#include "internal/utf\_converter.h"

ssize\_t utf8\_to\_utf16(const uint8\_t \*in, uint64\_t in\_len, uint16\_t \*out, uint64\_t max\_out) {

uint64\_t num\_out = 0;

uint64\_t num\_in = 0;

while (num\_in < in\_len) {

uint32\_t uni;

uint64\_t todo;

uint8\_t ch = in[num\_in];

num\_in++;

if (ch <= 0x7F) {

uni = ch;

todo = 0;

} else if (ch <= 0xBF) {

return -1;

} else if (ch <= 0xDF) {

uni = ch & 0x1F;

todo = 1;

} else if (ch <= 0xEF) {

uni = ch & 0x0F;

todo = 2;

} else if (ch <= 0xF7) {

uni = ch & 0x07;

todo = 3;

} else {

return -1;

}

for (uint64\_t j = 0; j < todo; ++j) {

if (num\_in == in\_len) return -1;

uint8\_t ch = in[num\_in];

num\_in++;

if (ch < 0x80 || ch > 0xBF) return -1;

uni <<= 6;

uni += ch & 0x3F;

}

if (uni >= 0xD800 && uni <= 0xDFFF) return -1;

if (uni > 0x10FFFF) return -1;

if (uni <= 0xFFFF) {

if (num\_out == max\_out) return -1;

out[num\_out] = uni;

num\_out++;

} else {

uni -= 0x10000;

if (num\_out + 1 >= max\_out) return -1;

out[num\_out] = (uni >> 10) + 0xD800;

out[num\_out + 1] = (uni & 0x3FF) + 0xDC00;

num\_out += 2;

}

}

if (num\_out == max\_out) return -1;

out[num\_out] = 0;

return num\_out;

}