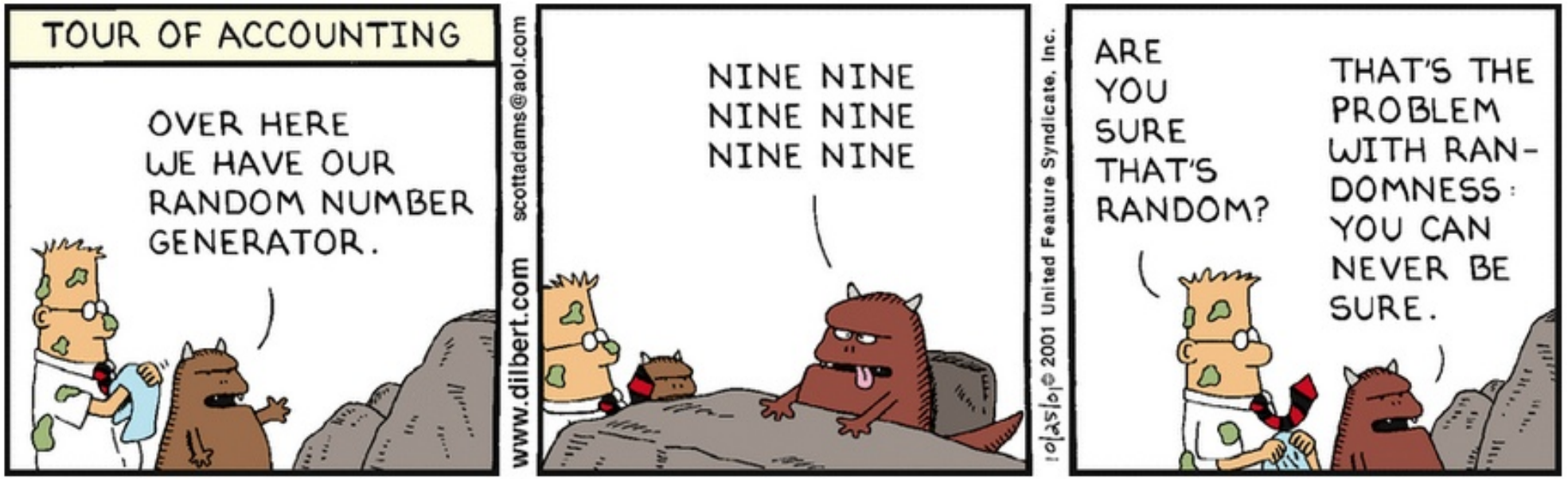
TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
www.dilbert.com scottadams@aol.com
NINE NINE NINE NINE NINE NINE NINE NINE
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.
© 2001 United Feature Syndicate, Inc.

stdlib.h

```
void srand(unsigned int seed)

int rand()
```

rand

Return uniformly distributed integers in [0,RAND_MAX]

C standard only requires RAND_MAX to be greater than 32767 ($2^{15}-1$)

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
printf ("RAND_MAX: %d\n", RAND_MAX);
  return 0;
}
```

srand

Set the initial seed of the PRG

Default seed 1

```
srand ( time (NULL))
```

Current time in seconds from epoch date

It modifies the internal state of the PRG

The internal state is implementation dependent

Example: Linear Congruential Generator

$$I_{n+1} = a\ I_n + c \pmod{n}$$

I_0 is the initial seed

The generated sequence is periodic with period not greater than n (can be way smaller than n for bad combinations of the parameters a, c and n)

The internal PRG state is accessed and modified by `rand()` calls.

No thread safe

```
int rand_r(unsigned int *seedp)
```

How to choose a different range:

If we want to generate integers in the range [A,B] we can use:

```
A+(int)((B-A+1)*rand()/(RAND_MAX+1.0))
```

Better than

```
A+(rand()%(B-A+1))
```

In some PRG last bits have a smaller period
It skews the uniform distribution (worse if the output range is closer to the input range)

WARNING: the values involved are BIG, so watch for overflows.

```
/* overflow problems example */
#include <stdio.h>      /* printf, NULL */
#include <stdlib.h>     /* srand, rand */
#include <time.h>       /* time */
int main ()
{
/* genero valori random nell'intervallo [0,20]*/
srand (time(NULL));
int f=rand();
/* risultato errato per overflow*/
int prova=(int)((21)*f/(RAND_MAX+1.0));
printf ("Random number: %d \nÈ sempre 0 perchè ho overflow, infatti il valore
random era %d e valoreRand/(RAND_MAX+1.0) è %f\n", prova,f,f/(RAND_MAX+1.0));

/*risultato corretto utilizzando double*/
double temp2=f/(RAND_MAX+1.0);temp2=21*temp2;
int res2=(int)temp2;
printf("Random number non sempre uguale a 0: %d \n",res2);
return 0;
}
```