ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра вычислительных систем

КУРСОВАЯ РАБОТА
по дисциплине «Технологии разработки программного обеспечения»
на тему «Password Generator»

Выполнил:
ст. гр. ИВ-122
Клепче Г.В.

Проверил:
ст. преподаватель
Токмашева Е. И.

Новосибирск, 2022

Оглавление

Введение и постановка задачи.....2
Цель:.....2
Задание:.....2
Техническое задание 3
Функциональность проекта:.....3
Формат входных данных: 3
Интерфейс приложения:.....3
Описание выполненного проекта 5
Выполненный план работ :.....5
Пример работы: 5
Личный вклад в проект 6
Приложение. Текст программы.....7

Введение и постановка задачи

Цель:

Работая в команде и используя GitHub создать законченный программный продукт.

Задание:

- Сформулировать техническое задание (ТЗ) — документ, содержащий набор требований к проекту.
  На первой итерации ТЗ должно включать как минимум следующие пункты.

1. Функциональность проекта. Описание с точки зрения пользователя: какие задачи решает продукт, какие покрывает сценарии использования.
2. Формат входных данных.
3. Интерфейс приложения. В каком режиме работает приложение (интерактивный или нет, фоновый процесс, сетевой сервис и т. д.). Какие элементы интерфейса предусмотрены, их поведение.
4. Если приложение принимает аргументы командной строки, то их формат и описание.
5. Если предполагается использовать чтение исходных данных извне программы: конфигурационного файла, базы данных, источников в Интернет и т.д., то необходимо описание формата / протокола взаимодействия.

- Составить план работ После составления ТЗ необходимо декомпозировать проект на ряд задач. Каждая задача должна быть достаточно конкретизирована, чтобы участники команды понимали ее содержание, DoD (definition of done) и могли оценить сроки ее выполнения.

Техническое задание

Требование к функциональным характеристикам

Сгенерировать пароль из случайных символов с заданными пользователем параметрами.

Выполнение программы:

Перед началом пользователю необходимо запустить программу с указанием нужных ему параметров запуска. После чего программа выдает случайно сгенерированный пароль соответствующий параметрам запуска. Если пользователь ввел неправильные параметры запуска, то программа выведет сообщение о неправильности ввода параметров. Если пользователь не ввел длину пароля, то в программе задана длина пароля по умолчанию.

Формат входных данных:

Пользователь вводит параметры запуска: опции, длину пароля.

Используемые опции:

```
-l, --length=LENGTH         Длина пароля. Если не указан, то
                             он будет установлен в значение по-умолчанию.
                             Если указать 0, то программа запросит ввод с клавиатуры.

-c, --count=COUNT            Количество паролей за один вызов программы.

-P, --pattern=PATTERN        Строка, содержащая возможные символы для генерации пароля.
                             Если ключ указан, то ключи
                             -u, -d, -n, -s будут проигнорированы.

-u, --letter-up              Использовать английские буквы в верхнем регистре
-d, --letter-down            Использовать английские буквы в нижнем регистре
-n, --number                 Использовать цифры
-s, --symbol                 Использовать символы и знаки препинания
-e, --enable-space           Добавить пробел в шаблон пароля
```

Примеры использования:

```
./passgen -l0 -u              # Ввести длину пароля с клавиатуры и использовать буквы в врехне
./passgen -l25 -d             # Использовать только английские буквы в нижнем регистре
./passgen -l25 -s             # Использовать только символы и знаки препинания
./passgen -l25 -dn            # Использовать только английские буквы в нижнем регистре
./passgen -l25 -du            # Использовать только английские буквы в нижнем и верхнем регистре
./passgen -l25 -P123abc       # Использовать только перечисленные символы: '123abc'
./passgen -l25 -P123abc -e    # Использовать только перечисленные символы: '123abc' и пробел
```

Интерфейс:

Запуск программы и управление ей осуществляются через терминал Linux.

Пример работы:

При запуске программы пользователь должен указать длину пароля и аргументы.
В ходе работы генерируется пароль указанной длины с учетом указанных аргументов (рис.1).

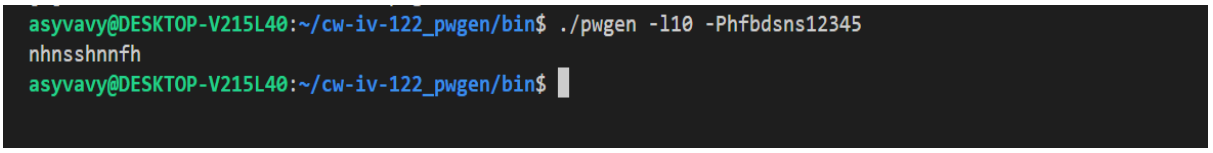

Рис.1

Если не указана длина пароля вручную, будет указана длина, заданная автоматически (25символов) (рис.2).

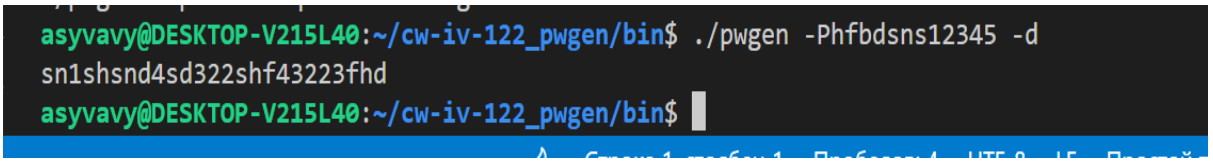

Рис.2

Если не было указано хотя бы одного из обязательных аргументов, программа выведет сообщение об ошибке и покажет, какие аргументы являются обязательными (рис.3).

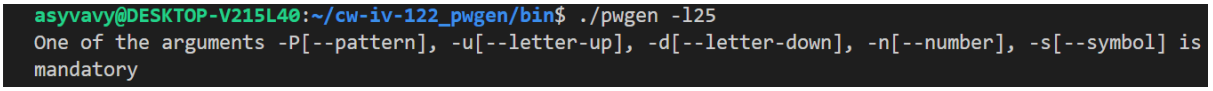

Рис.3

Описание выполненного проекта

Была разработана программа pwgen, которая реализовывает рандомную генерацию паролей пользователю.

Был написан основной код программы, в котором были реализованы:

main.c:

Принимает на вход аргументы, обрабатывает их, обрабатывает ошибки, после чего вызывает функцию генерации пароля.

passgen.c:

В данной функции происходит генерацию паролей, где функция getchar, генерирует случайную букву, цифру соответствующих параметрам.

Конечный результат записывается в переменную "*result*", которая после возвращается функцией.

print.c:

Функция необходимая для вывода справочной информации пользователю, когда он использует аргумент "**-h**" при запуски программы.

valid.h:

Проверяет, что символ является цифрой.

Выполненный план работы (рис.4):

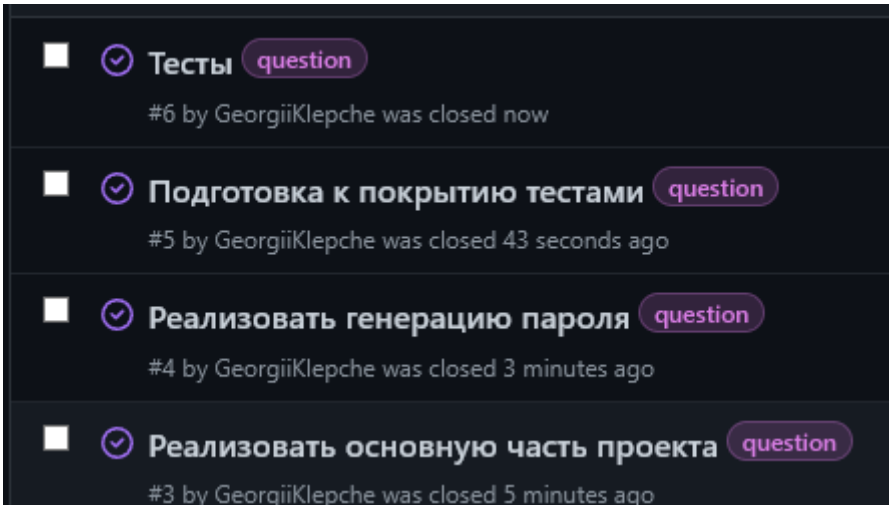

Рис.3

Личный вклад в проект:
Мной было написано Техническое задание (приложение) и составлен план работ и распределены задачи.
Также мной были реализованы

- Был разработан .gitignore.
- Функция main.c(так же main.h):
  Которая принимает на вход значения командной строки, обрабатывает их и передает в pwgen.c
- Функция argument.c(также argument.h).
- Был разработан config.h, в котором хранятся все константы необходимые для работы программы
- Был разработан Makefile, который собирает программу.

Приложение:
main.c

```
#include <libpwgen/arguments.h>
#include <libpwgen/passgen.h>
#include <pwgen/print.h>
#include <libpwgen/valid.h>


int main (int argc, char * const argv[])
{
	srand((unsigned) time(NULL) * getpid());

	textdomain(PACKAGE_NAME);

	int length = DEFAULT_PASSWORD_LENGTH;
	int count = 1;
	int useCount = 0;
	int usePattern = 0;
	char *password;
	char *source = malloc(MAX_PATTERN_LENGTH + 1);
	*source = '\0';
	int iargs=0;
	int option_index;

	const char *short_options = "hvl:c:P:udnse";
	const struct option long_options[] = {
		{ArgKeys[0].name, optional_argument, NULL, ArgKeys[0].sname},
		{ArgKeys[1].name, optional_argument, NULL, ArgKeys[1].sname},
		{ArgKeys[2].name, required_argument, NULL, ArgKeys[2].sname},
		{ArgKeys[3].name, required_argument, NULL, ArgKeys[3].sname},
		{ArgKeys[4].name, required_argument, NULL, ArgKeys[4].sname},
		{ArgKeys[5].name, optional_argument, NULL, ArgKeys[5].sname},
		{ArgKeys[6].name, optional_argument, NULL, ArgKeys[6].sname},
		{ArgKeys[7].name, optional_argument, NULL, ArgKeys[7].sname},
		{ArgKeys[8].name, optional_argument, NULL, ArgKeys[8].sname},
		{ArgKeys[9].name, optional_argument, NULL, ArgKeys[9].sname},
		{NULL, 0, NULL, 0}
	};


	while ( (iargs = getopt_long(argc, argv, short_options, long_options, &option_index)) != -1)
	{
		switch (iargs)
		{
			case 'h':
				Print.help();
				break;
			case 'v':
				Print.version();
				break;
			case 'l':
				if(isNumber(optarg) == 1)
				{
					// Пресекаем попытку ввести слишком большую длину
пароля
					if (atoi(optarg) > MAX_PASSWORD_LENGTH) {
						Print.error("%s: %d", _("Max password length"),
MAX_PASSWORD_LENGTH);
					}
```

```
52                                      // Если указана нулевая длина, то будет запрошен ввод с
53  клавиату
54                                      if (atoi(optarg) > 0) {
55                                              useCount = 1;
56                                      }
57                                      sscanf(optarg, "%d", &length);
58                              } else {
59                                      Print.error(_("Option %s must be a number!"),
60  ArgKeys[2].manstr);
61                              }
62                              break;
63                      case 'c':
64                              if(isNumber(optarg) == 1)
65                              {
66
67                                      // Пресекаем попытку ввести слишком большое
68  количество итераций
69
70                                      if (atoi(optarg) > MAX_PASSWORD_COUNT) {
71                                              Print.error("%s: %d", _("Max password count"),
72  MAX_PASSWORD_COUNT);
73                                      }
74                                      sscanf(optarg, "%d", &count);
75                              } else {
76                                      Print.error(_("Option %s must be a number!"),
77  ArgKeys[3].manstr);
78                              }
79                              break;
80                      case 'P':
81                              usePattern = 1;
82
83                              // Пресекаем попытку ввести слишком длинный шаблон
84                              if (strlen(optarg) > MAX_PATTERN_LENGTH) {
85                                      Print.error("%s: %d", _("Max pattern length"),
86  MAX_PATTERN_LENGTH);
87                              }
88                              sscanf(optarg, "%s", source);
89                              break;
90                      case 'u':
91                              if (usePattern != 1) {
92                                      strcat(source, PATTERN_UP);
93                              }
94                              break;
95                      case 'd':
96                              if (usePattern != 1) {
97                                      strcat(source, PATTERN_DOWN);
98                              }
99                              break;
100                     case 'n':
101                             if (usePattern != 1) {
102                                     strcat(source, PATTERN_NUMBER);
                                }
                                break;
                        case 's':
                                if (usePattern != 1) {
                                        strcat(source, PATTERN_SYMBOL);
                                }
```

```
                                        break;
                                case 'e':
                                        // * Пресекаем попытку ввести слишком длинный шаблон
                                        if ((strlen(source) + 1) > MAX_PATTERN_LENGTH) {
                                                char __e__[ERROR_MSG_BUFFER_LENGTH];
                                                sprintf(__e__, _("No use key %s with long pattern"),
ArgKeys[9].manstr);
                                                Print.error("%s: %d (%s)", _("Max pattern length"),
MAX_PATTERN_LENGTH, __e__);
                                        }
                                        strcat(source, " ");
                                        break;
                                case '?':
                                        exit(EXIT_FAILURE);
                                        break;
                                default:
                                        break;
                        };
                };
if (strlen(source) > 0)
{
                if (useCount > 0) {
                        while(count >= 1) {
                                password = PassGen.getPassword(source, length);
                                printf("%s\n", password);
                                count--;
                        }
                } else {
                        password = PassGen.getPassword(source, length);
                        printf("%s\n", password);
                }
} else {
                char __s__[ERROR_MSG_BUFFER_LENGTH];
                sprintf(__s__, "%s, %s, %s, %s, %s", ArgKeys[4].manstr, ArgKeys[5].manstr, ArgKeys[6].manstr,
ArgKeys[7].manstr, ArgKeys[8].manstr);
                Print.error(_("One of the arguments %s is mandatory"), __s__);
}

free(source);

exit(EXIT_SUCCESS);

}
```

passgen:

```c
#include <libpwgen/passgen.h>
#include <pwgen/print.h>
#include <libpwgen/valid.h>

char getChar(char* symbols)
{
   return symbols[rand() % strlen(symbols)];
}

char* getPassword(char* symbols, int length)
{
   int counter = 0;
   int isTyped = 0;
   char* result = malloc(length + 1);

   if (length < 1) {
      while (isTyped != 1) {
         int t;
         char ct[5] = {0};
         printf("%s:\n» ",
               _("Type in a password length. Use only 4 first symbols."));
         /**
         * Если не присвоить вызов scanf переменной,
         то компилятор выдаст warning
         */
         int _readchar = scanf("%4s", ct);

         /**
         Если не использовать переменную,
         то компилятор выдаст warning:
         */
         if (_readchar < 1) {
            // Аварийный выход сработает если, например,
            exit(EXIT_FAILURE);
         }

         // Если введенная строка состоит из цифр
         if (isNumber(ct) == 1) {
            // Конвертируем в число
            t = atoi(ct);
            // Нельзя иметь длину пароля < 0 )))
            if (t < 1) {
               isTyped = 0;
            } else {
               length = t;
               isTyped = 1;
            }
         } else {
            isTyped = 0;
         }
      }
```

```
45      }
46
47      if (length > MAX_PASSWORD_LENGTH) {
48          Print.error("%s: %d", _("Max password length"), MAX_PASSWORD_LENGTH);
49      }
50
51      while (counter < length) {
52          result[counter] = getChar(symbols);
53          counter++;
54      }
55
56      result[counter] = '\0'; // НАДО ИСПОЛЬЗОВАТЬ ОДИНАРНЫЕ КАВЫЧКИ
57      return result;
58  }

    const struct passgen PassGen = {.getPassword = getPassword, .getChar = getChar};
```

config.h

```
1   #define DEFAULT_PASSWORD_LENGTH 25
2
3   // Константа для ERROR_MSG_BUFFER_LENGTH
4   #define ERROR_MSG_BUFFER_LENGTH 300
5
6   // Константа для MAX_PASSWORD_COUNT
7   #define MAX_PASSWORD_COUNT 1000000
8
9   // Константа для MAX_PASSWORD_LENGTH
10  #define MAX_PASSWORD_LENGTH 4096
11
12  // Константа для MAX_PATTERN_LENGTH
13  #define MAX_PATTERN_LENGTH 200
14
15  // Константа для package description
16  #define PACKAGE_DESCRIPTION "Console password
17  generator"
18
19  // Константа для full name of this package.
20  #define PACKAGE_NAME "passgen"
21
22  // Константа для full name.
    #define PACKAGE_STRING "passgen"
```