Introduction to SPRNG

October 31, 2003

1 Framework to Use Random Number Generator

Each Random Number Generator (RNG) provides at least two basic functions for the programmer:

1. Initialization
2. Generation a random number

And the programmer *may* need to adjust the range of the number from RNG. The structure of the program would be:

1.1 Initialization

The RNG should be initialization once for each execution before any random number is output. The stream of (pseudo-) random number depends on the initialization parameters. The most important one may be the *seed*.

A program can regenerate the whole stream of random numbers in different executions by using the same seed during initialization. This is important for debug and investigation of program results.

1.2 Generation a random number

Usually, a random number is returned from a function. Some RNGs provide both integer and floating-point random numbers (like SPRNG). Some only provide integer random number (like C Standard Library). In both cases, the programmer may need to adjust the range of numbers.

The range of integer random number is usually between 0 and a large integer (which is defined as a macro). The range of floating-point random number is usually between 0 and 1.

1.3 Adjustment of range

The range of a random number `x` may need to adjust as,

- Integer → Integer. Let the range expected is between `a` and `b` (inclusively). The random number `x` can be modified by,

  ```
  y = a + (x % (b-a+1));
  ```

  However it may not work well for some old RNG since the lower-order bits are much less random than the higher-order bits. So in this case, `x` can be modified by,

  ```
  y = a + (int)((float)(b-a+1)*x/(RAND_MAX+1.0));
  ```

- Integer → Floating-point $[0, 1)$. The range of an integer random number can be adjusted to be between 0 and 1 (double precision) by,

```
y = (double)x / (RAND_MAX + 1);
```

- Floating-point $[0, 1)$ → Floating-point. Let the range be between `a` and `b`,

```
y = a + x * (b-a);
```

2 Review Of RNG In C Standard Library

C standard library provides two functions for integer random number,

- `void srand(unsigned int seed);`, which is the initialization of the RNG. The parameter `seed` is the seed of the stream.
- `int rand(void);`, which returns the next integer random number in the stream.

The programmer needs to adjust the range by himself.

3 SPRNG

SPRNG was developed in the Florida State University (http://www.sprng.cs.fsu.edu). SPRNG can generate the random numbers by different methods. It also supports several streams (*streams*) of random number in a program. Parallel program (by MPI) and FORTRAN program are also supported.

Version 2.0 provide three groups (interfaces) of functions.

3.1 Simple Interface

Charactistics: Only one stream on each process.

How to use: By defining the macro `SIMPLE_SPRNG` before including a SPRNG header file.

Initialization: -

```
int *init_sprng(int rng_type, int seed, int param);
```

- `rng_type`: the type of generator (`DEFAULT_RNG_TYPE`, `SPRNG_LFG`, `SPRNG_LCG`, `SPRNG_LCG64`, `SPRNG_CMRG`, `SPRNG_MLFG`, `SPRNG_PMLCG`).
- `seed`: the seed of generator
- `param`: the parameters of different generators.

Integer generation: -

```
int isprng();
```

Floating-point generation: -

```
double sprng();
```

Other: To print the information of current stream by,

```
int print_sprng();
```

To make a new seed using system date and time by,

```
int make_sprng_seed();
```

To pack the state of the random number stream into an array (so that you may save it in a file and continue in next execution) by,

```
int pack_sprng(char **buffer);
```

To unpack the state by,

```
int *unpack_sprng(char *buffer);
```

Example: .

```
#include <stdio.h>

#define SIMPLE_SPRNG    /* simple interface    */
#include "sprng.h"      /* SPRNG header file   */

#define SEED 985456376

main()
{
    int seed, i, irn;
    double rn;

    init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);
                /* initialize stream            */
    printf(" Print information about new stream:\n");
    print_sprng();


    printf(" Printing 3 random numbers in [0,1):\n");
    for (i=0;i<3;i++)
    {
        rn = sprng(); /* generate double precision random number */
        printf("%f\n",rn);
    }
```

```
    printf(" Printing 3 random integers in [0,2^31):\n");
    for (i=0;i<3;i++)
    {
        irn = isprng();     /* generate an integer random number        */
        printf("%16d\n",irn);
    }

}
```

3.2 Default Interface

Charactistics: More than one stream on any process.

How to use: No other macros

Initialization: -

```
int *init_sprng(int rng_type, int streamnum, int nstreams, int seed, int param)
```

- `rng_type`: the type of generator.
- `streamnum`: the stream number in $[0, \texttt{nstreams} - 1]$.
- `nstreams`: the number of distinct streams.
- `seed`: the seed, and should be the same for all streams.
- `param`: the parameters of different generators.
- *return value*: the ID of the stream if successfully.

Integer generation: -

```
int isprng(int *stream);
```

where `stream` is the ID of the stream.

Floating-point generation: -

```
double sprng(int *stream);
```

where `stream` is the ID of the stream.

Other: To print the information of a stream by,

```
int print_sprng(int *stream);
```

To make a new seed using system date and time by,

```
int make_sprng_seed();
```

To pack the state of a random number stream into an array (so that you may save it in a file and continue in next execution) by,

```
int pack_sprng(int *stream, char **buffer);
```

To unpack the state by,

```
int *unpack_sprng(char *buffer);
```

which returns an ID for the recreated stream.

To free the memory used to store information concerning the stream identified by the stream ID by,

```
int free_sprng(int *stream);
```

To spawn new streams from an old one by,

```
int spawn_sprng(int *stream, int nspawned, int ***newstreams);
```

where `nspawned` is the number of new streams to be spawned.

Example: -

```
#include <stdio.h>
#include <string.h>

#include "sprng.h"  /* SPRNG header file   */

#define SEED 985456376

main()
{
    int streamnum, nstreams, *stream;
    double rn;
    int irn;
    int i;
    int gtype;

    printf("Type in an integer for the generator type:  ");
    scanf("%d", &gtype);

    streamnum = 0;
    nstreams = 1;

    /* initialize stream */
    stream = init_sprng(gtype,streamnum,nstreams,SEED,SPRNG_DEFAULT);
    printf(" Print information about new stream:\n");
```

```
    print_sprng(stream);


    printf(" Printing 3 random numbers in [0,1):\n");
    for (i=0;i<3;i++)
    {
        rn = sprng(stream);
        printf("%f\n", rn);
    }

    free_sprng(stream);  /* free memory used to store stream state */
}
```

3.3 Interface With Pointer Checking

Charactistics: The stream ID is checked to be valid before using it in different functions.

How to use: By defining the macro `CHECK_POINTERS` before including a SPRNG header file.

Initialization: Same as default interface.

Integer generation: Same as default interface.

Floating-point generation: Same as default interface.

Other: Same as default interface.

Example: -

```
#include <stdio.h>
#include <string.h>

#define CHECK_POINTERS
#include "sprng.h"  /* SPRNG header file   */

#define SEED 985456376

main()
{
    int streamnum, nstreams, *stream, *wrong_stream;
    double rn;
    int irn;
    int i;
    int gtype;

    printf("Type in an integer for the generator type:  ");
    scanf("%d", &gtype);

    streamnum = 0;
    nstreams = 1;
```

```
    /* initialize stream */
    stream = init_sprng(gtype,streamnum,nstreams,SEED,SPRNG_DEFAULT);
    printf(" Print information about new stream:\n");
    print_sprng(stream);


    printf(" Printing 3 random numbers in [0,1):\n");
    for (i=0;i<3;i++)
    {
        rn = sprng(wrong_stream);
        printf("%f\n", rn);
    }

    free_sprng(stream);  /* free memory used to store stream state */
}
```

4 SPRNG With MPI

In order to make sure that SPRNG works properly with MPI, the macro `USE_MPI` has been defined before including a SPRNG header file.

```
....
#include <mpi.h>

#define SIMPLE_SPRNG
#define USE_MPI
#include <sprng.h>
....
```

In a parallel program, the random number stream should be different and independent for each process, even the seeds are the same. Moreover, for easy analysis and debug, the random number stream of a process should be the same for any number of processes in the program.

SPRNG can support both ideas in MPI program. In simple interface, `init_sprng(...)` makes some MPI calls to determine the number of processes and the rank of each process (so this function must be called after MPI initialization). This value is used to ensure that different streams are produced on different processes, even they have the same seed. Moveover, `make_sprng_seed` involves some inter-processes communication so that the same seed is returned to all processes. In default interface, distinct streams are returned based on their seed and the stream number. So if all processes use their ranks in MPI as the stream number, they can have different streams.

Example for simple interface:

```
#include <stdio.h>
#include <mpi.h>

#define SIMPLE_SPRNG
#define USE_MPI
#include <sprng.h>
```

```
#define SEED 985456376

main(int argc, char *argv[])
{
    double rn;
    int i, myid;
    int gtype;

    MPI_Init(&argc, &argv);       /* Initialize MPI */
    MPI_Comm_rank(MPI_COMM_WORLD, &myid); /* find process id */

    if(myid == 0)
    {
        printf("Type in an integer for the generator type:  ");
        scanf("%d", &gtype);
    }
    MPI_Bcast(&gtype,1,MPI_INT,0,MPI_COMM_WORLD ); /*--- broadcast gen type */

    init_sprng(gtype,SEED,SPRNG_DEFAULT); /* initialize stream  */
    printf("Process %d, print information about stream:\n", myid);
    print_sprng();

    for (i=0;i<3;i++)
    {
        rn = sprng(); /* generate double precision random number */
        printf("Process %d, random number %d: %.14f\n", myid, i+1, rn);
    }

    MPI_Finalize(); /* Terminate MPI  */
}
```

Example for default interface:

```
#include <stdio.h>
#include <mpi.h>/* MPI header file       */
#include <sprng.h>/* SPRNG header file     */

#define SEED 985456376

main(int argc, char *argv[])
{
    int streamnum, nstreams, *stream;
    double rn;
    int i, myid, nprocs;
    int gtype;

    MPI_Init(&argc, &argv); /* Initialize MPI   */
    MPI_Comm_rank(MPI_COMM_WORLD, &myid); /* find process id  */
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs); /* find number of processes  */

    streamnum = myid;
```

```
    nstreams = nprocs; /* one stream per processor                */

    if(myid == 0)
    {
        printf("Type in an integer for the generator type:  ");
        scanf("%d", &gtype);
    }
    MPI_Bcast(&gtype,1,MPI_INT,0,MPI_COMM_WORLD );

    stream = init_sprng(gtype,streamnum,nstreams,SEED,SPRNG_DEFAULT);
                    /* initialize stream */
    printf("\n\nProcess %d, print information about stream:\n", myid);
    print_sprng(stream);

    for (i=0;i<3;i++)
    {
        rn = sprng(stream); /* generate double precision random number */
        printf("Process %d, random number %d: %.14f\n", myid, i+1, rn);
    }

    free_sprng(stream); /* free memory used to store stream state  */

    MPI_Finalize(); /* Terminate MPI  */
}
```

5 Exercise

1. Modify the example of simple interface with MPI. Try to erase the statement `#define USE_MPI` and note the effects.

2. Modify the default interface examples (non-mpi and mpi) so that the program prints several streams.