How did we end up parsing Savvyday 29 Oatmeal 94 as Saturday 29 October 1994?

devblogs.microsoft.com/oldnewthing/20200929-05

September 29, 2020

Raymond Chen

Some time ago, we learned that the `InternetTimeToSystemTime` function manages to parse "Savvyday 29 Oatmeal 94" as "Saturday 29 October 1994". How did that happen? Is it finding the date with the shortest English Levenshtein distance?

Nothing that fancy.

Warning: This article discusses implementation details, which are not contractual. The algorithm is subject to change in the future. The only thing that `InternetTimeToSystemTime` formally guarantees is that it can parse properly-formatted HTTP timestamps.

The parsing is very simple. The official format for HTTP date strings is

1. DayOfWeek, Day Month Year Hour Minute Second GMT

In practice, not everybody follows the rules, so the parser accepts these three formats:

1. DayOfWeek Day Month Year Hour Minute Second TZ
2. DayOfWeek Month Day Hour Minute Second TZ Year
3. DayOfWeek Month Day Hour Minute Second Year TZ

After discarding non-alphanumerics, the parser takes each word in the input string and converts it to a number somehow. If it consists of digits, then it's parsed to a number in the usual way. If it consists of alphabetics, then it's parsed to a number by trying to match it against the list of valid tokens:

DayOfWeek	Month		TZ
Sun = 0	Jan = 1	Jul = 7	GMT
Mon = 1	Feb = 2	Aug = 8	UTC
Tue = 2	Mar = 3	Sep = 9	

Wed = 3	Apr = 4	Oct = 10	
Thu = 4	May = 5	Nov = 11	
Fri = 5	Jun = 6	Dec = 12	
Sat = 7			

If no match is found, then we look for an entry which shares the most initial characters with the word being parsed. If there is a unique such entry, then the parsed value is as given in the table. If there is no such entry, or the longest match is not unique, then parsing fails.[1]

Since there only one time zone permitted in HTTP time/date strings, all we have to remember is “Yup, it’s a time zone. There’s a time zone marker here.”

For example, the string “Savvyday” is not in the above table, but it does share the following prefixes:

Length 1	**Length 2**
S(un)	Sa(t)
S(ep)	

The longest match is length 2, and there’s only one such match, so the word “Savvyday” is parsed as if it were “Sat”.

Similarly, “Oatmeal” has only one match: Oct (length 1).

After everything is parsed into a number, we decide which of the three formats we are looking at.

If the second word was parsed from digits, then we are in case 1. If the seventh word was parsed from letters, then we are in case 2. Otherwise, we are in case 3.

Once we’ve decided what case we’re in, we know where the year is. If the caller provided a two-digit year, upgrade it to a four-digit year.

Finally, we copy the fields into the output structure. If a field is missing, it is taken from the current date and time.

That’s it. Nothing fancy. The algorithm is optimized for the case where the string follows the correct format. If you pass something that’s not in the correct format, it does what it can. Sometimes it even comes up with something vaguely sensible!

Usually not.[2]

[1] If you think about it, this can be done very quickly by a simple decision tree:

Character			
1	**2**	**3**	**Result**
A	P		Apr
	U		Aug
D			Dec
F	E		Feb
	R		Fri
G			GMT
J	A		Jan
	U	L	Jul
		N	Jun
M	A	R	Mar
		Y	May
	O		Mon
N			Nov
O			Oct
S	A		Sat
	E		Sep
	U		Sun
T	H		Thu
	U		Tue
U			UTC
W			Wed

[2] For example, `Friday Friday Friday Friday Friday Friday Friday Friday Friday` parses to “day 5 of month 5 year 5, hour 5 minute 5, and 5 seconds” or “May 5, 2005 at 05:05:05 GMT”.

Raymond Chen

Follow