NAME

pattern — exhaustively locate and count user-defined patterns in a Humdrum input

SYNOPSIS

pattern [**-ciy**] **-f** *templatefile* [**-s** *regexp*] [*inputfile* ...]

DESCRIPTION

The **pattern** command is used to locate all occurrences of a user-defined pattern in some Humdrum input. The patterns sought may span innumerable data records. Occurrences of the pattern are identified in the output by line number.

The pattern sought must be defined as a separate "template" file. The template file is identified using the **-f** command option.

Pattern templates consist of one or more records. Each record specifies a standard UNIX regular expression — followed by an optional "record-count metacharacter." The input is scanned from beginning to end, in order to find passages that match the defined template. In the simplest case, a match is deemed to take place when successive records in the input stream match the regular expressions given in each of the corresponding records of the template. However, the number of records in the matching input need not be the same as the number of records in the template.

Consider, first, a simple example where the template consists of the numbers 1, 2, 3 — each on a separate line:

```
1
2
3
```

This template will match inputs such as the following:

```
1
112
43.9
```

or even

```
0x.=%&1*
Figure 32
abc(...32...)
```

A more circumspect regular expression template might look like this:

```
^1$
^2$
^3$
```

(The caret (^) and dollar sign ($) are regular expression *anchors* that indicate the beginning of the record and end of the record respectively.)

Standard regular expression syntax provides three "counting" metacharacters that can be used to specify the number of occurrences of a given pattern on a single line. The counting metacharacters are + * and ?. If *p* is a regular expression pattern, then (*p*)+ will match one or more consecutive instances of *p*. Similarly, (*p*)* will match zero or more consecutive instances of *p*, whereas (*p*)? will match zero or one instance of *p*. The use of these metacharacters is illustrated below:

X+	matches X, XX, XXX, etc.
X*	matches X, XX, XXX, etc. as well as the null string
X?	matches X or the null string
XX	matches XX
(XX)+	matches XX, XXXX, XXXXXX, etc.

Regular expression counting metacharacters

These metacharacters can be used in conjunction with other regular expression operators and anchors to specify complex patterns. See **regexp (6)** for further details.

In the **pattern** command, the regular expression counting metacharacters may *also* be used to specify the *number of successive records* that match the regular expression. We refer to this use as "record-count metacharacters." Record-count metacharacters are specified by following the regular expression with a tab — followed by either +, *, or ?. For example, consider the following **pattern** template:

```
X	+
Y	*
Z	?
```

The intervening tab characters are important here. They indicate that the metacharacters refer to the number of records rather than to the number of patterns in a given record. The first template record (X<*tab*>+) will match one or more lines containing the letter X. The second template record (Y<*tab*>*) will match zero or more lines containing the letter Y. The third template record (Z<*tab*>?) will match zero or one line containing the letter Z. The above template would match an input such as the following: three successive lines containing the letter X, followed by eight successive lines containing the letter Y, followed by a single line containing the letter Z. Similarly, the above template would also match a one-line input containing the letter X.

Note that the strings <*tab*>+ <*tab*>*, and <*tab*>?, are recognized by **pattern** as record-count metacharacters *only* if they appear at the end of a regular expression.

The **pattern** command will identify *all* possible matching patterns beginning at each point in the input. Consider, by way of example, the following template file (named `template`):

```
1   +
2   +
3   +
4   +
5   +
```

The following Humdrum input file is named `example1`:

```
**num    **num
1        1
2        2
3        2
4        3
5        4
5        5
6        6
*-       *-
```

Given the command:

```
pattern -f template example1
```

the **pattern** command will produce the following output:

```
4 patterns found from line 2 to line 7 of file example1
1 pattern found from line 2 to line 6 of file example1
```

The patterns are: 1-2-3-4-5, 1-2-2-3-4-5, 1-2-3-3-4-5, 1-2-3-4-4-5 and 1-2-3-4-5-5. Note that the entire input line is used for matching purposes. It doesn't matter, for example, whether the number "2" is matched in the left spine or the right spine — only that the number "2" is present on a given line. This feature is useful for identifying *Klangfarbenmelodie* and other "threaded" or "diagonal" patterns that can be traced between spines. If the user wishes to avoid such diagonal patterns, individual spines should be extracted separately before invoking the **pattern** command.

OPTIONS

The **pattern** command supports the following options:

-c	makes pattern-matching sensitive to comments
-h	displays a help screen summarizing the command syntax
-i	makes pattern-matching sensitive to interpretations
-s *regexp*	skip (ignore) data records containing the defined regular expression
-y	outputs appropriate 'yank' commands in place of regular output

Options are specified in the command line.

By default, the **pattern** command is insensitive to the presence or absence of Humdrum comments and interpretations. Pattern searches may be made sensitive to occurrences of comments (defined in the template) by specifying the **-c** option. Similarly, pattern searches

may be made sensitive to occurrences of interpretations by specifying the **-i** option.

Certain types of data records may be ignored in the pattern-search by invoking the **-s** (skip) option. This option must be accompanied by a user-define regular expression. All input data records matching the regular expression are ignored. This option is useful, for example, in skipping null data tokens, barlines, marked embellishment tones, or other types of data.

The **pattern** command does not directly implement an "echo" option — such as provided by the related **patt** command. With the **-y** option, however, **pattern** will produce an output suitable for use with the Humdrum **yank** command. This permits the user to extract the appropriate matching passages from the input.

EXAMPLES

For additional examples pertinent to the **pattern** command, refer to the EXAMPLES section in the documentation for the related **patt** command.

Note that in the above example, the extensive capabilities for defining complex regular expressions have not been used. Refer to **regexp (6)** for further pertinent information.

PORTABILITY

DOS 2.0 and up, with the MKS Toolkit. OS/2 with the MKS Toolkit. UNIX systems supporting the *Korn* shell or *Bourne* shell command interpreters, and revised *awk* (1985).

SEE ALSO

grep (UNIX), **egrep** (UNIX), **patt** (4), **regexp** (4), **regexp** (6), **simil** (4)

WARNINGS

If a comment is present in the template pattern, failing to specify the **-c** option will make pattern matching a logically impossibility.

NOTE

The **pattern** command differs from the related **patt** command in the following ways: (1) **patt** always produces output conforming to the Humdrum syntax whereas **pattern** never produces Humdrum output. (2) **patt** does not support multi-record 'wild cards' in the template file, and so limits the sophistication of the regular expressions. (3) The **patt** command provides "echo" and "tag" options.