Subject: : AmigaOS4
Topic: : vswprintf implementation (need floating point support)
vswprintf implementation (need floating point support)
Author: : kas1e
Date: : 2019/8/27 9:10:00
URL:

@All

I have found 2 vswprintf() implementations which we can use for os4 : one done by Salas00 (in his new unrar port), there is:

```
 #include <wchar.h>
#include <string.h>
#include <stdarg.h>

struct putchdata {
    wchar_t *buffer;
    size_t maxlen, count;
};

static void reverse(char *str, size_t len) {
    if (len > 1) {
        char *start = str;
        char *end = str + len - 1;
        char tmp;

        while (start < end) {
            tmp = *end;
            *end-- = *start;
            *start++ = tmp;
        }
    }
}

static size_t itoa(unsigned num, char *dst, unsigned base, int issigned, int addplus, int uppercase) {
    int a = uppercase ? 'A' : 'a';
    int negative = 0;
    char *d = dst;
    size_t len;

    if (num == 0) {
        *d = '0';
```

```
        return 1;
    }

    if (issigned && (int)num < 0 && base == 10) {
        negative = 1;
        num = -num;
    }

    while (num > 0) {
        unsigned rem = num % base;
        num /= base;
        *d++ = (rem > 9) ? (rem - 10 + a) : (rem + '0');
    }

    if (negative)
        *d++ = '-';
    else if (addplus)
        *d++ = '+';

    len = d - dst;
    reverse(dst, len);

    return len;
}

static size_t lltoa(unsigned long long num, char *dst, unsigned base, int issigned, int addplus, int uppercase) {
    int a = uppercase ? 'A' : 'a';
    int negative = 0;
    char *d = dst;
    size_t len;

    if (num == 0) {
        *d = '0';
        return 1;
    }

    if (issigned && (signed long long)num < 0 && base == 10) {
        negative = 1;
        num = -num;
    }

    while (num > 0) {
        unsigned rem = num % base;
        num /= base;
        *d++ = (rem > 9) ? (rem - 10 + a) : (rem + '0');
    }

    if (negative)
        *d++ = '-';
    else if (addplus)
        *d++ = '+';
```

```
    len = d - dst;
    reverse(dst, len);

    return len;
}

static void putchproc(wchar_t ch, struct putchdata *pcd) {
    size_t index = pcd->count;

    if (ch != '')
        pcd->count++;

    if (pcd->count > pcd->maxlen)
        return;

    if (pcd->count == pcd->maxlen) {
        pcd->buffer[index] =  '';
        return;
    }

    pcd->buffer[index] = ch;
}

#define PUTCH(ch) putchproc((ch), &pcd)

int vswprintf(wchar_t *buffer, size_t maxlen, const wchar_t *fmt, va_list ap) {
    struct putchdata pcd;
    wchar_t ch;
    const wchar_t *src;
    char tmp[128];
    const char *hsrc;

    pcd.buffer = buffer;
    pcd.maxlen = maxlen;
    pcd.count = 0;

    while ((ch = *fmt++) != '') {
        if (ch == '%') {
            int left = 0;
            int addplus = 0;
            int alternate = 0;
            int padzeros = 0;
            size_t width = 0;
            size_t limit = 0;
            int longlong = 0;
            int half = 0;
            size_t len, i;
            int uppercase;
            void *ptr;
            const wchar_t *start = fmt;
```

```
if ((ch = *fmt++) == '')
    goto out;

for (;;) {
    if (ch == '-')
        left = 1;
    else if (ch == '+')
        addplus = 1;
    else if (ch == '#')
        alternate = 1;
    else if (ch == '0')
        padzeros = 1;
    else
        break;

    if ((ch = *fmt++) == '')
        goto out;
}

while (ch >= '0' && ch <= '9') {
    width = 10 * width + (ch - '0');

    if ((ch = *fmt++) == '')
        goto out;
}

if (ch == '.') {
    if ((ch = *fmt++) == '')
        goto out;

    if (ch == '*') {
        limit = va_arg(ap, size_t);

        if ((ch = *fmt++) == '')
            goto out;
    } else {
        while (ch >= '0' && ch <= '9') {
            limit = 10 * limit + (ch - '0');

            if ((ch = *fmt++) == '')
                goto out;
        }
    }
}

if (ch == 'L') {
    longlong = 1;

    if ((ch = *fmt++) == '')
        goto out;
```

```
} else if (ch == 'l') {
    if ((ch = *fmt++) == '')
        goto out;

    if (ch == 'l') {
        longlong = 1;

        if ((ch = *fmt++) == '')
            goto out;
    }
} else if (ch == 'h') {
    half = 1;

    if ((ch = *fmt++) == '')
        goto out;
}

switch (ch) {
    default:
        fmt = start;
        /* Fall through */
    case '%':
        PUTCH('%');
        break;
    case 'C':
    case 'c':
        PUTCH(va_arg(ap, wchar_t));
        break;
    case 'D':
    case 'd':
        uppercase = (ch == 'D');
        if (longlong)
            len = lltoa(va_arg(ap, long long), tmp, 10, 1, addplus, uppercase);
        else
            len = itoa(va_arg(ap, int), tmp, 10, 1, addplus, uppercase);

        hsrc = tmp;

        if (width > len)
            width -= len;
        else
            width = 0;

        if (left == 0) {
            for (i = 0; i < width; i++) PUTCH(padzeros ? '0' : ' ');
        }

        for (i = 0; i < len; i++) PUTCH(hsrc[i]);

        if (left != 0) {
            for (i = 0; i < width; i++) PUTCH(' ');
```

```
        }
        break;
    case 'U':
    case 'u':
        uppercase = (ch == 'U');
        if (longlong)
            len = lltoa(va_arg(ap, unsigned long long), tmp, 10, 0, addplus, uppercase);
        else
            len = itoa(va_arg(ap, unsigned int), tmp, 10, 0, addplus, uppercase);

        hsrc = tmp;

        if (width > len)
            width -= len;
        else
            width = 0;

        if (left == 0) {
            for (i = 0; i < width; i++) PUTCH(padzeros ? '0' : ' ');
        }

        for (i = 0; i < len; i++) PUTCH(hsrc[i]);

        if (left != 0) {
            for (i = 0; i < width; i++) PUTCH(' ');
        }
        break;
    case 'X':
    case 'x':
        uppercase = (ch == 'X');
        if (longlong)
            len = lltoa(va_arg(ap, unsigned long long), tmp, 16, 0, addplus, uppercase);
        else
            len = itoa(va_arg(ap, unsigned int), tmp, 16, 0, addplus, uppercase);

        hsrc = tmp;

        if (width > len)
            width -= len;
        else
            width = 0;

        if (left == 0) {
            for (i = 0; i < width; i++) PUTCH(padzeros ? '0' : ' ');
        }

        for (i = 0; i < len; i++) PUTCH(hsrc[i]);

        if (left != 0) {
            for (i = 0; i < width; i++) PUTCH(' ');
        }
```

```
        break;
    case 'P':
    case 'p':
        uppercase = (ch == 'P');
        ptr = va_arg(ap, void *);
        len = itoa((unsigned)ptr, tmp, 16, 0, 0, uppercase);

        hsrc = tmp;
        width = 8;

        if (width > len)
            width -= len;
        else
            width = 0;

        if (alternate && ptr != NULL) {
            PUTCH('0');
            PUTCH('x');
        }

        for (i = 0; i < width; i++) PUTCH('0');

        for (i = 0; i < len; i++) PUTCH(hsrc[i]);
        break;
    case 'S':
    case 's':
        if (half) {
            hsrc = va_arg(ap, const char *);
            if (hsrc == NULL)
                hsrc = "(null)";

            len = strlen(hsrc);

            if (limit > 0 && len > limit)
                len = limit;

            if (width > len)
                width -= len;
            else
                width = 0;

            if (left == 0) {
                for (i = 0; i < width; i++) PUTCH(' ');
            }

            for (i = 0; i < len; i++) PUTCH(hsrc[i]);

            if (left != 0) {
                for (i = 0; i < width; i++) PUTCH(' ');
            }
        } else {
```

```
                    src = va_arg(ap, const wchar_t *);
                    if (src == NULL)
                        src = L"(null)";

                    len = wcslen(src);

                    if (limit > 0 && len > limit)
                        len = limit;

                    if (width > len)
                        width -= len;
                    else
                        width = 0;

                    if (left == 0) {
                        for (i = 0; i < width; i++) PUTCH(' ');
                    }

                    for (i = 0; i < len; i++) PUTCH(src[i]);

                    if (left != 0) {
                        for (i = 0; i < width; i++) PUTCH(' ');
                    }
                }
                break;
            }
        } else {
            PUTCH(ch);
        }
    }

out:

    PUTCH('');

    return pcd.count;
}

int swprintf(wchar_t *buffer, size_t maxlen, const wchar_t *fmt, ...) {
    va_list ap;
    int wc;

    va_start(ap, fmt);
    wc = vswprintf(buffer, maxlen, fmt, ap);
    va_end(ap);

    return wc;
}
```

And another one i use since some time done by AROS team, there is:

```
 #include <stdarg.h>
#include <stdlib.h>
#include <ctype.h>
#include <wchar.h>
#include <wctype.h>
#include <limits.h>
#include <string.h>


#define ZEROPAD    1        /* pad with zero */
#define SIGN    2        /* unsigned/signed long */
#define PLUS    4        /* show plus */
#define SPACE    8        /* space if plus */
#define LEFT    16        /* left justified */
#define SPECIAL    32        /* 0x */
#define LARGE    64        /* use 'ABCDEF' instead of 'abcdef' */


#define do_div(n,base) ({
int __res;
__res = ((unsigned long long) n) % (unsigned) base;
n = ((unsigned long long) n) / (unsigned) base;
__res; })


static int skip_atoi(const wchar_t **s)
{
    int i=0;

    while (iswdigit(**s))
        i = i*10 + *((*s)++) - L'0';
    return i;
}


static wchar_t *
number (wchar_t *str, long long num, int base, int size, int precision,
    int type)
{
   wchar_t c,sign,tmp[66];
   const wchar_t *digits = L"0123456789abcdefghijklmnopqrstuvwxyz";
   int i;

   if (type & LARGE)
     digits = L"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   if (type & LEFT)
     type &= ~ZEROPAD;
   if (base < 2 || base > 36)
     return 0;
```

```
c = (type & ZEROPAD) ? L'0' : L' ';
sign = 0;

if (type & SIGN)
  {
 if (num < 0)
   {
      sign = L'-';
      num = -num;
      size--;
   }
 else if (type & PLUS)
   {
      sign = L'+';
      size--;
   }
 else if (type & SPACE)
   {
      sign = L' ';
      size--;
   }
  }

if (type & SPECIAL)
  {
 if (base == 16)
    size -= 2;
 else if (base == 8)
    size--;
  }

i = 0;
if (num == 0)
  tmp[i++]='0';
else while (num != 0)
  tmp[i++] = digits[do_div(num,base)];
if (i > precision)
  precision = i;
size -= precision;
if (!(type&(ZEROPAD+LEFT)))
  while(size-->0)
    *str++ = L' ';
if (sign)
  *str++ = sign;
if (type & SPECIAL)
  {
 if (base==8)
   {
      *str++ = L'0';
   }
```

```
  else if (base==16)
    {
      *str++ = L'0';
      *str++ = digits[33];
    }
   }
  if (!(type & LEFT))
    while (size-- > 0)
      *str++ = c;
  while (i < precision--)
    *str++ = '0';
  while (i-- > 0)
    *str++ = tmp[i];
  while (size-- > 0)
    *str++ = L' ';
  return str;
}


int _vsnwprintf(wchar_t *buf, size_t cnt, const wchar_t *fmt, va_list args)
{
   int len;
   unsigned long long num;
   int i, base;
   wchar_t * str;
   const char *s;
   const wchar_t *sw;

   int flags;        /* flags to number() */

   int field_width;    /* width of output field */
   int precision;       /* min. # of digits for integers; max
                number of chars for from string */
   int qualifier;       /* 'h', 'l', 'L', 'w' or 'I' for integer fields */

   for (str=buf ; *fmt ; ++fmt) {
       if (*fmt != L'%') {
           *str++ = *fmt;
           continue;
       }

       /* process flags */
       flags = 0;
       repeat:
           ++fmt;        /* this also skips first '%' */
           switch (*fmt) {
               case L'-': flags |= LEFT; goto repeat;
               case L'+': flags |= PLUS; goto repeat;
               case L' ': flags |= SPACE; goto repeat;
               case L'#': flags |= SPECIAL; goto repeat;
               case L'0': flags |= ZEROPAD; goto repeat;
```

```
    }

/* get field width */
field_width = -1;
if (iswdigit(*fmt))
    field_width = skip_atoi(&fmt);
else if (*fmt == L'*') {
    ++fmt;
    /* it's the next argument */
    field_width = va_arg(args, int);
    if (field_width < 0) {
        field_width = -field_width;
        flags |= LEFT;
    }
}

/* get the precision */
precision = -1;
if (*fmt == L'.') {
    ++fmt;
    if (iswdigit(*fmt))
        precision = skip_atoi(&fmt);
    else if (*fmt == L'*') {
        ++fmt;
        /* it's the next argument */
        precision = va_arg(args, int);
    }
    if (precision < 0)
        precision = 0;
}

/* get the conversion qualifier */
qualifier = -1;
if (*fmt == 'h' || *fmt == 'l' || *fmt == 'L' || *fmt == 'w') {
    qualifier = *fmt;
    ++fmt;
} else if (*fmt == 'I' && *(fmt+1) == '6' && *(fmt+2) == '4') {
    qualifier = *fmt;
    fmt += 3;
}

/* default base */
base = 10;

switch (*fmt) {
case L'c':
    if (!(flags & LEFT))
        while (--field_width > 0)
            *str++ = L' ';
    if (qualifier == 'h')
        *str++ = (wchar_t) va_arg(args, int);
```

```
    else
        *str++ = (wchar_t) va_arg(args, int);
    while (--field_width > 0)
        *str++ = L' ';
    continue;

case L'C':
    if (!(flags & LEFT))
        while (--field_width > 0)
            *str++ = L' ';
    if (qualifier == 'l' || qualifier == 'w')
        *str++ = (wchar_t) va_arg(args, int);
    else
        *str++ = (wchar_t) va_arg(args, int);
    while (--field_width > 0)
        *str++ = L' ';
    continue;

case L's':
    if (qualifier == 'h') {
        /* print ascii string */
        s = va_arg(args, char *);
        if (s == NULL)
            s = "<NULL>";

        len = strlen (s);
        if ((unsigned int)len > (unsigned int)precision)
            len = precision;

        if (!(flags & LEFT))
            while (len < field_width--)
                *str++ = L' ';
        for (i = 0; i < len; ++i)
            *str++ = (wchar_t)(*s++);
        while (len < field_width--)
            *str++ = L' ';
    } else {
        /* print unicode string */
        sw = va_arg(args, wchar_t *);
        if (sw == NULL)
            sw = L"<NULL>";

        len = wcslen (sw);
        if ((unsigned int)len > (unsigned int)precision)
            len = precision;

        if (!(flags & LEFT))
            while (len < field_width--)
                *str++ = L' ';
        for (i = 0; i < len; ++i)
            *str++ = *sw++;
```

```
        while (len < field_width--)
            *str++ = L' ';
    }
    continue;

case L'S':
    if (qualifier == 'l' || qualifier == 'w') {
        /* print unicode string */
        sw = va_arg(args, wchar_t *);
        if (sw == NULL)
            sw = L"<NULL>";

        len = wcslen (sw);
        if ((unsigned int)len > (unsigned int)precision)
            len = precision;

        if (!(flags & LEFT))
            while (len < field_width--)
                *str++ = L' ';
        for (i = 0; i < len; ++i)
            *str++ = *sw++;
        while (len < field_width--)
            *str++ = L' ';
    } else {
        /* print ascii string */
        s = va_arg(args, char *);
        if (s == NULL)
            s = "<NULL>";

        len = strlen (s);
        if ((unsigned int)len > (unsigned int)precision)
            len = precision;

        if (!(flags & LEFT))
            while (len < field_width--)
                *str++ = L' ';
        for (i = 0; i < len; ++i)
            *str++ = (wchar_t)(*s++);
        while (len < field_width--)
            *str++ = L' ';
    }
    continue;
case L'p':
    if (field_width == -1) {
        field_width = 2*sizeof(void *);
        flags |= ZEROPAD;
    }
    str = number(str,
        (unsigned long) va_arg(args, void *), 16,
        field_width, precision, flags);
    continue;
```

```
case L'n':
    if (qualifier == 'l') {
        long * ip = va_arg(args, long *);
        *ip = (str - buf);
    } else {
        int * ip = va_arg(args, int *);
        *ip = (str - buf);
    }
    continue;

/* integer number formats - set up the flags and "break" */
case L'o':
    base = 8;
    break;

case L'b':
    base = 2;
    break;

case L'X':
    flags |= LARGE;
case L'x':
    base = 16;
    break;

case L'd':
case L'i':
    flags |= SIGN;
case L'u':
    break;

default:
    if (*fmt != L'%')
        *str++ = L'%';
    if (*fmt)
        *str++ = *fmt;
    else
        --fmt;
    continue;
}

if (qualifier == 'l')
    num = va_arg(args, unsigned long long);
else if (qualifier == 'l')
    num = va_arg(args, unsigned long);
else if (qualifier == 'h') {
    if (flags & SIGN)
        num = va_arg(args, int);
    else
        num = va_arg(args, unsigned int);
```

```
        }
        else {
            if (flags & SIGN)
                num = va_arg(args, int);
            else
                num = va_arg(args, unsigned int);
        }
        str = number(str, num, base, field_width, precision, flags);
    }
    *str = L'';
    return str-buf;
}


int swprintf(wchar_t *buf, size_t n,const wchar_t *fmt, ...)
{
    va_list args;
    int i;

    va_start(args, fmt);
    i=_vsnwprintf(buf,n,fmt,args);
    va_end(args);
    return i;
}


int _snwprintf(wchar_t *buf, size_t cnt, const wchar_t *fmt, ...)
{
    va_list args;
    int i;

    va_start(args, fmt);
    i=_vsnwprintf(buf,cnt,fmt,args);
    va_end(args);
    return i;
}


int vswprintf(wchar_t *buf, size_t n, const wchar_t *fmt, va_list args)
{
    return _vsnwprintf(buf,n,fmt,args);
}
```

Both of them working, but both of them didn't handle floating point. I.e. there in both realisations no case 'f': code, and
because of that , when i want to do something like this:

```
swprintf(tmp, 255, L"triangles:%0.3f mio/s", driver->getPrimitiveCountDrawn(1) * (1.f / 1000000.f));
```

Then instead of necessary value (which should be 0.xxxx), i have some random/garbage character.

Is there anyone who can wrote missing case 'f': in any of those examples ? I assume it will be just copy of case 'd' with some (?) little changes.

Checking google for different vsnwprintf() implementations didn't help much at moment.

Thanks !