NAME

fill — replace null tokens with previous data token

SYNOPSIS

fill [**-c** *chars*] [**-p**] [**-s** *regexp*] [*inputfile* ...]

DESCRIPTION

The **fill** command accepts any Humdrum input and replaces each null token with the previous non-null data token in the same spine.

Humdrum null-tokens are place-holders that do not themselves encode data. Null-tokens consist of a single period character (".") — separated from other tokens by tabs, or appearing on a line by itself. The **fill** command replaces occurrences of null-tokens with the most recent non-null data occurring in the same spine. When the **-p** option is invoked, the replacement data tokens are enclosed in parentheses ( ). If the initial data tokens in a spine are null-tokens, then null-tokens (.) are output.

In repeating previous data tokens, if the **-s** option is invoked, **fill** skips over any data records matching *regexp*. For example, if *regexp* is the equals-sign (the "common system" barline), then barline data tokens will not be repeated in subsequent data records containing null tokens. Thus, if a data token 'X' is followed by a token that matches the regular expression /=/, then subsequent null-tokens will be replaced by the token 'X' rather than by the equals sign.

The **fill** command correctly handles spine path changes such as exchange-path indicators (*x), join-path indicators (*v), split-path indicators (*^), terminate-path indicators (*-), and begin-spine (*+). In the case where two spines join together, **fill** outputs a double-stop where necessary.

OPTIONS

The **fill** command provides the following options:

-h	displays a help screen summarizing the command syntax
-c *chars*	repeats only characters listed in *chars*
-p	place repeated data tokens in parentheses
-s *regexp*	skip data records matching *regexp*

Options are specified in the command line.

EXAMPLES

The following inputs and outputs illustrate the operation of the **fill** command. Consider the following input:

```
!! Example 1
**kern    **kern
16e-      8r
16d       .
16e-      8gg
16f       .
16g       8cc
16f       .
16g       8gg
16e-      .
16a       [2aa
16g       .
16a       .
16b-      .
16cc      .
16b-      .
16cc      .
16a       .
=78       =78
.         .
*-        *-
```

Invoking the command:

```
fill input > output
```

produces the following output:

```
!! Example 1
**kern    **kern
16e-      8r
16d       8r
16e-      8gg
16f       8gg
16g       8cc
16f       8cc
16g       8gg
16e-      8gg
16a       [2aa
```

```
16g       [2aa
16a       [2aa
16b-      [2aa
16cc      [2aa
16b-      [2aa
16cc      [2aa
16a       [2aa
=78       =78
=78       =78
*-        *-
```

Notice that all of the null tokens have been replaced by the preceding data token in the same spine. Notice also that the barline for measure 78 has been repeated. For many applications repeating of barlines will be inappropriate.

The following, more complex example, illustrates the use of the **-p** and **-s** options. The input is shown on the left and the corresponding output is shown on the right:

INPUT

```
!! Example 2
**foo    **foo    **bar
a        xyz      .
.        23       (%&)
=2       =2       =2
.        .        .
!! A comment.
.        .        49
*x       *        *x
.        .        .
*        *v       *v
.        .
abc      XYZ
*        *^
.        .        .
.        1a       2b
=3       =3       =3
*-       *        *
.        .
====     ====
*+       *
**foo    **foo    **bar
.        .        .
*-       *-       *-
```

OUTPUT

```
!! Example 2
**foo    **foo    **bar
a        xyz      .
(a)      23       (%&)
=2       =2       =2
(a)      (23)     ((%&))
!! A comment.
(a)      (23)     49
*x       *        *x
(49)     (23)     (a)
*        *v       *v
(49)     (23 a)
abc      XYZ
*        *^
(abc)    (XYZ)    (XYZ)
(abc)    1a       2b
=3       =3       =3
*-       *        *
(1a)     (2b)
====     ====
*+       *
**foo    **foo    **bar
(1a)     .        (2b)
*-       *-       *-
```

The output was produced by invoking the following command:

```
fill -p -s ^= input > output
```

In order to avoid repeating the barlines, the skip option has been invoked with the regular expression "^=" — meaning any equals sign at the beginning of a line. (See **regexp** in Section 6 of this manual for details concerning regular expression syntax.) In addition, the **-p** option has been invoked so that all repeated tokens are placed in parentheses. Notice that **fill** adapts to changing spine-paths. Note especially the join-spine (*v) interpretations leading to the double-stop: `(23 a)`.

A final example illustrates the use of the **-c** option. Once again, the input is shown on the left and the corresponding output is shown on the right. The output was produced by invoking the following command:

```
fill -c '[a-gA-G#-]' input > output
```

INPUT		OUTPUT	
**kern	**kern	**kern	**kern
(4g	8b	(4g	8b
.	8cc	g	8cc
8f#	4dd	8f#	4dd
4.g)	.	4g)	dd
.	8cc	g	8cc
.	8b	g	8b
4d	4a	4d	4a
.	.	d	a
*-	*-	*-	*-

The effect of this command has been to propagate the `**kern` pitch signifiers, without propagating any non-pitch 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

patt (4), **pattern** (4), **regexp** (4), **regexp** (6), **simil** (4)