NAME
nsr_getdate − convert time and date from ASCII

SYNOPSIS
#include <sys/types.h>

#include <sys/types.h>

time_t nsr_getdate(buf)
char *buf;

DESCRIPTION
The nsr_getdate() routine converts most common time specifications to standard UNIX format. It takes a character string containing time and date as an argument and converts it to a time format.

The character string consists of zero or more specifications of the following form:

tod A tod is a time of day, which is of the form hh[[:mm[:ss]]] (or hhmm) [meridian] [zone]. If no meridian − am or pm − is specified, a 24-hour clock is used. A tod may be specified as just hh followed by a meridian. If no zone (e.g. GMT) is specified, the current timezone, as determined by the second parameter, now, is assumed.

date A date is a specific month and day, and possibly a year. The acceptable formats are mm/dd [yy] and monthname dd[, yy] If omitted, the year defaults to the current year. If a year is specified as a number in the range 70 and 99, 1900 is added. If a year is in the range 00 and 30, 2000 is added. The treatment of other years less than 100 is undefined. If a number not followed by a day or relative time unit occurs, it will be interpreted as a year if a tod, monthname, and dd have already been specified; otherwise, it will be treated as a tod. This rule allows the output from date(1) or ctime(3) to be passed as input to nsr_getdate.

day A day of the week may be specified; the current day will be used if appropriate. A day may be preceded by a number, indicating which instance of that day is desired; the default is 1. Negative numbers indicate times past. Some symbolic numbers are accepted: last, next, and the ordinals first through twelfth (second is ambiguous, and is not accepted as an ordinal number). The symbolic number next is equivalent to 2; thus, next monday refers not to the immediately coming Monday, but to the one a week later.

relative time
Specifications relative to the current time are also accepted. The format is [number] unit; acceptable units are year, month, fortnight, week, day, hour, minute, and second.

The actual date is formed as follows: first, any absolute date and/or time is processed and converted. Using that time as the base, day-of-week specifications are added; last, relative specifications are used. If a date or day is specified, and no absolute or relative time is given, midnight is used. Finally, a correction is applied so that the correct hour of the day is produced after allowing for daylight savings time differences.

Ns_r_getdate accepts most common abbreviations for days, months, etc.; in particular, it will recognize them with upper or lower case first letter, and will recognize three-letter abbreviations for any of them, with or without a trailing period. Units, such as weeks, may be specified in the singular or plural. Timezone and meridian values may be in upper or lower case, and with or without periods.

SEE ALSO
cftime(3), date(1), ftime(3c), localtime(2), time(2).

BUGS
The grammar and scanner are rather primitive; certain desirable and unambiguous constructions are not accepted. Worse yet, the meaning of some legal phrases is not what is expected; next week is identical to 2 weeks.

The daylight savings time correction is not perfect, and can get confused if handed times between midnight and 2:00 am on the days that the reckoning changes.
Because `localtime(2)` accepts an old-style time format without zone information, passing `nsr_getdate` a current time containing a different zone will probably fail.