```
c===========================================================
c     tsrand: Driver routine illustrating use of srand()
c     to "seed" the random number generator, rand(),
c     available on the SGIs.
c
c     Given seed >= 0 and, optionally, number of deviates to
c     generate, outputs
c
c       <i>   <random number>
c
c     i = 1 ... number of deviates on standard output.
c===========================================================
      program         tsrand
      implicit        none
c-----------------------------------------------------------
c     Uniform (on [0.0 .. 1.0]) random number generator.
c-----------------------------------------------------------
      real*8          rand
      integer         iargc,        i4arg
```

```
c-------------------------------------------------------------
c     Command-line arguments:
c
c     seed:     Integer-valued argument to srand() which
c               seeds the rand() random number generator.
c     n:        Number of deviates to generate
c-------------------------------------------------------------
      integer         seed,         n,
     &                              default_n
      parameter     (               default_n = 1 000 )
      integer         i

      if( iargc() .lt. 1 ) go to 900
      seed = i4arg(1,-1)
      if( seed .lt. 0 ) go to 900
      n    = i4arg(2,default_n)

      call srand(seed)
      do i = 1 , n
         write(*,*) i, rand()
      end do

      stop

 900  continue
         write(0,*) 'usage: tsrand <seed> [<n deviates>]'
      stop

      end
```

```
c===========================================================
c     nurand:  Uses SGI-specific uniform-random number
c     generator rand() to generate non-uniformly generated
c     random numbers on the interval [xmin..xmax]. User
c     must supply probability distribution function
c     having a header
c
c        subroutine pdf(x,pofx,maxpofx)
c
c     where 'x' is the input value, 'pofx' is the value
c     of the PDF evaluated at 'x' and 'maxpofx' is the
c     maximum value of the PDF (also a return argument).
c
c     Uses straight-forward algorithm based on area-under-
c     curve (PDF) idea--i.e. generate random point in
c     rectangle [xmin..xmax] x [0..maxpofx], accept point
c     and return x coordinate of point as random number
c     only if random point lies below PDF curve.
c===========================================================
      double precision function nurand(pdf,xmin,xmax)
         implicit      none

         external      pdf
         real*8        rand

         real*8        xmin,         xmax

         real*8        x,            y,
     &                 pofx,         maxpofx
```

```
c------------------------------------------------------------
c        Loop until a good deviate has been generated.
c------------------------------------------------------------
100      continue
c------------------------------------------------------------
c           Generate a uniform number in the interval xmin
c           to xmax.
c------------------------------------------------------------
            x  =  xmin + rand() * (xmax - xmin)
c------------------------------------------------------------
c           Evaluate PDF at x.
c------------------------------------------------------------
            call pdf(x,pofx,maxpofx)
c------------------------------------------------------------
c           Generate another uniform number in the interval
c           0 to maxpofx ...
c------------------------------------------------------------
            y = rand() * maxpofx
c------------------------------------------------------------
c           ... and accept the original random number, x,
c           if y < pofx.
c------------------------------------------------------------
            if( y .lt. pofx ) then
               nurand = x
               return
            end if
         go to 100

      end
```

```
c==============================================================
c     Sample probability distribution functions.
c==============================================================
c--------------------------------------------------------------
c     Generates uniform deviates.
c--------------------------------------------------------------
      subroutine puniform(x,pofx,maxpofx)
         implicit      none
         real*8        x,     pofx,    maxpofx

         maxpofx = 1.0d0
         if( 0.0d0 .le. x  .and.  x .le. 1.0d0 ) then
            pofx = 1.0d0
         else
            pofx = 0.0d0
         end if

         return
      end
c--------------------------------------------------------------
c     Generates gaussian-distributed (unit sigma) deviates.
c--------------------------------------------------------------
      subroutine pgauss(x,pofx,maxpofx)
         implicit      none
         real*8        normalize
c--------------------------------------------------------------
c        Normalization can be any non-zero value,
c        might as well be unity.  "True" normalization
c        is 1 / sqrt(Pi) = 0.5641 8958 3547 7563d0.
c--------------------------------------------------------------
         parameter   ( normalize = 1.0d0 )
         real*8        x,     pofx,    maxpofx

         maxpofx = normalize
         pofx    = normalize * exp(-x**2)

         return
      end
```

```
c===========================================================
c     usage: tnurand <xmin> <xmax> <n> [<nbin> <option>]
c===========================================================
c     tnurand: Driver program for nurand(). This driver
c     generates non-uniformly distributed random-numbers
c     using a user-specified distribution function.  The
c     program is currently set up with two distribution
c     functions (see 'pdfs.f'):
c
c     option = 0            --> uniform
c     option = 1 (default)  --> unit-sigma Gaussian
c
c     The routine calls nurand() to generate n random
c     numbers, then writes binned counts (the interval
c     xmin ... xmax is divided into nbin equal width bins)
c
c       <i>   <count i>
c
c     i = 1 ... nbin, on standard output.
c
c     Note that nurand() uses rand(), so srand() can be
c     called to "seed" nurand().
c===========================================================
      program         tnurand

      implicit        none

c-----------------------------------------------------------
c     External declarations for the user-defined PDFs and
c     declaration of nurand.
c-----------------------------------------------------------
      external        puniform,     pgauss
      real*8          nurand

      integer         iargc,        i4arg
      real*8          r8arg
      real*8          r8_never
      parameter     ( r8_never = -1.0d-60 )
```

```
c-------------------------------------------------------------
c     Command-line arguments:
c
c     xmin:   Minimum, maximum values of deviates
c     xmax:
c     n:      Number of deviates to generate
c     nbin:   Number of binning intervals
c     option: Selects probability distribution function
c-------------------------------------------------------------
      real*8           xmin,         xmax
      integer          n,            nbin,         option

      integer          max_nbin
      parameter      ( max_nbin = 10 000 )
      real*8           x(max_nbin),  count(max_nbin)

      real*8           dx,           rnum
      integer          i,            j

c-------------------------------------------------------------
c     Argument parsing.
c-------------------------------------------------------------
      if( iargc() .lt. 1 ) go to 900

      xmin   = r8arg(1,r8_never)
      if( xmin .eq. r8_never ) go to 900
      xmax   = r8arg(2,r8_never)
      if( xmax .eq. r8_never ) go to 900
      n      = i4arg(3,-1)
      if( n .le. 0 ) go to 900
      nbin   = min(i4arg(4,1000),max_nbin)
      option = i4arg(5,1)
```

```
c-----------------------------------------------------------
c     Set up bins and bin-coordinates (mid-points of bin
c     intervals).
c-----------------------------------------------------------
      dx = (xmax - xmin) / nbin
      do i = 1 , nbin
         count(i) = 0.0d0
         if( i .eq. 1 ) then
            x(1) = xmin + 0.5d0 * dx
         else
            x(i) = x(i-1) + dx
         end if
      end do

c-----------------------------------------------------------
c     Generate and bin random numbers.
c-----------------------------------------------------------
      do i = 1 , n
         if(      option .eq. 0 ) then
            rnum = nurand(puniform,xmin,xmax)
         else if( option .eq. 1 ) then
            rnum = nurand(pgauss,xmin,xmax)
         else
            write(0,*) 'tnurand: Unimplemented option ',
     &                 option
            stop
         end if
         j = min(int((rnum - xmin) / dx) + 1,nbin)
         count(j) = count(j) + 1.0d0
      end do

c-----------------------------------------------------------
c     Normalize bin counts.
c-----------------------------------------------------------
      do i = 1 , nbin
         count(i) = count(i) / (dx * n)
      end do
```

```
c-----------------------------------------------------------
c     Output bin counts.
c-----------------------------------------------------------
      call dvvto('-',x,count,nbin)

      stop

 900  continue
         write(0,*) 'tnurand: <xmin> <xmax> <n> '//
     &              '[<nbin> <option>]'
      stop

      end
```