Inctime Documentation

Release V0.01

João Gerd Zell de Mattos

Nov 14, 2019

Contents

1 Main routine and modules **3**

1.1 Main 3

2 Usage **15**

2.1 Usage 15

2.2 Indices and tables 17

Fortran Module Index **19**

Index **21**

Inctime is a Fortran program to calculate dates. It can convert between date formats and output in a variety of ways, being suitable to use within fortran based modeling applications.

CHAPTER 1

Main routine and modules

1.1 Main

The following code is related to the inctime core.

1.1.1 Inctime

Inctime is the main program.

Program

`program inctime`

Main program

Package Overview Inctime is a set of routines in fortran 90 to perform calculations with dates; The main function is to calculate dates to past or to future from any date. Other operations can be performed by using the modules available in this routine. Among other duties, they are calculating the Julian day, number of years, months, days and hours between any two dates.

Autor João Gerd Zell de Mattos

Affiliation Group on Data Assimilation Development - CPTEC/INPE

Date March 22, 2011

Usage The way to use this program is through the command line as follows:

inctime [yyyymmddhh, yyyymmdd] [<+,->nynmndnhnnns] [Form. output]

where

- **[yyyymmddhh, yyyymmdd]**
  - Initial Time

- **[<+,->nynmndnhnnns]**
  - (-) calculate the passed date
  - (+) calculate the future date (default)
  - (ny) Number of year (default is 0)
  - (nm) Number of months (default is 0)
  - (nd) Number of days (default is 0)
  - (nh) Number of hours (default is 0)
  - (nn) Number of minutes (default is 0)
  - (ns) Number of seconds (default is 0)
- **[Form. Output]**
  - Format to output date. is a template Format The format descriptors are similar to those used in the GrADS.

  - "%y4" substitute with a 4 digit year
  - "%y2" a 2 digit year
  - "%m1" a 1 or 2 digit month
  - "%m2" a 2 digit month
  - "%mc" a 3 letter month in lower cases
  - "%Mc" a 3 letter month with a leading letter in upper case
  - "%MC" a 3 letter month in upper cases
  - "%d1" a 1 or 2 digit day
  - "%d2" a 2 digit day
  - "%h1" a 1 or 2 digit hour
  - "%h2" a 2 digit hour
  - "%h3" a 3 digit hour (?)
  - "%n2" a 2 digit minute
  - "%e" a string ensemble identify
  - "%jd" a julian day without hours decimals
  - "%jdh" a julian day with hour decimals
  - "%jy" a day of current year without hours decimals
  - "%jyh" a day of current year with hours decimals

Can use words to compose the output format.

Examples

- inctime 2001091000 +1d %d2/%m2/%y4
- inctime 2001091000 +48h30n %h2Z%d2%MC%y4
- inctime 2001091000 -1h30n 3B42RT.%y4%m2%d2%h2.bin

- inctime 2001091000 -2h45n 3B42RT.%y4%m2%d2%h2.bin
- inctime 2001091000 -1y3m2d1h45n ANYTHING.%y4%m2%d2%h2.ANYTHING

History

- 22 Mar 2011 - Joao Gerd - Initial Code
- 27 Jul 2011 - Joao Gerd - Bug into the end print
- 19 Jul 2012 - Joao Gerd - Add option to month increment/decrement
- 15 Apr 2013 - Joao Gerd - correct bug to incr/decr month
- **03 May 2013 - Joao Gerd - correct bug to incr/decr all times (ym was not** being properly initialized)
- 05 Feb 2014 - Joao Gerd - Add julian day
- **20 Jul 2014 - Joao Gerd - upgrade to new version of m_string.f90**
  - update help banner and documentation

Use `m_stdio`, *`time_module`*, *`m_string`*

Call to *`usage()`*, *`jul2cal()`*, *`doy()`*

Subroutines and functions

`subroutine usage``()`

Use `m_stdio`

Called from *`inctime`*

1.1.2 m_time

Inctime is the main program.

Module

Description

Description This module contains routines and functions to manipulate time periods, e.g, functions to calculate total number of hours, days, months and years between two dates, also contains routines to convert julian days to gregorian day and vice and versa.

History

- 15 Jun 2005 - J. G. de Mattos - Initial Version
- **18 Mar 2010 - J. G. de Mattos - Include Time calculation:**
  - End day of Month [eom]
  - Number of hours [noh]
  - Number of days [nod]
  - Number of months [nom]

    - Number of year [moy]

- **23 Mar 2010 - J. G. de Mattos - Modified the call for Cal2Jul routine** was created the interface block for use of this new Cal2Jul
- 09 May 2013 - J. G. de Mattos - Removed Bug in Number of Hours
- 05 Feb 2014 - J. G. de Mattos - Include day of year calculation

Quick access

Variables `cal2jul`

Routines `jul2cal()`, `cal2jul_()`, `noy()`, `nod()`, `noh()`, `nom()`, `doy()`, `eom()`, `cal2jul__()`

Variables

- `time_module/`**`cal2jul`**

  type

  attrs public

Convert from gregorian to julian day

Subroutines and functions

`function` `time_module/`**`eom`**(*year*, *month*)

Description This function calculate the end day of month.

History

- 18 Mar 2010 - J. G. de Mattos - Initial Version

Parameters

- **year** *[integer,in]*
- **month** *[integer,in]*

Return **day** *[integer]*

`function` `time_module/`**`noh`**(*di*, *df*)

Description This function calculate the total number of hours between two dates.

History

- 18 Mar 2010 - J. G. de Mattos - Initial Version

Parameters

- **di** *[integer,in]* :: Starting Date
- **df** *[integer,in]* :: Ending Date

Return **nhour** *[integer]*

function `time_module/`**`nod`**(*di*, *df*)

Description This function calculate the total number of days between two dates.

History

- 18 Mar 2010 - J. G. de Mattos - Initial Version

Parameters

- **di** *[integer,in]* :: Starting Date
- **df** *[integer,in]* :: Ending Date

Return **nday** *[integer]*

function `time_module/`**`nom`**(*di*, *df*)

Description This function calculate the total number of months between two dates.

History

- 18 Mar 2010 - J. G. de Mattos - Initial Version

Parameters

- **di** *[integer,in]* :: Starting Date
- **df** *[integer,in]* :: Ending Date

Return **nmonth** *[integer]*

function `time_module/`**`noy`**(*di*, *df*)

Description This function calculate the total number of Years between two dates.

History

- 18 Mar 2010 - J. G. de Mattos - Initial Version

Parameters

- **di** *[integer,in]* :: Starting Date
- **df** *[integer,in]* :: Ending Date

Return **nyear** *[integer]*

function `time_module/`**`doy`**(*nymd*, *nhms*)

Description This function calculate the day of the year

History

- 05 Feb 2014 - J. G. de Mattos - Initial Version

Parameters

- **nymd** *[integer,in]* :: year month day (yyyymmdd)
- **nhms** *[integer,in]* :: hour minute second (hhmnsd)

Return **day** *[real]*

Called from `inctime`

Call to `cal2jul__()`

function `time_module/`**`cal2jul_`**(*caldate*)

Description This function calculate the julian day from gregorian day.

History

- 15 Jun 2005 - J. G. de Mattos - Initial Version

Parameters **caldate** *[integer,in]*

Return **julian** *[real]*

Call to `cal2jul__()`

function `time_module/`**`cal2jul__`**(*ymd*, *hms*)

Description This function calculate the julian day from gregorian day

History

- 15 Jun 2005 - J. G. de Mattos - Initial Version

Parameters

- **ymd** *[integer,in]*
- **hms** *[integer,in]*

Return **julian** *[real]*

Called from `doy()`, `cal2jul_()`

subroutine `time_module/`**`jul2cal`**(*jd*, *ymd*, *hms*)

Description This function calculate the gregorian date from julian day.

History

- 15 Jun 2005 - J. G. de Mattos - Initial Version
- **23 Mar 2011 - J. G. de Mattos - Modified Interface** to a subroutine call

Remarks This algorithm was adopted from Press et al.

Parameters

- **jd** *[real,in]*
- **ymd** *[integer]* :: year month day (yyyymmdd)
- **hms** *[integer]* :: hour minute second (hhmnsd)

Called from `inctime`

1.1.3 m_string

Inctime is the main program.

Module

Description

A module to process strings.

Description Make some operations in strings

History

- 15 Dec 2010 - J. G. de Mattos - Initial Version
- 02 Mar 2011 - J. G. de Mattos - Initial code to strTemplate
- 30 Nov 2012 - J. G. de Mattos - All input parameters optionals in strTemplate
- 05 Fev 2014 - J. G. de Mattos - Adding julian day mask
- 20 jun 2014 - J. G. de Mattos - Adding GetTokens feature

Quick access

Variables *gettokens*, *replace*, *str_template*, *num2str*, *mon_lc*, *mon_uc*, *mon_wd*

Routines *gettokens_()*, *float2str()*, *int2str()*, *replace_()*, *str_template_()*

Variables

- m_string/**mon_wd**

  shape

  12.

  type character

  attrs private/parameter=(/'jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'/)

- m_string/**gettokens**

  type

  attrs public

  Get tokens by line

- m_string/**replace**

  type

  attrs public

  Replace a string by another

- m_string/**str_template**

  type

  attrs public

  Replace variables in a template

- m_string/**mon_uc**

shape

12.

type character

attrs private/parameter=(/'jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'/)

- `m_string/`**`mon_lc`**

  shape

  12.

  type character

  attrs private/parameter=(/'jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'/)

- `m_string/`**`num2str`**

  type

  attrs public

  convert a number to string

Subroutines and functions

`subroutine` `m_string/`**`str_template_`**`(`*strg*[, *nymd*[, *nhms*[, *fymd*[, *fhms*[, *jd*[, *doy*[, *label*]]]]]]]`)`

A template formatting a string with variables.

Description A template resolver formatting a string with a string variable and time variables. The format descriptors are similar to those used in the GrADS.

Variables

- %y4 substitute with a 4 digit year
- %y2 a 2 digit year
- %m1 a 1 or 2 digit month
- %m2 a 2 digit month
- %mc a 3 letter month in lower cases
- %Mc a 3 letter month with a leading letter in upper case
- %MC a 3 letter month in upper cases
- %d1 a 1 or 2 digit day
- %d2 a 2 digit day
- %h1 a 1 or 2 digit hour
- %h2 a 2 digit hour
- %h3 a 3 digit hour (?)
- %n2 a 2 digit minute
- %e a string ensemble identify
- %jd a julian day without hours decimals

- %jdh a julian day with hour decimals
- %jy a day of current year without hours decimals
- %jyh a day of current year with hours decimals
- %ix1 initial 1 digit decade
- %ix3 initial 3 digit decade
- %iy2 initial 2 digit year
- %iy4 initial 4 digit year
- %im1 initial 1 or 2 digit month
- %im2 initial 2 digit month (leading zero if needed)
- %imc initial 3 character month abbreviation
- %id1 initial 1 or 2 digit day (leading zero if needed)
- %id2 initial 2 digit day
- %ih1 initial 1 or 2 digit hour
- %ih2 initial 2 digit hour
- %ih3 initial 3 digit hour
- %in2 initial 2 digit minute (leading zero if needed)
- %fx1 forecast 1 digit decade
- %fx3 forecast 3 digit decade
- %fy2 forecast 2 digit year
- %fy4 forecast 4 digit year
- %fm1 forecast 1 or 2 digit month
- %fm2 forecast 2 digit month (leading zero if needed)
- %fmc forecast 3 character month abbreviation
- %fd1 forecast 1 or 2 digit day (leading zero if needed)
- %fd2 forecast 2 digit day
- %fh1 forecast 1 or 2 digit hour
- %fh2 forecast 2 digit hour
- %fh3 forecast 3 digit hour
- %fn2 forecast 2 digit minute (leading zero if needed)

History

- Joao Gerd - 02Mar2011 - Codigo Inicial
- Joao Gerd - 30Nov2012 - All input parameters optionals in strTemplate
- Joao Gerd - 05Fev2014 - Adding julian day mask

Parameters **strg** *[character,inout]*

Options

- **nymd** *[integer,in,optional]*

- **nhms** *[integer,in,optional]*
- **fymd** *[integer,in,optional]*
- **fhms** *[integer,in,optional]*
- **jd** *[real,in,optional]*
- **doy** *[real,in,optional]*
- **label** *[character,in,optional]*

Call to `replace_()`, `int2str()`, `float2str()`

subroutine `m_string/`**`replace_`**(*strg*, *mask*, *repl*)

Description Rotina para substituir a mask pela repl na strg.

History

- Joao Gerd - 20Feb2011 - Codigo Inicial.

Parameters

- **strg** *[character,inout]* :: String
- **mask** *[character,in]* :: maskout
- **repl** *[character,in]* :: replacing string

Called from `str_template_()`

subroutine `m_string/`**`gettokens_`**(*line*, *tokens*, *ntokens*[, *del*])

Parameters

- **line** *[character]*
- **tokens** (*) *[character]*
- **ntokens** *[integer]*

Options **del** *[character,optional]*

function `m_string/`**`int2str`**(*num*, *format*)

Parameters

- **num** *[integer,in]*
- **format** *[character,in]*

Return **int2str** *[character]*

Called from `str_template_()`

function `m_string/`**`float2str`**(*num*, *format*)

Parameters

- **num** *[real,in]*
- **format** *[character,in]*

Return **float2str** *[character]*

Called from `str_template_()`

1.1.4 m_stdio

A F90 module defines std. I/O parameters.

CHAPTER 2

Usage

2.1 Usage

In this page is given an overview on how to use inctime.

2.1.1 Download

The code is hosted in the Redmine portal at CPTEC. To checkout the code, use the following command:

```
$ svn checkout https://svn.cptec.inpe.br/gdad/jgerd/tags/inctime
```

2.1.2 Compile

All is needed is a fortran compiler. To compile the code, enter into the `src` directory and type `make`. The `Makefile` will also compile the associated modules.

Once the compilation is done, an executable called `inctime` is created.

2.1.3 Use

The way to use this program is through the command line as follows:

```
$ ./inctime [yyyymmddhh, yyyymmdd] [<+,->nynmndnhnnns] [Form. output]
```

The inctime parameters are as follows:

- [yyyymmddhh, yyyymmdd]
  - Initial Time
- [<+,->nynmndnhnnns]

  - (-) calculate the passed date
  - (+) calculate the future date (default)
  - (ny) Number of year (default is 0)
  - (nm) Number of months (default is 0)
  - (nd) Number of days (default is 0)
  - (nh) Number of hours (default is 0)
  - (nn) Number of minutes (default is 0)
  - (ns) Number of seconds (default is 0)

- [Form. Output]
  - Format to output date. is a template Format The format descriptors are similar to those used in the GrADS:
    - “%y4” substitute with a 4 digit year
    - “%y2” a 2 digit year
    - “%m1” a 1 or 2 digit month
    - “%m2” a 2 digit month
    - “%mc” a 3 letter month in lower cases
    - “%Mc” a 3 letter month with a leading letter in upper case
    - “%MC” a 3 letter month in upper cases
    - “%d1” a 1 or 2 digit day
    - “%d2” a 2 digit day
    - “%h1” a 1 or 2 digit hour
    - “%h2” a 2 digit hour
    - “%h3” a 3 digit hour (?)
    - “%n2” a 2 digit minute
    - “%e” a string ensemble identify
    - “%jd” a julian day without hours decimals
    - “%jdh” a julian day with hour decimals
    - “%jy” a day of current year without hours decimals
    - “%jyh” a day of current year with hours decimals

See also:

It is possible to use words to compose the output format.

More examples:

```
$ ./inctime 2001091000 +1d %d2/%m2/%y4
$ ./inctime 2001091000 +48h30n %h2Z%d2%MC%y4
$ ./inctime 2001091000 -1h30n 3B42RT.%y4%m2%d2%h2.bin
$ ./inctime 2001091000 -2h45n 3B42RT.%y4%m2%d2%h2.bin
$ ./inctime 2001091000 -1y3m2d1h45n ANYTHING.%y4%m2%d2%h2.ANYTHING
```

2.2 Indices and tables

- genindex
- modindex
- search

Fortran Module Index

m

m_string, 9

t

time_module, 5

Index

C

cal2jul (fortran variable in module time_module), 6
cal2jul_() (fortran function in module time_module), 8
cal2jul__() (fortran function in module time_module), 8

D

doy() (fortran function in module time_module), 7

E

eom() (fortran function in module time_module), 6

F

float2str() (fortran function in module m_string), 12

G

gettokens (fortran variable in module m_string), 9
gettokens_() (fortran subroutine in module m_string), 12

I

inctime (fortran program), 3
int2str() (fortran function in module m_string), 12

J

jul2cal() (fortran subroutine in module time_module), 8

M

m_string (module), 9
mon_lc (fortran variable in module m_string), 10
mon_uc (fortran variable in module m_string), 9
mon_wd (fortran variable in module m_string), 9

N

nod() (fortran function in module time_module), 6
noh() (fortran function in module time_module), 6
nom() (fortran function in module time_module), 7
noy() (fortran function in module time_module), 7
num2str (fortran variable in module m_string), 10

R

replace (fortran variable in module m_string), 9
replace_() (fortran subroutine in module m_string), 12

S

str_template (fortran variable in module m_string), 9
str_template_() (fortran subroutine in module m_string), 10

T

time_module (module), 5

U

usage() (fortran subroutine), 5