Subject: StrToDate(Date& d, const char *s, Date def)
Posted by sapiency on Thu, 07 Jan 2010 18:32:11 GMT
View Forum Message <> Reply to Message

Hi,

this function works nice for a lot of situations, but IMHO it make things it not should do ..

1.1.10 -> 01.01.2010 is OK
1.1.1  -> 01.01.2001 is not OK

I modified it and now it should work as I expect:

```
1.1.10 -> 01.01.2010
1.1.0  -> 01.01.   0
1.1.1  -> 01.01.   1
1.1.01 -> 01.01.2001
1.1.011-> 01.01.  11
```

It is still not possible to insert Dates bevor year "0", but this seems not be supported by the popup too ...

btw. I extended the code to accept 2 digit input for years +20/-80 from current year.

Maybe you find the modification helpfull.

regards and a Happy New Year

reinhard

ps: I set the level for blank to 1, so it is still allowed to insert "1 1 01" to get "01.01.2001"

```
const char *StrToDate(Date& d, const char *s, Date def)
{
 const char *fmt = s_date_scan;
 if(*s == 0) {
  d = Null;
  return s;
 }
 d = Nvl(def, GetSysDate());

 int cc = ( d.year / 100 );
 cc *= 100;
 int oc = cc -100;
 int level = d.year - cc + 20;

 //RLOG( oc << " " << cc << " " << level );
```

```
while(*fmt) {
 bool y2 = false;
 int blank = 0;

 while(*s && !IsDigit(*s) && !IsAlpha(*s) && (byte)*s < 128 )
 {
  if ( 0 == cmp( *s, ' ' ) ) blank++;
  s++;
 }
 int n;
 if(IsDigit(*s)) {
  char *q;
  n = strtoul(s, &q, 10);
  if( 2 == (q-s) ) y2 = true;
  s = q;
 }
 else
 if(IsAlpha(*s) || (byte)*s >= 128) {
  if(*fmt != 'm')
   return NULL;
  String m;
  while(IsAlpha(*s) || (byte)*s >= 128)
   m.Cat(*s++);
  m = ToUpper(m);
  for(int i = 0; i < 12; i++)
   if(m == ToUpper(MonthName(i)) || m == ToUpper(MonName(i))) {
    n = i + 1;
    goto found;
   }
  return NULL;
 found:
  ;
 }
 else
  break;

 switch(*fmt) {
 case 'd':
  if(n < 1 || n > 31)
   return NULL;
  d.day = n;
  break;
 case 'm':
  if(n < 1 || n > 12)
   return NULL;
  d.month = n;
  break;
  case 'y':
   d.year = n;
   if (y2 && ( 2 > blank ) )
   {
    if(d.year < level)
     d.year += cc; // Check again in 2015.... or maybe never ...
    else
     d.year += oc;
   }
   break;
  default:
   NEVER();
  }
  fmt++;
 }
 return d.IsValid() ? s : NULL;
}
```

---

Subject: Re: StrToDate(Date& d, const char *s, Date def)
Posted by mirek on Fri, 08 Jan 2010 16:40:04 GMT
View Forum Message <> Reply to Message

Well, it is not unreasonable, but is not it a bit nitpicking?

Besides, at year 1, current calendar did not even existed yet...

Also, what you expect as "expected" does not really sound expected to me...

I would like to hear somebody's else opinion first..

Mirek

---

Subject: Re: StrToDate(Date& d, const char *s, Date def)
Posted by sapiency on Fri, 08 Jan 2010 21:14:04 GMT
View Forum Message <> Reply to Message

Hi Mirek,

luzr wrote on Fri, 08 January 2010 17:40Well, it is not unreasonable, but is not it a bit nitpicking?

you are right . I noticed the same effect when I checked this input in openoffice - calc. It is not possible to insert a Date before 1.1.1000

luzr wrote on Fri, 08 January 2010 17:40

Besides, at year 1, current calendar did not even existed yet...

Also, what you expect as "expected" does not really sound expected to me...

I would like to hear somebody's else opinion first..

Mirek

That's right, and not many dates are known so detailed from the years before 1582 . And for this it seems to be much more comfortable to use a EditString. I just stumble over this lines of code, when I looked for the reason of the other problem I posted and found it interesting to understand what happens.

Now I'm curious if anybody else is nitpicking too

reinhard

---

## Subject: Re: StrToDate(Date& d, const char *s, Date def)
Posted by koldo on Fri, 08 Jan 2010 21:38:20 GMT

View Forum Message <> Reply to Message

Hello sapiency

Quote:this function works nice for a lot of situations, but IMHO it make things it not should do ..

1.1.10 -> 01.01.2010 is OK
1.1.1 -> 01.01.2001 is not OK

I modified it and now it should work as I expect:

1.1.10 -> 01.01.2010
1.1.0 -> 01.01. 0
1.1.1 -> 01.01. 1
1.1.01 -> 01.01.2001
1.1.011-> 01.01. 11
Well, for me

1.1.1 -> 01.01.2001 is OK
1.1.1 -> 01.01. 1 is not OK
1.1.0 -> 01.01.0 sound logical, but strange
1.1.10 -> 01.01.2010 is OK and does not match with the later... it should have to be
1.1.10 -> 01.01.10, that sound logical, but strange

Perhaps if you are working in something about Roman Empire it sounds clever, but for events happened nowadays, it is not simple to understand.

So I prefer the actual implementation . Sorry  !

Best regards
Koldo