NAME

procmailsc − procmail weighted scoring technique

SYNOPSIS

[*****] **wˆx condition**

DESCRIPTION

In addition to the traditional true or false conditions you can specify on a recipe, you can use a weighted scoring technique to decide if a certain recipe matches or not. When weighted scoring is used in a recipe, then the final score for that recipe must be positive for it to match.

A certain condition can contribute to the score if you allocate it a ‘weight’ (**w**) and an ‘exponent’ (**x**). You do this by preceding the condition (on the same line) with:

wˆx

Whereas both **w** and **x** are real numbers between -2147483647.0 and 2147483647.0 inclusive.

Weighted regular expression conditions

The first time the regular expression is found, it will add *w* to the score. The second time it is found, *w*x* will be added. The third time it is found, *w*x*x* will be added. The fourth time *w*x*x*x* will be added. And so forth.

This can be described by the following concise formula:

$$w * \sum_{k=1}^{n} x^{k-1} = w * \frac{x^n - 1}{x - 1}$$

It represents the total added score for this condition if **n** matches are found.

Note that the following case distinctions can be made:

x=0 Only the first match will contribute w to the score. Any subsequent matches are ignored.

x=1 Every match will contribute the same w to the score. The score grows linearly with the number of matches found.

0<x<1 Every match will contribute less to the score than the previous one. The score will asymptotically approach a certain value (see the **NOTES** section below).

1<x Every match will contribute more to the score than the previous one. The score will grow exponentially.

x<0 Can be utilised to favour odd or even number of matches.

If the regular expression is negated (i.e., matches if it isn’t found), then **n** obviously can either be zero or one.

Weighted program conditions

If the program returns an exitcode of EXIT_SUCCESS (=0), then the total added score will be **w**. If it returns any other exitcode (indicating failure), the total added score will be **x**.

If the exitcode of the program is negated, then, the exitcode will be considered as if it were a virtual number of matches. Calculation of the added score then proceeds as if it had been a normal regular expression with **n=‘exitcode’** matches.

Weighted length conditions

If the length of the actual mail is **M** then:

```
* w^x  > L
```

will generate an additional score of:

$$w * \left| \frac{M}{L} \right|^{x}$$

And:

```
* w^x  < L
```

will generate an additional score of:

$$w * \left| \frac{L}{M} \right|^{x}$$

In both cases, if L=M, this will add w to the score. In the former case however, larger mails will be favoured, in the latter case, smaller mails will be favoured. Although x can be varied to fine-tune the steepness of the function, typical usage sets x=1.

MISCELLANEOUS

You can query the final score of all the conditions on a recipe from the environment variable **$=**. This variable is set *every* time just after procmail has parsed all conditions on a recipe (even if the recipe is not being executed).

EXAMPLES

The following recipe will ditch all mails having more than 150 lines in the body. The first condition contains an empty regular expression which, because it always matches, is used to give our score a negative offset. The second condition then matches every line in the mail, and consumes up the previous negative offset we gave (one point per line). In the end, the score will only be positive if the mail contained more than 150 lines.

```
:0 Bh
* -150^0
*    1^1  ^.*$
/dev/null
```

Suppose you have a priority folder which you always read first. The next recipe picks out the priority mail and files them in this special folder. The first condition is a regular one, i.e., it doesn't contribute to the score, but simply has to be satisfied. The other conditions describe things like: john and claire usually have something important to say, meetings are usually important, replies are favoured a bit, mails about Elvis (this is merely an example :-) are favoured (the more he is mentioned, the more the mail is favoured, but the maximum extra score due to Elvis will be 4000, no matter how often he is mentioned), lots of quoted lines are disliked, smileys are appreciated (the score for those will reach a maximum of 3500), those three people usually don't send interesting mails, the mails should preferably be small (e.g., 2000 bytes long mails will score -100, 4000 bytes long mails do -800). As you see, if some of the uninteresting people send mail, then the mail still has a chance of landing in the priority folder, e.g., if it is about a meeting, or if it contains at least two smileys.

```
:0 HB
*       !^Precedence:.*(junk|bulk)
* 2000^0   ^From:.*(john@home|claire@work)
* 2000^0   ^Subject:.*meeting
*  300^0   ^Subject:.*Re:
* 1000^.75 elvis|presley
* -100^1   ^>
*  350^.9  :-\)
* -500^0   ^From:.*(boss|jane|henry)@work
* -100^3   > 2000
priority_folder
```

If you are subscribed to a mailinglist, and just would like to read the quality mails, then the following recipes could do the trick. First we make sure that the mail is coming from the mailinglist. Then we check if it is from certain persons of whom we value the opinion, or about a subject we absolutely want to know everything about. If it is, file it. Otherwise, check if the ratio of quoted lines to original lines is at most 1:2. If it exceeds that, ditch the mail. Everything that survived the previous test, is filed.

```
:0
^From mailinglist-request@some.where
{
 :0:
 * ^(From:.*(paula|bill)|Subject:.*skiing)
 mailinglist

 :0 Bh
 *  20^1 ^>
 * -10^1 ^[^>]
 /dev/null

 :0:
 mailinglist
}
```

For further examples you should look in the **procmailex**(5) man page.

CAVEATS

Because this speeds up the search by an order of magnitude, the procmail internal egrep will always search for the leftmost *shortest* match, unless it is determining what to assign to **MATCH**, in which case it searches the leftmost *longest* match. E.g. for the leftmost *shortest* match, by itself, the regular expression:

.* will always match a zero length string at the same spot.

.+ will always match one character (except newlines of course).

SEE ALSO

procmail(1), **procmailrc**(5), **procmailex**(5), **sh**(1), **csh**(1), **egrep**(1), **grep**(1),

BUGS

If, in a length condition, you specify an **x** that causes an overflow, procmail is at the mercy of the **pow**(3) function in your mathematical library.

Floating point numbers in ‘engineering’ format (e.g., 12e5) are not accepted.

MISCELLANEOUS

As soon as ‘plus infinity’ (2147483647) is reached, any subsequent *weighted* conditions will simply be skipped.

As soon as ‘minus infinity’ (-2147483647) is reached, the condition will be considered as ‘no match’ and the recipe will terminate early.

NOTES

If in a regular expression weighted formula **0<x<1**, the total added score for this condition will asymptotically approach:

$$\frac{w}{1-x}$$

In order to reach half the maximum value you need

$$n = \frac{-\ln 2}{\ln x}$$

matches.

AUTHORS

Stephen R. van den Berg
<srb@cuci.nl>
Philip A. Guenther
<guenther@sendmail.com>