

Help Us, Help You:

Effectively Troubleshooting OS X

Michael Lynn

MacTech 2013

When I was initially asked if I'd like to speak at this conference, I was very flattered.

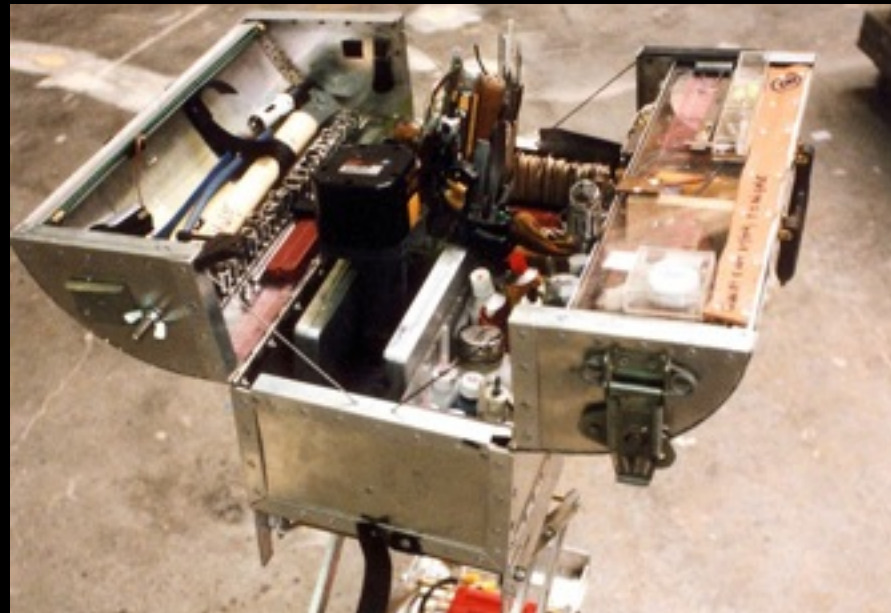
I have great respect for MacTech and how they're working to improve our community and get that information out there in a timely fashion that can help us with our jobs.

But I was immediately faced with a challenge - they asked me what I'd like to speak about.

For the people that know me in here, they might guess that my immediate first response would be something involving the python language - maybe some pyObjC.

However, I'm pleased to say that enough other people - Greg Neagle in particular later this evening - are already doing a very nice job of encouraging others to learn the python language and learn about the power it can have to help you manage OS X. As a side note, I look forward to attending that workshop this evening and encourage all of you to do the same.

So instead I'm talking here today about another subject that's very near and dear to my heart - effectively troubleshooting problems in OS X.



Build Up Your Toolbox

(It's important!)

My introduction to the majority of the people I know here came from participating in the Freenode IRC channel `##osx-server`, where I help to administrate the channel and generally just live for the problems that people bring to the channel and trying to solve those problems. Somewhere inside, I'm wired to get a kick out of solving problems - it's a feeling that I think a lot of you share and understand.

But what I don't think is understood quite as well is what I see as the real reason you should have this skill set under your belt if you don't already.

So today I'm going to cover some of the core diagnostic tools that I would hope we all know and maybe throw some extras on top that you might not know - but first I want to take a moment here to explain why I think this is so very important, now more than ever.

In case you hadn't noticed lately, we've got a lot of newcomers to the Apple support community. Apple's success with iPhones and iPads in enterprise and government has resulted in some businesses taking a second look at using Macs for their desktops. Unfortunately, for many of the IT shops at these businesses, up until now they've never really dealt with an Apple product. So one week it might be "Let's deploy 100 iPhones, it shouldn't take too long, right?" only next month to have that same manager come back and ask "How long would it take to get everyone set up with a MacBook Air?"

This has lead to a pretty phenomenal growth rate in our community as of late, it seems. But at times, I've noticed it feels like it's turning our forums and mailing lists into a huge echo chamber



The Echo Chamber

- with people asking the same things and the same types of questions, over and over.

“How can I control that feature on my devices?”

“Why am I getting this error message?”

“Where is that setting stored?”

And at the root of it all - I feel that it's not even the newcomers that are the biggest contributing factor here. It's something a little deeper, something that Apple's been doing within the past few years that's really, truly, at the heart of this trend.

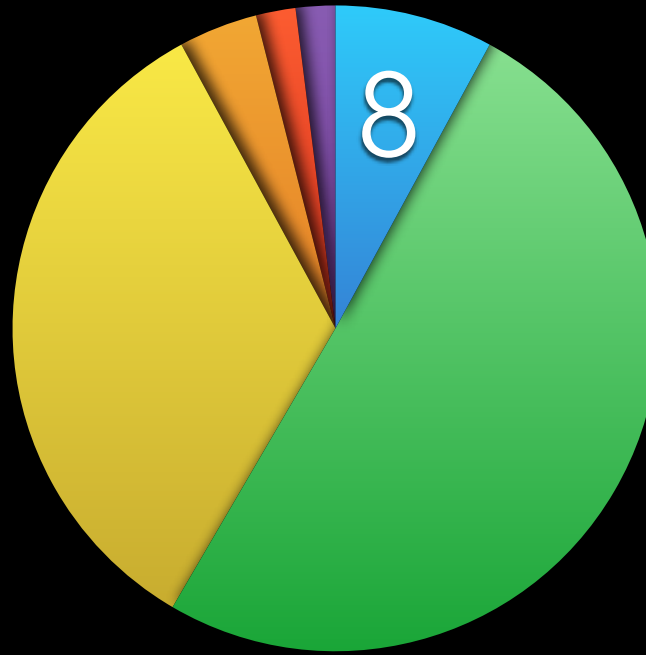


"Interesting Times"

Professional Apple device support is an *immensely* interesting field to be in right now - more interesting than it's ever been in the past. The support environment our customers are coming to expect from us these days is ending up unlike anything else out there in the personal computing market.

To get an idea how unusual our support environment is, the only way I can do that is to compare it against the environment of what's arguably still the dominant desktop operating system in enterprise today: Microsoft's Windows.

Windows 8 has been out for a little over a year now - an almost equal amount of time that Mountain Lion was available, actually.



1 Year: Windows 8

Source: NetMarketShare 10.2013

The current estimated adoption rate for Windows 8 right now is hovering around 8% or so of Windows desktops (and that's me being pretty generous). Less than 1/10th of Windows desktops out there right now are running that version of the operating system from Microsoft more than a *year* after its release.

And just now at the end of last year, Windows 7 *finally* became Microsoft's !!most used!! desktop operating system. That's more than 4 years after it was released - before they were able to reach that landmark occasion.



4 Years: Windows 7

Source: NetMarketShare 10.2013

Right now Windows 7 itself is hovering somewhere around 50 percent representation for all of Microsoft's desktop computers.

What's in second place you might ask? Well, no you probably wouldn't, you know the answer: Windows XP. Still.



12 Years: Windows XP

Source: NetMarketShare 10.2013

Released 12 years ago and almost a third of the Windows PCs out there are running it right now. That's over a decade that some Windows IT shops have had to perfect their support game and troubleshooting toolset for an operating system. Mind you, the number is dropping, slowly, but it is dropping - but the thing that's driving it isn't because of real desire for a new operating system from Microsoft's customer base, it's because Microsoft is finally pulling the plug on extended security releases for Windows XP sometime next year.



OS X 10.8: Mountain Lion

Now back to Apple and our “interesting” environment.

Right up until October 22nd, for the single year that OS X Mountain Lion was out, about 50% of Apple desktops were running it by the end of that year. 50% !



OS X 10.8: Mountain Lion

50% in 1 Year !

Lion, out for 2 years, and Snow Leopard, out for 4, pretty much evenly split most of the remaining Macs out there. Current statistics suggest that any internet connected Mac system older than Snow Leopard currently makes up less than 10% of all Apple desktop machines online. Anything older than a 4 year old operating makes up less than 1 in 10 Macs.

In fact, it only took a month after the release of Mountain Lion for Apple to reach a 10% adoption rate.

That means that in those 30 days, Apple was able to pull something off with their 10.8 operating system that Microsoft, over 365 days after the release of their 8.0 operating system - they STILL hasn't been able to achieve numbers like that.

Now, if you've been paying attention I mentioned October 22nd just a moment ago because, in case you haven't been following the news, that was the day Apple released their newest operating system - 10.9 - Mavericks.



OS X 10.9: Mavericks

Remember Mountain Lion reaching 10% in 30 days? According to GoSquared, Mavericks hit that same 10% in 2 days.



OS X 10.9: Mavericks

10% in 48 Hours !

48 hours.

How about other Apple products.



iOS 7

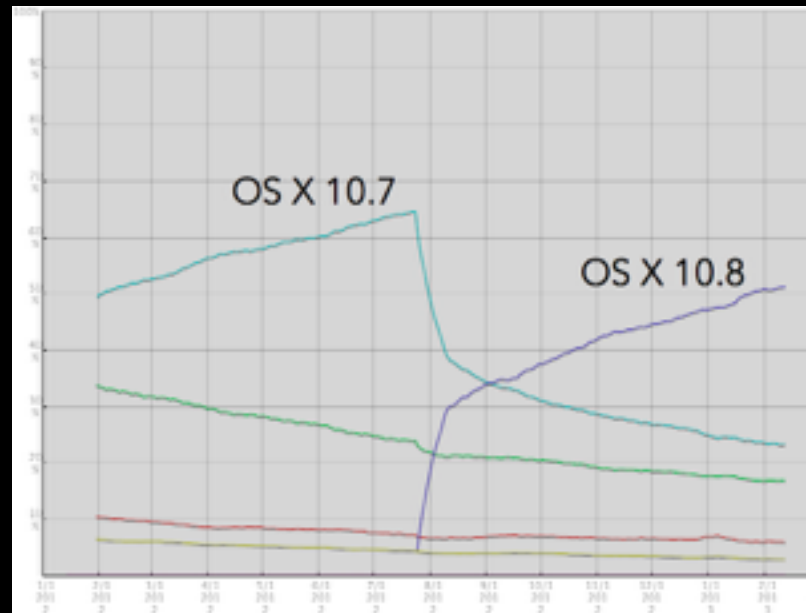
iOS 7?



iOS 7

30% in 48 Hours !

30% adoption rate in 48 hours. Over 50% by the end of the first week. Today it's somewhere just north of 70% and that's at a month and a half.



The OS X Release Curve

Source: OmniGroup

A skyrocketing adoption curve while the old OS plummets off into the background - curves like this right here are our current support reality. This is why it's so interesting and so unlike any other support environment out there.

This is the real reason, I feel, for the seeming questions and sometimes confusion that's out there in our community.

Tricks and tips that someone else figured out, that worked with one OS version last year, or the year before, may no longer be valid when the next new OS comes out - or there may be new and better ways to do it.

We now have 1 year before our customers - and this is unlike any other IT community - before they will come knocking on our doors, asking when they'll get to use that new OS that they just painlessly updated to on their own devices last night. Especially now that OS X is free and registers itself like an app store update.

It's this rapid cycle of change, encouraged by Apple and now gleefully adopted by the average consumer, that is going to make all of your resources for fresh knowledge so key in doing effective tech support.



Conferences

It's what makes conferences like MacTech so important.

##osx-server

on Freenode

macte.ch/helpyou1

Internet Relay Chat

It's why helpful internet chat rooms

managingosx.wordpress.com

krypted.com

afp548.com

Blogs

great up-to-date and knowledgeable blogs

MacEnterprise

@ lists.PSU.edu

macte.ch/helpyou2

Mailing Lists

and fantastic mailing lists

are something you should definitely take full advantage of.

And most importantly - it's why you need to figure out now, not later - how to peek under the hood of OS X, so to speak, so when the next thing changes and stuff starts breaking - you can be your own first "go-to" source for new knowledge.

And then maybe you can share it back out to the rest of us and help drop some of the noise level a bit and raise that signal up a few notches.



The Basics

So, enough background and personal philosophy of the IT world ... let's actually look at some of the tools that ship with OS X that you can use to diagnose what's going on with your machines and some of the additional tools that are available out there that can help you do your oh so very interesting job better.

Let's start with a very quick coverage of just a few basics that hopefully everyone should know.



"Why is my application crashing?"

"Why did my installation fail?"

If you're having problems with something on OS X, especially if you see an error message dialog or an application quits unexpectedly, the first place you should probably go check out is the Console. The Console is OS X's centralized graphical interface for most error logging.

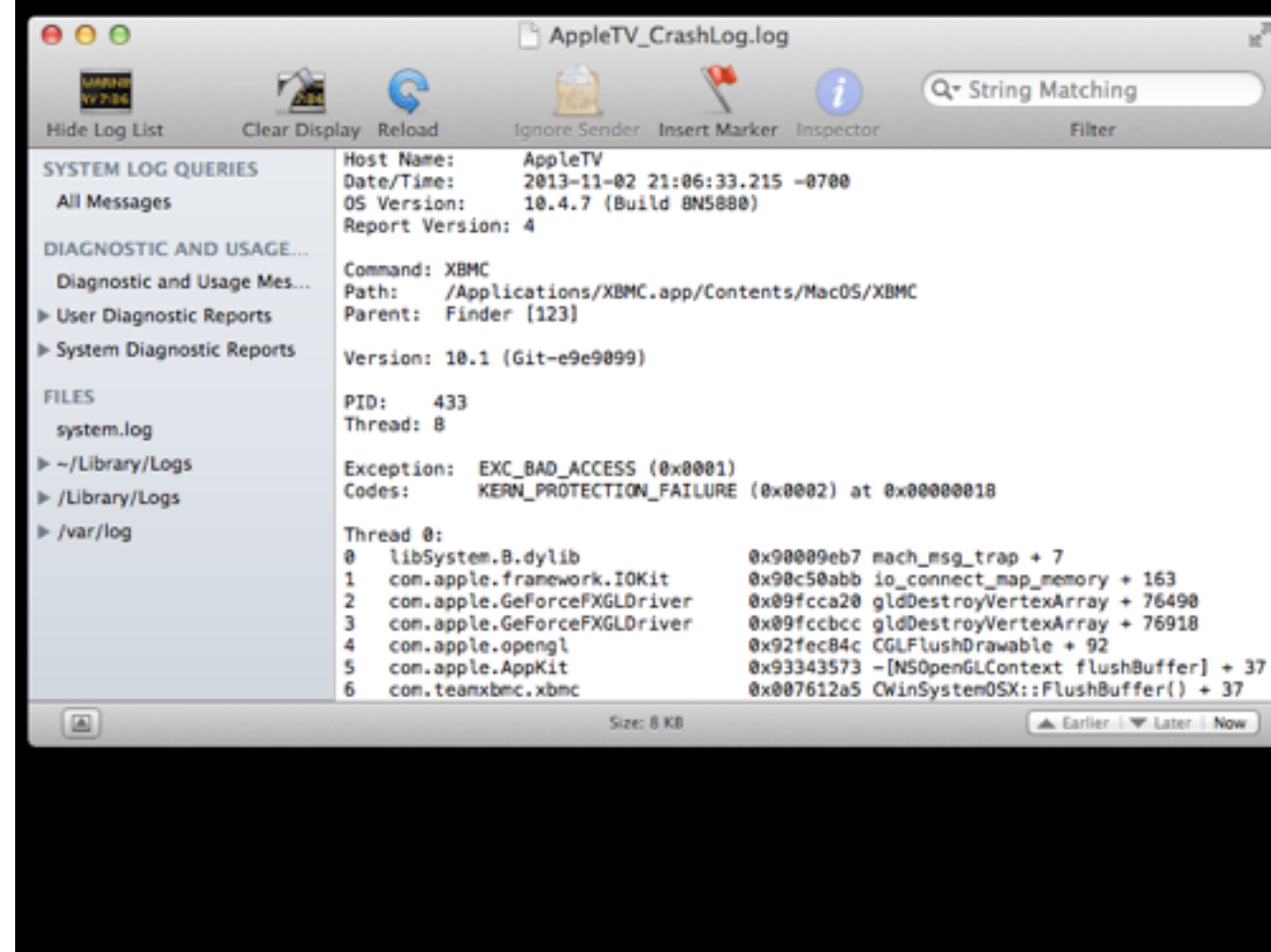
For example, whenever a package installation happens with OS X's installer, log entries are generally created in the installer.log which is linked directly into the Console for quick and easy access. For simple packages that are just installing files, the log file will rarely be illuminating. But most packages these days also include one or more scripts that contain executable logic to check if an install will be allowed, or to run commands before or after files get installed. If one of these scripts has a problem, you may be able to see the error and more details about what triggered it in the logs.

Logs are also generated when an application crashes for some reason. These logs can be rather detailed and likely contain more information than you can really make use of - but sometimes they can provide extremely useful clues as to why it crashed.

Interestingly enough, I personally inadvertently ran into a pretty great example of a crash log for my talk which I thought I'd share here today.

At home, we've got quite a few Apple products but one that tends to get the most use is an original 1st generation AppleTV that we've got in the master bedroom upstairs. Now the first AppleTVs didn't run on iOS like the current models do, they actually ran with a full Intel chipset on a stripped down variant of the version of OS X that was shipping at the time - Tiger - 10.4.

The cool thing about having a gen 1 model like this is that they're pretty trivial to install new software on them with the help of nothing more than a USB thumb drive. In my case, one of the things I installed was XBMC - which is a pretty decent universal media player that can do very useful things like stream media over your local network from a fileshare, and in formats that Quicktime doesn't natively support as well. And the installation just patches right into the menu system on the AppleTV - just shows up as an additional choice.



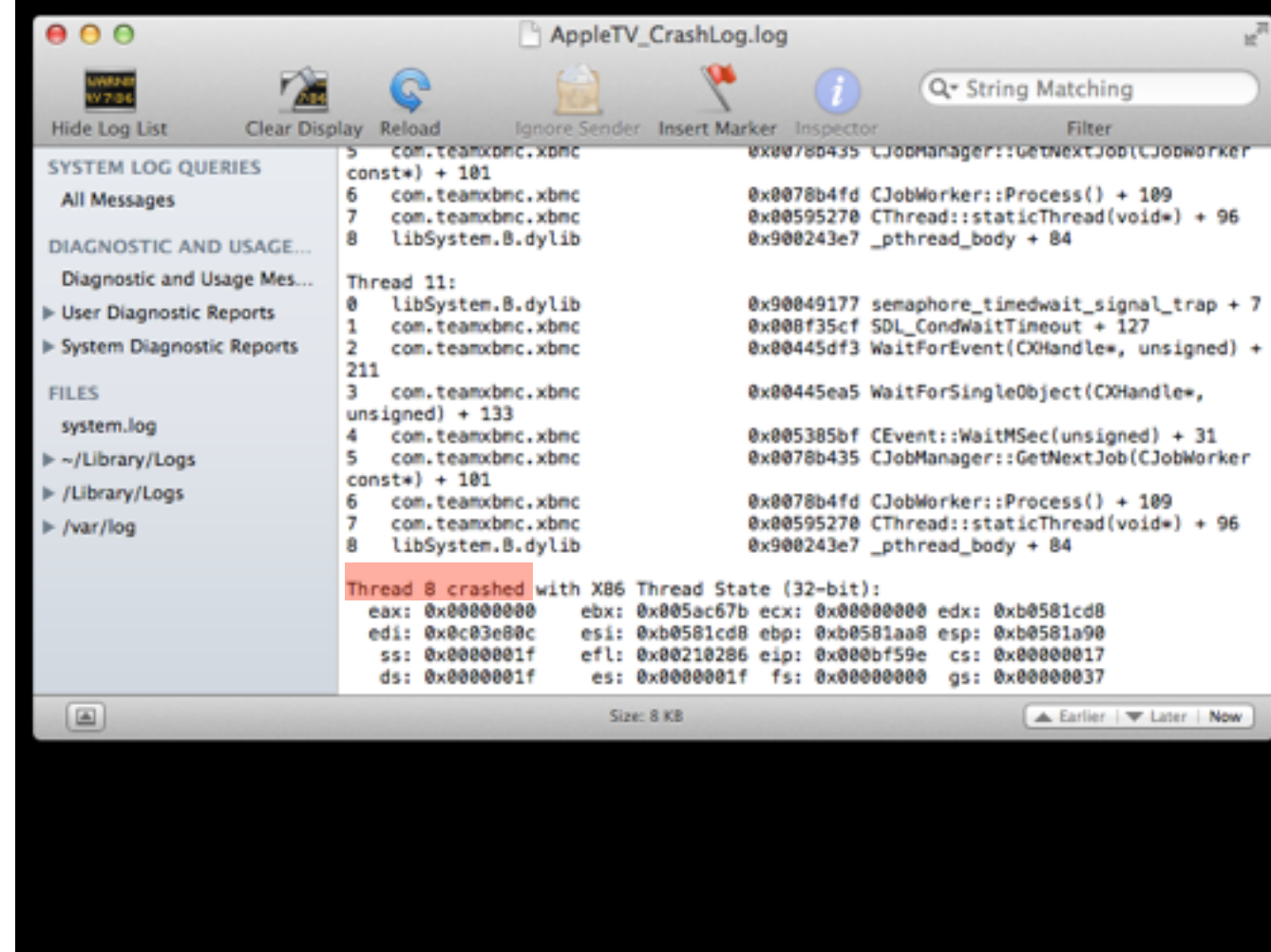
Exception: EXC_BAD_ACCESS

That doesn't sound too good.

Codes: KERN_PROTECTION_FAILURE

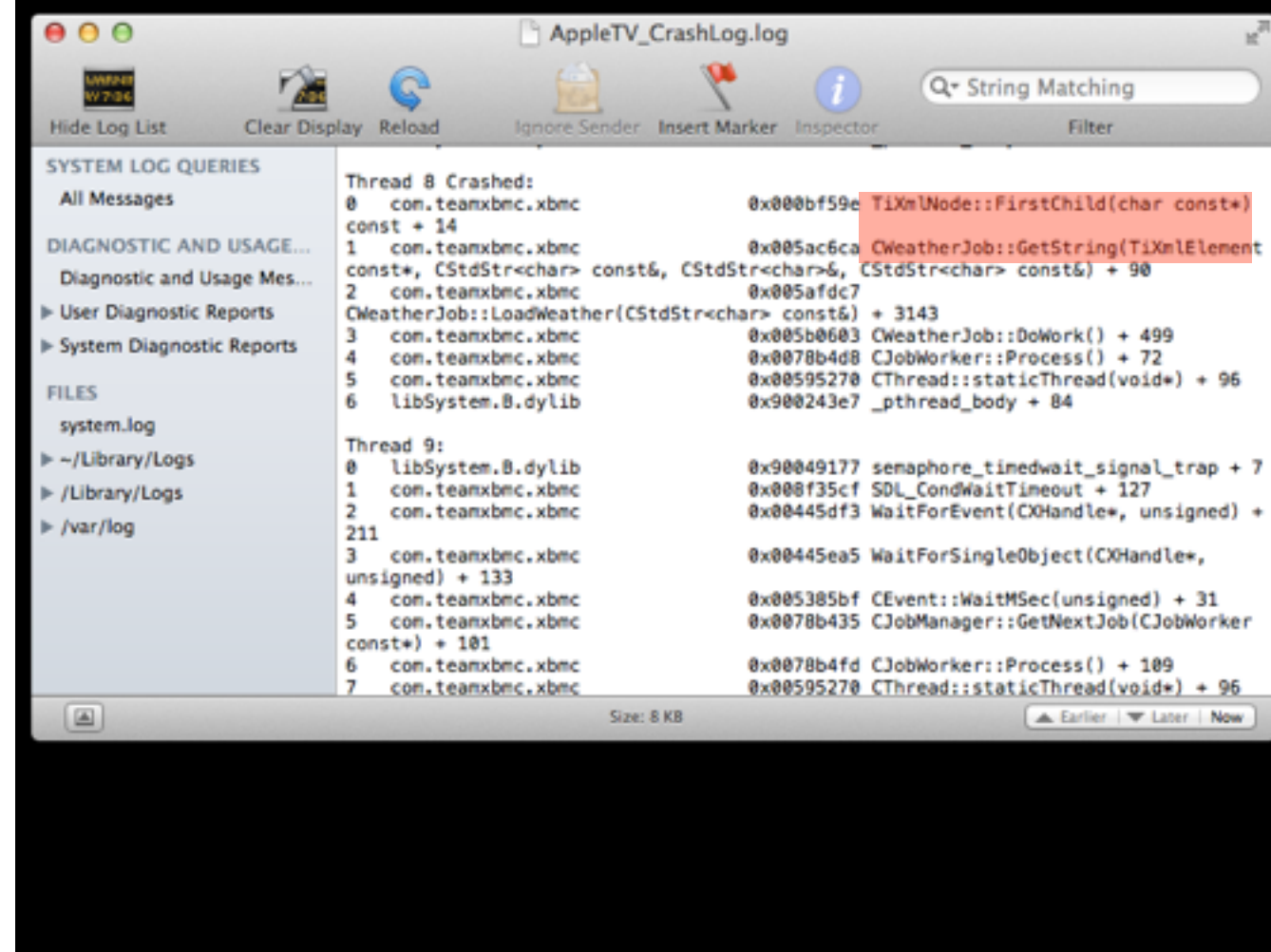
Hmm.

Pages and pages of stack traces mentioning drivers and frameworks. But if you know where to look, there are actually some really great clues here. The first thing you should try to find is the line where it tells you which thread crashed.



In this case, you can see at the bottom there the highlighted text "Thread 8 crashed".

So you take that bit of information and go back up to where Thread 8's stack is listed.



This stack seems much cleaner - very few drivers mentioned, it's mostly all the XBMC bundle identifier on the left. Meaning that most of the code involved with the crash is located likely within XBMC itself.

And on the right - "TiXmlNode FirstChild" right above "CWeatherJob GetString"

So thread 8 would appear to be crashing getting a weather-related string and trying to parse it as XML - and apparently the code has some bad error handling when it doesn't get what it expects and as a result XBMC is crashing.

As it turns out, XBMC has a plugin system where in addition to playing videos, you can add all sorts of additional functionality. In this case, the version of XBMC we have shipped out of the box with a built-in weather plugin that, at application launch, tried to pull local weather information to display on the home screen.

Honestly - until it started crashing my copy of XBMC, I never cared about knowing the temperature outside. The only time I was up there watching the AppleTV was when I was headed to bed.

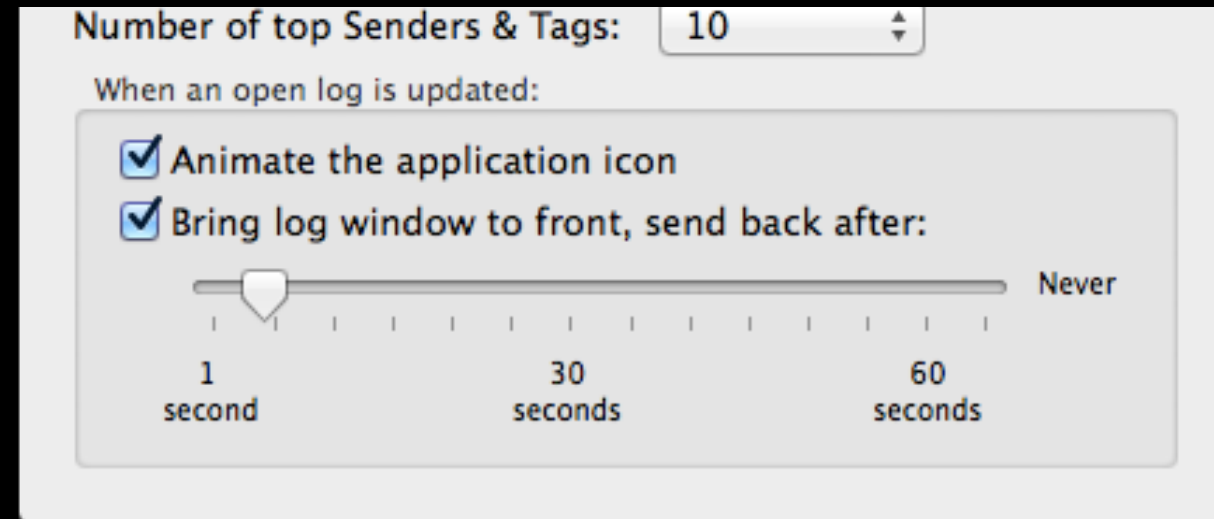
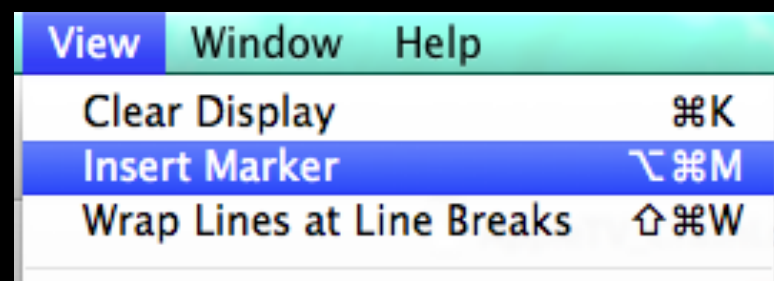
So rather than wait for the weather service to fix their malformed XML - or submit a bug report to the XBMC project and download newer version and hope it already had a fix, I just turned the weather plugin off.

Poof. Problem solved, crisis averted, back to your regularly scheduled Crow T Robot.

These are the kinds of clues you can get from the Console. And for properly developed software, the developers hopefully take advantage of OS X's logging services and will put out some useful details there that might help you quickly solve a problem when it really matters.

And speaking of that - I'd like to point out a change that happened recently with 10.8 and 10.9 that's related and give you a head's up on it. When you double-click an application in the Finder, the process that launches the application you chose is launchd. Amongst all its other automation duties, launchd is the parent process responsible for a GUI application while it's running. As a result, it's also the process that's the recipient of any messages printed out to the stdout or stderr file descriptors. For a command-line program stdout and stderr would print to the Terminal or wherever you redirected it - with a GUI application, prior to 10.8, launchd would redirect those file descriptors to the Console. However, the source code for launchd changed in 10.8 - you can see it at opensource.apple.com - and with that change it stopped doing that redirection.

Apple made the call, basically, that if you want to intentionally have something show up in the log for a graphical application - don't print it out to stderr or stdout - actually use the logging system APIs. That means that you might be



Oh - and I'll just mention, if you're trying to catch an error while it happens, Console also has some nice visualization tools I don't see mentioned too often, like icon animation and bringing the log window to the top when the log you're viewing is updated. Using these helps me get a better sense of what I was doing right when an error happens if I'm trying to re-create it.



Activity Monitor

The Activity monitor isn't particularly amazing from a diagnostic standpoint, but it *can* show you a few things that might help. The most common problem it can help with is "Why does this computer feel so sluggish?" It can show you the resource usage of currently running applications in OS X - including processor utilization. If you find a process that's run away with a high amount of CPU usage, that's probably the main source of your sluggish performance and what process it is should serve as a starting clue for what application on your computer is acting up - like a runaway mds or mdworker process. Both of those processes are integral to the Spotlight indexing service in OS X. When you see them go crazy and you haven't just recently changed a ton of files on your machine, you might actually be suffering from a Spotlight database corruption.

When a user tells you that their machine just feels slow, it's a good place to check.

Oh - and additionally, for CPU load, with 10.9, on a Mac laptop, you can click on the battery indicator in the menubar and it may list high CPU usage applications there as being bad for your battery life. It's limited to applications you'll likely only see in a force quit dialog, but it's quick and easy to check and it's a nice addition to 10.9.



Safe Mode isn't so much an application as it is an automatic problem solving tool in its own right. Maybe you just imaged a machine - or maybe you just installed some new software and rebooted. Now your machine won't start up. Safe Mode eliminates a few common sources of startup problems and can possibly point the way towards why you're having those problems.

It'll do an automatic disk check - and a repair if necessary. It clears out several different types of cache, such as the font cache. It disables all startup and login items and loads only the necessary kernel extensions for boot - disabling third-party installed extensions.

If your problems come right back the next time you boot out of Safe Mode, those last two items - startup items and third-party kernel extensions - are going to be a pretty good bet to check for the source of your problems.



Disk Utility & Apple Diagnostics

And to finish out the most basic tools, I wanted to make a special point to remind that it's not always software misconfiguration or a bad install. I've helped with problems with an application that simply cannot be repeated on another machine - only come to find out it's because the filesystem on the disk the application was trying to work with was corrupted in some subtle fashion. For an hour they went round and round, trying to determine why the application couldn't read a file that was very obviously right there - the Finder was able to see it, the Terminal could list it, but the application refused to properly open the file. 10 seconds with Disk Utility verification was enough to reveal the corruption - the application was doing nothing wrong, it truly could not read the file.

The same goes for Apple Hardware Test and Apple Diagnostics.

There was a machine that had been freshly imaged with a new OS and I found that shortly afterwards, wifi was not working correctly on it. The machine was unable to detect any wifi at all, actually. Being the first machine in the imaging batch, the immediate first thought I had was that something was wrong with the image. Wifi used to work, so why wouldn't it with the new OS? Rather than redo the image, an Apple Hardware Test scan turned up that indeed the wifi card on the machine was having problems. Sometimes things like that just happen.



More Advanced Tools

Now on to a few of the advanced tools that will let us peak behind the scenes of what's going on with OS X.

opensnoop

(and the dtrace family of commands)

opened files & more

opensnoop is one tool of many in the dtrace family of tools. If you want to see all the tools available in that family, `man -k dtrace` will give you the list of what OS X ships with. But before you go diving into all of them, let me talk about opensnoop a bit first.

The opensnoop command does just that - it snoops file opening by processes. It'll show you a running list of full file paths and the process names that opened them as they happen. It can also do the opposite function and for a given file, it can tell you the process that opens only that single file.

However, opensnoop and the dtrace family of scripts in OS X do have some limitations - several of the dtrace scripts included with OS X simply don't work. Others work, but in a limited fashion. opensnoop only prints out 15 characters tops of the process name - and it doesn't print the path to the process. Additionally, you may run into errors if you try to use the argument where you specify the process name you want to watch. It's better with opensnoop to just grep filter the output stream.

It's a useful tool in that it's quick'n'dirty and simple - it'll let you watch everything a program touches when you launch it. While this list may feel overwhelming, it may also include in there some file that you didn't know it was paying attention to settings in which turn out to be critical to the problem that you're running into.

The dtrace family of scripts offer all sorts of introspection into your machine, but again - they haven't been maintained with the new OS versions and kept compatible. Fortunately, for what we're interested in here, there are a few other tools that combined together can provide some great details.

fs_usage -f exec -w

spawned processes

The `fs_usage` command is an Apple provided command that gets you access to the File System events API at the kernel level of your machine. If you're not familiar with the API, it's what powers a lot of key components that depend on monitoring changes in OS X - like watched folder scripts and Spotlight indexing, Time Machine

Remember how I mentioned a lack of a full path to a process in `opensnoop`? Using `fs_usage` with the `-f exec` argument will cause it to print entries for the starting of processes - and the output contains the entire path to the executable in question.

This is a great way to capture all the moving components involved when kicking off an application and see short-lived helper processes that may be running during the startup of a program - possible key pieces that might need to be inspected if something isn't working quite right.

But `fs_usage` is a bit more powerful than just that.

```
fs_usage -f filepath -w | grep open
```

detailed open files

With some alternate arguments, and a little filtering help from grep, it can also output information similar to open snoop - showing files that are currently being opened. It also includes output showing the process name, process ID, and even the file modes like read or write etc. when the open action was triggered.

In fact, with a little work, you can combine them

```
fs_usage -f exec,filesys -w | egrep  
'spawn|exec|open '
```

spawned + opened

and get process paths as they launch, along with the files that are being opened, all in the same output.

I'd put up a slide here showing what the output comes out like - but it's a bit busy for display like this.

With `fs_usage` monitoring, you're pretty well covered for finding out most of the files involved for an application's startup. In fact, probably one of the major drawbacks to it is that there's a little bit too much information - and filtering it can sometimes be tricky.

Enter the third-party tool "fseventer"



fseventer

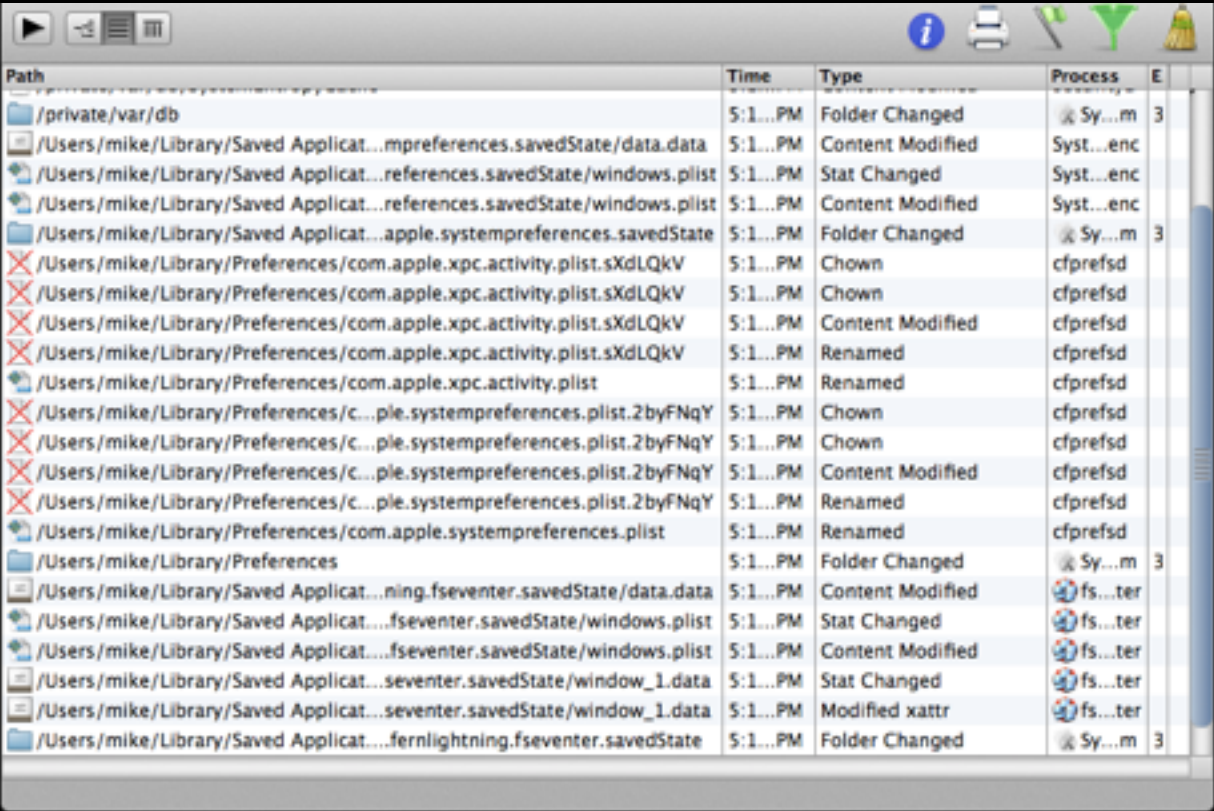
filesystem changes

macte.ch/helpyou3

I love this tool and I'm always amazed when people don't know about it. It's not included with OS X, but it really is an essential diagnostic tool in my opinion.

The one feature it doesn't have that `fs_usage` has is capturing simple file openings as they happen - a pure file read that doesn't modify a file will be missed in fseventer.

For everything else, and quite a bit more, though, fseventer is a fantastic logging tool.



The screenshot shows the fseventer application window with a table of filesystem events. The table has columns for Path, Time, Type, Process, and E. The events listed include folder changes, content modifications, stat changes, and renames for various system and user files, primarily in the /Users/mike/Library/Preferences directory.

Path	Time	Type	Process	E
/private/var/db	5:1...PM	Folder Changed	Sy...m	3
/Users/mike/Library/Saved Applicat...mpreferences.savedState/data.data	5:1...PM	Content Modified	Syst...enc	
/Users/mike/Library/Saved Applicat...references.savedState/windows.plist	5:1...PM	Stat Changed	Syst...enc	
/Users/mike/Library/Saved Applicat...references.savedState/windows.plist	5:1...PM	Content Modified	Syst...enc	
/Users/mike/Library/Saved Applicat...apple.systempreferences.savedState	5:1...PM	Folder Changed	Sy...m	3
/Users/mike/Library/Preferences/com.apple.xpc.activity.plist.sXdlQkV	5:1...PM	Chown	cfprefsd	
/Users/mike/Library/Preferences/com.apple.xpc.activity.plist.sXdlQkV	5:1...PM	Chown	cfprefsd	
/Users/mike/Library/Preferences/com.apple.xpc.activity.plist.sXdlQkV	5:1...PM	Content Modified	cfprefsd	
/Users/mike/Library/Preferences/com.apple.xpc.activity.plist.sXdlQkV	5:1...PM	Renamed	cfprefsd	
/Users/mike/Library/Preferences/com.apple.xpc.activity.plist	5:1...PM	Renamed	cfprefsd	
/Users/mike/Library/Preferences/c...ple.systempreferences.plist.2byFNqY	5:1...PM	Chown	cfprefsd	
/Users/mike/Library/Preferences/c...ple.systempreferences.plist.2byFNqY	5:1...PM	Chown	cfprefsd	
/Users/mike/Library/Preferences/c...ple.systempreferences.plist.2byFNqY	5:1...PM	Content Modified	cfprefsd	
/Users/mike/Library/Preferences/c...ple.systempreferences.plist.2byFNqY	5:1...PM	Renamed	cfprefsd	
/Users/mike/Library/Preferences/com.apple.systempreferences.plist	5:1...PM	Renamed	cfprefsd	
/Users/mike/Library/Preferences	5:1...PM	Folder Changed	Sy...m	3
/Users/mike/Library/Saved Applicat...ning.fseventer.savedState/data.data	5:1...PM	Content Modified	fs...ter	
/Users/mike/Library/Saved Applicat...fseventer.savedState/windows.plist	5:1...PM	Stat Changed	fs...ter	
/Users/mike/Library/Saved Applicat...fseventer.savedState/windows.plist	5:1...PM	Content Modified	fs...ter	
/Users/mike/Library/Saved Applicat...seventer.savedState/window_1.data	5:1...PM	Stat Changed	fs...ter	
/Users/mike/Library/Saved Applicat...seventer.savedState/window_1.data	5:1...PM	Modified xattr	fs...ter	
/Users/mike/Library/Saved Applicat...fernlightning.fseventer.savedState	5:1...PM	Folder Changed	Sy...m	3

fseventer

fseventer logs processes being launched and just about any kind of filesystem change you can think of, including deletions, renames, extended attribute changes, volumes being loaded and unloaded, and more. Included with the logging is a configurable filter where you can show only entries that contain a string or only files whose contents have been modified.

Content modification watching is a fantastic way to discover the various plists squirreled away in your system where a particular setting might be located when you toggle the switch for that setting on and off again. Countless times I've had people ask where a setting is stored only to direct them to this tool. I can't recommend it enough.

As a side note here, with 10.8 and later, talking about plist settings - OS X 10.8 introduced cfprefsd which is a background daemon tied to the Core Foundation preferences APIs. When the OS or an Apple application accesses a preference, if it's not loaded into memory - cfprefsd will pull it from the files on disk if it's there, but once it's read, it actually caches it in memory for a bit. This has the side effect that when you change a setting somewhere, hoping to trigger a file modification to see where that setting is stored - there's no immediate write back to a file on disk because the setting is cached in memory. In practice, though, it will eventually write the changes back to disk, usually within 30 seconds - tops.

Also, for those who are familiar with the product, for a while there the author had a warning that it appeared to be incompatible with 10.9 - if you visit the page again, you'll see the author has updated the message to state the source of the problem was found and fseventer once again is working quite nicely with Mavericks.

There are just a few more bits I'd like to cover here but they're mostly conceptual in nature.



Critical Thinking

Process of Elimination

Having a rock solid understanding of the process of elimination is critical when working with any major OS, OS X is no exception.

There are a lot of moving pieces when it comes to operating a modern desktop computer and eliminating as many of them as possible will hopefully point you to where the problem lies - or at least simplify things enough that you can get some outside help or suggestions as to what to do next.

Remove Extra Variables

- Is it reproducible ?
- Recreate the user profile
- Same machine, different user
- Network or local user account?

If you've got a problem application or a workstation where something odd is happening - try some very easy tricks to help narrow down the source.

Does the problem exist only on one machine or can you get it to happen on more than one? Is it related to a specific user? If you log in with a different local account, does the problem go away? Or if it's a network login, do they get the issue when they log into a different computer?

Remove Extra Variables

- Application version
- Operating system version
- Does it happen on a vanilla install ?
- Does it happen on only certain models of Mac ?

Do you have this same problem when you try running it on a machine that had OS X cleanly installed on it, direct from Apple's installation media - or does it only happen on machines that you built an image or configuration for? Do you have different models of Mac to try it out on?

Remove Extra Variables

- ~/Library/Preferences vs. /Library/Preferences
- Directory service binding
- Managed configuration: MCX and profiles
- Network configuration (**hint**: it's always DNS)

Make sure you've got a good understanding of the different locations where preferences can reside in OS X. In some instances, the settings may be coming down from an external source like MCX or a profile installed on the machine. Do you have the same problem when you try the machine on a different subnet or network? etc.

By doing simple checks like this, you can help prove your case that the problem isn't specific to something about your environment - or help reveal that maybe it is.



Your Time Has Value

And something I tend to personally lose track of, I admit it, is that it's important - from a process standpoint - to realize that your time has value. There 2 key things right off the top of my head that come to mind about this.



Paid Product Support ?

Use it !

First and foremost - if you're paying for support for a product in some fashion, by all means use that support! I'm trying to encourage self investigation of an issue, sure, but that doesn't mean you can't also have a ticket open while you're looking. Maybe they already know the answer. Maybe you'll get your ticket assigned to someone who knows the product you're wrestling with inside and out. At worst you might solve it before they do - but at best, they could solve your problem in a heartbeat.



Mac Dev - \$99/year

Test everything early

And please, if you're doing OS X support and you aren't participating in the Mac Dev program - do yourself a favor and get it now. It's not just for developers. You'll get pre-release versions of OS X in advance of the public release, which can help you test your environment before the next year rolls around - and file bugs if you find them, hopefully to get fixed before the final version comes out. It works out to less than \$10 a month, it should be a no-brainer sell for a tool purchase.

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

	HOW OFTEN YOU DO THE TASK					
	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES		6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR		10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS				2 MONTHS	2 WEEKS	1 DAY
1 DAY					8 WEEKS	5 DAYS

Source: xkcd.com

(Yes, sometimes you just re-image)

And in the end, when all is said and done - you have to take stock of how much time you've spent chasing a particular problem. If it's a problem that only affects a small handful of machines - or only one - a re-image may be the way to go. Sometimes the fish gets away. There's a diminishing return on the amount of time you spend solving a problem if the problem isn't actually one that blocks you from working - sometimes, at a certain point, you're spending more time trying to figure out the source of the issue - which you may never see again until that next cosmic ray comes down and flips a bit again in your drive - more time than it would have taken to just re-image to a known-good configuration. Set some limits on what's reasonable for an amount of time for a problem.

... and in the end, when you've do finally discover what the solution is to your problem

... share what you
learn !

Thank You

share what you learn! Blog it, post it in a knowledge base, whatever it takes. Let's raise that signal level out there.