

Random mystery



Module 2

Self study material

Operating systems 2018

1DT044 and 1DT096

Pseudorandom number generator

A pseudorandom number generator (PRNG), also known as a **deterministic** random bit generator is an **algorithm** for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers.

Pseudorandom number generator

- ★ A PRNG-generated sequence is not truly random, because it is completely determined by a relatively small set of initial values, called the PRNG's **seed** (which may include truly random values).
- ★ A PRNG algorithm will always produce the same sequence from a given starting point (seed).
- ★ Good statistical properties are a central requirement for the output of a PRNG.

Parent Process

Seed the PRNG

Use PRNG to **generate** a
“random” number.

Use **fork()** to create
child **n** processes.

A parent process seeds the PRNG and generates a “random” number. Next the parent forks n child processes, each generating a “random” number.

Child Process 1

Use PRNG to **generate** a
“random” number.

Child Process 2

Use PRNG to **generate** a
“random” number.

Child Process 3

Use PRNG to **generate** a
“random” number.



Child Process n

Use PRNG to **generate** a
“random” number.

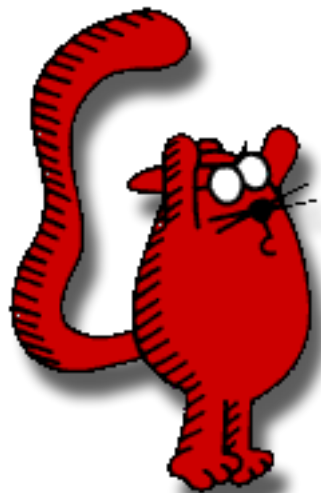
Parent Process

Seed the PRNG

Use PRNG to **generate** a
“random” number. → 127

Use **fork()** to create
child **n** processes.

The parent and all
children generates the
same “random”
number - why?



Child Process 1

Use PRNG to **generate** a
“random” number. → 127

Child Process 2

Use PRNG to **generate** a
“random” number. → 127

Child Process 3

Use PRNG to **generate** a
“random” number. → 127

•
•
•

Child Process n

Use PRNG to **generate** a
“random” number. → 127

Parent

```
srand(time(NULL));  
n = rand();  
fork();
```

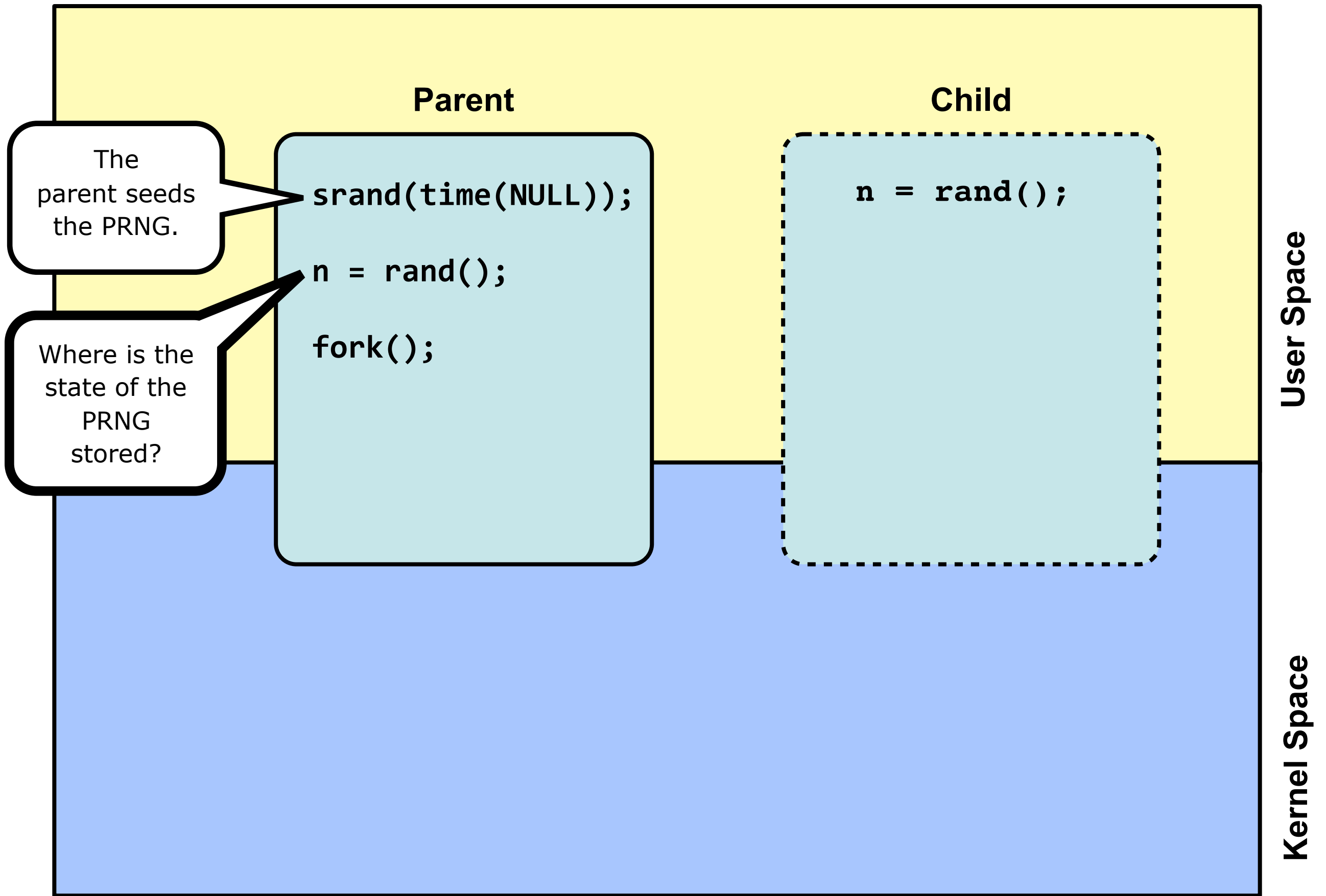
Child

```
n = rand();
```

User Space

Kernel Space

OS creates a new process, a child process
The child process is a copy of the parent
(copy of TEXT, DATA and Resources)



Parent

```
srand(time(NULL));  
  
n = rand();  
  
fork();
```

The state of the PRNG must be kept in a static variable somewhere in the memory area belonging to the parent process.

Child

```
n = rand();
```

Every child is a copy of the parent - including the state of the PRNG! Every child will get the same initial PRNG state and hence generate the same PRN (sequence).

User Space

Kernel Space