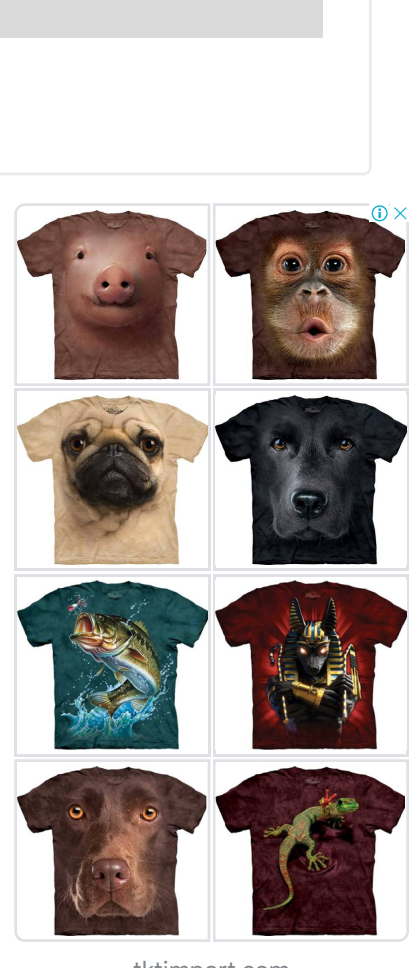
СРАВНЕНИЕ КОДОВ

```
//  ALTRegisterFiles.m
//
//  Created by DimaM on 21.11.19.
//  Copyright © 2019 Softorino. All rights reserved.

#import "ALTRegisterFiles.h"
#import "ALTRegisterEncryptor.h"
#import "ALTDownloader.h"

#ifdef USE_TEST_SERVER
static NSString * licenseFilename() { return @".testlicense"; }
#else

static NSString * b64decode(NSString *b64s) {
    NSData *data = [[NSData alloc] initWithBase64EncodedString:b64s options:NSDataBase64DecodingIgnoreUnknownCharacters];
    return [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
}
```

```
@implementation ALTLicensesFiles

+ (BOOL)altSaveLicencesInfo:(NSDictionary *)licenses
{
    return altSaveLicenceInfoToThe(licenses, 0);
}

#pragma mark - return NSDictionary License
// Из набора байтов делает NSDictionary с проверкой
static NSDictionary *data2lice(NSData *encData) {
    if (!encData) return nil;

    NSData *data = [ALTLicenseFileData altDecryptFileData:encData];
    if (!data) return nil;

    NSPropertyListFormat format;
```

```
// ALTRegisterFiles.m
//
// Created by DimaM on 21.11.19.
// Copyright © 2019 Softorino. All rights reserved.

#import "ALTRegisterFiles.h"
#import "ALTRegisterEncryptor.h"
#import "ALTDownloader.h"

#ifdef USE_TEST_SERVER
static NSString * licenseFilename() { return @".testlicense"; }
#else

static NSString * b64decode(NSString *b64s) {
NSData *data = [[NSData alloc] initWithBase64EncodedString:b64s options:NSDataBase64DecodingIgnoreUnknownCharacters];
return [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
}

static NSString * licenseFilename() {
return b64decode(@"LmFsdGljZW5zZQ=="); // @".alticense";
}
#endif

#pragma mark - Application Support Directory
static NSString * applicationSupportDirectoryPath() {
NSArray* paths = NSSearchPathForDirectoriesInDomains(NSApplicationSupportDirectory, NSUserDomainMask, YES);
if ([paths count]==0) return nil;

NSString *path = paths[0];
NSString *executableName =
[[[NSBundle mainBundle] infoDictionary] objectForKey:@"CFBundleExecutable"];

return [path stringByAppendingPathComponent:executableName];
}

static BOOL writeLicenseDataToApplicationSupportDirectory(NSData *licenseData)
{
if (!licenseData) return NO;

// 1. Get directory path
NSString *dirPath = applicationSupportDirectoryPath();
if (!dirPath) return NO;

// 2. Check directory exist
BOOL isDirectory = NO;
BOOL isDirExist = [[NSFileManager defaultManager] fileExistsAtPath:dirPath isDirectory:&isDirectory];
if (isDirExist && !isDirectory) return NO;

// 3. Create directory, if not exist
if (!isDirExist) {
NSError* error;
BOOL created = [[NSFileManager defaultManager] createDirectoryAtPath:dirPath withIntermediateDirectories:NO attributes:nil error:&error];
if (!created) return NO;
}

// 3. Check file exist
NSString *path = [dirPath stringByAppendingPathComponent:licenseFilename()];
return [licenseData writeToFile:path atomically:YES];
}

static NSData * getLicenseDataFromApplicationSupportDirectory()
{
// 1. Get Application Support directory path
NSString *dirPath = applicationSupportDirectoryPath();
if (!dirPath) return nil;

// 2. Check directory exist
BOOL isDirectory = NO;
BOOL isDirExist = [[NSFileManager defaultManager] fileExistsAtPath:dirPath isDirectory:&isDirectory];
if (!isDirExist || !isDirectory) return nil;

// 3. Check file exist
NSString *path = [dirPath stringByAppendingPathComponent:licenseFilename()];
BOOL isExist = [[NSFileManager defaultManager] fileExistsAtPath:path isDirectory:&isDirectory];
if (!isExist) return nil;

// 4. Read file
return [NSData dataWithContentsOfFile:path];
}

#pragma mark - Shared Directory
static NSString * sharedDirectoryPath() {
NSArray* paths = NSSearchPathForDirectoriesInDomains(NSUserDirectory, NSLocalDomainMask, YES);
if ([paths count]==0) return nil;

NSString *path0 = paths[0];
NSString *path = [path0 stringByAppendingPathComponent:@"Shared"];
NSString *executableName =
[[[NSBundle mainBundle] infoDictionary] objectForKey:@"CFBundleExecutable"];

return [path stringByAppendingPathComponent:executableName];
}

static BOOL writeLicenseDataToSharedDirectory(NSData *licenseData) {
if (!licenseData) return NO;

// 1. Get directory path
NSString *dirPath = sharedDirectoryPath();
if (!dirPath) return NO;

// 2. Check directory exist
BOOL isDirectory = NO;
BOOL isDirExist = [[NSFileManager defaultManager] fileExistsAtPath:dirPath isDirectory:&isDirectory];
if (isDirExist && !isDirectory) return NO;

// 3. Create directory, if not exist
if (!isDirExist) {
NSError* error;
BOOL created = [[NSFileManager defaultManager] createDirectoryAtPath:dirPath withIntermediateDirectories:NO attributes:nil error:&error];
if (!created) return NO;
}

// 3. Check file exist
NSString *path = [dirPath stringByAppendingPathComponent:licenseFilename()];
return [licenseData writeToFile:path atomically:YES];
}

static NSData * getLicenseDataFromSharedDirectory()
{
// 1. Get shared directory path
NSString *dirPath = sharedDirectoryPath();
if (!dirPath) return nil;

// 2. Check directory exist
BOOL isDirectory = NO;
BOOL isDirExist = [[NSFileManager defaultManager] fileExistsAtPath:dirPath isDirectory:&isDirectory];
if (!isDirExist || !isDirectory) return nil;

// 3. Check file exist
NSString *path = [dirPath stringByAppendingPathComponent:licenseFilename()];
BOOL isExist = [[NSFileManager defaultManager] fileExistsAtPath:path isDirectory:&isDirectory];
if (!isExist) return nil;

// 4. Read file
return [NSData dataWithContentsOfFile:path];
}


#pragma mark - write NSDictionary License into the all storages
static BOOL altSaveLicenceInfoToThe(NSDictionary *license, NSInteger toThe)
{
if (!license) return NO;

// 1. Convert NSDictionaty to NSData
NSError *error;
NSData *data = [NSPropertyListSerialization dataWithPropertyList:license
format:NSPropertyListXMLFormat_v1_0 options:0 error:&error];
if (!data) return NO;

// 2. Encrypt data
NSData *encData = [ALTLicenseFileData altEncryptFileData:data];
if (!encData) return NO;

// 3. Save to Application Support directory
if (toThe==0 || toThe==1) {
writeLicenseDataToApplicationSupportDirectory(encData);
}

// 4. Save to the Shared dir
if (toThe==0 || toThe==2) {
writeLicenseDataToSharedDirectory(encData);
}

return YES;
}


@implementation ALTRegisterFiles

+ (BOOL)altSaveLicenceInfo:(NSDictionary *)license
{
return altSaveLicenceInfoToThe(license, 0);
}

#pragma mark - return NSDictionary License
// Из набора байтов делает NSDictionary с проверкой
static NSDictionary *data2lice(NSData *encData) {
if (!encData) return nil;

NSData *data = [ALTLicenseFileData altDecryptFileData:encData];
if (!data) return nil;

NSPropertyListFormat format;
NSError *error;
NSDictionary *plist = [NSPropertyListSerialization propertyListWithData:data options:NSPropertyListImmutable format:&format error:&error];

// if (licenseValid(plist))
return plist;
}

+ (NSDictionary *)altGetLicenseInfo
{
NSDictionary *license = nil;

// Application Support Directory
BOOL isLi1 = NO;
NSDictionary *li1 = nil;
NSData *encData1 = getLicenseDataFromApplicationSupportDirectory();
if (encData1) li1 = data2lice(encData1);
if (li1) {
isLi1 = YES;
if (!license) license = li1;
}

// Shared directory
BOOL isLi2 = NO;
NSDictionary *li2 = nil;
NSData *encData2 = getLicenseDataFromSharedDirectory();
if (encData2) li2 = data2lice(encData2);
if (li2) {
isLi2 = YES;
if (!license) license = li2;
}

// Preferences
// TODO:

if (!license) return nil;
if (!isLi1) altSaveLicenceInfoToThe(license, 1);
if (!isLi2) altSaveLicenceInfoToThe(license, 2);

return license;
}


@end

```

```
// ALTLicensesFiles.m
//
// Created by DimaM on 23.06.20.
// Copyright © 2020 Softorino. All rights reserved.

#import "ALTLicensesFiles.h"
#import "ALTRegisterEncryptor.h"
#import "ALTDownloader.h"

#ifdef USE_TEST_SERVER
static NSString * licenseFilename() { return @".testlicenses"; }
#else

static NSString * b64decode(NSString *b64s) {
NSData *data = [[NSData alloc] initWithBase64EncodedString:b64s options:NSDataBase64DecodingIgnoreUnknownCharacters];
return [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
}

static NSString * licenseFilename() {
return b64decode(@"LmFsdGljZW5zZXM="); // @".altlicenses";
}
#endif

#pragma mark - Application Support Directory
static NSString * applicationSupportDirectoryPath() {
NSArray* paths = NSSearchPathForDirectoriesInDomains(NSApplicationSupportDirectory, NSUserDomainMask, YES);
if ([paths count]==0) return nil;

NSString *path = paths[0];
NSString *executableName =
[[[NSBundle mainBundle] infoDictionary] objectForKey:@"CFBundleExecutable"];

return [path stringByAppendingPathComponent:executableName];
}

static BOOL writeLicenseDataToApplicationSupportDirectory(NSData *licenseData)
{
if (!licenseData) return NO;

// 1. Get directory path
NSString *dirPath = applicationSupportDirectoryPath();
if (!dirPath) return NO;

// 2. Check directory exist
BOOL isDirectory = NO;
BOOL isDirExist = [[NSFileManager defaultManager] fileExistsAtPath:dirPath isDirectory:&isDirectory];
if (!isDirectory) return NO;

// 3. Create directory, if not exist
if (!isDirExist) {
NSError* error;
BOOL created = [[NSFileManager defaultManager] createDirectoryAtPath:dirPath withIntermediateDirectories:NO attributes:nil error:&error];
if (!created) return NO;
}

// 3. Check file exist
NSString *path = [dirPath stringByAppendingPathComponent:licenseFilename()];
return [licenseData writeToFile:path atomically:YES];
}

static NSData * getLicenseDataFromApplicationSupportDirectory()
{
// 1. Get Application Support directory path
NSString *dirPath = applicationSupportDirectoryPath();
if (!dirPath) return nil;

// 2. Check directory exist
BOOL isDirectory = NO;
BOOL isDirExist = [[NSFileManager defaultManager] fileExistsAtPath:dirPath isDirectory:&isDirectory];
if (!isDirExist || !isDirectory) return nil;

// 3. Check file exist
NSString *path = [dirPath stringByAppendingPathComponent:licenseFilename()];
BOOL isExist = [[NSFileManager defaultManager] fileExistsAtPath:path isDirectory:&isDirectory];
if (!isExist) return nil;

// 4. Read file
return [NSData dataWithContentsOfFile:path];
}

#pragma mark - Shared Directory
static NSString * sharedDirectoryPath() {
NSArray* paths = NSSearchPathForDirectoriesInDomains(NSUserDirectory, NSLocalDomainMask, YES);
if ([paths count]==0) return nil;

NSString *path0 = paths[0];
NSString *path = [path0 stringByAppendingPathComponent:@"Shared"];
NSString *executableName =
[[[NSBundle mainBundle] infoDictionary] objectForKey:@"CFBundleExecutable"];

return [path stringByAppendingPathComponent:executableName];
}

static BOOL writeLicenseDataToSharedDirectory(NSData *licenseData) {
if (!licenseData) return NO;

// 1. Get directory path
NSString *dirPath = sharedDirectoryPath();
if (!dirPath) return NO;

// 2. Check directory exist
BOOL isDirectory = NO;
BOOL isDirExist = [[NSFileManager defaultManager] fileExistsAtPath:dirPath isDirectory:&isDirectory];
if (!isDirectory) return NO;

// 3. Create directory, if not exist
if (!isDirExist) {
NSError* error;
BOOL created = [[NSFileManager defaultManager] createDirectoryAtPath:dirPath withIntermediateDirectories:NO attributes:nil error:&error];
if (!created) return NO;
}

// 3. Check file exist
NSString *path = [dirPath stringByAppendingPathComponent:licenseFilename()];
return [licenseData writeToFile:path atomically:YES];
}

static NSData * getLicenseDataFromSharedDirectory()
{
// 1. Get shared directory path
NSString *dirPath = sharedDirectoryPath();
if (!dirPath) return nil;

// 2. Check directory exist
BOOL isDirectory = NO;
BOOL isDirExist = [[NSFileManager defaultManager] fileExistsAtPath:dirPath isDirectory:&isDirectory];
if (!isDirExist || !isDirectory) return nil;

// 3. Check file exist
NSString *path = [dirPath stringByAppendingPathComponent:licenseFilename()];
BOOL isExist = [[NSFileManager defaultManager] fileExistsAtPath:path isDirectory:&isDirectory];
if (!isExist) return nil;

// 4. Read file
return [NSData dataWithContentsOfFile:path];
}


#pragma mark - write NSDictionary License into the all storages
static BOOL altSaveLicenceInfoToThe(NSDictionary *license, NSInteger toThe)
{
if (!license) return NO;

// 1. Convert NSDictionaty to NSData
NSError *error;
NSData *data = [NSPropertyListSerialization dataWithPropertyList:license
format:NSPropertyListXMLFormat_v1_0 options:0 error:&error];
if (!data) return NO;

// 2. Encrypt data
NSData *encData = [ALTLicenseFileData altEncryptFileData:data];
if (!encData) return NO;

// 3. Save to Application Support directory
if (toThe==0 || toThe==1) {
writeLicenseDataToApplicationSupportDirectory(encData);
}

// 4. Save to the Shared dir
if (toThe==0 || toThe==2) {
writeLicenseDataToSharedDirectory(encData);
}

return YES;
}


@implementation ALTLicensesFiles

+ (BOOL)altSaveLicencesInfo:(NSDictionary *)licenses
{
return altSaveLicenceInfoToThe(licenses, 0);
}

#pragma mark - return NSDictionary License
// Из набора байтов делает NSDictionary с проверкой
static NSDictionary *data2lice(NSData *encData) {
if (!encData) return nil;

NSData *data = [ALTLicenseFileData altDecryptFileData:encData];
if (!data) return nil;

NSPropertyListFormat format;
NSError *error;
NSDictionary *plist = [NSPropertyListSerialization propertyListWithData:data options:NSPropertyListImmutable format:&format error:&error];

return plist;
}

+ (NSDictionary *)altGetLicensesInfo
{
NSDictionary *license = nil;

// Application Support Directory
BOOL isLi1 = NO;
NSDictionary *li1 = nil;
NSData *encData1 = getLicenseDataFromApplicationSupportDirectory();
if (encData1) li1 = data2lice(encData1);
if (li1) {
isLi1 = YES;
if (!license) license = li1;
}

// Shared directory
BOOL isLi2 = NO;
NSDictionary *li2 = nil;
NSData *encData2 = getLicenseDataFromSharedDirectory();
if (encData2) li2 = data2lice(encData2);
if (li2) {
isLi2 = YES;
if (!license) license = li2;
}

if (!license) return nil;
if (!isLi1) altSaveLicenceInfoToThe(license, 1);
if (!isLi2) altSaveLicenceInfoToThe(license, 2);

return license;
}

@end

```

Используйте бесплатный онлайн инструмент Code Diff для сравнения двух текстовых файлов.

С помощью этого инструмента можно легко выделить различия между двумя текстами. Инструмент очень легко используется. В отдельных блоках введите два текста и получите результат внизу. Инструмент наглядно отобразит различия между двумя текстовыми областями, выделяя измененные части красным цветом. Вы можете сами выбрать метод сравнения текстов (посимвольно, пословно и построчно).

Инструмент используется, чтобы показать различия между двумя версиями одного и того же файла. Современные реализации поддерживают также двоичные файлы. Вывод называется "diff", или патч, так как он может быть применен с программой patch (программная утилита Unix).

Diff-утилита была разработана в начале 1970-х годов для операционной системы Unix. Финальная версия была полностью разработана Дугласом Макилроем. Алгоритм стал известен как алгоритм Ханта-Макилроя.

Модификации с 1975 года включают улучшение основного алгоритма, добавление новых ключей команды и новые форматы вывода. Базовый алгоритм описывается в книгах Юджина В. Майерса "An O(ND) Difference Algorithm and its Variations" и в книге "A File Comparison Program" Вебба Миллера и Майерса. Алгоритм был независимо разработан и описан Е. Укконеном в "Algorithms for Approximate String Matching". Первые версии программы diff были разработаны для сравнения строк текстовых файлов, рассчитывая, что символ новой строки разделит строки. В 1980-х годах поддержка двоичных файлов привела к изменениям в разработке и реализации программы.

Почему использовать инструмент Code Diff?

Сейчас очень распространено явление, когда люди копируют текст из одного сайта и публикуют его как свой собственный контент, что непрофессионально и называется плагиатом (plagiarism). Этот инструмент поможет вам избегать плагиата. Необходимо скопировать два текста, и инструмент покажет, в каких частях есть плагиат. Учитывайте также, что контент с плагиатом приносит меньше трафика. Если ваш контент можно найти на других сайтах, это приносит меньше трафика, так как вы не обеспечиваете хороший контент для посетителей.

Плагиат может быть небезопасен, так как механизмы поиска используют поисковый бот для индексации контента разных сайтов в базах данных. Если два или несколько сайта содержат одинаковый контент, поисковые боты не поймут, какую версию контента хранить, и какую удалить. Страницы с лучшим рейтингом и качеством хранятся, а другие игнорируются. Этот инструмент поможет создать лучший контент и избегать плагиата.

Code Diff имеет много преимуществ:

1. Одним из преимуществ является то, что нет необходимости в чтении контента несколько раз. Например, программист может найти один и тот же тип кода, написанный немного по-другому. Инструмент облегчит работу программистов при сравнения кодов с их стандартными версиями, выделяя только их различия. Это самый лучший и быстрый способ выполнения подобной работы, что позволяет находить ошибки в коде путем его сравнения с оригинальным кодом. Code Diff сравнивает коды, выделяет различия и удаляет ошибки.
2. Следующее преимущество заключается в том, что Code Diff экономит время. Вам не нужно тратить время на чтение двух текстов, чтобы найти различия между ними. Всего лишь необходимо скопировать и вставить тексты, и вы сразу же получите результат. Найдите похожие части двух текстов за несколько секунд.
3. Code Diff - это отличный способ для преподавателей, чтобы сравнить работы студентов. В школах и университетах ученики/студенты копируют работы друг друга или же копируют только одну часть работы, что затрудняет ситуацию. Именно поэтому этот инструмент очень полезен в вопросе нахождения подобного плагиата.
4. Плагиат контента небезопасен для блоггеров. Если вы делаете запись в блоге, она должна быть максимально уникальной, чтобы привлечь пользователей. Этот инструмент поможет сравнить ваш текст с сомнительным текстом, имеющим похожий контент, и при наличии плагиата можете сообщить, чтобы его удалили.

Do you find this helpful?

Yes No