LOG IN

ARTICLES RESOURCES DOWNLOADS FREQUENTLY ASKED QUESTIONS

Creating random, secure passwords in Go

Go's random number generator is a great way to generate difficult-to-guess passwords.

By Mihalis Tsoukalos

May 23, 2018 | 6 Comments | 5 min read

Register or Login to like

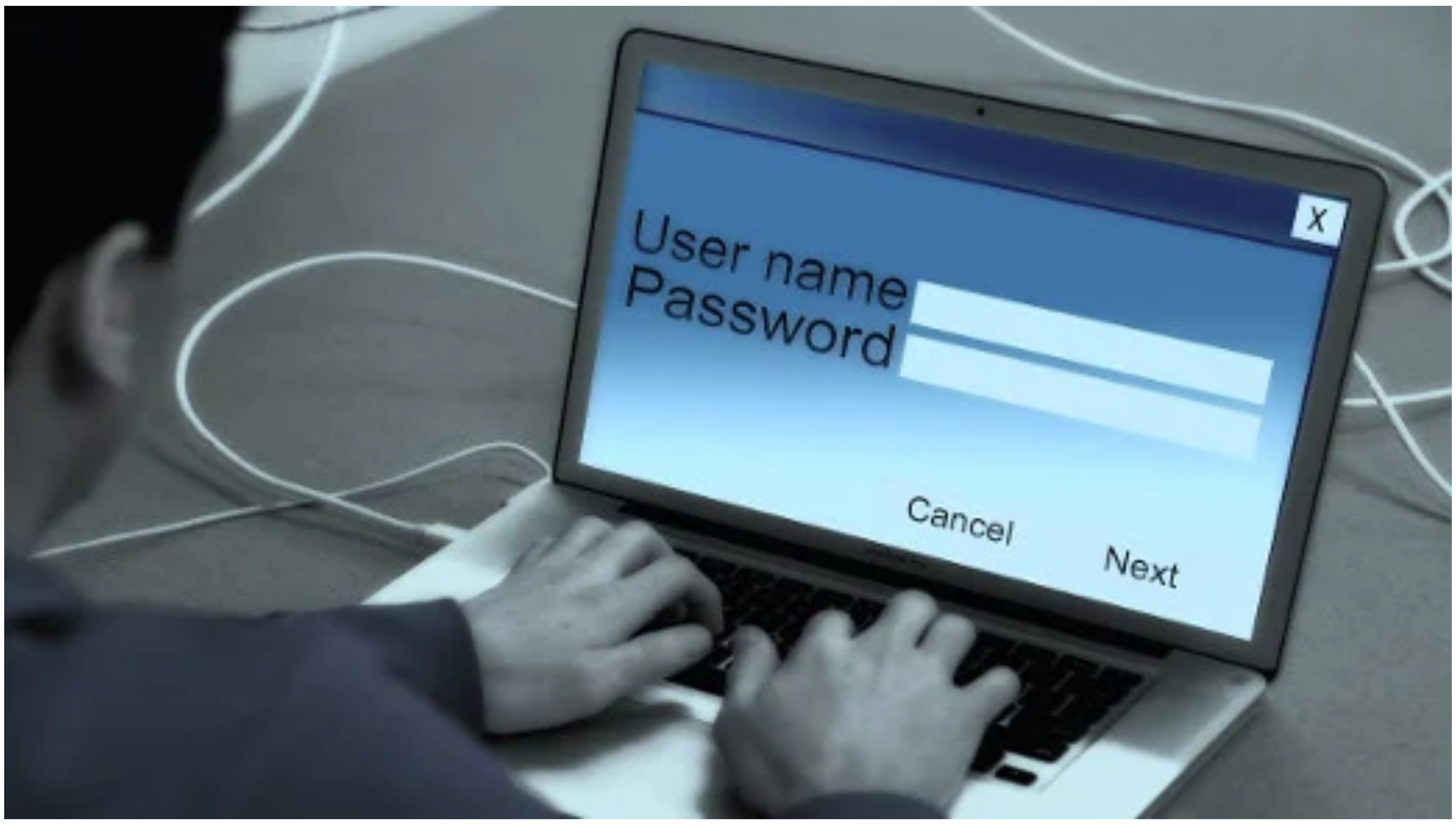


Image by: *geralt, via Pixabay. CC0.*

You can use the random number generator provided by the Go programming language to generate difficult-to-guess passwords comprised of ASCII characters. Although the code presented in this article is easy to read, it's best if you already know the basics of Go to understand it. If you're new to the programming language, take the Tour of Go to learn more, then come back here.

Before we get into the utilities and the code, take a look at this subset of the ASCII table as found in the output of the `man ascii` command:

```
    30  40  50  60  70  80  90  100  110  120
   ---------------------------------
  0:      (   2   <   F   P   Z   d    n    x
  1:      )   3   =   G   Q   [   e    o    y
```

```
2:    *  4  >  H  R  \  f   p   z
3: !  +  5  ?  I  S  ]  g   q   {
4: "  ,  6  @  J  T  ^  h   r   |
5: #  -  7  A  K  U  _  i   s   }
6: $  .  8  B  L  V  `  j   t   ~
7: %  /  9  C  M  W  a  k   u  DEL
8: &  0  :  D  N  X  b  l   v
9: '  1  ;  E  O  Y  c  m   w
```

The printable ASCII characters' decimal values range from 33 through 126; no other ASCII values are suitable for inclusion in passwords. Therefore, the utilities presented in this article will produce ASCII characters in that range.

Creating random integers

The first utility is named `random.go`, and it generates a specified number of random integers that reside in a given range. The most important part of `random.go` is this function:

```
func random(min, max int) int {
        return rand.Intn(max-min) + min
}
```

This function generates random integers that belong to a given range using the `rand.Intn()` Go function. Note that `rand.Intn()` returns a non-negative random number that belongs to `[0,n)`; the function will panic if its argument is a negative number. The panic message will be `panic: invalid argument to Intn`. You can find the documentation of the `math/rand` package at math/rand documentation.

The `random.go` utility accepts three command-line parameters: the minimum value of the generated integers, the maximum value, and the number of integers that will be generated.

Compiling and executing `random.go` will create this kind of output:

```
$ go build random.go
$ ./random
Usage: ./random MIX MAX TOTAL
$ ./random 1 3 10
2 2 1 2 2 1 1 2 2 1
```

If you wish to generate more secure random numbers in Go, use the `crypto/rand` package of the Go library.

Creating random passwords

The second utility, `randomPass.go`, generates the random passwords. `randomPass.go` uses the `random()` function to generate random numbers that will convert to ASCII characters using the following Go code:

```
for {
        myRand := random(MIN, MAX)
        newChar := string(startChar[0] + byte(myRand))
        fmt.Print(newChar)
        if i == LENGTH {
                break
        }
        i++
}
```

The value of MIN is **0** and the value of MAX is **94**, whereas the value of `startChar` is **!**, which is the first printable character in the ASCII table (with the decimal ASCII code of **33**). Therefore, all ASCII characters that will be generated are located after **!** and before the **~** character, which has a decimal ASCII code of **126**.

So, each random number that is generated is bigger than MIN, smaller than MAX, and converted into an ASCII character. The process keeps going until the generated password has the desired length.

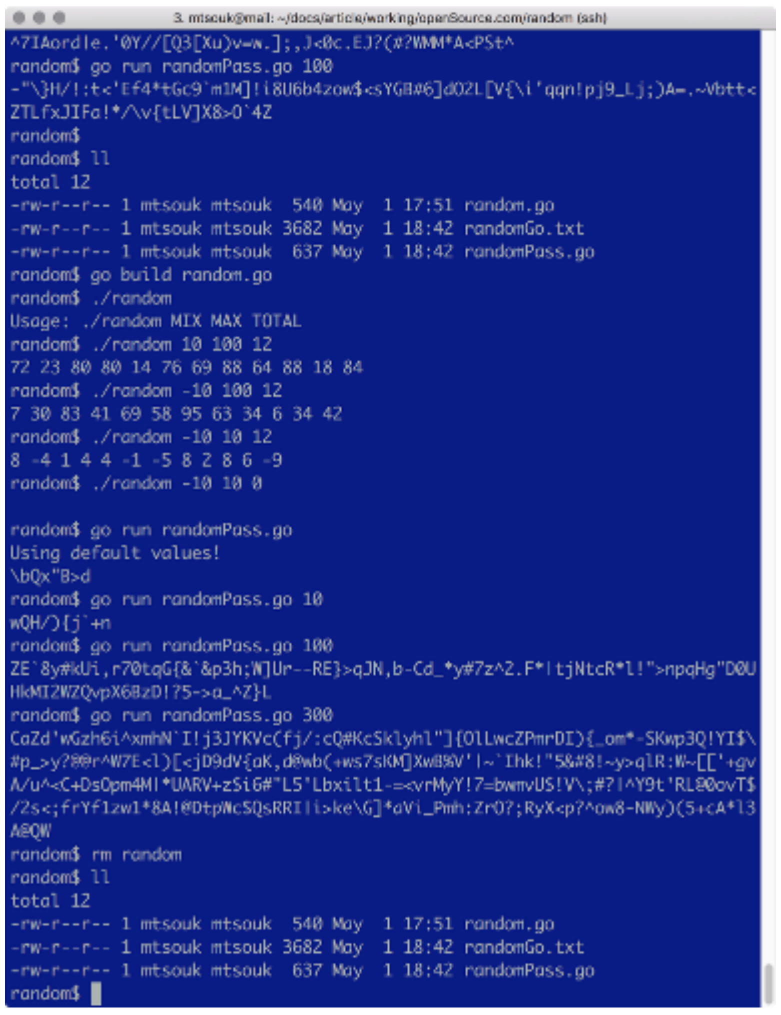
The `randomPass.go` utility accepts a single (optional) command-line parameter that defines the length of the generated password. Its default value is eight, which is a pretty common password length. Executing `randomPass.go` will generate the following kind of output:

```
$ go run randomPass.go 1
Z
$ go run randomPass.go 10
#Cw^a#IwkT
$ go run randomPass.go
Using default values!
[PP8@'Ci
```

Programming and development

- Red Hat Developers Blog
- Programming cheat sheets
- Try for free: Red Hat Learning Subscription
- eBook: An introduction to programming with Bash
- Bash Shell Scripting Cheat Sheet
- eBook: Modernizing Enterprise Java

One last detail: Don't forget to call `rand.Seed()` with a seed value in order to initialize the random number generator. If you use the same seed value all the time, the random number generator will create the same sequence of random integers.


3. mtsouk@mail: ~/docs/article/working/openSource.com/random (ssh)

```
^7IAordle.'0Y//[Q3[Xu)v=w.];,J<0c.EJ?(#?WMM*A<PSt^
random$ go run randomPass.go 100
-"\}H/!:t<'Ef4*tGc9`m1M]!i8U6b4zow$<sYG8#6]dO2L[V{\i'qqn!pj9_Lj;)A=.~Vbtt<
ZTLfxJIFa!*/\v{tLV]X8>0`4Z
random$
random$ ll
total 12
-rw-r--r-- 1 mtsouk mtsouk  540 May  1 17:51 random.go
-rw-r--r-- 1 mtsouk mtsouk 3682 May  1 18:42 randomGo.txt
-rw-r--r-- 1 mtsouk mtsouk  637 May  1 18:42 randomPass.go
random$ go build random.go
random$ ./random
Usage: ./random MIX MAX TOTAL
random$ ./random 10 100 12
72 23 80 80 14 76 69 88 64 88 18 84
random$ ./random -10 100 12
7 30 83 41 69 58 95 63 34 6 34 42
random$ ./random -10 10 12
8 -4 1 4 4 -1 -5 8 2 8 6 -9
random$ ./random -10 10 0

random$ go run randomPass.go
Using default values!
\bQx"B>d
random$ go run randomPass.go 10
wQH/){j`+n
random$ go run randomPass.go 100
ZE`8y#kUi,r70tqG{&`&p3h;W]Ur--RE}>qJN,b-Cd_*y#7z^Z.F*|tjNtcR*l!">npqHg"D0U
HkMIZWZQvpX6BzD!?5->a_^Z}L
random$ go run randomPass.go 300
CaZd'wGzh6i^xmhN`I!j3JYKVc(fj/:cQ#KcSklyhl"]{OlLwcZPmrDI){_om*-SKwp3Q!YI$\
#p_>y?8@r^W7E<l)[<jD9dV{aK,d@wb(+ws7sKM]Xw8%V'|~`Ihk!"5&#8!~y>qlR:W~[['+gv
A/u^<C+DsOpm4M|*UARV+zSi6#"L5'Lbxilt1-=<vrMyY!7=bwmvUS!V\;#?|^Y9t'RL@0ovT$
/2s<;frYf1zw1*8A!@DtpWcSQsRRI|i>ke\G]*aVi_Pmh:ZrO?;RyX<p?^ow8-NWy)(5+cA*l3
A@QW
random$ rm random
random$ ll
total 12
-rw-r--r-- 1 mtsouk mtsouk  540 May  1 17:51 random.go
-rw-r--r-- 1 mtsouk mtsouk 3682 May  1 18:42 randomGo.txt
-rw-r--r-- 1 mtsouk mtsouk  637 May  1 18:42 randomPass.go
random$ 
```

You can find both `random.go` and `randomPass.go` at GitHub. You can also execute it at play.golang.org.

I hope this has been helpful. If you have any questions, please leave a comment below or reach out to me on Twitter.

Tags: GO PROGRAMMING LANGUAGE PROGRAMMING

Mihalis Tsoukalos

Mihalis Tsoukalos is a Technical author, a UNIX Administrator and Developer, a DBA and a Mathematician. He is the author of Go Systems Programming and Mastering Go. You can reach him at http://www.mtsoukalos.eu/ and https://twitter.com/mactsouk.

More about me

6 Comments

These comments are closed, however you can **Register** or **Login** to post a comment on another article.

Song | May 23, 2018 Register or Login to like

Great article.
I don't understand the startChar[0] part, is it pointing to the character "!"?

Let's say I want to extend the program to include at least one special character I will need to set the MIN and MAX to 0 and 14?

If I want to limit to number only, I need MIN and MAX to be 15 and 24?
Thanks

Mihalis Tsoukalos | May 26, 2018 Register or Login to like

Hello!

Yes, startChar[0] points to the character "!", which is the first printable character in the ASCII table.

What kind of special character you want to include? The ASCII table contains non-printable characters, which will most likely not work in a password.

If you want to limit to numbers only, the MIN will still be 0 but the MAX should be 11. However, you will need to change the value of the startChar variable to "0".

vineet koul | May 27, 2018 Register or Login to like

I like your article on random numbers.Iam also working to evolve a secure password policy.At this time the level of security is very good as only a very talented programmer can break the random number password policy.Most of the users don't even know how to enter things on a computer.This is to be kept in mind.70% of people need help to even type things on a computer.For these people a very weak password policy is also tough to counter.This is actually the fact .There are exceptions in case of computer science students etc.However a computer virus can be damaging and not the people.

Mihalis Tsoukalos | May 29, 2018 Register or Login to like

Thanks for you comment - I am happy that you liked my article.

Mihalis Tsoukalos | June 26, 2018 Register or Login to like

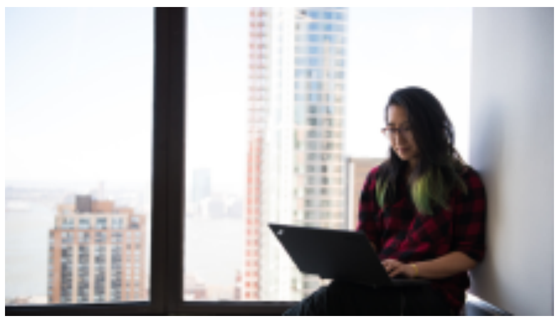
Added a random password generator utility named crypto/rand at https://github.com/mactsouk/opensource.com/ that uses crypto/rand.

Mihalis Tsoukalos | June 26, 2018 Register or Login to like

The name of the utility is cryptoRand.go :)

Related Content

C vs. Go: Comparing programming languages



16 reasons DDEV will be your new favorite web development environment

Packaging Job scripts in Kubernetes operators

ABOUT THIS SITE

CONTACT

Follow us on Twitter

Like us on Facebook

Watch us on YouTube

Follow us on Mastodon

RSS Feed