# Password Generator.

# Password Generator

The Password Generator is a command-line utility written in C that allows you to generate random passwords with different combinations of characters, including uppercase letters, lowercase letters, special characters, and digits. The utility provides various options to customize the generated passwords based on your requirements.

## Features

- Generate random passwords with specified lengths and character combinations.

- Customize the number of uppercase letters, lowercase letters, special characters, and digits in the passwords.

- Option to generate multiple different combinations of passwords.

## Usage

To run the password generator, you can use the following command-line arguments:

```
./pwd -l <length> -u <num_upper> -w <num_lower> -s <num_special> -n <num_digits>
-c <num_combinations>
```

- `-l <length>`: Specify the total length of the generated password.

- `-u <num_upper>`: Specify the number of uppercase characters in the password.

- `-w <num_lower>`: Specify the number of lowercase characters in the password.

- `-s <num_special>`: Specify the number of special characters in the password.

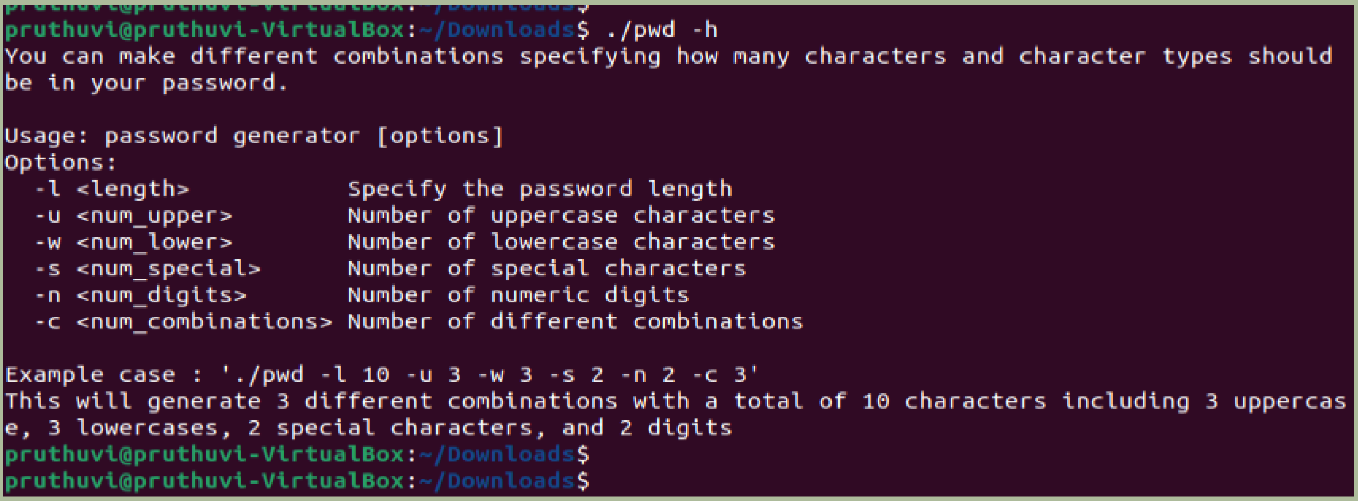
- `-n <num_digits>`: Specify the number of numeric digits in the password.

- `-c <num_combinations>`: Specify the number of different combinations of passwords to generate (optional, default is 1).

## Help

If you use the following command line argument, you can see a guideline:

```
./pwd --help

./pwd -h
```

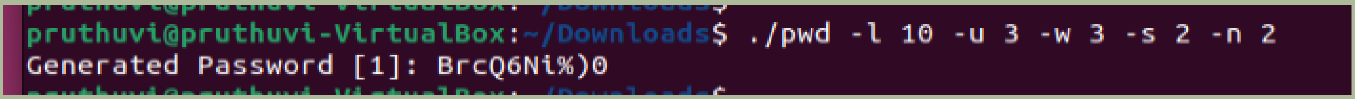


```
pruthuvi@pruthuvi-VirtualBox:~/Downloads$ ./pwd -h
You can make different combinations specifying how many characters and character types should
be in your password.

Usage: password generator [options]
Options:
  -l <length>           Specify the password length
  -u <num_upper>        Number of uppercase characters
  -w <num_lower>        Number of lowercase characters
  -s <num_special>      Number of special characters
  -n <num_digits>       Number of numeric digits
  -c <num_combinations> Number of different combinations

Example case : './pwd -l 10 -u 3 -w 3 -s 2 -n 2 -c 3'
This will generate 3 different combinations with a total of 10 characters including 3 uppercas
e, 3 lowercases, 2 special characters, and 2 digits
pruthuvi@pruthuvi-VirtualBox:~/Downloads$
pruthuvi@pruthuvi-VirtualBox:~/Downloads$
```

## Examples

1. Generate a password with 10 characters, including 3 uppercase, 3 lowercase, 2 special characters, and 2 digits:

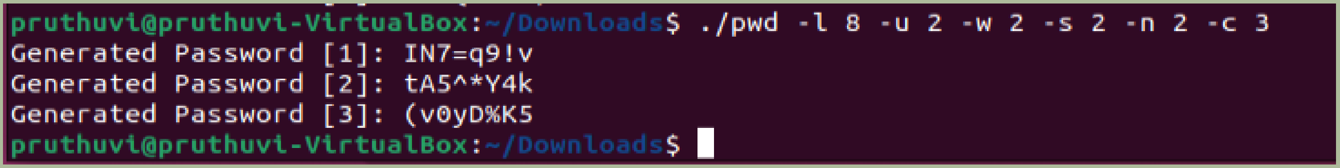
```
./pwd -l 10 -u 3 -w 3 -s 2 -n 2
```


```
pruthuvi@pruthuvi-VirtualBox:~/Downloads$ ./pwd -l 10 -u 3 -w 3 -s 2 -n 2
Generated Password [1]: BrcQ6Ni%)0
```

You don’t need to specify the number of combinations if you want only one combination because of we have set the default value for the number of combinations to 1.

2. Generate 3 different combinations of passwords, each with a total of 8 characters, including 2 uppercase, 2 lowercase, 2 special characters, and 2 digits:

```
./pwd -l 8 -u 2 -w 2 -s 2 -n 2 -c 3
```


```
pruthuvi@pruthuvi-VirtualBox:~/Downloads$ ./pwd -l 8 -u 2 -w 2 -s 2 -n 2 -c 3
Generated Password [1]: IN7=q9!v
Generated Password [2]: tA5^*Y4k
Generated Password [3]: (v0yD%K5
pruthuvi@pruthuvi-VirtualBox:~/Downloads$
```

## Building the Project

To build the Password Generator, make sure you have a C compiler (e.g., GCC) installed on your system. Then, use the following command:

```
gcc password_generator.c -o pwd
```

Or

```
gcc -o pwd password_generator.c
```

Source code for Password Generator:

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <unistd.h>


char randomCharacter(const char *charSet) {

    int index = rand() % strlen(charSet);

    return charSet[index];

}


// Function to shuffle an array of characters randomly

void shuffle(char *array, int n) {

    if (n > 1) {

        for (int i = 0; i < n - 1; i++) {

            int j = i + rand() / (RAND_MAX / (n - i) + 1);

            char temp = array[j];

            array[j] = array[i];

            array[i] = temp;

        }

    }

}


char *GenerateCharSet(char *string, int length, const char *charSet) {

    int charSetLength = strlen(charSet);


    // Create a temporary character array for shuffling

    char tempCharSet[charSetLength + 1];

    strcpy(tempCharSet, charSet);


    // Shuffle the characters randomly

    shuffle(tempCharSet, charSetLength);
    // Generate the password with the shuffled characters
    for (int i = 0; i < length; ++i) {
        string[i] = tempCharSet[i % charSetLength];
    }
    string[length] = '\0';
    return string + length;
}

void printUsage() {
    printf("Usage: password generator [options]\n");
    printf("Options:\n");
    printf("  -l <length>          Specify the password length\n");
    printf("  -u <num_upper>       Number of uppercase characters\n");
    printf("  -w <num_lower>       Number of lowercase characters\n");
    printf("  -s <num_special>     Number of special characters\n");
    printf("  -n <num_digits>      Number of numeric digits\n");
    printf("  -c <num_combinations> Number of different combinations\n");
}

int main(int argc, char *argv[]) {
    if (argc > 1 && (strcmp(argv[1], "--help") == 0 || strcmp(argv[1], "-h") == 0)) {
        printf("You can make different combinations specifying how many characters and character
types should be in your password.\n\n");
        printUsage();
        printf("\nExample case : './pwd -l 10 -u 3 -w 3 -s 2 -n 2 -c 3'\nThis will generate 3 different
combinations with a total of 10 characters including 3 uppercase, 3 lowercases, 2 special
characters, and 2 digits\n");
        exit(EXIT_FAILURE);
    }

    const char *lowercaseChars = "abcdefghijklmnopqrstuvwxyz";
    const char *uppercaseChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    const char *digitChars = "0123456789";
    const char *specialChars = "!@#%^&*()_+=";
```

```
int passwordLength = 0;
int NumOfUpper = 0;
int NumOfLower = 0;
int NumOfSpecial = 0;
int NumOfNum = 0;
int NumOfPwd = 1; // Default value for the number of different combinations


int opt;
while ((opt = getopt(argc, argv, "l:u:w:s:n:c:")) != -1) {
   switch (opt) {
   case 'l':
      passwordLength = atoi(optarg);
      break;
   case 'u':
      NumOfUpper = atoi(optarg);
      break;
   case 'w':
      NumOfLower = atoi(optarg);
      break;
   case 's':
      NumOfSpecial = atoi(optarg);
      break;
   case 'n':
      NumOfNum = atoi(optarg);
      break;
   case 'c':
      NumOfPwd = atoi(optarg);
      break;
   default:
      printUsage();
      exit(EXIT_FAILURE);
   }
}
```

```
// Check if all required options are provided
    if (passwordLength <= 0 || NumOfUpper < 0 || NumOfLower < 0 || NumOfSpecial < 0 ||
NumOfNum < 0) {
        printf("Error: Invalid arguments.\n");
        printUsage();
        exit(EXIT_FAILURE);
    }
    if (NumOfUpper + NumOfLower + NumOfSpecial + NumOfNum != passwordLength) {
        printf("Error: Password length does not match with the provided number of
characters.\n");
        printUsage();
        exit(EXIT_FAILURE);
    }

    srand(time(NULL));

    for (int i = 0; i < NumOfPwd; i++) {
        char password[100] = "";
        int totalChars = NumOfUpper + NumOfLower + NumOfSpecial + NumOfNum + 1; // +1 for
'\0'
        char charSet[totalChars];

        // Separate character sets for different types of characters
        char uppercaseSet[NumOfUpper + 1];
        char lowercaseSet[NumOfLower + 1];
        char specialSet[NumOfSpecial + 1];
        char digitSet[NumOfNum + 1];
        // Generate separate character sets for each type of character
        GenerateCharSet(uppercaseSet, NumOfUpper, uppercaseChars);
        GenerateCharSet(lowercaseSet, NumOfLower, lowercaseChars);
        GenerateCharSet(specialSet, NumOfSpecial, specialChars);
        GenerateCharSet(digitSet, NumOfNum, digitChars);
// Concatenate the character sets to form the main charSet
        strcpy(charSet, uppercaseSet);
        strcat(charSet, lowercaseSet);
        strcat(charSet, specialSet);
        strcat(charSet, digitSet);

        // Generate the password
        GenerateCharSet(password, passwordLength, charSet);

        // Print the generated password
        printf("Generated Password [%d]: %s\n", i + 1, password);
    }

    return 0;
}
```

## Author

- [Pruthuvi Samod](https://github.com/pruthuvi01)

Password_generator - https://github.com/pruthuvi01/password_generator