Mirror of Radovan Garabík's - Generic Colouriser for terminal apps http://korpus.juls.savba.sk/~garabik/...

7 commits | 1 branch | 0 releases | 2 contributors

master | New pull request | Find file | Clone or download

pengwynn Merge pull request #2 from em-/master ... Latest commit 278ec75 on Jun 22, 2013

contrib	Initial import from tarball	6 years ago
debian	Initial import from tarball	6 years ago
CHANGES	Initial import from tarball	6 years ago
COPYING	Initial import from tarball	6 years ago
CREDITS	Initial import from tarball	6 years ago
INSTALL	Initial import from tarball	6 years ago
README	Update RegEx link from Python docs	5 years ago
Regexp.txt	Initial import from tarball	6 years ago
TODO	Initial import from tarball	6 years ago
conf.configure	Initial import from tarball	6 years ago
conf.cvs	Initial import from tarball	6 years ago
conf.df	Colourize listings from `df`	5 years ago
conf.diff	Initial import from tarball	6 years ago
conf.esperanto	Initial import from tarball	6 years ago
conf.gcc	Do not split colour from the `-o` option in `gcc`	5 years ago
conf.irclog	Initial import from tarball	6 years ago
conf.ldap	Initial import from tarball	6 years ago
conf.log	Initial import from tarball	6 years ago
conf.mount	Colourize listings from `mount`	5 years ago
conf.netstat	Initial import from tarball	6 years ago
conf.ping	Initial import from tarball	6 years ago
conf.proftpd	Initial import from tarball	6 years ago
conf.traceroute	Initial import from tarball	6 years ago
conf.wdiff	Initial import from tarball	6 years ago
grc	Initial import from tarball	6 years ago
grc.1	Initial import from tarball	6 years ago
grc.conf	Colourize listings from `df`	5 years ago
grc.spec	Initial import from tarball	6 years ago
grc.spec.old	Initial import from tarball	6 years ago
grcat	Initial import from tarball	6 years ago
grcat.1	Initial import from tarball	6 years ago
install-stamp	Initial import from tarball	6 years ago
install.sh	Initial import from tarball	6 years ago

README

```
============================================================
                    Generic Colouriser

Radovan Garabík  http://kassiopeia.juls.savba.sk/~garabik/software/grc.html
garabik @ kassiopeia.juls.savba.sk

============================================================

Being overflooded with different logfile colo(u)?ri(s)(z)ers, colortails,
gccolors, colormakes and similar programs for making  text files or outputs
of different programs more readable by inserting ansi colour control codes
into them, I decided to write  my very own colouriser, eventually providing
the functions of all those others.

Two programs are provided: grc and grcat.  The main is grcat, which acts as
a filter, i.e. taking standard input, colourising it and writing to
standard output.

grcat takes as a parameter the name of configuration file.

Directories  ~/.grc/, /usr/local/share/grc/, /usr/share/grc/ are searched
for the file (in this order). If the file is not found, it is assumed to be
an absolute path of a configuration file located elsewhere.

Configuration file consists of entries, one per regexp, entries are
    separated with lines with first character non-alphanumeric (except #).
    Lines beginning with # or empty lines are ignored.

    Each entry consists of several lines.
    Each line has form:
        keyword=value
        where keyword is one of: regexp, colours, command, skip, count
        Only regexp is mandatory, but it does not have much sense by itself
        unless you specify at least a colour, skip or command keyword as well.

        regexp is the regular expression to match

        colours is the list of colours, separated by commas (you can specify only
        one colour), each colour per one regexp group specified in regexp.
        if you use special colour name "previous", colour of the previous line
        of text will be used (actually, if both the first and last character of
        the previous line are of different colour than the default one,
        colour of the first one will be used).
        Another special colour name "unchanged" will leave the colour
        unchanged, useful if you need some context for matching
        regular expression and you want to leave the colour of context
        unchanged.
        Yet another special name is an arbitrary string enclosed in
        straight quotes. This string will be inserted directly into
        the output in front of the matching expression. The string will
        be eval'ed, so you can use usual python escape sequences.
        This is useful on a 256-colour enabled xterm, where e.g.
         colours="\033[38;5;22m" will give you a dark green (inspired
        by Rutger Ovidius). Caveat: the string cannot contain a comma. This
        is due to my laziness :-)


        command is command to be executed when regexp matches. Its output will
        be mixed with normal stdout, use redirectors ( >/dev/null) if you want
        to suppress it.

        skip can be skip=yes, if that case the matched line is skipped
        (discarded from the output), or skip=no, when it is not skipped.
        Default (if you do not have skip keyword) is of course not skipped.

        count is one of words: once, more, stop, previous, block or unblock

            once means that if the regexp is matched, its first occurrence is coloured
            and the program will continue with other regexp's.

            more means that if there are multiple matches of the regexp in one line,
            all of them will be coloured.

            stop means that the regexp will be coloured and program will move to the
            next line (i.e. ignoring other regexp's)

            previous means the count will be the same as for the previous line

            block marks a start of a multiline block of text, coloured with
            the same colour

            unblock, obviously, marks the end of such a block


        example:

        # this is probably a pathname
        regexp=/[\w^\.]+
        colour=green
        count=more

        this will match /usr/bin, /usr/local/bin/, /etc/init.d/syslogd and similar
        strings and paint it with green.

        Another example:

        regexp=^.{1,2}\s{0,1}$
        colours=red
        count=block
        -
        regexp=^\s{0,5}$
        colours=default
        count=unblock

        this will turn all correctly formatted mail signatures red.

    Regular expressions are evaluated from top to bottom, this allows nested
    and overlapped expressions. (e.g. you colour everything inside parentheses
    with one colour, and if a following expression matches the text inside
    parentheses, it will be also coloured)

Typical usage:

grcat conf.log < /var/log/syslog
/usr/sbin/traceroute www.linux.org | grcat conf.traceroute
grcat conf.esperanto < Fundamento.txt  | less -r

To facilitate the use, command grc acts as frontend for grcat, automatically
choosing the configuration files, so you can write:

grc netstat
grc ping hostname
grc tail /var/log/syslog

etc...

    grc will execute command command with optional parameters piping its stdout
    into grcat.

    Configuration file for grcat is determined by /etc/grc.conf or
    ~/.grc/grc.conf file.

    Format of /etc/grc.conf or ~/.grc/grc.conf: each entry consists of 2 lines,
        between entries there can be any number of empty lines or lines beginning
        with # (comments)

        First line is regular expression, second line the name of configuration
        file for grcat

        Configuration file after the first regular expression matching the rest of
        line after grc will be passed to grcat as its configuration file

        For example, if you have

        # log file
        \b\w+\b.*log\b
        conf.log

        # traceroute command
        (^|[/\w\.])+/traceroute\s
        conf.traceroute

        in your /etc/grc.conf, then typing grc cat /var/log/syslog will use
        conf.log to colourise the output,
        grc /usr/sbin/traceroute www.linux.org will use conf.traceroute

Miscellaneous remarks:

    You should get yourself familiar with regular expressions. Good reading is
    at http://docs.python.org/dev/howto/regex.html

    The program is not yet optimized for speed. There are places that can
    give a big boost if optimized.

    Regular expressions are handled by python, it means that they may be
    slightly different from those you know from perl or grep. It's not my
    fault in that case.

    Colours are one of:
        none, default, bold, underline, blink, reverse, concealed,
        black, green, yellow, blue, magenta, cyan, white,
        on_black, on_green, on_yellow, on_blue, on_magenta, on_cyan, on_white
        beep
    on_red means that the background (instead of foreground) is painted
    with red etc...

    Additional colours can be: dark, italic, rapidblink, strikethrough.
    These are supported only on some terminals, so if you want to write
    portable configuration files, avoid using them (idea by James Rowe).

    there can be more attributes per line (separated by space), e.g.

    # this is probably a pathname
    regexp=/[\w^\.]+
    colours=bold blink green
    count=more

    will display pathnames in bold blinking green


Hint taken from logcolboriser README:

To have your syslog show on your tty12 in colour, do:
    mkfifo /dev/grc
    replace (or copy and edit) the /etc/syslog.conf line
    *.info;mail.*;authpriv.*;kern.*;local1.* |/dev/xconsole
    with :
    *.info;mail.*;authpriv.*;kern.*;local1.* |/dev/grc
    and add to your syslog startup script :
    grcat conf.log < /dev/grc >/dev/tty12 &

Well, simpler approach seems to be to use something like this
in your system startup script, if you have GNU tail:
    tail --follow=name /var/log/syslog | grcat conf.log >/dev/tty12
or, if you have recent BSD tail:

    tail -F /var/log/syslog | grcat conf.log >/dev/tty12
```