## NAME

charsets − programmer's view of character sets and internationalization

## DESCRIPTION

Linux is an international operating system. Various of its utilities and device drivers (including the console driver) support multilingual character sets including Latin-alphabet letters with diacritical marks, accents, ligatures, and entire non-Latin alphabets including Greek, Cyrillic, Arabic, and Hebrew.

This manual page presents a programmer's-eye view of different character-set standards and how they fit together on Linux. Standards discussed include ASCII, ISO 8859, KOI8-R, Unicode, ISO 2022 and ISO 4873. The primary emphasis is on character sets actually used as locale character sets, not the myriad others that can be found in data from other systems.

A complete list of charsets used in an officially supported locale in glibc 2.2.3 is: ISO-8859-{1,2,3,5,6,7,8,9,13,15}, CP1251, UTF-8, EUC-{KR,JP,TW}, KOI8-{R,U}, GB2312, GB18030, GBK, BIG5, BIG5-HKSCS and TIS-620 (in no particular order.) (Romanian may be switching to ISO-8859-16.)

The recommended encoding in all settings and locales is UTF-8.

### ASCII

ASCII (American Standard Code For Information Interchange) is the original 7-bit character set, originally designed for American English. It is currently described by the ECMA-6 standard.

Various ASCII variants replacing the dollar sign with other currency symbols and replacing punctuation with non-English alphabetic characters to cover German, French, Spanish and others in 7 bits exist. All are deprecated; glibc doesn't support locales whose character sets aren't true supersets of ASCII. (These sets are also known as ISO-646, a close relative of ASCII that permitted replacing these characters.)

As Linux was written for hardware designed in the US, it natively supports ASCII.

### ISO 8859

ISO 8859 is a series of 15 8-bit character sets all of which have US ASCII in their low (7-bit) half, invisible control characters in positions 128 to 159, and 96 fixed-width graphics in positions 160-255.

Of these, the most important is ISO 8859-1 (Latin-1). It is natively supported in the Linux console driver, fairly well supported in X11R6, and is the base character set of HTML.

Console support for the other 8859 character sets is available under Linux through user-mode utilities (such as **setfont**(8)) that modify keyboard bindings and the EGA graphics table and employ the "user mapping" font table in the console driver.

Here are brief descriptions of each set:

8859-1 (Latin-1)
: Latin-1 covers most Western European languages such as Albanian, Catalan, Danish, Dutch, English, Faroese, Finnish, French, German, Galician, Irish, Icelandic, Italian, Norwegian, Portuguese, Spanish, and Swedish. The lack of the ligatures Dutch ij, French oe and old-style ,,German'' quotation marks is considered tolerable.

8859-2 (Latin-2)
: Latin-2 supports most Latin-written Slavic and Central European languages: Croatian, Czech, German, Hungarian, Polish, Rumanian, Slovak, and Slovene.

8859-3 (Latin-3)
: Latin-3 is popular with authors of Esperanto, Galician, and Maltese. (Turkish is now written with 8859-9 instead.)

8859-4 (Latin-4)
: Latin-4 introduced letters for Estonian, Latvian, and Lithuanian. It is essentially obsolete; see 8859-10 (Latin-6) and 8859-13 (Latin-7).

8859-5 Cyrillic letters supporting Bulgarian, Byelorussian, Macedonian, Russian, Serbian and Ukrainian. Ukrainians read the letter "ghe" with downstroke as "heh" and would need a ghe with upstroke to

write a correct ghe. See the discussion of KOI8-R below.

8859-6 Supports Arabic. The 8859-6 glyph table is a fixed font of separate letter forms, but a proper display engine should combine these using the proper initial, medial, and final forms.

8859-7 Supports Modern Greek.

8859-8 Supports modern Hebrew without niqud (punctuation signs). Niqud and full-fledged Biblical Hebrew are outside the scope of this character set; under Linux, UTF-8 is the preferred encoding for these.

8859-9 (Latin-5)
This is a variant of Latin-1 that replaces Icelandic letters with Turkish ones.

8859-10 (Latin-6)
Latin 6 adds the last Inuit (Greenlandic) and Sami (Lappish) letters that were missing in Latin 4 to cover the entire Nordic area. RFC 1345 listed a preliminary and different "latin6". Skolt Sami still needs a few more accents than these.

8859-11
This only exists as a rejected draft standard. The draft standard was identical to TIS-620, which is used under Linux for Thai.

8859-12
This set does not exist. While Vietnamese has been suggested for this space, it does not fit within the 96 (non-combining) characters ISO 8859 offers. UTF-8 is the preferred character set for Vietnamese use under Linux.

8859-13 (Latin-7)
Supports the Baltic Rim languages; in particular, it includes Latvian characters not found in Latin-4.

8859-14 (Latin-8)
This is the Celtic character set, covering Gaelic and Welsh. This charset also contains the dotted characters needed for Old Irish.

8859-15 (Latin-9)
This adds the Euro sign and French and Finnish letters that were missing in Latin-1.

8859-16 (Latin-10)
This set covers many of the languages covered by 8859-2, and supports Romanian more completely then that set does.

**KOI8-R**

KOI8-R is a non-ISO character set popular in Russia. The lower half is US ASCII; the upper is a Cyrillic character set somewhat better designed than ISO 8859-5. KOI8-U is a common character set, based off KOI8-R, that has better support for Ukrainian. Neither of these sets are ISO-2022 compatible, unlike the ISO-8859 series.

Console support for KOI8-R is available under Linux through user-mode utilities that modify keyboard bindings and the EGA graphics table, and employ the "user mapping" font table in the console driver.

**JIS X 0208**

JIS X 0208 is a Japanese national standard character set. Though there are some more Japanese national standard character sets (like JIS X 0201, JIS X 0212, and JIS X 0213), this is the most important one. Characters are mapped into a 94x94 two-byte matrix, whose each byte is in the range 0x21-0x7e. Note that JIS X 0208 is a character set, not an encoding. This means that JIS X 0208 itself is not used for expressing text data. JIS X 0208 is used as a component to construct encodings such as EUC-JP, Shift_JIS, and ISO-2022-JP. EUC-JP is the most important encoding for Linux and includes US ASCII and JIS X 0208. In EUC-JP, JIS X 0208 characters are expressed in two bytes, each of which is the JIS X 0208 code plus 0x80.

**KS X 1001**

KS X 1001 is a Korean national standard character set. Just as JIS X 0208, characters are mapped into a 94x94 two-byte matrix. KS X 1001 is used like JIS X 0208, as a component to construct encodings such as EUC-KR, Johab, and ISO-2022-KR. EUC-KR is the most important encoding for Linux and includes US ASCII and KS X 1001. KS C 5601 is an older name for KS X 1001.

**GB 2312**

GB 2312 is a mainland Chinese national standard character set used to express simplified Chinese. Just like JIS X 0208, characters are mapped into a 94x94 two-byte matrix used to construct EUC-CN. EUC-CN is the most important encoding for Linux and includes US ASCII and GB 2312. Note that EUC-CN is often called as GB, GB 2312, or CN-GB.

**Big5**

Big5 is a popular character set in Taiwan to express traditional Chinese. (Big5 is both a character set and an encoding.) It is a superset of US ASCII. Non-ASCII characters are expressed in two bytes. Bytes 0xa1-0xfe are used as leading bytes for two-byte characters. Big5 and its extension is widely used in Taiwan and Hong Kong. It is not ISO 2022-compliant.

**TIS 620**

TIS 620 is a Thai national standard character set and a superset of US ASCII. Like ISO 8859 series, Thai characters are mapped into 0xa1-0xfe. TIS 620 is the only commonly used character set under Linux besides UTF-8 to have combining characters.

**UNICODE**

Unicode (ISO 10646) is a standard which aims to unambiguously represent every character in every human language. Unicode's structure permits 20.1 bits to encode every character. Since most computers don't include 20.1-bit integers, Unicode is usually encoded as 32-bit integers internally and either a series of 16-bit integers (UTF-16) (needing two 16-bit integers only when encoding certain rare characters) or a series of 8-bit bytes (UTF-8). Information on Unicode is available at <http://www.unicode.org>.

Linux represents Unicode using the 8-bit Unicode Transformation Format (UTF-8). UTF-8 is a variable length encoding of Unicode. It uses 1 byte to code 7 bits, 2 bytes for 11 bits, 3 bytes for 16 bits, 4 bytes for 21 bits, 5 bytes for 26 bits, 6 bytes for 31 bits.

Let 0,1,x stand for a zero, one, or arbitrary bit. A byte 0xxxxxxx stands for the Unicode 00000000 0xxxxxxx which codes the same symbol as the ASCII 0xxxxxxx. Thus, ASCII goes unchanged into UTF-8, and people using only ASCII do not notice any change: not in code, and not in file size.

A byte 110xxxxx is the start of a 2-byte code, and 110xxxxx 10yyyyyy is assembled into 00000xxx xxyyyyyy. A byte 1110xxxx is the start of a 3-byte code, and 1110xxxx 10yyyyyy 10zzzzzz is assembled into xxxxyyyy yyzzzzzz. (When UTF-8 is used to code the 31-bit ISO 10646 then this progression continues up to 6-byte codes.)

For most people who use ISO-8859 character sets, this means that the characters outside of ASCII are now coded with two bytes. This tends to expand ordinary text files by only one or two percent. For Russian or Greek users, this expands ordinary text files by 100%, since text in those languages is mostly outside of ASCII. For Japanese users this means that the 16-bit codes now in common use will take three bytes. While there are algorithmic conversions from some character sets (esp. ISO-8859-1) to Unicode, general conversion requires carrying around conversion tables, which can be quite large for 16-bit codes.

Note that UTF-8 is self-synchronizing: 10xxxxxx is a tail, any other byte is the head of a code. Note that the only way ASCII bytes occur in a UTF-8 stream, is as themselves. In particular, there are no embedded NULs ('\0') or '/'s that form part of some larger code.

Since ASCII, and, in particular, NUL and '/', are unchanged, the kernel does not notice that UTF-8 is being used. It does not care at all what the bytes it is handling stand for.

Rendering of Unicode data streams is typically handled through "subfont" tables which map a subset of Unicode to glyphs. Internally the kernel uses Unicode to describe the subfont loaded in video RAM. This means that in UTF-8 mode one can use a character set with 512 different symbols. This is not enough for Japanese, Chinese and Korean, but it is enough for most other purposes.

At the current time, the console driver does not handle combining characters. So Thai, Sioux and any other script needing combining characters can't be handled on the console.

**ISO 2022 and ISO 4873**

The ISO 2022 and 4873 standards describe a font-control model based on VT100 practice. This model is (partially) supported by the Linux kernel and by **xterm**(1). It is popular in Japan and Korea.

There are 4 graphic character sets, called G0, G1, G2 and G3, and one of them is the current character set for codes with high bit zero (initially G0), and one of them is the current character set for codes with high bit one (initially G1). Each graphic character set has 94 or 96 characters, and is essentially a 7-bit character set. It uses codes either 040-0177 (041-0176) or 0240-0377 (0241-0376). G0 always has size 94 and uses codes 041-0176.

Switching between character sets is done using the shift functions **^N** (SO or LS1), **^O** (SI or LS0), ESC n (LS2), ESC o (LS3), ESC N (SS2), ESC O (SS3), ESC ~ (LS1R), ESC } (LS2R), ESC | (LS3R). The function LS*n* makes character set G*n* the current one for codes with high bit zero. The function LS*n*R makes character set G*n* the current one for codes with high bit one. The function SS*n* makes character set G*n* (*n*=2 or 3) the current one for the next character only (regardless of the value of its high order bit).

A 94-character set is designated as G*n* character set by an escape sequence ESC ( xx (for G0), ESC ) xx (for G1), ESC * xx (for G2), ESC + xx (for G3), where xx is a symbol or a pair of symbols found in the ISO 2375 International Register of Coded Character Sets. For example, ESC ( @ selects the ISO 646 character set as G0, ESC ( A selects the UK standard character set (with pound instead of number sign), ESC ( B selects ASCII (with dollar instead of currency sign), ESC ( M selects a character set for African languages, ESC ( ! A selects the Cuban character set, etc. etc.

A 96-character set is designated as G*n* character set by an escape sequence ESC − xx (for G1), ESC . xx (for G2) or ESC / xx (for G3). For example, ESC − G selects the Hebrew alphabet as G1.

A multibyte character set is designated as G*n* character set by an escape sequence ESC $ xx or ESC $ ( xx (for G0), ESC $ ) xx (for G1), ESC $ * xx (for G2), ESC $ + xx (for G3). For example, ESC $ ( C selects the Korean character set for G0. The Japanese character set selected by ESC $ B has a more recent version selected by ESC & @ ESC $ B.

ISO 4873 stipulates a narrower use of character sets, where G0 is fixed (always ASCII), so that G1, G2 and G3 can only be invoked for codes with the high order bit set. In particular, **^N** and **^O** are not used anymore, ESC ( xx can be used only with xx=B, and ESC ) xx, ESC * xx, ESC + xx are equivalent to ESC − xx, ESC . xx, ESC / xx, respectively.

## SEE ALSO

**console**(4), **console_codes**(4), **console_ioctl**(4), **ascii**(7), **iso_8859-1**(7), **unicode**(7), **utf-8**(7)

## COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.