my Ada corner

Nasser M. Abbasi

August 28, 2015 page compiled on August 28, 2015 at 9:45pm

Contents

1 Original Ada web site and Ada links **1**

2 how to call Lapack and Blas directly from Ada **1**
2.1 Introduction . . . 1
2.2 Review of the LAPACK and BLAS Ada binding . . . 2
2.3 Installation instruction . . . 4
2.4 source code . . . 4

3 How to compile GTK Ada program **5**

4 How to make Ada generate an exception on some floating points operations? **6**

5 How to use Ada 2005 OO? **6**

6 how to make simple Ada program **7**

7 Ada implementation of decimal representation of exp() **7**

8 Ada implementation of getopt() **12**

9 GNAT 2012 installation log file **18**

1 Original Ada web site and Ada links

My original Ada web site with links is here

2 how to call Lapack and Blas directly from Ada

Important note added June 2013:

This page is obsolete now, left here for archive and information only.

The Ada Lapack code is now housed at `http://sourceforge.net/projects/ada-lapack/`

2.1 Introduction

I downloaded the original BLAS Ada binding written by Duncan Sands from `http://topo.math.u-psud.fr/~sands/Programs/BLAS/index.html` and the LAPACK Ada binding written by Wasu Chaopanon from `ftp://ftp.cs.kuleuven.be/pub/Ada-Belgium/mirrors/gnu-ada/OLD/contrib/lapack-ada/`

and made some minor improvments to the bindings.

This page describes the minor changes made and instructions how to use these bindings from Ada in order call LAPACK and BLAS Fortran functions.

A new tar file for LAPACK and for BLAS with all the changes can be downloaded from the link below.

The changes made to LAPACK binding involve streamlining the source tree structure, writing new Makefiles, simplify the binding to use one package called `lapack` and also adding the documentation shown below.

Changes for the BLAS binding were minimal. It involved changes to the source tree structure and writing Makefiles and adding the documentation shown below.

2.2 Review of the LAPACK and BLAS Ada binding

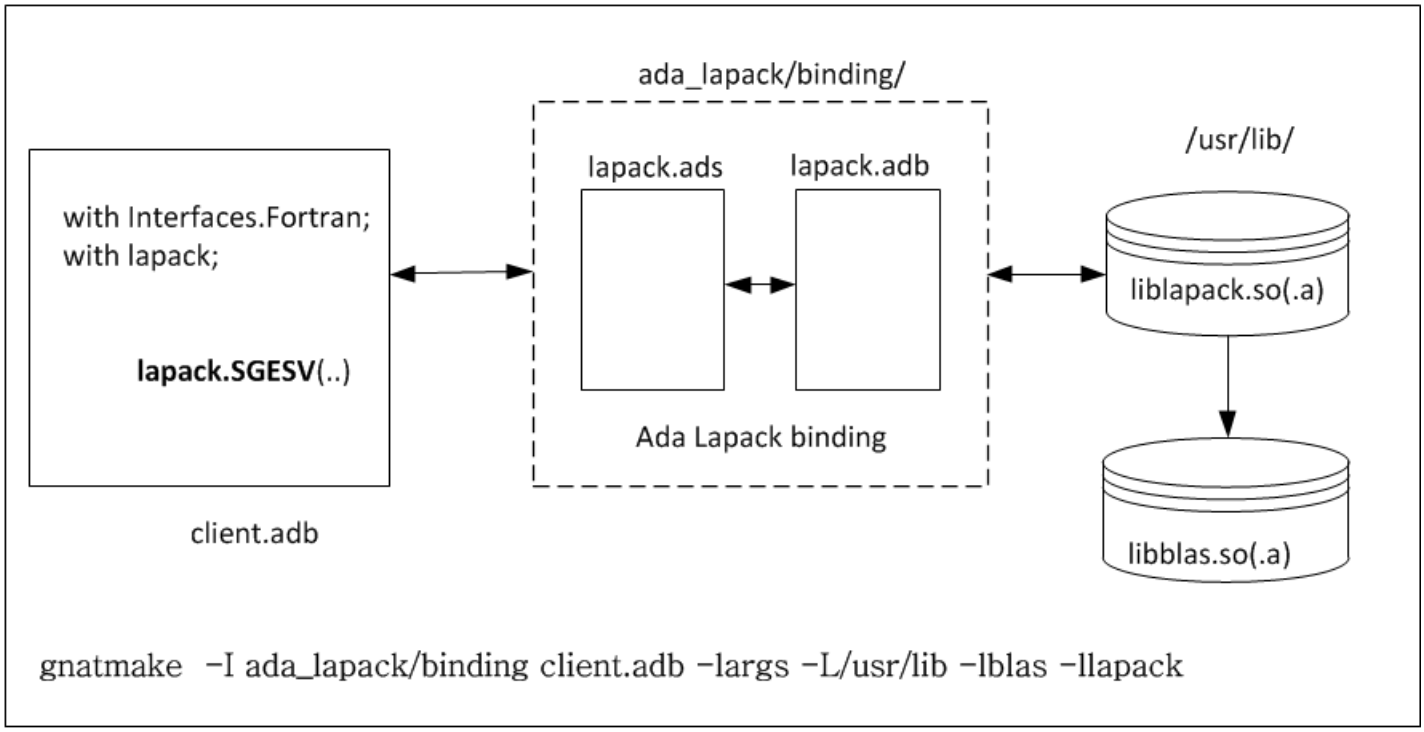
The Ada binding to LAPACK and BLAS is an Ada package which allows one to interface to the native lapack and blas libraries.

The native lapack and blas libraries need to be first installed on the system (on Linux, these libraries will normally be found in `/usr/lib/liblapack.so` and `/usr/lib/libblas.so`)

To use LAPACK from Ada, one needs to install both the native LAPACK and BLAS libraries since LAPACK depends on BLAS.

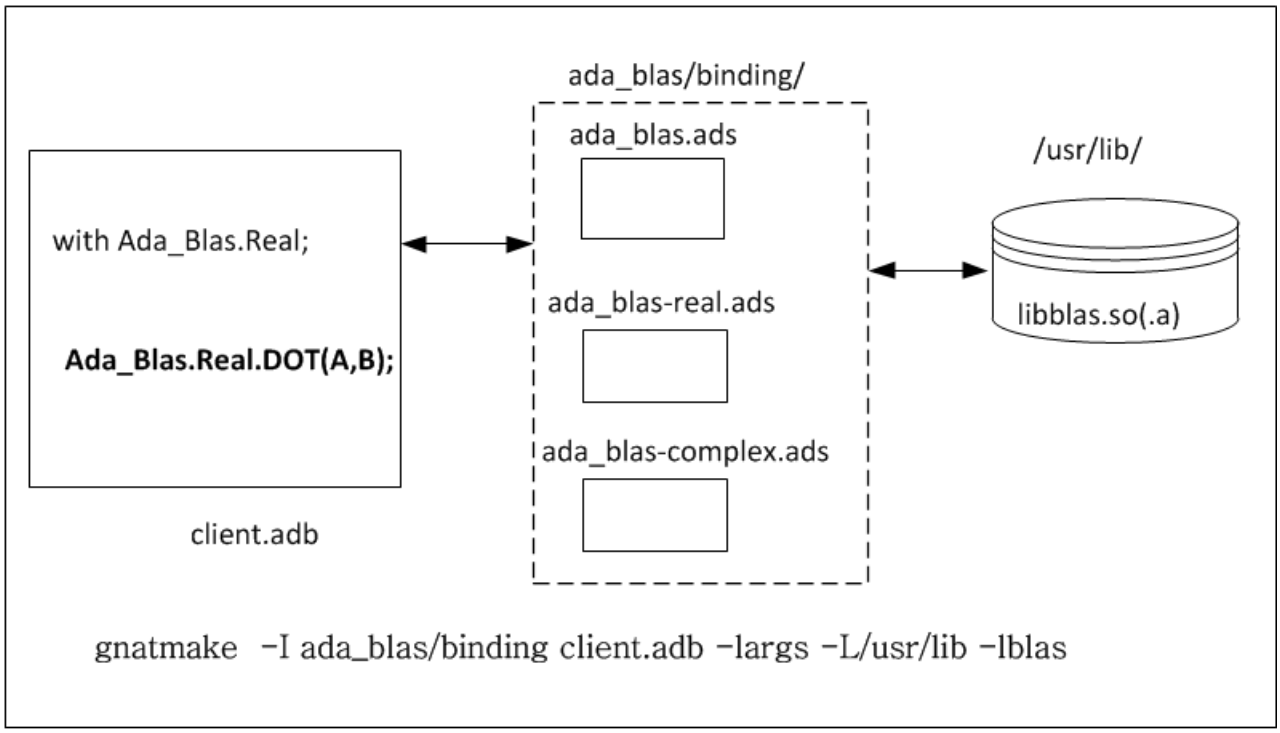
The Ada binding is a thin binding, meaning there is 1-1 mapping between the call to the Ada routine and the corresponding Fortran routine using the same function name in the Fortran libraries.

The following diagram illustrates the use of the LAPACK binding with the needed `gnatmake` command to compile and link the client Ada program.



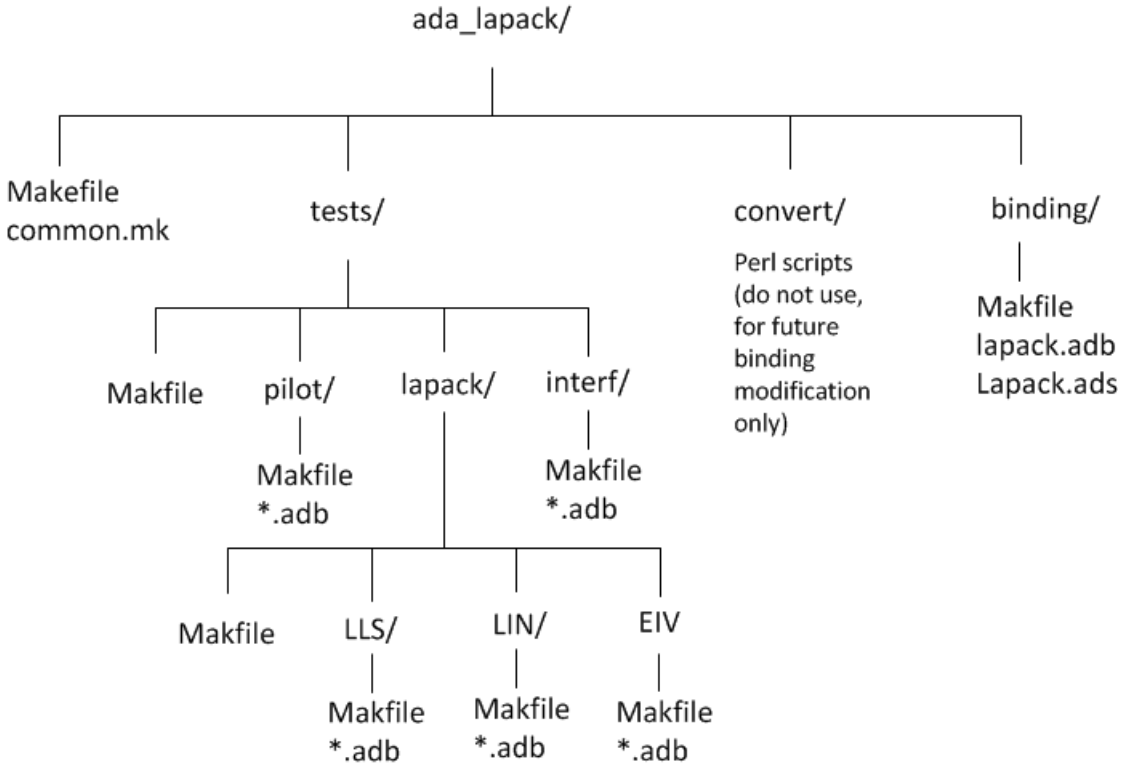
Ada program access to Lapack 77 library via Ada binding interface

And a similar diagram for the BLAS binding interface



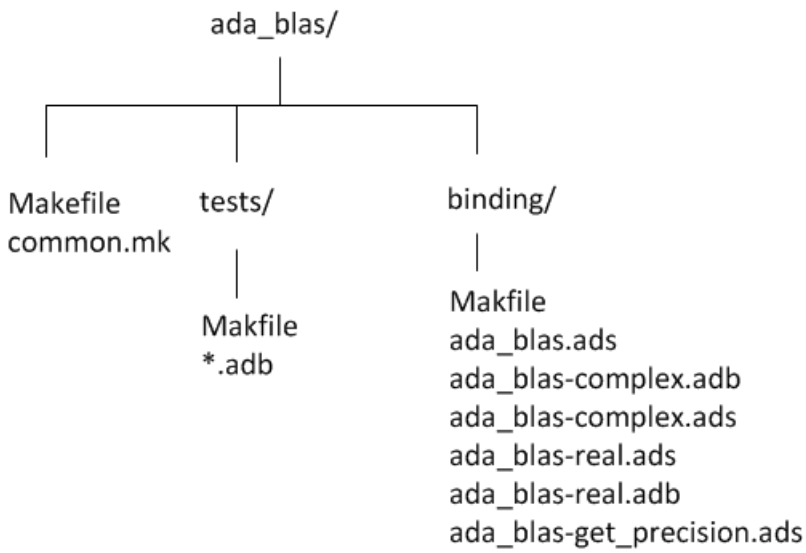
Ada program access to BLAS library via Ada BLAS interface

The source tree structure for LAPACK is described in this diagram



Source ree structure for Ada Lapack77 binding

The full content of the LAPACK tree is listed here `lapack_tree_listing.txt`
The source tree structure for the BLAS binding is described in this diagram



Source ree structure for Ada BLAS binding

2.3 Installation instruction

These instructions explain how to use the Ada LAPACK and BLAS binding in the updated snapshot tar file.

1. install native lapack 77 and blas libraries on your system. These will normally be installed in `/usr/lib/`.
2. download the updated Ada binding in the zip files given in the links at the bottom of this page. They are `ada_lapack_073012.zip` and `ada_blas_073012.zip`
3. Extract the zip files to some location. This will create 2 source trees as shown in the diagrams above.
4. At the top of the each source tree, there is a file called `common.mk` where a Makefile variable is set to point to the directory that contains the native LAPACK and BLAS libraries. This is currently defined to point to `/usr/lib`. Edit this line to change this location only if the location is different in your system.
5. To build the binding, just type `make` from the top of each tree. Make will build the whole tree, including the bindings packages and the test programs.
6. To run the test program, type `make testing` from the top of tree for LAPACK and for BLAS.
7. Examples of clients using the bindings are found in the `tests/` directory of each tree.
8. The binding is in the `binding/` directory in each tree. This is the directory that you need to set the `-I` option to point to when using `gnatmake` as shown in the diagrams above.
9. The following is a simple example of using the Ada LAPACK binding to solve $Ax = b$ mysolve.adb.txt which can also be found in the `ada_lapack/tests/pilot/` directory
10. That is all! Have fun using Ada with LAPACK and BLAS.

2.4 source code

1. `ada_lapack_073012.zip`
2. `ada_blas_073012.zip`

3 How to compile GTK Ada program

```
$ gnatmake -I../pragmarc mine_detector.adb `gtkada-config`
gcc-4.4 -c -I../pragmarc -I/usr/share/ada/adainclude/gtkada2 mine_detector.adb
gcc-4.4 -c -I../pragmarc -I/usr/share/ada/adainclude/gtkada2 user_if.adb
gcc-4.4 -c -I../pragmarc -I/usr/share/ada/adainclude/gtkada2 field.ads
gcc-4.4 -c -I../pragmarc -I/usr/share/ada/adainclude/gtkada2 field-operations.adb

gnatbind -I../pragmarc -aI/usr/share/ada/adainclude/gtkada2 -aO/usr/lib/ada/adalib/gtkada2 -x mine_detector.

gnatlink mine_detector.ali -L/usr/lib -lgtkada2 -pthread -lgtk-x11-2.0 -lgdk-x11-2.0 -latk-1.0 -lgio-2.0
-lpangoft2-1.0 -lpangocairo-1.0 -lgdk_pixbuf-2.0 -lm -lcairo -lpango-1.0 -lfreetype
-lfontconfig -lgobject-2.0 -lgmodule-2.0 -lgthread-2.0 -lrt -lglib-2.0
$
```

4 How to make Ada generate an exception on some floating points operations?

```
On Thu, 20 Nov 2008 12:09:41 +0100, Markus Schoepflin wrote:

> is it possible to influence the behaviour of GNAT regarding the handling of
> NANs? (Most importantly in the special case of division by zero.)
>
> We need to get exceptions whenever a NAN is generated, is this possible
> somehow? (For example by setting Machine_Overflow to True and recompiling
> the compiler itself.)

You can scrap IEEE stuff in favor of Ada semantics by declaring your own
floating-point [sub]type with a range specified. The compiler will be
forced to check values:

   type Safe_Float is digits 6 range -10.0E10..+10.0E10;

or

   subtype Safe_Float is Float range Float'Range;

then

   X : Safe_Float := 1.0;
   Y : Safe_Float := 0.0;
begin
   Y := X / Y;
exception
   when Error : others =>  -- Should print "range check failed"
      Put_Line (Exception_Message (Error));
end;

--
Regards,
Dmitry A. Kazakov
http://www.dmitry-kazakov.de
```

5 How to use Ada 2005 OO?

from http://en.wikibooks.org/wiki/Ada_Programming/Object_Orientation

```
package X is
   type Object is tagged null record;

   procedure do (This: in Object; That: in Boolean);
end X;

with X;
procedure Main is
   Obj : X.Object;
begin
   Obj.do (That => True);
end Main;
```

6 how to make simple Ada program

If gnat is not installed, install it (on linux) using something similar to

```
sudo apt-get install gnat-4.6
```

write the following code in file called `hello_world.adb`

```
with ada.text_io; use ada.text_io;
procedure hello_world is
 begin
   put_line("hello world");
end hello_world;
```

compile using

```
gnatmake hello_world.adb

         gcc-4.6 -c hello_world.adb
         gnatbind -x hello_world.ali
         gnatlink hello_world.ali
```

Run it using `./hello_world`

7 Ada implementation of decimal representation of exp()

This is an Ada implementation of decimal representation of e based on SPIGOT algorithm for π by S. Rabinowitz & S. Wagon, The American Mathematical Monthly, March 1995.

```
-- More e digits trivia.
-- Feel free to copy, distribute as long as this header attached so
-- original algorithm creators and implementors are known.
--
-- This is an Ada implementation of decimal representation of 'e'
```

```
-- based on SPIGOT algorithm for \pi by
-- S.  Rabinowitz & S. Wagon, _The_American_Mathematical_Monthly_, March 1995
--
--  A C implementation of the above was posted on the net by
--  Ed Hook
--  MRJ Technology Solutions, Inc.
--  NAS, NASA Ames Research Center
--  Internet: hook@nas.nasa.gov
--
-- This is an Ada implementation of the above using GNAT (gnu Ada compiler),
-- with the added feature is that it computes the frequency of each digit in e,
-- and computes the largest consecutive sequences of each digit within the
-- expression that represents digits of e.
--
-- the following is the result. my PC is still running trying to find the
-- frequency for 200,000 digits and more for e, and it's been several days
-- and not finished. So this is a partial results. (PC is 200 MHz pentium,
-- running Linux 2.0.36, and compiler is GNAT 3.11p
--
-- offcourse as number of digits of e goes very large, each digit is expected
-- to show as often as any other digit.
--
-- by Nasser M. Abbasi  nma@12000.org    feb. 20, 1999.
--
-- results:

-- this is distribution table for digits in e as function of how many
-- digits.
-- for example, when looking at 5000 digits of e, we find 497 0's,
-- 478 1's, etc..  (this is for digits after the decimal point of e)
--
--
--                                  #digits  in e
--            ---------------------------------------------------------------
--               500  5,000  20,000  50,000  200,000
--            ----------------------------------------------------------------
--how many 0's   51    497    1,949  4,948   19,916
--how many 1's   43    478    2,010  5,055   20,367
--how many 2's   50    492    2,020  4,969   19,794
--how many 3's   53    514    2,080  5,026   20,071
--how many 4's   52    470    1,989  4,966   20,082
--how many 5's   44    478    1,979  5,046   20,038
--how many 6's   51    545    2,057  5,133   20,221
--how many 7's   60    525    1,977  4,959   19,817
--how many 8's   40    509    1,966  4,972   19,939
--how many 9's   56    492    1,974  4,926   19,755
--
-------------------------------------------------------------------------
--most occurring  '7'   '7'     '3'    '6'     '1'
-------------------------------------------------------------------------
--least occurring '8'   '4'     '0'    '9'     '9'
-------------------------------------------------------------------------
```

```
--difference
--between largest 20   55      131    207     612
--and smallest
--in frequency
-------------------------------------------------------------------------
--difference
--between largest 4%  1.1%   0.655%  0.414%  0.306%
--and smallest
--frequency in %
--
--
--consecutive frequencies: under each column, there are 3 values, the first
--is the number of digits that occurred next to each others for that digit,
--and the start of this sub sequence, and its end, in position values.
--
--for example, for 5,000 digits of e, we see that largest consecutive
--sequence of digit '0' had length of 3, and it started at digit position
--328 to position 330.  Digit positions are counted from left to right at
--the decimal point.  for example e=2.718, here digit '7' is at position 1,
--'1' is at position 2, etc..
--
--                                  #digits  in e
--     ----------------+-----------------+-----------------------------------
--          5,000      |   20,000        |    50,000         | 100,000
--     ----------------+------------------+------------------+---------------
-- 0's   (3,328,330)   | (4,7688,7691)   |  *no change*      |(6,89296,89301)
-- 1's   (3,427,429)   | (5,12220,12224) |  *no change*      | *no change*
-- 2's   (2,2744,2746) | (4,17309,17312) |  (5,33483,33487)  | *no change*
-- 3's   (4,3354,3375) |   *no change*   |  *no change*      | *no change*
-- 4's   (3,787,789)   | (4,11806,11809) |  *no change*      | *no change*
-- 5's   (4,3620,3623) |   *no change*   |  *no change*      | *no change*
-- 6's   (5,4992,4996) |   *no change*   |  *no change*      | *no change*
-- 7's   (4,1071,1074) |   *no change*   |  *no change*      | *no change*
-- 8's   (4,723,726)   |   *no change*   |  *no change*      | *no change*
-- 9's   (3,47,49)     |   *no change*   |  (4,29344,29347)  | *no change*
--
--
--
--Compiler:    GNAT 3.11p , see http://www.adahome.com to download
--To compile:  save this file as dist_e_final.adb and type
--             gnatmake dist_e_final.adb
--system:      Linux 2.0.36
--Date:        feb. 17, 1999
--To Run:      ./dist_e_final
--             For example, to see e for 70 digits do:
--
-- ./dist_e_final 70
-- 2.7182818284590452353602874713526624977572470936999595749669676277240766
-- frequency of  0 is  4
-- frequency of  1 is  3
-- frequency of  2 is  9
-- frequency of  3 is  4
```

```
-- frequency of  4 is  7
-- frequency of  5 is  7
-- frequency of  6 is  10
-- frequency of  7 is  12
-- frequency of  8 is  5
-- frequency of  9 is  9
--
-- performance note: On Pentium PRO 200 MHZ, using GNAT 3.11p, Linux 2.0.36,
-- 128 MB RAM. No other activity on PC, and for 1,000,000 digits, this
-- program will generate about 50 digits each minutes. So, for 1,000,000
-- digits it will take about 13 days. for larger than 1,000,000 you might
-- encounter stack overrun, depending on amount of memory you have...
--
-- notice the main algorithm is O(n^2).
--

with Ada.Text_Io;         use Ada.Text_Io;
with ada.command_line;    use ada.command_line;

procedure Dist_E_final is
   type E_Type is array( Natural range <> ) of Natural;
   Distribution : array(0..9) of Natural := (others => 0);
   Num_Of_Digits : Natural;

   type Sequence_item is record
      Starts_At, Ends_At, Length : Natural;
   end record;
   Sequence: array(0..9) of Sequence_Item := (others=>(0,0,0));
   current_Digit, Current_Sequence_Length, Current_Sequence_Start: Natural :=0;

   procedure Update_Sequence(Next_Digit_Position, next_digit: Natural) is
   begin
      if( next_Digit /= Current_Digit) then
         if( Sequence( current_Digit ).Length < Current_Sequence_Length) then
            Sequence( current_Digit ).Length := Current_Sequence_Length;
            Sequence( current_Digit ).Starts_At := Current_Sequence_start;
            Sequence( Current_Digit ).Ends_At := Next_Digit_Position -1;
         end if;

         Current_Digit := Next_Digit;
         Current_Sequence_Length := 1;
         Current_Sequence_Start  := Next_Digit_Position;

      else
         Current_Sequence_Length := Current_Sequence_Length +1;
      end if;

   end Update_Sequence;

   procedure Done_Sequence( Current_Digit_Position: Natural)  is
   begin
      if( Sequence( current_Digit ).Length < Current_Sequence_Length) then
```

```
         Sequence( current_Digit ).Length := Current_Sequence_Length;
         Sequence( current_Digit ).Starts_At := Current_Sequence_start;
         Sequence( Current_Digit ).Ends_At := current_Digit_Position ;
      end if;
   end Done_Sequence;

begin

   if( Argument_Count /= 1 ) then
      Put_Line("usage: dist_e ");
      return;
   end if;

   begin
      Num_Of_Digits := natural'value( Argument(1));

      if( Num_Of_Digits = 0 ) then
         Put_Line("value for number of digits must be larger than zero");
         return;
      end if;

   exception
      when others =>
         Put_Line("Exception. invalid value for number of digits");
         return;
   end;

   declare            -- the algorithm itself is in this block
      E: E_Type( 1 .. Num_Of_Digits+2 ) := (others=> 1);
      Carry : Natural;
   begin

      Put("2.");

      for I in E'first .. E'Last-2 loop
         Carry := 0;
         for J in reverse E'first .. E'Last loop
            E(J) := ( E(J) * 10 ) + Carry;
            Carry := E(J)/(J+1);
            E(J) := E(J) rem (J+1);
         end loop;

         Put(Natural'Image(Carry)(2));            -- print current digit of e
         Distribution( Carry ) := Distribution( Carry ) + 1;
         Update_Sequence(I,Carry);
      end loop;

      Done_Sequence(E'Last-2);
   end;

   New_Line;
   for I in Distribution'Range loop
```

```
      Put_line("frequency of " & Natural'Image(I) & " is "
               & natural'Image( Distribution(I)  ));
   end loop;


   for I in sequence'Range loop
      if( Sequence(I).Length = 0 ) then
         Put_Line("Digit "& Natural'Image(I) & " was not seen.");
      else

         Put_line("largest concecutive  seq of " & Natural'Image(I)
                  &" started at digit "
                  & natural'Image( sequence(I).Starts_at )
                  & " and ended at digit "
                  & natural'Image( sequence(I).ends_at )
                  & " of length "
                  & natural'Image( sequence(I).length ));
      end if;
   end loop;

end Dist_E_final;
```

8 Ada implementation of getopt()

This package is an Ada implementation of getopt() as specified by the document "The Single UNIX Specification, Version 2", Copyright 1997 The Open Group

```
------------------------------------------------------------------------------
--                                                                          --
--                              G E T O P T                                 --
--                                                                          --
--                                S p e c                                   --
--                                                                          --
-- $Header: getopt.ads,v 1.1.1.1 1999/03/01 12:23:04 nabbasi Exp $          --
--                                                                          --
--                  Copyright (C) 1998 Nasser Abbasi                        --
--                                                                          --
-- This is free software;  you can  redistribute it  and/or modify it under --
-- terms of the  GNU General Public License as published  by the Free Soft- --
-- ware  Foundation;  either version 2,  or (at your option) any later ver- --
-- sion. GETOPT is distributed in the hope that it will be useful, but WITH --
-- OUT ANY WARRANTY;  without even the  implied warranty of MERCHANTABILITY --
-- or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License --
-- for  more details. Free Software Foundation,  59 Temple Place - Suite    --
-- 330,  Boston, MA 02111-1307, USA.                                        --
--                                                                          --
-- As a special exception,  if other files  instantiate  generics from this --
-- unit, or you link  this unit with other files  to produce an executable, --
-- this  unit  does not  by itself cause  the resulting  executable  to  be --
-- covered  by the  GNU  General  Public  License.  This exception does not --
-- however invalidate  any other reasons why  the executable file  might be --
```

```
-- covered by the  GNU Public License.                                  --
-------------------------------------------------------------------------

-- change history:                                                      --
--                                                                      --
-- name         changes                                                 --
-- ----------   -------------------------------------------------------
-- NMA021899    created                                                 --
-- NMA030299    Made it modified GPL. chanegd header.                   --
--                                                                      --
-- description:                                                         --
--                                                                      --
-- This package is an Ada implementation of getopt() as specified by the    --
-- document "The Single UNIX Specification, Version 2", Copyright 1997 The  --
-- Open Group                                                           --
--                                                                      --
-- Compiler used: GNAT 3.11p                                            --
-- Platform:      Linux 2.0.36 ( Red hat 5.2)                           --
--                                                                      --

with Ada.Strings.Unbounded; use Ada.Strings.Unbounded;

package Getopt is

   function Getopt (Optstring : String)  return Integer;


   Optind : Positive;
   Optarg : Unbounded_String;
   Optopt : Character := ' ';
   Opterr : Integer := 1;

end Getopt;

-------------------------------------------------------------------------
--                                                                      --
--                              G E T O P T                             --
--                                                                      --
--                                BODY                                  --
--                                                                      --
-- $Header: getopt.adb,v 1.2 1999/03/01 12:54:03 nabbasi Exp $          --
--                                                                      --
--                                                                      --
--                                                                      --
--                  Copyright (C) 1998 Nasser Abbasi                    --
--                                                        --
--                                                                      --
-- This is free software;  you can  redistribute it  and/or modify it under --
-- terms of the  GNU General Public License as published  by the Free Soft- --
-- ware  Foundation;  either version 2,  or (at your option) any later ver- --
-- sion. GETOPT is distributed in the hope that it will be useful, but WITH --
-- OUT ANY WARRANTY;  without even the  implied warranty of MERCHANTABILITY --
```

```
-- or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License --
-- for  more details. Free Software Foundation,  59 Temple Place - Suite    --
-- 330,  Boston, MA 02111-1307, USA.                                        --
--                                                                          --
-- As a special exception,  if other files  instantiate  generics from this --
-- unit, or you link  this unit with other files  to produce an executable, --
-- this  unit  does not  by itself cause  the resulting  executable  to  be --
-- covered  by the  GNU  General  Public  License.  This exception does not --
-- however invalidate  any other reasons why  the executable file  might be --
-- covered by the  GNU Public License.                                      --
--                                                                          --
------------------------------------------------------------------------------
--                                                                          --
-- change history:                                                          --
--                                                                          --
-- name         changes                                                     --
-- ----------   -------------------------------------------------------------
-- NMA021899    created                                                     --
-- NMA030299    Changed header to make it modified GPL                      --
--                                                                          --
-- description:                                                             --
--                                                                          --
-- This package is an Ada implementation of getopt() as specified by the    --
-- document "The Single UNIX Specification, Version 2", Copyright 1997 The  --
-- Open Group                                                               --
--                                                                          --
-- This describes the items involveed using example                         --
--                                                                          --
--                                                                          --
--         curopt                                                           --
--           |                                                              --
--           V                                                              --
-- "-f foo -dbc -k"                                                         --
--  ^                                                                       --
--  |                                                                       --
-- optind                                                                   --
--                                                                          --
-- optind is position (index) that tells which command line argument is     --
-- being processed now.                                                     --
-- curopt tells which optchar is being processed within one command line    --
-- argument. This is needed only if more that one optchar are stuck         --
-- togother in one argument with no space, as in -df where both d and f     --
-- are valid optchar and d takes no optarg.                                 --
--                                                                          --
--                                                                          --
-- Compiler used: GNAT 3.11p                                                --
-- Platform:      Linux 2.0.36 ( Red hat 5.2)                               --
--

with Ada.Strings.Unbounded; use Ada.Strings.Unbounded;
with Ada.Command_Line; use Ada.Command_Line;
with Ada.Text_Io; use Ada.Text_Io;
```

```
package body Getopt is

   Curopt  : Natural := 2;

   --------------------
   -- No_Optarg_Case --
   --------------------

   procedure No_Optarg_Case is
   begin
      if (Curopt < Argument (Optind)'Length) then
         Curopt := Curopt + 1;
      else
         Curopt := 2;
         Optind := Optind + 1;
      end if;
   end No_Optarg_Case;

   ------------
   -- Getopt --
   ------------

   function Getopt (Optstring : String)  return Integer is
   begin

      if (Argument_Count = 0  or else optind > Argument_Count
          or  else (Argument (optind)(1) /= '-')) then
         return -1;
      end if;

      if (Argument (optind)'Length = 1) then
         return -1;
      end if;

      --  according to The Single UNIX  Specification, Version 2, if "--"
      --  is found, return -1 after  ++optind.
      if (Argument (Optind)(2) = '-') then
         Optind := Optind + 1;
         return -1;
      end if;

      --  if we get here, the command argument has "-X"
      for I in Optstring'Range loop
         if (Optstring (I) = Argument (optind)(Curopt)) then
            if (I < Optstring'Length) then
               if (Optstring (I + 1) = ':') then

                  --  see if optarg stuck to optchar
                  if (Argument (Optind)'Length -  Curopt > 0) then
                     Optarg  := To_Unbounded_String
                      (Argument (optind)(Curopt + 1 .. Argument (optind)'Length));
```

```
                  Curopt := Curopt + 1;
                  optind := Optind + 1;
                  return character'Pos (Optstring (I));
               end if;

               --  see if optarg on separate argument
               if (Optind < Argument_Count) then
                  Curopt := 2;
                  optind  := optind + 1;
                  optarg  := To_Unbounded_String (Argument (optind));
                  optind  := optind + 1;
                  return character'Pos (Optstring (I));
               else
                  Optind := Optind + 1;
                  Optopt := Optstring (I);

                  if (Opterr = 1  and Optstring (1) /= ':') then
                     Put_Line (Standard_Error,
                              "Argument expected for the -"&
                              Optstring (I .. I) & " option");
                  end if;

                  if (Optstring (1) = ':') then
                     return Character'Pos (':');
                  else
                     return  Character'Pos ('?');
                  end if;
               end if;
            else  -- current optchar matches and has no arg option
               No_Optarg_Case;
               return character'Pos (Optstring (I));
            end if;
         else -- last char in optstring, can't have argument
            No_Optarg_Case;
            return character'Pos (Optstring (I));
         end if;
      end if;
   end loop;

   Optopt := Argument (Optind)(Curopt);
   No_Optarg_Case;

   --  we get here if current command argument not found in optstring
   return character'Pos ('?');

   end Getopt;

begin
   Optarg := To_Unbounded_String ("");
   Optind := 1;
end Getopt;
```

This is a test program of the above package.

```
-- Test example showing how to use GETOPT Ada package
-- Nasser Abbasi

with Ada.Text_Io; use Ada.Text_Io;
with Ada.Command_Line; use Ada.Command_Line;
with Ada.Strings.Unbounded; use Ada.Strings.Unbounded;

with Getopt;

procedure Test_Getopt is
   Test_String : String := "c:di:n:p:u:V";
   Optchar : character;
   Value   : Integer;
begin
    Getopt.Opterr := 1;


   loop
      Value := Getopt.Getopt( Test_String );
      exit when Value = -1;

      optchar :=  Character'Val( Value );
      case optchar is
         when 'c' =>
            Put_Line("commant is "& To_String(Getopt.Optarg));
         when 'd' =>
             Put_Line("debug on");
         when 'i' =>
            Put_line("got -i, its argument is:" & To_String(Getopt.Optarg) );
         when 'n' =>
            Put_line("got -n, its argument is:" & To_String(Getopt.Optarg));
         when 'p' =>
            Put_line("got -p, its argument is:" & To_String(Getopt.Optarg));
         when 'u' =>
            Put_line("got -u, its argument is:" & To_String(Getopt.Optarg));
         when 'V' =>
            Put_line("got -V");

         when '?' =>
            Put_Line("got ?, optopt is " & Getopt.Optopt);

         when ':' =>
            Put_Line("get :, optopt is "& Getopt.optopt);

         when others => null;
      end case;
   end loop;
```

```
   -- now lets print the remaining arguments if any
   declare
     Index : positive;
   begin
      Index := Getopt.Optind;
      for I in Index..Argument_Count loop
         Put_Line( Argument(I) );
      end loop;
   end;

end Test_Getopt;
```

9 GNAT 2012 installation log file

gnat2012_installation_log_file.txt