
Subject: Re: 16 bits wchar

Posted by [cbpporter](#) on Fri, 12 Oct 2007 08:25:07 GMT

[View Forum Message](#) <> [Reply to Message](#)

I created a function that takes a valid or invalid Utf8 string and returns the lenght in bytes of the corresponding error-escaped Utf-8 string. The function utf8codepointEE is an internal function and should not be made public.

```
inline int utf8codepointEE(const byte *s, const byte *z, int &lmod, int &dep)
{
    if (s < z)
    {
        word code = (byte)*s++;
        int codePoint = 0;

        if(code < 0x80)
        {
            dep = 1;
            lmod = 1;
            return code;
        }
        else if (code < 0xC2)
        {
            dep = 1;
            lmod = 3;
            return 0xEE00 + code;
        }
        else if (code < 0xE0)
        {
            if(s >= z)
                return -1;
            if (s[0] < 0x80 || s[0] >= 0xC0)
            {
                dep = 1;
                lmod = 3;
                return 0xEE00 + code;
            }
            codePoint = ((code - 0xC0) << 6) + *s - 0x80;
            if(codePoint < 0x80 || codePoint > 0x07FF)
            {
                dep = 1;
                lmod = 3;
                return 0xEE00 + code;
            }
        }
        else
        {
            dep = 2;
```

```

    lmod = 2;
    return codePoint;
}
}
else if (code < 0xF0)
{
    if(s + 1 >= z)
        return -1;
    if(s[0] < 0x80 || s[0] >= 0xC0 || s[1] < 0x80 || s[1] >= 0xC0)
    {
        dep = 1;
        lmod = 3;
        return 0xEE00 + code;
    }
    codePoint = ((code - 0xE0) << 12) + ((s[0] - 0x80) << 6) + s[1] - 0x80;
    if(codePoint < 0x0800 || codePoint > 0xFFFF)
    {
        dep = 1;
        lmod = 3;
        return 0xEE00 + code;
    }
    else
    {
        dep = 3;
        lmod = 3;
        return codePoint;
    }
}
else if (code < 0xF5)
{
    if(s + 2 >= z)
        return -1;
    if(s[0] < 0x80 || s[0] >= 0xc0 || s[1] < 0x80 || s[1] >= 0xc0 ||
        s[2] < 0x80 || s[2] >= 0xc0)
    {
        dep = 1;
        lmod = 3;
        return 0xEE00 + code;
    }
    codePoint = ((code - 0xf0) << 18) + ((s[0] - 0x80) << 12) +
        ((s[1] - 0x80) << 6) + s[2] - 0x80;
    if(codePoint < 0x010000 || codePoint > 0x10FFFF)
    {
        dep = 1;
        lmod = 3;
        return 0xEE00 + code;
    }
    else

```

```

{
    dep = 1;
    lmod = 3;
    return codePoint;
}
}
else
{
    dep = 1;
    lmod = 3;
    return 0xEE00 + code;
}
}
else
    return -1;
}

int utf8lenEE(const char *_s, int len)
{
    const byte *s = (const byte *)_s;
    const byte *lim = s + len;
    int codePoint = 0;
    int length = 0;
    while(s < lim) {
        int lmod, dep;
        int codePoint = utf8codepointEE(s, lim, lmod, dep);
        if (codePoint == -1)
            return -1;

        length += lmod;
        s += dep;
    }
    return length;
}

```
