

Source file: dm routines.f

```

c=====
c   Implements matrix-matrix multiply
c
c   c = a b
c
c   where a, b and c are n x n (square) real*8 matrices.
c=====
      subroutine dmmult(a,b,c,n)

         implicit      none

         integer       n
         real*8        a(n,n),  b(n,n),  c(n,n)

         integer       i,      j,      k

         do j = 1, n
           do i = 1, n
             c(i,j) = 0.0d0
             do k = 1, n
               c(i,j) = c(i,j) + a(i,k) * b(k,j)
             end do
           end do
         end do

         return

      end

c=====
c   Writes a double precision matrix (two dimensional
c   array) to file 'fname'.  If 'fname' is the
c   string '-', the matrix is written to standard input.
c
c   This routine is modelled on 'dvto' previously
c   discussed in class: see ~phy329/ex3/dvto.f
c=====
      subroutine dmto(fname,a,d1,d2)
c-----
c   Arguments:
c
c   fname: (I)  File name
c   a:      (I)  Input matrix
c   d1:     (I)  First dimension of a
c   d2:     (I)  Second dimension of a
c-----
         implicit      none

         integer       indlnb,      getu

         character*(*) fname
         integer       d1,          d2
         real*8        a(d1,d2)

         integer       ust dout
         parameter     ( ust dout = 6 )

         integer       uto,          rc

c-----
c   Parse fname: either "attach" 'uto' to stdout or
c   get a unit number using 'getu', and open the
c   file 'fname' for formatted I/O via 'uto'
c-----
         if( fname .eq. '-' ) then
           uto = ust dout
         else
           uto = getu()
           open(uto,file=fname(1:indlnb(fname)),
             & form='formatted',iostat=rc)
           if( rc .ne. 0 ) then
             write(0,*) 'dmto: Error opening ',
             & fname(1:indlnb(fname))
           return
         end if
         end if

c-----

```

```

c   Write dimensions, then array elements
c-----
         write(uto,*,iostat=rc) d1, d2
         if( rc .ne. 0 ) then
           write(0,*) 'dmto: Error writing dimensions'
           go to 500
         end if

         write(uto,*,iostat=rc) a
         if( rc .ne. 0 ) then
           write(0,*) 'dmto: Error reading matrix'
         end if

c-----
c   Exit: Close file and return
c-----
500      continue
         if( uto .ne. ust dout ) then
           close(uto)
         end if

         return

      end

c=====
c   Returns a double precision matrix (two dimensional
c   array) read from file 'fname'.  If 'fname' is the
c   string '-', the matrix is read from standard input.
c
c   The dimensions of the matrix must precede the matrix
c   elements themselves in the file.  Specifically, the
c   file should have been created using the following
c   list-directed, formatted READ statement
c   (or equivalent):
c
c   integer      d1,      d2
c   real*8       a(d1,d2)
c   integer      uout
c
c   write(uout,*) d1, d2
c   write(uout,*) a
c
c   This routine is modelled on 'dvfrom' previously
c   discussed in class: see ~phy329/ex3/dvfrom.f
c
c   Note the use of helper routine 'dmfrom1' which
c   reads actual array values once bounds have been
c   extracted from file.
c=====
      subroutine dmfrom(fname,a,d1,d2,asize)
c-----
c   Arguments:
c
c   fname: (I)  File name
c   a:      (O)  Return matrix
c   d1:     (O)  First dimension of a
c   d2:     (O)  Second dimension of a
c   asize: (I)  Maximum size (d1 * d2) of a
c-----
         implicit      none

         integer       indlnb,      getu

         character*(*) fname
         integer       d1,          d2,          asize
         real*8        a(d1,d2)

         integer       ust din
         parameter     ( ust din = 5 )

         integer       ufrom,      rc,          i,          j

c-----
c   Parse fname: either "attach" 'ufrom' to stdin or
c   get a unit number using 'getu', and open the
c   file 'fname' for formatted I/O via 'ufrom'
c-----
         if( fname .eq. '-' ) then
           ufrom = ust din

```

```

else
  ufrom = getu()
  open(ufrom,file=fname(1:indlnb(fname)),
    & form='formatted',iostat=rc,status='old')
  if( rc .ne. 0 ) then
    write(0,*) 'dmfrom: Error opening ',
    & fname(1:indlnb(fname))
  return
end if
end if

c-----
c Read dimensions and abort if there is insufficient
c storage for the entire matrix. Note the 'go to'
c to the 'exit block' since we've opened a file now
c and should close it, even if there's an error.
c Also, we set the dimensions to 0 for all error
c conditions as a way of communicating failure to
c the calling routine.
c-----
read(ufrom,*,iostat=rc) d1, d2
if( rc .ne. 0 ) then
  write(0,*) 'dmfrom: Error reading dimensions'
  d1 = 0
  d2 = 0
go to 500
end if
if( (d1 * d2) .gt. asize ) then
  write(0,*) 'dmfrom: Insufficient storage'
  d1 = 0
  d2 = 0
go to 500
end if

c-----
c Now that dimensions have been determined call
c helper routine to read values
c-----
call dmfrom1(ufrom,a,d1,d2,rc)
if( rc .ne. 0 ) then
  write(0,*) 'dmfrom: Error reading matrix'
  d1 = 0
  d2 = 0
end if

c-----
c Exit: Close file and return
c-----
500 continue
if( ufrom .ne. ustdin ) then
  close(ufrom)
end if

return

end

c=====
c Helper routine for dmfrom: Reads array values, returns
c I/O status to calling routine via 'rc'
c=====
subroutine dmfrom1(ufrom,a,d1,d2,rc)

  implicit none

  integer d1, d2, ufrom, rc
  real*8 a(d1,d2)

  read(ufrom,*,iostat=rc) a

  return

end

```

Source file: tdm.f

```

c=====
c Test program for subroutine 'dmfrom', 'dmto' and
c 'dmmult' (see 'dmroutines.f')
c
c Program expects one argument, the name of a file which
c contains a real*8 square matrix written as described
c in the documentation for 'dmfrom' in 'dmroutines.f'
c Use '-' to read from stdin. Program then computes
c square of matrix and outputs result to stdout.
c=====

program tdm

implicit none

integer iargc

character*256 fname

c-----
c Maximum size for input and output arrays (matrices).
c-----
integer maxsize
parameter ( maxsize = 100 000 )
real*8 a(maxsize), asq(maxsize)
integer dia, d2a

if( iargc() .ne. 1 ) go to 900
call getarg(1,fname)

c-----
c Read matrix ...
c-----
call dmfrom(fname,a,dia,d2a,maxsize)
if( dia .gt. 0 .and. d2a .gt. 0 ) then
  if( dia .eq. d2a ) then

c-----
c Compute square ...
c-----
call dmmult(a,a,asq,dia,dia)

c-----
c ... and output.
c-----
call dmto('-',asq,dia,dia)
else
  write(0,*) 'tdm: Input array not square'
end if
else
  write(0,*) 'tdm: dmfrom() failed'
end if

stop

900 continue
write(0,*) 'usage: tdm <file name>'
write(0,*)
write(0,*) ' Use ''tdm -'' to read ',
& 'from standard input'

stop

end

```

Source file: Makefile

```
#####
# Note that this 'Makefile' assumes that the following
# environment variables are set:
#
# F77
# F77PP
# F77FLAGS
# F77CFLAGS
# F77LFLAGS
#
# F77PP is the name of the program which wil translate
# Fortran 77 source code written on the SGIs to a form
# appropriate for the target machine:
#
# SGIs:      setenv  F77PP  touch
# Crays:     setenv  F77PP  f77transcray
#
# EXERCISE: Put the appropriate 'setenv' commands in
# your '~/.cshrc'. See 'phy329@einstein:~/cshrc' for
# specifics.
#####
.IGNORE:

F77_COMPILE = $(F77) $(F77FLAGS) $(F77CFLAGS)
F77_LOAD    = $(F77) $(F77FLAGS) $(F77LFLAGS)

.f.o:
    $(F77PP) $*.f
    $(F77_COMPILE) $*.f

EXECUTABLES = tdm

all: $(EXECUTABLES)

tdm: tdm.o dmoutines.o
    $(F77_LOAD) tdm.o dmoutines.o -lp329f -o tdm

clean:
    rm *.o
    rm $(EXECUTABLES)
```

Source file: sgi_cshrc

```
# Default .cshrc for PHY329, Fall 1998

# Enable file-completion
set filec

# Be careful with output-redirection
set noclobber

# Add some dirs. containing executables to the search path
setenv HOMEMWC ~matt
set path=($HOME/bin $HOMEMWC/scripts /usr/local/bin $path)

# Define the default printer (two-sided queue on Center
# for Relativity HP LaserJet in room 9.301
setenv PRINTER lp2

# Define the default editor (vi, emacs, pico)
setenv EDITOR vi

# Remember last 100 commands
set history = 100

# For communication with 'make'
setenv F77      'f77'
setenv F77PP   'touch'
setenv F77FLAGS '-g -n32'
setenv F77CFLAGS '-c'
setenv F77LFLAGS '-L/usr/localn32/lib -n32'

# The following commands (within the scope of the 'if'), are
# executed only for interactive shells.
if ( (! $?ENVONLY) && $?prompt ) then
# set the prompt to show the host name and event number.
set prompt="hostname' \> "
```

```
# Define some aliases (see '~/.aliases' for actual definitions)
source ~/.aliases

# Tell LaTeX and TeX where to find input files:
setenv TEXINPUTS ":/usr/local/tex/inputs/:"

setenv XENVIRONMENT $HOME/.Xdefaults.mine
endif
```

Source file: sgi_output

```
#####
# Building 'tdm' and sample output
#####

einstein% pwd; ls
/usr2/people/phy329/f77/ex7
Makefile      dmoutines.f      sgi_output      tdm.f

einstein% make
touch tdm.f
f77 -g -n32 -c tdm.f
touch dmoutines.f
f77 -g -n32 -c dmoutines.f
f77 -g -n32 -L/usr/localn32/lib -n32 tdm.o dmoutines.o -lp329f -o tdm

einstein% tdm
usage: tdm <file name>

Use 'tdm -' to read from standard input

einstein% tdm -
2 2
1 2 3 4
                2          2
7.000000000000000    10.000000000000000    15.000000000000000
22.000000000000000

einstein% tdm -
2 3
1 2 3 4 5 6
tdm: Input array not square
```

Source file: cray_cshrc

```
#####
# This is ~/.cshrc_user. The Cray system administrators
# recommend that ~/.cshrc remain unmodified.
#####
setenv HOMEMWC ~phaz337
set path=( $path . . . \
    $HOMEMWC/bin $HOMEMWC/scripts \
    /usr/ucb /local/bin /bin /usr/bin /usr/ucb /local/uns \
    /usr/bin/X11 /local/bin/X11)

umask 022

# Define some environment variable for communication with 'make'
setenv F77      'cf77'
setenv F77PP   'f77transcray'
setenv F77CFLAGS '-c'
setenv F77FLAGS '-g'
setenv F77LFLAGS "-L$HOME/lib"
setenv LIBBLAS

if ( ( ! $?ENVIRONMENT ) && ( $?prompt ) ) then
setenv VSHOST 'Rhost'
setenv DISPLAY 'Rhost':0
set prompt="hostname' \> "

# Define some aliases
source ~/.aliases
endif
```

Source file: cray_output

```
#####
# Illustration of general technique for porting Fortran
# code to Cray J90. It is highly recommended that you port
# only thoroughly tested codes to minimize the amount of
# debugging etc. you need to do on the J90
#####

#####
# All class members should be able to 'rlogin' into 'charon'
# via
#
# % rlogin charon.cc.utexas.edu -l phas761
#
# from any of the course machines. E-mail me if you can't.
# From your account on einstein, this can be abbreviated to
#
# % rlogin phas761@charon
#
# NOTE: Use ONLY 'rlogin' (not 'telnet') to connect to the
# J90 since I will NOT be distributing a password for the
# account.
#####
charon% who am i
phas761      tty037      Sep 28 10:01 (newton.ph.utexas.edu)

charon% pwd
/home/utexas/ph/phas761

#####
# Each student has a working directory rooted in phas761's
# home directory. Note: The home directory has very limited
# space, so your directories are actually located on an
# archive partition which has essentially unlimited space.
# Your "top level" directories are actually symbolic links
# to the real directories.
#
# When using this Cray account, work ONLY within your
# own directory.
#####
charon% ls
Teaching@  jmolmes@  marcelo@  onager@  schaefer@
benton@    kelbird@  martin@   phy329@  slinger@
cray_output lamadorj@ matt@      rcrane@
doc@       lib@      mcarthur@ ristroph@
eromberg@  liwj@    moter@    sandor@

charon% ls -l phy329
lrwxrwxrwx 1 phas761 phas 33 \
Sep 28 09:51 phy329 -> /archive/utexas/ph/phas761/phy329/

#####
# 'tdm' example
#####
charon% cd ~/phy329/f77/ex7

#####
# Download files from the SGIs using 'ftp'
#####
charon% ftp einstein.ph.utexas.edu
Connected to einstein.ph.utexas.edu.

.
.
.
Remote system type is UNIX.
Using binary mode to transfer files.
Name (einstein.ph.utexas.edu:phas761): phy329
331 Password required for phy329.
Password:
230 User phy329 logged in.

ftp> cd f77/ex7
250 CWD command successful.

ftp> prompt
Interactive mode off.

ftp> mget *.f Makefile
200 PORT command successful.
150 Opening BINARY mode data connection for 'dmroutines.f' (7392 bytes).
```

```
226 Transfer complete.
7392 bytes received in 0.0031 seconds (2.3e+03 Kbytes/s)
200 PORT command successful.
150 Opening BINARY mode data connection for 'tdm.f' (2047 bytes).
226 Transfer complete.
2047 bytes received in 0.023 seconds (87 Kbytes/s)
200 PORT command successful.
150 Opening BINARY mode data connection for 'Makefile' (946 bytes).
226 Transfer complete.
946 bytes received in 0.0023 seconds (4e+02 Kbytes/s)

ftp> quit
221 Goodbye.

charon% ls
Makefile      dmroutines.f  tdm.f

#####
# The instructor-supplied script 'f77transcray' will
# convert Fortran source files to Cray-compatible form.
#
# WARNING: The script may fail if you violate any of the
# conventions discussed in class; particularly those having
# to do with the specification of real*8 constants!
#####
charon% f77transcray
usage: f77transcray <f77 source file> [<f77 source file> ...]

Converts 'canonical' double precision FORTRAN programs written for
a 32-bit machine to equivalent Cray FORTRAN. Source code is modified
in place, but original source is always saved in '.0' files.

#####
# Build the application. Note that 'f77transcray' is
# automatically invoked by the Makefile prior to compilation
# of any Fortran source file.
#####
charon% make
f77transcray tdm.f
f77transcray: Translating tdm.f
cf77 -g -c tdm.f
f77transcray dmroutines.f
f77transcray: Translating dmroutines.f
cf77 -g -c dmroutines.f
cf77 -g -L/home/utexas/ph/phas761/lib tdm.o dmroutines.o -lp329f -o tdm

#####
# Test the application
#####
charon% tdm
usage: tdm <file name>

Use 'tdm -' to read from standard input
STOP executed at line 60 in Fortran routine 'TDM'
CPU: 0.004s, Wallclock: 0.034s, 1.4% of 8-CPU Machine
Memory HWM: 401465, Stack HWM: 204161, Stack segment expansions: 0

charon% tdm -
2 2
1 2 3 4
2*2
7., 10., 15., 22.
STOP executed at line 52 in Fortran routine 'TDM'
CPU: 0.006s, Wallclock: 3.249s
Memory HWM: 411707, Stack HWM: 204161, Stack segment expansions: 0
```