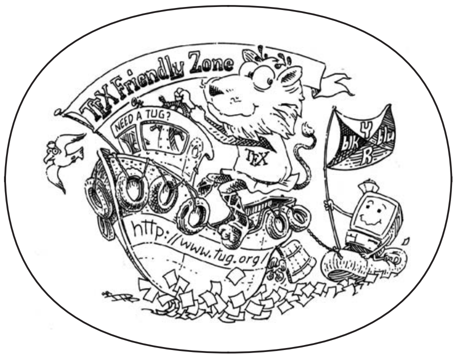
MY ROAD OF 2014

ZHIWEI YAN



Strugling for Happy Lives and Free Sprits

[May 5, 2014 at 10:40 – `classicthesis` version 4.1]

CONTENTS

Part I

SPRING 2014

1 JANUARY - 2014

1.1 REMOVE COMMENTS FROM FILES IN C

You use the tool *gcc* to remove the comments in files.

```
gcc -fpreprocessed -dD -E test.c
```

edited on2014-03-05.

1.2 BIG DATA ARE WE MAKING A BIG MISTAKE

edited on 2014-04-02.

Big data: are we making a big mistake? By Tim Harford
  http://www.ft.com/cms/s/2/21a6e7d8-b479-11e3-a09a-00144feabdc0.html#axzz2xS1VXiUc

2 FEB - 2014

2.1 HANDLE EXCEL FILES IN C

Perhaps, we begin to handle Microsoft Excel files in C. We do the search over the Internet. There are several packages or libraries that can be used.

1. xlsLIB, C++/C library to construct Excel .xls files in code. Detail in http://sourceforge.net/projects/xlslib/files/
2. Office (2007) Open XML File Formats. Detail in http://msdn.microsoft.com/en-us/library/aa338205.aspx
3. CSV formats. The format do not handle Chinese characters very well, since it includes the information of encoding in itself.

The last two methods are recommended strongly.

2.2 20 YEARS OF WEB DEVELOPMENT

I read the ariticle *20 Years of Web Development*, written by Reuven M. Lerner, a well-known columnist of Linux Journal. The idea from the article make me feel something inside, same as the author.

If I can not get my fix of e-mail, or blogs, or newspapers or Hacker News, I feel cut off from the rest of the world. I'd like to take the opportunity to look at where Web technologies have come from, and where they're going.

2.2.1 *In the Beginning*

In the beginning, the Web was static. A Web brower requested a document from the server. Instead, the server could lie to the browser, creating a document on the fly, by executing a program. The Web servers invoke external programs in a 'cgi-bin' directory. The 'cgi' programs could be written in C and shell, later, in Perl and Tcl in those days.

For example, MIT students put their newspapers on the Web in 1993, and made it possible for people to search through the newspaper's archives.

In 1995, no one really thought the web programming as serious software development. At that time, applications were serious desktop software. I began to understand where the Web was head when you learned about relational databases and understand how powerful Web applications could be when connected to a powerful, flexible system for storing and retrieving data. DB ranking list is at http://db-engines.com/en/ranking.

And at that point, the browser could be the beginning of a new platform for applications. [PPTs also turn the bullets scheme into words-scheme.] Nowadays, the vision has turned into a reality.

edited on 2014-02-27

2.2.2 *Libraries and Frameworks*

It did not take long before Web development really began to take off. The growth of the Web happened at about the same time that the term *open source* was coined. Open-source operating systems and languages - in those days, Perl, Python and PHP - grew in popularity, both because they were free of charge and because they offered enormous numbers of standardized, debugged and highly useful libraries.

3 MARCH - 2014

3.1 REMOVE COMMENTS FROM FILES IN C

You use the tool *gcc* to remove the comments in files.

```
gcc -fpreprocessed -dD -E test.c
```

edited on2014-03-05.

3.2 COLOR SCHEME IN BORLAND C++

The backgroud is blue, 0, 0, 168, 0000A8. The text is yellow, FFFF44, 255,255,68

3.3 SPLIT TESTING

Author: Reuven M. Lerner.

The rate at which your web visitors become customers is called the *conversion rate*, and it's probably the top priority for Web-based business. What leads people to convert more more often? That's a question to which an entire industry of Internet marketers and 'conversion optimization' and 'converion optimization' experts. One of the most popular, and effective, ways to check the effectiveness of your copy is to do 'split testing', sometimes known as 'A/B testing'. The most important thing to keep in mind when you are doing split testing is that you are trying to get users to do something. What that something is depends on your site. The goal at Amazon is to get you to buy things. The goal at Google is to get you to click on ads.

Once visitors have achieved your goal, they have been 'converted' into customers. The goal is to increase the number of such conversions - giving you more customers, subscribers or users of your system. The key insight with split testing is that by changing the text, graphics and even layout of your page, the number of conversions will chang as well.

In order for split testing to work, you need to do several things:

1. Define what counts as a conversion. This often is described in terms of the user arriving at a particular page on the site, such as a 'thank you' for shopping that appear after a successful salee.

2. Define the control and alternate texts.

3. Wait for enough users to see both.
4. Analyze the numbers.

edited on 2014-03-07.

3.4 BEHAVIORS YOU NEVER WANT TO SEE IN A LEADER

1. Complaining. One of the many challenges an organizational leader faces is buy-in from his people. It establishes your reputation as a gossip hound. When people know where you stand, they also know what you stand for.
2. Emotional volatility. Not to be confused with expressing emotion. It also requires understanding different personalities, because some people learn easier after having a heart-to-heart converstation while others need a more direct kick in the buttocks.
3. Playing "nice". People need a leader, not a friend. Friends help you out with your business; leaders help you fit in according to the business. Leaders seek to understand and align your values and goals with the company's vision and strategy.
4. Minding other people's matters (micromanagement). Starting out as an entrepreneur, you have to wear all the hats, but as your company grows, you are now focused on higher-level planning. It's not easy removing the tactical, operational and strategic hats. If you want your company to grow then you must focus on what only you can affect – and let your people do the same.

3.5 ENGLISH PARAGRAPHS

The words in the following are extracted from the GitHub CEO & Co-Founder, Chris Wanstrath. Making sure GitHub employees are getting the right feedback and have a safe way to voice their concerns is a primary focus of the company. We wish Julie well in her future endeavors.

3.6 A STARTLINGLY SIMPLE THEORY ABOUT THE MISSING MALAYSIA AIRLINES JET

They had to ditch in the ocean. He just didn't have the time.

3.7 TELSA CAN TOPPLE THE CAR-DEALER MONOPOLY

What is revolutionary, however, is Elon Musk's desire to build a retail network free from the franchise-dealer monopoly.

3.8 APPLE DESIGNER JONATHAN IVE TALKS ABOUT STEVE JOBS AND NEW PRODUCTS - TIME

Many of us spend more time with his screens than with our families. Some of us like his screens more than our families.

Objects and their manufacture are inseparable. You understand a product if you understand how it's made. Apple is notorious for making the insides of its machines look as good as the outside. We did it because we cared. I want to know what things are for, how they work, what they can or should be made of, before I even begin to think what they should look like. More and more people do. There is a resurgence of the idea of craft. They may be revolutionary, high-tech magic boxes, but they look so elegantly.

Because when you realize how well you can make something, falling short, whether seen or not, feels like failure. He likes the idea of this interview series because he sees himself as more of a maker than a designer. His love of simplicity and directness extends beyond tech. The simple truth is, Ive hates fuss and relishes simplicity.

They remake what they saw as the bland, lazy world around them. Everyone I work with shares the same love of and respect for making, he says. We can be bitterly critical of our work.

And we would ask the same questions, have the same curiosity about things. If you do something and it turns out pretty good, then you should go do something else wonderful, not dwell on it for too long. Just figure out what's next.

Ive talks so much more about making things than designing them. The product you have in your hand, or put into your ear, or have in your pocket, is more personal than the product you have on your desk. People have an incredibly personal relationship with what we make.

Developing life-changing products is very expensive. It's pretty and doubtless costs a pretty penny to make . It's thousands and thousands of hours of struggle. It's only when you've achieved what you set out to do that you can say, We are at the beginning of a remarkable time, when a remarkable number of products will be developed. There is this almost pre-verbal, instinctive understanding about what we do, why we do it. We share the same values.

Most people won't realize that writing is a craft. You have to take your appernticeship in it like anything else. -Katherine Anne Porter

4 APRIL - 2014

4.1 WHAT IF YOU DIDN'T NEED MONEY OR ATTETION?

edited on 2014-04-01.

We do so many things for the money. It's so deeply built into our culture that it takes a real effort to realized it's the reason behind so many of our actions.

If you stop doing all these things you're just doing for the money, or the attention, what's left?

4.2 BIG DATA ARE WE MAKING A BIG MISTAKE

edited on 2014-04-02.

Big data: are we making a big mistake? By Tim Harford

`http://www.ft.com/cms/s/2/21a6e7d8-b479-11e3-a09a-00144feabdc0.html#axzz2xS1VXiUc`

Sometimes, **Big Data analysis** produces uncannily accurate results; They are cheap to collect relative to their size and they are a messy collage of datapoints collected for disparate purposes and they can be updated in real time.

1. cheap to collect
2. collage of datapoints
3. disparate purposes
4. update in real time

With enough data, "the numbers speak for themselves".

As our communication leisure and commerce have moved to the internet and the internet has moved into our phones, our cars and even our glasses or watches, life can be recorded and quantified in a way that would have been hard to imagine just a decade ago. There are a lot of small data problems that occur in big data, says Spiegelhalter. "They don't disappear because you've got lots of the stuff. They get worse."

We seek new ways to understand our lives. Figuring out what causes what is hard (impossible" some say). If you have no idea what is behind a correlation, you have no idea what might cause that correlation to break down.

Statisticians have spent the past 200 years figuring out what traps lie in wait when we try to understand the world through data. They cared about correlation rather than causation. When it comes to data, size isn't everything.

US-based Twitter users were disproportionately young urban or suburban and black. There must always be a question about who and what is missing" especially with a messy pile of found data. Twitter users are not representative of the population as a whole. Who cares about causation or sampling bias" though" when there is money to be made? We found data contain systematic biases and it takes careful thought to spot and correct for those biases.

There are two issues: sample error and sample bias. The larger the sample, the smaller the margin of error (sample error). The sample bias is more dangerous, because the sample is not randomly choosen at all. You should find an unbias sample. Find the detail in *Number sense*, wroten by Kaiser Fung. Another bias example is about the iPhone app, *Boston Street Bump*. You should find data contain systematic biases and it takes careful thought to spot and correct for those biases.

Multiple-comparisons problem: We observed pattern could have emerged at random, we call that pattern "statistically significant". "Why Most Published Research Findings Are False" There are a few cases in which analysis of very large data sets has worked miracles, like Google Translate.

Big data do not solve the problem that has obsessed statisticians and scientists for centuries: the problem of insight , of inferring what is going on, and figuring out how we might intervene to change a system for the better. To use big data to produce such answers will require large strides in statistical methods. "But nobody wants 'data'. What they want are the answers." Many contrary results are languishing in desk drawers because they just didn't seem interesting enough to publish.

"Big data" has arrived but big insights have not. The challenge now is to solve new problems and gain new answers – without making the same old statistical mistakes on a grander scale than ever. As for the idea that "with enough data" the numbers speak for themselves" – that seems hopelessly naive in data sets where spurious patterns vastly outnumber genuine discoveries.

4.3 NEW RFS – BREAKTHROUGH TECHNOLOGIES

edited on 2014-04-02.

Great companies:

1. be with a series of small wins that compound over time
2. focus on solving real problems for real customers, and not just developing technology for its own sake.
3. not to bite off an initial project that is far too big and expensive.

Breathrough Technologies

1. Energy: Cheap energy, energy storage and transmission.

2. AI:
3. Robotics: We count a self-driving car as robot.
4. Biotech:
5. Healthecare: preventative healthcare.
6. Education: Using the Internet to distribute traditional content to a wider audience. One-on-one in -person interaction.
7. Internet Infrastructure: The Internet is a transformative power. We hope use the Internet to fix government. An important trend is the API-ification of everything. As more and more businesses are accessible with a web API, the Internet becomes more and more powerful.
8. Science: new business models for basic research.
9. Transportation and housing. About half of all energy is used on transportation, and people spend a huge amount of time unhappily commuting.

4.4 BORED PEOPLE QUIT

edited on 2014-04-22.

Bored People Quit By
http://randsinrepose.com/archives/bored-people-quit/

Situation These are the people who show up when your single best engineer casually and unexpectedly announces, ″I'm quitting. I'm join my good friend to found a start-up. This is my two weeks' notice″.

Detcting Boredom

1. A decrease in productivity is a great ealy sign.
2. You ask, "Are you bored?"
3. They tell you. And you listen.

In reality, the boredom was a seed. You always need to be able to answer two questions regarding each person on your team:

1. Where are they going?
2. What are you currently doing to get them there?

There's no shit work when the work is all yours, there's just work you like to do and work you have to do. Occasinal stints on the latter are a good perspective reset for everyone on the team, but being left too long on "have to" work is a guarantee of evental boredom.

Random meetings, phone calls, interviews disturb your team members. Your attention is only half the solution. The other half is regularly keeping folks in the loop regarding your thoughts.

My gig is the care and feeding of engineers, and their productivity is my productivity. If they all leave, I have exactly no job. Your job is to help your team succeed.

4.5 DON'T FUCK UP THE CULTURE

edited on 2014-04-22.

Title: Don't Fuck Up the Culture
Author: Brian
URL: https://medium.com/p/597cde9ee9d4

Culture I thought to myself, how many company CEOs are focused on culture above all else or their culture?

Culture is simply a shared way of doing something with passion.

Our culture is the foundation for our company. The thing that will endure for 100 years, the way it has for most 100 year companies, is the culture.

The culture is what creates the foundation for all future innovation. If you break the culture, you break the machine that creates your products.

When the culture is strong, you can trust everyone to do the right thing. People can be independent and autonomous. The stronger the culture, the less corporate process a company needs. In organizations (or even in a society) where culture is weak, you need an abundance of heavy, precise rules and processes.

You make decisions by comparing to culture with your other targets. If compared to culture, they are relatively short-term. These problems will come and go. But culture is forever.

4.6 GAME SERVERS: UDP VS TCP

Title: Game servers: UDP vs TCP, http://1024monkeys.wordpress.com/2014/04/01/game-servers-udp-vs-tcp/
Author: Christoffer Lerno

The most damning property of TCP is the congestion control. You use reliable UDP instead of TCP - to get rid of its congestion contol.

1. Use HTTP/HTTPS: if you are making occasional, client-initiated stateless queries and an occasinal delay is OK.

2. Use persistent TCP sockets: if both client and server may independently send packets but an occasional delay is OK.

3. Use UDP: if both client and server may independently send packets and occasional lag is not OK.

These are mixable too. Your MMO might first use HTTP to get the latest updates, then connect to the game servers using UDP.

Never be afraid of using the best tool for a task.

Starting an action before confirmation is a typical latency/lag hiding technique.

4.7 HOW TO BECOME A GREATER DEVELOPER?

edited on 2014-04-23.

Title: How to become a greater developer, `http://peternixey.com/post/83510597580/how-to-be-a-great-software-developer`
Author: Peter Nixey

Your seniority and value as a programmer is measured not in what you know, it's measured in what you put out. The two are realted but definitely not the same. Your value is in how you move your project forward and how you empower your team to do the same.

You should aim for simplicity. Simplicity is far more easily attained by time spent working and refactoring than hours of pure thought and 'brilliance'.

Simplicity and excellence are most reliably attained by starting with something, anything that gets the job done and reworking back from that point.

Companies are built on people and teams who day in, day out, commit good code that enables others do the same. Great product is built by work horses, not dressage hourses.

Not only is their output erratic but their superiority is aspiratinal and infectious. Their arrogance bleeds toxically into the rest of the team. Your projects depend on reliable people who work in reliable work. Greate developers are not people who can produce bubble sorts or link shorteners on demand. They are the people who when harness them up to a project, never stop moving forward and inspire everyone aroubnd them to do the same. You do not need *RockStars*.

Name your functions and variables well
It is one of the MOST IMPORTANT skills in programming. Function naming is the manifestation of problem definition which is frankly the hardest part of programming.

Names are the boundary conditions on your code. Names are what you should be solving for. If you name correctly and then solve for that boundary conditions that the names creates you will almost inevitably be left with highly function code.

Function names create contracts between functions and the code that calls them. Good naming defines good architeture. Good function and variable naming makes code more readable and tightens the thousands of contracts which criss-cross your codebase. Sloppy naming means sloppy contracts, bugs, and even sloppier contracts built on top of them.

Notice how much stronger this approach is than using comments. If you change the logic there is immediate pressure on you to change the variable names. Not so with comments.

Go deep before you go wide - learn your chosen stack inside out

The marjority of the things you are trying to do have already been solved by the very stack you are already using. Most programmers waste huge amounts of time by lazily re-creating implementations of pre-exsiting functionality. You should undestand those existed technologies well that you use everyday.

Learn from those good codes

The grand chessmasters spend proportionally much more time studying previous other good chess player's games than the average players. You should develop an aesthetic appreciation for code. Simplicity is beatuiful and simplicity is what we want.

The truth is that the truth is sometimes ugly but you should always strive for beauty.

Your code has two functions: the first is its immediate job. The second is to get out of the way of everyone who comes after you and it should therefore always be optimised for readability and resilience.

Weight features on their lifetime cost, not their implementation cost

Features and architecture choices have maintenance costs that affect everything you ever build on top of them. Abstractions leak and the deeper you bury badly insulated abstractions the more things will get stained or poisoned when they leak through. Experimental architecture and shinny features should be embarked on every carefully and only for very good reasons.

Build the features you need before the features you want and be VERY careful about architecture.

Leverage of Technical Debt

Einstein once said that " there is no force so powerful in the universe as compound interst'. If you do not absolutely need them; do not write them. You are in an exploratory phase. You will pivot both on product and on technical implementation. Your initial code is scouting code. It should move you forward fast, illuminate the problem and the solution and give you just enough space to build camp.

Check and re-check your code. You should make sure your code works. It's not the testers' job and it's not your team-mates'job. It's your job. Lazily written code slows you down, increases cycle times, releases bugs and pisses everyone off.

Do actual work for at least (only) four hours every day, and you will be one of the best contributing members of your team. Proper work is work that includes no email, no hacker news, no meeting, no dicking around. It means staying focussed at least 45 minutes at a time.

Write up the things you've done and share them with the team

For example, have a tough time getting a fresh install of Postgres or ImageMagik to work? If you found it hard, the rest of your team will probably also find it hard so take a moment to throw down a few paragraphs telling them what you did and saving them the time next time.

Think of testing like armour. The more of it you wear the harder it is to hurt you but the harder it is to fight too.

Make your team better

Does your presense make your team better or worse? Does the quality of your code, your documentation and your technical skills help and improve those around you? Do you encourage and inspire your team-mates to become better developers?

Who are you?

It's not who you are underneath, it's what you do that defines you.

5 MAY - 2014

5.1 STOP WASTING USERS' TIME

Title: Stop Wasting Users' Time BY , URL:http://www.smashingmagazine.com/2014/04/25/stop-wasting-users-time/

What is the single most precious commodity in Westen society? Money? Status? I would argue it is time.

1. CAPTCHA: The Ultimate Time-Waster, any system that forces the user to prove they are human.
2. Why Are Password So Complicated: Can't we come up with a better solution than an arcane mix of uppercase, numbers and symbols? complexicity and long phrase.
3. Don't Make Users Correct 'Their' Mistakes in Forms. Sometimes we even waste the user's time when we are trying to help them. Forms are paticularly painful on touchscreens.
4. Pay Special Attention To Repetitive Tasks. For that matter, a more robust solution to 'Remember me' functionality would be nice, so that users are, in fact, remembered.
5. Help Users Process Our Content Faster: We can also make our content a lot more scannable, with better use of headings, pull-out quotes and lists.

2014-05-05.