Managing the INTNX Function  
H. Ian Whitlock, Westat

Abstract
What is wrong with the INTNX function? INTNX moves by interval boundaries. Consider
nxtbrtdy = intnx ("year", today(), 1);
This produces January 1, 2000. Y2k compliant, but not exactly what I would like when today is a birthday and I want the next one. The same problem occurs for monthly increments.

What's the answer?
nxtbrtdy = %w_intnx ("year", today(), 1);

What's the code? Come and find out.

When was last Friday? That's simple, but it depends on a trick.

Introduction
The INTNX function moves date and time values by a specified number of boundary points. The first argument gives the type of boundary (year, month, week, etc.), the second gives the starting point, and the third gives the number of boundaries to cross. For example,
intnx ("year", "22may1999", 1)
yields January 1, 2000 (one year boundary from May 1999. On the other hand 12 months ago would be
intnx ("months", "22may1999", -12)
or May 1, 1998. It would be convenient to have function that could move by intervals instead of boundaries. For example, two months from May 22, 1999 should be July 22, not July 1. Well, there is a problem, what is one year from February 29, 2000? There isn't any February 29, 2001. Humans have no problem with this - they would accept February 28, 2000 without hesitation. It is the right month and it is the best day that you are going to get in that month. It is not perfection, but it is the way we think, so it would be convenient to have a SAS function that works this way, or at least a macro that acts like a function.

The Logic
To move years, multiply by 12 and move months instead. To move weeks, multiply by 7 and move days instead. Since moving days is trivial, the problem really reduces to moving months.

Consider

\[
\text{intnx('month',}&\text{,}&\text{&date,1)}
\]

This moves one month boundary, i.e. to the first of the next month. Now to get to the correct day we can add

day ( &date )

if the month is long enough, and then subtract one because the first of the month is day 1 not zero. When the month is short, we have over shot the correct day. In this case we want

\[
\text{intnx('month',}&\text{,}&\text{&date,&int+1)}&\text{ }&\text{ }&\text{ }&\text{ }&\text{ }&\text{-1)}
\]
i.e. go to the next month boundary and back up one to get the end of the month. Now how can one decide which formula is the correct one? Well the second formula always yields then end of the month and the first yields the correct day when the month is long enough and a day past the end of the month when it is not. Hence we can always choose the earlier date, but this means simply applying the MIN function to the two formulas. Hence the formula to move &NINT months from &DATE is

\[
\text{min(intnx('month',}&\text{,}&\text{&date,&int)}
\]

+ day ( &date ) + 1,

\[
\text{intnx('month',}&\text{,}&\text{&date,&int+1)}&\text{ }&\text{ }&\text{ }&\text{ }&\text{ }&\text{-1)}
\]

With this problem solved we are ready to look at the macro.

The Macro
/ * move day by specified interval (Feb29 + yr- Feb29)* /
%
macro w_intnx ( interval , / "Month"/

   date , / "from date "/

   int / "#intervals "/

) ;
/ * -----------------------------------------------

MODULE: W_INTNX

PURPOSE: The DATA step function

   INTNX(interval, date, int) moves forward int intervals from date stopping at a boundary point. For example

   intnx('year',"'12dec1998'd,1)

   yields 'jan1999'd. In contrast

   %w_intnx(year, "'12dec1998'd,1)

   yields the more intuitive

   '12dec1999'd.

CLASS: Acts as a DATA step function.
usage: data _null_;
dt = '31 jan 1998'd;
next = wintnx(month, dt, 1);
put next= dt=; 
run;
produces next=28feb1998

parameters: R INTERVAL year, month, week, and
day. Value may be in
quotes, but it need not
be.
R DATE SAS date given as date
literal, variable, or
expression.
R NINT Number of intervals to
move forward (positive) or
backward (negative)

side effects: None

notes: Moving forward months follows the natural
concept of month, i.e. the next month date
falls on the same day of the next month or
the last day of the next month when the month
is too short.

document: Here.
support: Ian Whitlock <whiloit@westat.com>

"/
*/
/* ---- get rid if quotes when present ---- */\n%if %qsubstr(&interval,1,1) = %str(')
or %qsubstr(&interval,1,1) = %str(')
%then %let interval =
%substr(&interval,2,length(&interval)-2);

%if %upcase (&interval ) = YEAR %then
%let nint = %eval ( 12 * &nint ) ;
%if %upcase (&interval ) = MONTH
or %upcase (&interval ) = YEAR
%then
min(intnx('month',&date,&nint) +
day (&date) - 1,
intnx('month',&date,&nint+1) - 1 ) ;
%else
%if %upcase(&interval) = WEEK %then
&date + 7 * &nint ;
%else
%if %upcase(&interval) = DAY %then
intnx ( 'day', &date , &nint ) ;
%else
%put
ERROR(W_IntNx): Parm error
interval=&interval date=&date nint=&nint ;
%mend w_intnx ;

when was last friday?

when the first parameter of intnx is "week", one can
add ".6" to indicate weeks as starting on Fridays. Hence
intnx(\"week.6\",&date,0)
should take one to the Friday of the week containing the
date, &DATE. The problem comes when &DATE is a
Friday. In this case the function yields the given Friday
instead of the previous Friday as one expects in the
expression "last Friday". Warren Repole pointed out a
simple solution when I provided a more complex answer to
this question. His formula was

intnx(\"week.6\",&date-.05,0)

by subtracting just a little bit it is no longer Friday. What
about other days? Well in that case it doesn't matter.
Actually one could subtract one, but I will stick to Waren's
.05 to soothe my conscience for having missed the point.

conclusion

many "sas programmers" miss many opportunities by not
storing dates as SAS dates. SAS is very rich in the
manipulation of SAS dates and life gets easier when one
learns the tools for manipulating SAS dates. It also helps
greatly, in avoiding Y2K problems. This paper has worked
with one of those functions, the intnx function to provide
a macro programming tool for handling a common
expression such as "The next meeting will be three
months from today."

the author may be contacted by e-mail at
whilo1@westat.com
or by mail at
Ian Whitlock (RA1356)
1650 Research Boulevard
Rockville, MD 20850
Questions are welcome.

SAS is a registered trademark or trademark of SAS
Institute Inc. in the USA and other countries.® indicates
USA registration.