NAME

amatch − look for anchored match of regular expression

SYNOPSIS

#include "mjsu.h"

BOOL amatch(CHAR *buf, UINT n, UINT idx, UINT *nmindx,
USHORT *pat, TAGMATCH *psubs);

DESCRIPTION

amatch() tests the *n* character buffer starting at *buf*[*idx*] for a match with the encoded pattern starting at *pat*; the match is constrained to match characters starting at *buf*[*idx*].

It is assumed that the encoded pattern was built by the function **pattern()**, whose manual page describes the notation for regular expressions accepted by these routines.

The type TAGMATCH is a structure declared in the header file **mjsu.h**, thus:

```
typedef struct
        {
        UINT mlen;
        CHAR *mtext;
        } TAGMATCH;
```

If (*psubs* is not NULL) then every balanced pair \(. . . \) within the pattern will have the substring it matches recorded at *psubs*[*i*], where *i* counts up from one for the leftmost "\(" in the pattern. Upto 10 such "tagged expressions" are acceptable; surplus ones are ignored. *psubs*[*i*].*mtext* points at the first character of the matching substring, and *psubs*[*i*].*mlen* is its length. *psubs*[0] always records the full match.

RETURNS

If the match was successful, **amatch()** returns YES and sets **nmidx* to the index of the first character in buf that was not matched. Otherwise **amatch()** returns NO. The array at psubs is also filled in, if present.

EXAMPLE

To find input lines starting with a pattern specified by the first command line argument, and output matching lines to stdout:

```
if (pattern(patbuf, '\0', *av))
        {
        while (fgets(buf, BUFSIZ, stdin))
                {
                if (amatch(buf, n, 0, &junk, patbuf, NULL))
                        fputs(buf, stdout);
                }
        }
```

SEE ALSO

match(3), **pattern(3)**, **mjsu(7)**

AVAILABILITY

amatch() is written in C, conforming to ANSI X3.159-1989.

BUGS

Closures (regular expression of "*") will not work properly unless *buf* is terminated by a newline.