The Little Printf

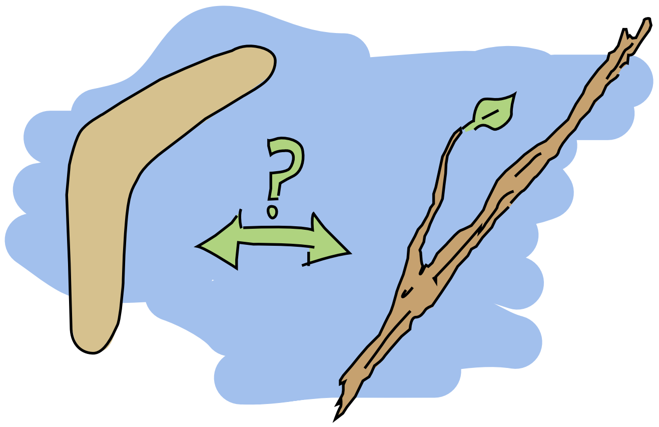
why do we code?

by Fred Hebert

Chapter 1

I've been lucky enough to have been born before computers and video games were ubiquitous. I had the luck to play outdoors with friends and my brother, and of inventing our own games.

We could be our own heroes, use a twig that would instantly become a bow, a gun, a sword, or a telescope. It could be anything, except maybe a boomerang because once you throw the stick away, you have to go fetch it back.



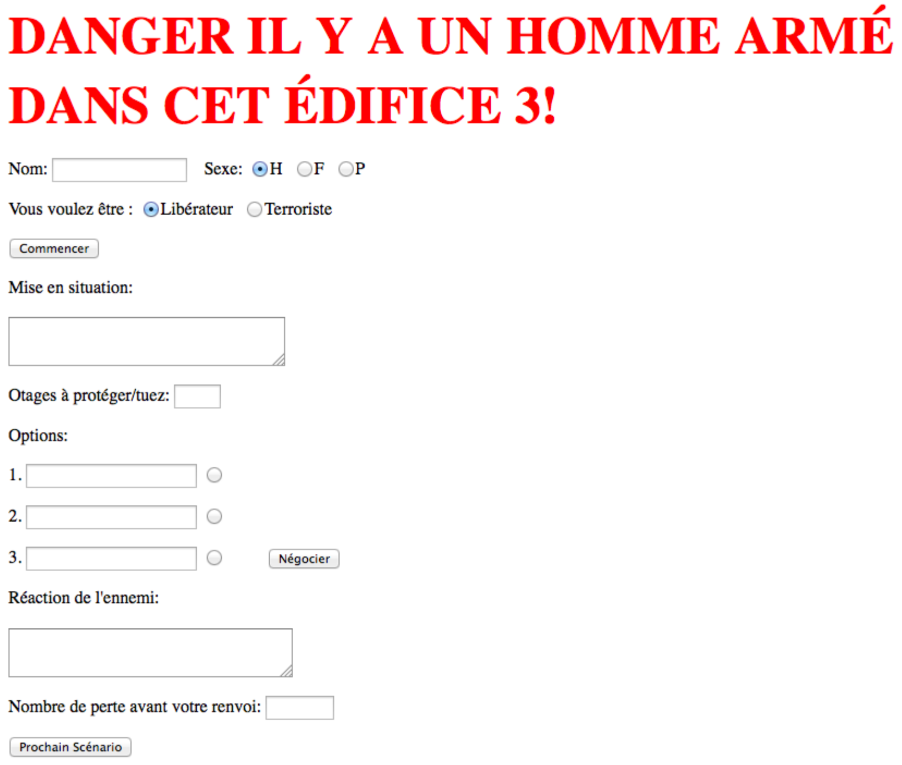
At some point I grew up, and it became embarrassing to play that way. You can't treat a pinecone as a grenade and pretend to have magical powers when other kids think being an adult is cool. You just don't fit in anymore. You eventually get pressured into growing up. Still, that's a very lucky childhood.

At some point I got the chance to play video games, and to use computers. *There* could be the imaginary world you had wanted all this time, materialized in front of you. It's consuming you, and for a moment you live a different life.

But there's something particular about most video games: you don't create, you react, you consume. I eventually did improvisational theatre as a teenager. Then, again, it was okay to be with people and create and pretend out of nothing.

Of course, improvisational theatre in Quebec is different; there's an ice rink in there – everything's hockey.

When I got to a vocational college to study multimedia from 2005 to 2008, I eventually tripped into programming work. I found it amazing! Creativity was there again, and it could get me money! I then designed the mechanism of my first game, and it blew my mind.



DANGER IL Y A UN HOMME ARMÉ DANS CET ÉDIFICE 3!

Nom: Sexe: [✓ H] F P

Vous voulez être : [✓ Libérateur] Terroriste

Commencer

Mise en situation:

Otages à protéger/tuez:

Options:

1.

2.

3. Négocier

Réaction de l'ennemi:

Nombre de perte avant votre renvoi:

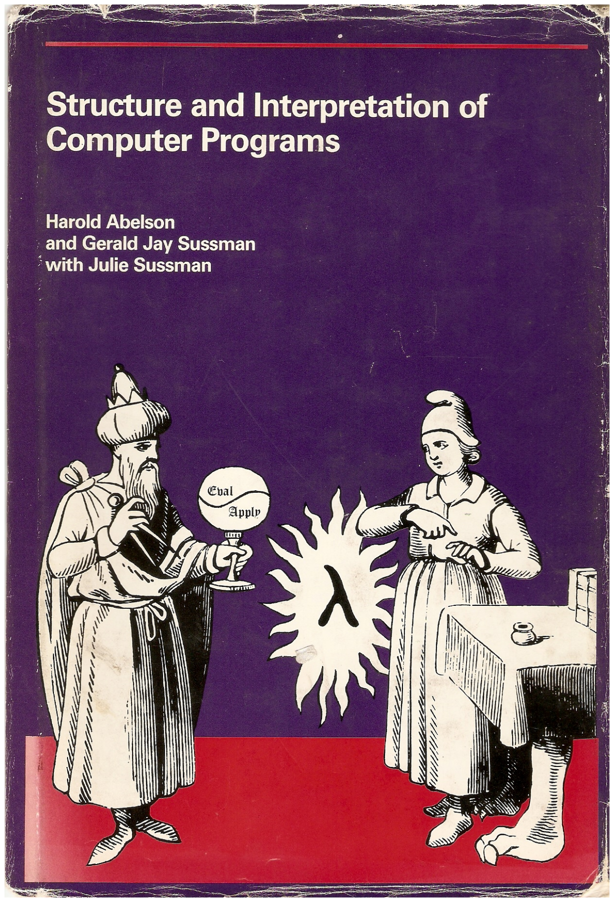
Prochain Scénario

”That’s not a real videogame,” I was told. ”That’s just an HTML form. You should have used an array for the text and options it would have been better. The code needs cleaning up.”

I was a bit disheartened; the game was really about the 11 pages of text I had written for the ”choose your adventure” aspect of it. But I realized that if I wanted to make stuff more people thought was good, I’d have to learn a lot.

I’d have to learn ”real programming”. Move from JScript in a GUI toolkit to something better, like PHP. So I learned that, along with Javascript. Then eventually I was told to learn how to do real programming again; PHP is terrible. I was told to maybe try Python, which I then learned.

But real programmers knew fancier stuff, and python’s lambdas didn’t cut it, object-oriented programming was not where you wanted to be. Reading SICP would be the next good step, I was told, because it was like the bible of computer science.



That got me to Scheme. And I got the K&R book because real programmers in the real world did C, and I registered for part time classes at my local university while juggling them with work, because real programmers knew data structures and math, which I learned to some extent. I started reading papers and books, because real programmers stayed up to date and knew fancy algorithms.

Somewhere through that I picked up Erlang and started making a career out of it. I wrote a book on it. Curiously enough, nobody ever questioned if I were a real author, or a real writer, or a real illustrator. Hell, I got a job teaching Erlang without ever having used it in a production system.

Chapter 2

So I lived my life flying around the world, telling people how to do things I had sometimes never done myself, while everyone suddenly seemed to believe I was a real programmer because of things I did that were mostly not related to programming in the first place.

One day, I was stuck in an airport coming back from a conference, furiously typing at a terminal, when an odd, gentle voice asked me:

"If you please, design me a system!"

"What?!"

"Design me a system!"

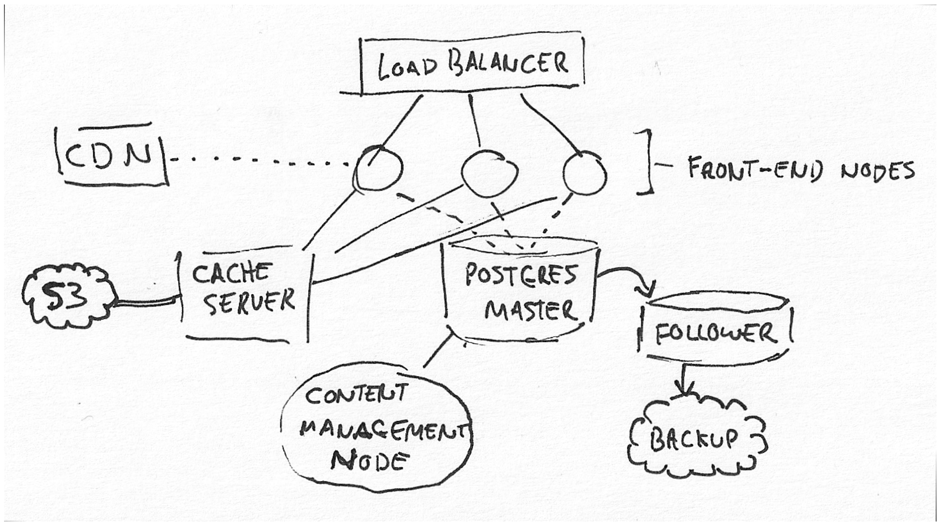
I looked up from my screen, surprised by the request. I looked around and saw this kid who aspired to be a developer and wanted me to call him "printf", which I felt was very stupid and gimmicky. He looked a bit like this:

"I don't know computers much yet, but it seems you do. I want to write programs and blog about them and have people use and read them. Please, design me a system!"

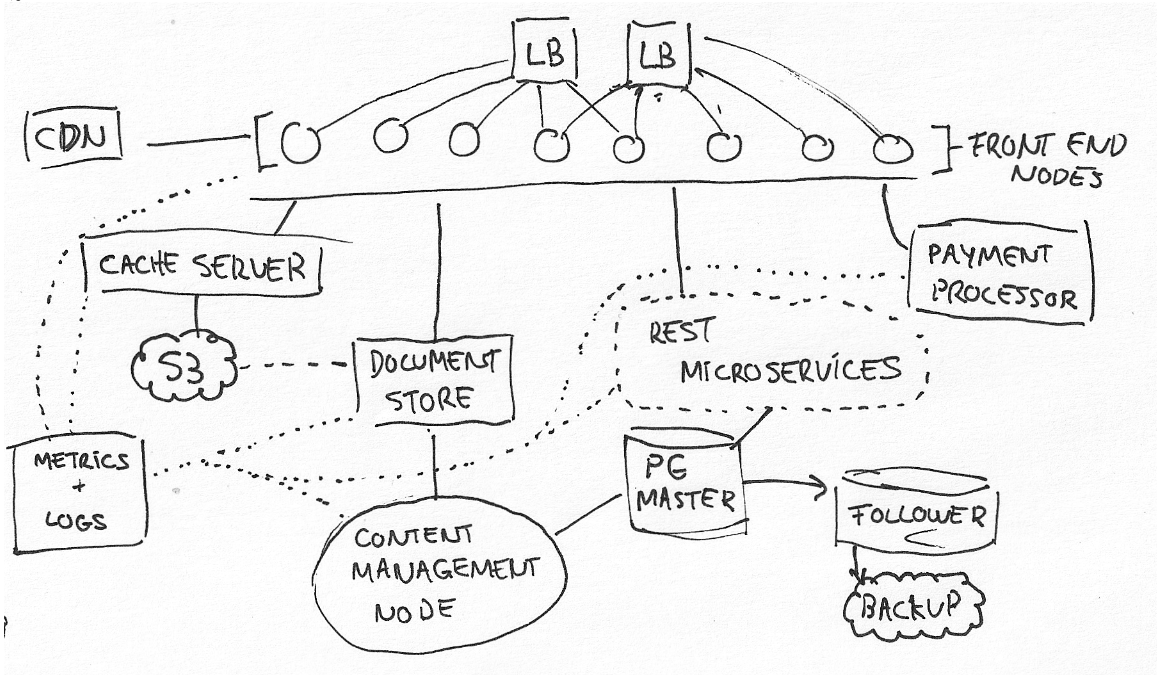
Now that was a surprising request, and I had been awake for 20 hours by then, not too sure I fully understood or felt like it. I told him systems were hard. I didn't know what he wanted to do, how he wanted it to fail, how many readers it should support, where he'd want to host it, and I could therefore not design a proper system with so little information.

"That doesn't matter. Design me a system."

So I made the following architecture diagram:



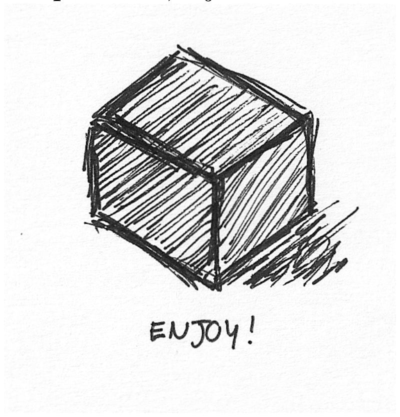
He looked at it and said "No, this system is not good enough. Make me another." So I did:



and I gave him a rundown of how it would work.

My new friend smiled politely. "That is not what I want, it's way too complex and does a lot of stuff I don't need"

I felt a bit insulted, having considered redundancy, monitoring, backups, caches and other mechanisms to reduce load, external payment processor for legal protection, failovers, easy deployment, and so on. I could have charged decent money as a consulting fee for that! Out of patience, I just drew this:



And I added: "this is your design. The system you want is inside the black box," hoping this shitty answer would have him leave me alone. But I was surprised to hear back:

"That is exactly the way I wanted it!"

And that is how I made the acquaintance of the little printf.

Chapter 3

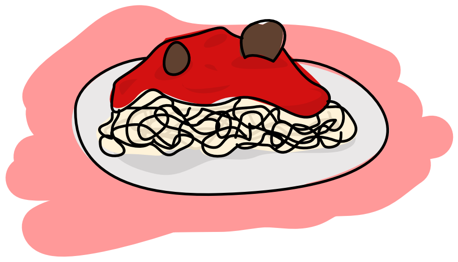
I soon learned of this little guy's portfolio. In his repositories were only small programs, simple web pages with forms, trivial command line utils. They would be unspectacular, would come into being, and no sooner disappear.

Then at some point, he started working on a bigger program, that used multiple modules. It needed sockets, accessed the disk, talked to an actual database. When it first built and ran properly, little printf was amazed. But the program was not enough yet.

It needed refactorings, better tests, documentation, linting and analysis. The program would run for a while, and one morning, it crashed.

And it crashed again, and again.

The configurations were wrong, the logs would not rotate, the disk had unpredictable speed, the network would get the hiccups, bugs would show up, the encodings would be confused, the database needed vacuuming, transactions would hang, certificates would expire, CVEs would keep coming, and the metrics would remain silent.



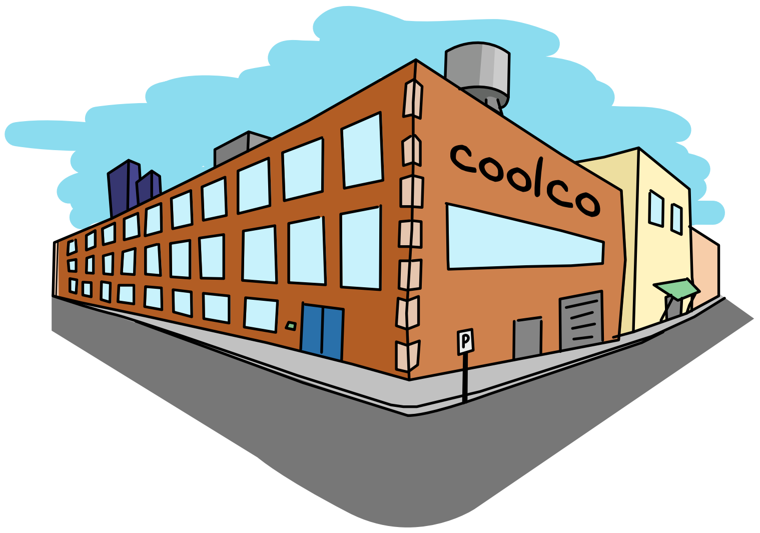
It kept turning to Spaghetti.

He told me: "the fact is I didn't know anything! I ought to have judged by my needs. I got the hubris of writing a fancy system, and I spent so much time fixing it, it felt like it cancelled the time it saved me. Still, I should have known what a wonderful thing it was."

One morning, he decided to leave his office. "Goodbye", he said to a blinkenlight that seemed to have burnt out. He left to see what the world of software had to offer aside from his messy little server.

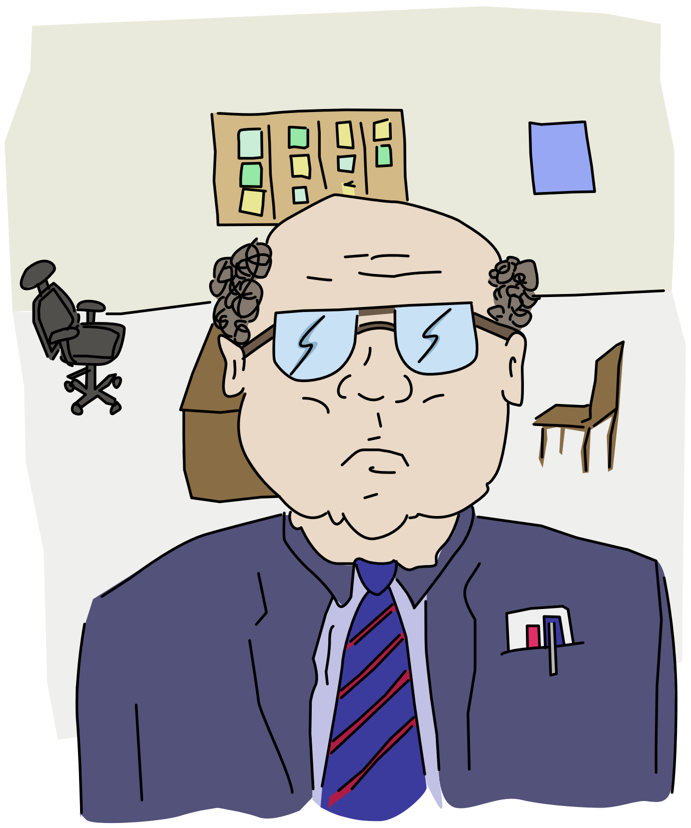
The logs would keep accumulating, until the hard drive would fill no more.

Chapter 4



He went to a workspace, looking for experienced developers from whom to get tips and help.

The first one he met was a very proud senior engineer who seemed to feel rather superior.



"Ah, here comes a learner! Welcome to my domain, of which I am the expert" he said.

"An expert?" Little printf asked. "Does this mean you can program anything and everything?"

"Yes!" the expert answered. He added "Well almost; I only program programs that are worth programming. I don't lose my time on trivialities. Many programs I have never written but could write with all the ease in the world."

"Ah, so could you help me with my system?" As soon as the little printf started explaining his business, the domain expert interrupted him:

"I'm sorry, but I don't really see the point of doing that."

”Why not?”

”Experience. I am good at programming the things I program, and I program things I am good at. By getting better at this fairly restricted set of things I’m already good at, I make sure I’m more valuable than ever at it. Call it job security, call it survival of the fittest, but that’s how I roll.”

”And why can’t you help me?”

”Well you see, taking my time away to help you means I divert important self-investment into furthering the progress of others – that’s a losing strategy for me. The best way to learn for you is the way I took myself: struggle very hard and figure it out yourself. It helps forge character.”

”That doesn’t seem very efficient...”

”Well you can go to school and learn, or you can learn on your own. Really what it does is weed out the lazy people who just want it easy, and forces everyone who stays here to be those who really deserve it. The moment we let moochers in, the very value of the work I produce goes down with it.”

”Do you not think cooperation or colleagues could help you?”

”Not really. I work best when left alone and not being distracted. Every time I end up forced working with others, it’s nigh impossible to get our stuff working together. Out of exasperation, I grab their work and rewrite most of it in a sane way; then it works right.”

Little printfs was surprised to meet an expert who seemed so disinterested in helping others, yet so annoyed by their perceived lack of skill. It was a bit sad that this man narrowed his vision of himself to just the one area he knew, to the point where he didn’t do anything else than create problems for himself to fix!

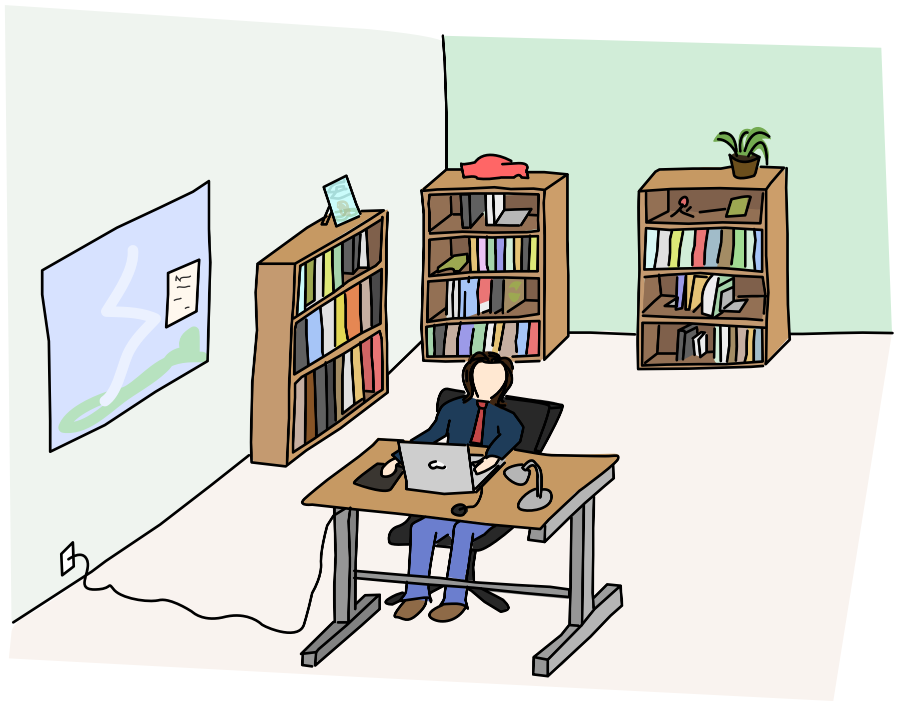
”I see... well I guess I’m happy you won’t give me your help”, said my little friend.

”What do you mean?” asked the meritocratic man, whose value seemed suddenly downgraded. ”Don’t you think the work I do is interesting?”

”Oh that I do. It just seems like you would see me as a hindrance and annoyance more than anything else, and what I am looking for is help, not affliction."

And little printf left swiftly, leaving the expert to realize he had made himself untouchable in more ways than just his job security.

Chapter 5



On his way, little printf went in front of the door to an office occupied by a man surrounded by thick hardcover books, with fancy images on them like wizards and dragons and fractals and mathematical patterns.

"Nice books, sir", said printf

"Thanks. I think they're essential material for programmers. If you don't have them, you're not really a pro"

"I guess I'm not a pro then", said little printf. "Which one is your favorite?"

"Oh, well I haven't read most of them."

"Are you not a good programmer then?"

”No, I am not.” The developer proudly added: ”In fact, I'm a terrible programmer.”

”That's a shame”, said little printf, who continued: ”I'm getting better myself.”

”Have you heard of the Dunning-Kruger effect?”, asked the man.

”No, what is it?”

”It's a cognitive bias thing. It basically says that people who are less competent tend to overestimate their qualifications, and people who are competent tend to systemically underestimate theirs.”

”So if I think I'm getting better, I'm probably not great”

”Yeah, exactly. You're probably bad. On the other hand, *I* openly say I'm a terrible programmer. But according to Dunning-Kruger, I'm probably underestimating myself, and that makes me a good developer, don't you see?”

”I guess?”

”That's because self-deprecation is a vital tool of the developer. The moment you feel you're good, you relax and stop improving.”

”Doesn't this mean that the moment you feel good about yourself, you're on your way to failure and then you should feel bad?”

”yes. But the way to go about this is to say that everything is terrible, even if you have no solutions to offer. That way you look smart, but don't have much to contribute.”

”What do you mean?”

”Say I go online and see a project I dislike. The trick is to point out everything that is wrong, give no more information than that. You can probably subtly point ways in which the person who did the thing is an idiot and get away with it.”

”And how is anyone better for this?”

”Well I like to think they are better for knowing they're on the wrong track, and I'm better off for showing them that. It's a bit of smoke and mirrors. Nobody knows what they're doing but that way it looks like *I* do."

"And what happens when you are asked for help and can't do anything about it?"

"That's when you go back to saying everything is terrible; you have too much yak shaving to do, improving other things, and being overly pessimistic. They're on their own."

"So this is all posturing? You're gaming your way through? You're the person who pretends to be incompetent at things they know, which makes people who actually know nothing there feel even worse, and you're the person who pretends to be competent at things you don't know, so that people trying to improve there also feel bad."

"In any case, competence has very little to do with it. Reputation is pretty important though. People hire friends, and people who aren't liked and non-essential get fired first; try to change the system and you become disliked. It's all a very social game. It's how it works in the industry, and probably in academia too, though I wouldn't know, now would I? It's all about who you know, selling yourself, your own personal brand you know? That's how you get jobs in the business."

"If this is how things are and that you must feel bad and make others feel bad to do well, maybe I don't want a job in the business", said little printf, before walking out.

Chapter 6

During the time that would have been lunch break, Printf interrupted a person who had seemingly forgotten to eat their lunch, a sandwich growing cold by the minute, while sitting at their desk and looking at their screen.

That seemed like quite a busy person who might have known what they were doing. Printf asked:

”If a primary database can fail, can the follower fail too?”

”Everything you run”, the person said, ”can and will sooner or later fail.”

”Even the things telling you things have failed?”

”Yes, even these ones. All large systems are in some state of partial failure at any given time.”

”Then, trying to make reliable systems, what use is it?”

The person did not know, for at that moment, they were trying to answer a page for the sky falling out onto their head due to a broken cloud, wondering the same thing.

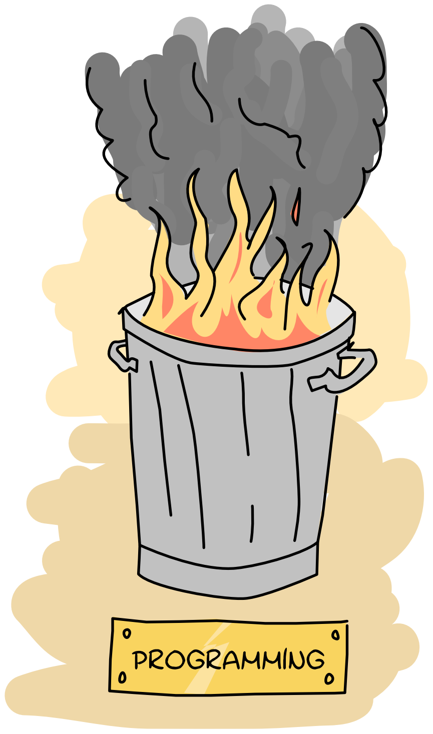
”Then making reliable systems, what use is it?” pressed little Printf again

Upset as the person was dealing with a production issue, with this kid not letting go and a sandwich going to waste, the person impatiently shot back:

”It’s of no use at all. Programming is all shit anyway.”

”oh!”, he gasped.

Then there was a moment of complete silence.



The little guy responded, with a hint of resent:

"I don't believe you. Programs are fragile, but programmers can make good efforts and make things better and useful."

No answer came back. At that point the person had opened the document explaining how to boot a new copy of the whole cluster from scratch, and things seemed to go from bad to worse.

"And you actually believe good reliable prog-"

"Oh no!" the person said. "No no no! I don't believe in good or reliable programs! Not anymore! They're all terrible! I just told you the first thing that came to my head because I'm dealing with one of these shitty systems right now. Don't you see I'm trying to keep this stuff running? *This* shit is actually of consequence."

Printf stared back, with a shocked expression.

”Actually of consequence? You talk just like a ’real programmer’.”

He added:

”You mix everything up, confuse everything. There’s been millions of programs, and for years they’ve been running and failing just the same. And people have used them and needed them. And I know of some programs that run nowhere but on a single laptop, and in a single mistake could destroy entire communities, without even noticing. And you think that this is not important?”

The person remained silent.

Chapter 7

motor-mvc, GoQuery, cometeor, quadrangular JS, some japanese soundy thing, alchemist, bongodb, mochascript, walktime.js, portasql, ...

The fourth workspace my friend visited had a man whose computer was covered in so many stickers nobody could tell what brand it was.

"motor-mvc, quadrangular JS, GoQuery, cometeor, some japanese soundy thing, ..."

"Hi," interrupted printf. "What are you doing?"

"alchemist, bongodb, mochascript, walktime.js, portasql, ...", the man kept going

"What are you doing?", he asked again, louder this time.

"Oh, I'm trying out new frameworks, tools, databases, languages."

"Whoa, you seem to be going fast, maybe as fast as 10 programmers put together!"

"yes! well, the industry moves so very fast!", he looked at his phone for a second, and added "there! the cardboard.io framework came up with version 3.5 which broke compatibility with 3.4 and this yielded 4 forks in the community! I have to try them all to know which to choose!"

"and what do you do learning all of these?"

"I'm an early adopter. If you don't stay up to date you get stuck writing COBOL or MUMPS for a living. You want to find the next big thing, and ride the wave to the top!"

"Has it ever worked?"

"Oh yes! I found out about Rails before it got big, and I figured out node.js before it was popular, and I was on the first beta copies of redis and mongodb and riak! I was the first one to use vagrant and then I got us to switch to docker but of course now it's all about unikernels.."

"Cool, and all these things you were at the forefront of, how did it pay off?"

"oh it didn't; by the time rails became huge I had moved on to the next big thing so I didn't get left behind. Similarly for the other ones. Here's hoping for unikernels though"

"I see", added little printf, pensively. "What problems do you solve with all of these frameworks?"

"Oh, I make sure we don't use something that is not going to be big, so that this company doesn't get to bet on technologies that have no future. It's very important work, because if you don't do that, you can't find anyone to hire except old grey beards behind the times, and you want self-motivated go-getters, who are also early adopters.", said the man.

"That is funny", chimed our friend.

"It is very hard! in the startup world, if you want a-players, you need good technology to bring them in! Otherwise you're stuck with inflexible laggards. Nobody wants to be an inflexible laggard."

The little printf interjected: "No, that's not what I mean," and he then added "I mean it's funny that tools are meant to solve problems for us, but for you, the tools themselves have become a problem."

And while the man stood there in silence (on his new cool treadmill desk), little printf hopped out of the room.

Chapter 8

In the next office over sat a tired employee, with dozens of empty coffee cups, slouched over, typing angrily.

"Hi," said little printf.

The woman didn't stop what she was doing. She kept typing furiously.

"Hello?" he asked again.

The woman stopped at once, got a flask out of a drawer in her desk, and took a swig.

"I have a terrible job," she said. "I do devops. It started okay, where I'd mostly develop and then sometimes debug stuff, but as time moved on, it got worse and worse. I started fighting fires in our stack, and then more fires kept happening. I got rid of most of them, pulling small miracles here and there to then meet the deadlines on dev stuff I also had to do"

"And did they hire anyone to help?"

"No, that's the thing. Small fires kept happening here and there, and because of the time I took to fight them, I couldn't be as careful as before with the dev stuff, so I created more fires all the time. Now I'm fighting fires all day and all night and I hate my job"

"Why doesn't your employer do anything?"

"I'm good at my job, and I managed to keep things under control long enough that everyone got used to it. *When you make a habit of small miracles, people get used to it. Then you're stuck doing miracles all the time or they will think you won't do your job at all.*"

"That sounds very sad"

"It is; and because you're the most familiar person with these fires, you get to only work on them more and more, until your employer hires someone else to cover your old job, the one you loved. *If you care hard enough about your work to be the one doing the stuff everyone else hates, you're thanked by doing more and more of that work you don't like, until that's all you do. And then there's nothing left for you to enjoy."*

"Then you're unlucky", said little printf.

And her pager went off again.

"That woman," said little printf to himself, as he continued farther on his journey, "that woman would be scorned by all the others: by the senior expert, by the rockstar developer, by the serial early adopter. Nevertheless she is the only one of them all who seems helpful. Perhaps that is because she is thinking of something else besides herself."

Chapter 9

At the corner of the building, printf found a large office with big windows giving a stunning view of the area. In it sat an old gentleman with reams of documentation on his desk.

"Ah, here comes a developer" exclaimed the man, as printf stood in the doorway. "Come in!"

Looking through the windows, little printf noticed that they were full of writing. With the help of a dry-erase pen, the view to the outside world was masked by tons of circles, arrows, cylinders, and clouds. While it was curious the man needed clouds drawn where real ones could be seen outdoors, the whole ensemble was more intriguing.

"What is this?", asked our friend, pointing at the windows.

"Oh this? This is our production system!" Said the man, not once thinking the question was about the outside world. "I am a software architect."

"What's a software architect?"

"Mostly, it's someone who knows how to best structure and coordinates the components of a large system so they all fit together well. It's someone who has to know about databases, languages, frameworks, editors, serialization formats, protocols, and concepts like encapsulation and separation of concerns."

"That is very interesting!" said little printf, "here is someone who can answer all my questions!" He glanced at the architecture diagrams. "Your system is very impressive. Is it running very fast?"

"I couldn't tell you," said the architect. "It should, though"

"How's the code then, is it good?"

"I couldn't tell you"

"Are the users happy about it?"

"I couldn't tell you either, I'm afraid"

"But you're a software architect!"

"Exactly! But I am not a developer. It is not the architect who goes and writes the modules and classes, combines the libraries. The software architect is much too important to go around touching code. But he talks with programmers and developers, asks them questions, provides them guidance. And if the problem is looking interesting enough, the architect takes over the planning."

"And why is that?"

"Because we are more experienced. We know more about systems and what works or not. Developers can then be an extension of our knowledge to produce great systems!"

"But how do you know if things are going well without getting involved with code?"

"We trust the developers"

”So you trust them to implement your ideas correctly, but not enough to come up with their own ideas?”

The software architect was visibly shaken by this comment. ”I guess I might have been a bit disconnected”, he finally admitted. ”The problem is that after a while you are asked to work with ideas so much you don’t have a good way to get them tested or verified...” he stared down, pensively. ”Sometimes a software architect does neither software nor architecture, it seems.”

Little printf left the room, and being done with his visit, exited the building.

Chapter 10

Little printf, once outside, met a man collecting money for some charity.

"Hi," said the man. "how would you feel about helping someone today?"

"It would probably make me feel better," said little printf back. "I have been in this office all day, and now I'm more confused than ever."

"Ah I see. These people are all developers. They are not really helpful, are they? What they love to say is that they're changing the world, and they pretty much succeed at that, in fact."

"Why does it feel so awkward, then?" Asked little printf.

"Well, the best they do is often help convert some people's jobs into programs, or make everyone's leisure more leisurely. Software is eating the world and that changes its face for sure... but deep down it's the same old world, with a mangled face. The reason it feels awkward is that changing in that way doesn't mean things are getting any better. We have the same flaws and problems we always had, the same holes to fill deep down inside.”

”So how can I feel better?” Little Printf was visibly anxious.

The man thought for a while, and offered printf to come help him help others, as this was this man’s way of feeling better. During the afternoon, printf told the man about his problems and his adventure. After a long silence, the man said:

”The games people play, the roles and reputations they chase and entertain, the fleeting pleasure they derive from solving intricate problems, is all fun for a while. Ultimately though, if you do not solve anything worthwhile, if you forget about the people involved, it’s never gonna be truly fulfilling.

”And that may be fine, and it might not be, and you may or may not get that from somewhere else than your workplace when you grow up. Work can be work; it can be for the money, it can be for the fun of it. That’s okay. As long as you manage to get that fulfillment somewhere in your life.

”In the end though, it is only when you solve problems with a human face that you can feel truly right; What is essential is invisible to the computer.

”It is the time you have spent on your system that makes it so important”, the man added, ”and when you lost sight of why it made sense to spend time on it, when it became a game of pride, it caused more grief than relief.

”Developers have often forgotten this truth; If you lose sight of things, working on your system becomes its own problem, and the most effective solution is to get rid of the system, given it’s the problem.”

”It is only when you solve problems with a human face that you can feel truly right”, repeated little printf to himself, so he would remember.

Chapter 11

Printf, who's now sitting right in front of me, is on his way home. Talking with him made me realize how much of what I do flies in the face of what I liked, what I started programming for. Each of the people Printf met are roles I see myself taking one day or another over time. I was encouraged by them to become them, and probably encouraged people to do the same.

Where I got dragged in the game of trying to become a real programmer, Printf didn't. He said he was okay with not being a real programmer, that he preferred to be a programmer with a human face.

Today I'm stuck in the situation where I look back, and have to figure out if I can, too, become a programmer with a human face; or if everything I do is just a job. There doesn't seem to be too much that's worthwhile in-between.

In any case, where printf felt he didn't need to be a real programmer, I think I feel the same now.