** statement. I think getting rid of it makes the underlying structure of this code fragment clearer; it's a while loop with an extra exit condition. And I don't see a good reason **not** to remove the **goto** statement.

I could spend the rest of the day profiling the slightly different versions of the function to determine if there is any significant performance difference at all, but for my present application that amount of work isn't justifiable.

Ideally, I'd have a version of this function in optimized assembly language to use. Then I would just treat it as a black box in my code. But I haven't been able to find one. I suppose I could work on writing one (but see again my earlier comment on splitting myself into multiple clones to free up time…)

Wednesday

Costco closes at 8:30 and I made it in the door at exactly 8:10 last night. I bought fruit, celery, pork medallions, lunch meat, rolls, a box of ramen for the kids, and a chicken pot pie. Unfortunately I didn't have enough time to return all the returnable cans and bottles, so they are still rattling around in the back of my car. And I made the mistake of going to Costco while I was quite hungry, so I brought home a few extras that weren't strictly necessary including a box of cookies, "Petite Palmiers." I also bought some dark chocolate caramel candies, a bag of toasted chick peas, and a bag of toasted hazelnuts. I really do start to

crave carbs, *hard*, as the days get shorter. I have to watch myself a little more closely; we didn't actually need those things, although they will certainly be eaten.

Grace made chicken soup in the Instant Pot using the second chicken we bought last Friday. It was quite good. We've got a fair amount of leftovers, so Friday's shopping trip shouldn't need to be all that big.

Things have been difficult with our housemate; I'll leave it at that for now.

Daughter of Dreams by Michael Moorcock, Continued

While Grace was getting ready for bed, I read a bit more in *Daughter of Dreams*. I've finished the second of three parts of the novel. Things have gotten complicated as Elric, whose own body is an enchanted sleep, and who is now piloting Von Bek's body, travels the Moonbeam Roads and meets his daughter, Oona. We're getting into highly abstracted locations now, enchanted places, and I have to confess I am not really enjoying this part of the story quite as much as I enjoyed the earlier parts. If the third part is good, I will still consider it a good book. If the third part doesn't improve and the story doesn't end well, then I'll consider it an interesting book that had a lot of promise but which didn't quite work out.

I heard today from our realtor that she is showing the house again.

Grace and I continue struggling to try to figure out what to do with the house. I'm considering whether it might be possible to borrow a few thousand dollars to put in a furnace. That might make it simpler to lease the house as-is.

Thursday

The Saginaw house has been shown a few more times and there are more showings scheduled, but so far we haven't gotten any new offers.

Last night we had a pot pie for dinner. The kids were begging for a movie afterwards but Grace and I were really not up for it and wanted to get to bed early. I read some of the kids a few more chapters of *The Wild Robot Escapes* and we sent them off to bed. Then I read a little bit more of *Daughter of Dreams*. Not very much, though — I was too tired to concentrate. So Grace and I did get on to sleep at a more reasonable time, about 11:30. We were woken up by Sam and Joshua making ramen for breakfast, and then Pippin screaming his head off, as someone apparently said something to him that set him off. I had breakfast at Harvest Moon Café. I should get paid tonight. Grace found the paperwork for renewing my driver license, which was behind one of the benches in the family room for some reason. I'd been wondering where it was. I need to get that paperwork done and also change my voter registration so I can vote locally.

A Special Character for the Amulet GUI

I spent a good chunk of time working on what seemed like a simple task: getting our Amulet GUI to display a special character, the "plus/minus" mark. (There's an HTML entity for it, `&plusmn;`, and if it works correctly in your browser, or you are reading this in some other derived format such as a printed PDF file, you might be able to see the character between the quotation marks here: "±".

The standard Arial Bold 12 font that comes with GEMStudio doesn't have this character; it contains a limited subset of the character from the Windows TrueType font. GEMScript has a number of strange limitations. It doesn't handle escaped character codes in string constants, although it will do it in character constants.

What I finally wound up doing was copying and pasting the character directly into the string constant in the source code. That's not something that I would ever consider doing in a more conventional programming language, like C, or Python, or Java, or JavaScript. In fact sometimes it works to put special characters in strings, but the C standard specifies that implementations are only *required* to support uppercase and lowercase letters, numerals, and a handful of special characters including punctuation, brackets, etc. (basically, everything printed on the keys of an American keyboard), and a few whitespace characters like tab and form feed.

GEMScript supposedly supports Unicode strings, but given that, it's pretty crazy that you can't specify Unicode characters in strings.

There's a tool which will generate an Amulet font file from your installed Windows fonts. You can specify a range of characters to include. So I extended it a bit, to include the plus/minus character (it has the code value 0xB1). But it didn't work right; my characters were all way too big, now.

I finally learned that the fonts that the GEM Font Converter creates are affected by my Windows accessibility settings. I had my on-screen text set to use "medium" size fonts, instead of the default "small," because I'm fifty years old.

The idea that this setting would affect the fonts that the GEM Font Converter can retrieve from Windows is insane, but not actually surprising. Windows is a collection of layers and layers of historic hacks. Of *course* they just made it so that the accessibility settings simply scaled up the fonts that the Windows API provides to clients, rather than adding a new API or modifying an existing API to allow the client to indicate whether the client wants the raw font or the font scaled per the user accessibility settings. Microsoft pretty much always takes the route of maximum backwards compatibility, even if the results are surprising or confusing.

Anyway, that's how you can waste half a work day trying to get your Amulet GUI to display a single special character. I don't recommend it. (But the page finally looks just the way I want it.)

My Grayscale Phone

As an experiment, I set my phone to show all images in grayscale. This is supposed to make your phone and the applications on it a little bit less addictive and sleep-disruptive. It seems so far like that is true, but it also seems like this setting makes the phone eat its battery charge a lot faster than usual. So I'm not sure I can leave it that way.

Friday

The 36th Chamber of Shaolin (1978 Film)

Last night we roasted the pork medallions from Costco and at them with kale salad. Then after a long delay trying to get everyone to brush their teeth, I took the kids down into the basement to watch another Kung Fu movie. This one was *The 36th Chamber of Shaolin*, from 1978.

Watching it, we had the same problem we had watching the last one: at a random point in the movie, the playback froze up, and I had to force iTunes to quit. After launching it again, I could play that same part of the movie with no issues. Technical problems like this are one of the reasons I don't really like buying movies through the iTunes store, although I've been doing it since I still have a lot of iTunes store credit.

This movie is rated PG, compared to *Iron Monkey* which was rated PG-13, but this one actually feels more violent and includes scenes of blood. The blood is a pretty blatantly fake color (almost magenta), but it was a little surprising. We also have moderately convincing makeup showing gangrene and bruising. So I guess this movie makes the attempt to be a little more realistic, in that it connects fighting with violence and injury and even death.

The version available from iTunes is strange, in that it is both subtitled and dubbed. The dubbing doesn't match the subtitles, so if you are reading along and listening it's really laughably confusing at times. The movie also feels a little long, at almost 2 hours. But many of the training sequences are really fun. I was talking to the kids about their favorite rooms. My personal favorite was the one in which our hero has to walk through a room full of swinging sandbags, knocking them out of the way with his head. All around him, fellow students are slamming their foreheads into the sandbags and then falling down, dazed. I just hope this doesn't encourage my kids to give themselves concussions.

The "Real" Shaolin Temple

The actual history of the Shaolin temple, and what was taught there, is a lot more mundane, although still interesting:

> In the 5th century, an Indian Buddhist master named Buddhabhadra traveled to China to spread Buddhism, and by the year 477, he had become influential enough that Emperor Xiaowen of Northern Wei

> built the original Shaolin Temple for him to begin teaching Chinese monks. These are among the few facts of the early Shaolin that scholars generally agree upon.

But:

> Over the ensuing centuries, the Shaolin Temple performed basically the same function as it still did into the 20th century. It was essentially a boarding school for boys. The younger the recruit, the better; students as young as 5 or 6 years old were preferred. They developed tremendous flexibility and agility, and studied Buddhism.

And:

> In 1972, when American TV viewers first saw Kwai Chang Caine wrap his arms around the hot cauldron and brand himself with the marks, what they didn't see was the rest of this elaborate test called the Wooden Men Labyrinth. But nobody else ever saw it either, because like so much of what we think we know of the Shaolin, it was the purely fictional invention of modern authors.

Ouch.

Kung Fu Films

But still, I grew up occasionally watching *Kung Fu*, the television show, and my step-brother Tony liked to show me Bruce Lee movies. I'm a fan of Jackie Chan and an even bigger fan of the fantastically beautiful and artistic wuxia genre. While it's important to be skeptical of the Orientalism and stereotyping that goes into these films, I love the tales of legendary heroes fighting corruption and injustice with humility, tenacity, bravery, and discipline. I'm a big fan of *Crouching Tiger, Hidden Dragon*, and *Hero*, and *House of Flying Daggers*. I'm curious about *The Forbidden Kingdom*, although the reviews aren't terribly good, and *Curse of the Golden Flower*, although the reviews for that one also aren't very good, and it is rated R, so I don't want to show it to the kids. I'm also curious about *The Assassin* (the 2015 film), and *Dragon* (the 2011 film), although *The Dragon* is also rated R. Are there any more I should be looking for? How are the twenty-six(!) old Zatoichi films? How is the 2003 film? How is *Tai Chi Zero?*

Some of these are available via the iTunes store, and I still have a credit, so maybe we'll try some more of them.

I'd like to pick up the 2013 Criterion Zatoichi set, with 27 discs(!), but that's not really in the budget.

If you are wondering what I'd like for my fifty-first birthday, besides "selling the old house," that's it: the Criterion Collection Blu-ray set called *Zatoichi: The Blind Swordsman*. My kids and I would watch the hell out of those. In fact I thing those movies would be great to show on Potts House movie nights, when

we get our basement set up with a projector or a big screen and enough chairs to have folks over to have dinner and watch movies, also known as "chili and Netflix." But unfortunately the very basic basement home theater we hope to set up is another one of those goals, like beds for our children, that keeps receding into the future, as we continue to fail to get our finances under control.

There are three showings of our old house today. Grace and I are praying that one of these showings will turn into an offer. We just had to pay choir tuition for two of our sons and it is a big expense. We still haven't received our last reimbursement from Liberty Mutual. I got paid overnight, and today I am making a credit card payment; I have been setting aside money from the last three paychecks to put towards paying down one of our two credit cards. But it's discouraging when we just had to charge something that is bigger than our payment. It means our debt situation continues to move in the wrong direction.

For lunch I had a grilled cheddar and marmite sandwich with a side of pickles and olives, and a cherry Italian soda. It was tasty but like anything involving marmite, damned salty. So I'm guzzling extra water and hoping I haven't damaged myself. While eating I read two more chapters of *Daughter of Dreams*. The story does pick up a bit in the third section; we get more scenes with fighting, and Elric doing things, like summoning Meerclar, Mistress of Cats. So I'm feeling optimistic about the rest of the book.

Saturday

Last night after work I went to Costco as usual. It was a pretty standard load of groceries, except that I've been adding red meat at Grace's request. So I got some beef and some lamb chops. This adds quite a bit to the final bill, unfortunately. Grace and the kids did not make a pot of rice so when I brought salmon home we had salmon, salad, and one of the giant Costco pumpkin pies.

After dinner and a half-assed cleanup job we took the kids down into the basement. They wanted to watch Ninjago, but I'm kind of sick of Ninjago, so I put on the second disc of our DVD set for the original *Star Trek* animated series. One or two of these episodes we had seen before. So I wound up going into my little office room in the basement and continued the ongoing project of cataloguing books. I started a new box with all the Elric books I've finished and a couple of recent Library of America arrivals.

Organizing Our Books

I'm reusing a 12" by 12" by 12" book box. This is unfortunately the last empty one. If I'm going to have to continue to store most of my books while still slowly acquiring new books for much longer, I'm going to need to order more boxes. The plan for the last couple of years has been to get everything shelved, or at least large sections of the collection shelved, and do some purging as we un-box and shelve books. I suppose we could also try to purge while everything is still in boxes, but that sounds incredibly tedious: look through the online catalog,

identify some books we're willing to get rid of, take apart the stacked boxes to pull out the ones containing the books to purge, open up the boxes and take them out, leaving loose space in the boxes which I would presumably fill with newsprint or something like that, and then re-stack everything.

This Old Mac Pro

My Mac Pro seems to be running slower and slower and it's been hot in the basement. The Delicious Library program just crawls, when you do things like add a shelf. I don't know why it is so slow, but it makes me wish I hadn't put my whole book inventory into this program. Maybe I should have stuck with a simple spreadsheet, or just flat text files. The integeration with Amazon is nice, though; I like the ability to look a book up that I have in my hand, in Amazon's database, and then find it and add it to my library with one click. The bar code scanning functionality is nice, too, although it tends to fail sometimes, which I think is more an artifact of the uneven and inconsistent bar coding on products themselves, rather than the scanning code.

I have a suspicion that the boot drive is going, or is at least very bogged down. Maybe the CPUs are throttling? But I don't remember that ever happening before, even when I had the machine running in my hot attic office in Saginaw. I have blown out dust pretty recently so it shouldn't be a matter of clogged airflow. And the fans aren't blasting.

I am really not sure why it is so hot in the basement. Last summer even when it was hot outside, the basement stayed cooler than the first floor. I think it may have something to do with the old dehumidifier we are running down there; it puts out a lot of heat, and the basement level may be so tightly insulated that the heat can't escape. I need to make sure everything gets a fresh backup. I haven't had the money to replace drives, but it's overdue.

Dragon Ball Z: Broly – The Legendary Super Saiyan (1993 Film, 2018 Fathom Events Theatrical Release)

While I was waiting for Delicious Library to catch up, I looked up movies this weekend. I saw a special movie scheduled, with just two showings: *Dragon Ball Z: Broly – The Legendary Super Saiyan*. Knowing almost nothing about the whole Dragon Ball Z franchise, I thought maybe I would take the kids. I'm trying to get them interested in animation and film that is not all part of their current obsession, Lego Ninjago. This movie is apparently in limited release as part of the marketing campaign for a a forthcoming movie, *Dragon Ball Super: Broly*, which I also know next to nothing about.

After the kids finished four old animated *Star Trek* episodes, they were bored with it and we finished up and came back upstairs. We managed to get to bed at a fairly reasonable time. This morning we didn't sleep all that late. Grace drove down to Milan to pick up some bread at The Mother Loaf Breads. It turns out if you get there before noon, they have a lot more bread! (That's sarcasm…

we're just always running late to do everything.) We got several loaves today, including a breakfast brioche which was fantastic. While she was getting bread, I hauled out the griddle and made pancakes. It is still really hard to get good results on our cast-iron griddle. Even letting it heat for five minutes, it's too cold for the first batch, and then too hot for the second batch. So the pancakes were unevenly cooked, although not too uneven to eat.

I took the kids to the movie. I had not really intended to take Benjamin, but he was unhappy at the last minute and I wanted to avoid a meltdown, so he came with us. When we got into the theater, the trailer for the new Broly movie was playing, but there was no sound. Some of us in the audience spent a minute or two entertaining ourselves doing our own voiceover, karaoke, and foley. But the sound remained off, so I walked back down towards the lobby and told the ticket-taker about it. He called someone on his radio. After a few more minutes we had sound, but we had missed a few minutes, so we didn't really know what was going on. And that situation sort of continued through the movie. Most of the movie consists of big, dramatic fight scenes, which I expected, but I was expecting a little more plot and story. Maybe Dragon Ball Z is the wrong franchise for people who want plot and story.

Anyway, Wikipedia has a plot summary that explains the part that didn't have sound:

> On his planet in the Otherworld, King Kai senses the destruction of the South Galaxy by an unknown Super Saiyan, telepathically contacting Goku upon realizing that the North Galaxy will be targeted next. At that moment, Goku and Chi-Chi are sitting down having an interview at a private school which they hope Gohan will attend, Goku abrupted uses Instant Transmission to reach King Kai's planet and get the entire story.
>
> Back on Earth, the Z-Fighters are having a picnic in an unknown peaceful area when a spaceship lands and an army of emerging humanoids greet Vegeta as their king. Their leader is a Saiyan, Paragus, who claims that he has created a New Planet Vegeta and wishes for Vegeta to accompany him in order to rule as the new king. Vegeta initially refuses, but agrees after Paragus tells him that a being known as the "Legendary Super Saiyan" is running rampant throughout the galaxy and must be destroyed before he comes to Earth. Skeptical of Paragus' story, Gohan, Trunks, Krillin, Master Roshi and Oolong go along with Vegeta.

So. The drawing style is all over the map, constantly shifting, which is fun. There are a lot of explosions, and the way the characters fight and smash each other and the scenery seems quite reminiscent of *Akira*. None of us really enjoyed the film all that much. I'm going to go out on a limb and say that hearing the first few minutes wouldn't really have helped. Joshua described it as "all abs and eyebrows." The movie was mercifully short, and I was grateful for that.

Sam and Joshua think it could have used more dialogue and more story. I guess this wasn't a good introduction to the Dragon Ball Z franchise. Maybe I'll poke around in the iTunes store and see if there's something else.

Sam's Bike

After the movie Grace and I put my old aluminum-frame Marin mountain bike, which I used to use for commuting, in the back of my Element, and took it to a local bike shop. The bike has been in storage for about ten years. Sam is behind on learning to ride, so I asked them to remove the pedals, so he can use it as a balance bike, and then maybe after a while we can put the pedals back on. I am embarrassed to report that I actually didn't know that Sam could not yet ride a bike. We have a bunch of bikes and I see the kids riding all the time; I just thought he was riding, too. This is what happens when we've been living in "crisis mode" to one degree or another for about five years; we've been putting off every expense, including keeping the kids in rideable bikes. For a year and a half, I was only living with my family half the time, and the kids weren't even playing outside. And since we've moved, it feels like all I've done is worry about money and try to figure out what we are going to do with the old house.

He's tall enough now to stand over the top tube, which was a little startling. So they will check out the inner tubes which may well have dry rot by now, brake pads, etc. The bike was decked out for commuting, with a water bottle cage, frame pump, lights, lock, and upright handlebars. So I stripped some of that stuff off of it. I guess I'm giving up hopes of ever riding it again.

I used to be so into bikes — reading bike magazines, taking road rides on weekends, trail riding occasionally. Somehow all that ended, for various reasons. One reason is that most of the bike makers that I really liked, such as Cannondale, stopped making bikes in America, and I'm still disgusted about that. Another is that starting about 2001 we moved to places that I didn't find to be very bike-friendly. I never found long road rides around Saginaw like I used to have in Ann Arbor. While I used to love commuting from my old apartment on West Hoover in Ann Arbor, to the medical campus, when we moved out to Medford, there didn't seem to be a good, safe route to my job on South Industrial. Then in Saginaw, I worked from home at first, and later had a 45-minute commute to Dow in Midland, which would not have worked by bike at all. And now I'm about 13 miles away from my job, at least as I-94 runs. That doesn't seem all that far, but I really can't imagine a safe bike route. That part of town is extremely hostile to cycling.

So maybe Sam can use it, now. But I really have to find a way to get some regular exercise. My regular walks in downtown Saginaw were doing a lot to put a floor under my physical health and prop up my tendency to fall into depressive spirals in times of great stress. And I am getting nearly no exercise at all, other than the occasional shopping trip.

Joshua has been asking to learn how to play electric bass. So I'm going to get

him set up with my old bass downstairs and give him access to a lesson CD and book. We'll see if he actually shows any inclination to work at it.

I invited someone I know only through Twitter, ___@verysmallanna___, to join us for a podcast episode. She's a pastry chef and, I think, a millennial, in New York City. She has a podcast with a couple other folks, the Bread Line Podcast. It seems we have some common interests. So I'm excited to talk to her. She's free to record on Tuesday or Wednesday night, so we'll try that.

I'm not sure yet what we're going to do for tomorrow's show.

A Very Specific Memory

A few days ago, my father sent me a picture which was sent to him by a relative; it shows my family in Washington state. I'm wearing an R.E.M. concert t-shirt. Because of the t-shirt, and because of Google, and my memories of the show, I was able to place the exact date. I was seventeen years old. The show was at the Paramount theater in Seattle, July 12, 1985. I would have been 17, and this would have been the summer between my Senior year in high school and starting college in the fall. I don't remember a lot of details about that trip, but I just happen to have kept a journal, which I still have. It's short on dates and times and locations, but contained enough detail to remind me where we were, and when. I should transcribe it and find some extracts that I can add to the piles of writings that I'm trying to hammer into book-length manuscripts.

Books, Music, Movies, and TV Shows Discussed This Week

This list does not include books, chapters of books, or other works that I only mentioned briefly in the text above.

- *Iron Monkey* (1993 Film)
- *Icehenge* by Kim Stanley Robinson
- *Hacker's Delight, Second Edition* by Henry S. Warren
- *Daughter of Dreams* by Michael Moorcock, in *Elric: The Moonbeam Roads* by Michael Moorcock, Edited by John Davey (Gollancz Michael Moorcock Collection)
- *The 36th Chamber of Shaolin* (1978 Film)
- *Dragon Ball Z: Broly – The Legendary Super Saiyan* (1993 Film, 2018 Fathom Events Theatrical Release)

Pittsfield Township, Michigan
The Week Ending Saturday, September 15th, 2018