Oracle Password Generator

Norman Dunbar

Version 1.0, 2019-09-11

Oracle Password Generator

I got fed up with the in-house Oracle script (which I wrote/adapted!) which generates passwords - you have to be logged into a database, load the script and execute it. You can only do one password at a time, it doesn't generate a script, etc etc. This new utility *generatePassword* is a Windows and/or Linux command line replacement which can generate multiple passwords, write to a script etc. No database connection is required - until you wish to run the generated code.

Passwords generated will begin with a letter, upper or lower case, and will be followed by 14 additional characters made up of:

- Letters UPPER CASE - but without the letter 'O' as it looks like a zero in some fonts;
- Letters lower case - but without the letter 'l' as it looks like a digit one in some fonts;
- Digits 2 through 9 - to avoid mistaking zero for O and 1 for lower case L in some fonts);
- Special characters - '$', '_' or '#' which are the three permitted by Oracle to be used in passwords. Others *could* be used, but are not advised.

Generate a Single Password.

```
generatePassword
```

Which results in a single password:

```
-- Created using 'Oracle Password Generator'

"F6K#5b226r34T49"
```

You can of course, redirect this output to a file, but there's not much point.

Generate User Passwords.

```
generatePassword username [ username ...]
```

Which results in an `alter user` SQL statement for each user passed as parameters:

```
generatePassword fred barney wilma betty dino
```

```
-- Created using 'Oracle Password Generator'

alter user fred identified by "y2B##6aU9yx2I85" account unlock;
alter user barney identified by "u9d99So4XAmj9B8" account unlock;
alter user wilma identified by "i35h#Aan$I$b#u9" account unlock;
alter user betty identified by "z$#96E45_24nVx$" account unlock;
alter user dino identified by "HHi5mK9655#rwjs" account unlock;
```

You can redirect this output to a file, where it can then be run against the appropriate database:

```
generatePassword fred barney wilma betty dino > password.sql

sqlplus yourname/yourPassword@yourDatabase @password.sql

User altered.

User altered.

User altered.

User altered.

User altered.

SQL>
```

Compiling the Code

The code is written in such a way as to be compilable on both Windows and Linux. If you have a Mac, well, who knows. All compilations have been done using `g++`, both on Linux where it is installed pretty much by default, and on Windows where I'm using the 32/64 bit version of `g++` 5.1.0 available from TDB. Your mileage may vary when compiling with the likes of Visual C++ etc. (Borland aka Embarcadero C++ does work though!)

Command Line Compiling - Windows

For Windows users, a CodeBlocks project file exists which can be used, but you will need to amend the build options to select your own compiler and output folders etc.

If you wish to compile using command line tools instead, then assuming that the TDM version of `g++` is on your path, do this for a release version (no debugging information):

```
mkdir -P bin\Release
g++ -O3 -o bin\Release\generatePassword.exe -std=c++11 main.cpp
```

or this following for a debug version:

```
mkdir -p bin\Debug
g++ -O0 -g -o bin\Debug\generatePassword.exe -std=c++11 main.cpp
```

Obviously, you only need to create the directories once.

For Borland/Embarcadero users, and assuming that the compiler `bcc32x` is on the path:

```
mkdir -P bin\Release
bcc32x -O3 -o bin\Release\generatePassword.exe -std=c++11 main.cpp
```

or this following for a debug version:

```
mkdir -p bin\Debug
bcc32x -O0 -g -o bin\Debug\generatePassword.exe -std=c++11 main.cpp
```

There's a different version of the Borland compiler, `bcc32c` which takes different command line options to do the same as the above. I prefer the `bcc32x` version as it takes almost all the same options as the `g++` compiler does. Standards!

Command Line Compiling - Linux

The supplied CodeBlocks project file can be used under Linux as well as under Windows. You will still need to amend the build options to select your own compiler and output folders etc.

If you wish to compile using command line tools instead, then it's *almost* the same as for Windows above.

For a release version:

```
g++ -O3 -o generatePassword -std=c++11 main.cpp
strip generatePassword
```

And for a debug version:

```
g++ -O0 -g -o generatePassword -std=c++11 main.cpp
```

You can then `mv` or `cp` the output file(s) to somewhere on your `$PATH`.