# Кодировки текста

## История появления кодировок

**кодировка** - это таблица сопоставлений символов, которые мы можем видеть на экране, определенным числовым кодам.

Т.е. каждый символ, который мы вводим с клавиатуры, либо видим на экране монитора, закодирован определенной последовательностью битов (нулей и единиц). 8 бит, как вы, наверное, знаете, равны 1 байту информации.

**Внешний вид самих символов определяется файлами шрифтов**, которые установлены на вашем компьютере. Поэтому процесс вывода на экран текста можно описать как постоянное сопоставление последовательностей нулей и единиц каким-то конкретным символам, входящим в состав шрифта.

Стандартизирована только половина таблицы, т.н. ASCII-код - первые 128 символов, которые включают в себя буквы латинского алфавита. И с ними никогда не бывает проблем. Вторая же половина таблицы (а всего в ней 256 символов - по количеству состояний, который может принять один байт) отдана под национальные символы, и в каждой стране эта часть различна. Но только в России умудрились придумать целых 5 различных кодировок. Термин "различные" обозначает то, что одному и тому же символу соответствует разный цифровой код. Т.е. если мы неправильно определим кодировку текста, то нашему вниманию предстанет абсолютно нечитаемый текст.

Кодировки появились исторически. Первая широко используемая российская кодировка называлась KOI-8. Ее придумали, когда адаптировали к русскому языку систему UNIX. Это было еще в семидесятых - до появления персоналок. И до сих пор в UNIX это считается основной кодировкой.

Потом появились первые персональные компьютеры, и началось победное шествие DOS. Вместо того чтобы воспользоваться уже придуманной кодировкой, Microsoft решила сделать свою, ни с чем не совместимую. Так появилась DOS-кодировка (или 866 кодовая страница). В ней, кстати, были введены спецсимволы для рисования рамок, что широко использовалось в программах написанных под DOS. Например, в том же Norton Commander-е.

Параллельно с IBM-совместимыми развивались и Macintosh-компьютеры. Несмотря на то, что их доля в России очень мала, тем не менее, потребность в русификации существовала и, разумеется, была придумана еще одна кодировка - MAC.

Время шло, и 1990 году Microsoft явила на свет первую успешную версию Windows 3.0-3.11. А вместе с ней и поддержку национальных языков. И снова был проделан такой же фокус, как и с DOS. По непонятным причинам они не поддержали ни одну, из уже существовавших ранее (как это сделала OS/2, принявшая за стандарт DOS-кодировку), а предложили новую Win-кодировку (или кодовая страница 1251). Де-факто, она стала самой распространенной в России.

И, наконец, пятый вариант кодировки связан уже не с конкретной фирмой, а с попытками стандартизации кодировок на уровне всей планеты. Занималась этим ISO - международная организация по стандартам. И, догадайтесь, что они сделали с русским языком? Вместо того, чтобы принять за "стандартную русскую" какую-нибудь из вышеописанных, они придумали еще одну (!) и назвали ее длинным неудобоваримым сочетанием ISO-8859-5. Разумеется, она тоже оказалась ни с чем не совместимой. И в настоящий момент эта кодировка практически нигде не применяется. Кажется, ее используют только в базе данных Oracle. По крайней мере, я ни разу не видел текст в этой кодировке. Тем не менее, ее поддержка присутствует во всех браузерах.

Сейчас идет работа над созданием новой универсальной кодировки (UNICODE), в которой предполагается в одну кодовую таблицу запихнуть все языки мира. Тогда точно проблем не будет. Для этого на каждый символ отвели 2 байта. Таким образом, максимальное количество знаков в таблице расширилось до 65535. Но до момента, когда все перейдут на UNICODE, остается еще слишком много времени.

## Кодировки для кириллического текста

*ASCII* (англ. *American Standard Code for Information Interchange*) — американская стандартная кодировочная таблица для печатных символов и некоторых специальных кодов. В американском варианте английского языка произносится [э́ски], тогда как в Великобритании чаще произносится [а́ски]; по-русски произносится также [а́ски] или [аски́].

Кодировка ASCII была разработана в 1963 году Американской Ассоциацией Стандартов (которая позже стала Американским Национальным Институтом Стандартов — ANSI), впоследствии несколько раз обновлялась — в 1967 и 1986 годах. ASCII — 7-битная кодировка, включающая в себя 128 символов: 33 непечатных управляющих символа (влияющих на обработку текста и пробелов) и 95 печатных символов, включая цифры, буквы латинского алфавита в строчном и прописном вариантах и ряд пунктуационных символов.


## Расширенные версии Аски — кодировки CP866 и KOI8-R

Итак, мы с вами начали говорить про ASCII, которая являлась как бы отправной точкой для развития всех современных кодировок (Windows 1251, юникод, UTF 8).

Изначально в нее было заложено только 128 знаков латинского алфавита, арабских цифр и еще чего-то там, но в расширенной версии появилась возможность использовать все 256 значений, которые можно закодировать в одном байте информации. Т.е. появилась возможность добавить в Аски символы букв своего языка.

**изначально появилась CP866, в которой была возможность использовать символы русского алфавита**

Т.е. ее верхняя часть полностью совпадала с базовой версией Аски (128 символов латиницы, цифр и еще всякой лабуды), которая представлена на приведенном чуть выше скриншоте, а вот уже нижняя часть таблицы с кодировкой CP866 имела указанный на скриншоте чуть ниже вид и позволяла закодировать еще 128 знаков (русские буквы и всякая там псевдографика):

Видите, в правом столбце цифры начинаются с 8, т.к. числа с 0 до 7 относятся к базовой части ASCII (см. первый скриншот). Т.о. русская буква «М» в CP866 будет иметь код 9C (она находится на пересечении соответствующих строки с 9 и столбца с цифрой C в шестнадцатеричной системе счисления), который можно записать в одном байте информации, и при наличии подходящего шрифта с русскими символами эта буква без проблем отобразится в тексте.

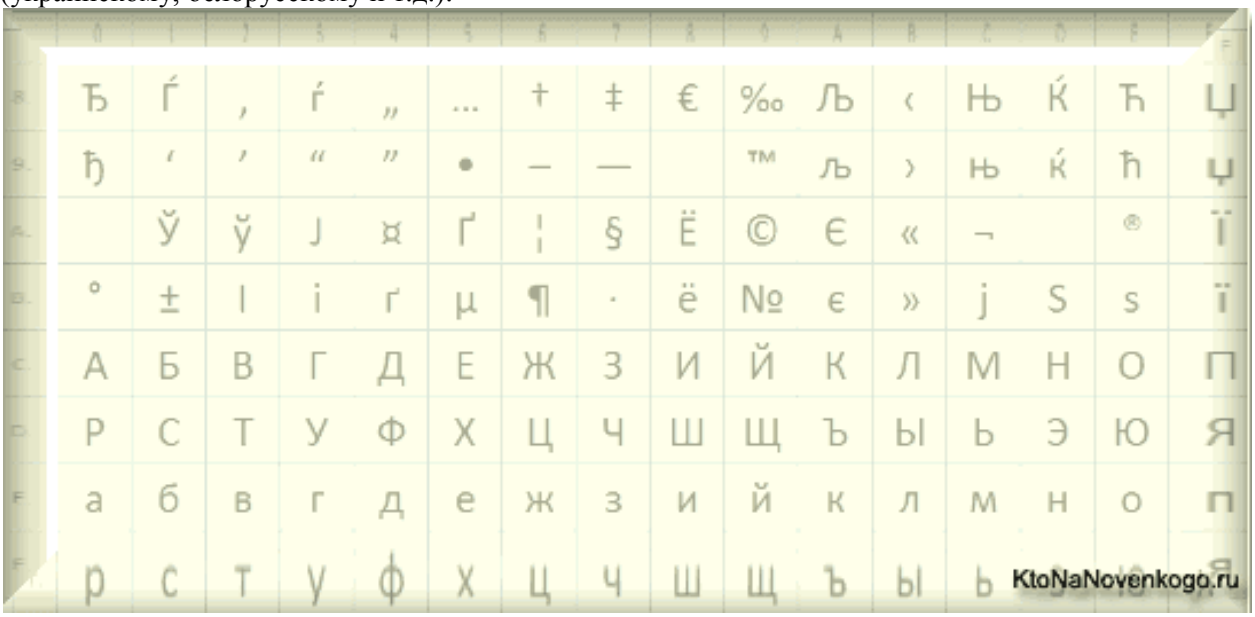
CP866 распространяла компания IBM, но кроме этого для символов русского языка были разработаны еще ряд кодировок, например, к этому же типу (расширенных ASCII) можно отнести **KOI8-R**:


Принцип ее работы остался тот же самый, что и у описанной чуть ранее CP866 — каждый символ текста кодируется одним единственным байтом. На скриншоте показана вторая половина таблицы KOI8-R, т.к. первая половина полностью соответствует базовой Аски, которая показана на первом скриншоте в этой статье.

Среди особенностей кодировки KOI8-R можно отметить то, что русские буквы в ее таблице идут не в алфавитном порядке, как это, например, сделали в CP866.

## Windows 1251 — современная версия ASCII

Она выгодно отличалась от используемых ранее CP866 и KOI8-R тем, что место символов псевдографики в ней заняли недостающие символы русской типографики (окромя знака ударения), а также символы, используемые в близких к русскому славянских языках (украинскому, белорусскому и т.д.):



## Юникод (Unicode) — универсальные кодировки UTF 8, 16 и 32

Тысячи знаков языковой группы юго-восточной Азии никак невозможно было описать в одном байте информации, который выделялся для кодирования символов в расширенных версиях ASCII. В результате был создан консорциум под названием Юникод (Unicode — Unicode Consortium) при сотрудничестве многих лидеров IT индустрии (те, кто производит софт, кто кодирует железо, кто создает шрифты), которые были заинтересованы в появлении универсальной кодировки текста.

Первой вариацией, вышедшей под эгидой консорциума Юникод, была **UTF 32**. Цифра в названии кодировки означает количество бит, которое используется для кодирования одного символа. 32 бита составляют 4 байта информации, которые понадобятся для кодирования одного единственного знака в новой универсальной кодировке UTF.

В результате развития Юникода появилась **UTF-16**, которая получилась настолько удачной, что была принята по умолчанию как базовое пространство для всех символов, которые у нас используются. Она использует два байта для кодирования одного знака.

Для удовлетворения всех и вся в консорциуме Unicode было решено придумать **кодировку переменной длины**. Ее назвали **UTF-8.** Несмотря на восьмерку в названии, она действительно имеет переменную длину, т.е. каждый символ текста может быть закодирован в последовательность длиной от одного до шести байт.

На практике же в UTF-8 используется только диапазон от одного до четырех байт, потому что за четырьмя байтами кода ничего уже даже теоретически не возможно представить. Все латинские знаки в ней кодируются в один байт, так же как и в старой доброй ASCII.

Многобайтовые кодировки

**Что значит многобайтовая кодировка**? Это означает, что на один символ может выделяться больше чем один байт. Ведь действительно, все символы представлены байтами, чтобы закодировать символ потребуется некоторое их количество и одного может и не хватить. Особенно это касается необычных символов и букв каких-либо языков. Поэтому многобайтовые кодировки нужны, их поддержка конечно же есть в PHP.

Есть функции, которые способны самостоятельно определять кодировку текста. Также в них можно самому указать нужную кодировку при необходимости. Существует часть функций, которые начинаются **с префикса mb_**. Они специально предназначены для работы с текстом, **mb значит многобайтовость.**

Посмотрим, какие есть основные **функции mb в PHP**, ниже приводятся только самые используемые:

- **mb_convert_case** - производит смену регистра символов в строке,
- **mb_convert_encoding** - преобразует кодировку символов,
- **mb_detect_encoding** - определение кодировки символов,
- **mb_internal_encoding** – установка или получение внутренней кодировки скрипта,
- **mb_ord** - получает кодовую точку символа,
- **mb_split** - разделение строк в многобайтных кодировках, используя регулярное выражение,
- **mb_strcut** - получение части строки,
- **mb_stripos** - регистронезависимый поиск позиции первого вхождения одной строки в другую,
- **mb_strlen** - получает длину строки,
- **mb_strpos** - поиск позиции первого вхождения одной строки в другую,
- **mb_strripos** - поиск последнего вхождения одной строки в другую, нечувствительный к регистру,
- **mb_strrpos** - поиск позиции последнего вхождения одной строки в другую,
- **mb_strstr** - находит первое вхождение подстроки в строке,
- **mb_strtolower** - приведение строки к нижнему регистру,
- **mb_strtoupper** - приведение строки к верхнему регистру,
- **mb_substr** - возвращает часть строки.

Таким образом, для работы с текстом лучше всего использовать **многобайтовые кодировки**. Они позволяют правильно осуществлять операции с символами.

## Передача данных через почтовый протокол: base64

**Base64** — стандарт кодирования двоичных данных при помощи только 64 символов ASCII. Алфавит кодирования содержит текстово-цифровые латинские символы A-Z, a-z и 0-9 (62 знака) и 2 дополнительных символа, зависящих от системы реализации. Каждые 3 исходных байта кодируются 4 символами (увеличение на ⅓).

Эта система широко используется в электронной почте для представления бинарных файлов в тексте письма (транспортное кодирование).

Применение в веб-приложениях

Кодирование Base64 может быть полезно, если в окружении HTTP используется информация, длину которой можно точно определить. Также многим приложениям необходимо кодировать двоичные данные для удобства включения в URL, скрытые поля форм, и здесь Base64 удобно не только для компактного представления, но и относительной нечитаемостью для попытки выяснения случайным человеком-наблюдателем природы данных.

Использование URL-кодировщика над стандартом Base64, несмотря на это, неудобно, так как он преобразует символы / и + в специальные шестнадцатеричные последовательности. Если позднее эта строка используется вместе с базой данных или через гетерогенные системы, они прекращают работу на символе %, сгенерированном URL-кодировщиком (потому что символ % также используется в ANSI SQL как шаблон).

По причине этого существует изменённый Base64 для URL, где не используется заполнение символом = и символы + и / соответственно заменяются на * и -, так что использование кодеров/декодеров URL перестаёт быть необходимым и не имеет никакого воздействия на длину закодированного значения, оставляя ту же самую закодированную форму, неповреждённую для использования в реляционных базах данных, веб-формах и идентификаторах объекта вообще. Стандартом Base64-кодирования URL адресов признается вариант, когда символы + и / заменяются, соответственно, на - и _

Другой вариант называется **изменённый Base64 для регулярных выражений**, использует ! и - вместо * и -, для того, чтобы заменить стандартный Base64 +/, потому что оба + и * могут быть зарезервированы для регулярных выражений (отметим, что [], используемый выше в IRCu варианте, может не работать в этом контексте).

Имеются другие варианты, которые используют _ и - или . и _, если строка Base64 должна быть использована вместе с идентификаторами для программ, или . и - для использования в токенах имён XML (Nmtoken), или _ и : в более ограниченных идентификаторах XML (Name). В некоторых случаях для URL применяется Base58, который не использует символы + и /.