

Forum:[Hack Forums](#)

Thread: [best way to generate a random username/password?](#)

pmtobefriend - October 31, 2018, 4:24 pm

the way I do it, it uses time of computer so I get the same unless I make my program sleep for around a second..

is there any faster way? (i thought about just generating one random username, and then reversing it, so x2 speed xd)

?

Souffrance - October 31, 2018, 5:48 pm

The line with:

Code `srand(time(NULL));` Should be in your main function, before the while loop. That way it will be random.

pmtobefriend - October 31, 2018, 5:50 pm

(Oct 31, 2018 05:48 PM)Souffrance Wrote: [\[CITE\]](#) The line with:

Code `srand(time(NULL));` Should be in your main function, before the while loop. That way it will be random.

thanks, this worked. Any explanation on this though? ?

Souffrance - October 31, 2018, 10:26 pm

(Oct 31, 2018 05:50 PM)pmtobefriend Wrote: [\[CITE\]](#) thanks, this worked. Any explanation on this though? ?

From Stackoverflow since I can't link till 50 posts.

Randomization is performed as a function that has a starting value, namely the seed.

So, for the same seed, you will always get the same sequence of values.

If you try to set the seed every time you need a random value, and the seed is the same number, you will always get the same "random" value.

Seed is usually taken from the current time, which are the seconds, as in `time(NULL)`, so if you always set the seed before taking the random number, you will get the same number as long as you call the `srand/rand` combo multiple times in the same second.

To avoid this problem, `srand` is set only once per application, because it is doubtful that two of the application instances will be initialized in the same second, so each instance will then have a different sequence of random numbers.

However, there is a slight possibility that you will run your app (especially if it's a short one, or a command line tool or something like that) many times in a second, then you will have to resort to some other way of choosing a seed (unless the same sequence in different application instances is ok by you). But like I said, that depends on your application context of usage.

SuckySucky - November 2, 2018, 8:49 pm

(Oct 31, 2018 10:26 PM)Souffrance Wrote: [\[CITE\]](#) From Stackoverflow since I can't link till 50 posts.

Randomization is performed as a function that has a starting value, namely the seed.

So, for the same seed, you will always get the same sequence of values.

If you try to set the seed every time you need a random value, and the seed is the same number, you will always get the same "random" value.

Seed is usually taken from the current time, which are the seconds, as in `time(NULL)`, so if you always set the seed before taking the random number, you will get the same number as long as you call the `srand/rand` combo multiple times in the same second.

To avoid this problem, `srand` is set only once per application, because it is doubtful that two of the application instances will be initialized in the same second, so each instance will then have a different sequence of random numbers.

However, there is a slight possibility that you will run your app (especially if it's a short one, or a command line tool or something like that) many times in a second, then you will have to resort to some other way of choosing a seed (unless the same sequence in different application instances is ok by you). But like I said, that depends on your application context of usage.

<https://stackoverflow.com/questions/1108780/why-do-i-always-get-the-same-sequence-of-random-numbers-with-rand>

There you go because you can't post the link! ?

0xDEAD10CC - November 4, 2018, 11:46 am

Then don't reset the seed in the loop to the system time which will reset the PRNG sequence all the time... The seed is used to set the starting sequence, if you set the seed again with the same value (the same representation of system time), obviously you're going to get the same number within the same timeframe because that number comes from the first number in the sequence.

spiderfruit - November 4, 2018, 12:56 pm

(Oct 31, 2018 04:24 PM)pmtobefriend Wrote: [\[CITE\]](#) the way I do it, it uses time of computer so I get the same unless I make my program sleep for around a second..

is there any faster way? (i thought about just generating one random username, and then reversing it, so x2 speed xd)

?

- no need to call `clear()` after you're done using that object (`std::string`), once username goes out of scope the destructor is called and it takes care of freeing the memory used
- I would pass `int` type parameter (used as length/size) and use for loop nice and clean I guess
- why create another string? `final_username`?

just my 2 cents

pmtobefriend - November 4, 2018, 12:58 pm

yes I removed the useless `clear`

and all
it works now just saying
just cant edit the thread
xd

incognito04 - November 14, 2018, 12:38 pm

I mean you could replace std:string with a "heap free" alternative like in this example I just cooked up. As an added bonus it uses std:random_shuffle for extra randomness ?

PHP Code

```
<?php
#include <array>
#include <iostream>
#include <numeric>
#include <algorithm>
#include <random>

template<size_t I = 1>
static std::array<std::array<char, 32>, I> generateRandom()
{
    std::random_device rd;
    std::mt19937 g(rd());

    std::array<std::array<char, 32>, I> data;

    for (auto& s : data)
    {
        std::iota(s.begin(), s.end(), 'A');
        std::shuffle(s.begin(), s.end(), g);
    }

    return data;
}

int main()
{
    const auto str = generateRandom<10>();

    for (const auto& s : str)
    {
        std::cout.write(s.data(), s.size());
        std::cout.put('\n');
    }

    return 0;
}
```

3DES - November 15, 2018, 12:19 pm

Use CryptGenRandom for secure random data

Sajuuk - November 16, 2018, 6:09 pm

You only need to seed the rand() function once per instance. That being said, yes prior to my post, there technically is no way to generate more user/pass combos any faster unless you want to get into further optimization. I guess the only "simple" way to make the program run "faster" is to run it on a faster system. Even then, there will be only a minuscule difference.

Either that, or... run two different instances, making sure the numbers don't overlap and the seed is different between the two. That, and make sure to exclude duplicate combinations.

0xDEAD10CC - November 17, 2018, 11:19 am

You could use one random value to generate all the data for a user and password in one, or perhaps even multiple user:pass credentials by splitting the random data up a bit more. Less calls for more work done = faster.

Sajuuk - November 17, 2018, 10:01 pm

(Nov 17, 2018 12:19 PM)0xDEAD10CC Wrote: [\[CITE\]](#) You could use one random value to generate all the data for a user and password in one, or perhaps even multiple user:pass credentials by splitting the random data up a bit more. Less calls for more work done = faster.

Faster or more efficient? I was hoping someone would catch on, lol.

But further into that topic, faster always implies better efficiency but the same program with different techniques/methods (or "algorithms", I guess) can generate more unique combinations than the other (which has the other, less efficient algorithms) even though they're both running at the same speed. That's what I meant by further optimization.

I didn't expect to elaborate that much on such a simple topic but I've already typed it up so, might as well post it along with this. And there are several ways to optimize combo generation. With split/completion you switch two halves of two different strings twice ("user" and "pass"), then include the originals (totaling six). You can reverse the strings themselves, adding another two (totaling eight). And additionally, you can switch the "user" and "pass" entirely, resulting in a grand total of ten possible combinations. And that's just with simple string manipulation. The possibilities are endless until you generate the first duplicate...

EDIT: You can't total six twice. Grand total is actually ten, lol.

0xDEAD10CC - November 17, 2018, 10:17 pm

(Nov 17, 2018 11:01 PM)Sajuuk Wrote: [\[CITE\]](#) Faster or more efficient? I was hoping someone would catch on, lol.

But further into that topic, faster always implies better efficiency but the same program with different techniques/methods (or "algorithms", I guess) can generate more unique combinations than the other (which has the other, less efficient algorithms) even though they're both running at the same speed. That's what I meant by further optimization.

I didn't expect to elaborate that much on such a simple topic but I've already typed it up so, might as well post it along with this. And there are several ways to optimize combo generation. With split/completion you switch two halves of two different strings twice ("user" and "pass"), then include the originals (totaling six). You can reverse the strings themselves, adding another two (totaling eight). And additionally, you can switch the "user" and "pass" entirely, resulting in a grand total of ten possible combinations. And that's just with simple string manipulation. The possibilities are endless until you generate the first duplicate...

EDIT: You can't total six twice. Grand total is actually ten, lol.

Faster doesn't always imply efficient though. Take sqrt calculations as an example, there are

approximation techniques that make the calculation faster but less accurate. It all depends on what you need; speed or precision. In his case he simply asked for faster, he didn't say how unique they need to be.

His main problem was not understanding how to get unique results at faster intervals than a second though because of his lack of understanding with seeding. Chances are now that he knows, just not seeding with the same value every iteration will make this fast enough for him.

There are a lot of bytes in rand() calls though that are being wasted with his naive approach above. As for the high and low byte entropy, I doubt for his purposes that he needs closer to uniform distribution.

Sajuuk - November 17, 2018, 11:16 pm

(Nov 17, 2018 11:17 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Faster doesn't always imply efficient though. Take sqrt calculations as an example, there are approximation techniques that make the calculation faster but less accurate. It all depends on what you need; speed or precision. In his case he simply asked for faster, he didn't say how unique they need to be.

His main problem was not understanding how to get unique results at faster intervals than a second though because of his lack of understanding with seeding. Chances are now that he knows, just not seeding with the same value every iteration will make this fast enough for him.

There are a lot of bytes in rand() calls though that are being wasted with his naive approach above. As for the high and low byte entropy, I doubt for his purposes that he needs closer to uniform distribution.

And with that being the case, assuming he doesn't need the combinations to be too unique, it's all about speed. I wasn't given anything to base string generation accuracy off of, so I just stated the obvious. And even at that, what's the basis to go by for "randomness", lol?

And thank you for the sqrt() example. In this case, efficiency revolves around combo count without regard to what exactly is being produced. With the sqrt() test, efficiency is also based on calculation speed but accuracy is also taken into consideration. I would not have put that into play otherwise, so I'll be more careful when I say "faster implies better efficiency". In the end, it does, but compromises might have to be made. It's just a user/pass generator though...