NAME

match - pattern matcher

SYNOPSIS

```
char    *ppcursor;
char    *ppdot;

int match(patptr,progarg0,progarg1,...)  /* old interface */
    PPAT *patptr;
    PPROGARG progarg0;
    PPROGARG progarg1;
      .
      .
      .
```

DESCRIPTION

Ppmatch and **match** provide two ways to call the common pattern package **pattern matcher.** In general a pattern matcher takes a pattern and one (or more) strings and determines if the pattern matches the string(s). The common pattern package pattern matcher preforms this function and several other functions to include:

1) Pattern matching on one or more strings given in the progarg array as determined by the **switch** built-in pattern.

2) Return an integer value as specified by the **succ** built-in pattern.

3) Mark one or more positions in any of the strings provided by the **dot** and **mdot** built-in patterns.

4) Provide the addresses of one or more pieces of the string or pattern in a user supplied buffer (specified by ppmdot(3L)) as a first step in reformating one or more strings using the **startfld, endfld** and **deffld** built-in patterns.

The arguments to **match**() are as follows:

patptr is a pointer to the pattern to be used by the matcher.

progarg0,progarg1,... are application defined inputs. The first element (**patarg0**) must point to the start of the first text-area. All other elements may point to any valid program argument type as defined in the **<ppsubs.h>** header file.

Ppmatch and match never change anything pointed to by their arguments.

Ppmatch and match sets the value of several external variables as described below.

ppcursor - contains the value of the matcher cursor (pointer to first text-area) at the time the matcher returned. In the old version of the pattern matcher **cursor** was used instead of **ppcursor** For upward compatibility purposes **cursor** is equivalent to **ppcursor**

ppdot - is set to the current cursor position when a **dot** built-in pattern is encountered in the pattern. If no **dot** built-in pattern is encountered, Then the value of ppdot is not changed. In the old version of the pattern matcher **dot** was used instead of **ppdot** For upward compatibility purposes **dot** is equivalent to **ppdot**

The first element (zero subscript) of the **patarg** array (and patarg0 in match()) should be a text-area. This element is used to initialize the matcher cursor (pointer to the text-area being pattern matched). A **switch** keyword in the pattern may change the text-area being pattern matched (as well as the pattern). Therefore, the use of a **switch** keyword in the pattern may require additional text-areas which must have pointers (to them) included in the array. The index of the pointer in the array corresponds to the number argument in the **switch** keyword. For example the keyword **switch(2,arb 'aaa')** requires progarg[2] to be a pointer to a text-area.

Ppmatch and match returns one of the integer values described below:

PPSUCCESS - indicates a successful match

PPABORT - indicates an unsuccessful match

PPUNDEFKEY - indicates a zero value primitive was found in the pattern. This indicates that the pattern has been scribbled (or is not a pattern).

n - where **n >= 0**; and **n** is the value of a **succ** built-in pattern argunment which is encountered by ppmatch and match

SEE ALSO

ppchkpat(3L), ppmatch(3L), ppsmdot(3L)

DIAGNOSTICS

Ppmatch and match produces no diagnostics except that a **PPUNDEFKEY** value will be returned when a zero value primitive is discovered in the pattern (zero is an invalid primitive value).

BUGS

Ppmatch and match do not check the pattern or the elements of **progarg** If any of their values are improper, then unpredictable/terrible things may occur (e.g., trying to execute instructions in data or stack space). To avoid some of the posible problems **ppchkpat(3L)** should be used.