utility.cm

```
/*
    ================================================================================

    Copyright (c) 2012-2016 Seppo Laakko
    http://sourceforge.net/projects/cmajor/

    Distributed under the GNU General Public License, version 3 (GPLv3).
    (See accompanying LICENSE.txt or http://www.gnu.org/licenses/gpl.html
        )

 ================================================================================
    */

// Copyright (c) 1994
// Hewlett-Packard Company
// Copyright (c) 1996
// Silicon Graphics Computer Systems, Inc.
// Copyright (c) 2009 Alexander Stepanov and Paul McJones

namespace System.Support
{
    public nothrow inline ulong Align(ulong n, ulong alignment)
    {
        #assert(alignment > 0u);
        return (n + alignment - 1u) & -alignment;
    }

    public nothrow int StrLen(const char* s)
    {
        int len = 0;
        if (s != null)
        {
            while (*s != '\0')
            {
                ++len;
                ++s;
            }
        }
        return len;
    }

    public nothrow int StrLen(const wchar* s)
    {
        int len = 0;
        if (s != null)
        {
            while (*s != '\0')
            {
                ++len;
                ++s;
            }
        }
        return len;
    }

    public nothrow int StrLen(const uchar* s)
    {
        int len = 0;
        if (s != null)
        {
            while (*s != '\0')
            {
                ++len;
                ++s;
            }
        }
        return len;
    }

    public nothrow void StrCopy(char* buf, const char* from)
    {
        #assert(buf != null);
        if (from != null)
        {
            while (*from != '\0')
            {
                *buf++ = *from++;
            }
        }
        *buf = '\0';
    }

    public nothrow void StrCopy(wchar* buf, const wchar* from)
    {
        #assert(buf != null);
        if (from != null)
        {
            while (*from != '\0')
            {
                *buf++ = *from++;
            }
        }
        *buf = '\0';
    }

    public nothrow void StrCopy(uchar* buf, const uchar* from)
    {
        #assert(buf != null);
        if (from != null)
        {
            while (*from != '\0')
            {
                *buf++ = *from++;
            }
        }
        *buf = '\0';
    }

    public nothrow int StrCopy(char* buf, const char* from, int length)
    {
        int resultLen = 0;
        #assert(buf != null);
        if (from != null)
        {
            while (resultLen < length)
            {
                if (*from == '\0')
                {
                    break;
                }
                *buf++ = *from++;
                ++resultLen;
            }
        }
        *buf = '\0';
        return resultLen;
    }

    public nothrow int StrCopy(wchar* buf, const wchar* from, int length)
    {
        int resultLen = 0;
        #assert(buf != null);
        if (from != null)
        {
            while (resultLen < length)
            {
                if (*from == '\0')
                {
                    break;
                }
                *buf++ = *from++;
                ++resultLen;
            }
        }
        *buf = '\0';
        return resultLen;
    }

    public nothrow int StrCopy(uchar* buf, const uchar* from, int length)
    {
        int resultLen = 0;
        #assert(buf != null);
        if (from != null)
        {
            while (resultLen < length)
            {
                if (*from == '\0')
                {
                    break;
                }
                *buf++ = *from++;
                ++resultLen;
            }
        }
        *buf = '\0';
        return resultLen;
    }

    public nothrow ulong MemGrow(ulong size)
    {
        if (size < 8u)
        {
            return 8u;
        }
        else if (size < 64u)
        {
            return 64u;
        }
        else if (size < 512u)
        {
            return 512u;
        }
        else if (size < 4096u)
        {
            return 4096u;
        }
        else
        {
            return 2u * Align(size, 4096u);
        }
    }

    public nothrow int Write(int fd, const char* s)
    {
        return write_64(fd, s, cast<ulong>(StrLen(s)));
    }

    public const int EXIT_INSUFFICIENT_MEMORY = 253;

    public nothrow void* MemAlloc(ulong size)
    {
        void* block = malloc(size);
        if (block == null)
        {
            Write(stderr, "insufficient memory\n");
            exit(EXIT_INSUFFICIENT_MEMORY);
        }
        return block;
    }
    public nothrow void MemFree(void* block)
    {
        free(block);
    }

    public nothrow void* DebugHeapMemAlloc(ulong size)
    {
        void* block = dbgheap_malloc(size);
        if (block == null)
        {
            Write(stderr, "insufficient memory\n");
            exit(EXIT_INSUFFICIENT_MEMORY);
        }
        return block;
    }

    public nothrow void DebugHeapMemFree(void* block)
    {
        dbgheap_free(block);
    }
}
```