Macworld

Tracking down trouble with the Console

by Kirk McElhearn, Macworld.com

Your Mac talks to itself a lot. OS X takes copious notes on what its various parts are doing; your applications send a constant stream of status messages to the operating system, too. All these notes and messages are stored in log files, which can be really handy when you need to troubleshoot your Mac.

These logs are plain text files, which you could view with TextEdit or any other text editor. But Mac OS X's Console (found in /Applications/Utilities) is a much better tool: it not only lets you read your system's logs, it helps you find and monitor them and filter their contents.

The basics

When you open Console, it automatically displays one of the most important log files: console.log. For most troubleshooting, this and system.log may be all you need. (To view system.log, click on the Logs icon in the toolbar, then click on system.log in the sidebar's Logs list.) These logs display their information in reverse chronological order; newest entries are at the bottom, and as entries get added, the log scrolls to always show the last line.

If you have problems with a specific program or with Mac OS X, start by checking console.log. One of the handiest ways to do so is to type the name of the program that's having problems, or a description of the problem, into the Filter field in Console's toolbar. For example, when I was having trouble syncing some data to my .Mac account, I searched for syn c and found the following log entry:

2007-09-23 14:06:10.433 cSync[1565] Unable to copy dbBlob for keychain 'login.keychain': -25294

I can't pretend that message meant much to me, but something was clearly amiss. So I went to Google and searched for the text Unable to copy dbBlob for keychain , which led me to a forum thread that suggested a solution. That's the general rule for using log files to troubleshoot your Mac: select the log file text that looks descriptive (yet doesn't include any personal information), do a Web search for it, and you may find forums, knowledge bases, or support sites that'll help you resolve the problem.

In another example, a friend of mine was having problems with Mail crashing. I asked him to send me his console.log. (That's the other great thing about log files: they're easy to pass around.) It showed that Mail was trying but failing to load several plug-ins, and that failure was causing the crashes. My friend had been trying out some software that installed Mail plug-ins. But when he deleted the software in question, he didn't delete the plug-ins from your user folder/ Library/Mail/ Bundles. Since console.log had pinpointed the problem, I was able to tell him what to delete, solving the problem.

Lots of programs write to console.log. For example, if you use Alsoft's $100 DiskWarrior, and you have it set to run automatic hardware diagnostics on your hard disks, it'll generate a long list of log entries, which can help you diagnose disk problems. Apple's Disk Utility also writes log entries to console.log whenever you create a disk image or format a disk. Finally, if your Mac is having trouble with a peripheral, it may be having problems loading drivers for the device; console.log should record those snags. (For more on that, see Ted Landau's discussion.)

While console.log tracks events in your particular user account, system.log tracks events affecting your Mac as a whole. You'll see, for example, that a number of entries are added to it each time you start up your Mac. Scanning through system.log, you'll find entries for drivers and kernel extensions that load when you boot. It's useful to review system.log for device drivers or programs you no longer use; you can then track down and remove the software elements you don't need. If you're having a problem with a hardware device and you don't find any trace of it in console.log, check system.log to see if its driver is loading; if not, you may need to reinstall the driver. As with console.log, you may not always be able to understand system.log entries, but they can point you in the right direction.

Other logs

At times console.log and system.log won't tell you what you need to know. That's when you should click on the Logs icon in Console's toolbar and search for the other log files OS X maintains.

The first entry after console.log and system.log is ~/Library/ Logs. Click on the disclosure triangle, and you'll see a list of logs and other folders, all in your home folder; they track activity just within your own user account. The other logs in that list, /Library/Logs and /var/log, contain systemwide logs. (Note that your personal console.log file is actually located in /Library/Logs/ Console/ your user ID ; that ID will be 501 if you're the only user of your Mac and a higher number for other users.)

You'll need to investigate these other logs when some piece of software crashes for no apparent reason. Your Mac stores special types of logs, called crash logs, for each program that quits unexpectedly. In Console's Logs list, click on the disclosure triangle next to ~/Library/Logs, and then select CrashReporter. There you'll see a list of all the programs that have crashed on your Mac. Click on one of these logs to see its contents. Unless you're a programmer, those contents will look like gibberish. But, as with console.log, you can glean some useful bits of information and then find out more online.

If you manage many Macs, you'll also want to know about secure.log, located in /var/log. It records every time a user authenticates as an administrator, such as when installing software or changing certain system preferences, as well as each time a user runs the

sudo

command in Terminal (or via a remote connection). This log can tell you if users are exceeding their authorizations and what authorized users are doing. It does, however, have its limits; for example, while it can tell you which users are installing certain programs and when, it can't tell you which programs they installed.

Process of elimination

All of these logs can help you troubleshoot software and hardware problems. But given the many lines in console.log, it can sometimes be hard to find the specific entries that relate to the problem you're having.

There is, however, a work-around: if you're trying to isolate the cause of a specific problem, go to console.log, and then click on the Clear button in Console's toolbar; this clears the display of the log's entries. Next, reproduce the actions that caused your problem—launching the program, connecting the hardware device, whatever. That should generate new log entries in console.log (or one of the other log files I've described), preceded by a date stamp. In many cases, the cause of the problem will be right there.

Macworld