
Technically, These Are Random Thoughts

Technically, These Are Random Thoughts

A month of responses to single-word prompts,
collected from AdatoSystems.com

by

Leon Adato

Copyright and Licensing

Dedications

To Debbie:

As with anything I have ever done that mattered, this is dedicated to my best friend and my closest confidant. Even after more than a score of years, you still mean more to me than everything else.

Contents

Forward

Technically, this is a forward to a book about some random thoughts.

The collection of thoughts and post on the following pages represent not only Leon's thoughts but the thoughts of each and every system administrator. I have no doubt that anyone reading this book has had a majority of these same thoughts, if not all of them at one point or another in their career. Leon's willingness to share these thoughts through his writing is a gift for us all, as it helps everyone who has ever felt like they were alone against a flood of barbarians at the gates of IT.

I would encourage you to read this book one thought at a time. Then, take the time to reflect on how you would have communicated that same thought to those around you. Then, write your thoughts down.

Who knows, maybe you'll have the makings of your very own book, too. With the most important message you could deliver: yours.

Thanks,

Thomas LaRock

ZeroDay: Introducing #BlogElul

tl;dr version

A daily blog-athon happens every year, with hundreds of people writing a daily post on a specific theme. This was my month's-worth of contributions in 2015.

The Technical Details

Around September every year, Jews all over the world celebrate Rosh Hashana, the Jewish New Year.

However, it's not – to put it in business terms – a year-end review. It's a job interview. It's the time when you reflect on how the past year went so that you can commit to making adjustments in attitude and behavior so that we continue to improve as people. It's high stakes stuff. Have a bad interview, and The Boss might not renew our contract.

As anyone who's been in a high-stakes interview can attest this is not something that you just show up for unprepared. You've got walk into that interview room with your facts straight, with a clear understanding of the numbers, your accomplishments, your shortcomings, etc.

Which leads me to the point of this – the month before Rosh Hashana (called "*Elul*" in Hebrew) is the time for getting one's balance sheet in order.

To help with that, a bunch of folks from all walks of life participate in #BlogElul: A daily prompt provides the theme and people riff on that – sometimes a few hundred words, sometimes an image, sometimes a poem or just a single sentence. It's something I've done for a few years now.

I thought I'd add a twist and also do an I.T. Professional's version of #BlogElul and post the essays on my technology-specific blog: www.adatosystems.com. A reflection on each of the daily prompts and what they mean in an I.T. context.

You're probably thinking "Leon, this is a Jewish thing and completely outside the scope of my experience or interest as an I.T. Professional."

To which I emphatically reply: Yes and no.

If you have worked in I.T. for more than 15 minutes, you've likely been involved in a large development project, system roll-out, or upgrade. And as the date for the big cut-over approaches, there are usually daily status updates. Consider this the notes from my status updates before the roll-out of "TheWorld v.5776".

If you are interested in joining in the fun next year you can find more information on the blog of Rabbi Phyllis Sommer, the woman who started it.

For 2015, the daily prompts were:

- Day 1: Prepare
- Day 2: Act
- Day 3: Search
- Day 4: Understand
- Day 5: Accept
- Day 6: Know
- Day 7: Be
- Day 8: Hear
- Day 9: See
- Day 10: Count
- Day 11: Trust
- Day 12: Forgive
- Day 13: Remember

- Day 14: Learn
- Day 15: Change
- Day 16: Pray
- Day 17: Awaken
- Day 18: Ask
- Day 19: Judge
- Day 20: Dare
- Day 21: Love
- Day 22: End
- Day 23: Begin
- Day 24: Hope
- Day 25: Intend
- Day 26: Create
- Day 27: Bless
- Day 28: Give
- Day 29: Return

With all of that out of the way, let's get started...

Prepare

For I.T. professionals, there is all kinds of preparations that we have learned are essential, if not extremely wise:

- As my friend Tom (SQLRockstar) says, “having backups is job #1”
- Tom also advises that “...You get paid for performance, but you keep your job with recovery.”
- Testing a process (upgrade, install, migration, etc.) before running it in production
- having current versions and data in a sandbox, development, and QA environment so you aren’t doing all your work in production

…and more…

But the first step in any preparation is choosing to do it.

The work of preparation is physical. But the act of it is mental. If you have chosen to prepare – whether “…for the worst”, for the weekend cut-over, or simply for the day – then the hardest part is already done.

Act

After 27 years in I.T., I've developed a conflicted relationship with the idea of "acting".

On the one hand, I've seen co-workers locked into an infinite loop of planning, research, testing, etc. and ultimately failing to act (the now cliché concept of "analysis paralysis"), and I've been afflicted with it myself.

And on the other, I've seen people make the mistake (sometimes career-ending) of acting too soon, without consideration or forethought or (as I discussed in the previous essay) planning and preparation.

Ultimately I have no solid advice for avoiding either anti-pattern. Except perhaps to always commit to only making a particular mistake once, and then learning everything you can from it.

As a famous story relates, there once was a student who sought out a great Rabbi for advice.

> **"Oh great Rabbi," this person implored, "What is the secret to a happy life?"**
>
> **"That's easy!" the Rabbi exclaimed, "Use good judgement."**
>
> **"OK," said the petitioner, "and how does one acquire good judgement?"**
>
> **"Also easy!" enthused the Rabbi, "Gain experience."**
>
> **"Wonderful," said the truth-seeker, feeling like they were finally getting somewhere. "And what does one do to gain experience?"**

“Ah, that is the heart of the matter,” said the Rabbi, his voice dropping to a whisper. “Use bad judgement.”

Search

In the context of IT, searches are an every hour, if not an every minute, experience. Almost nothing that we need is not literally at our fingertips, just an internet query away.

Which is interesting, since I.T. Professionals tend to make a big deal of knowing things by heart. Can you calculate subnets off the top of your head? Do you know these OS commands (and all of the sub-commands)? Can you set up this-or–that without referring to the manual?

Back in High School, one of my best friends was on the path to what would turn into a very fulfilling career as a micro-biologist. I vividly recall sitting in the hallway before school as I quizzed her on the periodic table of elements for her chemistry exams.

Several years after graduation, I reconnected with her. One of my first calls to her started with me demanding to know the atomic weight of germanium. She knew immediately what I was asking, but responded with:

“I couldn’t care less.”

I was surprised, and asked if it was an example of things you learn in school that turn out not to be important later on.

“Nope, I use that kind of thing every day.” she said.

“Then how come you don’t know it?”

“Because,” she replied. “I don’t have to know it. I simply have to know where to find it. What is actually important to know,” she explained, “Is what to do with that information once you have it.”

I think that's what differentiates experienced I.T. professionals from so-called newbies. The new comers focus on (and stress out about) specific factoids. The atomic elements that make up a particular technology.

The veterans know that it's not so much the specific commands, verbs, or syntax but the larger patterns and use-cases which make or break you as a professional.

Everything else is just a text-search away.

Understand

Information Technology is one of the few places I can think of where people who call themselves professionals can be successful even while they don't understand huge swaths of the technology they use.

I have met entire teams of server administrators who can't explain the first thing about IP addresses, or networking in general. Similarly, I have met network engineers who don't know (and don't care) how operating systems communicate.

This is partially by design, and partially by convenience. DBA's don't need to understand how packets are built up and broken down as they traverse switches and routers. In a handful of situations, they may be able to more effectively troubleshoot an issue if they did know, but most of the time it's not important. The network is a big black box where their data goes (and, if you ask them, the network is the reason their data is delivered so slowly. But they're wrong. IT'S NEVER THE NETWORK!!)

However…

There is a difference between not understanding and not caring to understand. One is due to a lack of opportunity but not curiosity. The other is a willing abdication of responsibility to know.

And I think the second is extremely unhealthy.

As I.T. Professionals, we need to be committed to lifelong (or at least career-long) learning and growth. No area of I.T. is too esoteric to want to know about. We may not have time right now; we may not be able to utilize the knowledge immediately.

But rest assured that understanding how and why something works the way it does is always better than the alternative.

Accept

Larry Wall (creator of the Perl programming language) famously said,

> **Most of you are familiar with the virtues of a programmer. There are three, of course: laziness, impatience, and hubris."**

In one brilliantly succinct phrase, Mr. Wall took three traits commonly understood to be character flaws and re-framed them as virtues.

As I sat and thought about how acceptance is generally seen as a positive trait in life, I realized that in I.T. it could be just the opposite.

Accepting the status quo, that the system "is what it is", that things are (or aren't) changing (or staying the same) and there is nothing that we can do to affect that… all of these are anti-patterns which do us no good.

As I sat and pondered it in the wee hours of the morning, I heard the voice of Master Yoda whisper in my ear:

- Not accepting leads to curiosity
- Curiosity leads to hacking
- Hacking leads to discovery
- Discovery leads to innovation
- Innovation leads to growth

When we refuse to accept, we grow.

Bringing this back around to personal growth, I think there is a time and place when refusing to accept – perceived limitations (whether that perception is our own or those around us), our place (whether that's in the org chart, or in society at large), our past failures (or things we believe to

be failures), etc. When we refuse to permit those external forces to define or limit us – that is when we find the path toward personal growth.

Hear

It's an understatement to say that working in the world of Information Technology means that sometimes – perhaps many times – you need to fight to make yourself heard. This is a normal experience, part of the give and take of ideas and teams of people wrestle with technical issues until a solution is found.

So what happens when you aren't heard? Not as in "you were speaking softly or the other person wasn't paying attention". What happens when you speak your mind clearly and concisely, and nobody responds?

And then, what happens when that same thought is stated by another person – perhaps just minutes afterwards, or maybe much later in an email or a different context? And when that person says it – your thought that you expressed earlier – all of a sudden people respond! They discuss, debate, agree, or whatever.

What would it be like to live as an I.T. persona non grata?

You need to understand that this isn't a hypothetical situation for some people. Women & minorities, both inside I.T. and out, experience this on a regular basis.

If you haven't noticed this, your first reaction might be a variation of #NotAllMen, #NotAllCompanies, or #NotHere. But if you haven't noticed it, the statistics are against it being due to it not happening.

To hear someone, sometimes you must first ask a hard question and be ready for an answer which makes you uncomfortable.

To hear someone, first you must stop talking.

Know

Over on my other blog (EdibleTorah.com), I suggest that "I don't know" is a powerful phrase with regard to personal growth.

I submit that it is equally powerful in the world of I.T.

We I.T. Pro's make a big deal about knowing. Knowing all the command-line switches for all the commands on a Cisco router; knowing all the differences in all the versions of Windows from Windows 286 onward; knowing that more than 3 layers of containers will corrupt a CORBA database; knowing every episode of Star Trek TOS.

Whether the knowledge is rediculously ephemeral or soberly relevant, Geeks take pride in knowing, both in quantity and in quality.

But some of the most brilliant, gifted, and successful I.T. Pro's I have known over my 27 years in the business have been the ones who willingly, even eagerly, admit "I don't know".

Which is usually followed by "…but I will find out and get back to you."

When we are afraid to admit we don't know, when we avoid those subjects in which our mastery is incomplete or when we obfuscate to make it appear we have the situation under control… when we do those things we miss out on an amazing opportunity to reconnect with the humility and curiosity that got us into I.T. in the first place.

See

Unlike “know” (which I wrote about earlier), “see” is powerful in the I.T. context all on it’s own.

“Let’s see” expresses being open to the truth that our data shows.

“We’ll see” is a way of showing both patience as well as a healthy scepticism that ensures our own biases aren’t affecting our understanding of a particular result.

There’s an old saying that when someone says “but”, they are going to contradict whatever they just said. “That’s a great idea, but (it’s really not)”; “I would love to help you, but (I really don’t want to)”, etc.

In the same way that “But” is a linguistic flag to the listener for contradiction, “See” can be the word that indicates a willingness to explore, to innovate, to test, to change.

Listen for it, and act accordingly.

Count

The "Fizz-Buzz test" is an interview question designed to identify programmers who don't actually know how to program. The text of the programming assignment is as follows:

"Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz"."

At first blush, this is a simple exercise, intended to help highlight how someone approaches a task.

But it can be seen as something deeper than that. By exposing the thought process of creating this simple script, we gain insight into how we view the world. How we organize, sort, and group. How we hear, confirm, and deliver requirements. How we interact with others. How we deal with requests by people of importance (the person who can give me a job) which is clearly beneath my skills (this is something any freshman computer science student should be able to do).

It turns out how we answer this simple counting exercise is as much about who we are as it is about what we are able to do.

Trust

Part of the my job as Head Geek is writing. A lot of writing, actually. When I first started, I expected the usual concerns of new writers: the challenge of writing a specific story from a specific point of view under a deadline. The challenge of having your work edited and critiqued by other people who may not share or appreciate your style. Dealing with writer's block. Dealing with competing requests.

But I wasn't ready for trust issues.

I don't mean others not trusting me to deliver, or to meet their expectations. I mean my own trust issues.

- Will the reader want to come with me on the story I'm building?
- Will they hold it against me if this is the 3rd or 4th time I've written on the same subject?
- Will they be annoyed if I say something they've heard before?
- Will my editor catch my inconsistencies?

…and a million other moments of doubt which are really self-doubt.

To do this job, I find I need to re-invest in trust.

I do not need to write the definitive essay on "xyz". I simply have to write one essay on "xyz". I need to trust that my audience (and editors) will give me permission to revise my opinion, to grow in my expertise, to expand on the original idea.

I have to trust that the reader is willing (happy, in fact) to come with me on this journey.

Forgive

Suffice to say that if you have been involved in the world of I.T. for more than 15 minutes, you have also likely gotten into a disagreement. It could have been about Star Wars vs Star Trek, or which comic book character could beat another comic book character's butt, or whether the board game Settlers of Catan is better than Flight of the Boodles.

Or it could be a disagreement about how to migrate to a new software platform, or how to configure the network, or how (or if) to implement security protocols.

The challenge is that these disagreements can become (or at least feel) extremely personal – as if you are being criticised, rather than the idea you are presenting.

And when the shouting is over and decision is made, you need to find a way to move on. Except frequently people don't. Working relationships are strained or ruined. Professionalism is tested. People quit.

In thinking about today's prompt I was struck again about the difference between disagreements in I.T. and those in "my other life" as an orthodox Jew. In the traditional setting for Jewish learning – the Yeshivah – there is rarely a moment of quiet. Two, three, or more people gather around a book to analyze a section of text and begin to debate its meaning, its relevance, its relationship to other text. The debates are loud, forceful, impassioned. Each side refuses to back down or give an inch until they have either completely proven their point or had it refuted beyond a doubt.

And then there are smiles, pats on the back, shaking hands, “Yasher koach” (equivalent to “Nice job!”) spoken.

What’s the difference?

In the Yeshivah world, the argument is for the sake of capital-T “Truth”. Not for ego, personal advancement, or reputation. If the group feels they have made progress toward a better understanding, then the shouting was worthwhile.

Thinking about this dichotomy led me to recall a time when I.T. and Yeshivah overlapped.

I was working overseas, and was teamed up with a brilliant monitoring designer named François Cellerier. To say that we didn’t get along would have been an understatement. Every single design choice I made was questioned. I tried being diplomatic. I gave a little to his questioning, validated his points (even when I thought they were dumb), tried to find compromise. And the more I did that, the more frustrated he became.

After about a week, it became intolerable. I pulled him into a empty conference room and asked him what the issue was. Why was he challenging every single thing I said?

“Because that’s my job. And I don’t understand why you aren’t showing me the same respect. Do you think I’m that stupid that it’s not worth it?”

I was flabbergasted. After a while François, who had more experience with Americans than I had with his culture, was able to put it into terms I could understand.

“You and I are the two people in the entire company who understand any of

this. But everyone is going to question our decisions later on. Our job is to make sure every choice is solid. If I can defend my design against your best argument, then nobody else's question has a chance. But if you don't question me, I will never know whether someone is going to blow it up later on. We have to push each other now while we have the chance to revise things."

François was fighting for the capital-T "Truth". And in that moment, I understood. And in that moment, every perceived slight, every question of my design (and authority, and intelligence, and permission to exist) was forgiven.

The vast majority of the time, people argue with us for reasons that have nothing to do with us. If we can't see it, forgiveness is impossible. Once we understand it, forgiveness becomes a non-issue.

Remember

Earlier, when I wrote about know-ing , I related a story about how the remembering part wasn't important, it was knowing where to find the information, and knowing what to do with it once you had it.

I'm going to contradict myself now.

First, it's fairly obvious that if you can't remember where to find the right information, you don't really know it. But that's just a nit-picky point on the semantics of "know" versus "remember" in a particular context.

What I want to suggest is that – if you look at it from another perspective – all of I.T. is the creation of systems that aid in the act of remembering.

One of the most elegant examples of this is a common first-year programming exercise: creating a script that uses a single variable to produce a Fibonacci sequence.

A Fibonacci sequence is a set of numbers where the NEXT number is the sum of the previous two. So it starts out 1 1 2 3 5 8 13 21 34 55 and so on.

The exercise to create a script which calculates as many iterations as the user desires using a single variable is challenging because somehow the script must remember:

- The number of iterations requested
- the current number
- the previous number
- …so that the NEXT number can be calculated.

One such example looks like this:

```
sub fibRec {
my $n = shift;
$n < 2 ? $n : fibRec($n - 1) + fibRec($n – 2);
}
```

What makes this script move beyond being merely functional and into that magical realm which programmers call “elegant” is the way it uses recursion (calling itself) and the shift command to maintain it’s coherence.

It is also a great lesson for us in life: With structure and a bit of creativity, we can eliminate all the dross, all the superfluous bits of our life because what we need to remember is tied up with what we are doing and what we need to do.

Learn

A few months back, SearchNetworking editor, Chuck Moozakis interviewed me for an article discussing the future of network engineers in the I.T. landscape. As part of our discussion, he asked me, "what in your mind, defined you as a networking pro in 1995, in 2005, and in 2015?"

The question caught me off guard. What defined me? What kind of question even was that?

After we wrangled it over for a few minutes, I offered to take the question offline and get back to him. After several days, I finally had my answer – a laundry list of skills and knowledge that I felt lent credibility to my claim of being a network professional in each of those time periods.

But the question stuck with me, and so I posted the background along a question on the SolarWinds "GeekSpeak" forum.

The answers were thrilling and inspirational. I encourage you to read them.

What they all include is a list of the things people learned along the way – sometimes technical skills, sometimes "soft" skills, and sometimes just things they learned about themselves.

What I learned, both from answering the question and seeing others' answers in response, is that the willingness to learn is the only skill people really need.

Learning is the journey. The rest is just work.

Change

If you were reading this series in sequence, as it was posted on AdatoSystems.com, then today would be Sunday. As it is, you'll have to bear with me a little, because this post is all about Saturday. You could probably get by if you just thought back to the last Saturday you can remember. Hopefully it was less than 8 days ago.

So, what did you do on Saturday?

If you were like most I.T. professionals, you probably spent some of the time on work stuff. You checked email a few times, maybe started working on a design, did some research.

Or you worked on your own projects. Even thought you could categorize it as "fun", it still put you in contact with technology in a way that was extremely similar to what you do at work.

It's took me almost 20 years, but I finally realized that rarely does this help me get ahead. Mostly it makes me feel crushed, rushed, and stretched to the limit.

Not too long ago, for reasons outside the scope of this blog, I started disconnecting for one day every week. Totally unplugging. If it had an on-off switch, I didn't touch it. That went for lights, cars, phones, game consoles, and computers. And not just me, but my whole family. From Friday night until Saturday night, we went off the grid.

Unsurprisingly, doing this did not drive us to the brink of madness.

We invested in some board games. We frequented the library and grabbed a ton of books. We read out loud to each other. We learned to juggle. We had

long meals around the table with lots of conversation. We went for walks. We visited friends. We took mid-day naps. We discovered parks and trails and parts of our neighborhood we didn't realize existed. We thought big thoughts and dreamed big dreams.

So I'm going to challenge you: Look ahead to next Saturday. Perhaps you can't just go off the grid for an entire day. Maybe you can't do it every week.

But maybe you can.

You'll never know unless you try, unless you attempt to alter the status quo.

The point is, change is something we usually must do in reaction to something around us. But we also can choose to change.

And maybe the act of choosing is the first part of the change that gives us back a piece of ourselves we didn't even realize was missing.

Pray

They say there are no atheists in foxholes. I'd argue there aren't any atheists in the datacenter during a weekend upgrade either.

Prayer is a powerful force – both for individuals and for communities (and believe me, an I.T. department is a community). But among the less religiously inclined Technorati, the place of - and for - prayer is often misunderstood.

Popular culture likes to portray "pray-ers" as people who throw up their hands and "give it to God". "Sheeple" who are unable or unwilling to take ownership or responsibility for their choices and lives. And I'm sure there are some people who move through life with that mindset.

A story from Torah brings into sharp focus when, where, and how prayer can help:

Jacob was about to meet his estranged brother Esau for the first time in decades. This is the same brother from whom he had stolen the birthright – the inheritance due a first-born son – as well as the blessing from his father, also meant for the first-born. There was a lot of history going on here, and word came that Esau was on his way with 400 armed fighting men to meet Jacob and his 4 wives and 13 children.

Jacob first divided his family into 3 groups.

Then he assigned gifts to each of the groups and told them to approach Esau with a little time between groups and give the gifts, along with the message that Jacob was coming along shortly.

Then he prayed to God for help.

The plan worked – Esau met different members of Jacob's family and received lavish gifts from each one. And then when Jacob approached, Esau met him with an embrace.

To summarize, Jacob planned, then he prepared, and finally – when all the rest was done, he prayed.

This resonates for me as both an I.T. professional and a "pray-er". We make our best plans based on the information at our disposal. We prepare – both for the expected outcome and with mitigation strategies for predicted negative events.

But whether we understand the phrase "Der mentsh trakht und Got lakht" or not, we understand it's gist.

They say "the best laid plans of mice and men..."

And in the face of that stark reality, the only thing we can do is pray for the best.

Awaken

After two and a half decades working in I.T., I can tell you there is a definite cultural bias in my business against sleep.

Whether we're talking about physical sleep or metaphorical sleep, being in I.T. often means being awake longer, or at times that put us out of sync from our non-I.T. peers, family, and friends.

But it's not enough to simply be awake (meaning "to not be sleeping"). Working a 24 hour cut-over, or being on an emergency call at 2 a.m. because a major system is down, or realizing that security has been compromised: all of those things require being more than merely "not-asleep". It requires being sharp, aware, actively involved in the situation.

In I.T., awakening is more than just opening up the store because it's time to make the donuts.

Which is why it's all the more important that the work we do excites us, engages our intellect, inspires us to work our hardest.

If the work we do doesn't awaken our passion, then it's likely we'll be asleep at the wheel when there is real work to be done.

Ask

In the world of IT, asking for things is an activity that is somewhat fraught. First, there is the risk of rejection (which I wrote about on my other blog). Second it often opens up a can of worms that, as I.T. pros, we really don't ever want to deal with – the business side of things.

To ask for something – from a new server to permission to work from home to a network refresh to a day off – means, in many companies, means justifying the request from a business perspective. It means spending (wasting) valuable time hacking away at an RFP, doing hours of research, and generally guessing at the minds of management to figure out which magical datapoint will make our case and win us the request.

Many of us sidestep the entire issue by somewhat passive-aggressively waiting until the thing we want becomes and unavoidable issue. We wait until the current server is hopelessly old, we call in sick but say we can work remote to keep from infecting others, we allow the network to reach capacity, we hold onto our vacation days until HR is pushing our manager because of "use it or lose it" policies.

Such tactics are not good for the business nor are they good for us, both in the short and long terms.

Recently I spoke with SolarWinds CTO/CIO Joel Dolisy about this phenomenon as part of a presentation we were giving (you can watch it here). I came out of those working sessions with a much clearer understanding of both where I.T. pros tend to go wrong with their requests, and how those mis-steps create a negative feedback loop – a poorly framed

request leads to rejection which leads to negative feelings (about management, the request process, and/or business in general) which leads to avoiding making requests in the future.

The answer, believe it or not, is hinted at in Rabbi Davidovich's blog on this same topic (asking). He writes:

> **"When a relationship is healthy, and everyone is "cool" with one another, you don't have parents snittily insisting that you say "please" and in the proper tone. [You] Just ask. […] Lighten up. Get to a place where asking is enough. And then asking will be enough. "**

What came out of my talk with Joel was the understanding that if the relationship between the I.T. pro (as an individual, or as part of a community within the organization) and management (again, as an individual or as a whole) is generally strained, then when management says "no", the engineer reads into it things like "they think my idea stinks" or "it's all politics".

When the relationship is good (even as I recognize that for many I.T. pros it is not) then "no" is simply one word in an ongoing conversation about how to do what is best for the business.

In those situations, it is nothing for the I.T. pro to just ask.

Judge

"Don't judge me" is a pretty common theme these days, a catch-all phrase meant to say "I know many won't approve, but I gotta be me!"

While often found on social media, where people post pictures of their fried-oreo-wrapped-in-chocolate-waffle-with-a-donut-on-top-breakfast ("This is a healthy way to start the day! Don't judge me!"), the phrase can be found in the I.T. world as well (I'll leave the pictures to your imagination)

- Picture of my server rack. I'll organize the cables some day. Don't judge me.
- 100 lines of code instead of a for loop. Don't judge me.
- 100 meter cable to connect the two sides of my data center. Don't judge me.
- Cleaned off my desk, and it still looks like this. Don't judge me.

However, the phrase "don't judge me" betrays an inherent bias on the part of the speaker. It implies that others will not only judge, but judge negatively. But that isn't the only way to "judge".

Earlier in this anthology, I wrote about "Ask", and how in I.T. asking can lead to frustration because of a communication breakdown between the ask-er (us) and the ask-ee (management).

But I pointed out that, when the relationship is good, everyone is working toward a positive overall outcome. Sometimes that means the answer is "yes", sometimes "no", sometimes "not now", etc.

The same is true for our bias toward judgement. Imagine a work

environment where the relationship are all good:

- When they see the rats nest of cables on the rack, the response is "wow, I can only imagine the fires you were putting out that led to you not having time to do this the way you would want. Let me see how I can help you so you have time to revisit this.
- Or when they see that 100-line script, they say "Would you like me to set up a couple of hours with Joe over in development, and you two can use this script as a starting point to improve your skills?"
- Or when they see the 100-meter cable, they say "You're right. We've put off the redesign of the floor plan too long. Can you share your experience so we get it right the first time?"

Of course, judging favorably works both ways. This is a perfect time to start checking our judgements as they come up, and making sure we're giving those around us the same benefit we would like from them.

Dare

I've never been at a company that openly said "We're risk averse here. We would prefer you don't try anything new. No innovation unless you absolutely can't avoid it." No, most of the time the message is "We're a company of risk takers! Dream big, dare to try new things!".

But in the I.T. world, being "daring" is a double-edged sword.

On the one hand, taking a risk can result in a huge payoff – savings to the company, improvements to services, or a true competitive edge in business.

But it can also be the ultimate career-limiting (or ending) choice. I have seen more than one person escorted to the door because their "daring behavior" violated security, change control, or some other corporate policy which was considered inviolable.

But even with that risk, I.T. professionals are a daring bunch. Most of us got into this type of work because we had a vision of how things could be better.

So I wanted to share a story that I heard recently during my discussion with SolarWinds CTO Joel Dolisy on ways that I.T. professionals can do a better job of "selling" their request for new software, hardware, and projects (you can see my discussion here).

Near the end of our talk, Joel and I were discussing ways we can respond to having our request turned down. Joel proposed going out and doing a small prototype anyway. In fact, asking to be allowed to do a proof of concept, as a response to being told "no".

Pretty daring stuff.

But Joel backed up this advice with a real-world story: about a particular engineer at Amazon. It seems this guy had an idea for recommending products based on what was in the user's shopping cart at that moment. Leadership heard the proposal but was concerned that having such recommendations would distract a buyer for completing their purchase, and so they refused.

The engineer would not be deterred, and implemented the code on just a couple of Amazon's massive farm of servers. Then he collected the data on how buying behavior changed. Faced with the overwhelming evidence that "Recommended Purchases" drove sales higher, leadership revised their thinking and agreed to develop the code fully and implement it system-wide.

Nothing ventured, nothing gained.

Love

In IT, there are jobs, there are careers, and there are passions.

Back in 2006 there was a guy in my monitoring team, a 7 year veteran of I.T. who had made his career being one of those "server guys" who knows every variation of component in every model of the "big 3" server vendors. Looking at the chassis, he knew which video card or disk controller needed to be ordered when something was on the fritz.

It made him a reasonably competent monitoring guy. He understood the failure patterns, and was able to help us instrument for them. He wasn't particularly passionate about monitoring, but his career to that point had given him some valuable and unique insight.

At the time, VMWare was getting big. ESX (and ESXi) was just hitting the streets, and this guy's eyes lit up. Since he already got along with the server admins and data center ops guys, getting the scoop on this new tech was not hard. He stayed late, helped with testing, read up on the topic.

VMWware hired him a couple of years later. He's doing pretty well for himself, in fact.

Job. Career. Passion.

There are people who never find this. There are those who know where their passion lies, but never get the chance to reconcile it with their job or career. It seems too hard to get from wherever "here" is to "there".

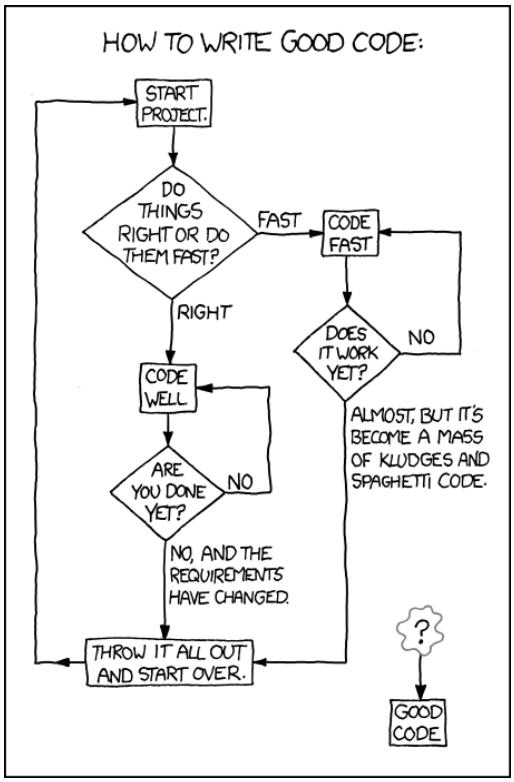
But as someone who has been lucky enough to get to that promised land, I can tell you that it's worth the effort.

Think deeply about your passion – meditate on it, pray on it, ponder it over an IPA or a glass of 21-year-old scotch or a fine craft root beer – and then pursue it.

Trust me when I tell you that the effort is worth it. The journey is worth it. The struggle and the chase and the ups and downs are worth it.

Love, even in the context of work, is worth it.

End



If you have ever written even a single line of code, you have also probably done "it" – the program without end. The infinite loop. Hopefully it was just to print out "Hello World" to the screen, or all the numbers to infinity, or calculate pi.

BUT… probably not. It was probably to delete a series of files (that went tragically too far). Or to reboot a set of servers ("and then the entire data center went dark"), or to log something to a file (until the hard disk filled up), or some other time-saving automation gone horribly wrong.

That's when you learned: Everything needs an escape, a way out, a check for a conditional to know you've a limit was reached.

Same with our lives.

Too much overtime (or too much work in general). Too much to drink, or eat. To many hours spent online in a fantasy world. Too much time spent crafting the "perfect" subroutine, or network design, or email response. Too much time spent trying to create the perfect anything, really.

There's a point where our bodies are no longer meant to carry washing machines up 3 flights of stairs. A point where peekaboo is no longer the funniest game in a child's life. A point where we should hand off the mantle of leadership – of our company, of our family, of the backyard grill – to someone else because, like us at some point in the past, they have prepared for this day and patiently waited their turn.

Not only do we need to realize that all things will end, we need to recognize that all things must have an end. And that the things we start or engage in regularly need to have an end point built-in – by us if it's not part of the natural order.

As former Chief Rabbi Jonathan Saks wrote:

> **For each of us, even for the greatest, there is a Jordan we will not cross, a promised land we will not enter, a destination we will not reach. That is what Rabbi Tarfon meant when he said: It is not for you to complete the task, but neither are you free to desist from it. What we began, others will continue. What matters is that we undertook the journey. We did not stand still.**

As I.T. professionals, we have the chance to see and experience what happens when actions loop without end. It is never pretty.

Begin

In my essay "I Wish Someone Had Told Me" I talk about getting started in I.T., and how at the beginning, you are just all over the place. You are pulled into different projects, working with different teams.

And that's the problem. Because if you keep letting yourself get pulled around, you will never settle into one area, and you will never get really good at something.

What I didn't say in the essay is that beginning – meaning starting to focus on the area that excites you most – means not focusing on other things. And that can be hard. It can be a challenge when you realize you no longer know every variation of every component in 3 vendor's line of servers; or that you no longer think in code; or that all the old keyboard tricks you knew were for operating systems that are now defunct.

But that's the price, and it's one worth paying.

Hope

“Star Wars: A New Hope” (which at the time was simply called “Star Wars”) came out when I was 10 years, 1 month, and 21 days old. Like many 10-year-old boys at that time, the movie consumed my attention like nothing else had (but like so many new geeky obsessions have since).

My room was plastered with posters, magazine clippings, and factoids. My daydreams were littered with fantasies of inventing a real light saber, or constructing R2D2, or joining the Rebel Alliance. My free time was filled with the practice of any aspiring young Jedi – duelling and jumping and consuming vast quantities of science fiction.

It was not without cost.

My parents took exception to the damage on the bedroom walls. My classmates and teachers were quick to comment (and worse) about my spacier-than-normal demeanor. My arms and legs (and in one case the side of my head) bore silent black-and-blue testimony that young boys are poor swordsmen and broomsticks a poor choice of practice weapons.

But it was worth it. I discovered not just a genre, not just a pastime. I discovered my place in the world.

While it would take years before I could fully reconcile the passion I felt with the ostracism it often generated (I could have used a good old Wil Wheaton pep talk back then), when everything finally clicked it was with a sense of “right”-ness that left me no doubt.

While it’s true that in subsequent years the Star Wars franchise has often

fallen short of my personal expectations, it doesn't diminish the raw joy and excitement and, yes, hope the original gave me. It's an association that hasn't faded with time.

A few months ago, I sat my boys (aged 14 and 11) down and put the first disk in the player. I watched as the initial blast of John Williams' iconic score blew their hair back . And I basked in joy only a geeky parent can feel as a familiar feeling spread across their faces like the double sunrise racing across the surface of Tattooine.

Hope.

Intend

There is a popular saying about the road to hell. It may equally apply to the road that leads to a system outage.

- If we intended to back up our servers, but never requested the SAN storage…
- If we intended to patch the application, but never cleared time for the change control
- If we intended to upgrade the network, but never included it in the budget
- If we intended to get that certification, but never scheduled the class

It's fair to say that when something goes wrong in I.T., it is rarely because that was the intent of the people responsible.

So why do we let things like that happen? After more than a half-century of modern I.T. operations, you would think most of this would be utterly and unquestionably standardized.

Part of this is that things appear to change (something I've written about elsewhere) and therefore we fool ourselves that the methods that provide stability somehow don't apply any more.

But part of it is that we intend to do it, but allow other things to get in our way. From the manic ("we have to inventory and integrate 3 new acquisitions this quarter") to the mundane ("someone said something factually incorrect on the internet!") there are plenty of compelling reasons to postpone that change control request or certification exam.

At least, until it's setting off alarms. Then we realize our intentions were just more fuel for the fire.

Create

Back in August, 2015, SolarWinds held their 4th ThwackCamp – a completely free, completely online conference. In the keynote, Head Geek Patrick Hubbard spoke to an interesting dichotomy in the life of an I.T. Professional.

> **“(In IT) we’re always fixing things. […] but fixing just eventually wears us down. So stop fixing things, and start to solve things.”**

More than just semantics, Patrick highlighted both a mindset and an action plan. Fixing things is merely the act of making the problem go away. Maybe permanently, but that’s not part of the job. It’s just to make the pain/alarm/complaining stop.

Solving things goes deeper. It speaks to the root of the problem and to a willingness to address both the symptoms and the core cause of the issue. Solving means changing architectural or organizational issues. Solving means building a better system based on everything we’re learned from the old one. Solving means playing a long game.

If you are sitting in the hot sun, fixing the problem means dumping some water over your head. Solving the problem means planting a tree for shade.

Over the course of this anthology, many of my essays have spoken to action. What do we intend, plan, hope, or pray to do. What do we believe? How will we start (or end)?

Today I want to suggest that DO-ing is the same as FIX-ing. It’s a good start, but it’s not the whole story.

Beyond asking ourselves what we're going to do, we should think deeply and seriously about what we are going to create.

The question is not about having the ability, or intention, or permission to create. You already have those. The question is what you are creating right now, what you will create tomorrow.

So as we look ahead to the coming year, don't settle for just fixing things. Look at ways you can start solving them. And don't just get caught up in what you are going to do.

Decide what you are going to CREATE.

Bless

As I've been writing each of these essays, I've come back repeatedly to the idea that – if you choose a career in I.T. – it is important to find a niche (both in terms of the company where you work, the company you keep, and the work you do) that you love.

I realize this is far easier said than done. But just because it's difficult doesn't mean the effort is not worth it.

That would have been hard for me to prove until recently. As I write this, I've been blessed to have found a place as Head Geek at SolarWinds. It's a job that engages so many facets of my experience – from my theatrical training in college to my love of writing to my years as an I.T. generalist, and of course to my passion for monitoring. But it's also a job that allows for the facets of my life which are not work related but equally important to me – my religious convictions, my family, and even the city where I've chosen to put down roots.

So I'm not going to drag this out. I have found my blessing in this work that I'm doing.

I hope – maybe as a result of reading these words and feeling inspired – you do, too.

Give

If you work in IT, there are a few things which I know are true about your job even if I've never met you:

- You are busy. You have enough work for you, your clone, and your clone's cousin. Same goes for your co-workers (and their clones and cousins).
- You know things. Maybe a lot of things. If you have worked in I.T. for more than a month, you have a few tricks up your sleeve that other people who "know computers" will never have heard of.
- You hold the things which are known – by you, your co-workers, their clones, and cousins – in very high regard. The things you know are what make you valuable as an I.T. professional.

Knowing all of that, I understand how what I'm about to say may shock you:

Give. It. Away. All of it.

Despite being busy. Despite the fact that the person you are going to give away all your hard-won knowledge may not have "put in their time" or "earned their stripes". Despite the fact that if two of you know the tricks, there's a chance that you may not be as valuable. Despite all of that.

Because I know something else: You are wrong.

You are not too busy to give your knowledge – to write or podcast or vlog or just sit someone down and tell them (ewwww! How analog!). You are not too busy for that.

Because in sharing what you have, you will gain. In spreading what you know, you will never lose. In giving away for free the things which cost you in the precious coin of time and sweat and tears you increase rather than

decrease the value of it.

In fact, giving may be the greatest gift you ever receive.

Return

RETURN is one of the most basic of all constructs in IT. Whether you are a programmer, sysadmin, network engineer, Virtualization architect, or something else, there is an almost 100% likelihood that you have needed to find out the RETURN code from one of your systems at some time in the past.

For that reason, I love that this is the last prompt for this series. It's a way of saying "We're all done here. Run garbage collection, write to the log files, and close this puppy down."

```
RETURN 0
```

Credits

I cannot say it better than Fred Rogers did in 1997, as he accepted a lifetime achievement award:

“ All of us have special ones who have loved us into being...”

Listing all the people who have loved me (and taught me, and in some cases dragged me against my will) to this point in my life would be impossible. Specific to this project, however, there are a few people who deserve mention:

- The SolarWinds Head Geeks: Patrick Hubbard, Thomas LaRock, and Kong Yang (who each deserve the honorific “The inimitable” in front of their name). You set the bar very VERY high in all aspects of I.T. life. I'm humbled and honored to be part of this exclusive club.
- Kimberly Deal, for the “encouragement” (some might call it “abject whining”) to publish this ebook. Thanks for the kick in the pants!
- The SolarWinds editors – Damon Garcia, Aaron Searle, and Dave Jensen for continuing to show me what good writing should look like
- Everyone I've worked with, ever.

Credit is also due http://theWebalyst.com for the template that helped create this ebook.