

Log4Tailer
Not just a simple log tailer

Colors, like features, follow the changes
of the emotions.

Pablo Picasso

Copyright 2009 by Jordi Carrillo

Permission is granted to copy, distribute and/or modify *this document* under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

Log4Tailer User's Guide

Version 2.0

<http://code.google.com/p/log4tailer>

Jordi Carrillo

October 17, 2009

Contents

I	The Basics	5
1	Introduction	5
1.1	Why yet another tailer?	5
1.2	Why should I use it?	6
2	Installation and system requirements	6
2.1	System Requirements	6
2.2	Installation	6
3	Command line parameters	6
II	Advanced features	7
4	Targets	7
4.1	Every log with its own target scheme	7
5	Throttling	7
6	Actions	7
6.1	PrintAction	8
6.1.1	Available colors	8
6.1.2	Providing user defined colors	8
6.1.3	Every log with its own color	8
6.2	MailAction	9
6.2.1	Flood control	9
6.2.2	When should I use MailAction	9
6.3	InactivityAction	9
6.3.1	InactivityAction Mail Notification	10
7	Pause Modes	10
8	Analytics	10
8.1	Reports by email	11
9	Silent Mode	11

10 Coloring Standard Input	11
11 Tailing last N lines	11
12 SSH Tailing	12
12.1 Dependencies for SSH tailing	12
13 Configuration file	12
III Case Studies	15
14 Full Automatic Log Monitorization	15
15 Semi Automatic Log Monitorization	15

Part I

The Basics

1 Introduction

Log4Tailer aims to provide a different approach to traditional log tailers. While the famous `tail` linux command line tool does its job just well, `log4tailer` adds some more advanced features, like throttling, user defined colors and email notification just to name a few. List of features:

- Multitailing capability. It can tail multiple logs at a time
- Colors for every level: warn, info, debug, error and fatal
- Emphasize multiple targets (log traces) given regular expressions
- Follow log upon truncation by default
- User defined colors for each level
- Silent (daemonized) mode for full automatic log monitorization
- Throttling mode. Slow down the information being printed in the terminal
- Inactivity log monitoring
- mail notification
- Every log with its own color scheme
- Log reporting by email or standard output
- Tailing remote logs by means of ssh (as of version 2.0)

1.1 Why yet another tailer?

Most people use `tail -F` to tail the logs these days. When debugging enterprise class applications you cannot just follow (in many situations) what is going on unless you go to the log, less it and check if something was wrong, or just `Ctrl-C` tail program and scroll back. Human eye cannot distinguish or grab a line out of thousands when that information is showed incredibly fast in the screen. By providing colors, the human eye will discern and quickly identify specific levels or lines.

The `tail` command line Unix/Linux tool has frustrated me many times, so I decided to write my own `tailer` adding more functionalities as I needed them. There might be other tools out there, but to use them you must install them system wide or write your own regexes and remember the terminal escape characters to colorize the output. It wouldn't be the first time I see people writing perl one liners with complex regexes in order to tail logs. **Log4Tailer** is an standalone application that you can run from your `/home` directory if you desire with default common sense default colours. Of course, you will need read access to those logs you want to monitor.

1.2 Why should I use it?

Just to name a few advantages:

1. **Log4Tailer** is opensource, free software.
2. It runs in standalone mode. At this very moment, no installation is necessary. If you are a sysadmin or an engineer monitoring logs in a network operation center, you can run **Log4Tailer** from its folder. Just untar and run the tailer.
3. Why do it in black and white, when you can do it with colors?

2 Installation and system requirements

2.1 System Requirements

You'll need at least version 2.4 of Python installed in your system. Python is multiplatform and you can have a copy in <http://www.python.org>.

Log4Tailer has been tested in Linux servers and it should run in any xterm compatible terminal such as Putty, GNOME terminal and others.

2.2 Installation

Log4Tailer can run in standalone mode, so no installation is necessary if you don't want to. Untar and run ;-). If for convenience, you want to install it, just type (having root access):

```
python setup.py install
```

and it will be installed system wide. For convenience, you can download the rpm, which has been packaged for SUSE Enterprise Server 10.

3 Command line parameters

The full command line list is as follows:

```
./log4tail [-c configfile] [-n numlines] [-t targets] [--throttle secs] [-i inactivityTime]  
          [-m mailAction] pathToLogs
```

or in silent mode, just:

```
./log4tail -s [-t targets] [-c configfile] [-i inactivityTime] fullPathtoLogs
```

Part II

Advanced features

4 Targets

Target option `-t` provides an easy way to identify, by means of a regular expression, a particular log trace in your logs. By default, it will tail as normal and if your regex is matched, then that line will be emphasized in a red background.

Example:

```
this is a log trace matched by target option
```

You can specify multiple comma separated targets. Each target is just an string being itself a simple string or just a regular expression. It must be enclosed within simple or double quotes.

```
./log4tail -t [--target] 'regex1,regex2,regex3,...,regexN' pathToLogs
```

4.1 Every log with its own target scheme

If you want every log with its own targets, then you must provide them in a config file. In order to do that, just edit a text file and specify the next key, value optional parameters:

```
targets fullpathToLog0 = regex0, regex1, regex2,...., regexN
targets fullpathToLog1 = regex0, regex1, regex2,...., regexN
```

An example of that could be:

```
targets /var/log/messages = iptables$, ^2009-08-07 user
targets /var/log/mail.log = incoming \d+, \d{1,3} something
```

5 Throttling

In release v05, the throttling option is provided. Many applications when started usually tend to upload configuration parameters and log them very fast, which makes nearly impossible to identify anything (unless you open the log and check). In that case, or in other ones where an application logs very fast, you can provide the *throttle* option specifying the number of seconds you want to output in the screen the logging information one line at a time. The number of seconds can be a floating point number.

```
./log4tail --throttle 0.5 pathToLogs
```

That would tail the logs showing the information every half a second one line at a time.

6 Actions

Log4Tailer uses *Actions* to identify what kind of action needs to trigger when it matches a certain line with an specific log4j level. The actions provided as of version v05 are:

- PrintAction
- InactivityAction
- MailAction

6.1 PrintAction

PrintAction does what the famous *tail* command does, just printing in the screen (terminal) but with colors. When an application logs information very fast, colors provide an easy way to quickly identify a certain pattern or problem. Every color identifies a specific level according to the log4j java framework.

6.1.1 Available colors

Log4Tailer provides the following colors:

- black (default for debug)
- red (default for fatal and critical)
- green (default for info)
- yellow (default for warning)
- blue
- magenta (default for error)
- cyan
- white

6.1.2 Providing user defined colors

Default colors used in **Log4Tailer** work well in a clear background terminal, such as white. If your terminal has black as background you can provide your own colors in a config file such as:

```
warn = yellow
fatal = red
critical = red
error = magenta
info = green
debug = black
```

and pass `-c pathtoconfig` as a parameter to **Log4Tailer**.

The colors provided above are actually the default ones. It is not necessary to provide all the colors in the config file. If yellow is fine for warn and red for fatal, you could say something like:

```
error = red
info = magenta
debug = green
```

error, info and debug colors will be overridden by your ones provided in the config file.

6.1.3 Every log with its own color

If you want to just tail different logs and you want each log with its own specific color, then you can specify that in the config file. This feature overrides the level colors for that specific log, printing all traces with the same color.

```
fullPathLog0 = green
fullPathLog1 = yellow
```

In that specific case, if we run:

```
log4tail -c configfile fullPathLog0 fullPathLog1 fullPathLog2
```

In the output you'll see *all* traces from fullPathLog0 in green, *all* traces from fullPath1 in yellow and in the specific case of fullPathLog2 it will use the default color for each level. An example will clarify that:

```
==> fullPathLog0 <==  
INFO this is an info log trace from fullPathLog0  
ERROR this is an error log trace from fullPathLog0  
==> fullPathLog1 <==  
INFO this is an info log trace from fullPathLog1  
DEBUG this is a debug log trace from fullPathLog1  
==> fullPathLog2 <==  
FATAL this is a fatal log trace from fullPathLog2  
INFO this is an info log trace from fullPathLog2
```

6.2 MailAction

MailAction is used when you want to be notified by email when a target or level (error or fatal) has been found in the logs. It is very useful when you are tailing for long hours and you cannot take a look at the screen from time to time. It's not necessary to run in silent mode anymore to use this action. This action can be triggered by specifying the command line option -m.

```
./log4tail -m [--mail] [-t [--targets] 'regex1,regex2,regex3,...,regexN'] pathToLogs
```

Automatically, log4tailer will request the SMTP details of an email account, like SMTP host, username, password, and the from and to email address.

6.2.1 Flood control

In order to avoid being sent lots of notification emails when a flood of undesirable log traces turn up, log4tailer has a way to control that by means of a 60 second gap, which means that non desirable levels that happen within that gap are not notified. After that expiration time, the first undesirable log trace to be found will be notified, triggering again a 60 second gap period.

6.2.2 When should I use MailAction

MailAction would be like having additional eyes taking a look at the logs. That means, that you can take a rest from time to time basically. If something is found, then you are notified. At this stage, it will notify errors and fatals, considered to be non desired levels in an application. Along with that, you can specify a series of patterns (regexes), log4tailer's targets, that if found could mean that the application is not behaving as expected. In that specific case, you will get notified as well.

6.3 InactivityAction

InactivityAction monitors for inactivity time in the logs. Inactivity as of this release will just send an emphasized line in the standard output notifying that there has been a lot of inactivity in that log. The inactivity time must be provided in the command line with the -i parameter followed by the number of seconds of inactivity to be monitored in the log. If there has not been any activity for the number of seconds given, **Log4Tailor** will print an emphasized line in the standard output.

As an example:

Inactivity in the log for 5.99955296516 seconds

The command line interface to activate the inactivity monitoring is:

```
./log4tail -i [--inact] numberinseconds pathToLogs
```

6.3.1 InactivityAction Mail Notification

If you want a notification by email when inactivityAction is raised, just specify in the config file:

```
inactivitynotification = mail
```

By default is notification to the standard output as shown before.

7 Pause Modes

As of release 1.2 **Log4Tailer** includes pausemodes feature. You will be able to pause or freeze the output by a number of seconds if an specific level or target has been found. In order to enable pausemodes, you must configure them in a config file providing any of the following keys¹:

```
pausedebug = secondsfordebug
pauseinfo = secondsforinfo
pausewarn = secondsforwarn
pauseerror = secondsforerror
pausefatal = secondsforfatal
pausetarget = secondsfortarget
```

You specify only those ones you want to use. For instance, if we want to freeze the output momentarily (one second) for warnings:

```
pausewarn = 1
```

Then, we should run log4tailer like:

```
./log4tail -c yourconfig /pathToLogs
```

Pausetarget keyword will pause the output for any regex found in the log when running log4tailer with -t option.

8 Analytics

Every time we finish the tailing, log4tailer will output a report, specifying how long log4tailer has been running and the number of events for debug, info and warn. In case of error and fatal, it will provide the timestamps when they were found and their corresponding logtrace. Example:

```
Analytics:
Uptime:
0.0 years 0.0 days 0.0 hours 0.0 mins 45.9482619762 secs
Report for Log out.log
Levels Report:
FATAL:
```

¹the keys are case insensitive, so is the same pauseDEBUG or pausedebug...

```
ERROR:
15 May 2009 17:17:43=>> There was an error here
15 May 2009 17:17:44=>> There was another one in here
15 May 2009 17:17:45=>> Oops, another one
WARN:
4
INFO:
9
DEBUG:
14
TARGET:
3
Ended log4tailer, because colors are fun
```

8.1 Reports by email

If you want a report by email after a given amount of time, then you can do that by means of the config file. There are two values that can be setup, namely:

```
analyticsnotification = mail
analyticsgaptime = 10.5
```

If these two values are uncommented, then you will be asked to give the SMTP details of your email. The analyticsgaptime should be given in seconds, by default is 3600 seconds (1 hour). You'll receive a report after that period. After that period the statistical information is flushed and then sent again once the gap notification time is expired and so on.

9 Silent Mode

Silent mode, tails the logs in the background (daemonized tailer) and triggers the MailAction, notifying if error, fatal or any target has been found in the logs.

```
./log4tail -s [-t [--targets] 'regex1,regex2,regex3,...,regexN'] fullPathToLogs
```

Log4Tailer will ask you the SMTP hostname, username and password to be used (MailAction) in order to notify you in case of a problem.

10 Coloring Standard Input

Log4tailer can colorize its standard input to the standard output. Main use would be when your application does some output and finishes. In order to do that just type:

```
yourapplication | log4tail -
cat somelog.log | log4tail -
```

11 Tailing last N lines

You can tail last *N* lines from the log with the *-n* option. Just type:

```
./log4tail -n numberOfLines
```

and it will output the last *numberOfLines* from the log colorizing the corresponding levels.

12 SSH Tailing

SSH Tailing or remote tailing will allow you to tail multiple remote logs from different hosts. As of now, only PrintAction is available, so you'll be able to tail multiple remote logs in a colorful way as specified in section 6.1. In order to tail remotely you'll need to pass as a parameter the -r option along with some config file parameters:

```
./log4tail -r -t targets -c yourconfig.txt
```

In your configfile you must provide the following parameters:

```
sshhostnames = hostname0, hostname1, hostnameN-1
hostname0 = username0, /var/log/log0, /var/log/log1, /var/log/logN-1
hostname1 = username1, /var/log/log0, /var/log/log1, /var/log/logN-1
hostnameN-1 = usernameN-1, /var/log/log0, /var/log/log1, /var/log/logN-1
```

Where:

- **sshhostnames** is a comma separated values of hostnames
- every **hostname** must be a parameter itself where first value should be its username and then the logs you want to tail.

Once setup, log4tailer will ask you for the password and if that password is the same for all hostnames, in this way, you'll not have to write down the password in any file, it will be dynamically requested.

Some considerations are to be taking into account. As of now, remote tailing **only** provides PrintAction along with targets, that means that you will be able to tail with colors and emphasize those log traces that match the comma separated regexes provided with -t. Besides, you'll be able to use pauseModes set up in the config file as explained in section 7.

To finish, just Ctrl-c and it will close all channels opened to communicate to the remote hosts.

12.1 Dependencies for SSH tailing

It is very important to note that for remote tailing, you'll need to install the **paramiko module**, available in major Linux distributions. In most of them is available under the name of python-paramiko. In Debian systems, you'll need to type:

```
sudo apt-get install python-paramiko
```

13 Configuration file

Config file is provided fully documented for convenience; just uncomment those lines you are interested to enable and modify them for your specific purposes. In order to enable those values in the config file, you must notify that to **Log4Tailer** as a parameter in startup time.

```
./log4tail -c yourconfig.txt logs
```

The config file provided is called log4tailerconfig.txt and is quoted below:

```
# Optional config for log4tailer
# to activate it
# log4tail -c config yourlogs

# =====
```

```

# SSH Tailing parameters
# =====

# sshhostnames = hostname0, hostname1, hostnameN-1
# hostname0 = username0, /var/log/log0, /var/log/log1, /var/log/logN-1
# hostname1 = username1, /var/log/log0, /var/log/log1, /var/log/logN-1
# hostnameN-1 = usernameN-1, /var/log/log0, /var/log/log1, /var/log/logN-1

# =====
# targets: which lines do you want
# to emphasize by using regexes
# uncomment and provide your values.
# =====

# targets path/to/log0 = regex0,regex1,regex2
# targets path/to/log1 = regex0,regex1,regex2

# =====
# Custom colours for every level. Available
# colours are: red, green, yellow, blue, magenta, cyan
# and white. Uncomment to override the default ones.
# =====

# warn = yellow
# error = magenta
# fatal = red
# info = green
# debug = black

# =====
# Every log with its own color scheme, overriding colors
# for every level.
# =====

# /path/to/log0 = yellow
# /path/to/log1 = red

# =====
# Pause the output by the number of seconds specified if
# a level or target has been found. Uncomment the ones
# you want. The value can be any number in seconds.
# =====

# pausedebug = 4
# pauseinfo = 2
# pausewarn = 1
# pauseerror = 1
# pausefatal = 1
# pausetarget = 1

# =====

```

```
# Inactivity action notification, by email or stdout.
# Possible values can be "mail" or "print". By default is "print".
# =====

# inactivitynotification = mail

# =====
# Analytics notification. You can make log4tailer send you
# a report every analyticsgaptime seconds. By default it will be
# printed out once finished. Uncomment analyticsnotification to
# report by email. Another possible value can be "print".
# =====

# analyticsnotification = mail
# analyticsgaptime = 10.5
```

Part III

Case Studies

14 Full Automatic Log Monitorization

Full Automatic log monitorization can be performed when you execute **Log4Tailer** in silent mode passing the parameters `-s`. Log4Tailer will run silently in the background notifying by email when something goes wrong. As of now, it will notify errors, fatals and those targets specified as a parameter or in the config file. It is important to notice that every log can have its own set of targets (regexes). Apart from that, you can make log4tailer to monitor inactivity in the log and notify you by email as well. You just need to specify that in the config file as explained in the section 6.3.

Summing up, full automatic monitorization will monitor inactivity, errors, fatals and targets specified in the config file or command line as parameters. This will give you extra confidence on the monitoring of your application if your application uses already nagios or other monitoring software. ²

15 Semi Automatic Log Monitorization

You can have a mix of email notification and normal colorized print action. You just need to execute log4tailer passing as a parameter `-m` and the corresponding configfile if you want to enable additional features.

²It is important to notice that pausemodes should not be enabled. That feature is to pause the output when having PrintAction enabled.