STRINGSEXT(1) Version 2.3.4 | Stringsext Documentation

NAME

Search for multi-byte encoded strings in binary data.

SYNOPSIS

```
stringsext [options] [-e ENC...] [--] [FILE...]
stringsext [options] [-e ENC...] [--] [-]
```

DESCRIPTION

stringsext is a Unicode enhancement of the *GNU strings* tool with additional functionalities: **stringsext** recognizes Cyrillic, CJKV characters and other scripts in all supported multi-byte-encodings, while *GNU strings* fails in finding any of these scripts in UTF-16 and many other encodings.

stringsext is mainly useful for determining the Unicode content in binary data: It prints all graphic character sequences in *FILE* or *stdin* that are at least *MIN* bytes long.

Unlike *GNU strings* **stringsext** can be configured to search for valid characters not only in ASCII but also in many other input encodings, e.g.: *utf-8, utf-16be, utf-16le, big5, euc-jp, koi8-r* and many others. **--list-encodings** shows a list of valid encoding names based on the WHATWG Encoding Standard. When more than one encoding is specified, the scan is performed in different threads simultaneously.

When searching for UTF-16 encoded strings, 96% of all possible two byte sequences, interpreted as UTF-16 code unit, relate directly to Unicode code points. As a result, the probability of encountering valid Unicode characters in a random byte stream, interpreted as UTF-16, is also 96%. In order to reduce this big number of false positives, **stringsext** provides a parametrizable Unicode-block-filter. See **--encodings** option for more details.

stringsext reads its input data from (multiple) **FILE**s. With no **FILE** is given, or when **FILE** is “`-`”, it reads standard input *stdin*.

When invoked with "`stringsext -e ascii`", **stringsext** can be used as *GNU strings* replacement.

OPTIONS

-a *AF*, --ascii-filter=*AF*

Apply ASCII-Filter. After the string-findings had been decoded into UTF-8, the ASCII-filter is one of the 4 filters all string-findings have to pass before being printed. The ASCII-filter is applied to Unicode characters in "`U+0000..U+007F`"only. The filter parameter AF decides which of these codes will pass the filter. AF is some 128 bit integer, where each bit is mapped to one character in the above character-range, e.g. the character "`U+0020`" will pass the filter only, if bit no. 32 (=0x20) is set. If the filter is configured with bit no. 32 cleared, all characters "`U+0020`" will be rejected.

The integer AF is notated in hexadecimal with prefix "`0x...`". For the most common use-cases, predefined filters can be set: e.g. alias names like "`All-Ctrl`"or "`All-Ctrl+Wsp`" are shorthand terms for ASCII-filters "all codes, but no control-codes" or "all codes, including white-space, but no control-codes". See the output of "`--list-encodings`" for more details about filter-names.

-c, --no-metadata

Suppress all metadata in output. "`stringsext`"presents its string-findings in one or more output-lines. Each line shows some meta information before printing the finding itself. See the section "`Output Format`" for more information about metadata.

-d, --debug-options

Show how command-line-options are interpreted. When set, "`stringsext`"prints an exhaustive filter parameter synoptic. Can be used for debugging to check how the present command-line-arguments are interpreted or for documentation purpose. Does not run the scanner.

-e *ENC*, --encoding=*ENC*

Set (multiple) input search encodings.

ENC==[ENCNAME],[*MIN*],[*AF*],[*UBF*], [*GREP*]

ENCNAME

Search for strings encoded as ENCNAME. Encoding names *ENCNAME* are denoted following the WATHWG standard. "`--list-encodings`"prints a list of available encodings.

MIN, AF, UBF, GREP

Once the input is decoded to UTF-8, all characters have to pass 4 additional filters before being printed: MIN (see "`--chars-min`"), AF (see "`--ascii-filter`"), UBF (see "`--unicode-block-filter`") and GREP (see "`--grep-char`").

The values given here override - for this ENC only - the default values given by "`--chars-min`", "`--ascii-filter`", "`--unicode-block-filter`"and "`--grep-char`".

"`--list-encodings`" prints a list of predefined filter-names.

-g *ASCII_CODE*, --grep-char=*ASCII_CODE*

Print only findings having at least one character with ASCII_CODE. "`--grep-char`" is one of the 4 filters decoded output lines must pass before being printed. "`--grep-char`" checks for the presence of ASCII_CODE in the first output-line of a string-finding. The ASCII-code can be given as decimal or hexadecimal number. The latter starts with "`0x...`". Useful values are "`47`" (`/`) or "`92`" (`\`) for path search.

-h, --help

Print a synopsis of available options and default values.

-l, --list-encodings

List available encodings as WHATWG-Encoding-Standard-names, predefined ASCII-filter and Unicode-Block-Filter alias names.

-n *MIN*, --chars-min=*MIN*

Print only strings at least *MIN* characters long. The string length is measured in Unicode-characters (codepoints). **--help** shows the default value.

-p *FILE*, --output=*FILE*

Print to *FILE* instead of *stdout*.

-q *NUM*, --output-line-len=*NUM*

Set the printed output-line-length in UTF-8 characters (string-findings only, metadata excluded). The line-length is limited by some internal buffer size value (see "`OUTPUT_BUF_LEN`" in source code). A value "`NUM`" bigger than "`OUTPUT_BUF_LEN/2`"is set to "`OUTPUT_BUF_LEN/2`". The longer the line-length is, the fewer strings will be wrapped to the next line. The downside with long output lines is, that the scanner loses precision in locating the findings.

-r, --same-unicode-block

Require all characters in a finding to originate from the same Unicode block. This option helps to reduce false positives, especially when scanning for UTF-16. When set, "`stringsext`"prints only Unicode block homogenous strings. For example: "`-u All -n 10 -r`" finds a sequence of at least 10 Cyrillic characters in a row or finds at least 10 Greek characters in a row, whereas it ignores strings with randomly Cyrillic-Greek mixed characters. Technically, this option guarantees, that all multibyte characters of a finding - decoded into UTF-8 - start with the same leading byte. This might be the default behavoir, in some future version of **stringsext**.

-s ***NUM*****, --counter-offset=*****NUM***

Start offset NUM for the input-stream-byte-counter given as decimal or hexadecimal integer. This is useful when large input data is stored split in separate files and when these files are so big that they should be analysed in separate **stringsext** runs.

Note: in general, it is better to treat all input files in one run by listing them as command-line-parameter. Thus, **stringsext** concatenates the split input-files to one input-stream before analyzing it. This way it is able to even recognize split strings at the cutting edge between two input files.

-t ***RADIX*****, --radix=*****RADIX***

Print the position of the decoded string. The position indicated as input-stream bytes-offset. The single character argument specifies the RADIX of the offset: **o** for octal, **x** for hexadecimal, or **d** for decimal.

-u ***UBF*****, --unicode-block-filter=*****UBF***

Unicode-block-filter UBF applied after decoding to UTF-8.

The decoder first searches for validly encoded character sequences in the input stream. Then, the sequence of valid characters is decoded into a chunk of UTF-8 characters, which has to pass 4 filters before being printed: "`--chars-min`", "`--ascii-filter`", "`--unicode-bloc-filter`" and "`--grep-char`".

The Unicode-block-filter applies to all decoded UTF-8 characters "`> U+007f`"and can be parametrized with the `--unicode-block-filter`" option which is a 64-bit integer given in hexadecimal, prepended with "`0x...`".

Every bit "`0..=63`"maps to one leading-byte's code position in "`0xC0..0xFF`", e.g. if bit 0 is set -> all characters with leading byte "`0xC0`"pass the filter; if bit 1 is set -> all characters with leading byte "`0xC1`", pass the filter. Otherwise, the character is rejected. For example, to print only Syriac, set UFB to "`0x1000_0000`"(bit number 29 set) and AF to "0x0". This instructs the filter to let pass only UTF-8 characters, whose leading byte is "`0xDC`". Table 3 on page https://en.wikipedia.org/wiki/UTF-8 shows UTF-8-leading-bytes and their codes.

Alternatively, predefined alias names for the most common Unicode-blocks can be used: e.g. "`Latin`", "`Cyrillic`", "`Greek`"and many others. See the output of "`--list-encodings`" for more predefined filter names.

-V, --version

Print version info and exit.

EXIT STATUS

0

Success.

other values

Failure.

OUTPUT FORMAT

The way **stringsext** prints its output can be configured with the following options: "`--no-metadata`", "`--radix`"and "`--output-line-len`". The first "`--no-metadata`" controls if metadata is presented printed, "`--radix` determines if and how the byte-counter is shown and the latter" `--output-line-len`" at what byte position string-findings are wrapped to the next line.

stringsext's output syntax is best illustrated by example. Consider the following screen-shot:

```
stringsext -t x -q 30 -e utf8,10 -e ascii,50 test.txt test-small.txt (1)
                                                                     (2)
A   0    (a UTF-8)   Who Moved My Cheese?                            (3)
A <1e    (a UTF-8)   An A-Mazing Way To Deal With C                  (4)
A >1e+   (a UTF-8)   hange In                                        (5)
A <1e    (b ascii)   An A-Mazing Way To Deal With C                  (6)
A >1e+   (b ascii)   hange In                                        (7)
A  3c+   (a UTF-8)    Your Work                                      (8)
A >3c    (a UTF-8)   And In Your Life                                (9)
A  3c+   (b ascii)    Your Work                                      (10)
```

(3): The letter "`A`"in the first column indicates, that the input originates from the first input file "`test.txt`". "`B`" denotes the second input file, etc.

(3): "`0`"indicates, that the string-finding "`Who Moved My Cheese?`"was found at position "`0x0`".

(4): “`<1e`”means, that the string-finding “`An A-Mazing Way To Deal With C`”was found somewhere in “`0x1..=0x1e`”. In addition, the implemented algorithm guarantees that the string-finding is never more than 60 bytes (2* `-q 30`) away from the indicated position, here: “`0x1e`”.

(5): The string-finding “`hange In`”continues the previous string, hence “`+`”, and is situated “`>1e`”, meaning somewhere in the range “`0x1f..=3b`”. Here again, it is guaranteed, that the string-finding is always fewer than 60 bytes (2*`-q 30`) away from “`1e`”.

(3): “`a`”in “`(a UTF-8)`”indicates, that the string-finding “`Who Moved My Cheese?`” was found by the first scanner “`-e utf8,10`”.

(6): “`b`” refers to the second scanner, here “`-e ascii,50`”.

EXAMPLES

List available encodings and predefined filter names:

```
stringsext -l
```

Search for UTF-8 and UTF-16 Big-Endian encoded strings:

```
stringsext -t x -e utf-8 -e utf-16be -- someimage.raw
```

The same, but read from “`stdin`”:

```
cat someimage.raw | stringsext -t x -e utf-8 -e utf-16be -- -
```

Scan a non-file device:

```
stringsext -t x -e utf-8 -e utf-16be -- /dev/sda1
```

Reduce the number of false positives, when scanning for UTF-16LE or UTF-16BE:

```
stringsext -t x --same-unicode-block -e UTF-16le -- someimage.raw
```

Search for Cyrillic only:

```
stringsext -t x -e UTF-16le,,None,Cyrillic -- someimage.raw
```

Search for UTF-16LE encoded Arabic and the digits 0 to 9:

```
stringsext -t x -e UTF-16le,,0x3f000000000000,Arabic -- someimage.raw
```

Search for UTF-8 encoded Syriac and all ASCII, control-codes excluded:

```
stringsext -t x -e UTF-8,,All-Ctrl,0x10000000 -- someimage.raw
```

Combine Little-Endian and Big-Endian scanning:

```
stringsext -t x -e UTF-16be -e UTF-16le -- someimage.raw
```

Show the filter default values used in the above example for debugging:

```
stringsext -d -t x -e UTF-16be -e UTF-16le -- someimage.raw
```

Search for path-names and URLs in some disk-partition:

```
sudo stringsext -t x -e utf-8 -n 15 -g 47 -- /dev/disk/by-uuid/91C8-2721
```

Equivalent to the above:

```
sudo stringsext -t x -e utf-8,15,,,47 -- /dev/disk/by-uuid/91C8-2721
```

OPERATING PRINCIPLE

A *valid* string is a sequence of valid characters according to the encoding chosen with **--encoding**. A valid string may contain *control* characters and *graphic* (visible and human readable) characters. **stringsext** is a tool to extract graphic characters out of binary data streams.

Scanners are parametrized with the **--encoding ENC** option. Multiple scanners may operate in parallel. Their search field is divided into input chunks of "2 * `--output-line-len`" bytes (see source code documentation for details).

Before being printed, valid strings must pass four different **filter**, whose filter criteria are defined with the parameters: *MIN*, *AF*, *UBF* or *GREP* (see above).

LIMITATIONS

The ASCII-character GREP, searched with the "`--grep_char`" option, must appear in the first "`--output-line-len`"bytes to be reliably found in long strings. Increase "`--output-line-len`" if you search for very long strings.

Limitations related to the encoding_rs library

stringsext version 2 uses the external library **encoding_rs** to decode the incoming stream. Compared to the previous library **rust-encoding** used in **stringsext** version 1, the current library has some shortcomings mainly due to the restrictive API policy of the **encoding_rs** project.

1. **stringsext** could be faster, if **encoding_rs** were extensible (**rust-encoding** was): feature request: ASCII-filter · Issue #46 · hsivonen/encoding_rs[1]
2. **stringsext**'s location counter could be more precise if the encoder state were observable: Enhancement: get read access to the decoder's inner state · Issue #48 · hsivonen/encoding_rs[2]
3. **stringsext**'s encoding list could be more up to date, if **encoding_rs**' list were `public`: Make encoding lists public by getreu · Pull Request #47 · hsivonen/encoding_rs[3]

While being desirable, the **stringsext** project does not have the required resources to maintain a fork of **encoding_rs**.

RESOURCES

Project website: https://gitlab.com/getreu/stringsext

COPYING

Copyright (C) 2016-2020 Jens Getreu

Licensed under either of

[1] https://github.com/hsivonen/encoding_rs/issues/46
[2] https://github.com/hsivonen/encoding_rs/issues/48
[3] https://github.com/hsivonen/encoding_rs/pull/47

Contribution

AUTHORS

Jens Getreu `<getreu@web.de>`