
Subject: Re: 16 bits wchar

Posted by [cbporter](#) on Wed, 24 Oct 2007 09:58:48 GMT

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

I've been sick and I didn't leave the house so I couldn't post. But here is my code:

```
int utf8codepointEE(const byte *s, const byte *z, int &lmod, int & dep)
{
if (s < z)
{
dword 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)
{
dep = 1;
lmod = 3;
return 0xEE00 + code;
}
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)
    {
        dep = 1;
        lmod = 3;
        return 0xEE00 + code;
    }
    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)
    {
        dep = 1;
        lmod = 3;
        return 0xEE00 + code;
    }
    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 = 4;
        lmod = 4;
        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);

        ASSERT(codePoint != -1);

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

int utf8lenDeEE(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);

    ASSERT(codePoint != -1);

    if ((codePoint & 0xFFFFF00) == 0xEE00)
    {
        length++;
        s += dep;
    }
    else
    {
        length += lmod;
        s += dep;
    }
}
return length;
}

inline byte * putUtf8(byte *s, int codePoint)
{
if (codePoint < 0x80)
    *s++ = codePoint;
else if (codePoint < 0x0800)
{
    *s++ = 0xC0 | (codePoint >> 6);
    *s++ = 0x80 | (codePoint & 0x3f);
}
else if (codePoint < 0xFFFF)
{
    *s++ = 0xE0 | (codePoint >> 12);
    *s++ = 0x80 | (codePoint >> 6) & 0x3F;
    *s++ = 0x80 | (codePoint & 0x3F);
}
else
{
    *s++ = 0xF0 | (codePoint >> 18);
    *s++ = 0x80 | (codePoint >> 12) & 0x3F;
    *s++ = 0x80 | (codePoint >> 6) & 0x3F;
    *s++ = 0x80 | (codePoint & 0x3F);
}
return s;
}

```

```

String ToUtf8EE(const char *_s, int _len)
{
    int tlen = utf8lenEE(_s, _len);
    if (tlen == -1)
        return "";
    StringBuffer result(tlen);

    byte *s = (byte *) _s;
    const byte *lim = s + _len;

    byte *z = (byte *) ~result;
    int length = 0;
    while(s < lim) {
        int lmod, dep;
        int codePoint = utf8codepointEE(s, lim, lmod, dep);
        if (codePoint == -1)
            return "";
        length += lmod;
        s += dep;

        z = putUtf8(z, codePoint);
    }
    ASSERT(length == tlen);
    return result;
}

String FromUtf8EE(const char *_s, int _len)
{
    int tlen = utf8lenDeEE(_s, _len);
    if (tlen == -1)
        return "";
    StringBuffer result(tlen);

    byte *s = (byte *) _s;
    const byte *lim = s + _len;

    byte *z = (byte *) ~result;
    int length = 0;
    while(s < lim) {
        int lmod, dep;
        int codePoint = utf8codepointEE(s, lim, lmod, dep);
        if (codePoint == -1)
            return "";
        length += lmod;
        s += dep;

        if ((codePoint & 0xFFFFF00) == 0xEE00)
        {
            codePoint -= 0xEE00;

```

```
*z++ = codePoint;
lmod = 1;
}
else
z = putUtf8(z, codePoint);

length += lmod;
s += dep;
}
ASSERT(length == tlen);
return result;
}
```

It is up to you to decide what exactly you want to do with Unicode. And if you let me know, I could help. So please decide, and if you want to leave it as it is, I will find something else to work on.
