NAME

patt — locate and output user-defined patterns in a Humdrum input

SYNOPSIS

patt [**-m**] **-f** *templatefile* [**-s** *regexp*] [**-t** *output_tag*] [*inputfile* ...]

DESCRIPTION

The **patt** command is used to locate occurrences of a user-defined pattern in some Humdrum input. The patterns sought may span innumerable data records. Occurrences of the pattern may be identified by line number, echoed intact in the output stream, or tagged by a user-defined marker in a `**patt` output spine.

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 regular expression pattern. The input is scanned from beginning to end, in order to find passages that match the defined template. In order for a match to take place, successive records in the input stream must match the regular expressions given in each of the corresponding records of the template. For example, if the template is 9 lines in length, then the input stream must contain 9 successive lines of matching data in order for a pattern match to be successful.

The **patt** command implements a full UNIX regular expression syntax. Each line in the template file represents an independent regular expression. For example, the template:

```
1
2
3
```

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.) Refer to *regexp (6)* for further information regarding regular expressions. Note that the related **pattern** command implements a more powerful regular expression syntax for defining multi-record patterns.

OPTIONS

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

-c	makes pattern-matching sensitive to comments
-e	echoes matched patterns in the output
-f *templatefile*	use pattern specified in *templatefile*
-h	displays a help screen summarizing the command syntax
-m	invokes collapsed multiple-record matching mode
-s *regexp*	skip (ignore) data records containing the defined regular expression
-t *output_tag*	generate `**patt` output spine; tag each occurrence of the pattern with the string *output_tag*

Options are specified in the command line.

By default, the **patt** command is insensitive to the presence or absence of Humdrum comments. Pattern searches may be made sensitive to occurrences of comments by specifying the **-c** option.

In the default operation, **patt** outputs a Humdrum global comment for each pattern matched in the input. Each comment identifies the line number in the input where the found pattern begins.

With the **-e** option, each instance of the found pattern is echoed in the output. Each output pattern is preceded by the appropriate exclusive interpretation(s) and followed by appropriate spine-path terminator(s).

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 **-m** option invokes a multiple record matching mode. In this mode, **patt** attempts to match as many successive regular expressions in the template file as possible for a given input record, before continuing with the next input and template records. In this way, a template file consisting of several records, may possibly match a single input record. (See EXAMPLES below.)

The **-t** option causes **patt** to output a single spine of `**patt` output. The user specifies an *output tag* (character string) on the command line. Each instance of the found pattern causes

the tag to be output in the `**patt` spine at the position corresponding to the onset of each found pattern. (See EXAMPLES below.) Note that the **-t** and **-e** options are mutually exclusive.

Whatever options are invoked, **patt** always produces output that conforms to the Humdrum syntax.

EXAMPLES

The following examples illustrate the operation of the **patt** command. Consider the following target file and pattern template file:

template file:

```
1
2
3
```

target file:

```
**foo
1
2
=1
1 3
2 1
3 2
3
=2
1
2 3
4
2 3 1
*-
```

A simple search for the pattern template might use the command:

```
patt -f template target
```

Pattern matches are announced by outputting Humdrum global comments. Given the above command, the following output would result:

```
!! Pattern found at line 5 of file input
!! Pattern found at line 6 of file input
```

In the first instance, notice that **patt** is able to identify overlapping patterns. Two instances of the 1-2-3 pattern are identified beginning on the fifth and sixth lines respectively.

Note however, that the first instance of the pattern (beginning at line 2) was not identified due to the interruption of the common system barline in the fourth line. The barlines can be

ignored by invoking the **-s** option, followed by a regular expression that uniquely identifies the records to be skipped — in this case the equals sign. The command:

```
patt -s = -f template target
```

would produce the following output:

```
!! Pattern found at line 2 of file input
!! Pattern found at line 5 of file input
!! Pattern found at line 6 of file input
```

Actual instances of the pattern can be output by invoking the **-e** (echo) option:

```
patt -e -s = -f template target
```

The following output would result:

```
!! Pattern found at line 2 of file input
**foo
1
2
=1
1 3
*-
!! Pattern found at line 5 of file input
**foo
1 3
2 1
3 2
*-
!! Pattern found at line 6 of file input
**foo
2 1
3 2
3
*-
```

Notice that each pattern found is output as a self-contained Humdrum output with initial exclusive interpretations and concluding spine-path terminators. If a single continuous output is desired, the **rid -u** command may be used to eliminate the duplicate interpretations.

Instead of outputting the individual patterns, the **-t** option may be used to output a spine that marks each instance of the found pattern. In the following command, the beginning of each occurrence of the pattern is labelled in the `**patt` spine by the tag "one-two-three."

```
patt -t one-two-three -s = -f template target
```

The follow output would result:

```
**foo    **patt
1         one-two-three
2         .
=1        .
1 3       one-two-three
2 1       one-two-three
3 2       .
3         .
=2        .
1         .
2 3       .
4         .
2 3 1     .
*-        *-
```

For some tasks (such as the identification of tone-rows in 12-tone music), nominally "successive" elements of the pattern may be collapsed within a single sonority or record. The **-m** option invokes a *multiple record matching* mode. By way of example, the following command:

```
patt -m -t theme123 -s = -f template target
```

will produce the following output:

```
**foo    **patt
1         theme123
2         .
=1        .
1 3       theme123
2 1       theme123
3 2       .
3         .
=2        .
1         theme123
2 3       .
4         .
2 3 1     theme123
*-        *-
```

Note that in the above examples, 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), **pattern** (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 **patt** command differs from the related **pattern** command in the following ways: (1) **patt** always produces output conforming to the Humdrum syntax whereas **pattern** never produces Humdrum output. (2) **pattern** allows multi-record 'wild cards' in the template file, and so permits the creation of more sophisticated regular expressions. (3) The **pattern** command does not directly provide an "echo" option.