19 декабря 2016 г.
17:20

```
module base;

version = Dinrus;
version = Unicode;
version = ЛитлЭндиан;

int useWfuncs = 1;
alias useWfuncs __ЮНИКОД__;

//Константы
const бул нет = false;
const бул да = true;
const пусто = null;
alias пусто NULL;
//const неук = проц; //получить тип от инициализатора не удаётся...

/* ************* Константы *************** */
/** Строка, используемая для разделения в пути названий папок.
     * для Windows это обратный слэш, для Linux - прямой. */
const сим[1] РАЗДПАП = '\\';
/** Альтернативная версия разделителя, используемая на Windows (слэш).
* Для Linux - пустая константа. */
const сим[1] АЛЬТРАЗДПАП = '/';
/** Строка разделитель пути. На Windows точка с запятой, на
* Linux - двоеточие. */
const сим[1] РАЗДПСТР = ';';
/** Строка, используемая для разделения строк, \r\n на Windows и \n
* на Linux. */
const сим[2] РАЗДСТР = "\r\n"; /// Строка-разделитель строк.
const сим[1] ТЕКПАП = '.';     /// Строка, представляющая текущую папку.
const сим[2] РОДПАП = ".."; /// Строка, представляющая родительскую папку
(папку на уровень выше).

const сим[16] ЦИФРЫ16 = "0123456789ABCDEF";            /// 0..9A..F
const сим[10] ЦИФРЫ = "0123456789";            /// 0..9
const сим[8] ЦИФРЫ8 = "01234567";               /// 0..7
const сим[92] ПРОПИСНЫЕ =
"abcdefghijklmnopqrstuvwxyzабвгдеёжзийклмнопрстуфхцчшщъыьэюя";    /// a..z
a..я
const сим[92] СТРОЧНЫЕ =
"ABCDEFGHIJKLMNOPQRSTUVWXYZАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";    /// A..Z
A..Я
const сим[184] БУКВЫ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
"abcdefghijklmnopqrstuvwxyz" "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"
"абвгдеёжзийклмнопрстуфхцчшщъыьэюя";    /// A..Za..zA..Яа..я

const сим[6] ПРОБЕЛЫ = " \t\v\r\n\f";            /// пробелы в ASCII

const сим[3] РС = \u2028; /// Разделитель строк Unicode.
const дим РСд = 0x2028; /// определено
const сим[3] РА = \u2029; /// Разделитель абзацев Unicode.
const дим РАд = 0x2029; /// определено
static assert(РС[0] == РА[0] && РС[1] == РА[1]);

/// Метка перехода на новую строку для данной системы
version (Windows)
const сим[2] НОВСТР = РАЗДСТР;
else version (Posix)
const сим[1] НОВСТР = '\n';
```

```
/// Перечень меток порядка байтов UTF (BOM)
enum МПБ {
     Ю8,         /// UTF-8
     Ю16ЛЕ,     /// UTF-16 Little Эндиан
     Ю16БЕ,     /// UTF-16 Big Эндиан
     Ю32ЛЕ,     /// UTF-32 Little Эндиан
     Ю32БЕ,     /// UTF-32 Big Эндиан
     Нет
}

private const цел ЧМПБ = 5;

Эндиан[ЧМПБ] МПБЭндиан =
[ _эндиан,
Эндиан.Литл, Эндиан.Биг,
Эндиан.Литл, Эндиан.Биг
];

ббайт[][ЧМПБ] МеткиПорядкаБайтов =
[ [0xEF, 0xBB, 0xBF],
[0xFF, 0xFE],
[0xFE, 0xFF],
[0xFF, 0xFE, 0x00, 0x00],
[0x00, 0x00, 0xFE, 0xFF]
];


const{

version(ЛитлЭндиан)
{
private const цел Эндиан_Амбьент = 1;
}

version(БигЭндиан)
{
private const цел Эндиан_Амбьент = 2;
}

enum Эндиан
{
Неизвестно = 0, //!< Неизвестно эндиан-ness. Indicates an ошиб
Литл = 1, //!< Литтл эндиан architecture
Биг = 2 , //!< Биг эндиан architecture
Мидл = 3 , //!< Миддл эндиан architecture
БайтСекс = 4,
Амбьент = Эндиан_Амбьент
}

/*
     enum Эндиан
     {
          БигЭндиан,     /// порядок байта, когда его завершает наибольший -
          старший - бит
          ЛитлЭндиан     /// порядок байта, когда в конце него находится
          младший бит
     }

     */
     version(ЛитлЭндиан)
     {
          /// Исконная "эндианность" системы
               Эндиан _эндиан = Эндиан.Литл;
```

```
    }
    else
    {
            Эндиан _эндиан = Эндиан.Биг;
    }
}
    typedef дол т_время;
typedef бцел ФВремяДос;
const т_время т_время_нч = cast(т_время) дол.min;
alias сим рсим;
alias ткст рткст;
/+
alias typeof(delegate) делегат;
alias typeof(function) функция;
alias typeof(struct) структ;
alias typeof(union) союз;
+/

version(ЛитлЭндиан) {
static assert(реал.mant_dig == 53 || реал.mant_dig==64
|| реал.mant_dig == 113,
"Только 64-битные, 80-битные и 128-битные реалы"
" поддерживаются для процессоров ЛитлЭндиан");
} else {
static assert(реал.mant_dig == 53 || реал.mant_dig==106
|| реал.mant_dig == 113,
"Только 64-битные и 128-битные реалы поддерживаются для процессоров
БигЭндиан."
" дво-дво реалы имеют частичную поддержку.");
}

// Константы, используемые для извлечения компонентов из их представлений.
// Они обслуживают встроеннные свойства плавающей точки.
template плавТрэтс(T) {
// МАСКАВЫР - это бкрат маска для выделения экспонентной части (без знака)
// СТП2ЧИСМАНТ = pow(2, реал.mant_dig): ЭТО ЗНАЧЕНИЕis the value such that
// (smallest_denormal)*СТП2ЧИСМАНТ == реал.min
// ПОЗВЫР_КРАТ is the index of the exponent when represented as a ushort
array.
// ПОЗЗНАКА_БАЙТ is the index of the sign when represented as a ббайт array.
static if (T.mant_dig == 24) { // float
const бкрат МАСКАВЫР = 0x7F80;
const бкрат БИАСВЫР = 0x3F00;
const uint МАСКАВЫР_ЦЕЛ = 0x7F80_0000;
const uint МАСКАМАНТИССЫ_ЦЕЛ = 0x007F_FFFF;
const реал СТП2ЧИСМАНТ = 0x1p+24;
version(ЛитлЭндиан) {
const ПОЗВЫР_КРАТ = 1;
} else {
const ПОЗВЫР_КРАТ = 0;
}
} else static if (T.mant_dig == 53) { // double, or реал==double
const бкрат МАСКАВЫР = 0x7FF0;
const бкрат БИАСВЫР = 0x3FE0;
const uint МАСКАВЫР_ЦЕЛ = 0x7FF0_0000;
const uint МАСКАМАНТИССЫ_ЦЕЛ = 0x000F_FFFF; // for the MSB only
const реал СТП2ЧИСМАНТ = 0x1p+53;
version(ЛитлЭндиан) {
const ПОЗВЫР_КРАТ = 3;
const ПОЗЗНАКА_БАЙТ = 7;
} else {
const ПОЗВЫР_КРАТ = 0;
const ПОЗЗНАКА_БАЙТ = 0;
}
```

```
} else static if (Т.mant_dig == 64) { // real80
const бкрат МАСКАВЫР = 0x7FFF;
const бкрат БИАСВЫР = 0x3FFE;
const реал СТП2ЧИСМАНТ = 0x1p+63;
version(ЛитлЭндиан) {
const ПОЗВЫР_КРАТ = 4;
const ПОЗЗНАКА_БАЙТ = 9;
} else {
const ПОЗВЫР_КРАТ = 0;
const ПОЗЗНАКА_БАЙТ = 0;
}
} else static if (реал.mant_dig == 113){ // quadruple
const бкрат МАСКАВЫР = 0x7FFF;
const реал СТП2ЧИСМАНТ = 0x1p+113;
version(ЛитлЭндиан) {
const ПОЗВЫР_КРАТ = 7;
const ПОЗЗНАКА_БАЙТ = 15;
} else {
const ПОЗВЫР_КРАТ = 0;
const ПОЗЗНАКА_БАЙТ = 0;
}
} else static if (реал.mant_dig == 106) { // doubledouble
const бкрат МАСКАВЫР = 0x7FF0;
const реал СТП2ЧИСМАНТ = 0x1p+53; // doubledouble denormals are strange
// and the exponent byte is not unique
version(ЛитлЭндиан) {
const ПОЗВЫР_КРАТ = 7; // [3] is also an exp short
const ПОЗЗНАКА_БАЙТ = 15;
} else {
const ПОЗВЫР_КРАТ = 0; // [4] is also an exp short
const ПОЗЗНАКА_БАЙТ = 0;
}
}
}

// These apply to all floating-point types
version(ЛитлЭндиан) {
const МАНТИССА_МЗЧ = 0; //LSB
const МАНТИССА_БЗЧ = 1; //MSB
} else {
const МАНТИССА_МЗЧ = 1;
const МАНТИССА_БЗЧ = 0;
}

/**
*Константы математического характера
*/

const реал ЛОГ2Т = 0x1.a934f0979a3715fcp+1;
const реал ЛОГ2Е = 0x1.71547652b82fe178p+0;
const реал ЛОГ2 = 0x1.34413509f79fef32p-2;
const реал ЛОГ10Е = 0.43429448190325182765;
const реал ЛН2 = 0x1.62e42fefa39ef358p-1;
const реал ЛН10 = 2.30258509299404568402;
const реал ПИ = 0x1.921fb54442d1846ap+1;
const реал ПИ_2 = 1.57079632679489661923;
const реал ПИ_4 = 0.78539816339744830962;
const реал М_1_ПИ = 0.31830988618379067154;
const реал М_2_ПИ = 0.63661977236758134308;
const реал М_2_КВКОРПИ = 1.12837916709551257390;
const реал КВКОР2 = 1.41421356237309504880;
const реал КВКОР1_2 = 0.70710678118654752440;

const реал Е = 2.7182818284590452354L;
```

```
const реал МАКСЛОГ = 0x1.62e42fefa39ef358p+13L; /** лог(реал.max) */
const реал МИНЛОГ = -0x1.6436716d5406e6d8p+13L; /**
лог(реал.min*реал.epsilon) */
const реал ОЙЛЕРГАММА =
0.57721_56649_01532_86060_65120_90082_40243_10421_59335_93992L; /** Euler-
Mascheroni constant 0.57721566.. */

const ткст ИМЕЙЛ =
r"[a-zA-Z]([.]?([[a-zA-Z0-9_]-]+)*)?@([[a-zA-Z0-9_]\-_]+\.)+[a-zA-Z]{2,6}";
     const ткст УРЛ =
     r"(([h|H][t|T]|[f|F])[t|T][p|P]([s|S]?)\:\/\/|~/|/)?([\w]+:\w+@)?(([a-zA-
     Z]{1}([\w\-
     ]+\.)+([\w]{2,5}))(:[\d]{1,5})?)?((/?\w+/)+|/?)(\w+\.[\w]{3,4})?([,]\w+)*
     ((\?\w+=\w+)?(&\w+=\w+)*([,]\w*)*)?";

enum
{
     ЧасовВДне = 24,
     МинутВЧасе = 60,
     МсекВМинуте = 60 * 1000,
     МсекВЧасе = 60 * МсекВМинуте,
     МсекВДень = 86400000,
     ТиковВМсек = 1,
     ТиковВСекунду = 1000,
     ТиковВМинуту = ТиковВСекунду * 60,
     ТиковВЧас = ТиковВМинуту * 60,
     ТиковВДень = ТиковВЧас * 24,
}

enum ПМангл : сим
{
Тпроц = 'v',
Тбул = 'g',
Тбайт = 'b',
Тббайт = 'h',
Ткрат = 's',
Тбкрат = 't',
Тцел = 'i',
Тбцел = 'k',
Тдол = 'l',
Тбдол = 'm',
Тплав = 'f',
Тдво = 'd',
Треал = 'e',

Твплав = 'o',
Твдво = 'p',
Твреал = 'j',
Ткплав = 'q',
Ткдво = 'r',
Ткреал = 'c',

Тсим = 'a',
Тшим = 'u',
Тдим = 'w',

Тмассив = 'A',
Тсмассив = 'G',
Тамассив = 'H',
Туказатель = 'P',
Тфункция = 'F',
Тидент = 'I',
Ткласс = 'C',
```

```
Тструкт = 'S',
Тперечень = 'E',
Ттипдеф = 'T',
Тделегат = 'D',

Тконст = 'x',
Тинвариант = 'y',
}

enum
{
СБОЙ_ОП = ~cast(т_мера)0
}

enum
{ РАЗМЕР_СТРАНИЦЫ = 4096,
РАЗМЕР_ПОДАЧИ = (4096*16),
РАЗМЕР_ПУЛА = (4096*256),
}
const т_мера МАСКА_СТРАНИЦЫ = РАЗМЕР_СТРАНИЦЫ - 1;

    /**
* Элементы бит-поля, представляющего атрибуты блока памяти. Ими
* можно манипулировать функциями СМ дайАтр, устАтр, удалиАтр.
*/
enum ПАтрБлока : бцел
{
Финализовать = 0b0000_0001, /// Финализовать данные этого блока при сборке.
НеСканировать = 0b0000_0010, /// Не сканировать при сборке данный блок.
НеПеремещать = 0b0000_0100, /// Не перемещать данный блок при сборке.
         МожноДобавить = 0b0000_1000,
БезНутра = 0b0001_0000,
         ВсеБиты = 0b1111_1111
}
    interface ИнфоОтслежИскл
{
цел opApply( цел delegate( inout ткст ) );
}

version( Windows )
{
alias wchar wint_t, wchar_t, wctype_t, wctrans_t;

const wchar WEOF = 0xFFFF;
    alias WEOF ШКФ;
}
else
{
alias dchar wint_t, wchar_t, wctype_t, wctrans_t;

const dchar WEOF = 0xFFFF;
    alias WEOF ШКФ;
}

extern(Windows)
{

alias UINT SOCKET;
alias int socklen_t;

}

    alias DWORD ACCESS_MASK;
alias ACCESS_MASK *PACCESS_MASK;
```

```
alias ACCESS_MASK REGSAM;
    alias extern (Windows) int function() FARPROC, NEARPROC, PROC, ПРОЦУК;
extern (Windows)
{
version (0)
{ // Properly prototyped versions
alias BOOL function(HWND, UINT, WPARAM, LPARAM) DLGPROC;
alias VOID function(HWND, UINT, UINT, DWORD) TIMERPROC;
alias BOOL function(HDC, LPARAM, int) GRAYSTRINGPROC;
alias BOOL function(HWND, LPARAM) WNDENUMPROC;
alias LRESULT function(int code, WPARAM wParam, LPARAM lParam) HOOKPROC;
alias VOID function(HWND, UINT, DWORD, LRESULT) SENDASYNCPROC;
alias BOOL function(HWND, LPCSTR, HANDLE) PROPENUMPROCA;
alias BOOL function(HWND, LPCWSTR, HANDLE) PROPENUMPROCW;
alias BOOL function(HWND, LPSTR, HANDLE, DWORD) PROPENUMPROCEXA;
alias BOOL function(HWND, LPWSTR, HANDLE, DWORD) PROPENUMPROCEXW;
alias int function(LPSTR lpch, int ichCurrent, int cch, int code)
EDITWORDBREAKPROCA;
alias int function(LPWSTR lpch, int ichCurrent, int cch, int code)
EDITWORDBREAKPROCW;
alias BOOL function(HDC hdc, LPARAM lData, WPARAM wData, int cx, int cy)
DRAWSTATEPROC;
}
else
{
alias FARPROC DLGPROC;
alias FARPROC TIMERPROC;
alias FARPROC GRAYSTRINGPROC;
alias FARPROC WNDENUMPROC;
alias FARPROC HOOKPROC;
alias FARPROC SENDASYNCPROC;
alias FARPROC EDITWORDBREAKPROCA;
alias FARPROC EDITWORDBREAKPROCW;
alias FARPROC PROPENUMPROCA;
alias FARPROC PROPENUMPROCW;
alias FARPROC PROPENUMPROCEXA;
alias FARPROC PROPENUMPROCEXW;
alias FARPROC DRAWSTATEPROC;
}
}
//Базовые типы языка Динрус
version( X86_64 )
{
alias ulong т_мера, size_t;
alias long т_дельтаук, ptrdiff_t;
}
else
{
alias uint т_мера, size_t;
alias int т_дельтаук, ptrdiff_t;
}

alias т_мера т_хэш, hash_t;

alias char[] ткст, симма, string;
alias char[] *уткст, усимма;

alias wchar[] шткст, шимма, wstring;
alias wchar[] *ушткст, ушимма;

alias dchar[] юткст, димма, dstring;
alias dchar[] *юткст, удимма;

alias bool бул, бит, bit, BOOLEAN;
```

```
alias bool *убул, PBOOLEAN;

extern (C){

     version( Windows )
     {
          alias int c_long;
          alias uint c_ulong;
     }
     else
     {
     static if( (ук).sizeof > int.sizeof )
     {
          alias long c_long;
          alias ulong c_ulong;
     }
     else
     {
          alias int c_long;
          alias uint c_ulong;
     }
     }

alias int sig_atomic_t;
     alias byte int8_t;
     alias short int16_t;
     alias int int32_t;
     alias long int64_t;
     //alias cent int128_t;

     alias ubyte uint8_t;
     alias ushort uint16_t;
     alias uint uint32_t;
     alias ulong uint64_t;
     //alias ucent uint128_t;

     alias byte int_least8_t;
     alias short int_least16_t;
     alias int int_least32_t;
     alias long int_least64_t;

     alias ubyte uint_least8_t;
     alias ushort uint_least16_t;
     alias uint uint_least32_t;
     alias ulong uint_least64_t;

     alias byte int_fast8_t;
     alias int int_fast16_t;
     alias int int_fast32_t;
     alias long int_fast64_t;

     alias ubyte uint_fast8_t;
     alias uint uint_fast16_t;
     alias uint uint_fast32_t;
     alias ulong uint_fast64_t;

     version( X86_64 )
     {
          alias long intptr_t;
          alias ulong uintptr_t;
     }
     else
     {
          alias int intptr_t;
```

```
        alias uint uintptr_t;
    }

    alias long intmax_t, т_максцел;
    alias ulong uintmax_t, т_максбцел;
}

////////////////////////////

alias int цел, т_рав, т_фпоз, equals_t, fexcept_t, fpos_t,LONG, BOOL,WINBOOL,
HFILE, INT, LONG32, INT32;
alias int *уцел, PINT, LPINT, LPLONG, PWINBOOL, LPWINBOOL, PBOOL, LPBOOL,
PLONG, PLONG32, PINT32;
alias LONG HRESULT, SCODE, LPARAM, LRESULT;
    alias uint бцел, ЛКИД, DWORD, UINT, ULONG, FLONG, LCID,ULONG32, DWORD32,
    UINT32;
alias uint *убцел, PULONG, PUINT, PLCID, LPUINT, PULONG32, PDWORD32, PUINT32;
alias UINT WPARAM;
alias DWORD COLORREF;
alias DWORD *PDWORD, LPDWORD, LPCOLORREF;

alias long дол, LONGLONG, USN, LONG64, INT64;
alias long *удол, PLONGLONG,PLONG64, PINT64;

alias ulong бдол, ULONG64, DWORD64, UINT64, DWORDLONG, ULONGLONG;
alias ulong *убдол, PULONG64, PDWORD64, PUINT64, PDWORDLONG, PULONGLONG;

alias real реал;
alias real *уреал;

alias double дво, double_t;
alias double *удво;

alias char сим, CHAR, CCHAR;
alias char *усим, ткст0, PSZ, PCHAR;
alias CHAR *LPSTR, PSTR, LPCSTR, PCSTR;
//alias LPSTR LPTCH, PTCH, PTSTR, LPTSTR, LPCTSTR;

alias wchar шим, WCHAR, OLECHAR;
alias wchar *ушим, шткст0, PWCHAR, LPWCH, PWCH, LPWSTR, PWSTR, LPCWSTR,
PCWSTR, LPOLESTR, LPCOLESTR;
////////////////////////////////////////////////

enum : BOOL
{
FALSE = 0,
TRUE = 1,
}

version(Unicode) {
alias WCHAR TCHAR, _TCHAR;
} else {
alias CHAR TCHAR, _TCHAR;
}

alias TCHAR* PTCH, PTBYTE, LPTCH, PTSTR, LPTSTR, LP, PTCHAR, LPCTSTR;
alias TCHAR TBYTE;

alias dchar дим;
alias dchar *удим;

alias byte байт, FCHAR, INT8;
alias byte *убайт, PINT8;
```

```
alias ubyte ббайт, BYTE, UCHAR,UINT8;
alias ubyte *уббайт, PUINT8;
alias UCHAR *PUCHAR;
alias BYTE *PBYTE, LPBYTE;

alias short крат, SHORT,INT16;
alias short *украт, PSHORT, PINT16;

alias ushort бкрат, ИДЯз, WORD, USHORT, LANGID, FSHORT,UINT16;
alias ushort *убкрат,PUINT16;
alias USHORT *PUSHORT;
alias WORD ATOM, ATOM;
alias WORD *PWORD, LPWORD;

alias float плав, float_t, FLOAT;
alias float *уплав, PFLOAT;

alias void проц, VOID;
alias void *ук, спис_ва, va_list, HANDLE, PVOID, LPVOID, LPCVOID, PVOID64,
PCVOID;
alias HANDLE HINST, HGLOBAL, HLOCAL, HWND, HINSTANCE, HGDIOBJ, HACCEL,
HBITMA, HBRUSH, HCOLORSPACE, HDC, HGLRC, HDESK, HENHMETAFILE, HFONT, HICON,
HMENU, HMETAFILE, HPALETTE, HPEN, HRGN, HRSRC, HMONITOR, HSTR, HTASK,
HWINSTA, HKEY, HKL, HBITMAP;

alias HANDLE* PHANDLE, LPHANDLE;


version (Windows)
alias ук Дескр;
else
typedef цел Дескр = -1;
             alias ук ДЕСКР;
alias ДЕСКР гук, лук, экз;

alias HINSTANCE HMODULE;
alias HICON HCURSOR;
alias HKEY *PHKEY;

alias COLORREF ЦВПредст;
alias HBRUSH УКисть;
alias HCURSOR УКурсор;
alias HPEN УПеро;
alias HICON УПиктограмма, УИконка;
alias HFONT УШрифт;
alias HWND УОК;

alias ireal вреал;
alias ireal *увреал;

alias idouble вдво;
alias idouble *увдво;

alias ifloat вплав;
alias ifloat *увплав;

alias creal креал;
alias creal *укреал;

alias cdouble кдво;
alias cdouble *укдво;

alias cfloat кплав;
alias cfloat *укплав;
```

```
// ULONG_PTR должен способствовать сохранению указателя в виде целочисленного типа
version (Win64)
{
     alias long __int3264;
     const ulong ADDRESS_TAG_BIT = 0x40000000000;

     alias long INT_PTR, LONG_PTR;
     alias long* PINT_PTR, PLONG_PTR;
     alias ulong UINT_PTR, ULONG_PTR, HANDLE_PTR;
     alias ulong* PUINT_PTR, PULONG_PTR;
     alias int HALF_PTR;
     alias int* PHALF_PTR;
     alias uint UHALF_PTR;
     alias uint* PUHALF_PTR;
}
else // Win32
{
     alias int __int3264;
     const uint ADDRESS_TAG_BIT = 0x80000000;

     alias int INT_PTR, LONG_PTR;
     alias int* PINT_PTR, PLONG_PTR;
     alias uint UINT_PTR, ULONG_PTR, HANDLE_PTR;
     alias uint* PUINT_PTR, PULONG_PTR;
     alias short HALF_PTR;
     alias short* PHALF_PTR;
     alias ushort UHALF_PTR;
     alias ushort* PUHALF_PTR;
}


alias ULONG_PTR SIZE_T, DWORD_PTR;
alias ULONG_PTR* PSIZE_T, PDWORD_PTR;
alias LONG_PTR SSIZE_T;
alias LONG_PTR* PSSIZE_T;



//ип = импортируемая переменная
extern(C)
{
alias extern int ипЦ;
alias extern uint ипбЦ;
alias extern double ипД2;
alias extern float ипП;
alias extern void ип;
alias extern void *ипУ;
alias extern byte ипБ;
alias extern ubyte ипбБ;
alias extern char ипС;
alias extern char *ипуС;
alias extern wchar ипШ;
alias extern wchar *ипуШ;
alias extern long ипД;
alias extern ulong ипбД;
}

alias extern(D) int function() функЦ;
alias extern(D) uint function() функбЦ;
alias extern(D) double function() функД2;
alias extern(D) float function() функП;
alias extern(D) void function() функ;
```

```
alias extern(D) void *function() функУ;
alias extern(D) byte function() функБ;
alias extern(D) ubyte function() функбБ;
alias extern(D) char function() функС;
alias extern(D) char *function() функуС;
alias extern(D) wchar function() функШ;
alias extern(D) wchar *function() функуШ;
alias extern(D) Object function() функО;
alias extern(D) long function() функД;
alias extern(D) ulong function() функбД;

alias extern (Windows) проц function(цел) винфункЦ;
alias extern (Windows) проц function(цел, цел) винфункЦЦ;
alias extern (Windows) проц function(цел, цел, цел) винфункЦЦЦ;
alias extern (Windows) проц function(цел, цел, цел, цел) винфункЦЦЦЦ;
alias extern (Windows) проц function(цел, цел, цел, цел, цел) винфункЦЦЦЦЦ;
alias extern (Windows) проц function(сим, цел, цел) винфункСЦЦ;
alias extern (Windows) проц function(ббайт, цел, цел) винфункбБЦЦ;


alias extern (Windows) проц function(бцел, цел, цел, цел) винфункбЦЦЦЦ;

alias extern(Windows) цел function() винфункЦ;
alias extern (Windows) цел function(сим, цел, цел) винфункЦ_СЦЦ;
alias extern (Windows) цел function(ббайт, цел, цел) винфункЦ_бБЦЦ;
alias extern (Windows) цел function(цел) винфункЦ_Ц;
alias extern (Windows) цел function(цел, цел) винфункЦ_ЦЦ;
alias extern (Windows) цел function(цел, цел, цел) винфункЦ_ЦЦЦ;
alias extern (Windows) цел function(цел, цел, цел, цел) винфункЦ_ЦЦЦЦ;
alias extern (Windows) цел function (ук, бцел, бцел, цел) винфункЦ_УбЦбЦЦ;

alias extern(Windows) бцел function() винфункбЦ;
alias extern(Windows) бцел function (ук, бцел, бцел, цел) винфункбЦ_УбЦбЦЦ;
alias extern (Windows) бцел function(ук) винфункбЦ_У;

alias extern(Windows) дво function() винфункД2;
alias extern(Windows) плав function() винфункП;
alias extern(Windows) проц function() винфунк;
alias extern(Windows) ук function() винфункУ;
alias extern(Windows) байт function() винфункБ;
alias extern(Windows) ббайт function() винфункбБ;
alias extern(Windows) сим function() винфункС;
alias extern(Windows) усим function() винфункуС;
alias extern(Windows) шим function() винфункШ;
alias extern(Windows) ушим function() винфункуШ;
alias extern(Windows) дол function() винфункД;
alias extern(Windows) бдол function() винфункбД;

alias extern(Windows) бул function() винфункБ2;
alias extern(Windows) бул function(бцел) винфункБ2_бЦ;

//alias extern (Windows) struct винструкт;
//alias extern (Windows) class винкласс;

alias винфункЦ_УбЦбЦЦ ОКОНПРОЦ;
alias винфункбЦ_УбЦбЦЦ ОТКРФЛХУКПРОЦ;
alias винфункБ2_бЦ ОБРАБПРОЦ;
alias винфункЦ УДПРОЦ;
alias УДПРОЦ ДЛГПРОЦ;
alias УДПРОЦ ТАЙМЕРПРОЦ;
alias УДПРОЦ СЕРСТРПРОЦ;
alias УДПРОЦ РИССТПРОЦ;
alias бцел СОКЕТ;
typedef СОКЕТ т_сокет = cast(СОКЕТ)~0;
```

```
alias цел т_длинсок;
alias бцел ЦВЕТ;
alias шим ОЛЕСИМ;
alias ОЛЕСИМ олес;
alias цел Бул;
alias бцел МАСКА_ДОСТУПА;
alias ук УкТОКЕН_ДОСТУПА;
alias ук УкБИД;
alias бул РЕЖИМ_ОТСЛЕЖИВАНИЯ_КОНТЕКСТА_БЕЗОПАСНОСТИ;
alias бкрат УПР_ДЕСКРИПТОРА_БЕЗОПАСНОСТИ;

alias проц function( ткст файл, т_мера строка, ткст сооб = пусто )
типПроверОбр;
alias ИнфоОтслежИскл function( ук укз = пусто ) типСледОбр;
alias проц delegate( ук, ук ) сканФн;
alias бул function() ТестерМодулей;
alias бул function(Объект) ОбработчикСборки;
alias проц delegate( Исключение ) ОбработчикИсключения;
alias extern(D) проц delegate() ddel;
alias extern(D) проц delegate(цел, цел) dint;
alias проц delegate() ПередВходом;
alias проц delegate() ПередВыходом;
alias проц delegate(Объект) ДСобыт, DEvent;

extern (D) typedef цел delegate(ук) т_дг, dg_t;
extern (D) typedef цел delegate(ук, ук) т_дг2, dg2_t;

alias проц delegate( ук, ук ) фнСканВсеНити;

/* Тип для возвратного значения динамических массивов.
*/
alias long т_дмВозврат;

struct СМСтат
     {
          т_мера размерПула; // общий размер пула
          т_мера испРазмер; // выделено байтов
          т_мера свобБлоки; // число блоков, помеченных FREE
          т_мера размСпискаСвобБлоков; // всего памяти в списках освобождения
          т_мера блокиСтр;
     }
     alias СМСтат GCStats;
     /**
* Содержит инфоагрегат о блоке управляемой памяти. Назначение этой
* структуры заключается в поддержке более эффективного стиля опроса
* в тех экземплярах, где требуется более подробная информация.
*
* основа = Указатель на основание опрашиваемого блока.
* размер = Размер блока, вычисляемый от основания.
* атр = Биты установленных на блоке памяти атрибутов.
*/
     struct ИнфОБл
     {
          ук основа;
          т_мера размер;
          бцел атр;
     }
     alias ИнфОБл BlkInfo;
     /**
* Элементы бит-поля, представляющего атрибуты блока памяти. Ими
* можно манипулировать функциями дайАтр, устАтр, удалиАтр.
*/

     struct Пространство
```

```
    {
        ук Низ;
        ук Верх;
    };

    struct Array
    {
        size_t length;
        byte *data;
        ук ptr;
        alias length длина;
        alias data данные;
        alias ptr укз;
    }


struct Complex
{
реал re;
реал im;
}

    struct aaA
    {
        //aaA *left;
        //aaA *right;
        //hash_t hash;
    aaA *next;
hash_t hash;
        /* key */
        /* value */
    }

    struct BB
    {
        aaA*[] b;
        size_t nodes; // общее число узлов aaA
        TypeInfo keyti; // TODO: заменить на TypeInfo_AssociativeArray, если
        доступно через _aaGet()
        aaA*[4] binit; // начальное значение c[]
    }

    /* Это тип Ассоциативный Массив, который действительно виден
    программисту,
    * хотя он и правда, уплотнён.
    */

    struct AA
    {
        BB* a;
    }
    /+class Amac
    {
    private AA amac;
    }+/

struct D_CRITICAL_SECTION
{
D_CRITICAL_SECTION *next;
//CRITICAL_SECTION cs;
}

alias проц (*ФИНАЛИЗАТОР_СМ)(ук p, бул dummy);
```

```
//Функции, экспортируемые рантаймом
extern(C)
{

цел printf(усим, ...);
     alias printf эхо;
     проц _d_monitor_create(Object);
     проц _d_monitor_destroy(Object);
     проц _d_monitor_lock(Object);
     int _d_monitor_unlock(Object);
     //asm
     проц _minit();

//exception
     проц onAssertError( ткст file, т_мера line );
     проц onAssertErrorMsg( ткст file, т_мера line, ткст msg );
     проц onArrayBoundsError( ткст file, т_мера line );
     проц onFinalizeError( ClassInfo info, Исключение ex );
     проц onOutOfMemoryError();
     проц onSwitchError( ткст file, т_мера line );
     проц onUnicodeError( ткст msg, т_мера idx );
     проц onRangeError( string file, т_мера line );
     проц onHiddenFuncError( Object o );
     проц _d_assert( ткст file, uint line );
     static проц _d_assert_msg( ткст msg, ткст file, uint line );
     проц _d_array_bounds( ткст file, uint line );
     проц _d_switch_error( ткст file, uint line );
     проц _d_OutOfMemory();
     //ИнфоОтслежИскл контекстТрассировки( ук ptr = null );
     //бул устСледОбр( типСледОбр h );
     бул устПроверОбр( типПроверОбр h );
     }
     extern (C)
{
     //complex.c
     Complex _complex_div(Complex x, Complex y);
     Complex _complex_mul(Complex x, Complex y);
     // long double _complex_abs(Complex z);
     Complex _complex_sqrt(Complex z);

//critical.c
     проц _d_criticalenter(D_CRITICAL_SECTION *dcs);
     проц _d_criticalexit(D_CRITICAL_SECTION *dcs);
     проц _STI_critical_init();
     проц _STD_critical_term();
     //rt.adi
     char[] _adReverseChar(char[] a);
     шткст _adReverseWchar(шткст a);
     проц[] _adReverse(Array a, size_t szelem);
     char[] _adSortChar(char[] a);
     шткст _adSortWchar(шткст a);
     int _adEq(Array a1, Array a2, TypeInfo ti);
     int _adEq2(Array a1, Array a2, TypeInfo ti);
     int _adCmp(Array a1, Array a2, TypeInfo ti);
     int _adCmp2(Array a1, Array a2, TypeInfo ti);
     int _adCmpChar(Array a1, Array a2);
     //rt.aaA
     size_t _aaLen(AA aa);
     ук _aaGet(AA* aa, TypeInfo keyti, size_t valuesize, ...);
     ук _aaGetRvalue(AA aa, TypeInfo keyti, size_t valuesize, ...);
     ук _aaIn(AA aa, TypeInfo keyti, ...);
     проц _aaDel(AA aa, TypeInfo keyti, ...);
     т_дмВозврат _aaValues(AA aa, size_t keysize, size_t valuesize);
```

```
    ук _aaRehash(АА* раа, TypeInfo keyti);
    проц _aaBalance(АА* раа);
    т_дмВозврат _aaKeys(АА аа, size_t keysize);
    int _aaApply(АА аа, size_t keysize, т_дг дг);
    int _aaApply2(АА аа, size_t keysize, т_дг2 дг);
    ВВ* _d_assocarrayliteralT(TypeInfo_AssociativeArray ti, size_t length,
    ...);
    int _aaEqual(TypeInfo_AssociativeArray ti, АА e1, АА e2);
    //rt.aApply
    int _aApplycd1(char[] аа, т_дг дг);
    int _aApplywd1(шткст аа, т_дг дг);
    int _aApplycw1(char[] аа, т_дг дг);
    int _aApplywc1(шткст аа, т_дг дг);
    int _aApplydc1(юткст аа, т_дг дг);
    int _aApplydw1(юткст аа, т_дг дг);
    int _aApplycd2(char[] аа, т_дг2 дг);
    int _aApplywd2(шткст аа, т_дг2 дг);
    int _aApplycw2(char[] аа, т_дг2 дг);
    int _aApplywc2(шткст аа, т_дг2 дг);
    int _aApplydc2(юткст аа, т_дг2 дг);
    int _aApplydw2(юткст аа, т_дг2 дг);
    //rt.aApplyR
    int _aApplyRcd1(in char[] аа, т_дг дг);
    int _aApplyRwd1(in шткст аа, т_дг дг);
    int _aApplyRcw1(in char[] аа, т_дг дг);
    int _aApplyRwc1(in шткст аа, т_дг дг);
    int _aApplyRdc1(in юткст аа, т_дг дг);
    int _aApplyRdw1(in юткст аа, т_дг дг);
    int _aApplyRcd2(in char[] аа, т_дг2 дг);
    int _aApplyRwd2(in шткст аа, т_дг2 дг);
    int _aApplyRcw2(in char[] аа, т_дг2 дг);
    int _aApplyRwc2(in шткст аа, т_дг2 дг);
    int _aApplyRdc2(in юткст аа, т_дг2 дг);
    int _aApplyRdw2(in юткст аа, т_дг2 дг);
    //rt.alloca
    ук __alloca(int nbytes);
    //rt.arraycast
    проц[] _d_arraycast(size_t tsize, size_t fsize, проц[] a);
    проц[] _d_arraycast_frombit(uint tsize, проц[] a);

//rt.arraycat
    byte[] _d_arraycopy(size_t size, byte[] from, byte[] to);
    //rt.cast
    Object _d_toObject(ук p);
    Object _d_interface_cast(ук p, ClassInfo c);
    Object _d_dynamic_cast(Object o, ClassInfo c);
    int _d_isbaseof2(ClassInfo oc, ClassInfo c, ref size_t offset);
    int _d_isbaseof(ClassInfo oc, ClassInfo c);
    ук _d_interface_vtbl(ClassInfo ic, Object o);
    //rt.lifetime
    Object _d_newclass(ClassInfo ci);
    проц _d_delinterface(ук p);
    проц _d_delclass(Object *p);
    ulong _d_newarrayT(TypeInfo ti, size_t length);
    ulong _d_newarrayiT(TypeInfo ti, size_t length);
    ulong _d_newarraymT(TypeInfo ti, int ndims, ...);
    ulong _d_newarraymiT(TypeInfo ti, int ndims, ...);
    ук _d_allocmemory(size_t nbytes);
    проц _d_delarray(Array *p);
    проц _d_delmemory(проц *p);
    проц _d_callfinalizer(ук p);
    проц ртФинализуй(ук p, бул det = да);
    extern (C) проц rt_finalize_gc(ук p);
    byte[] _d_arraysetlengthT(TypeInfo ti, size_t newlength, Array *p);
```

```
        byte[] _d_arraysetlengthiT(TypeInfo ti, size_t newlength, Array *p);
        long _d_arrayappendT(TypeInfo ti, Array *px, byte[] y);
        byte[] _d_arrayappendcT(TypeInfo ti, inout byte[] x, ...);
        byte[] _d_arraycatT(TypeInfo ti, byte[] x, byte[] y);
        byte[] _d_arraycatnT(TypeInfo ti, uint n, ...);
        ук _d_arrayliteralT(TypeInfo ti, size_t length, ...);
        long _adDupT(TypeInfo ti, Array a);
            //rt.hash
        hash_t rt_hash_str(ук bStart,size_t длина, hash_t seed=cast(hash_t)0);
        hash_t rt_hash_block(size_t *bStart,size_t длина, hash_t
        seed=cast(hash_t)0);
        uint rt_hash_utf8(char[] str, uint seed=0);
        uint rt_hash_utf16(шткст str, uint seed=0);
        uint rt_hash_utf32(юткст str, uint seed=0);
        hash_t rt_hash_combine( hash_t val1, hash_t val2 );
        uint rt_hash_str_neutral32(ук bStart,uint длина, uint seed=0);
        ulong rt_hash_str_neutral64(ук bStart,ulong длина, ulong seed=0);
        uint rt_hash_combine32( uint знач, uint seed );
        ulong rt_hash_combine64( ulong value, ulong level);
        //rt.qsort
        long _adSort(Array a, TypeInfo ti);
        //rt.memset
        short *_memset16(short *p, short value, size_t count);
        int *_memset32(int *p, int value, size_t count);
        long *_memset64(long *p, long value, size_t count);
        cdouble *_memset128(cdouble *p, cdouble value, size_t count);
        реал *_memset80(реал *p, реал value, size_t count);
        creal *_memset160(creal *p, creal value, size_t count);
        ук _memsetn(ук p, ук value, int count, size_t sizelem);

    //rt.switch
        int _d_switch_string(char[][] table, char[] ca);
        int _d_switch_ustring(шим[][] table, шткст ca);
        int _d_switch_dstring(дим[][] table, юткст ca);


    //object
        проц _d_monitorrelease(Object h);
        проц _d_notify_release(Object o);
        проц _moduleCtor();
        проц _moduleCtor2(ModuleInfo[] mi, int skip);
        проц _moduleDtor();
        проц _moduleUnitTests();
        проц _moduleIndependentCtors();

        проц создайМонитор(Объект o);
        проц разрушьМонитор(Объект o) ;
        проц блокируйМонитор(Объект o) ;
        цел разблокируйМонитор(Объект o) ;
        проц _d_monitordelete(Object h, бул det);
        проц удалиМонитор(Объект o, бул уд);
        проц _d_monitorenter(Object h);
        проц войдиВМонитор(Объект o);
        проц _d_monitorexit(Object h);
        проц выйдиИзМонитора(Объект o);
        проц _d_monitor_devt(Monitor* m, Object h);
        проц событиеМонитора(Монитор* м, Объект o);
        проц rt_attachDisposeEvent(Object h, ДСобыт e);
        проц rt_detachDisposeEvent(Object h, ДСобыт e);
        int _fatexit(ук);
        //runtime
        сим[][] ртПолучиАрги(цел аргчло, сим **аргткст);
        бул рт_вЗадержке();
        бул ртПущен();
```

```
    бул ртОстановлен();
    бул ртСтарт(ПередВходом передвхо = пусто, ОбработчикИсключения дг =
    пусто);
    проц ртСтоп();
    проц ртСоздайОбработчикСледа( Следопыт h );
    Исключение.ИнфОСледе ртСоздайКонтекстСледа( ук ptr );
    проц ртУстановиОбработчикСборки(ОбработчикСборки h);
    ук ртНизСтэка();
    ук ртВерхСтэка();
    проц ртСканируйСтатДан( сканФн scan );
    проц _d_callinterfacefinalizer(ук p);
    size_t gc_newCapacity(size_t newlength, size_t size);
    ткст _d_arrayappendcd(inout ткст x, дим c);
    шткст _d_arrayappendwd(inout шткст x, дим c);
    //проц устСовместнПам( убайт буф);
    //убайт получиСовместнПам();
    //thread

    проц thread_init();
    проц thread_attachThis();
    проц thread_detachThis();
    проц thread_joinAll();

    бул thread_needLock();
    проц thread_suspendAll();
    проц thread_resumeAll();
    проц thread_scanAll( фнСканВсеНити scan, ук текВерхСтека = null );
    проц thread_yield();
    проц thread_sleep(double период);
    проц fiber_entryPoint();
    проц fiber_switchContext( ук* oldp, ук newp );
    //gc
    бул смПроверь(ук p);
бул смУменьши();
бул смДобавьКорень( ук p );
бул смДобавьПространство( ук p, т_мера разм );
бул смДобавьПространство2( ук p, ук разм );
бул смУдалиКорень( ук p );
бул смУдалиПространство( ук p );
т_мера смЁмкость(ук p);
бул смМонитор(ddel начало, dint конец );
бул смСтат();
СМСтат смДайСтат();
проц[] смПразместиМас(т_мера члобайт);
проц[] смПереместиМас(ук p, т_мера члобайт);
бул устИнфОТипе(ИнфОТипе иот, ук p);
ук дайУкНаСМ();
бул укНаСМ(ук p);
бул сбросьУкНаСМ();
бцел смДайАтр( ук p );
бцел смУстАтр( ук p, ПАтрБлока a );
бцел смУдалиАтр( ук p, ПАтрБлока a );
ук смПразмести( т_мера разм, бцел ba = 0 );
ук смКразмести( т_мера разм, бцел ba = 0 );
ук смПеремести( ук p, т_мера разм, бцел ba = 0 );
т_мера смРасширь( ук p, т_мера mx, т_мера разм );
т_мера смРезервируй( т_мера разм );
бул смОсвободи( ук p );
ук смАдрес( ук p );
т_мера смРазмер( ук p );
ук смСоздайСлабУк( Объект к );
бул смУдалиСлабУк( ук wp );
Объект смДайСлабУк( ук wp );
ИнфОБл смОпроси( ук p );
```

```
бул смВключи();
бул смОтключи();
бул смСобери();


проц setTypeInfo(TypeInfo ti, ук р);
ук getGCHandle();
проц setGCHandle(ук р);
проц endGCHandle();

проц gc_init();
проц gc_term();
size_t gc_capacity(ук р);
проц gc_minimize();
проц gc_addRoot( ук р );
проц gc_addRange( ук р, size_t разм );
проц gc_removeRoot( ук р );
проц gc_removeRange( ук р );
проц gc_monitor(ddel begin, dint end );
GCStats gc_stats();
проц _d_gc_addrange(проц *pbot, проц *ptop);
проц _d_gc_removerange(проц *pbot);
uint gc_getAttr( ук р );
uint gc_setAttr( ук р, uint а );
uint gc_clrAttr( ук р, uint а );
ук gc_malloc( size_t разм, uint ba = 0 );
ук gc_calloc( size_t разм, uint ba = 0 );
ук gc_realloc( ук р, size_t разм, uint ba = 0 );
size_t gc_extend( ук р, size_t мх, size_t разм );
size_t gc_reserve( size_t разм );
проц gc_free( ук р );
ук gc_addrOf( ук р );
size_t gc_sizeOf( ук р );
ук gc_weakpointerCreate( Object к );
проц gc_weakpointerDestroy( ук wp );
Object gc_weakpointerGet( ук wp );
BlkInfo gc_query( ук р );
проц gc_enable();
проц gc_disable();
проц gc_collect();
проц gc_check(проц *р);
проц gc_addRangeOld( ук р, ук разм );

ткст[] дайАргиКС();
}

extern (Windows) ук ДайДескрТекущейНити();
//extern (Windows) проц d_throw(Object *o);

extern (C) //Возвращает массив определённого типа с заданным количеством
элементов
{
ткст симмас(цел к);
байт[] байтмас(цел к);
ббайт[] ббайтмас(цел к);
плав[] плавмас(цел к);
дво[] двомас(цел к);
ткст[] ткстмас(цел к);//выдаёт ошибку; причина неясна
бдол[] бдолмас(цел к);
дол[] долмас(цел к);
цел[] целмас(цел к);
бцел[] бцелмас(цел к);
крат[] кратмас(цел к);
бкрат[] бкратмас(цел к);
```

```
    проц ошибка(ткст сооб, ткст файл = пусто , т_мера строка = 0 );
    проц ошибкаПодтверждения(ткст файл, т_мера строка);
    проц ошибкаГраницМассива(ткст файл, т_мера строка);
    проц ошибкаФинализации(ИнфОКлассе инфо, Исключение ис);
    проц ошибкаНехваткиПамяти();
    проц ошибкаПереключателя(ткст файл, т_мера строка);
    проц ошибкаЮникод(ткст сооб, т_мера индкс);
    проц ошибкаДиапазона(ткст файл, т_мера строка);
    проц ошибкаСкрытойФункции(Объект о);
    }
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
17:27

```
module cidrus;
//сиэф = экспортируемая си-функция

alias проц function(ук) _У;

extern(C)
{
alias проц function() сифунк;
alias проц function(цел) сифунк_Ц;
alias проц function(цел, цел) сифунк_ЦЦ;
alias проц function(цел, цел, цел) сифунк_ЦЦЦ;
alias проц function(цел, цел, цел, цел) сифунк_ЦЦЦЦ;
alias проц function(цел, цел, цел, цел, цел) сифунк_ЦЦЦЦЦ;
alias проц function(сим, цел, цел) сифунк_СЦЦ;
alias проц function(ббайт, цел, цел) сифунк_бБЦЦ;
alias проц function(ук) сифунк_У;
alias проц function(бцел, цел, цел, цел) сифунк_бЦЦЦЦ;

alias цел function() сифункЦ;
alias цел function(сим, цел, цел) сифункЦ_СЦЦ;
alias цел function(ббайт, цел, цел) сифункЦ_бБЦЦ;
alias цел function(цел) сифункЦ_Ц;
alias цел function(цел, цел) сифункЦ_ЦЦ;
alias цел function(цел, цел, цел) сифункЦ_ЦЦЦ;
alias цел function(цел, цел, цел, цел) сифункЦ_ЦЦЦЦ;
alias цел function (ук, бцел, бцел, цел) сифункЦ_УбЦбЦЦ;

alias бцел function() сифункбЦ;
alias бцел function (ук, бцел, бцел, цел) сифункбЦ_УбЦбЦЦ;
alias бцел function(ук) сифункбЦ_У;

alias дво function() сифункД2;
alias плав function() сифункП;
alias ук function() сифункУ;
alias байт function() сифункБ;
alias ббайт function() сифункбБ;
alias сим function() сифункС;
alias усим function() сифункуС;
alias шим function() сифункШ;
alias ушим function() сифункуШ;
alias дол function() сифункД;
alias бдол function() сифункбД;
```

```
alias бул function() сифункБ2;
alias бул function(бцел) сифункБ2_бЦ;
//alias extern (C) struct систрукт;
//alias extern (C) class сикласс;
}

const int _ЧЛОФ = 60;
extern (C) struct _iobuf
{
align (1):
char* _ptr;
int _cnt;
char* _base;
int _flag;
int _file;
int _charbuf;
int _bufsiz;
int __tmpnum;
alias _ptr Ук ;
alias _cnt Конт;
alias _base Ова ;
alias _flag Флаг ;
alias _file Файл ;
alias _charbuf Симбуф;
alias _bufsiz Буфразм ;
alias __tmpnum Времчло ;
    }
alias _iobuf ФАЙЛ, FILE;
alias ФАЙЛ *фук;
extern (C) extern ФАЙЛ[_ЧЛОФ] _iob;
extern (C) фук дайСтдвхо();
extern (C) фук дайСтдвых();
extern (C) фук дайСтдош();
extern (C) фук дайСтддоп();
extern (C) фук дайСтдпрн();

const фук стдвхо = &_iob[0];
const фук стдвых = &_iob[1];
const фук стдош = &_iob[2];
const фук стддоп = &_iob[3];
const фук стдпрн = &_iob[4];

    alias стдвхо stdin;
    alias стдвых stdout;
    alias стдош stderr;
    alias стддоп stdaux;
    alias стдпрн stdprn;


const string _P_tmpdir = "\\"; // non-standard
const wstring _wP_tmpdir = "\\"; // non-standard
const int L_tmpnam = _P_tmpdir.length + 12;
    extern (C) extern ubyte __fhnd_info[_ЧЛОФ];

    //Режимы открытия файла функцией откройфл:
const
{
ткст Ч = "r";
ткст З = "w";
ткст Д = "a";
ткст ЧП = "r+";
ткст ЗП = "w+";
ткст ДП = "a+";
```

```
}

extern (C)
{

struct лпреобр
{

ткст десятичная_точка;
ткст делит_тысяч;
ткст группировка;
ткст цел_валютн_символ;
ткст символ_валюты;
ткст мон_десятичная_точка;
ткст мон_делит_тыс;
ткст мон_группировка;
ткст положит_знак;
ткст отрицат_знак;
байт цел_дробн_цифры;
байт дробн_цифры;
байт p_cs_precedes;
байт p_sep_by_space;
байт n_cs_precedes;
байт n_sep_by_space;
байт p_sign_posn;
байт n_sign_posn;
байт int_p_cs_precedes;
байт int_p_sep_by_space;
байт int_n_cs_precedes;
байт int_n_sep_by_space;
байт int_p_sign_posn;
байт int_n_sign_posn;
}

struct т_цмаксдел
     {
          дол квот,
                         остат;
     }

struct т_фсред
{
     бкрат статус;
бкрат контроль;
бкрат округл;
бкрат[2] резерв;
}
     struct т_дели
     {
          цел квот,
               остат;
     }


struct т_делиц
     {
          цел квот,
               остат;
     }

struct т_делид
     {
          дол квот,
               остат;
```

```
    }
    }

enum
{
    ФУК_ДОБАВКА    = 0x04,
    ФУК_УСТРВО    = 0x08,
    ФУК_ТЕКСТ    = 0x10,
    ФУК_БАЙТ    = 0x20,
    ФУК_ШИМ    = 0x40,
ВВФБФ = 0,
ВВЛБФ = 0x40,
    ВВНБФ = 4,
ВВЧИТ = 1,      // non-standard
ВВЗАП = 2,      // non-standard
ВВМОЙБУФ = 8,      // non-standard
ВВКФ = 0x10, // non-standard
ВВОШ = 0x20, // non-standard
ВВСТРЖ = 0x40, // non-standard
ВВЧЗ = 0x80, // non-standard
ВВТРАН = 0x100, // non-standard
ВВПРИЛ = 0x200, // non-standard
}

enum
{
РАЗМБУФ = 0x4000,
КФ = -1,//конец файла
МАКС_ОТКРФ = 20,
МАКС_ИМЯФ = 256, // 255 plus NULL
МАКС_ВРЕМ = 32767,
СИС_ОТКР = 20,    // non-standard
}

const шим ШКФ = 0xFFFF;

const дво ДВОБЕСК = дво.infinity;
const дво ПЛАВБЕСК = плав.infinity;
const дво РЕАЛБЕСК = реал.infinity;

const СИМБИТ = 8;
const БАЙТМИН = байт.min;
const БАЙТМАКС = байт.max;
const ББАЙТМИН = ббайт.min;
const СИММИН = сим.max;
const СИММАКС = сим.max;
const МБДЛИНМАКС = 2;
const КРАТМИН = крат.min;
const КРАТМАКС = крат.max;
const БКРАТМАКС = бкрат.max;
const ЦЕЛМИН = цел.min;
const ЦЕЛМАКС = цел.max;
const БЦЕЛМАКС = бцел.max;
const ДОЛМИН = дол.min;
const ДОЛМАКС = дол.max;
const БДОЛМАКС = бдол.max;

const ПЛАВОКРУГЛ            = 1;
const ПЛАВМЕТОДОЦЕНКИ    = 2;
const ПЛАВКОРЕНЬ            = 2;

const ПЛАВЦИФР            = плав.dig;
const ДВОЦИФР            = дво.dig;
const РЕАЛЦИФР            = реал.dig;
```

```
const ПЛАВМАНТЦИФР        = плав.mant_dig;
const ДВОМАНТЦИФР        = дво.mant_dig;
const РЕАЛМАНТЦИФР        = реал.mant_dig;

const ПЛАВМИН            = плав.min;
const ДВОМИН            = дво.min;
const РЕАЛМИН            = реал.min;

const ПЛАВМАКС            = плав.max;
const ДВОМАКС            = дво.max;
const РЕАЛМАКС            = реал.max;

const ПЛАВЭПС        = плав.epsilon;
const ДВОЭПС        = дво.epsilon;
const РЕАЛЭПС        = реал.epsilon;

const ПЛАВМИНЭКСП        = плав.min_exp;
const ДВОМИНЭКСП        = дво.min_exp;
const РЕАЛМИНЭКСП        = реал.min_exp;

const ПЛАВМАКСЭКСП        = плав.max_exp;
const ДВОМАКСЭКСП        = дво.max_exp;
const РЕАЛМАКСЭКСП        = реал.max_exp;

const ПЛАВМИН10ЭКСП        = плав.min_10_exp;
const ДВОМИН10ЭКСП        = дво.min_10_exp;
const РЕАЛМИН10ЭКСП    = реал.min_10_exp;

const ПЛАВМАКС10ЭКСП        = плав.max_10_exp;
const ДВОМАКС10ЭКСП        = дво.max_10_exp;
const РЕАЛМАКС10ЭКСП    = реал.max_10_exp;

const плав БЕСКОНЕЧНОСТЬ = плав.infinity;
const плав Н_Ч = плав.nan;

const цел ПЗ_ИЛОГБ0 = цел.min;
const цел ПЗ_ИЛОГБНЧ = цел.min;

const цел МАТОШ = 1;//математическая ошибка
const цел МАТОШИСКЛ = 2;
const цел матошобработка = МАТОШ | МАТОШИСКЛ;

const ЛП_СИТИП = 0;
const ЛП_ЧИСЛО = 1;
const ЛП_ВРЕМЯ = 2;
const ЛП_КОЛЛЕЙТ = 3;
const ЛП_МОНЕТА = 4;
const ЛП_ВСЕ = 6;
const ЛП_БУМАГА = 7; // non-standard
const ЛП_ИМЯ = 8; // non-standard
const ЛП_АДРЕС = 9; // non-standard
const ЛП_ТЕЛЕФОН = 10; // non-standard
const ЛП_МЕРА = 11; // non-standard
const ЛП_ИДЕНТИФИКАЦИЯ = 12; // non-standard

enum ППозКурсора {
Уст,
Тек,
Кон,
}

enum ФИскл
{
```

```
Повреждён = 1,
Ненорм = 2, // non-standard
ДелениеНаНоль = 4,
Переполнение = 8,
Недополнение = 0x10,
Неточность = 0x20,
ВсеИскл = 0x3F,
КБлиж = 0,
Выше = 0x800,
Ниже = 0x400,
КНулю = 0xC00,
}

version( Windows )
{
extern т_фсред _FE_DFL_ENV;
}



version( Windows )//errno
{
const ОШНЕДОП = 1; // Недопустимая операция
const ОШНЕСУЩ = 2; // Файл или каталог не найден
const ОШПОИСК = 3; // Процес не найден
const ОШПРЕРВ = 4; // Прерванный системный вызов
const ОШВВ = 5; // Ошибка ввода-вывода
const ОШНЕТУСТРВА = 6; //Устройство или адрес не найдены
const ОШАРГСЛИШБ = 7; // Слишком много аргументов
const ОШНЕИСП = 8; // Exec format error
const ОШНЕВФАЙЛ = 9; // Неправильный номер файла
const ОШНЕТОТПРЫСК = 10; // Отсутствуют дочерние процессы
const ОШПОВТОР = 11; // Попробовать снова
const ОШНЕТПАМ = 12; // Вне памяти
const ОШДОСТУП = 13; // Доступ запрещён
const ОШНЕТОТАДР = 14; // Неправильный адрес
const ОШЗАНЯТ = 16; // Устройство или ресурс заняты
const ОШФЕСТЬ = 17; // Файл уже есть
const ОШКРОССУСТРССЫЛ = 18; // Ссылка на другое устройство
const ОШНЕТУСТР = 19; // Устройство не обнаружено
const ОШНЕПАП = 20; // Это не папка
const ОШПАП = 21; // Это папка
const ОШИНВАЛАРГ = 22; // Неверный аргумент
const ОШПЕРЕПФТ = 23; // Переполнение файловой таблицы
const ОШМНОГОТКРФ = 24; // Слишком много открытых файлов
const ОШНЕТП = 25; // Not a typewriter
const ОШФСЛИШБ = 27; // Файл слишком громоздкий
const ОШНЕТМЕСТ = 28; // На устойстве закончилось свободное пространство
const ОШПЕРЕМЕСТ = 29; // Недопустимое перемещение
const ОШФСТЧ = 30; // Файловая система только для чтения
const ОШСЛИШМССЫЛ = 31; // Слишком много ссылок
const ОШПАЙП = 32; // Broken pipe
const ОШДОМ = 33; // Математический аргумент вне домена или функции
const ОШДИАП = 34; // Математический результат невозможно представить
const ОШДЕДЛОК = 36; // Resource deadlock would occur
const ОШСЛИШБФИМЯ = 38; // Слишком длинное название файла
const ОШНЕТЗАМЗАП = 39; // No record locks available
const ОШФУНКНЕРЕАЛИЗ = 40; // Функция не реализована
const ОШПАПНЕПУСТ = 41; // Папка не пуста
const ОШБАЙТПОСЛ = 42; // Недопустимая байтовая
    const EPERM = 1; // Operation not permitted
const ENOENT = 2; // No such файл or directory
const ESRCH = 3; // No such process
const EINTR = 4; // Interrupted system вызов
```

```
const EIO = 5; // I/O ошибка
const ENXIO = 6; // No such устройство or адрес
const E2BIG = 7; // Аргумент список too дол
const ENOEXEC = 8; // Exec форматируй ошибка
const EBADF = 9; // Bad файл number
const ECHILD = 10; // No ветвь processes
const EAGAIN = 11; // Try again
const ENOMEM = 12; // Out of память
const EACCES = 13; // Permission denied
const EFAULT = 14; // Bad адрес
const EBUSY = 16; // Устройство or resource busy
const EEXIST = 17; // Файл есть_ли
const EXDEV = 18; // Cross-устройство link
const ENODEV = 19; // No such устройство
const ENOTDIR = 20; // Not a directory
const EISDIR = 21; // Is a directory
const EINVAL = 22; // Invalопр аргумент
const ENFILE = 23; // Файл table перебор
const EMFILE = 24; // Too many открой файлы
const ENOTTY = 25; // Not a typewriter
const EFBIG = 27; // Файл too large
const ENOSPC = 28; // No пространство left on устройство
const ESPIPE = 29; // Illegal сместись
const EROFS = 30; // Чтен-only файл system
const EMLINK = 31; // Too many линки
const EPIPE = 32; // Broken pipe
const EDOM = 33; // Math аргумент out of domain of func
const ERANGE = 34; // Math результат not representable
const EDEADLK = 36; // Resource deadlock would occur
const ENAMETOOLONG = 38; // Файл имя too дол
const ENOLCK = 39; // No record locks available
const ENOSYS = 40; // Function not implemented
const ENOTEMPTY = 41; // Directory not пустой
const EILSEQ = 42; // Illegal байт sequence
const EDEADLOCK = EDEADLK;

}

private alias проц function(цел) т_сигфн, sigfn_t;

const СИГОШ = cast(т_сигфн) -1;
const СИГДФЛ = cast(т_сигфн) 0;
const СИГИГН = cast(т_сигфн) 1;

// стандартные сигналы Си
const СИГАБОРТ = 22; // Ненормальное закрытие программы
const СИГОПЗ = 8; // Ошибка с плавающей запятой
const СИГИЛЛ = 4; // Недопустимая инструкция оборудования/-ю
const СИГПРЕРВ = 2; // Terminal interrupt character
const СИГСЕГВ = 11; // Invalid memory reference
const СИГТЕРМ = 15; // Termination


const УДАЧНЫЙ_ВЫХОД = 0;
const НЕУДАЧНЫЙ_ВЫХОД = 1;
const СЛУЧ_МАКС = 32767;
const МБ_ТЕК_МАКС = 1;

private
{
template типируй(T)
{
T типируй( T val ) { return val; }
}
```

```
}

const int8_t ЦЕЛ8_МИН = int8_t.min;
const int8_t ЦЕЛ8_МАКС = int8_t.max;
const int16_t ЦЕЛ16_МИН = int16_t.min;
const int16_t ЦЕЛ16_МАКС = int16_t.max;
const int32_t ЦЕЛ32_МИН = int32_t.min;
const int32_t ЦЕЛ32_МАКС = int32_t.max;
const int64_t ЦЕЛ64_МИН = int64_t.min;
const int64_t ЦЕЛ64_МАКС = int64_t.max;

const uint8_t БЦЕЛ8_МАКС = uint8_t.max;
const uint16_t БЦЕЛ16_МАКС = uint16_t.max;
const uint32_t БЦЕЛ32_МАКС = uint32_t.max;
const uint64_t БЦЕЛ64_МАКС = uint64_t.max;

const int_least8_t ЦЕЛ_МЕН8_МИН = int_least8_t.min;
const int_least8_t ЦЕЛ_МЕН8_МАКС = int_least8_t.max;
const int_least16_t ЦЕЛ_МЕН16_МИН = int_least16_t.min;
const int_least16_t ЦЕЛ_МЕН16_МАКС = int_least16_t.max;
const int_least32_t ЦЕЛ_МЕН32_МИН = int_least32_t.min;
const int_least32_t ЦЕЛ_МЕН32_МАКС = int_least32_t.max;
const int_least64_t ЦЕЛ_МЕН64_МИН = int_least64_t.min;
const int_least64_t ЦЕЛ_МЕН64_МАКС = int_least64_t.max;

const uint_least8_t БЦЕЛ_МЕН8_МАКС = uint_least8_t.max;
const uint_least16_t БЦЕЛ_МЕН16_МАКС = uint_least16_t.max;
const uint_least32_t БЦЕЛ_МЕН32_МАКС = uint_least32_t.max;
const uint_least64_t БЦЕЛ_МЕН64_МАКС = uint_least64_t.max;

const int_fast8_t ЦЕЛ_БЫСТР8_МИН = int_fast8_t.min;
const int_fast8_t ЦЕЛ_БЫСТР8_МАКС = int_fast8_t.max;
const int_fast16_t ЦЕЛ_БЫСТР16_МИН = int_fast16_t.min;
const int_fast16_t ЦЕЛ_БЫСТР16_МАКС = int_fast16_t.max;
const int_fast32_t ЦЕЛ_БЫСТР32_МИН = int_fast32_t.min;
const int_fast32_t ЦЕЛ_БЫСТР32_МАКС = int_fast32_t.max;
const int_fast64_t ЦЕЛ_БЫСТР64_МИН = int_fast64_t.min;
const int_fast64_t ЦЕЛ_БЫСТР64_МАКС = int_fast64_t.max;

const uint_fast8_t БЦЕЛ_БЫСТР8_МАКС = uint_fast8_t.max;
const uint_fast16_t БЦЕЛ_БЫСТР16_МАКС = uint_fast16_t.max;
const uint_fast32_t БЦЕЛ_БЫСТР32_МАКС = uint_fast32_t.max;
const uint_fast64_t БЦЕЛ_БЫСТР64_МАКС = uint_fast64_t.max;

const intptr_t ЦЕЛУК_МИН = intptr_t.min;
const intptr_t ЦЕЛУК_МАКС = intptr_t.max;

const uintptr_t БЦЕЛУК_МИН = uintptr_t.min;
const uintptr_t БЦЕЛУК_МАКС = uintptr_t.max;

const intmax_t ЦЕЛМАКС_МИН = intmax_t.min;
const intmax_t ЦЕЛМАКС_МАКС = intmax_t.max;

const uintmax_t БЦЕЛМАКС_МАКС = uintmax_t.max;

const ptrdiff_t ДЕЛЬТАУК_МИН = ptrdiff_t.min;
const ptrdiff_t ДЕЛЬТАУК_МАКС = ptrdiff_t.max;

const sig_atomic_t СИГАТОМ_МИН = sig_atomic_t.min;
const sig_atomic_t СИГАТОМ_МАКС = sig_atomic_t.max;

const size_t МЕРА_МАКС = size_t.max;

const wchar_t ШИМ_МИН = wchar_t.min;
```

```
const wchar_t ШИМ_МАКС = wchar_t.max;

const wint_t ВИНТ_МИН = wint_t.min;
const wint_t ВИНТ_МАКС = wint_t.max;

alias типируй!(int8_t) ЦЕЛ8_С;
alias типируй!(int16_t) ЦЕЛ16_С;
alias типируй!(int32_t) ЦЕЛ32_С;
alias типируй!(int64_t) ЦЕЛ64_С;

alias типируй!(uint8_t) БЦЕЛ8_С;
alias типируй!(uint16_t) БЦЕЛ16_С;
alias типируй!(uint32_t) БЦЕЛ32_С;
alias типируй!(uint64_t) БЦЕЛ64_С;

alias типируй!(intmax_t) ЦЕЛМАКС_С;
alias типируй!(uintmax_t) БЦЕЛМАКС_С;

extern(D)
{
//цел fpclassify(реал-floating x);
цел птклассифицируй(плав x) ;
цел птклассифицируй(дво x) ;
цел птклассифицируй(реал x);

//цел isfinite(реал-floating x);
цел конечен_ли(плав x) ;
цел конечен_ли(дво x) ;
цел конечен_ли(реал x) ;

//цел isinf(реал-floating x);
цел беск_ли(плав x) ;
цел беск_ли(дво x) ;
цел беск_ли(реал x) ;

//цел isnan(реал-floating x);
цел нечисло_ли(плав x) ;
цел нечисло_ли(дво x) ;
цел нечисло_ли(реал x) ;

//цел isnormal(реал-floating x);
цел нормаль_ли(плав x) ;
цел нормаль_ли(дво x) ;
цел нормаль_ли(реал x) ;

//цел signbit(реал-floating x);
цел знакбит(плав x) ;
цел знакбит(дво x) ;
цел знакбит(реал x);
     //цел isgreater(реал-floating x, реал-floating y);
цел больше_ли(плав x, плав y) ;
цел больше_ли(дво x, дво y) ;
цел больше_ли(реал x, реал y) ;

//цел большеравны_ли(реал-floating x, реал-floating y);
цел большеравен_ли(плав x, плав y) ;
цел большеравен_ли(дво x, дво y) ;
цел большеравен_ли(реал x, реал y) ;

//цел isless(реал-floating x, реал-floating y);
цел меньше_ли(плав x, плав y) ;
цел меньше_ли(дво x, дво y) ;
цел меньше_ли(реал x, реал y) ;
```

```
//цел меньше_лиequal(реал-floating x, реал-floating y);
цел меньшеравен_ли(плав x, плав y) ;
цел меньшеравен_ли(дво x, дво y) ;
цел меньшеравен_ли(реал x, реал y) ;

//цел меньше_лиgreater(реал-floating x, реал-floating y);
цел меньшебольше_ли(плав x, плав y) ;
цел меньшебольше_ли(дво x, дво y) ;
цел меньшебольше_ли(реал x, реал y) ;

//цел isunordered(реал-floating x, реал-floating y);
цел беспорядочны_ли(плав x, плав y) ;
цел беспорядочны_ли(дво x, дво y) ;
цел беспорядочны_ли(реал x, реал y) ;


дво акос(дво x);
плав акосп(плав x);
реал акосд(реал x);

дво асин(дво x);
плав асинп(плав x);
реал асинд(реал x);

дво атан(дво x);
плав атанп(плав x);
реал атанд(реал x);

дво атан2(дво y, дво x);
плав атан2п(плав y, плав x);
реал атан2д(реал y, реал x);

дво кос(дво x);
плав косп(плав x);
реал косд(реал x);

дво син(дво x);
плав синп(плав x);
реал синд(реал x);

дво тан(дво x);
плав танп(плав x);
реал танд(реал x);

дво акосг(дво x);
плав акосгп(плав x);
реал акосгд(реал x);

дво асинг(дво x);
плав асингп(плав x);
реал асингд(реал x);

дво атанг(дво x);
плав атангп(плав x);
реал атангд(реал x);

дво косг(дво x);
плав косгп(плав x);
реал косгд(реал x);

дво синг(дво x);
плав сингп(плав x);
реал сингд(реал x);
```

```
дво танг(дво х);
плав тангп(плав х);
реал тангд(реал х);

дво эксп(дво х);
плав экспп(плав х);
реал экспд(реал х);

дво эксп2(дво х);
плав эксп2п(плав х);
реал эксп2д(реал х);

дво экспм1(дво х);
плав экспм1п(плав х);
реал экспм1д(реал х);

дво фрэксп(дво value, цел* exp);
плав фрэкспп(плав value, цел* exp);
реал фрэкспд(реал value, цел* exp);

цел илогб(дво х);
цел илогбп(плав х);
цел илогбд(реал х);
/*
дво ldexp(дво х, цел exp);
плав ldexpf(плав х, цел exp);
реал ldexpl(реал х, цел exp);
*/
дво лог(дво х);
плав логп(плав х);
реал логд(реал х);

дво лог10(дво х);
плав лог10п(плав х);
реал лог10д(реал х);

дво лог1п(дво х);
плав лог1пп(плав х);
реал лог1пд(реал х);

дво лог2(дво х);
плав лог2п(плав х);
реал лог2д(реал х);

дво логб(дво х);
плав логбп(плав х);
реал логбд(реал х);

дво модф(дво значение, дво* цук);
плав модфп(плав значение, плав* цук);
реал модфд(реал значение, реал *цук);
/*
дво scalbn(дво х, цел n);
плав scalbnf(плав х, цел n);
реал scalbnl(реал х, цел n);

дво scalbln(дво х, цел n);
плав scalblnf(плав х, цел n);
реал scalblnl(реал х, цел n);
*/
дво кубкор(дво х);
плав кубкорп(плав х);
реал кубкорд(реал х);
```

```
дво фабс(дво х);
плав фабсп(плав х);
реал фабсд(реал х);

дво гипот(дво х, дво у);
плав гипотп(плав х, плав у);
реал гипотд(реал х, реал у);

дво степ(дво х, дво у);
плав степп(плав х, плав у);
реал степд(реал х, реал у);

дво квкор(дво х);
плав квкорп(плав х);
реал квкорд(реал х);

дво фцош(дво х);
плав фцошп(плав х);
реал фцошд(реал х);

дво фцошк(дво х);
плав фцошкп(плав х);
реал фцошкд(реал х);

дво лгамма(дво х);
плав лгаммап(плав х);
реал лгаммад(реал х);

дво тгамма(дво х);
плав тгаммап(плав х);
реал тгаммад(реал х);

дво вокругли(дво х);
плав вокруглип(плав х);
реал вокруглид(реал х);

дво нокругли(дво х);
плав нокруглип(плав х);
реал нокруглид(реал х);

дво ближцел(дво х);
плав ближцелп(плав х);
реал ближцелд(реал х);

дво ринт(дво х);
плав ринтп(плав х);
реал ринтд(реал х);
/*
цел lrint(дво х);
цел lrintf(плав х);
цел lrintl(реал х);

дол llrint(дво х);
дол llrintf(плав х);
дол llrintl(реал х);
*/
дво округли(дво х);
плав округлип(плав х);
реал округлид(реал х);
/*
цел lround(дво х);
цел lroundf(плав х);
цел lroundl(реал х);
```

```
дол llround(дво x);
дол llroundf(плав x);
дол llroundl(реал x);

дво trunc(дво x);
плав truncf(плав x);
реал truncl(реал x);

дво fmod(дво x, дво y);
плав fmodf(плав x, плав y);
реал fmodl(реал x, реал y);
*/
дво остаток(дво x, дво y);
плав остатокп(плав x, плав y);
реал остатокд(реал x, реал y);
/*
дво remquo(дво x, дво y, цел* quo);
плав remquof(плав x, плав y, цел* quo);
реал remquol(реал x, реал y, цел* quo);
*/
дво копируйзнак(дво x, дво y);
плав копируйзнакп(плав x, плав y);
реал копируйзнакд(реал x, реал y);

дво нечисло(ткст tangp);
плав нечислоп(ткст tangp);
реал нечислод(ткст tangp);

дво следза(дво x, дво y);
плав следзап(плав x, плав y);
//реал следзад(реал x, реал y);

//дво следк(дво x, реал y);
//плав следкп(плав x, реал y);
//реал следкд(реал x, реал y);
/*
дво fdim(дво x, дво y);
плав fdimf(плав x, плав y);
реал fdiml(реал x, реал y);

дво fmax(дво x, дво y);
плав fmaxf(плав x, плав y);
реал fmaxl(реал x, реал y);

дво fmin(дво x, дво y);
плав fminf(плав x, плав y);
реал fminl(реал x, реал y);

дво fma(дво x, дво y, дво z);
плав fmaf(плав x, плав y, плав z);
реал fmal(реал x, реал y, реал z);*/

}

extern (D)
{
проц перемотай(фук поток);
проц сбросьош(фук поток);
цел конфл(фук поток);
цел ошфл(фук поток);
}
extern (C):

/*проц перемотай(фук поток);
```

```
проц удалиош(фук поток);
цел кфф(фук поток);
цел ошибф(фук поток);*/

цел удали(in ткст фимя);
//alias удали remove;

цел переименуй(in ткст из, in ткст в);
//alias переименуй rename;

фук времфл();
//alias времфл tmpfile;

ткст времим(ткст s);
//alias времим tmpnam;

цел закройфл(фук поток);
//alias закройфл fclose;

цел слейфл(фук поток);
//alias слейфл fflush;

фук откройфл(in ткст фимя, in ткст режим);
//alias откройфл fopen;

фук переоткройфл(in ткст фимя, in ткст режим, фук поток);
//alias переоткройфл freopen;

проц устбуффл(фук поток, ткст буф);
//alias устбуф setbuf;

цел уствбуф(фук поток, ткст буф, цел режим, т_мера размер);
//alias уствбуф setvbuf;

цел вфвыводф(фук поток, in ткст формат, спис_ва арг);
//alias вфвыводф vfprintf;

цел вфсканф(фук поток, in ткст формат, спис_ва арг);
//alias вфсканф vfscanf;

цел всвыводф(ткст s, in ткст формат, спис_ва арг);
//alias всвыводф vsprintf;

цел вссканф(in ткст s, in ткст формат, спис_ва арг);
//alias вссканф vsscanf;

цел ввыводф(in ткст формат, спис_ва арг);
//alias ввыводф vprinf;

цел всканф(in ткст формат, спис_ва арг);
//alias всканф vscanf;

цел берисфл(фук поток);
//alias берисфл fgetc;

цел вставьсфл(цел с, фук поток);
//alias вставьсфл fputc;

ткст дайтфл(ткст s, цел n, фук поток);
//alias дайтфл fgets;

цел вставьтфл(in ткст s, фук поток);
//alias вставьтфл fputs;
```

```
ткст дайт(ткст s);
//alias дайт gets;

цел вставьт(in ткст s);
//alias вставьт puts;

цел отдайс(цел c, фук поток);
//alias отдайс ungetc;

т_мера читайфл(ук указат, т_мера размер, т_мера nmemb, фук поток);
//alias читайфл fread;

т_мера пишифл(in ук указат, т_мера размер, т_мера nmemb, фук поток);
//alias пишифл fwrite;

цел дайпозфл(фук поток, цел * поз);
//alias дайпозфл fgetpos;

цел устпозфл(фук поток, in цел* поз);
//alias устпозфл fsetpos;

цел сместисьфл(фук поток, цел смещение, цел куда);
//alias сместисьфл fseek;

цел скажифл(фук поток);
//alias скажифл ftell;

цел всквыводф(ткст s, т_мера n, in ткст формат, спис_ва арг);
//alias всквыводф _vsprintf;

проц укошиб(in ткст s);
//alias укошиб perror;

////////////////////////////////////////

т_сигфн сигнал(цел сиг, т_сигфн функ);
//alias сигнал signal;

цел влеки(цел сиг);
//alias влеки raise;

кдво какос(кдво z);
кплав какосп(кплав z);
креал какосд(креал z);

кдво касин(кдво z);
кплав касинп(кплав z);
креал касинд(креал z);

кдво катан(кдво z);
кплав катанп(кплав z);
креал катанд(креал z);

кдво ккос(кдво z);
кплав ккосп(кплав z);
креал ккосд(креал z);

кдво ксин(кдво z);
кплав ксинп(кплав z);
креал ксинд(креал z);

кдво ктан(кдво z);
кплав ктанп(кплав z);
креал ктанд(креал z);
```

```
кдво какосг(кдво z);
кплав какосгп(кплав z);
креал какосгд(креал z);

кдво касинг(кдво z);
кплав касингп(кплав z);
креал касингд(креал z);

кдво катанг(кдво z);
кплав катангп(кплав z);
креал катангд(креал z);

кдво ккосг(кдво z);
кплав ккосгп(кплав z);
креал ккосгд(креал z);

кдво ксинг(кдво z);
кплав ксингп(кплав z);
креал ксингд(креал z);

кдво ктанг(кдво z);
кплав ктангп(кплав z);
креал ктангд(креал z);

кдво кэксп(кдво z);
кплав кэкспп(кплав z);
креал кэкспд(креал z);

кдво клог(кдво z);
кплав клогп(кплав z);
креал клогд(креал z);

дво кабс(кдво z);
плав кабсп(кплав z);
реал кабсд(креал z);

кдво кстеп(кдво x, кдво y);
кплав кстепп(кплав x, кплав y);
креал кстепд(креал x, креал y);

кдво ккквкор(кдво z);
кплав ккквкорп(кплав z);
креал ккквкорд(креал z);

дво карг(кдво z);
плав каргп(кплав z);
реал каргд(креал z);

// дво квообр(кдво z);
//плав квообрп(кплав z);
//реал квообрд(креал z);

//кдво конъюнк(кдво z);
//кплав конъюнкп(кплав z);
//креал конъюнкд(креал z);

кдво кпроекц(кдво z);
кплав кпроекцп(кплав z);
креал кпроекцд(креал z);

//дво креал(кдво z);
//плав креалп(кплав z);
//реал креалд(креал z);
```

```
цел числобукв_ли(цел с);
//alias числобукв_ли isalnum;

цел буква_ли(цел с);
//alias буква_ли isalpha;

цел пробел_ли(цел с);
//alias пробел_ли isblank;

цел управ_ли(цел с);
//alias управ_ли iscntrl;

цел цифра_ли(цел с);
//alias цифра_ли isdigit;

цел граф_ли(цел с);
//alias граф_ли isgraph;

цел проп_ли(цел с);
//alias проп_ли islower;

цел печат_ли(цел с);
//alias печат_ли isprint;

цел пункт_ли(цел с);
//alias пункт_ли ispunct;

цел межбукв_ли(цел с);
//alias межбукв_ли isspace;

цел заг_ли(цел с);
//alias заг_ли isupper;

цел цифраикс_ли(цел с);
//alias цифраикс_ли isxdigit;

цел впроп(цел с);
//alias впроп tolower;

цел взаг(цел с);
//alias взаг toupper;


цел числобуквш_ли(шим с);
//alias числобуквш_ли iswalnum;

цел буквaш_ли(шим с);
//alias буквaш_ли iswalpha;

//цел пробелш_ли(шим с);
////alias пробелш_ли iswblank;

цел управш_ли(шим с);
//alias управш_ли iswcntrl;

цел цифраш_ли(шим с);
//alias цифраш_ли iswdigit;

цел графш_ли(шим с);
//alias графш_ли iswgraph;

цел пропш_ли(шим с);
//alias пропш_ли iswlower;
```

```
цел печатш_ли(шим с);
//alias печатш_ли iswprint;

цел пунктш_ли(шим с);
//alias пунктш_ли iswpunct;

цел межбуквш_ли(шим с);
//alias межбуквш_ли iswspace;

цел загш_ли(шим с);
//alias загш_ли iswupper;

цел цифраиксш_ли(шим с);
//alias цифраиксш_ли iswxdigit;

цел впропш(шим с);
//alias впропш towlower;

цел взагш(шим с);
//alias взагш towupper;

//шим втрансш(шим ш, шим опис);
//alias втрансш towctrans;

//шим трансш( in ткст0 свойство);
//alias трансш wctrans;

цел дайНомош();
//alias дайНомош getErrno;

цел устНомош(цел n);
//alias устНомош setErrno;

проц влекиисклфе(цел исклы);
//alias влекиисклфе feraiseexcept;

проц сотриисклфе(цел исклы);
//alias сотриисклфе feclearexcept;

цел тестисклфе(цел исклы);
//alias тестисклфе fetestexcept;

цел задержиисклфе(т_фсред* средп);
//alias задержиисклфе feholdexcept;

проц дайфлагисклфе(цел* флагп, цел исклы);
//alias дайфлагисклфе fegetexceptflag;

проц устфлагисклфе(in цел* флагп, цел исклы);
//alias устфлагисклфе fesetexceptglag;

цел дайкругфе();
//alias дайкругфе fegetround;

цел усткругфе(цел круг);
//alias усткругфе fesetround;

проц дайсредфе(т_фсред* средп);
//alias дайсредфе fegetenv;

проц устсредфе(in т_фсред* средп);
//alias устсредфе fesetenv;

проц обновисредфе(in т_фсред* средп);
```

```
//alias обновисредфе feupdateenv;

дол цмаксабс(дол j);
//alias цмаксабс imaxabs;

т_цмаксдел цмаксдел(дол число, дол делитель);
//alias цмаксдел imaxdiv;

дол ткствмаксц(in ткст чук, ткст* конук, цел основа);
//alias ткствмаксц strtoimax;

бдол ткствбмакс(in ткст чук, ткст* конук, цел основа);
//alias ткствбмакс strtoumax;

дол шимвцмакс(in шткст чук, шткст* конук, цел основа);
//alias шимвцмакс wcstoimax;

бдол шимвбмакс(in шткст чук, шткст* конук, цел основа);
//alias шимвбмакс wcstoumax;

ткст устлокаль(цел категория, in ткст локаль);
//alias устлокаль setlocale;

лпреобр* преобрлокаль();
//alias преобрлокаль localeconv;

бцел __пткласифицируй_п(плав х);
бцел __пткласифицируй_д(дво х);
бцел __пткласифицируй_дд(реал х);

проц _си_выход();
//alias _си_выход _c_exit;

проц _сивыход();
//alias _сивыход _cexit;

проц _выход(цел х);
//alias _выход _exit;

проц _аборт();
//alias _аборт _abort;

проц _деструкт();
//alias _деструкт _dodtors;

цел дайпид();
//alias дайпид getpid;

проц аборт();
//alias аборт abort;

проц выход(цел статус);
//alias выход exit;

цел навыходе(проц function() функц);
//alias навыходе atexit;

проц _Выход(цел статус);
//alias _Выход _Exit;

дво алфнапэ(in ткст укнач);
//alias алфнапэ atof;
цел алфнац(in ткст укнач);
//alias алфнац atoi;
```

```
цел алфнадл(in ткст укнач);
//alias алфнадл atol;

дол алфнадлдл(in ткст укнач);
//alias алфнадлдл atoll;

дво стрнад(in ткст укнач, ткст* укнакон);
//alias стрнад strtod;

плав стрнапз(in ткст укнач, ткст* укнакон);
//alias стрнапз strtof;

реал стрнадлд(in ткст укнач, ткст* укнакон);
//alias стрнадлд strtold;

цел стрнадл(in ткст укнач, ткст* укнакон, цел ова);
//alias стрнадл strtol;

дол стрнадлдл(in ткст укнач, ткст* укнакон, цел ова);
//alias стрнадлдл strtoll;

бцел стрнабдл(in ткст укнач, ткст* укнакон, цел ова);
//alias стрнабдл strtoul;

бдол стрнабдлдл(in ткст укнач, ткст* укнакон, цел ова);
//alias стрнабдлдл strtoull;

цел случ();
//alias случ rand;

проц сслуч(бцел семя);
//alias сслуч srand;

ук празмести(т_мера разм);
//alias празмести malloc;

ук кразмести(т_мера члочленов, т_мера разм);
//alias кразмести calloc;

ук перемести(ук указ, т_мера разм);
//alias перемести realloc;

проц освободи(ук указ);
//alias освободи free;

ткст дайсреду(in ткст имя);
//alias дайсреду getenv;

цел система(in ткст текст);
//alias система system;

ук бпоиск(in ук key, in ук ова, т_мера члочленов, т_мера разм, цел
function(in ук, in ук) compar);
//alias бпоиск bsearch;

проц бсорт(ук ова, т_мера члочленов, т_мера разм, цел function(in ук, in ук)
compar);
//alias бсорт qsort;

цел абс(цел j);
//alias абс abs;

цел абсц(цел j);
```

```
//alias абсц labs;

дол абсд(дол j);
//alias абсд llabs;

т_дели дели(цел число, цел делитель);
//alias дели div;

т_делиц делиц(цел число, цел делитель);
//alias делиц ldiv;

т_делид делид(дол число, дол делитель);
//alias делид lldiv;


цел мбдлин(in ткст s, т_мера n);
//alias мбдлин mblen;

цел мбнашк(шткст pwc, in ткст s, т_мера n);
//alias мбнашк mbtowc;

цел шкнамб(ткст s, шим wc);
//alias шкнамб wctomb;

т_мера мбтнашкт(шткст pwcs, in ткст s, т_мера n);
//alias мбтнашкт mbstowcs;

т_мера шктнамбт(ткст s, in шткст pwcs, т_мера n);
//alias шктнамбт wcstombs;

цел поместсфл(цел ц, фук ф);
цел поместшфл(цел ц, фук ф);
цел извлсфл(фук ф);
цел извлшфл(фук ф);
цел блокфл(фук ф);
проц разблокфл(фук ф);

цел ширфл(фук поток, цел реж);
//alias ширфл fwide;

ук разместа(т_мера разм);
//alias разместа alloca;
цел сравбуфлюб(ткст0 буф1, ткст0 буф2, т_мера члоб);
alias сравбуфлюб сравни_буферы_люб;//, memicmp;

ук ищисим(in ук строка, цел сим, т_мера члобайт);
alias ищисим ищи_символ;//, memchr;

цел сравбуф(in ук буф1, in ук буф2, т_мера члобайт);
alias сравбуф сравни_буферы ;//, memcmp;

ук копирбуф(ук приёмник, in ук исток, т_мера члобайт);
alias копирбуф копируй_буфер;//, memcpy;

ук перембуф(ук куда, in ук откуда, т_мера сколько);
alias перембуф перемести_буфер;//, memmove;

ук устбуф(ук где, цел сим, т_мера члосим);
alias устбуф установи_буфер;//, memset;

ткст0 копиртекс(ткст0 куда, in ткст0 откуда);
alias копиртекс копируй_символы;//, strcpy;

ткст0 копирчтекс(ткст0 куда, in ткст0 откуда, т_мера члосим);
```

```
alias копирчтекс копируй_чло_сим ;//, strncpy;

ткст0 сотекс(ткст0 текст1, in ткст0 текст_плюс);
alias сотекс соедини_тексты ;//, strcat;

ткст0 сочтекс(ткст0 ткст1, in ткст0 ткст2, т_мера члосим);
alias сочтекс соедини_чло_сим ;//, strncat;

цел сравтекс(in ткст0 текст1, in ткст0 текст2);
alias сравтекс сравни_тексты;//, strcmp;

цел кодстрсравнитекс( in ткст0 текст1, in ткст0 текст2);
alias кодстрсравнитекс кссравтекс;//, strcoll;

цел сравчтекс(in ткст0 текст1, in ткст0 текст2, т_мера члосим);
alias сравчтекс сравни_чло_сим ;//, strncmp;

т_мера форматчтекс(ткст0 в, in ткст0 из, т_мера чло);
alias форматчтекс преобразуй_чло_сим_лок;//, strxfrm;

ткст0 найдипер(in ткст0 т, цел с);
alias найдипер найди_перв_сим ;//, strchr;

т_мера персинд(in ткст0 где, in ткст0 что);
alias персинд дай_индекс_перв_сим;//, strcspn;

ткст0 найдитексвнаб(in ткст0 вчём, in ткст0 изчего);
alias найдитексвнаб найди_сим_из_набора ;//, strpbrk;

ткст0 найдипос(in ткст0 ткс, цел сим);
alias найдипос найди_посл_сим;//, strrchr;

т_мера найдитекснеизнаб(in ткст0 вчём, in ткст0 изчего);
alias найдитекснеизнаб найди_сим_не_из_набора ;//, strspn;

ткст0 найдиподтекс (in ткст0 стр, in ткст0 иском);
alias найдиподтекс найди_подтекст ;//, strstr;

ткст0 стрзнак(ткст стрзнак0, in ткст строгран);
//alias стрзнак strtok;

ткст строшиб(цел номош);
//alias строшиб strerror;

т_мера длинтекс(in ткст0 текст);
//alias длинтекс strlen;


т_мера длинашкс (in шим* с);
//alias длинашкс wcslen;

ук начнить(_У адр, бцел размстэка, ук аргспис);
//alias начнить _beginthread;

проц стопнить();
//alias стопнить _endthread;

ук начнитьдоп(ук безоп, бцел рамзстэка, винфункбЦ_У адр, ук аргспис, бцел
иницфлаг, бцел* адрнити);
//alias начнитьдоп _beginthreadex;

проц стопнитьдоп(бцел кодвых);
//alias стопнитьдоп _endthreadex;
```

```
//////////////////////////////////////

const CHAR_BIT = 8;
const SCHAR_MIN = byte.min;
const SCHAR_MAX = byte.max;
const UCHAR_MAX = ubyte.min;
const CHAR_MIN = сим.max;
const CHAR_MAX = сим.max;
const MB_LEN_MAX = 2;
const SHRT_MIN = short.min;
const SHRT_MAX = short.max;
const USHRT_MAX = ushort.max;
const INT_MIN = int.min;
const INT_MAX = int.max;
const UINT_MAX = uint.max;
const LONG_MIN = c_long.min;
const LONG_MAX = c_long.max;
const ULONG_MAX = c_ulong.max;
const LLONG_MIN = long.min;
const LLONG_MAX = long.max;
const ULLONG_MAX = ulong.max;

const int FP_ILOGB0 = int.min;
const int FP_ILOGBNAN = int.min;
enum
{
FP_NANS = 0,
FP_NANQ = 1,
FP_INFINITE = 2,
FP_NORMAL = 3,
FP_SUBNORMAL = 4,
FP_ZERO = 5,
FP_NAN = FP_NANQ,
FP_EMPTY = 6,
FP_UNSUPPORTED = 7,
}
     enum
{
SEEK_SET,
SEEK_CUR,
SEEK_END
}
     struct div_t
     {
          int quot,
               rem;
     }


struct ldiv_t
     {
          int quot,
               rem;
     }
struct lldiv_t
     {
          long quot,
               rem;
     }
     struct imaxdiv_t
{
intmax_t quot,
rem;
}
```

```
intmax_t imaxabs(intmax_t j);
imaxdiv_t imaxdiv(intmax_t numer, intmax_t denom);
intmax_t strtoimax(in char* nptr, char** endptr, int base);
uintmax_t strtoumax(in char* nptr, char** endptr, int base);
intmax_t wcstoimax(in wchar_t* nptr, wchar_t** endptr, int base);
uintmax_t wcstoumax(in wchar_t* nptr, wchar_t** endptr, int base);



void _c_exit();
void _cexit();
void _exit(int);
void abort();
void _dodtors();
int getpid();
void exit(int status);
int atexit(void function() func);
void _Exit(int status);



enum { _P_WAIT, _P_NOWAIT, _P_OVERLAY };

int execl(char *, char *,...);
int execle(char *, char *,...);
int execlp(char *, char *,...);
int execlpe(char *, char *,...);
int execv(char *, char **);
int execve(char *, char **, char **);
int execvp(char *, char **);
int execvpe(char *, char **, char **);


enum { WAIT_CHILD, WAIT_GRANDCHILD }

int cwait(int *,int,int);
int wait(int *);

version (Windows)
{

uint _beginthread( _Y ,uint, void *);

extern (Windows) alias uint (*stdfp)(void *);

uint _beginthreadex(void* security, uint stack_size,
     stdfp start_addr, void* arglist, uint initflag,
     uint* thrdaddr);

void _endthread();
void _endthreadex(uint);

int spawnl(int, char *, char *,...);
int spawnle(int, char *, char *,...);
int spawnlp(int, char *, char *,...);
int spawnlpe(int, char *, char *,...);
int spawnv(int, char *, char **);
int spawnve(int, char *, char **, char **);
int spawnvp(int, char *, char **);
int spawnvpe(int, char *, char **, char **);
```

```
int _wsystem(wchar_t *);
int _wspawnl(int, wchar_t *, wchar_t *, ...);
int _wspawnle(int, wchar_t *, wchar_t *, ...);
int _wspawnlp(int, wchar_t *, wchar_t *, ...);
int _wspawnlpe(int, wchar_t *, wchar_t *, ...);
int _wspawnv(int, wchar_t *, wchar_t **);
int _wspawnve(int, wchar_t *, wchar_t **, wchar_t **);
int _wspawnvp(int, wchar_t *, wchar_t **);
int _wspawnvpe(int, wchar_t *, wchar_t **, wchar_t **);

int _wexecl(wchar_t *, wchar_t *, ...);
int _wexecle(wchar_t *, wchar_t *, ...);
int _wexeclp(wchar_t *, wchar_t *, ...);
int _wexeclpe(wchar_t *, wchar_t *, ...);
int _wexecv(wchar_t *, wchar_t **);
int _wexecve(wchar_t *, wchar_t **, wchar_t **);
int _wexecvp(wchar_t *, wchar_t **);
int _wexecvpe(wchar_t *, wchar_t **, wchar_t **);
}


int iswalnum(wint_t wc);
int iswalpha(wint_t wc);
int iswblank(wint_t wc);
int iswcntrl(wint_t wc);
int iswdigit(wint_t wc);
int iswgraph(wint_t wc);
int iswlower(wint_t wc);
int iswprint(wint_t wc);
int iswpunct(wint_t wc);
int iswspace(wint_t wc);
int iswupper(wint_t wc);
int iswxdigit(wint_t wc);

int iswctype(wint_t wc, wctype_t desc);
wctype_t wctype(in char* property);
wint_t towlower(wint_t wc);
wint_t towupper(wint_t wc);
wint_t towctrans(wint_t wc, wctrans_t desc);
wctrans_t wctrans(in char* property);
     void* memchr(in void* s, int c, size_t n);
int memcmp(in void* s1, in void* s2, size_t n);
void* memcpy(void* s1, in void* s2, size_t n);
void* memmove(void* s1, in void* s2, size_t n);
void* memset(void* s, int c, size_t n);

char* strcpy(char* s1, in char* s2);
char* strncpy(char* s1, in char* s2, size_t n);
char* strcat(char* s1, in char* s2);
char* strncat(char* s1, in char* s2, size_t n);
int strcmp(in char* s1, in char* s2);
int strcoll(in char* s1, in char* s2);
int strncmp(in char* s1, in char* s2, size_t n);
size_t strxfrm(char* s1, in char* s2, size_t n);
char* strchr(in char* s, int c);
size_t strcspn(in char* s1, in char* s2);
char* strpbrk(in char* s1, in char* s2);
char* strrchr(in char* s, int c);
size_t strspn(in char* s1, in char* s2);
char* strstr(in char* s1, in char* s2);
char* strtok(char* s1, in char* s2);
char* strerror(int errnum);
size_t strlen(in char* s);
```

```
int memicmp(char* s1, char* s2, size_t n);
     int _fputc_nlock(int, FILE*);
     int _fputwc_nlock(int, FILE*);
     int _fgetc_nlock(FILE*);
     int _fgetwc_nlock(FILE*);
     int __fp_lock(FILE*);
     проц __fp_unlock(FILE*);
     int getErrno(); // for internal use
     int setErrno(int); // for internal use
     struct fenv_t
{
ushort status;
ushort control;
ushort round;
ushort[2] reserved;
}
     void feraiseexcept(int excepts);
     void feclearexcept(int excepts);

     int fetestexcept(int excepts);
     int feholdexcept(fenv_t* envp);

     void fegetexceptflag(fexcept_t* flagp, int excepts);
     void fesetexceptflag(in fexcept_t* flagp, int excepts);

     int fegetround();
     int fesetround(int round);

     void fegetenv(fenv_t* envp);
     void fesetenv(in fenv_t* envp);
     void feupdateenv(in fenv_t* envp);
     alias creal complex;
alias ireal imaginary;

cdouble cacos(cdouble z);
cfloat cacosf(cfloat z);
creal cacosl(creal z);

cdouble casin(cdouble z);
cfloat casinf(cfloat z);
creal casinl(creal z);

cdouble catan(cdouble z);
cfloat catanf(cfloat z);
creal catanl(creal z);

cdouble ccos(cdouble z);
cfloat ccosf(cfloat z);
creal ccosl(creal z);

cdouble csin(cdouble z);
cfloat csinf(cfloat z);
creal csinl(creal z);

cdouble ctan(cdouble z);
cfloat ctanf(cfloat z);
creal ctanl(creal z);

cdouble cacosh(cdouble z);
cfloat cacoshf(cfloat z);
creal cacoshl(creal z);

cdouble casinh(cdouble z);
cfloat casinhf(cfloat z);
```

```
creal casinhl(creal z);

cdouble catanh(cdouble z);
cfloat catanhf(cfloat z);
creal catanhl(creal z);

cdouble ccosh(cdouble z);
cfloat ccoshf(cfloat z);
creal ccoshl(creal z);

cdouble csinh(cdouble z);
cfloat csinhf(cfloat z);
creal csinhl(creal z);

cdouble ctanh(cdouble z);
cfloat ctanhf(cfloat z);
creal ctanhl(creal z);

cdouble cexp(cdouble z);
cfloat cexpf(cfloat z);
creal cexpl(creal z);

cdouble clog(cdouble z);
cfloat clogf(cfloat z);
creal clogl(creal z);

double cabs(cdouble z);
float cabsf(cfloat z);
real cabsl(creal z);

cdouble cpow(cdouble x, cdouble y);
cfloat cpowf(cfloat x, cfloat y);
creal cpowl(creal x, creal y);

cdouble csqrt(cdouble z);
cfloat csqrtf(cfloat z);
creal csqrtl(creal z);

double carg(cdouble z);
float cargf(cfloat z);
real cargl(creal z);

double cimag(cdouble z);
float cimagf(cfloat z);
real cimagl(creal z);

cdouble conj(cdouble z);
cfloat conjf(cfloat z);
creal conjl(creal z);

cdouble cproj(cdouble z);
cfloat cprojf(cfloat z);
creal cprojl(creal z);

// double creal(cdouble z);
float crealf(cfloat z);
real creall(creal z);
int isalnum(int c);
int isalpha(int c);
int isblank(int c);
int iscntrl(int c);
int isdigit(int c);
int isgraph(int c);
int islower(int c);
```

```
int isprint(int c);
int ispunct(int c);
int isspace(int c);
int isupper(int c);
int isxdigit(int c);
int tolower(int c);
int toupper(int c);

struct lconv
{
char* decimal_point;
char* thousands_sep;
char* grouping;
char* int_curr_symbol;
char* currency_symbol;
char* mon_decimal_point;
char* mon_thousands_sep;
char* mon_grouping;
char* positive_sign;
char* negative_sign;
byte int_frac_digits;
byte frac_digits;
byte p_cs_precedes;
byte p_sep_by_space;
byte n_cs_precedes;
byte n_sep_by_space;
byte p_sign_posn;
byte n_sign_posn;
byte int_p_cs_precedes;
byte int_p_sep_by_space;
byte int_n_cs_precedes;
byte int_n_sep_by_space;
byte int_p_sign_posn;
byte int_n_sign_posn;
}

char* setlocale(int category, in char* locale);
lconv* localeconv();
     uint __fpclassify_f(float x);
uint __fpclassify_d(double x);
uint __fpclassify_ld(real x);
     double acos(double x);
float acosf(float x);
real acosl(real x);

double asin(double x);
float asinf(float x);
real asinl(real x);

double atan(double x);
float atanf(float x);
real atanl(real x);

double atan2(double y, double x);
float atan2f(float y, float x);
real atan2l(real y, real x);

double cos(double x);
float cosf(float x);
real cosl(real x);

double sin(double x);
float sinf(float x);
real sinl(real x);
```

```
double tan(double x);
float tanf(float x);
real tanl(real x);

double acosh(double x);
float acoshf(float x);
real acoshl(real x);

double asinh(double x);
float asinhf(float x);
real asinhl(real x);

double atanh(double x);
float atanhf(float x);
real atanhl(real x);

double cosh(double x);
float coshf(float x);
real coshl(real x);

double sinh(double x);
float sinhf(float x);
real sinhl(real x);

double tanh(double x);
float tanhf(float x);
real tanhl(real x);

double exp(double x);
float expf(float x);
real expl(real x);

double exp2(double x);
float exp2f(float x);
real exp2l(real x);

double expm1(double x);
float expm1f(float x);
real expm1l(real x);

double frexp(double value, int* exp);
float frexpf(float value, int* exp);
real frexpl(real value, int* exp);

int ilogb(double x);
int ilogbf(float x);
int ilogbl(real x);

double ldexp(double x, int exp);
float ldexpf(float x, int exp);
real ldexpl(real x, int exp);

double log(double x);
float logf(float x);
real logl(real x);

double log10(double x);
float log10f(float x);
real log10l(real x);

double log1p(double x);
float log1pf(float x);
real log1pl(real x);
```

```
double log2(double x);
float log2f(float x);
real log2l(real x);

double logb(double x);
float logbf(float x);
real logbl(real x);

double modf(double value, double* iptr);
float modff(float value, float* iptr);
real modfl(real value, real *iptr);

double scalbn(double x, int n);
float scalbnf(float x, int n);
real scalbnl(real x, int n);

double scalbln(double x, c_long n);
float scalblnf(float x, c_long n);
real scalblnl(real x, c_long n);

double cbrt(double x);
float cbrtf(float x);
real cbrtl(real x);

double fabs(double x);
float fabsf(float x);
real fabsl(real x);

double hypot(double x, double y);
float hypotf(float x, float y);
real hypotl(real x, real y);

double pow(double x, double y);
float powf(float x, float y);
real powl(real x, real y);

double sqrt(double x);
float sqrtf(float x);
real sqrtl(real x);

double erf(double x);
float erff(float x);
real erfl(real x);

double erfc(double x);
float erfcf(float x);
real erfcl(real x);

double lgamma(double x);
float lgammaf(float x);
real lgammal(real x);

double tgamma(double x);
float tgammaf(float x);
real tgammal(real x);

double ceil(double x);
float ceilf(float x);
real ceill(real x);

double floor(double x);
float floorf(float x);
real floorl(real x);
```

```
double nearbyint(double x);
float nearbyintf(float x);
real nearbyintl(real x);

double rint(double x);
float rintf(float x);
real rintl(real x);

c_long lrint(double x);
c_long lrintf(float x);
c_long lrintl(real x);

long llrint(double x);
long llrintf(float x);
long llrintl(real x);

double round(double x);
float roundf(float x);
real roundl(real x);

c_long lround(double x);
c_long lroundf(float x);
c_long lroundl(real x);

long llround(double x);
long llroundf(float x);
long llroundl(real x);

double trunc(double x);
float truncf(float x);
real truncl(real x);

double fmod(double x, double y);
float fmodf(float x, float y);
real fmodl(real x, real y);

double remainder(double x, double y);
float remainderf(float x, float y);
real remainderl(real x, real y);

double remquo(double x, double y, int* quo);
float remquof(float x, float y, int* quo);
real remquol(real x, real y, int* quo);

double copysign(double x, double y);
float copysignf(float x, float y);
real copysignl(real x, real y);

double nan(char* tagp);
float nanf(char* tagp);
real nanl(char* tagp);

double nextafter(double x, double y);
float nextafterf(float x, float y);
real nextafterl(real x, real y);

double nexttoward(double x, real y);
float nexttowardf(float x, real y);
real nexttowardl(real x, real y);

double fdim(double x, double y);
float fdimf(float x, float y);
real fdiml(real x, real y);
```

```
double fmax(double x, double y);
float fmaxf(float x, float y);
real fmaxl(real x, real y);

double fmin(double x, double y);
float fminf(float x, float y);
real fminl(real x, real y);

double fma(double x, double y, double z);
float fmaf(float x, float y, float z);
real fmal(real x, real y, real z);
    int remove(in char* filename);
int rename(in char* from, in char* to);

FILE* tmpfile();
char* tmpnam(char* s);

int fclose(FILE* stream);
int fflush(FILE* stream);
FILE* fopen(in char* filename, in char* mode);
FILE* freopen(in char* filename, in char* mode, FILE* stream);

void setbuf(FILE* stream, char* buf);
int setvbuf(FILE* stream, char* buf, int mode, size_t size);

int fprintf(FILE* stream, in char* format, ...);
int fscanf(FILE* stream, in char* format, ...);
int sprintf(char* s, in char* format, ...);
int sscanf(in char* s, in char* format, ...);
int vfprintf(FILE* stream, in char* format, va_list apr);
int vfscanf(FILE* stream, in char* format, va_list apr);
int vsprintf(char* s, in char* format, va_list apr);
int vsscanf(in char* s, in char* format, va_list apr);
int vprintf(in char* format, va_list apr);
int vscanf(in char* format, va_list apr);
//int ǝxo(in char* format, ...);
int scanf(in char* format, ...);

int fgetc(FILE* stream);
int fputc(int c, FILE* stream);

char* fgets(char* s, int n, FILE* stream);
int fputs(in char* s, FILE* stream);
char* gets(char* s);
int puts(in char* s);

int ungetc(int c, FILE* stream);

size_t fread(void* ptr, size_t size, size_t nmemb, FILE* stream);
size_t fwrite(in void* ptr, size_t size, size_t nmemb, FILE* stream);

int fgetpos(FILE* stream, fpos_t * pos);
int fsetpos(FILE* stream, in fpos_t* pos);

int fseek(FILE* stream, c_long offset, int whence);
c_long ftell(FILE* stream);

int _snprintf(char* s, size_t n, in char* fmt, ...);
alias _snprintf snprintf;

int _vsnprintf(char* s, size_t n, in char* format, va_list apr);
alias _vsnprintf vsnprintf;
```

```
void perror(in char* s);

double atof(in char* nptr);
int atoi(in char* nptr);
c_long atol(in char* nptr);
long atoll(in char* nptr);

double strtod(in char* nptr, char** endptr);
float strtof(in char* nptr, char** endptr);
real strtold(in char* nptr, char** endptr);
c_long strtol(in char* nptr, char** endptr, int base);
long strtoll(in char* nptr, char** endptr, int base);
c_ulong strtoul(in char* nptr, char** endptr, int base);
ulong strtoull(in char* nptr, char** endptr, int base);

int rand();
void srand(uint seed);

void* malloc(size_t size);
void* calloc(size_t nmemb, size_t size);
void* realloc(void* ptr, size_t size);
void free(void* ptr);

char* getenv(in char* name);
int system(in char* string);

void* bsearch(in void* key, in void* base, size_t nmemb, size_t size, int
function(in void*, in void*) compar);
void qsort(void* base, size_t nmemb, size_t size, int function(in void*, in
void*) compar);

int abs(int j);
c_long labs(c_long j);
long llabs(long j);

div_t div(int numer, int denom);
ldiv_t ldiv(c_long numer, c_long denom);
lldiv_t lldiv(long numer, long denom);

int mblen(in char* s, size_t n);
int mbtowc(wchar_t* pwc, in char* s, size_t n);
int wctomb(char*s, wchar_t wc);
size_t mbstowcs(wchar_t* pwcs, in char* s, size_t n);
size_t wcstombs(char* s, in wchar_t* pwcs, size_t n);

version( DigitalMars )
    {
        void* alloca(size_t size); // non-standard
    }
    version( Windows )
{
struct tm
{
int tm_sec; // seconds after the minute - [0, 60]
int tm_min; // minutes after the hour - [0, 59]
int tm_hour; // hours since midnight - [0, 23]
int tm_mday; // day of the month - [1, 31]
int tm_mon; // months since January - [0, 11]
int tm_year; // years since 1900
int tm_wday; // days since Sunday - [0, 6]
int tm_yday; // days since January 1 - [0, 365]
int tm_isdst; // Daylight Saving Time flag
}
}
```

```
else
{
struct tm
{
int tm_sec; // seconds after the minute [0-60]
int tm_min; // minutes after the hour [0-59]
int tm_hour; // hours since midnight [0-23]
int tm_mday; // day of the month [1-31]
int tm_mon; // months since January [0-11]
int tm_year; // years since 1900
int tm_wday; // days since Sunday [0-6]
int tm_yday; // days since January 1 [0-365]
int tm_isdst; // Daylight Savings Time flag
c_long tm_gmtoff; // offset from CUT in seconds
char* tm_zone; // timezone abbreviation
}
}

alias c_long time_t;
alias c_long clock_t;


clock_t CLOCKS_PER_SEC = 1000;


clock_t clock();
double difftime(time_t time1, time_t time0);
time_t mktime(tm* timeptr);
time_t time(time_t* timer);
char* asctime(in tm* timeptr);
char* ctime(in time_t* timer);
tm* gmtime(in time_t* timer);
tm* localtime(in time_t* timer);
size_t strftime(char* s, size_t maxsize, in char* format, in tm* timeptr);

void tzset();         // non-standard
void _tzset();         // non-standard
char* _strdate(char* s); // non-standard
char* _strtime(char* s); // non-standard
    alias int mbstate_t;
//alias wchar_t wint_t;

//const wchar_t WEOF = 0xFFFF;

int fwprintf(FILE* stream, in wchar_t* format, ...);
int fwscanf(FILE* stream, in wchar_t* format, ...);
int swprintf(wchar_t* s, size_t n, in wchar_t* format, ...);
int swscanf(in wchar_t* s, in wchar_t* format, ...);
int vfwprintf(FILE* stream, in wchar_t* format, va_list apr);
int vfwscanf(FILE* stream, in wchar_t* format, va_list apr);
int vswprintf(wchar_t* s, size_t n, in wchar_t* format, va_list apr);
int vswscanf(in wchar_t* s, in wchar_t* format, va_list apr);
int vwprintf(in wchar_t* format, va_list apr);
int vwscanf(in wchar_t* format, va_list apr);
int wprintf(in wchar_t* format, ...);
int wscanf(in wchar_t* format, ...);

wint_t fgetwc(FILE* stream);
wint_t fputwc(wchar_t c, FILE* stream);

wchar_t* fgetws(wchar_t* s, int n, FILE* stream);
int fputws(in wchar_t* s, FILE* stream);

wint_t ungetwc(wint_t c, FILE* stream);
```

```
int fwide(FILE* stream, int mode);

double wcstod(in wchar_t* nptr, wchar_t** endptr);
float wcstof(in wchar_t* nptr, wchar_t** endptr);
real wcstold(in wchar_t* nptr, wchar_t** endptr);
c_long wcstol(in wchar_t* nptr, wchar_t** endptr, int base);
long wcstoll(in wchar_t* nptr, wchar_t** endptr, int base);
c_ulong wcstoul(in wchar_t* nptr, wchar_t** endptr, int base);
ulong wcstoull(in wchar_t* nptr, wchar_t** endptr, int base);

wchar_t* wcscpy(wchar_t* s1, in wchar_t* s2);
wchar_t* wcsncpy(wchar_t* s1, in wchar_t* s2, size_t n);
wchar_t* wcscat(wchar_t* s1, in wchar_t* s2);
wchar_t* wcsncat(wchar_t* s1, in wchar_t* s2, size_t n);
int wcscmp(in wchar_t* s1, in wchar_t* s2);
int wcscoll(in wchar_t* s1, in wchar_t* s2);
int wcsncmp(in wchar_t* s1, in wchar_t* s2, size_t n);
size_t wcsxfrm(wchar_t* s1, in wchar_t* s2, size_t n);
wchar_t* wcschr(in wchar_t* s, wchar_t c);
size_t wcscspn(in wchar_t* s1, in wchar_t* s2);
wchar_t* wcspbrk(in wchar_t* s1, in wchar_t* s2);
wchar_t* wcsrchr(in wchar_t* s, wchar_t c);
size_t wcsspn(in wchar_t* s1, in wchar_t* s2);
wchar_t* wcsstr(in wchar_t* s1, in wchar_t* s2);
wchar_t* wcstok(wchar_t* s1, in wchar_t* s2, wchar_t** ptr);
size_t wcslen(in wchar_t* s);

wchar_t* wmemchr(in wchar_t* s, wchar_t c, size_t n);
int wmemcmp(in wchar_t* s1, in wchar_t* s2, size_t n);
wchar_t* wmemcpy(wchar_t* s1, in wchar_t* s2, size_t n);
wchar_t* wmemmove(wchar_t*s1, in wchar_t* s2, size_t n);
wchar_t* wmemset(wchar_t* s, wchar_t c, size_t n);

size_t wcsftime(wchar_t* s, size_t maxsize, in wchar_t* format, in tm*
timeptr);

version( Windows )
{
wchar_t* _wasctime(tm*); // non-standard
wchar_t* _wctime(time_t*);     // non-standard
wchar_t* _wstrdate(wchar_t*); // non-standard
wchar_t* _wstrtime(wchar_t*); // non-standard
}

wint_t btowc(int c);
int wctob(wint_t c);
int mbsinit(in mbstate_t* ps);
size_t mbrlen(in char* s, size_t n, mbstate_t* ps);
size_t mbrtowc(wchar_t* pwc, in char* s, size_t n, mbstate_t* ps);
size_t wcrtomb(char* s, wchar_t wc, mbstate_t* ps);
size_t mbsrtowcs(wchar_t* dst, in char** src, size_t len, mbstate_t* ps);
size_t wcsrtombs(char* dst, in wchar_t** src, size_t len, mbstate_t* ps);

sigfn_t signal(int sig, sigfn_t func);
int raise(int sig);
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

```
module exception;
pragma(lib,"dinrus.lib");

типПроверОбр проверОбр = пусто;
типСледОбр трОбр = пусто;


extern(C) struct СисОш
{
static бцел последнКод ();
static ткст последнСооб ();
static ткст найди (бцел кодош);
}

extern (D):
class ИсклВнешнМодуля:Исключение
{
     this(ткст сооб);
this(бцел кодош);
}
class ФайлИскл : Исключение
{
бцел номош;
     this(ткст имя);
this(ткст имя, ткст сооб);
this(ткст имя, бцел номош);
}
class ИсклНедостающЧлена : Исключение
{
this() ;
this(ткст сообщение) ;
this(ткст имяКласса, ткст имяЧлена);
}

class ИсклКОМ : Исключение
{
this(цел кодОшибки) ;
this(ткст сообщение, цел кодОшибки) ;
цел кодОшибки();
}

class ВнеПамИскл : Исключение
     {
          this( ткст файл =__FILE__, т_мера строка =__LINE__ );
          override ткст toString();
     }
alias ВнеПамИскл OutOfMemoryException;


class ОтслежИскл : Исключение
     {
          this( ткст сооб );
          this( ткст сооб, Исключение е );
          this( ткст сооб, ткст файл, т_мера строка );
          ткст toString();
          ткст вТкст();
          цел opApply( цел delegate( inout ткст буф ) дг );
     }
alias ОтслежИскл TracedException;


class ПлатформИскл : ОтслежИскл
```

```
{
this( ткст сооб );
}
alias ПлатформИскл PlatformException;


class ПроверИскл : ОтслежИскл
{
this( ткст файл =__FILE__, т_мера строка = __LINE__);
this( ткст сооб, ткст файл = __FILE__, т_мера строка =__LINE__ );
}
alias ПроверИскл AssertException;


class ПроверОшиб : Искл
     {
         this(ткст имяф =__FILE__, бцел номстроки =__LINE__);
         this(ткст сооб, ткст имяф = __FILE__, бцел номстроки = __LINE__);
         ~this();
     }
alias ПроверОшиб AssertError;


class ГранМасИскл : ОтслежИскл
     {
         this( ткст файл = __FILE__, т_мера строка = __LINE__ );
     }
alias ГранМасИскл ArrayBoundsException;


class ГранМасОшиб : Искл
     {
         this(ткст имяф =__FILE__, т_мера номстроки = __LINE__);
     }
alias ГранМасОшиб ArrayBoundsError;


class ФинализИскл : ОтслежИскл
     {
         this( ИнфОКлассе с, Исключение е = пусто );
         override ткст toString();
     }
alias ФинализИскл FinalizeException;


class ЩитИскл : ОтслежИскл
     {
         this( ткст файл =__FILE__, т_мера строка =__LINE__ );
     }
alias ЩитИскл SwitchException;


class ЩитОшиб : Искл
     {
     this(ткст имяф = __FILE__, бцел номстроки =__LINE__);
         override void print();
     }
alias ЩитОшиб SwitchError;


class ТекстИскл : ОтслежИскл
     {
         this( ткст сооб );
     }
```

```
alias ТекстИскл TextException;


class ЮникодИскл : ТекстИскл
    {
        this( ткст сооб, т_мера idx );
    }
alias ЮникодИскл UnicodeException;


class НитьИскл : ПлатформИскл
    {
        this( ткст сооб );
    }
alias НитьИскл ThreadException;


class ФибраИскл : ThreadException
    {
        this( ткст сооб );
    }
alias ФибраИскл FiberException;


class СинхИскл : ПлатформИскл
    {
        this( ткст сооб );
    }
alias СинхИскл SyncException;


class ВВИскл : ПлатформИскл
    {
        this( ткст сооб );
    }
alias ВВИскл IOException;


class ВфсИскл : IOException
    {
        this( ткст сооб );
    }
alias ВфсИскл VfsException;


class КластерИскл : IOException
    {
        this( ткст сооб );
    }
alias КластерИскл ClusterException;


class СокетИскл : IOException
    {
        this( ткст сооб, цел ош = 0 );
    }
alias СокетИскл SocketException;


class ХостИскл : IOException
    {
        this( ткст сооб, цел ош = 0 );
    }
alias ХостИскл HostException;
```

```
class АдрИскл : IOException
    {
        this( ткст сооб, цел ош = 0 );
    }
alias АдрИскл AddressException;


class СокетПриёмИскл : СокетИскл
    {
        this( ткст сооб );
    }
alias СокетПриёмИскл SocketAcceptException;


class ПроцессИскл : ПлатформИскл
    {
        this( ткст сооб );
    }
alias ПроцессИскл ProcessException;


class РегВырИскл : TextException
    {
        this( ткст сооб );
    }
alias РегВырИскл RegexException;


class ИсклЛокали : TextException
    {
        this( ткст сооб );
    }
alias ИсклЛокали LocaleException;


class ИсклРеестра : ОтслежИскл
    {
        this( ткст сооб );
    }
alias ИсклРеестра RegistryException;


class НевернАргИскл : ОтслежИскл
    {
        this( ткст сооб );
    }
alias НевернАргИскл IllegalArgumentException, ИсклНелегальногоАргумента;


class НевернЭлемИскл : НевернАргИскл
    {
        this( ткст сооб );
    }
alias НевернЭлемИскл IllegalElementException;


class НетЭлементаИскл : ОтслежИскл
{
this( ткст сооб );
}
alias НетЭлементаИскл NoSuchElementException;
```

```
class ИсклПовреждённыйИтератор: NoSuchElementException
{
this( ткст сооб );
}
alias ИсклПовреждённыйИтератор CorruptedIteratorException;

class ФинализОшиб : Исключение
     {
          this( ИнфОКлассе c, Исключение e = пусто );
          override string toString();
     }
alias ФинализОшиб FinalizeError;


class ДиапазонИскл : Исключение
{
this( string файл, т_мера строка );
}


class СкрытФункцИскл : Исключение
{
this( ИнфОКлассе ci );
}

class ИсклРЯР : TextException
{
this(ткст сооб);
}

////////////////////////
class АргИскл : Исключение
{
this();
this(ткст сооб);
this(ткст сооб, ткст парамИмя) ;
final ткст парамИмя() ;
}

class ПустойАргИскл : АргИскл
{
this() ;
this(ткст парамИмя);
this(ткст парамИмя, ткст сооб) ;
}

class АргВнеИскл : АргИскл
{
this();
this(ткст парамИмя) ;
this(ткст парамИмя, ткст сооб);
}

class ФорматИскл : Исключение
{
this() ;
this(ткст сооб);
}

class КастИскл : Исключение
{
this() ;
this(ткст сооб);
```

```
}

class ОпИскл : Исключение
{
this();
this(ткст сооб) ;
}

class НереализИскл : Исключение
{
this();
this(ткст сооб) ;
}

class НеПоддерживаетсяИскл : Исключение
{
this() ;
this(ткст сооб);
}

class НулСсылкаИскл : Исключение
{
this() ;
this(ткст сооб);
}

class ВзломИскл : Исключение
{
this() ;
this(ткст сооб) ;
}

class БезопИскл : Исключение
{
this() ;
this(ткст сооб);
}

class МатИскл : Исключение
{
this() ;
this(ткст сооб) ;
}

class ПереполнИскл : МатИскл
{
this() ;
this(ткст сооб) ;
}
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
17:32

```
module gc;
import dinrus;

const бцел ВЕРСИЯ_СМ = 1;
```

```
alias проц (*ФИНАЛИЗАТОР_СМ)(ук р, бул dummy);
alias ФИНАЛИЗАТОР_СМ GC_FINALIZER;
////////////////////////////////////////////////
/**
* Данная структура инкапсулирует в себе функциональность сборщика мусора
* языка программирования Динрус.
*/
extern (D) class СборщикМусора
{
private т_см экз;

бцел версия();

this()
{
        this = смНовый();
        this.иниц();
}

~this()
{
смУдали(this);
}
проц иниц();
    проц Дтор();
    проц монитор (проц delegate() начало, проц delegate(цел, цел) конец);
проц вкл();
проц откл();
проц собери();//!!!
    проц уменьши();
бцел дайАтр( ук р );
бцел устАтр( ук р, ПАтрБлока а );
бцел удалиАтр( ук р, ПАтрБлока а );
ук празмести( т_мера разм, бцел ba = 0, т_мера *alloc_size = null );
ук кразмести( т_мера разм, бцел ba = 0, т_мера *alloc_size = null );
ук перемести( ук р, т_мера разм, бцел ba = 0, т_мера *alloc_size = null);
т_мера расширь(ук р, т_мера минразм, т_мера максразм);
т_мера резервируй( т_мера разм );
проц освободи( ук р );
ук адрес_у( ук р );
т_мера размер_у( ук р );
ИнфОБл опроси( ук р );
    проц проверь(ук р);
проц добавьКорень( ук р );
    цел delegate(цел delegate(ref ук)) обходКорня();
проц добавьПространство( ук р, т_мера разм );
    проц добавьПространство(ук Низ, ук Верх);
    цел delegate(цел delegate(ref Пространство)) обходПространства();
проц удалиКорень( ук р );
проц удалиПространство( ук р );
проц мониторируй( проц delegate() начало, проц delegate(цел, цел) конец );
ук создайСлабУк( Объект о );
проц удалиСлабУк( ук р );
Объект дайСлабУк( ук р );
т_мера нарастиДлину (т_мера newlength, т_мера elSize=1);
т_мера нарастиДлину(т_мера newlength, т_мера elSize, т_мера a, т_мера b=0,
т_мера minBits=1);
    проц полныйСбор();
    проц полныйСборБезСтэка();
    проц генСбор();
    проц естьУказатели(ук р);
    проц нетУказателей(ук р);
    проц устВ1_0();
    т_мера ёмкость(ук р);
```

```
    проц сканируйСтатДан(т_см g);
    проц отсканируйСтатДан(т_см g);
    проц экономь();
    проц дайСтат(out СМСтат стат);
    проц устФинализатор(ук p, ФИНАЛИЗАТОР_СМ pFn);
}
alias СборщикМусора СМ, т_см, gc_t;

extern (C):

//Закомментированнные импорты указаны в модуле base,
//хотя и относятся к сборщику мусора. base - это базовый модуль,
//поэтому его импортом будет импортироваться весь рантайм Динрус,
//без необходимости подключать данный модуль или какие-то ещё, например,
thread.

/+
бул смПроверь(ук p);
бул смУменьши();
бул смДобавьКорень( ук p );
бул смДобавьПространство( ук p, т_мера разм );
бул смДобавьПространство2( ук p, ук разм );
бул смУдалиКорень( ук p );
бул смУдалиПространство( ук p );
т_мера смЁмкость(ук p);
бул смМонитор(ddel начало, dint конец );
бул смСтат();
СМСтат смДайСтат();
проц[] смПразместиМас(т_мера члобайт);
проц[] смПереместиМас(ук p, т_мера члобайт);
бул устИнфОТипе(ИнфОТипе иот, ук p);
ук дайУкНаСМ();
бул укНаСМ(ук p);
бул сбросьУкНаСМ();
бцел смДайАтр( ук p );
бцел смУстАтр( ук p, ПАтрБлока a );
бцел смУдалиАтр( ук p, ПАтрБлока a );
ук смПразмести( т_мера разм, бцел ba = 0 );
ук смКразмести( т_мера разм, бцел ba = 0 );
ук смПеремести( ук p, т_мера разм, бцел ba = 0 );
т_мера смРасширь( ук p, т_мера mx, т_мера разм );
т_мера смРезервируй( т_мера разм );
бул смОсвободи( ук p );
ук смАдрес( ук p );
т_мера смРазмер( ук p );
ук смСоздайСлабУк( Объект r );
бул смУдалиСлабУк( ук wp );
Объект смДайСлабУк( ук wp );
ИнфОБл смОпроси( ук p );
бул смВключи();
бул смОтключи();
бул смСобери();+/
т_см смНовый();
проц смУдали(т_см см);
/+бул смИниц_ли();
//цел смОбходКорня();
//цел смОбходПространства();

проц setFinalizer(ук p, GC_FINALIZER pFn);

void setTypeInfo(TypeInfo ti, void* p);
void* getGCHandle();
void setGCHandle(void* p);
void endGCHandle();
```

```
void gc_init();
void gc_term();
size_t gc_capacity(void* p);
void gc_minimize();
void gc_addRoot( void* p );
void gc_addRange( void* p, size_t разм );
void gc_removeRoot( void* p );
void gc_removeRange( void* p );
void gc_monitor(ddel begin, dint end );+/
void gc_printStats(gc_t gc);
/+GCStats gc_stats();
void _d_gc_addrange(void *pbot, void *ptop);
void _d_gc_removerange(void *pbot);
uint gc_getAttr( void* p );
uint gc_setAttr( void* p, uint a );
uint gc_clrAttr( void* p, uint a );
void* gc_malloc( size_t разм, uint ba = 0 );
void* gc_calloc( size_t разм, uint ba = 0 );
void* gc_realloc( void* p, size_t разм, uint ba = 0 );
size_t gc_extend( void* p, size_t mx, size_t разм );
size_t gc_reserve( size_t разм );
void gc_free( void* p );
void* gc_addrOf( void* p );
size_t gc_sizeOf( void* p );
void* gc_weakpointerCreate( Object r );
void gc_weakpointerDestroy( void* wp );
Object gc_weakpointerGet( void* wp );
BlkInfo gc_query( void* p );
void gc_enable();
void gc_disable();
void gc_collect();
void gc_check(void *p);
void gc_addRangeOld( ук p, ук разм );
+/
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
17:34

```
module object;
public import base;


extern (D) class Object

{
проц dispose();
проц вымести();
проц print();
проц выведи();

ткст toString();
     ткст вТкст();

hash_t toHash();
     т_хэш вХэш();
```

```
     int opCmp(Object o);
     int opEquals(Object o) ;
     interface Monitor
{
проц lock();        alias lock блокируй;
проц unlock();        alias unlock разблокируй;
}
     alias Monitor Монитор;

     final проц notifyRegister(проц delegate(Object) дг);
     final проц уведомиРег(проц delegate(Объект) дг);

     final проц notifyUnRegister(проц delegate(Object) дг);
     final проц уведомиОтрег(проц delegate(Объект) дг);

static Object factory(ткст classname);
     static Объект фабрика(ткст имякласса);
     }
alias Object Объект;
alias Object.Monitor IMonitor, ИМонитор;

ИнфОКлассе дайИоК(Объект о){return о.classinfo ;}


//ИнфОКлассе дайИоК(Объект о){return о.classinfo;}

////////////////////////////////////////////////////////////////////////
/**
* Все невосстановимые исключения должны происходить от класса Ошибка.
*/
extern (D) class Exception : Object
{


     ткст msg; alias msg сооб;
ткст file; alias file файл;
size_t line;     alias line строка;
TraceInfo info;    alias info инфо;
Exception next;    alias next следщ;
     struct FrameInfo
     {
     long line;        alias line строка;
size_t iframe;        alias iframe икадр;
ptrdiff_t offsetSymb;        alias offsetSymb симвСмещ;
size_t baseSymb;        alias baseSymb симвОвы;
ptrdiff_t offsetImg;        alias offsetImg обрСмещ;
size_t baseImg;        alias baseImg обрОвы;
size_t address;        alias address адрес;
ткст file;        alias file файл;
ткст func;        alias func функц;
ткст extra;        alias extra экстра;
bool exactAddress;        alias exactAddress точныйАдрес;
bool internalFunction;        alias internalFunction внутрФункция;
alias проц function(FrameInfo*,проц delegate(char[])) FramePrintHandler,
ОбработчикПечатиКадра;
         static FramePrintHandler defaultFramePrintingFunction;
         alias defaultFramePrintingFunction дефФцияПечатиКадра;
         проц writeOut(проц delegate(char[])sink);
         проц выпиши(проц delegate(ткст) синк);
проц clear();
         проц сотри();
         }
     alias FrameInfo ИнфОКадре;//
     interface TraceInfo
```

```
{
int opApply( int delegate( ref FrameInfo fInfo ) );
проц writeOut(проц delegate(char[])sink);
     alias writeOut выпиши;
}
     alias TraceInfo ИнфОСледе;//
     this( ткст сооб, ткст file, long line, Exception next, TraceInfo info );
this( ткст сооб, Exception next=null );
this( ткст сооб, ткст file, long line, Exception next=null );
     override проц print();
     override проц выведи();
     override ткст toString();
     override ткст вТкст();
     /+
проц writeOutMsg(проц delegate(char[])sink);
     проц выпишиСооб(проц delegate(ткст) синк);
     проц writeOut(проц delegate(char[])sink);
     проц выпиши(проц delegate(ткст) синк);
     +/
     проц сбрось();
}
alias Exception Исключение, Искл, Ошибка, Ош;
////////////////////////////////////////////////////

alias Исключение.ИнфОСледе function( ук укз = пусто ) TraceHandler, Следопыт;

private Следопыт следопыт = пусто;
////////////////////////////////////////////////////////
/+
extern (D) class Error : Exception
{
Error next; alias next следщ;
     ткст msg; alias msg сооб;
     override проц print();
     override проц выведи();
     override ткст toString();
     override ткст вТкст();
/**
* Конструктор; сооб - сообщение, описывающее исключение.
*/
     this(ткст сооб);
this(ткст сооб, Error next);
}

alias Error Ошибка, Ош;
////////////////////////////////////////////////////////////////////////
+/
alias проц delegate(Object) DEvent, ДСобыт;

extern (D) struct Monitor
{
проц delegate(Object)[] delegates;
     extern(C) extern IMonitor impl;
extern(C) extern ДСобыт[] devt;
}
alias Monitor Монитор;

/**********************
* Информация о каждом модуле.
*/
alias ModuleInfo ИнфОМодуле;
extern(D) class ModuleInfo
{
extern(C) extern char name[];
```

```
extern(C) extern ИнфОМодуле importedModules[];
extern(C) extern ИнфОКлассе localClasses[];

extern(C) extern бцел flags;        // initialization state

проц function() ctor; // module static constructor (order dependent)
проц function() dtor; // module static destructor
проц function() unitTest;
    /*проц (*ctor)();    // module static constructor (order dependent)
    проц (*dtor)();    // module static destructor
проц (*unitTest)();    // module unit tests*/

extern(C) extern ук xgetMembers;    // module getMembers() function

проц function() ictor;//проц (*ictor)();    // module static constructor
(order independent)

    static int opApply( int delegate( ref ModuleInfo ) дг );
/******************
* Возвращает коллекцию всех модулей в программе.
*/
static ИнфОМодуле[] модули();
}

extern(D) class ОшКтораМодуля : Исключение
{
this(ИнфОМодуле m);
}

//////////////////

extern (C) struct Interface
{
extern(C) extern ИнфОКлассе classinfo;     alias classinfo классинфо;
extern(C) extern ук [] vtbl;    alias vtbl вирттаб;
extern(C) extern цел offset;    alias offset смещение;

}
alias Interface Интерфейс;
////////////////////////////////////////////////////////////////////////

alias ClassInfo ИнфОКлассе;
extern (D) class ClassInfo
{

extern(C) extern byte[] init;    alias init иниц;
    byte[] getSetInit(byte[] init = null);
    байт[] дайУстИниц(байт[] иниц = пусто);
    extern(C) extern ткст name;    alias name имя;
    ткст getSetName(ткст name = null);
    ткст дайУстИмя(ткст имя = пусто);

extern(C) extern ук [] vtbl;    alias vtbl вирттаб;
    ук[] getSetVtbl(ук[] vtbl = null);
    ук[] дайУстВирттаб(ук[] вирттаб = пусто);

extern(C) extern Interface[] interfaces;    alias interfaces интерфейсы;
    Interface[] getSetInterfaces(Interface[] interfaces = null);
    Интерфейс[] дайУстИнтерфейсы(Интерфейс[] интерфейсы = пусто);

extern(C) extern ClassInfo base;    alias base основа;
    ИнфОКлассе getSetBase(ИнфОКлассе base = null);
    ИнфОКлассе дайУстОву(ИнфОКлассе основа = пусто);
```

```
extern(C) extern ук destructor;    alias destructor деструктор;
     ук getSetDestructor(ук destructor = null);
     ук дайУстДестр(ук деструктор = пусто);

проц (*classInvariant)(Object);

extern(C) extern бцел flags;    alias flags флаги;
     //    1:            // ИИнкогнито (IUnknown)
//    2:            // нет возможных указателей на память СМ
//    4:            // есть члены offTi[]
//    8:            // есть конструкторы
//    32:            // есть инфотипе
     бцел getSetFlags(бцел flags = бцел.init);

extern(C) extern ук deallocator;    alias deallocator выместитель;
     ук getSetDeallocator(ук deallocator = null);
     ук дайУстДеаллок(ук выместитель = пусто);

extern(C) extern OffsetTypeInfo[] offTi;    alias offTi смТи;
     OffsetTypeInfo[] getSetOffTi(OffsetTypeInfo[] offTi = null);
     OffsetTypeInfo[] дайУстСмТи(ИнфОТипеИСмещ[] смТи = пусто);

проц function(Object) defaultConstructor;

extern(C) extern TypeInfo typeinfo; alias typeinfo инфотипе;
     ИнфОТипе getSetTypeinfo(ИнфОТипе typeinfo = null);
     ИнфОТипе дайУстИнфОТипе(ИнфОТипе инфотипе = пусто);

static ClassInfo find(ткст classname);
     static ИнфОКлассе найди (ткст имякласса);

     Object create();
     Объект создай();
}

////////////////////////////////////////////////////////////////////////////
//////
extern (C) struct OffsetTypeInfo
{
extern(C) extern size_t offset;    alias offset смещение;
extern(C) extern TypeInfo ti;    alias ti иот;
}
alias OffsetTypeInfo ИнфОТипеИСмещ;

////////////////////////////////////////////////////////////////////////////

alias TypeInfo ИнфОТипе;
extern (D) class TypeInfo
{
hash_t toHash();
     т_хэш вХэш();

override int opCmp(Object o);
override int opEquals(Object o);

hash_t getHash(in ук p);
     т_хэш дайХэш(in ук п);
     int equals(in ук p1, in ук p2) ;
     цел равны(in ук п1, in ук п2);

int compare(in ук p1, in ук p2) ;
     цел сравни(in ук п1, in ук п2);
     size_t tsize();
     т_мера тразм();
```

```
     проц swap(ук p1, ук p2);
     проц поменяй(ук п1, ук п2);

TypeInfo next();
     ИнфОТипе следщ();
     проц[] init();
     проц[] иниц();
бцел flags();
     бцел флаги();
OffsetTypeInfo[] offTi();
     ИнфОТипеИСмещ[] смТи();
          }
////////////////////////////////////////////////////////////
extern (D) class TypeInfo_Typedef : ИнфОТипе
{
override ткст toString();
     override ткст вТкст();

override int opEquals(Object o);

override hash_t getHash(in ук p) ;
     override т_хэш дайХэш(in ук п);
     override int equals(in ук p1, in ук p2) ;
     override цел равны(in ук п1, in ук п2);
     override int compare(in ук p1, in ук p2) ;
     override цел сравни(in ук п1, in ук п2);
     override size_t tsize();
     override т_мера тразм();
     override проц swap(ук p1, ук p2) ;
     override проц поменяй( ук п1, ук п2);
     override ИнфОТипе next() ;
     override ИнфОТипе следщ();
     override бцел flags() ;
override бцел флаги();
override проц[] init() ;
     override проц[] иниц();

extern(C) extern ИнфОТипе base;    alias base основа;
     ИнфОТипе getSetBase(ИнфОТипе base = null);
     ИнфОТипе дайУстОву(ИнфОТипе основа = пусто);
     extern(C) extern ткст name;    alias name имя;
     ткст getSetName(ткст name = null);
     ткст дайУстИмя(ткст имя = пусто);
     extern(C) extern проц[] m_init;
}
alias TypeInfo_Typedef ТипТипдеф;
////////////////////////////////////////////

extern (D) class TypeInfo_Enum : TypeInfo_Typedef
{


}
alias TypeInfo_Enum ТипПеречень;
////////////////////////////////////////////

extern (D) class TypeInfo_Pointer : ИнфОТипе
{
override ткст toString() ;
     override ткст вТкст();
     override int opEquals(Object o);

hash_t getHash(ук p);
     т_хэш дайХэш(ук п);
```

```
int equals(ук p1, ук p2);
     цел равны(ук п1, ук п2);
     int compare(ук p1, ук p2);
     цел сравни(ук п1, ук п2);

override size_t tsize();
     override т_мера тразм();

override проц swap(ук p1, ук p2);
     override проц поменяй( ук п1, ук п2);

override ИнфОТипе next();
     override ИнфОТипе следщ();
     override бцел flags();
     override бцел флаги();

extern(C) extern ИнфОТипе m_next;
}
alias TypeInfo_Pointer ТипУказатель;
////////////////////////////////////////////

extern (D) class TypeInfo_Array : ИнфОТипе
{
override ткст toString() ;
     override ткст вТкст();

override int opEquals(Object o);

hash_t getHash(ук p);
     override т_хэш дайХэш(ук п);

int equals(ук p1, ук p2);
     цел равны(ук п1, ук п2);

int compare(ук p1, ук p2);
     цел сравни(ук п1, ук п2);
     override size_t tsize();
     override т_мера тразм();

override проц swap(ук p1, ук p2);
     override проц поменяй( ук п1, ук п2);

extern(C) extern ИнфОТипе value;

override ИнфОТипе next();
     override ИнфОТипе следщ();

override бцел flags();
     override бцел флаги();
}
alias TypeInfo_Array ТипМассив;

////////////////////////////////////////////////////////////////////////////

extern (D) class TypeInfo_StaticArray : ИнфОТипе
{
override ткст toString();
     override ткст вТкст();

override int opEquals(Object o);

     override hash_t getHash(in ук p);
     override т_хэш дайХэш(in ук п);
```

```
    override int equals(in ук p1, in ук p2);
    override цел равны(in ук п1, in ук п2);

override int compare(in ук p1, in ук p2);
    override цел сравни(in ук п1, in ук п2);

override size_t tsize();
    override т_мера тразм();

override проц swap(ук p1, ук p2);
    override проц поменяй( ук п1, ук п2);

override проц[] init() ;
    override проц[] иниц();
    override ИнфОТипе next() ;
    override ИнфОТипе следщ();
    override бцел flags();
    override бцел флаги();

extern(C) extern ИнфОТипе value;    alias value значение;
    ИнфОТипе getSetValue(ИнфОТипе value = null);
    ИнфОТипе дайУстЗначение(ИнфОТипе значение = пусто);
    extern(C) extern size_t len;    alias len длин;
    т_мера getSetLength(т_мера len = т_мера.init);
    т_мера дайУстДлину(т_мера длин = т_мера.init);
}
alias TypeInfo_StaticArray ТипСтатМас;

////////////////////////////////////////////////////////////////////////////
///////////////
extern (D) class TypeInfo_AssociativeArray : ИнфОТипе
{

override ткст toString();
    override ткст вТкст();

override /*int*/ int opEquals(Object o);

override hash_t getHash(in ук p);
    override т_хэш дайХэш(in ук п);

override size_t tsize();
    override т_мера тразм();
    override int equals(in ук p1, in ук p2);
    override цел равны(in ук п1, in ук п2);

override int compare(in ук p1, in ук p2);
    override цел сравни(in ук п1, in ук п2);

override ИнфОТипе next() ;
override ИнфОТипе следщ();
override бцел flags() ;
    override бцел флаги();

extern(C) extern ИнфОТипе value;    alias value значение;
    ИнфОТипе getSetValue(ИнфОТипе value = null);
    ИнфОТипе дайУстЗначение(ИнфОТипе значение = пусто);
    extern(C) extern ИнфОТипе key;    alias key ключ;
    ИнфОТипе getSetKey(ИнфОТипе key = null);
    ИнфОТипе дайУстКлюч(ИнфОТипе ключ = пусто);
}
alias TypeInfo_AssociativeArray ТипАссоцМас;
////////////////////////////////////////////////////////////////////////////
///////////////
```

```
extern (D) class TypeInfo_Function : ИнфОТипе
{
override ткст toString();
     override ткст вТкст();

override int opEquals(Object o);
override size_t tsize();
     override т_мера тразм();

extern(C) extern ИнфОТипе next;    alias next следщ;
     ИнфОТипе getSetNext(ИнфОТипе next = null);
     ИнфОТипе дайУстСледщ(ИнфОТипе следщ = null);
     }
alias TypeInfo_Function ТипФункция;
////////////////////////////////////////////////////////////////////////////
////////

extern (D) class TypeInfo_Delegate : ИнфОТипе
{
override ткст toString();
     override ткст вТкст();

override int opEquals(Object o);

override size_t tsize();
     override т_мера тразм();

override бцел flags();
     override бцел флаги();

extern(C) extern ИнфОТипе next;    alias next следщ;
     ИнфОТипе getSetNext(ИнфОТипе next = null);
     ИнфОТипе дайУстСледщ(ИнфОТипе следщ = null);
}
alias TypeInfo_Delegate ТипДелегат;
////////////////////////////////////////////////////////////////////////////
/////////

extern (D) class TypeInfo_Class : ИнфОТипе
{
override ткст toString();
     override ткст вТкст();

override int opEquals(Object o);

hash_t getHash(ук p);
     override т_хэш дайХэш(ук п);

int equals(ук p1, ук p2);
     цел равны(ук п1, ук п2);

int compare(ук p1, ук p2);
     цел сравни(ук п1, ук п2);

override size_t tsize();
     override т_мера тразм();

override бцел flags();
     override бцел флаги();

override OffsetTypeInfo[] offTi();
     override ИнфОТипеИСмещ[] смТи();
```

```
extern(C) extern ClassInfo info;    alias info инфо;
    ИнфОКлассе getSetInfo(ИнфОКлассе info = null);
    ИнфОКлассе дайУстИнфо(ИнфОКлассе инфо = пусто);
}
alias TypeInfo_Class ТипКласс;
////////////////////////////////////////////////////////////////////////////
//////////////
extern (D) class TypeInfo_Interface : ИнфОТипе
{
override ткст toString();
    override ткст вТкст();

override int opEquals(Object o);

hash_t getHash(ук p);
    т_хэш дайХэш(ук п);

int equals(ук p1, ук p2);
    цел равны(ук п1, ук п2);

int compare(ук p1, ук p2);
    цел сравни(ук п1, ук п2);

override size_t tsize();
    override т_мера тразм();

override бцел flags();
    override бцел флаги();

extern(C) extern ClassInfo info;    alias info инфо;
    ИнфОКлассе getSetInfo(ИнфОКлассе info = null);
    ИнфОКлассе дайУстИнфо(ИнфОКлассе инфо = пусто);
}
alias TypeInfo_Interface ТипИнтерфейс;
////////////////////////////////////////////////////////////////////////////
//////

extern (D) class TypeInfo_Struct : ИнфОТипе
{
override ткст toString();
    override ткст вТкст();

override int opEquals(Object o);

hash_t getHash(ук p);
    т_хэш дайХэш(ук п);

int equals(ук p1, ук p2);
    цел равны(ук п1, ук п2);

int compare(ук p1, ук p2);
    цел сравни(ук п1, ук п2);

override size_t tsize();
    override т_мера тразм();

override проц[] init();
    override проц[] иниц();

override бцел flags();
    override бцел флаги();

extern(C) extern ткст name;    alias name имя;
    ткст getSetName(ткст name = null);
```

```
     ткст дайУстИмя(ткст имя = пусто);
     extern(C) extern проц[] m_init;
     hash_t function(проц*) xtoHash;
int function(проц*,проц*) xopEquals;
int function(проц*,проц*) xopCmp;
ткст function(проц*) xtoString;

extern(C) extern бцел m_flags;
}
alias TypeInfo_Struct ТипСтрукт;
////////////////////////////////////////////////////////////////////////////

extern (D) class TypeInfo_Tuple : ИнфОТипе
{
extern(C) extern ИнфОТипе[] elements;    alias elements элементы;
     ИнфОТипе[] getSetElements(ИнфОТипе[] elements = null);
     ИнфОТипе[] дайУстЭлементы(ИнфОТипе[] элементы = пусто);

override ткст toString();
     override ткст вТкст();

override int opEquals(Object o);

hash_t getHash(ук p);
     т_хэш дайХэш(ук п);

int equals(ук p1, ук p2);
     цел равны(ук п1, ук п2);

int compare(ук p1, ук p2);
     цел сравни(ук п1, ук п2);

override size_t tsize();
     override т_мера тразм();

override проц swap(ук p1, ук p2);
     override проц поменяй( ук п1, ук п2);
}
alias TypeInfo_Tuple ТипКортеж;
////////////////////////////////////////////////////////////////////////////
/

extern (D) class TypeInfo_Const : ИнфОТипе
{
override ткст toString() ;
     override ткст вТкст();

override int opEquals(Object o);
hash_t getHash(ук p);
     т_хэш дайХэш(ук п);
     int equals(ук p1, ук p2) ;
     цел равны(ук п1, ук п2);
     int compare(ук p1, ук p2) ;
     цел сравни(ук п1, ук п2);
     override size_t tsize() ;
     override т_мера тразм();
     override проц swap(ук p1, ук p2) ;
     override проц поменяй( ук п1, ук п2);

override ИнфОТипе next() ;
     override ИнфОТипе следщ();
     override бцел flags() ;
     override бцел флаги();
     override проц[] init();
```

```
    override проц[] иниц();

extern(C) extern ИнфОТипе base;    alias base основа;
    ИнфОТипе getSetBase(ИнфОТипе base = null);
    ИнфОТипе дайУстОву(ИнфОТипе основа = пусто);
}
alias TypeInfo_Const ТипКонстанта;
////////////////////////////////////////////////////////////////////

extern (D) class TypeInfo_Invariant : TypeInfo_Const
{

override ткст toString();
    override ткст вТкст();
}
alias TypeInfo_Invariant ТипИнвариант;
////////////////////////////////////////////////////////////////////////////

// Object[]
extern (D) class TypeInfo_AC : TypeInfo_Array
{
override т_хэш getHash(in ук p);
    override т_хэш дайХэш(in ук п);
override цел equals(in ук p1, in ук p2);
    override цел равны(in ук п1, in ук п2);
override цел compare(in ук p1, in ук p2);
    override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
    override т_мера тразм();
override бцел flags();
    override бцел флаги();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_AC ТипОбъмас;
////////////////////////////////
// кдво[]
extern (D) class TypeInfo_Ar : TypeInfo_Array
{
override ткст toString();
    override ткст вТкст();
override т_хэш getHash(in ук p);
    override т_хэш дайХэш(in ук п);
override цел equals(in ук p1, in ук p2);
    override цел равны(in ук п1, in ук п2);
override цел compare(in ук p1, in ук p2);
    override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
    override т_мера тразм();
override бцел flags();
    override бцел флаги();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Ar ТипКдвомас;
////////////////////////////////

// кплав[]
extern (D) class TypeInfo_Aq : TypeInfo_Array
{

override ткст toString();
    override ткст вТкст();
override т_хэш getHash(in ук p) ;
```

```
    override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
    override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
    override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
    override т_мера тразм();
override бцел flags();
    override бцел флаги();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Aq ТипКплавмас;
////////////////////////////////////

// креал[]
extern (D) class TypeInfo_Ac : TypeInfo_Array
{

override ткст toString();
    override ткст вТкст();
override т_хэш getHash(in ук р) ;
    override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
    override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
    override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
    override т_мера тразм();
override бцел flags();
    override бцел флаги();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Ac ТипКреалмас;
////////////////////////////////////////////

// дво[]
extern (D) class TypeInfo_Ad : TypeInfo_Array
{

override ткст toString();
    override ткст вТкст();
override т_хэш getHash(in ук р) ;
    override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
    override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
    override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
    override т_мера тразм();
override бцел flags();
    override бцел флаги();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Ad ТипДвомас;
////////////////////////////

// вдво[]
extern (D) class TypeInfo_Ap : TypeInfo_Ad
{

ткст toString();
```

```
    override ткст вТкст();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Ap ТипВдвомас;
////////////////////////////////////

// плав[]
extern (D) class TypeInfo_Af : TypeInfo_Array
{

override ткст toString();
    override ткст вТкст();
override т_хэш getHash(in ук р);
    override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
    override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
    override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
    override т_мера тразм();
override бцел flags();
    override бцел флаги();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Af ТипПлавмас;
////////////////////////////////////

// вплав[]
extern (D) class TypeInfo_Ao : TypeInfo_Af
{
override ткст toString();
    override ткст вТкст();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Ao ТипВплавмас;
////////////////////////////////////

// байт[]
extern (D) class TypeInfo_Ag : TypeInfo_Array
{

override ткст toString();
    override ткст вТкст();
override т_хэш getHash(in ук р) ;
    override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
    override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
    override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
    override т_мера тразм();
override бцел flags();
    override бцел флаги();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Ag ТипБайтмас;
////////////////////////////////////

// ббайт[]
extern (D) class TypeInfo_Ah : TypeInfo_Ag
```

```
{
override ткст toString();
     override ткст вТкст();
override цел compare(in ук p1, in ук p2);
     override цел сравни(in ук п1, in ук п2);
override ИнфОТипе next();
     override ИнфОТипе следщ();
}
alias TypeInfo_Ah ТипБбайтмас;
////////////////////////////////

// проц[]
extern (D) class TypeInfo_Av : TypeInfo_Ah
{
override ткст toString();
     override ткст вТкст();
override ИнфОТипе next();
     override ИнфОТипе следщ();
}
alias TypeInfo_Av ТипПроцмас;
////////////////////////////////

// bool[]
extern (D) class TypeInfo_Ab : TypeInfo_Ah
{
override ткст toString();
     override ткст вТкст();
override ИнфОТипе next();
     override ИнфОТипе следщ();
}
alias TypeInfo_Ab ТипБулмас;
////////////////////////////////

// ткст
extern (D) class TypeInfo_Aa : TypeInfo_Ag
{
override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук p);
     override т_хэш дайХэш(in ук п);
override ИнфОТипе next();
     override ИнфОТипе следщ();
}
alias TypeInfo_Aa ТипТкст;
////////////////////////////////

// цел[]
extern (D) class TypeInfo_Ai : TypeInfo_Array
{

override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук p);
override цел equals(in ук p1, in ук p2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук p1, in ук p2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override бцел flags();
     override бцел флаги();
override ИнфОТипе next();
     override ИнфОТипе следщ();
}
```

```
alias TypeInfo_Ai ТипЦелмас;
//////////////////////////////

// бцел[]
extern (D) class TypeInfo_Ak : TypeInfo_Ai
{
override ткст toString();
    override ткст вТкст();
override цел compare(in ук p1, in ук p2);
    override цел сравни(in ук п1, in ук п2);
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Ak ТипБцелмас;
//////////////////////////////
// юткст, дим[]
extern (D) class TypeInfo_Aw : TypeInfo_Ak
{
override ткст toString() ;
    override ткст вТкст();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Aw ТипЮткст;
////////////////////////////////////

// дол[]
extern (D) class TypeInfo_Al : TypeInfo_Array
{
override ткст toString();
    override ткст вТкст();
override т_хэш getHash(in ук p);
override цел equals(in ук p1, in ук p2);
    override цел равны(in ук п1, in ук п2);
override цел compare(in ук p1, in ук p2);
    override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
    override т_мера тразм();
override бцел flags();
    override бцел флаги();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Al ТипДолмас;
////////////////////////////////////////////

// бдол[]
extern (D) class TypeInfo_Am : TypeInfo_Al
{
override ткст toString();
    override ткст вТкст();
override цел compare(in ук p1, in ук p2);
    override цел сравни(in ук п1, in ук п2);
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Am ТипБдолмас;
////////////////////////////////////////////

// реал[]
extern (D) class TypeInfo_Ae : TypeInfo_Array
{
override ткст toString();
    override ткст вТкст();
```

```
override т_хэш getHash(in ук р);
    override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
    override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
    override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
    override т_мера тразм();
override бцел flags();
    override бцел флаги();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Ae ТипРеалмас;
////////////////////////////////////

// вреал[]
extern (D) class TypeInfo_Aj : TypeInfo_Ae
{
override ткст toString();
    override ткст вТкст();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_Aj ТипВреалмас;
////////////////////////////////////////

// крат[]
extern (D) class TypeInfo_As : TypeInfo_Array
{
override ткст toString() ;
    override ткст вТкст();
override т_хэш getHash(in ук р);
    override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
    override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
    override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
    override т_мера тразм();
override бцел flags();
    override бцел флаги();
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_As ТипКратмас;
//////////////////////////////

// бкрат[]
extern (D) class TypeInfo_At : TypeInfo_As
{
override ткст toString();
    override ткст вТкст();
override цел compare(in ук р1, in ук р2);
    override цел сравни(in ук п1, in ук п2);
override ИнфОТипе next();
    override ИнфОТипе следщ();
}
alias TypeInfo_At ТипБкратмас;
//////////////////////////////

// шткст, шим[]
extern (D) class TypeInfo_Au : TypeInfo_At
{
```

```
override ткст toString();
     override ткст вТкст();
override ИнфОТипе next();
     override ИнфОТипе следщ();
}
alias TypeInfo_Au ТипШткст;
////////////////////////////////

// байт
extern (D) class TypeInfo_g : ИнфОТипе
{
override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
}
alias TypeInfo_g ТипБайт;
////////////////////////////////////////

// Объект
extern (D) class TypeInfo_C : ИнфОТипе
{
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override бцел flags();
     override бцел флаги();
}
alias TypeInfo_C ТипОбъ;
////////////////////////////////////
// кдво
extern (D) class TypeInfo_r : ИнфОТипе
{
override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override проц[] init();
     override проц[] иниц();
}
alias TypeInfo_r ТипКдво;
////////////////////////////////////////
// кплав
```

```
extern (D) class TypeInfo_q : ИнфОТипе
{
override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override проц[] init();
     override проц[] иниц();
}
alias TypeInfo_q ТипКплав;
////////////////////////////////////////

//сим
extern (D) class TypeInfo_a : ИнфОТипе
{

override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override проц[] init();
     override проц[] иниц();
}
alias TypeInfo_a ТипСим;
////////////////////////////////////

// креал
extern (D) class TypeInfo_c : ИнфОТипе
{

override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override проц[] init();
     override проц[] иниц();
}
alias TypeInfo_c ТипКреал;
////////////////////////////////////////
```

```
// дим
extern (D) class TypeInfo_w : ИнфОТипе
{

override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override проц[] init();
     override проц[] иниц();
}
alias TypeInfo_w ТипДим;
////////////////////////////////////

// delegate
alias проц delegate(цел) дг;

extern (D) class TypeInfo_D : ИнфОТипе
{

override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override бцел flags();
     override бцел флаги();
}
alias TypeInfo_D ТипДг;
////////////////////////////

// дво
extern (D) class TypeInfo_d : ИнфОТипе
{
override ткст toString() ;
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override проц[] init();
     override проц[] иниц();
}
alias TypeInfo_d ТипДво;
//////////////////////////////

// плав
```

```
extern (D) class TypeInfo_f : ИнфОТипе
{

override ткст toString() ;
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override проц[] init();
     override проц[] иниц();
}
alias TypeInfo_f ТипПлав;
//////////////////////////////

// вдво
extern (D) class TypeInfo_p : TypeInfo_d
{

override ткст toString();
     override ткст вТкст();
}
alias TypeInfo_p ТипВдво;
//////////////////////////////
// вплав
extern (D) class TypeInfo_o : TypeInfo_f
{

override ткст toString();
     override ткст вТкст();
}
alias TypeInfo_o ТипВплав;
//////////////////////////
// цел
extern (D) class TypeInfo_i : ИнфОТипе
{

override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
}
alias TypeInfo_i ТипЦел;
////////////////////////////
// вреал
extern (D) class TypeInfo_j : TypeInfo_e
{

override ткст toString();
     override ткст вТкст();
```

```
}
alias TypeInfo_j ТипВреал;
//////////////////////////////
// дол
extern (D) class TypeInfo_l : ИнфОТипе
{

override ткст toString() ;
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
}
alias TypeInfo_l ТипДол;
//////////////////////////////////

// указатель
extern (D) class TypeInfo_P : ИнфОТипе
{

override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override бцел flags();
     override бцел флаги();
}
alias TypeInfo_P ТипУк;
//////////////////////////

// реал
extern (D) class TypeInfo_e : ИнфОТипе
{

override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override проц[] init();
     override проц[] иниц();
}
alias TypeInfo_e ТипРеал;
////////////////////////////////////
```

```
// крат
extern (D) class TypeInfo_s : ИнфОТипе
{

override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
}
alias TypeInfo_s ТипКрат;
//////////////////////////////

// ббайт
extern (D) class TypeInfo_h : ИнфОТипе
{

override ткст toString() ;
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
}
alias TypeInfo_h ТипББайт;
//////////////////////////////////////

extern (D) class TypeInfo_b : TypeInfo_h
{

override ткст toString() ;
     override ткст вТкст();
}
alias TypeInfo_b ТипБул;
//////////////////////////////////

// бцел
extern (D) class TypeInfo_k : ИнфОТипе
{

override ткст toString() ;
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
```

```
override проц swap(ук p1, ук p2);
     override проц поменяй( ук п1, ук п2);
}
alias TypeInfo_k ТипБцел;
////////////////////////////////////

// бдол
extern (D) class TypeInfo_m : ИнфОТипе
{

override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук p);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук p1, in ук p2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук p1, in ук p2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук p1, ук p2);
     override проц поменяй( ук п1, ук п2);
}
alias TypeInfo_m ТипБдол;
//////////////////////////////

//бкрат
extern (D) class TypeInfo_t : ИнфОТипе
{
override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук p);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук p1, in ук p2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук p1, in ук p2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук p1, ук p2);
     override проц поменяй( ук п1, ук п2);
}
alias TypeInfo_t ТипБкрат;
////////////////////////////////////

// проц
extern (D) class TypeInfo_v : ИнфОТипе
{

override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук p);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук p1, in ук p2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук p1, in ук p2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук p1, ук p2);
     override проц поменяй( ук п1, ук п2);
override бцел flags();
     override бцел флаги();
}
```

```
alias TypeInfo_v ТипПроц;
//////////////////////////////

//шим
extern (D) class TypeInfo_u : ИнфОТипе
{

override ткст toString();
     override ткст вТкст();
override т_хэш getHash(in ук р);
     override т_хэш дайХэш(in ук п);
override цел equals(in ук р1, in ук р2);
     override цел равны(in ук п1, in ук п2);
override цел compare(in ук р1, in ук р2);
     override цел сравни(in ук п1, in ук п2);
override т_мера tsize();
     override т_мера тразм();
override проц swap(ук р1, ук р2);
     override проц поменяй( ук п1, ук п2);
override проц[] init();
     override проц[] иниц();
}
alias TypeInfo_u ТипШим;
////////////////////////////////////////////////////
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
17:37

```
module runtime;
pragma(lib,"dinrus.lib");
private import gc, global, win;

//////////////////////////////
//Структура для инициализации рантайма Динрус
//extern(C) ИнфОМодуле[] _moduleinfo_array;
extern(C)
{
Рантайм рантайм();
ук консБуфЭкрана();
бцел идБазовогоПроцесса();
//ИНФОСТАРТА дайСтартИнфо();
бцел идПроцесса();
}

extern(C) struct Рантайм
{
alias старт opCall;
бул старт();
проц стоп();
бул интегрируй(ИнфОМодуле[] масмод);
т_см дайСборщикМусора();
ИнфОМодуле[] дайКонструкторы();
ИнфОМодуле[] дайДеструкторы();
}


////////////////////////////////////////////////////
```

```
extern (D)
{
     void setFinalizer(void *p, GC_FINALIZER pFn);
     void addRoot(void *p);
     void removeRoot(void *p);
     void addRange(void *pbot, void *ptop);
     void removeRange(void *pbot);
     void fullCollect();
     void fullCollectNoStack();
     void genCollect();
     void minimize();
     void disable();
     void enable();
     void getStats(out GCStats stats);
     void hasPointers(void* p);
     void hasNoPointers(void* p);
     void setV1_0();
     void printStats(gc_t gc);
     //void[] malloc(size_t разм, uint ba = 0);
     //void[] realloc(void* p, size_t разм, uint ba = 0);
     size_t capacity(void* p);
     size_t capacity(void[] p);
     void[] malloc(size_t nbytes);
     void[] realloc(void* p, size_t nbytes);
     size_t extend(void* p, size_t minbytes, size_t maxbytes);
     //size_t capacity(void* p);
     void new_finalizer(void *p, bool dummy);
}
////////////////////////////////////////////////////////////////////

extern(C) struct ДанныеОДлл
{
ткст имя;
экз указатель;
ткст путь;
}

/*Получить данные от базовой библиотеки (Dinrus.Base.dll)
*/
extern (C)    ДанныеОДлл данныеБазовойДлл();


         unittest
{
ДанныеОДлл д;

д = данныеБазовойДлл();
скажифнс ( "Имя: %s, Handle: %s, Путь: %s", д.имя, д.указатель, д.путь);
for (бцел i = 0; i < _модули.length; i++)
{
              ИнфОМодуле m = _модули[i];

              if (!m)
                   continue;

              //эхо("\tmodule[%d] = '%.*s'\n", i, m.name);
              эхо("Module[%d] = '%.*s', m = x%x, m.flags = x%x\n", i, m.name,
              m, m.flags);
}
}
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
17:39

```
module stdrus;

private import sys.DStructs, sys.DConsts,sys.DFuncs;
private import tpl.args, tpl.stream;
import cidrus, global;
public import exception;


extern (D) struct Процессор
{
     ткст производитель();
     ткст название()         ;
     бул поддержкаММЭкс()     ;
     бул поддержкаФЭксСР()     ;
     бул поддержкаССЕ()         ;
     бул поддержкаССЕ2()         ;
     бул поддержкаССЕ3()         ;
     бул поддержкаСССЕ3()     ;
     бул поддержкаАМД3ДНау()     ;
     бул поддержкаАМД3ДНауЭкст();
     бул поддержкаАМДММЭкс()     ;
     бул являетсяИА64()         ;
     бул являетсяАМД64()         ;
     бул поддержкаГиперПоточности();
     бцел потоковНаЦПБ()         ;
     бцел ядерНаЦПБ()         ;
     бул являетсяИнтел()         ;
     бул являетсяАМД()         ;
     бцел поколение()         ;
     бцел модель()             ;
     бцел семейство()         ;
     ткст вТкст()             ;
}

////////////////////////////////

бул число_ли(ИнфОТипе[] _arguments, спис_ва _argptr)
{
ткст s = "";
шткст ws = "";
юткст ds = "";

if (_arguments.length == 0)
return нет;

if (_arguments[0] == typeid(ткст))
return чис_ли(ва_арг!(ткст)(_argptr));
else if (_arguments[0] == typeid(шткст))
return чис_ли(вЮ8(ва_арг!(шткст)(_argptr)));
else if (_arguments[0] == typeid(юткст))
return чис_ли(вЮ8(ва_арг!(юткст)(_argptr)));
else if (_arguments[0] == typeid(реал))
return да;
else if (_arguments[0] == typeid(дво))
return да;
else if (_arguments[0] == typeid(плав))
```

```
return да;
else if (_arguments[0] == typeid(бдол))
return да;
else if (_arguments[0] == typeid(дол))
return да;
else if (_arguments[0] == typeid(бцел))
return да;
else if (_arguments[0] == typeid(цел))
return да;
else if (_arguments[0] == typeid(бкрат))
return да;
else if (_arguments[0] == typeid(крат))
return да;
else if (_arguments[0] == typeid(ббайт))
{
s.length = 1;
s[0]= ва_арг!(ббайт)(_argptr);
return чис_ли(cast(ткст)s);
}
else if (_arguments[0] == typeid(байт))
{
s.length = 1;
s[0] = ва_арг!(сим)(_argptr);
return чис_ли(cast(ткст)s);
}
else if (_arguments[0] == typeid(вреал))
return да;
else if (_arguments[0] == typeid(вдво))
return да;
else if (_arguments[0] == typeid(вплав))
return да;
else if (_arguments[0] == typeid(креал))
return да;
else if (_arguments[0] == typeid(кдво))
return да;
else if (_arguments[0] == typeid(кплав))
return да;
else if (_arguments[0] == typeid(сим))
{
s.length = 1;
s[0] = ва_арг!(сим)(_argptr);
return чис_ли(s);
}
else if (_arguments[0] == typeid(шим))
{
ws.length = 1;
ws[0] = ва_арг!(шим)(_argptr);
return чис_ли(вЮ8(ws));
}
else if (_arguments[0] == typeid(дим))
{
ds.length = 1;
ds[0] = ва_арг!(дим)(_argptr);
return чис_ли(вЮ8(ds));
}
else
return нет;
}

бул число_ли(...){ return cast(бул) число_ли(_arguments, _argptr);}


/*********************************
```

```
* Возвращает !=0, если x нормализован (не равен 0, не субнормален, не
бесконечен, не $(NAN)).
*/

/* Need one for each format because подстnormal floats might
* be converted to normal reals.
*/

цел нормален_ли(Х)(Х х)
{
alias плавТрэтс!(Х) П;

static if(real.mant_dig==106) { // doubledouble
// doubledouble is normal if the least significant part is normal.
return нормален_ли((cast(дво*)&х)[МАНТИССА_МЗЧ]);
} else {
// ridiculous DMD warning
бкрат е = cast(бкрат)(П.МАСКАВЫР & (cast(бкрат *)&х)[П.ПОЗВЫР_КРАТ]);
return (е != П.МАСКАВЫР && е!=0);
}
}
////////////////////////////////////////////////////////////////////

struct т_регсвер
{
цел рснач;
цел рскон;
}

////////////////////////////////////////////
extern (D)
{

     бул вОбразце(дим с, ткст образец);
     бул вОбразце(дим с, ткст[] образец);
     ткст вТкст(сим с);
     ткст вТкст(бул с);
     ткст вТкст(ббайт с);
     ткст вТкст(бкрат с);
     ткст вТкст(бцел с);
     ткст вТкст(бдол с);
     ткст вТкст(байт с);
     ткст вТкст(крат с);
     ткст вТкст(цел с);
     ткст вТкст(дол с);
     ткст вТкст(плав с);
     ткст вТкст(дво с);
     ткст вТкст(реал с);
     ткст вТкст(вплав с);
     ткст вТкст(вдво с);
     ткст вТкст(вреал с);
     ткст вТкст(кплав с);
     ткст вТкст(кдво с);
     ткст вТкст(креал с);
     ткст вТкст(дол знач, бцел корень);
     ткст вТкст(бдол знач, бцел корень);
     ткст вТкст(сим *с);
     ИнфОТипе простаяИнфОТипе(ПМангл m);
     Биб загрузиБиб(ткст имяб);
     Биб загрузиБиб(ткст[] именаб);
     проц выгрузиБиб(Биб биб);
     ук дайПроцИзБиб(Биб биб, ткст имяПроц);
     ткст дайТкстОшибки();
     цел генМакетИмпорта(ткст имяМ, ткст[] список);
```

```
    бул создайЛистинг(ткст имяБ);
    бул создайБибИзДлл(ткст имяБ, ткст имяД = пусто, ткст путь = пусто, ткст
    расшД = "dll");
}

extern (D) class Биб
{
ткст имя();
this(ук укз, ткст имя);
}


extern (D) struct ЖанБибгр {

проц заряжай(ткст winLibs, проц function(Биб) userLoad, ткст versionStr =
"");
проц загружай(ткст libNameString = пусто);
проц загружай(ткст[] libNames);
    alias загружай иниц, init;
    ткст строкаВерсии();
проц выгружай();
бул загружено();
ткст имяБиб();
static ~this();
}
//Зависимый Жанровый Загрузчик Библиотек
extern (D) struct ЗавЖанБибгр {

проц заряжай(ЖанБибгр* dependence, проц function(Биб) userLoad);
проц загружай();
ткст строкаВерсии();
проц выгружай();
бул загружено();
ткст имяБиб();
}

struct Вяз(Т) {
проц opCall(ткст n, Биб lib) {
*fptr = дайПроцИзБиб(lib, n);
}
ук* fptr;
}


template вяжи(Т) {
Вяз!(Т) вяжи(inout Т а) {
Вяз!(Т) рез;
рез.fptr = cast(ук*)&а;
return рез;
}
}

////////////////////////////////////////////////////
extern (D) struct ЧленАрхиваЗИП //ArchiveMember
{

бкрат версияСборки = 20;
бкрат версияИзвлечения = 20;
бкрат флаги;
бкрат методСжатия;
ФВремяДос время;
бцел цпи32;
бцел сжатыйРазмер;
бцел расжатыйРазмер;
```

```
бкрат номерДиска;
бкрат внутреннииеАтрибуты;
бцел внешниеАтрибуты;
private бцел смещение;
ткст имя;
ббайт[] экстра;
ткст комментарий;
ббайт[] сжатыеДанные;
ббайт[] расжатыеДанные;

проц выведи();
}

extern (D) class АрхивЗИП
{

extern (C) extern
{
ббайт[] данные;
бцел смещКПоследнЗаписи;

бцел номерДиска;
бцел стартПапкаДиска;
бцел члоЗаписей;
бцел всегоЗаписей;
ткст комментарий;
}
проц выведи();
this();
проц добавьЧлен(ЧленАрхиваЗИП de);
проц удалиЧлен(ЧленАрхиваЗИП de);
this(проц[] буфер);
ббайт[]расжать(ЧленАрхиваЗИП de);
}

extern(D) class СжатиеЗлиб
{
     enum
     {
          БЕЗ_СЛИВА = 0,
          СИНХ_СЛИВ = 2,
          ПОЛН_СЛИВ = 3,
          ФИНИШ = 4,
     }

     this(цел ур);
     this();
     ~this();
     проц[] сжать(проц[] буф);
     проц[] слей(цел режим = ФИНИШ);
}

extern(D) class РасжатиеЗлиб
{
     this(бцел размБуфЦели);
     this();
     ~this();
     проц[] расжать(проц[] буф);
     проц[] слей();
}

extern(D) class РегВыр
{
this(рсим[] образец, рсим[] атрибуты = пусто);
```

```
static РегВыр opCall(рсим[] образец, рсим[] атрибуты = пусто);
РегВыр ищи(рсим[] текст);
цел opApply(цел delegate(inout РегВыр) дг);
ткст сверь(т_мера n);
ткст перед();
     ткст после();
бцел члоподстр;
т_регсвер[] псовп;
рсим[] ввод;
рсим[] образец;
рсим[] флаги;
цел ошибки;
бцел атрибуты;

enum РВА
{
     глоб         = 1,
     любрег     = 2,
     многострок     = 4,
     тчксовплф     = 8,
}

проц компилируй(рсим[] образец, рсим[] атрибуты);
рсим[][] разбей(рсим[] текст);
цел найди(рсим[] текст);
рсим[][] сверь(рсим[] текст);
рсим[] замени(рсим[] текст, рсим[] формат);
рсим[][] выполни(рсим[] текст);
рсим[][] выполни();
цел проверь(рсим[] текст);
цел проверь();
цел проверь(ткст текст, цел стартиндекс);
цел чр(inout бцел ит, рсим с);
проц выведиПрограмму(ббайт[] прог);
цел пробнсвер(цел рс, цел пценд);
цел разборРегвыр();
цел разборКуска();
цел разборАтома();
рсим[] замени(рсим[] формат);
static рсим[] замени3(рсим[] формат, рсим[] ввод, т_регсвер[] псовп);
рсим[] заменитарый(рсим[] формат);
~this();
}


extern(D) struct Дата
     {
     цел год = цел.min;
цел месяц;
цел день;
цел час;
цел минута;
цел секунда;
цел мс;
цел день_недели;
цел коррекцияЧП = цел.min;
     проц разбор(ткст т);
}

     extern(D) struct ПапЗап
     {
     ткст имя;
бдол размер = ~0UL;
т_время времяСоздания = т_время_нч;
```

```
т_время времяПоследнегоДоступа = т_время_нч;
т_время времяПоследнейЗаписи = т_время_нч;
бцел атрибуты;

проц иниц(ткст путь, ПОИСК_ДАННЫХ_А *дф);
проц иниц(ткст путь, ПОИСК_ДАННЫХ *дф);
бцел папка_ли();
бцел файл_ли();
}
     extern(D) class БуферВывода
{
ббайт данные[];
бцел смещение;

invariant
{
     assert(смещение <= данные.length);
     }
     this();
     ~this();
     ббайт[] вБайты();
     проц резервируй(бцел члобайт);
проц пиши(ббайт[] байты);
проц пиши(ббайт b);
проц пиши(байт b);
проц пиши(сим c);
проц пиши(бкрат w);
проц пиши(крат s);
проц пиши(шим c);
проц пиши(бцел w);
проц пиши(цел i);
проц пиши(бдол l);
проц пиши(дол l);
проц пиши(плав f);
проц пиши(дво f);
проц пиши(реал f);
проц пиши(ткст s);
проц пиши(БуферВывода буф);
проц занули(бцел члобайт);
проц расклад(бцел мера);
проц расклад2();
проц расклад4();
ткст вТкст();
проц ввыводф(ткст формат, спис_ва арги);
проц выводф(ткст формат, ...);
проц простели(бцел индекс, бцел члобайт);
}

extern(D) struct МассивБит
{

т_мера длин;
бцел *ук;

     т_мера разм();
     т_мера длина();
     проц длина(т_мера новдлин);
бул opIndex(т_мера i);
бул opIndexAssign(бул b, т_мера i);
МассивБит дубль();
цел opApply(цел delegate(inout бул) дг);
цел opApply(цел delegate(inout т_мера, inout бул) дг);
     МассивБит реверсни();
     МассивБит сортируй();
```

```
цел opEquals(МассивБит a2);
цел opCmp(МассивБит a2);
     проц иниц(бул[] бм);
     проц иниц(проц[] в, т_мера члобит);
проц[] opCast();
МассивБит opCom();
МассивБит opAnd(МассивБит e2);
МассивБит opOr(МассивБит e2);
МассивБит opXor(МассивБит e2);
МассивБит opSub(МассивБит e2);
МассивБит opAndAssign(МассивБит e2);
МассивБит opOrAssign(МассивБит e2);
МассивБит opXorAssign(МассивБит e2);
МассивБит opSubAssign(МассивБит e2);
МассивБит opCatAssign(бул b);
МассивБит opCatAssign(МассивБит b);
МассивБит opCat(бул b);
МассивБит opCat_r(бул b);
МассивБит opCat(МассивБит b);
}

extern(D) class Модуль
{

     this(ук модуль, бул овладеть);
     this(ткст имяМодуля);
     проц закрой();
     ук дайСимвол(in ткст симв);
     ук найдиСимвол(in ткст симв);
     ук Ук();
     ткст Путь();
     ~this();
}



//Файл, Размещённый в Карте Памяти (MMFile)
extern (D) class РПФайл
{
alias длина length;

enum Режим
{
     Чтение,         /// read existing файл
     ЧтенЗапНов,     /// delete existing файл, write new файл
     ЧтенЗап,    /// read/write existing файл, create if not existing
     ЧтенКопирПриЗап, /// read/write existing файл, copy on write
     }
     static this(){};
this(ткст имяф);

this(ткст имяф, Режим режим, бдол размер, ук адрес,
               т_мера окно = 0);
     ~this();
     проц слей();
     бдол длина();
     Режим режим();
     проц[] opSlice();
     проц[] opSlice(бдол i1, бдол i2);
     ббайт opIndex(бдол i);
     ббайт opIndexAssign(ббайт значue, бдол i);

}
```

```
extern (D) class Файл: Поток
{
     static this(){};
this();
this(ук файлУк, ПФРежим режим);
this(ткст имяф, ПРежимФайла режим = cast(ПФРежим)1);
проц открой(ткст имяф, ПРежимФайла режим = cast(ПФРежим)1);
проц создай(ткст имяф);
проц создай(ткст имяф, ПРежимФайла режим);
override проц закрой();
~this();
override бдол размер() ;
т_мера читайБлок(ук буфер, т_мера размер);
т_мера пишиБлок(ук буфер, т_мера размер);
бдол сместись(дол смещение, ППозКурсора rel);
override т_мера доступно();
ук хэндл();
}

extern (D) class ФильтрПоток : Поток
{
     extern(C) extern
     {
     Поток п; // source stream
     бул закрытьГнездо;
     }
     бул закрытьИсток();
проц закрытьИсток(бул б);
     static this(){};
this(Поток исток);
Поток исток();
проц исток(Поток s);
проц сбросьИсток() ;
т_мера читайБлок(ук буфер, т_мера размер);
т_мера пишиБлок(ук буфер, т_мера размер) ;
override проц закрой();
бдол сместись(дол смещение, ППозКурсора откуда) ;
override т_мера доступно ();
override проц слей() ;
~this();
}

extern (D) class БуфПоток : ФильтрПоток
{
extern(C) extern
{
ббайт[] буфер;
     бцел текБуфПоз;
     бцел длинаБуф;
     бул черновойБуф;
     бцел позИстокаБуф;
     бдол позПотока;
}
          static this(){};
     проц устБуфер(ббайт[] буф);//setter
     ббайт[] дайБуфер();    //getter
     проц устТекБуфПоз(бцел тбп);
     бцел дайТекБуфПоз();
     проц устДлинуБуф(бцел дб);
     бцел дайДлинуБуф();
     проц устЧерновой(бул чб);
     бул дайЧерновойБуф();
     проц устПозИстокаБуф(бцел пиб);
     бцел дайПозИстокаБуф();
```

```
    проц устПозПотока(бдол пп);
    бдол дайПозПотока();
    const бцел дефРазмБуфера = 8192;

this(Поток исток, бцел размБуф = дефРазмБуфера);
override проц сбросьИсток();
override т_мера читайБлок(ук результат, т_мера длин);
override т_мера пишиБлок(ук результат, т_мера длин) ;
override бдол сместись(дол смещение, ППозКурсора откуда);
override ткст читайСтр(ткст буфввода);
override шткст читайСтрШ(шткст буфввода) ;
override проц слей();
override бул кф();
override бдол размер() ;
override т_мера доступно() ;
~this();
}

extern (D) class БуфФайл: БуфПоток {

    static this(){};
this();
this(ткст имяф, ПРежимФайла режим = cast(ПФРежим) 1,
бцел размБуф = дефРазмБуфера);
this(Файл файл, бцел размБуф = дефРазмБуфера);
this(ук файлУк, ПРежимФайла режим, бцел размбуфа);
проц открой(ткст имяф, ПРежимФайла режим = cast(ПФРежим) 1);
проц создай(ткст имяф, ПРежимФайла режим = cast(ПФРежим) 6) ;
override проц удали(ткст фимя);
override проц закрой();
~this();
}
extern(D) БуфФайл объБуфФайл();


extern (D) class ПотокЭндианец : ФильтрПоток {

Эндиан эндиан;
    static this(){};
this(Поток исток, Эндиан end = _эндиан);
проц устЭндиан(Эндиан э);
проц выведиЭндиан();
цел читайМПБ(цел размВозврСим = 1);
проц фиксируйПБ(ук буфер, бцел размер);
проц фиксируйБлокПБ(ук буфер, бцел размер, т_мера повтор);
проц читай(out байт х) ;
проц читай(out ббайт х);
проц читай(out крат х);
проц читай(out бкрат х) ;
проц читай(out цел х) ;
проц читай(out бцел х) ;
проц читай(out дол х) ;
проц читай(out бдол х) ;
проц читай(out плав х) ;
проц читай(out дво х) ;
проц читай(out реал х) ;
проц читай(out вплав х) ;
проц читай(out вдво х) ;
проц читай(out вреал х) ;
проц читай(out кплав х) ;
проц читай(out кдво х) ;
проц читай(out креал х) ;
проц читай(out шим х) ;
проц читай(out дим х) ;
```

```
шим бериш();
шткст читайТкстШ(т_мера длина) ;
проц пишиМПБ(МПБ b);
проц пиши(байт x);
проц пиши(ббайт x);
проц пиши(крат x) ;
проц пиши(бкрат x) ;
проц пиши(цел x) ;
проц пиши(бцел x) ;
проц пиши(дол x) ;
проц пиши(бдол x) ;
проц пиши(плав x) ;
проц пиши(дво x) ;
проц пиши(реал x) ;
проц пиши(вплав x) ;
проц пиши(вдво x) ;
проц пиши(вреал x) ;
проц пиши(кплав x);
проц пиши(кдво x);
проц пиши(креал x);
проц пиши(шим x) ;
проц пиши(дим x) ;
проц пишиТкстШ(шткст str);
бул кф();
бдол размер() ;
~this();
}

extern (D) class ПотокПамяти : ТПотокМассив!(ббайт[])
{
    static this(){};
this(ббайт[] буф = пусто) ;
this(байт[] буф);
this(ткст буф) ;
проц резервируй(т_мера count);
override т_мера пишиБлок(ук буфер, т_мера размер);
override т_мера читайБлок(ук буфер, т_мера размер);
override бдол сместись(дол смещение, ППозКурсора rel);
override т_мера доступно ();
override ббайт[] данные();
ткст вТкст();
~this();

}

extern (D) class РПФайлПоток : ТПотокМассив!(РПФайл)
{
    static this(){};
this(РПФайл файл) ;
override проц слей() ;
override проц закрой();
override т_мера пишиБлок(ук буфер, т_мера размер);
override т_мера читайБлок(ук буфер, т_мера размер);
override бдол сместись(дол смещение, ППозКурсора rel);
override т_мера доступно ();
override ббайт[] данные();
override ткст вТкст();
override проц удали(ткст фимя);
~this();
}
extern (D) class ПотокСрез : ФильтрПоток
{
extern (C) extern
{
```

```
бдол поз; // our позиция relative to low
бдол низ; // низ stream смещение.
бдол верх; // верх stream смещение.
бул ограничен; // upper-ограничен by верх.
     Поток п;
}
     static this(){};
this (Поток s, бдол нз);
this (Поток s, бдол нз, бдол вх);
override т_мера читайБлок (проц *буфер, т_мера размер);
override т_мера пишиБлок (проц *буфер, т_мера размер) ;
override бдол сместись(дол смещение, ППозКурсора rel) ;
override т_мера доступно () ;
~this();
}

extern (D) class СФайл : Поток
{
extern (C) extern    фук файлси;
     static this(){};
this(фук файлси, ПРежимФайла режим, бул сканируемый = нет);
~this();
фук файл();
проц файл(фук файлси);
override проц слей() ;
override проц закрой();
override бул кф() ;
override сим берис() ;
override сим отдайс(сим с);
override т_мера читайБлок(ук буфер, т_мера размер);
override т_мера пишиБлок(ук буфер, т_мера размер);
бдол сместись(дол смещение, ППозКурсора rel);
override проц пишиСтр(ткст s);
override проц пишиСтрШ(шткст s);
}

extern (D) class СокетПоток: Поток
{
     static this(){};
     this(Сокет сок, ПРежимФайла режим);
     this(Сокет сок);
     Сокет сокет();
     override т_мера читайБлок(ук _буфер, т_мера размер);
     override т_мера пишиБлок(ук _буфер, т_мера размер);
     бдол сместись(дол смещение, ППозКурсора куда);
     override ткст вТкст();
     override проц закрой();
}
////////////////////////////////////////////////////
extern (D) class СчётчикВысокойПроизводительности
{
     alias дол т_интервал;
     //alias PerformanceCounterScope!(СчётчикВысокойПроизводительности)
     scope_type;
     static this();
     проц старт();
     проц стоп();
     т_интервал счётПериодов();
     т_интервал секунды();
     т_интервал миллисекунды();
     т_интервал микросекунды();
}
extern (D) class СчётчикТиков
{
```

```
    alias дол т_интервал;
    //alias PerformanceCounterScope!(СчётчикТиков) scope_type;
    проц старт();
    проц стоп();
    т_интервал счётПериодов();
    т_интервал секунды();
    т_интервал миллисекунды();
    т_интервал микросекунды();
}

extern (D) class СчётчикВременИНити
{
    alias дол т_интервал;
    //alias PerformanceCounterScope!(СчётчикВременИНити) scope_type;
    this();
    проц старт();
    проц стоп();
    т_интервал счётПериодаЯдра();
    т_интервал секундыЯдра();
    т_интервал миллисекундыЯдра();
    т_интервал микросекундыЯдра();
    т_интервал счётПользовательскогоПериода();
    т_интервал секундыПользователя();
    т_интервал миллисекундыПользователя();
    т_интервал микросекундыПользователя();
    т_интервал счётПериодов()    ;
    т_интервал секунды();
    т_интервал миллисекунды()    ;
    т_интервал микросекунды()    ;
}
    extern (D) class СчётчикВременИПроцесса
{
    alias long т_интервал;
    //alias PerformanceCounterScope!(СчётчикВременИПроцесса) scope_type;
    проц старт();
    проц стоп();
    т_интервал счётПериодаЯдра();
    т_интервал секундыЯдра();
    т_интервал миллисекундыЯдра();
    т_интервал микросекундыЯдра();
    т_интервал счётПользовательскогоПериода()    ;
    т_интервал секундыПользователя()    ;
    т_интервал миллисекундыПользователя()    ;
    т_интервал микросекундыПользователя();
    т_интервал счётПериодов()    ;
    т_интервал секунды();
    т_интервал миллисекунды()    ;
    т_интервал микросекунды();
    static this();
}
    abstract class Адрес
{
    protected адрессок* имя();
    protected цел длинаИм();
    ПСемействоАдресов семействоАдресов();    /// Family of this address.
    ткст вТкст();        /// Human readable string representing this address.
}

extern (D) class Протокол
{
    ППротокол тип;
    ткст имя;
    ткст[] алиасы;
```

```
    проц заполни(протзап* прото);
    бул дайПротоколПоИмени(ткст имя);
    бул дайПротоколПоТипу(ППротокол тип);
}

extern (D) class Служба
{
    ткст имя;
    ткст[] алиасы;
    бкрат порт;
    ткст имяПротокола;

    проц заполни(служзап* служба);
    бул дайСлужбуПоИмени(ткст имя, ткст имяПротокола);
    бул дайСлужбуПоИмени(ткст имя);
    бул дайСлужбуПоПорту(бкрат порт, ткст имяПротокола);
    бул дайСлужбуПоПорту(бкрат порт);
}

extern (D) class ИнтернетХост
{
    ткст имя;
    ткст[] алиасы;
    бцел[] списокАдр;

    проц реальнаяХостзап(хостзап* хз);
    проц заполни(хостзап* хз);
    бул дайХостПоИмени(ткст имя);
    бул дайХостПоАдр(бцел адр);
    бул дайХостПоАдр(ткст адр);
}

extern (D) class НеизвестныйАдрес: Адрес
{
protected:
    override адрессок* имя();
    override цел длинаИм();
    public:
    override ПСемействоАдресов семействоАдресов();
    override ткст вТкст();
}

extern (D) class ИнтернетАдрес: Адрес
{
    protected:
    адрессок_ин иас;
    override адрессок* имя();
    override цел длинаИм();
    this();
    public:
    const бцел АДР_ЛЮБОЙ = 0;//INADDR_ANY;    /// Любое адресное число IPv4.
    const бцел АДР_НЕУК = 0xFFFFFFFF;//INADDR_NONE;    /// Любое неверное
    адресное число IPv4.
    const бкрат ПОРТ_ЛЮБОЙ = 0;    /// Любое число порта IPv4.
    override ПСемействоАдресов семействоАдресов();
    бкрат порт();
    бцел адр();
    this(ткст адр, бкрат порт);
    this(бцел адр, бкрат порт);
    this(бкрат порт);
    ткст вАдрТкст();
    ткст вПортТкст();
    override ткст вТкст();
    static бцел разбор(ткст адр);
```

```
}


extern (D) class НаборСокетов
{
     this(бцел макс);
     this();
     проц переуст();
     проц прибавь(т_сокет с);
     проц прибавь(Сокет с);
     проц удали(т_сокет с);
     проц удали(Сокет с);
     цел вНаборе(т_сокет с);
     цел вНаборе(Сокет с);
     бцел макс();
     набор_уд* вНабор_уд();
     цел выберич();
}

extern (D) class Сокет
{

this(ПСемействоАдресов са, ПТипСок тип, ППротокол протокол);
this(ПСемействоАдресов са, ПТипСок тип);
this(ПСемействоАдресов са, ПТипСок тип, ткст имяПротокола);
~this();
т_сокет Ук();
бул блокируемый();
проц блокируемый(бул б);
ПСемействоАдресов семействоАдресов() ;
бул жив_ли() ;
проц свяжи(Адрес адр);
проц подключись(Адрес к);
проц слушай(цел backlog);
Сокет принимающий();
Сокет прими();
проц экстрзак(ПЭкстрЗакрытиеСокета how);
проц закрой();
static ткст имяХоста() ;
Адрес удалённыйАдрес();
Адрес локальныйАдрес();

const цел ОШИБКА = -1;

цел шли(проц[] буф, ПФлагиСокета флаги);
цел шли(проц[] буф);
цел шли_на(проц[] буф, ПФлагиСокета флаги, Адрес куда);
цел шли_на(проц[] буф, Адрес куда);
цел шли_на(проц[] буф, ПФлагиСокета флаги);
цел шли_на(проц[] буф);
цел получи(проц[] буф, ПФлагиСокета флаги);
цел получи(проц[] буф);
цел получи_от(проц[] буф, ПФлагиСокета флаги, out Адрес от);
цел получи_от(проц[] буф, out Адрес от);
цел получи_от(проц[] буф, ПФлагиСокета флаги);
цел получи_от(проц[] буф);
цел дайОпцию(ППротокол уровень, ПОпцияСокета опция, проц[] результат);
цел дайОпцию(ППротокол уровень, ПОпцияСокета опция, out цел результат);
цел дайОпцию(ППротокол уровень, ПОпцияСокета опция, out заминка результат);
проц установиОпцию(ППротокол уровень, ПОпцияСокета опция, проц[] значение);
проц установиОпцию(ППротокол уровень, ПОпцияСокета опция, цел значение);
проц установиОпцию(ППротокол уровень, ПОпцияСокета опция, заминка значение);
static цел выбери(НаборСокетов checkRead, НаборСокетов checkWrite,
НаборСокетов checkError, значврем* tv);
```

```
static цел выбери(НаборСокетов checkRead, НаборСокетов checkWrite,
НаборСокетов checkError, цел микросекунды);
static цел выбери(НаборСокетов checkRead, НаборСокетов checkWrite,
НаборСокетов checkError);
}

extern (D) class ПутСокет: Сокет
{
     this(ПСемействоАдресов семейство);
     this();
     this(Адрес подкл_к);
}

extern (D) class ППдСокет: Сокет
{
     this(ПСемействоАдресов семейство);
     this();
}

alias ук нук;
alias бцел нид;

extern (Windows) alias бцел (*stdfp)(ук);

extern (C) нук начниНитьДоп(ук безоп, бцел размстека, stdfp стартадр, ук
списаргов, бцел иницфлаг, нид* адрнити);

extern (D) class Нить
{
this(т_мера размстека = 0);
this(цел (*fp)(ук), ук арг, т_мера размстека = 0);
this(цел delegate() дг, т_мера размстека = 0);
~this();
ук низСтэка;

проц старт();
цел пуск();
проц жди();
проц жди(бцел миллисек);
     бцел дайЧлоНитей();
     enum СН
{
     НАЧАЛЬНОЕ,     /// The thread hasn't been started yet.
     ПУЩЕНА,    /// The thread is running or paused.
     ПРЕРВАНА,     /// The thread has ended.
ЗАВЕРШЕНА /// The thread has been cleaned up
}

СН дайСостояние();

enum ПРИОРИТЕТ
{
     УВЕЛИЧЬ,     /// Increase thread priority
     УМЕНЬШИ,     /// Decrease thread priority
     НИЗКИЙ,         /// Assign thread low priority
     ВЫСОКИЙ,     /// Assign thread high priority
     НОРМАЛЬНЫЙ,
}

проц устПриор(ПРИОРИТЕТ р);
бул сама_ли();
static Нить дайЭту();
static Нить[] дайВсе();
проц пауза();
```

```
проц возобнови();
static проц паузаВсем();
static проц возобновиВсе();
static проц рви();
     extern (C) static бцел стартнити(ук р);

          public static проц пускНити();
          static ~this();
          static ук дайУкНаТекНить();

}


////////////////////////////////////////////////////////////////////////
     extern(D) struct ПерестановкаБайт
{

/*************************************************************************

Реверсирует двубайтные цепочки. Параметр приёмн указывает
число байтов, которое должно быть кратно 2

*************************************************************************/

final static проц своп16 (проц[] приёмн);
/*************************************************************************

Реверсирует четырёхбайтные цепочки. Параметр приёмн указывает
число байтов, к-е д.б. кратно 4

*************************************************************************/

final static проц своп32 (проц[] приёмн);

/*************************************************************************

Реверсирует 8-байтные цепочки. Параметр приёмн указывает
число байтов, к-е д.б. кратно 8

*************************************************************************/

final static проц своп64 (проц[] приёмн);

/*************************************************************************

Реверсирует 10-байтные цепочки. Параметр приёмн указывает
число байтов, к-е д.б. кратно 10

*************************************************************************/

final static проц своп80 (проц[] приёмн);

/*************************************************************************

Реверсирует 2-байтные цепочки. Параметр приёмн указывает
число байтов, к-е д.б. кратно 2

*************************************************************************/

final static проц своп16 (ук приёмн, бцел байты);

/*************************************************************************

Реверсирует четырёхбайтные цепочки. Параметр приёмн указывает
```

```
число байтов, к-е д.б. кратно 4

**************************************************************************/

final static проц своп32 (ук приёмн, бцел байты);

/**************************************************************************

Реверсирует 8-байтные цепочки. Параметр приёмн указывает
число байтов, к-е д.б. кратно 8

**************************************************************************/

final static проц своп64 (ук приёмн, бцел байты);

/**************************************************************************

Реверсирует 10-байтные цепочки. Параметр приёмн указывает
число байтов, к-е д.б. кратно 10

**************************************************************************/

final static проц своп80 (ук приёмн, бцел байты);
}
////////////////////////////////////////////////////////////////////////////
extern(D)
{

     бцел байтЮ(ткст т, т_мера и);
     бцел байтЮ(шткст т, т_мера и);
     бцел байтЮ(юткст т, т_мера и);
     т_мера доИндексаУНС(ткст т, т_мера и);
     т_мера доИндексаУНС(шткст т, т_мера и);
     т_мера доИндексаУНС(юткст т, т_мера и);
     т_мера вИндексЮ(ткст т, т_мера и);
     т_мера вИндексЮ(шткст т, т_мера и);
     т_мера вИндексЮ(юткст т, т_мера и);
     дим раскодируйЮ(ткст т, inout т_мера инд);
     дим раскодируйЮ(шткст т, inout т_мера инд);
     дим раскодируйЮ(юткст т, inout т_мера инд);
     проц кодируйЮ(inout ткст т, дим с);
     проц кодируйЮ(inout шткст т, дим с);
     проц кодируйЮ(inout юткст т, дим с);
     проц оцениЮ(ткст т);
     проц оцениЮ(шткст т);
     проц оцениЮ(юткст т);
     ткст вЮ8(ткст т);
     ткст вЮ8(шткст т);
     ткст вЮ8(юткст т);
     ткст вЮ8(сим[4] буф, дим с);
     шткст вЮ16(ткст т);
     шим* вЮ16н(ткст т);
     шткст вЮ16(шткст т);
     шткст вЮ16(юткст т);
     шткст вЮ16(шим[2] буф, дим с);
     юткст вЮ32(ткст т);
     юткст вЮ32(шткст т);
     юткст вЮ32(юткст т);
     проц пишиф(...);
проц пишифнс(...);
проц скажифнс(...);
проц скажиф(...);
проц пишиф_в(фук чф, ...);
проц пишифнс_в(фук чф, ...);
```

```
ткст фм(...);
alias фм форматируй;

проц форматДелай(проц delegate(дим) putc, ИнфОТипе[] arguments, спис_ва
аргук);
ткст форматируйс(ткст т, ...);

проц разборСпискаАргументов(ref ИнфОТипе[] арги, ref спис_ва аргук, out ткст
format);

ткст[] списпап(ткст имяп, РегВыр рег);
проц списпап(ткст имяп, бул delegate(ткст имяф) обрвызов);
проц списпап(ткст имяп, бул delegate(ПапЗап* пз) обрвызов);

ткст подставь(ткст текст, ткст образец, ткст delegate(РегВыр) дг, ткст
атрибуты = пусто);

РегВыр ищи(ткст текст, ткст образец, ткст атрибуты = пусто);

т_время вЦел(т_время n);

     реал абс(креал х);
     реал абс(вреал х);
     креал конъюнк(креал у);
     вреал конъюнк(вреал у);
     креал кос(креал х);
     реал кос(вреал х);
     креал син(креал х);
     вреал син(вреал х);
     креал квкор(креал х);
     цел больш_из(цел а, цел б);
     дол больш_из(дол а, дол б);
     цел больш_из(цел[] ч);
     дол больш_из(дол[] ч);
     цел меньш_из(цел а, цел б);
     дол меньш_из(дол а, дол б);
     цел меньш_из(цел[] ч);
     дол меньш_из(дол[] ч);
     цел сумма(цел[] ч);
     дол сумма(дол[] ч);
     дол квадрат(цел а);
     цел квадрат(цел а);
}
////////////////////////////////////
extern(D)
{
     ткст ДАТА(); ////alias _ДАТА ДАТА;
     ткст ВРЕМЯ(); //alias _ВРЕМЯ ВРЕМЯ;

     ук дай_низ_стека(); //alias _дай_низ_стека дай_низ_стека;

     //МД5
     проц суммаМД5(ббайт[16] дайджест, проц[] данные); //alias _суммаМД5
     суммаМД5;
     проц выведиМД5Дайджест(ббайт дайджест[16]); //alias _выведиМД5Дайджест
     выведиМД5Дайджест;
     ткст дайджестМД5вТкст(ббайт[16] дайджест); //alias _дайджестМД5вТкст
     дайджестМД5вТкст;
     //ЗИП

     //ЗЛИБ
     бцел адлер32(бцел адлер, проц[] буф); //alias _адлер32 адлер32;
     бцел цпи32(бцел кс, проц[] буф); //alias _цпи32 цпи32;
```

```
проц[] сожмиЗлиб(проц[] истбуф, цел ур = цел.init); //alias _сожмиЗлиб
сожмиЗлиб;
проц[] разожмиЗлиб(проц[] истбуф, бцел итдлин = 0u, цел винбиты = 15);
//alias _разожмиЗлиб разожмиЗлиб;

//Универсальный Идентификатор Ресурса
бцел аски8гекс(дим с); alias аски8гекс аскиВгекс;

ткст раскодируйУИР(ткст кодирУИР);
alias раскодируйУИР раскодуир;

ткст раскодируйКомпонентУИР(ткст кодирКомпонУИР);
alias раскодируйКомпонентУИР раскодкомпуир;

ткст кодируйУИР(ткст уир);
alias кодируйУИР кодуир;

ткст кодируйКомпонентУИР(ткст уирКомпон);
alias кодируйКомпонентУИР кодкомпуир;

//Юникод

бул юпроп_ли(дим с); //alias _юпроп_ли юпроп_ли;
бул юзаг_ли(дим с); //alias _юзаг_ли юзаг_ли;
дим в_юпроп(дим с); //alias _в_юпроп в_юпроп;
дим в_юзаг(дим с); //alias _в_юзаг в_юзаг;
бул юцб_ли(дим с); alias юцб_ли юцифрабукв_ли;

//утилиты
ткст текстСисОшибки(бцел кодош); //alias _текстСисОшибки текстСисОшибки;

//Динамически загружаемая библиотека
цел иницМодуль(); //alias _иницМодуль иницМодуль;
проц деиницМодуль(); //alias _деиницМодуль деиницМодуль;
ук загрузиМодуль(in ткст имямод); //alias _загрузиМодуль загрузиМодуль;
ук добавьССылНаМодуль(ук умодуль); //alias _добавьССылНаМодуль
добавьССылНаМодуль;
проц отпустиМодуль(inout ук умодуль); //alias _отпустиМодуль
отпустиМодуль, высвободиМодуль;
ук дайСимволИМодуля(inout ук умодуль, in ткст имяСимвола); //alias
_дайСимволИМодуля дайСимволМодуля;
ткст ошибкаИМодуля(); //alias _ошибкаИМодуля ошибкаМодуля;

цел пуб(бцел х); alias пуб дайПервУстБит;
цел пубр(бцел х); alias пубр найдиПервУстБит;
цел тб(in бцел *х, бцел номбит); alias тб тестируйБит;
цел тбз(бцел *х, бцел номбит); alias тбз тестируйЗаполниБит;
цел тбп(бцел *х, бцел номбит); alias тбп тестируйИзмениБит;
цел тбу(бцел *х, бцел номбит); alias тбу тестируйУстановиБит;
бцел развербит(бцел б); alias развербит разверниБайт;
ббайт чипортББ(бцел адр_порта); alias чипортББ читайПортББайт;
бкрат чипортБк(бцел адр_порта); alias чипортБк читайПортБкрат;
бкрат чипортБц(бцел адр_порта); alias чипортБц читайПортБцел;
ббайт пипортББ(бцел адр_порта, ббайт зап); alias пипортББ пишиПортББайт;
бкрат пипортБк(бцел адр_порта, бкрат зап); alias пипортБк пишиПортБкрат;
бкрат пипортБц(бцел адр_порта, бцел зап); alias пипортБц пишиПортБцел;
цел члоустбит32( бцел х ); //alias _члоустбит32 члоустбит32;
бкрат битсвоп( бцел х ); alias битсвоп переставьБит ;

бкрат кодируйДлину64(бцел сдлин); //alias _кодируйДлину64 кодируйДлину64;
ткст кодируй64(ткст стр, ткст буф = ткст.init); //alias _кодируй64
кодируй64;
бкрат раскодируйДлину64(бцел кдлин); //alias _раскодируйДлину64
раскодируйДлину64;
```

```
ткст раскодируй64(ткст кстр, ткст буф = ткст.init); //alias _раскодируй64
раскодируй64;
проц пишификс(фук фу, ИнфОТипе[] аргументы, ук аргук, цел нс = нет);
//alias _пишификс пишификс;

ткст читайстр(); //alias _читайстр читайстр, читайКонсоль;
т_мера читайстр(фук чф, inout ткст буф);
т_мера читайстр(inout ткст буф);

цел числобукв_ли(дим б); //alias _числобукв_ли числобукв_ли;
цел буква_ли(дим б); //alias _буква_ли буква_ли;
цел управ_ли(дим б); //alias _управ_ли управ_ли;
цел цифра_ли(дим б); //alias _цифра_ли цифра_ли;
цел проп_ли(дим б); //alias _проп_ли проп_ли;
цел пунктзнак_ли(дим б); //alias _пунктзнак_ли пунктзнак_ли;
цел межбукв_ли(дим б); //alias _межбукв_ли межбукв_ли;
цел заг_ли(дим б); //alias _заг_ли заг_ли;
цел цифраикс_ли(дим б); //alias _цифраикс_ли цифраикс_ли;
цел граф_ли(дим б); //alias _граф_ли граф_ли;
цел печат_ли(дим б) ; //alias _печат_ли печат_ли;
цел аски_ли(дим б); //alias _аски_ли аски_ли;
дим впроп(дим б); //alias _впроп впроп;
дим взаг(дим б); //alias _взаг взаг;
цел руспроп_ли(дим б); //alias _руспроп_ли руспроп_ли;
цел русзаг_ли(дим б); //alias _русзаг_ли русзаг_ли;

бул пробел_ли(дим s); //alias _пробел_ли пробел_ли;
дол ткствцел(ткст s); //alias _ткствцел ткствцел;
реал ткствдробь(ткст s); //alias _ткствдробь ткствдробь;
цел сравни(ткст s1, ткст s2); //alias _сравни сравни;
цел сравнлюб(ткст s1, ткст s2); //alias _сравнлюб сравнлюб;
сим* вТкст0(ткст s); //alias _вТкст0 вТкст0;
цел найди(ткст s, дим c); //alias _найди найди;
цел найдлюб(ткст s, дим c); //alias _найдлюб найдлюб;
цел найдрек(ткст s, дим c); //alias _найдрек найдрек;
цел найдлюбрек(ткст s, дим c); //alias _найдлюбрек найдлюбрек;
цел найди(ткст s, ткст подст);
цел найдлюб(ткст s, ткст подст);
цел найдрек(ткст s, ткст подст);
цел найдлюбрек(ткст s, ткст подст);
ткст впроп(ткст s);
ткст взаг(ткст s);
ткст озаг(ткст т); //alias _озаг озаг;
ткст озагслова(ткст т); //alias _озагслова озагслова;
ткст повтори(ткст т, т_мера м);
ткст объедини(ткст[] слова, ткст разд); //alias _объедини объедини;
ткст [] разбейдоп(ткст т, ткст разделитель); //alias _разбейдоп
разбейдоп;
ткст [] разбей(ткст т); //alias _разбей разбей;
ткст [] разбейнастр(ткст т); //alias _разбейнастр разбейнастр;
ткст уберислева(ткст т); //alias _уберислева уберислева;
ткст убериспрова(ткст т); //alias _убериспрова убериспрова;
ткст убери(ткст т); //alias _убери убери;
ткст убериразгр(ткст т); //alias _убериразгр убериразгр;
ткст уберигран(ткст т); //alias _уберигран уберигран;
ткст полев(ткст т, цел ширина); //alias _полев полев;
ткст поправ(ткст т, цел ширина); //alias _поправ поправ;
ткст вцентр(ткст т, цел ширина); //alias _вцентр вцентр;
ткст занули(ткст т, цел ширина); //alias _занули занули;
ткст замени(ткст т, ткст с, ткст на); //alias _замени замени;
ткст заменисрез(ткст т, ткст срез, ткст замена);
ткст вставь(ткст т, т_мера индекс, ткст подст); //alias _вставь вставь;
т_мера счесть(ткст т, ткст подст); //alias
```

```
ткст заменитабнапбел(ткст стр, цел размтаб=8); //alias _заменитабнапбел
заменитабнапбел;
ткст заменипбелнатаб(ткст стр, цел размтаб=8); //alias _заменипбелнатаб
заменипбелнатаб;
ткст постройтранстаб(ткст из, ткст в); //alias _постройтранстаб
постройтранстаб;
ткст транслируй(ткст т, ткст табтранс, ткст удсим); //alias _транслируй
транслируй;
т_мера посчитайсимв(ткст т, ткст образец); //alias _посчитайсимв
посчитайсимв;
ткст удалисимв(ткст т, ткст образец); //alias _удалисимв удалисимв;
ткст сквиз(ткст т, ткст образец= пусто); //alias _сквиз сквиз;
ткст следщ(ткст т); //alias _следщ следщ;
ткст тз(ткст ткт, ткст из, ткст в, ткст модифф= пусто); //alias _тз тз;

бул чис_ли(in ткст т, in бул раздВкл = нет); //alias _чис_ли чис_ли;
т_мера колном(ткст ткт, цел размтаб=8); //alias _колном колном;
ткст параграф(ткст т, цел колонки = 80, ткст первотступ = пусто, ткст
отступ = пусто, цел размтаб = 8); //alias _параграф параграф;
ткст эладр_ли(ткст т); //alias _эладр_ли эладр_ли;
ткст урл_ли(ткст т); //alias _урл_ли урл_ли;
ткст целВЮ8(ткст врем, бцел знач); //alias _целВЮ8 целВЮ8;
ткст бдолВЮ8(ткст врем, бцел знач); //alias _бдолВЮ8 бдолВЮ8;


цел вЦел(ткст т); //alias _вЦел вЦел;
бцел вБцел(ткст т); //alias _вБцел вБцел;
дол вДол(ткст т); //alias _вДол вДол;
бдол вБдол(ткст т); //alias _вБдол вБдол;
крат вКрат(ткст т); //alias _вКрат вКрат;
бкрат вБкрат(ткст т); //alias _вБкрат вБкрат;
байт вБайт(ткст т); //alias _вБайт вБайт;
ббайт вББайт(ткст т); //alias _вББайт вББайт;
плав вПлав(ткст т); //alias _вПлав вПлав;
дво вДво(ткст т); //alias _вДво вДво;
реал вРеал(ткст т); //alias _вРеал вРеал;

проц установиИсходнуюПапкуДляКовер(ткст путь);
проц установиПапкуЗаписиДляКовер(ткст путь);
проц установиСлияниеКовер(бул флаг);
проц регитьКовер(ткст фимя, МассивБит оу, бцел[] данные);

бул дим_ли(дим д);

проц случсей(бцел семя, бцел индекс);
бцел случайно();
бцел случген(бцел семя, бцел индекс, реал члоциклов);

проц[] читайФайл(ткст имяф); //alias _читайФайл читайФайл;
проц пишиФайл(ткст имяф, проц[] буф); //alias _пишиФайл пишиФайл;
проц допишиФайл(ткст имяф, проц[] буф); //alias _допишиФайл допишиФайл;
проц переименуйФайл(ткст из, ткст в); //alias _переименуйФайл
переименуйФайл;
проц удалиФайл(ткст имяф); //alias _удалиФайл удалиФайл;
бдол дайРазмерФайла(ткст имяф); //alias _дайРазмерФайла дайРазмФайла,
дайРазмерФайла;
проц дайВременаФайла(ткст имяф, out т_время фтц, out т_время фта, out
т_время фтм);
бул естьФайл(ткст имяф); //alias _естьФайл естьФайл;
бкрат дайАтрибутыФайла(ткст имяф); alias дайАтрибутыФайла дайАтрыФайла;
бул файл_ли(ткст имяф); //alias _файл_ли файл_ли;
бул папка_ли(ткст имяп); //alias _папка_ли папка_ли;
проц сменипап(ткст имяп);    alias сменипап перейди_в;
проц сделайпап(ткст имяп); alias сделайпап сделайПапку;
```

```
проц удалипап(ткст имяп); alias удалипап удалиПапку;
ткст дайтекпап(); alias дайтекпап дайТекущуюПапку;
ткст [] списпап(ткст имяп, ткст образец); alias списпап списокПапки;
ткст [] списпап(ткст имяп);
проц копируйФайл(ткст из, ткст в); //alias _копируйФайл копируйФайл;
сим* вМБТ_0(ткст т);
бул выведиФайл(ткст имяф);

Дата разборДаты(ткст т);

проц вГодНедИСО8601(т_время t, out цел год, out цел неделя);
цел День(т_время t);
цел високосныйГод(цел у);
цел днейВГоду(цел у);
цел деньИзГода(цел у);
т_время времяИзГода(цел у);
цел годИзВрем(т_время t);
бул високосный_ли(т_время t);
цел месяцИзВрем(т_время t);
цел датаИзВрем(т_время t);
цел часИзВрем(т_время t);
цел минИзВрем(т_время t);
цел секИзВрем(т_время t);
цел мсекИзВрем(т_время t);
цел времениВДне(т_время t);
цел ДеньНедели(т_время вр);
т_время МВ8Местное(т_время вр);
т_время местное8МВ(т_время вр);
т_время сделайВремя(т_время час, т_время мин, т_время сек, т_время мс);
т_время сделайДень(т_время год, т_время месяц, т_время дата);
т_время сделайДату(т_время день, т_время вр);

цел датаОтДняНеделиМесяца(цел год, цел месяц, цел день_недели, цел ч);
цел днейВМесяце(цел год, цел месяц);
ткст вТкст(т_время время);
ткст вТкстМВ(т_время время);
ткст вТкстДаты(т_время время);
ткст вТкстВремени(т_время время);
т_время разборВремени(ткст т);
т_время дайВремяМВ();
т_время ФВРЕМЯ8т_время(ФВРЕМЯ *фв);
т_время СИСТВРЕМЯ8т_время(СИСТВРЕМЯ *св, т_время вр);
т_время дайМестнуюЗЧП();
цел дневноеСохранениеЧО(т_время вр);
т_время вДвремя(ФВремяДос вр);
ФВремяДос вФВремяДос(т_время вр);

ткст о_ЦПУ();

ткст размангliруй(ткст имя);
ткст извлекиРасш(ткст пимя); alias извлекиРасш дайРасш;
ткст дайИмяПути(ткст пимя);

ткст извлекиИмяПути(ткст пимя);
ткст извлекиПапку(ткст пимя);

ткст извлекиМеткуДиска(ткст пимя);
ткст устДефРасш(ткст пимя, ткст расш);
ткст добРасш(ткст фимя, ткст расш);
бул абсПуть_ли(ткст путь); //alias _абсПуть_ли абсПуть_ли;
ткст слейПути(ткст п1, ткст п2); //alias _слейПути слейПути, объедини;
бул сравниПути(дим п1, дим п2);
бул сравниПутьОбразец(ткст фимя, ткст образец);
ткст развернiТильду(ткст путь);
```

```
реал абс(реал х); //alias _абс абс;
дол абс(дол х);
цел абс(цел х);
реал кос(реал х); //alias _кос кос;
реал син(реал х); //alias _син син;
реал тан(реал х); //alias _тан тан;
реал акос(реал х); //alias _акос акос;
реал асин(реал х); //alias _асин асин;
реал атан(реал х); //alias _атан атан;
реал атан2(реал у, реал х); //alias _атан2 атан2;
реал гкос(реал х); //alias _гкос гкос;
реал гсин(реал х); //alias _гсин гсин;
реал гтан(реал х); //alias _гтан гтан;
реал гакос(реал х); //alias _гакос гакос;
реал гасин(реал х); //alias _гасин гасин;
реал гатан(реал х); //alias _гатан гатан;
дол округливдол(реал х); //alias _округливдол округливдол;
дол округливближдол(реал х); //alias _округливближдол округливближдол;
плав квкор(плав х); //alias _квкор квкор;
дво квкор(дво х);
реал квкор(реал х);
реал эксп(реал х); //alias _эксп эксп;
реал экспм1(реал х); //alias _экспм1 экспм1;
реал эксп2(реал х); //alias _эксп2 эксп2;
креал экспи(реал х); //alias _экспи экспи;
реал прэксп(реал знач, out цел эксп); //alias _прэксп прэксп;
цел илогб(реал х); //alias _илогб илогб;
реал лдэксп(реал н, цел эксп); //alias _лдэксп лдэксп;
реал лог(реал х); //alias _лог лог;
реал лог10(реал х); //alias _лог10 лог10;
реал лог1п(реал х); //alias _лог1п лог1п;
реал лог2(реал х); //alias _лог2 лог2;
реал логб(реал х); //alias _логб логб;
реал модф(реал х, inout реал у); //alias _модф модф;
реал скалбн(реал х, цел н); //alias _скалбн скалбн;
реал кубкор(реал х); //alias _кубкор кубкор;
реал фабс(реал х); //alias _фабс фабс;
реал гипот(реал х, реал у); //alias _гипот гипот;
реал фцош(реал х); //alias _фцош фцош;
реал лгамма(реал х); //alias _лгамма лгамма;
реал тгамма(реал х); //alias _тгамма тгамма;
реал потолок(реал х); //alias _потолок потолок;
реал пол(реал х); //alias _пол пол;
реал ближцел(реал х); //alias _ближцел ближцел;
цел окрвцел(реал х); //alias _окрвцел окрвцел;
реал окрвреал(реал х); //alias _окрвреал окрвреал;
дол окрвдол(реал х); //alias _окрвдол окрвдол;
реал округли(реал х); //alias _округли округли;
дол докругли(реал х); //alias _докругли докругли;
реал упрости(реал х); //alias _упрости упрости;
реал остаток(реал х, реал у); //alias _остаток остаток;
бул нч_ли(реал х); //alias _нч_ли нч_ли;
бул конечен_ли(реал р); //alias _конечен_ли конечен_ли;
бул субнорм_ли(плав п); //alias _субнорм_ли субнорм_ли;
бул субнорм_ли(дво п);
бул субнорм_ли(реал п);
бул беск_ли(реал р); //alias _беск_ли беск_ли;
бул идентичен_ли(реал р, реал д); //alias _идентичен_ли идентичен_ли;
бул битзнака(реал р); //alias _битзнака битзнака;
реал копируйзнак(реал кому, реал у_кого); //alias _копируйзнак
копируйзнак;
реал нч(ткст тэгп); //alias _нч нч;
реал следщБольш(реал р); //alias _следщБольш следщБольш;
```

```
дво следщБольш(дво р);
плав следщБольш(плав р);
реал следщМеньш(реал р); //alias _следщМеньш следщМеньш;
дво следщМеньш(дво р);
плав следщМеньш(плав р);
реал следщза(реал а, реал б); //alias _следщза следщза;
плав следщза(плав а, плав б);
дво следщза(дво а, дво б);
реал пдельта(реал а, реал б); //alias _пдельта пдельта;
реал пбольш_из(реал а, реал б); //alias _пбольш_из пбольш_из;
реал пменьш_из(реал а, реал б); //alias _пменьш_из пменьш_из;
реал степень(реал а, бцел н); //alias _степень степень;
реал степень(реал а, цел н);
реал степень(реал а, реал н);
бул правны(реал а, реал б); //alias _правны правны;
бул правны(реал а, реал б, реал эпс);
реал квадрат(цел а); //alias _квадрат квадрат;
реал дробь(реал а); //alias _дробь дробь;
цел знак(цел а); //alias _знак знак;
цел знак(дол а);
цел знак(реал а);
реал цикл8градус(реал ц); //alias _цикл8градус цикл8градус;
реал цикл8радиан(реал ц); //alias _цикл8радиан цикл8радиан;
реал цикл8градиент(реал ц); //alias _цикл8градиент цикл8градиент;
реал градус8цикл(реал г); //alias _градус8цикл градус8цикл;
реал градус8радиан(реал г); //alias _градус8радиан градус8радиан;
реал градус8градиент(реал г); //alias _градус8градиент градус8градиент;
реал радиан8градус(реал р); //alias _радиан8градус радиан8градус;
реал радиан8цикл(реал р); //alias _радиан8цикл радиан8цикл;
реал радиан8градиент(реал р); //alias _радиан8градиент радиан8градиент;
реал градиент8градус(реал г); //alias _градиент8градус градиент8градус;
реал градиент8цикл(реал г); //alias _градиент8цикл градиент8цикл;
реал градиент8радиан(реал г); //alias _градиент8радиан градиент8радиан;
реал сариф(реал[] ч); //alias _сариф сариф;
реал сумма(реал[] ч); //alias _сумма сумма;
реал меньш_из(реал[] ч); //alias _меньш_из меньш_из;
реал меньш_из(реал а, реал б);
реал больш_из(реал[] ч); //alias _больш_из больш_из;
реал больш_из(реал а, реал б);
реал акот(реал р); //alias _акот акот;
реал асек(реал р); //alias _асек асек;
реал акосек(реал р); //alias _акосек акосек;
реал кот(реал р); //alias _кот кот;
реал сек(реал р); //alias _сек сек;
реал косек(реал р); //alias _косек косек;
реал гкот(реал р); //alias _гкот гкот;
реал гсек(реал р); //alias _гсек гсек;
реал гкосек(реал р); //alias _гкосек гкосек;
реал гакот(реал р); //alias _гакот гакот;
реал гасек(реал р); //alias _гасек гасек;
реал гакосек(реал р); //alias _гакосек гакосек;
реал ткст8реал(ткст т); //alias _ткст8реал ткст8реал;

ткст подставь(ткст текст, ткст образец, ткст формат, ткст атрибуты =
пусто); //alias _подставь подставь;

цел найди(ртткст текст, ткст образец, ткст атрибуты);
цел найдирек(ртткст текст, ткст образец, ткст атрибуты = пусто);
ткст [] разбей(ткст текст, ткст образец, ткст атрибуты = пусто);

цел система (ткст команда); alias система сис;
цел пауза(); alias пауза пз;
цел пускпрог(цел режим, ткст путь, ткст[] арги);
цел выппрог(ткст путь, ткст[] арги);
```

```
    цел выппрог(ткст путь, ткст[] арги, ткст[] перемср);
    цел выппрогср(ткст путь, ткст[] арги);
    цел выппрогср(ткст путь, ткст[] арги, ткст[] перемср);

    //std.crc32 (ЦПИ - Циклическая Проверка Избыточности)
    бкрат иницЦПИ32();
    бкрат обновиЦПИ32б(ббайт зн, бцел црц);
    бкрат обновиЦПИ32с(сим зн, бцел црц);
    бкрат ткстЦПИ32(ткст т);

    бул закройДисковод(ткст меткаДиска);
    бул откройДисковод(ткст меткаДиска);

    проц инфо(ткст сооб); //alias _инфо инфо;
} //end of extern C
////////////////////////////////////////////////////
/+
СФайл двхо;
СФайл двых;
СФайл дош;

static this() {
// open standard I/O devices
двхо = new СФайл(СТДВВОД, ПФРежим.Ввод);
двых = new СФайл(СТДВЫВОД, ПФРежим.Вывод);
дош = new СФайл(СТДОШ, ПФРежим.Вывод);
}

static ~this()
{
двхо.слей();
двхо.закрой();
двых.слей();
двых.закрой();
дош.слей();
дош.закрой();
}
+/
extern(D):

ткст дайПеремСреды(ткст пер);
проц устПеремСреды(ткст пер, ткст знач);
ткст[] дайПуть();
бул гдеЯ(ткст арги, inout ткст пап, inout ткст имя);
ткст канонПуть(ткст origpath);
проц сделпапР(ткст пап);
проц удалиРек(ткст имя);
бул естьФайлВКэш(ткст имяФ);
проц удалиКэшСущФайлов();
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
17:44

```
module stringz;

extern(D):
```

```
/*********************************
* Преобразует массив символов в строку в стиле Си с нулевым окончанием.
* При предоставлении врм этот буфер будет по возможности использован
* вместо того, чтобы использовать размещение в куче.
*/

сим* вТкст0 (ткст s, ткст врм=null);

/*********************************
* Преобразует серию ткст в строки в стиле Си с нулевым окончанием,
* врм используется как рабочее пространство, а прм как место,
* куда помещается результат.
* Предназначено для эффективного одновременного конвертирования нескольких
строк.
*
* Возвращает заполненный срез прм
*
* Since: 0.99.7
*/

сим*[] вТкст0 (ткст врм, сим*[] прм, ткст[] строки...);

/*********************************
* Преобразует строку с нулевым окончанием в стиле Си в массив типа сим
*/

ткст изТкст0 (сим* s);

/*********************************
* ПЯреобразует массив типа шткст s[] в строку нулевого окончания в стиле Си.
*/

шим* вТкст16н (шткст s);

/*********************************
* Преобразует строку с нулевым окончанием в стиле Си в массив шим
*/

шткст изТкст16н (шим* s);

/*********************************
* Преобразует массив типа юткст s[] в строку нулевого окончания в стиле Си.
*/

дим* вТкст32н (юткст s);

/*********************************
* Преобразует строку нулевого окончания в стиле Си в массив типа дим
*/

юткст изТкст32н (дим* s);

/*********************************
* портабельный strlen
*/

т_мера длинтекс0(Т) (Т* s)
{
т_мера i;

if (s)
while (*s++)
++i;
return i;
```

```
}
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
17:45

```
module thread;
pragma(lib,"dinrus.lib");

extern (Windows) бцел нить_точкаВхода( ук арг );
extern (Windows)    ук ДайДескрТекущейНити();

extern (D)    class Нить
     {

          this( проц function() fn, т_мера разм = 0 );
          this( проц delegate() дг, т_мера разм = 0 );
          ~this();
          final проц старт()    ;
          final Объект присоедини( бул rethrow = да );
          final ткст имя();
          final проц имя( ткст знач );
          final бул демон_ли();
          final проц демон_ли( бул знач );
          final бул пущена_ли();
          static const цел МИНПРИОР;
          static const цел МАКСПРИОР;

          final цел приоритет();
          final проц приоритет( цел знач );
          static проц спи( double period );
          static проц жни();
          static Нить дайЭту();
          static Нить[] дайВсе();
          static цел opApply( цел delegate( ref Нить ) дг );
          static const бцел МАКСЛОК = 64;

          static бцел создайЛок();
          static проц удалиЛок( бцел key );
          static ук дайЛок( бцел key );
          static ук устЛок( бцел key, ук знач );
          static this()
          {
                    МИНПРИОР = -15;
                    МАКСПРИОР = 15;
          }
     private:
          final проц пуск();
               //
          // Тип процедуры, передаваемой при конструкции нити.
          //
          enum Вызов
          {
               НЕТ,
               ФН,
               ДГ
```

```
    }
    //
    // Стандартные типы
    //
        alias бцел КлючНЛХ;
        alias бцел АдрНити;

    //
    // Локальное хранилище
    //
    static бул[МАКСЛОК] см_локал;
    static КлючНЛХ см_эта;

    ук[МАКСЛОК] м_локал;


    //
    // Стандартные данные нити
    //
    version( Win32 )
    {
        ук м_дескр;
    }
    public АдрНити м_адр;
    Вызов м_вызов;
    ткст м_имя;
    union
    {
        проц function() м_фн;
        проц delegate() м_дг;
    }
    т_мера м_рр;
    version( Posix )
    {
        бул м_пущена;
    }
    бул м_демон;
    public Объект м_необработ;
    static проц установиЭту( Нить t );
    final проц суньКонтекст( Контекст* c );
    final проц выньКонтекст();
public final Контекст* топКонтекст();

    public static struct Контекст
    {
        ук нстэк,
                        встэк;
        Контекст* внутри;
        Контекст* следщ,
                        предщ;
    }


    Контекст м_глав;
    Контекст* м_тек;
    бул м_блок;

    version( Win32 )
    {
        бцел[8] м_рег; // edi,esi,ebp,esp,ebx,edx,ecx,eax
    }
    static Объект slock();
    static Контекст* см_кнач;
    static т_мера см_кдлин;
```

```
        static Нить см_ннач;
        static т_мера см_ндлин;

        //
        // Используется для упорядочивания нитей в глобальном списке.
        //
        Нить предщ;
        Нить следщ;
        static проц добавь( Контекст* с );
        static проц удали( Контекст* с );
        static проц добавь( Нить т );
        static проц удали( Нить т );

}
    class НитьЛок( Т )
    {
        this( Т def = Т.init )
        {
            м_деф = def;
            м_ключ = Нить.создайЛок();
        }

        ~this()
        {
            Нить.удалиЛок( м_ключ );
        }

        Т знач()
        {
            Обёртка* wrap = cast(Обёртка*) Нить.дайЛок( м_ключ );

            return wrap ? wrap.знач : м_деф;
        }

        Т знач( Т newval )
        {
            Обёртка* wrap = cast(Обёртка*) Нить.дайЛок( м_ключ );

            if( wrap is null )
            {
                wrap = new Обёртка;
                Нить.устЛок( м_ключ, wrap );
            }
            wrap.знач = newval;
            return newval;
        }

    private:

        struct Обёртка
        {
            Т знач;
        }

        Т м_деф;
        бцел м_ключ;
    }

extern (D)    class ГруппаНитей
    {
        final Нить создай( проц function() фн );
        final Нить создай( проц delegate() дг );
        final проц добавь( Нить т );
```

```
    final проц удали( Нить t );
    final цел opApply( цел delegate( ref Нить ) дг );
    final проц объединиВсе( бул rethrow = да );
}
    extern (D) class Фибра
{
    static class Планировщик
    {
        alias ук Дескр;

        enum Тип {Чтение =1, Запись=2, Приём=3, Подключение=4,
        Трансфер=5}

        проц пауза (бцел ms);
        проц готов (Фибра fiber);
        проц открой (Дескр fd, ткст имя);
        проц закрой (Дескр fd, ткст имя);
        проц ожидай (Дескр fd, Тип t, бцел timeout);
        проц ответви (ткст имя, проц delegate() дг, т_мера stack=8192);
    }

    final static Планировщик планировщик ();
    this(т_мера разм);
    this( проц function() fn, т_мера разм = РАЗМЕР_СТРАНИЦЫ);
    this( проц delegate() дг, т_мера разм = РАЗМЕР_СТРАНИЦЫ, Планировщик
    s = пусто );
    ~this();
    final Object вызови( бул rethrow = да );
    final проц сбрось();
    final проц сбрось( проц function() фн );
    final проц сбрось( проц delegate() дг );
    final проц сотри();
    final проц глуши ();
    static проц жни();
    static проц жниИБросай( Объект объ );
    static Фибра дайЭту();
    final проц пуск();
    enum Состояние
    {
        ЗАДЕРЖ, ///
        ВЫП, ///
        ТЕРМ ///
    }
            struct Событие
    {
        бцел инд; // поддержка удаления таймера
        Фибра следщ; // линкованный список просроченных фибр
        ук данные; // данные для обмена
        бдол время; // требуемая продолжительность таймаута
        Планировщик.Дескр хэндл; // хэндл запроса на ввод-вывод
        Планировщик планировщик; // ассоциированный планировщик (м. б.
        null)
    }

    final Состояние состояние();
    т_мера размерСтэка();

            enum Вызов
    {
        НЕТ,
        ФН,
        ДГ
    }
```

```
        //
        // Standard fiber данные
        //
        Вызов м_вызов;
        union
        {
             проц function() м_фн;
             проц delegate() м_дг;
        }
        бул м_пущена;
        Объект м_необработ;
        Состояние м_состояние;
        ткст м_имя;
    public:
        Событие событие;
        final проц разместиСтэк( т_мера разм );
        final проц освободиСтэк();
        final проц инициализуйСтэк();
        public Нить.Контекст* м_кткст;
        public т_мера м_разм;
        ук м_пам;
        static проц установиЭту( Фибра f );
        final проц подключись();
        final проц отключись();
        }

extern (C)
{
проц фибра_точкаВхода();
проц фибра_переклКонтекст( ук* старук, ук новук );
         проц нить_жни();
проц нить_спи(дво период);

проц нить_иниц();
проц нить_прикрепиЭту();
проц нить_открепиЭту();
проц нить_объединиВсе();
бул нить_нужнаБлокировка();
проц нить_заморозьВсе();
проц нить_размерозьВсе();
проц нить_сканируйВсе( фнСканВсеНити скан, ук текВерхСтека = null );

}
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
17:47

```
module sync;

alias Мютекс Стопор;

extern (D) class Условие
{

this( Мютекс m );
~this();
```

```
проц жди();
бул жди( дол период );
проц уведоми();
проц уведомиВсе();
}


extern (D) class Барьер
{
this( бцел предел );
     проц жди();
}

extern (D) class Семафор
{
this( бцел счёт = 0 );
~this();
проц жди();
бул жди( дол период );
проц уведоми();
бул пробуйЖдать();

}

extern (D) class Мютекс : Объект.Монитор
{

this();
this( Object o );
~this();
проц блокируй();
проц разблокируй();
void lock();
     void unlock();
бул пытайсяБлокировать();
}

extern(D) class ЧЗМютекс
{
enum Политика
{
ПОЧЁТ_ЧИТАТЕЛЮ, /// Readers get preference. This may starve writers.
ПОЧЁТ_ПИСАТЕЛЮ /// Writers get preference. This may starve readers.
}


this( Политика политика = Политика.ПОЧЁТ_ПИСАТЕЛЮ );

Политика политика();
Читатель читатель();
Писатель писатель();
     class Читатель : Объект.Монитор
          {
               this();
               проц блокируй();
               проц разблокируй();
          void lock();
          void unlock();
               бул пытайсяБлокировать();
          }

     class Писатель : Объект.Монитор
          {
               this();
```

```
            проц блокируй();
            проц разблокируй();
        void lock();
        void unlock();
            бул пытайсяБлокировать();
        }
}

extern (D) class ИсключениеСинх : Исключение
{
this( string msg );
}

////////////////////////////////////////////////////////////////////
//////ШАБЛОНЫ АТОМНЫХ ОПЕРАЦИЙ

////////////////////////////////////////////////////////////////////
private
{

template целыйТип_ли( T )
{
const бул целыйТип_ли = целыйЗначныйТип_ли!(T) ||
целыйБеззначныйТип_ли!(T);
}

template указательИлиКласс_ли(T)
{
const указательИлиКласс_ли = is(T == class);
}

template указательИлиКласс_ли(T : T*)
{
const указательИлиКласс_ли = true;
}
template целыйЗначныйТип_ли( T )
{
const бул целыйЗначныйТип_ли = is( T == byte ) ||
is( T == short ) ||
is( T == int ) ||
is( T == long )/+||
is( T == cent )+/;
}

template целыйБеззначныйТип_ли( T )
{
const бул целыйБеззначныйТип_ли = is( T == ббайт ) ||
is( T == ushort ) ||
is( T == бцел ) ||
is( T == ulong )/+||
is( T == ucent )+/;
}
template УкНаКласс(T){
static if (is(T==class)){
alias ук УкНаКласс;
} else {
alias T УкНаКласс;
}
}
}


template атомныеЗначенияПравильноРазмещены( T )
{
```

```
бул атомныеЗначенияПравильноРазмещены( т_мера адр )
{
return адр % УкНаКласс!(Т).sizeof == 0;
}
}

version(D_InlineAsm_X86){
проц барьерПамяти(bool ll, bool ls, bool sl,bool ss,bool device=false)(){
static if (device) {
if (ls || sl || ll || ss){
// cpid should sequence even more than mfence
volatile asm {
push EBX;
mov EAX, 0; // model, stepping
cpuid;
pop EBX;
}
}
} else static if (ls || sl || (ll && ss)){ // use a sequencing operation like
cpuid or simply cmpxch instead?
volatile asm {
mfence;
}
// this is supposedly faster and correct, but let's play it safe and use the
specific instruction
// push rax
// xchg rax
// pop rax
} else static if (ll){
volatile asm {
lfence;
}
} else static if( ss ){
volatile asm {
sfence;
}
}
}
} else version(D_InlineAsm_X86_64){
проц барьерПамяти(bool ll, bool ls, bool sl,bool ss,bool device=false)(){
static if (device) {
if (ls || sl || ll || ss){
// cpid should sequence even more than mfence
volatile asm {
push RBX;
mov RAX, 0; // model, stepping
cpuid;
pop RBX;
}
}
} else static if (ls || sl || (ll && ss)){ // use a sequencing operation like
cpuid or simply cmpxch instead?
volatile asm {
mfence;
}
// this is supposedly faster and correct, but let's play it safe and use the
specific instruction
// push rax
// xchg rax
// pop rax
} else static if (ll){
volatile asm {
lfence;
}
```

```
} else static if( ss ){
volatile asm {
sfence;
}
}
}
} else {
pragma(msg,"WARNING: no atomic operations on this architecture");
pragma(msg,"WARNING: this is *slow* you probably want to change this!");
цел dummy;
// acquires a блокируй... probably you will want to skip this
synchronized проц барьерПамяти(bool ll, bool ls, bool sl,bool ss,bool
device=false)(){
dummy =1;
}
enum{LockVersion = true}
}

static if (!is(typeof(LockVersion))) {
enum{LockVersion= false}
}

// use stricter fences
enum{strictFences=false}

/// utility function for a пиши barrier (disallow store and store reorderig)
проц барьерЗаписи();
/// utility function for a read barrier (disallow load and load reorderig)
проц барьерЧтения();
/// utility function for a full barrier (disallow reorderig)
проц полныйБарьер();


version(D_InlineAsm_X86) {
T атомнаяПерестановка( T )( inout T val, T newval )
in {
// NOTE: 32 bit x86 systems support 8 byte CAS, which only requires
// 4 byte alignment, so use т_мера as the align type here.
static if( T.sizeof > т_мера.sizeof )
assert( атомныеЗначенияПравильноРазмещены!(т_мера)( cast(т_мера) &val ) );
else
assert( атомныеЗначенияПравильноРазмещены!(T)( cast(т_мера) &val ) );
} body {
T*posVal=&val;
static if( T.sizeof == byte.sizeof ) {
volatile asm {
mov AL, newval;
mov ECX, posVal;
lock; // блокируй always needed to make this op atomic
xchg [ECX], AL;
}
}
else static if( T.sizeof == short.sizeof ) {
volatile asm {
mov AX, newval;
mov ECX, posVal;
lock; // блокируй always needed to make this op atomic
xchg [ECX], AX;
}
}
else static if( T.sizeof == цел.sizeof ) {
volatile asm {
mov EAX, newval;
mov ECX, posVal;
```

```
lock; // блокируй always needed to make this op atomic
xchg [ECX], EAX;
}
}
else static if( T.sizeof == дол.sizeof ) {
// 8 Byte swap on 32-Bit Processor, use CAS?
static assert( false, "Указан неверный шаблонный тип, 8 байт в 32-битном
режиме: "~T.stringof );
}
else
{
static assert( false, "Указан неверный шаблонный тип: "~T.stringof );
}
}
} else version (D_InlineAsm_X86_64){
T атомнаяПерестановка( T )( inout T val, T newval )
in {
assert( атомныеЗначенияПравильноРазмещены!(T)( cast(т_мера) &val ) );
} body {
T*posVal=&val;
static if( T.sizeof == byte.sizeof ) {
volatile asm {
mov AL, newval;
mov RCX, posVal;
lock; // блокируй always needed to make this op atomic
xchg [RCX], AL;
}
}
else static if( T.sizeof == short.sizeof ) {
volatile asm {
mov AX, newval;
mov RCX, posVal;
lock; // блокируй always needed to make this op atomic
xchg [RCX], AX;
}
}
else static if( T.sizeof == цел.sizeof ) {
volatile asm {
mov EAX, newval;
mov RCX, posVal;
lock; // блокируй always needed to make this op atomic
xchg [RCX], EAX;
}
}
else static if( T.sizeof == дол.sizeof ) {
volatile asm {
mov RAX, newval;
mov RCX, posVal;
lock; // блокируй always needed to make this op atomic
xchg [RCX], RAX;
}
}
else
{
static assert( false, "Указан неверный шаблонный тип: "~T.stringof );
}
}
} else {
T атомнаяПерестановка( T )( inout T val, T newval )
in {
assert( атомныеЗначенияПравильноРазмещены!(T)( cast(т_мера) &val ) );
} body {
T oldVal;
synchronized(typeid(T)){
```

```
oldVal=val;
val=newval;
}
return oldVal;
}
}

//---------------------
// internal conversion template
private T aCasT(T,V)(ref T val, T newval, T equalTo){
union UVConv{V v; T t;}
union UVPtrConv{V *v; T *t;}
UVConv vNew,vOld,vAtt;
UVPtrConv valPtr;
vNew.t=newval;
vOld.t=equalTo;
valPtr.t=&val;
vAtt.v=atomicCAS(*valPtr.v,vNew.v,vOld.v);
return vAtt.t;
}
/// internal reduction
private T aCas(T)(ref T val, T newval, T equalTo){
static if (T.sizeof==1){
return aCasT!(T,ббайт)(val,newval,equalTo);
} else static if (T.sizeof==2){
return aCasT!(T,ushort)(val,newval,equalTo);
} else static if (T.sizeof==4){
return aCasT!(T,бцел)(val,newval,equalTo);
} else static if (T.sizeof==8){ // unclear if it is always supported...
return aCasT!(T,ulong)(val,newval,equalTo);
} else {
static assert(0,"неверный тип "~T.stringof);
}
}

version(D_InlineAsm_X86) {
version(darwin){
extern(C) ббайт OSAtomicCompareAndSwap64(дол oldValue, дол newValue,
дол *theValue); // assumes that in C sizeof(_Bool)==1 (as given in osx IA-32
ABI)
}
T atomicCAS( T )( ref T val, T newval, T equalTo )
in {
// NOTE: 32 bit x86 systems support 8 byte CAS, which only requires
// 4 byte alignment, so use т_мера as the align type here.
static if( УкНаКласс!(T).sizeof > т_мера.sizeof )
assert( атомныеЗначенияПравильноРазмещены!(т_мера)( cast(т_мера) &val ) );
else
assert( атомныеЗначенияПравильноРазмещены!(УкНаКласс!(T))( cast(т_мера) &val
) );
} body {
T*posVal=&val;
static if( T.sizeof == byte.sizeof ) {
volatile asm {
mov DL, newval;
mov AL, equalTo;
mov ECX, posVal;
lock; // блокируй always needed to make this op atomic
cmpxchg [ECX], DL;
}
}
else static if( T.sizeof == short.sizeof ) {
volatile asm {
mov DX, newval;
```

```
mov AX, equalTo;
mov ECX, posVal;
lock; // блокируй always needed to make this op atomic
cmpxchg [ECX], DX;
}
}
else static if( УкНаКласс!(T).sizeof == цел.sizeof ) {
volatile asm {
mov EDX, newval;
mov EAX, equalTo;
mov ECX, posVal;
lock; // блокируй always needed to make this op atomic
cmpxchg [ECX], EDX;
}
}
else static if( T.sizeof == дол.sizeof ) {
// 8 Byte StoreIf on 32-Bit Processor
version(darwin){
union UVConv{дол v; T t;}
union UVPtrConv{дол *v; T *t;}
UVConv vEqual,vNew;
UVPtrConv valPtr;
vEqual.t=equalTo;
vNew.t=newval;
valPtr.t=&val;
while(1){
if(OSAtomicCompareAndSwap64(vEqual.v, vNew.v, valPtr.v)!=0)
{
return equalTo;
} else {
volatile {
T res=val;
if (res!is equalTo) return res;
}
}
}
} else {
T res;
volatile asm
{
push EDI;
push EBX;
lea EDI, newval;
mov EBX, [EDI];
mov ECX, 4[EDI];
lea EDI, equalTo;
mov EAX, [EDI];
mov EDX, 4[EDI];
mov EDI, val;
lock; // блокируй always needed to make this op atomic
cmpxch8b [EDI];
lea EDI, res;
mov [EDI], EAX;
mov 4[EDI], EDX;
pop EBX;
pop EDI;
}
return res;
}
}
else
{
static assert( false, "Указан неверный шаблонный тип: "~T.stringof );
}
```

```
}
} else version (D_InlineAsm_X86_64){
T atomicCAS( T )( ref T val, T newval, T equalTo )
in {
assert( атомныеЗначенияПравильноРазмещены!(T)( cast(т_мера) &val ) );
} body {
T*posVal=&val;
static if( T.sizeof == byte.sizeof ) {
volatile asm {
mov DL, newval;
mov AL, equalTo;
mov RCX, posVal;
lock; // блокируй always needed to make this op atomic
cmpxchg [RCX], DL;
}
}
else static if( T.sizeof == short.sizeof ) {
volatile asm {
mov DX, newval;
mov AX, equalTo;
mov RCX, posVal;
lock; // блокируй always needed to make this op atomic
cmpxchg [RCX], DX;
}
}
else static if( УкНаКласс!(T).sizeof == цел.sizeof ) {
volatile asm {
mov EDX, newval;
mov EAX, equalTo;
mov RCX, posVal;
lock; // блокируй always needed to make this op atomic
cmpxchg [RCX], EDX;
}
}
else static if( УкНаКласс!(T).sizeof == дол.sizeof ) {
volatile asm {
mov RDX, newval;
mov RAX, equalTo;
mov RCX, posVal;
lock; // блокируй always needed to make this op atomic
cmpxchg [RCX], RDX;
}
}
else
{
static assert( false, "Задан неправильный шаблонный тип: "~T.stringof );
}
}
} else {
T atomicCAS( T )( ref T val, T newval, T equalTo )
in {
assert( атомныеЗначенияПравильноРазмещены!(T)( cast(т_мера) &val ) );
} body {
T oldval;
synchronized(typeid(T)){
oldval=val;
if(oldval==equalTo) {
val=newval;
}
}
return oldval;
}
}
```

```
бул atomicCASB(T)( ref T val, T newval, T equalTo ){
return (equalTo is atomicCAS(val,newval,equalTo));
}


T атомнаяЗагрузка(T)(ref T val)
in {
assert( атомныеЗначенияПравильноРазмещены!(T)( cast(т_мера) &val ) );
static assert(УкНаКласс!(T).sizeof<=т_мера.sizeof,"неверный размер для
"~T.stringof);
} body {
volatile T res=val;
return res;
}


проц атомноеСохранение(T)(ref T val, T newVal)
in {
assert( атомныеЗначенияПравильноРазмещены!(T)( cast(т_мера) &val ), "неверная
раскладка" );
static assert(УкНаКласс!(T).sizeof<=т_мера.sizeof,"наверный размер для
"~T.stringof);
} body {
volatile val=newVal;
}


version (D_InlineAsm_X86){
T атомнаяПрибавка(T,U=T)(ref T val, U incV_){
T incV=cast(T)incV_;
static if (целыйТип_ли!(T)||указательИлиКласс_ли!(T)){
T* posVal=&val;
T res;
static if (T.sizeof==1){
volatile asm {
mov DL, incV;
mov ECX, posVal;
lock;
xadd byte ptr [ECX],DL;
mov byte ptr res[EBP],DL;
}
} else static if (T.sizeof==2){
volatile asm {
mov DX, incV;
mov ECX, posVal;
lock;
xadd short ptr [ECX],DX;
mov short ptr res[EBP],DX;
}
} else static if (T.sizeof==4){
volatile asm
{
mov EDX, incV;
mov ECX, posVal;
lock;
xadd int ptr [ECX],EDX;
mov int ptr res[EBP],EDX;
}
} else static if (T.sizeof==8){
return атомнаяОп(val,delegate (T x){ return x+incV; });
} else {
static assert(0,"Неподдерживаемый размер типа");
}
return res;
```

```
} else {
return атомнаяОп(val,delegate T(T a){ return a+incV; });
}
}
} else version (D_InlineAsm_X86_64){
T атомнаяПрибавка(T,U=T)(ref T val, U incV_){
T incV=cast(T)incV_;
static if (целыйТип_ли!(T)||указательИлиКласс_ли!(T)){
T* posVal=&val;
T res;
static if (T.sizeof==1){
volatile asm {
mov DL, incV;
mov RCX, posVal;
lock;
xadd byte ptr [RCX],DL;
mov byte ptr res[EBP],DL;
}
} else static if (T.sizeof==2){
volatile asm {
mov DX, incV;
mov RCX, posVal;
lock;
xadd short ptr [RCX],DX;
mov short ptr res[EBP],DX;
}
} else static if (T.sizeof==4){
volatile asm
{
mov EDX, incV;
mov RCX, posVal;
lock;
xadd int ptr [RCX],EDX;
mov int ptr res[EBP],EDX;
}
} else static if (T.sizeof==8){
volatile asm
{
mov RAX, val;
mov RDX, incV;
lock; // блокируй always needed to make this op atomic
xadd qword ptr [RAX],RDX;
mov res[EBP],RDX;
}
} else {
static assert(0,"Неподдерживаемый размер для типа:"~T.stringof);
}
return res;
} else {
return атомнаяОп(val,delegate T(T a){ return a+incV; });
}
}
} else {
static if (LockVersion){
T атомнаяПрибавка(T,U=T)(ref T val, U incV_){
T incV=cast(T)incV_;
static assert( целыйТип_ли!(T)||указательИлиКласс_ли!(T),"неверный тип:
"~T.stringof );
synchronized(typeid(T)){
T oldV=val;
val+=incV;
return oldV;
}
}
```

```
} else {
T атомнаяПрибавка(T,U=T)(ref T val, U incV_){
T incV=cast(T)incV_;
static assert( целыйТип_ли!(T)||указательИлиКласс_ли!(T),"неверный тип:
"~T.stringof );
synchronized(typeid(T)){
T oldV,newVal,nextVal;
volatile nextVal=val;
do{
oldV=nextVal;
newV=oldV+incV;
auto nextVal=atomicCAS!(T)(val,newV,oldV);
} while(nextVal!=oldV)
return oldV;
}
}
}
}


T атомнаяОп(T)(ref T val, T delegate(T) f){
T oldV,newV,nextV;
цел i=0;
nextV=val;
do {
oldV=nextV;
newV=f(oldV);
nextV=aCas!(T)(val,newV,oldV);
if (nextV is oldV || newV is oldV) return oldV;
} while(++i<200)
while (true){
нить_жни();
volatile oldV=val;
newV=f(oldV);
nextV=aCas!(T)(val,newV,oldV);
if (nextV is oldV || newV is oldV) return oldV;
}
}


T флагДай(T)(ref T флаг){
T res;
volatile res=флаг;
барьерПамяти!(true,false,strictFences,false)();
return res;
}

T флагУст(T)(ref T флаг,T newVal){
барьерПамяти!(false,strictFences,false,true)();
return атомнаяПерестановка(флаг,newVal);
}

T флагОп(T)(ref T флаг,T delegate(T) op){
барьерПамяти!(false,strictFences,false,true)();
return атомнаяОп(флаг,op);
}

T флагДоб(T)(ref T флаг,T incV=cast(T)1){
static if (!LockVersion)
барьерПамяти!(false,strictFences,false,true)();
return атомнаяПрибавка(флаг,incV);
}

T следщЗнач(T)(ref T val){
```

```
return атомнаяПрибавка(val,cast(T)1);
}
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
17:53

```
module winthread;
import dinrus, time.Time;

alias HANDLE Handle;
extern(Windows)
{

struct SECURITY_ATTRIBUTES {
uint nLength;
void* lpSecurityDescriptor;
int bInheritHandle;
}

const uint INFINITE = 0xFFFFFFFF;

enum : uint {
WAIT_OBJECT_0 = 0,
WAIT_ABANDONED = 0x80,
WAIT_ABANDONED_0 = 0x80,
WAIT_TIMEOUT = 258,
SYNCHRONIZE = 0x00100000,
ERROR_ACCESS_DENIED = 5,
}

uint GetLastError();

uint WaitForSingleObject(Handle hHandle, uint dwMilliseconds);

uint WaitForSingleObjectEx(Handle hHandle, uint dwMilliseconds, BOOL
bAlertable);

uint WaitForMultipleObjects(uint nCount, in Handle* lpHandles, BOOL bWaitAll,
uint dwMilliseconds);

uint WaitForMultipleObjectsEx(uint nCount, in Handle* lpHandles, BOOL
bWaitAll, uint dwMilliseconds, BOOL bAlertable);

uint SignalObjectAndWait(Handle hObjectToSignal, Handle hObjectToWaitOn, uint
dwMilliseconds, BOOL bAlertable);

void Sleep(uint dwMilliseconds);

uint SleepEx(uint dwMilliseconds, int bAlertable);

uint TlsAlloc();

int TlsFree(uint dwTlsIndex);

void* TlsGetValue(uint dwTlsIndex);
```

```
int TlsSetValue(uint dwTlsIndex, void* lpTlsValue);

Handle CreateEventW(SECURITY_ATTRIBUTES* lpEventAttributes, int bManualReset,
int bInitialState, in wchar* lpName);
alias CreateEventW CreateEvent;

int SetEvent(Handle hEvent);

int ResetEvent(Handle hEvent);

Handle CreateMutexW(SECURITY_ATTRIBUTES* lpMutexAttributes, int
bInitialOwner, in wchar* lpName);
alias CreateMutexW CreateMutex;

enum : uint {
MUTEX_MODIFY_STATE = 0x0001
}
Handle OpenMutexW(uint dwDesiredAccess, int bInheritHandle, in wchar*
lpName);
alias OpenMutexW OpenMutex;

int ReleaseMutex(Handle hMutex);

Handle CreateSemaphoreW(SECURITY_ATTRIBUTES* lpSemaphoreAttributes, int
lInitialCount, int lMaximumCount, in wchar* lpName);
alias CreateSemaphoreW CreateSemaphore;

int ReleaseSemaphore(Handle hSemaphore, int lReleaseCount, out int
lpPreviousCount);

int CloseHandle(Handle hObject);
}

class ВинНитьЛок(Т) {


private struct ДанныеНлх {
Т значение;
}

private бцел слот_;
private Т дефЗначение_;

/**
* Initializes a new instance.
*/
this(lazy Т дефЗначение = Т.init) {
дефЗначение_ = cast(Т)дефЗначение();
слот_ = TlsAlloc();
}

~this() {
if (auto данныеНлх = cast(ДанныеНлх*)TlsGetValue(слот_))
удалиКорень(данныеНлх);

TlsFree(слот_);
слот_ = cast(бцел)-1;
}

/**
* Gets the значение in the current thread's copy of this instance.
* Возвращает: The current thread's copy of this instance.
*/
final Т дай() {
```

```
if (auto данныеНлх = cast(ДанныеНлх*)TlsGetValue(слот_))
return данныеНлх.значение;
return дефЗначение_;
}

/**
* Sets the current thread's copy of this instance to the specified _value.
* Параметры: значение = The _value to be stored in the current thread's copy
of this instance.
*/
final проц установи(Т значение) {
auto данныеНлх = cast(ДанныеНлх*)TlsGetValue(слот_);
if (данныеНлх is null) {
данныеНлх = new ДанныеНлх;
добавьКорень(данныеНлх);

TlsSetValue(слот_, данныеНлх);
}
данныеНлх.значение = значение;
}

}


/**
* Suspends the current thread for a specified time.
* Параметры: миллисек = The number of _milliseconds for which the thread is
blocked. Specify -1 to block the thread indefinitely.
*/
проц спи(бцел миллисек);

/**
* Suspends the current thread for a specified time.
* Параметры: таймаут = The amount of time for which the thread is blocked.
Specify -1 to block the thread indefinitely.
*/
проц спи(ИнтервалВремени таймаут);

enum ПРежимСбросаСобытия {
Авто,
Ручной
}

abstract class УкОжидание {

private Дескр дескр_ = cast(Дескр)НЕВЕРНХЭНДЛ;

проц закрой() ;

бул ждиОдин(бцел таймаутВМиллисек = INFINITE);

бул ждиОдин(ИнтервалВремени таймаут) ;

static бул ждиВсе(УкОжидание[] ждиуки, бцел таймаутВМиллисек = INFINITE);

static бцел ждиЛюбой(УкОжидание[] ждиуки, бцел таймаутВМиллисек = INFINITE);

static бул сигнализируйИЖди(УкОжидание toSignal, УкОжидание toWaitOn, бцел
таймаутВМиллисек = INFINITE) ;

static бул сигнализируйИЖди(УкОжидание toSignal, УкОжидание toWaitOn,
ИнтервалВремени таймаут);

проц дескр(Дескр значение) ;
```

```
Дескр дескр() ;

}

class ДескрОжиданияСобытия : УкОжидание {

this(бул начСостояние, ПРежимСбросаСобытия режим) ;

final бул установи();

final бул сбрось() ;

}

final class СобытиеАвтоСброса : ДескрОжиданияСобытия {

this(бул начСостояние) ;

}

final class СобытиеРучногоСброса : ДескрОжиданияСобытия {

this(бул начСостояние);

}

final class ВинМютекс : УкОжидание {

this(бул initiallyOwned = false, ткст имя = null);

проц отпусти() ;

}

final class ВинСемафор : УкОжидание {

this(цел начСчёт, цел максСчёт, ткст имя = пусто) ;

цел отпусти(цел релизСчёт = 1);

}
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
17:56

```
module winapi;
public import sys.uuid;

extern (D)
{
WORD HIWORD(int l) { return cast(WORD)((l >> 16) & 0xFFFF); }
WORD LOWORD(int l) { return cast(WORD)l; }
bool FAILED(int status) { return status < 0; }
bool SUCCEEDED(int Status) { return Status >= 0; }
}
```

```
extern (Windows)
{

/+
alias uint ULONG;
alias ULONG *PULONG;
alias ushort USHORT;
alias USHORT *PUSHORT;
alias ubyte UCHAR;
alias UCHAR *PUCHAR;
alias char *PSZ;
alias wchar WCHAR;

alias void VOID;
alias char CHAR;
alias short SHORT;
alias int LONG;
alias CHAR *LPSTR;
alias CHAR *PSTR;

alias CHAR *LPCSTR;
alias CHAR *PCSTR;

alias LPSTR LPTCH, PTCH;
alias LPSTR PTSTR, LPTSTR;
alias LPCSTR LPCTSTR;

alias WCHAR* LPWSTR;

alias WCHAR *LPCWSTR, PCWSTR;

alias uint DWORD;
alias ulong DWORD64;
alias int BOOL;
alias ubyte BYTE;
alias ushort WORD;
alias float FLOAT;
alias FLOAT *PFLOAT;
alias BOOL *PBOOL;
alias BOOL *LPBOOL;
alias BYTE *PBYTE;
alias BYTE *LPBYTE;
alias int *PINT;
alias int *LPINT;
alias WORD *PWORD;
alias WORD *LPWORD;
alias int *LPLONG;
alias DWORD *PDWORD;
alias DWORD *LPDWORD;
alias void *LPVOID;
alias void *LPCVOID;

alias int INT;
alias uint UINT;
alias uint *PUINT;

// ULONG_PTR must be able to store a pointer as an integral type
version (Win64)
{
alias long INT_PTR;
alias ulong UINT_PTR;
alias long LONG_PTR;
alias ulong ULONG_PTR;
alias long * PINT_PTR;
```

```
alias ulong * PUINT_PTR;
alias long * PLONG_PTR;
alias ulong * PULONG_PTR;
}
else // Win32
{
alias int INT_PTR;
alias uint UINT_PTR;
alias int LONG_PTR;
alias uint ULONG_PTR;
alias int * PINT_PTR;
alias uint * PUINT_PTR;
alias int * PLONG_PTR;
alias uint * PULONG_PTR;
}

typedef void *HANDLE;
alias void *PVOID;
alias HANDLE HGLOBAL;
alias HANDLE HLOCAL;
alias LONG HRESULT;
alias LONG SCODE;
alias HANDLE HINSTANCE;
alias HINSTANCE HMODULE;
alias HANDLE HWND;

alias HANDLE HGDIOBJ;
alias HANDLE HACCEL;
alias HANDLE HBITMAP;
alias HANDLE HBRUSH;
alias HANDLE HCOLORSPACE;
alias HANDLE HDC;
alias HANDLE HGLRC;
alias HANDLE HDESK;
alias HANDLE HENHMETAFILE;
alias HANDLE HFONT;
alias HANDLE HICON;
alias HANDLE HMENU;
alias HANDLE HMETAFILE;
alias HANDLE HPALETTE;
alias HANDLE HPEN;
alias HANDLE HRGN;
alias HANDLE HRSRC;
alias HANDLE HSTR;
alias HANDLE HTASK;
alias HANDLE HWINSTA;
alias HANDLE HKL;
alias HICON HCURSOR;

alias HANDLE HKEY;
alias HKEY *PHKEY;
alias DWORD ACCESS_MASK;
alias ACCESS_MASK *PACCESS_MASK;
alias ACCESS_MASK REGSAM;

alias int function() FARPROC;

alias UINT WPARAM;
alias LONG LPARAM;
alias LONG LRESULT;

alias DWORD COLORREF;
alias DWORD *LPCOLORREF;
alias WORD ATOM;
```

```
version (0)
{ // Properly prototyped versions
alias BOOL function(HWND, UINT, WPARAM, LPARAM) DLGPROC;
alias VOID function(HWND, UINT, UINT, DWORD) TIMERPROC;
alias BOOL function(HDC, LPARAM, int) GRAYSTRINGPROC;
alias BOOL function(HWND, LPARAM) WNDENUMPROC;
alias LRESULT function(int code, WPARAM wParam, LPARAM lParam) HOOKPROC;
alias VOID function(HWND, UINT, DWORD, LRESULT) SENDASYNCPROC;
alias BOOL function(HWND, LPCSTR, HANDLE) PROPENUMPROCA;
alias BOOL function(HWND, LPCWSTR, HANDLE) PROPENUMPROCW;
alias BOOL function(HWND, LPSTR, HANDLE, DWORD) PROPENUMPROCEXA;
alias BOOL function(HWND, LPWSTR, HANDLE, DWORD) PROPENUMPROCEXW;
alias int function(LPSTR lpch, int ichCurrent, int cch, int code)
EDITWORDBREAKPROCA;
alias int function(LPWSTR lpch, int ichCurrent, int cch, int code)
EDITWORDBREAKPROCW;
alias BOOL function(HDC hdc, LPARAM lData, WPARAM wData, int cx, int cy)
DRAWSTATEPROC;
}
else
{
alias FARPROC DLGPROC;
alias FARPROC TIMERPROC;
alias FARPROC GRAYSTRINGPROC;
alias FARPROC WNDENUMPROC;
alias FARPROC HOOKPROC;
alias FARPROC SENDASYNCPROC;
alias FARPROC EDITWORDBREAKPROCA;
alias FARPROC EDITWORDBREAKPROCW;
alias FARPROC PROPENUMPROCA;
alias FARPROC PROPENUMPROCW;
alias FARPROC PROPENUMPROCEXA;
alias FARPROC PROPENUMPROCEXW;
alias FARPROC DRAWSTATEPROC;
}
+/

    struct TRACKMOUSEEVENT
    {
        DWORD cbSize;
        DWORD dwFlags;
        HWND hwndTrack;
        DWORD dwHoverTime;
    }
    alias TRACKMOUSEEVENT* LPTRACKMOUSEEVENT;
        struct INITCOMMONCONTROLSEX
    {
        DWORD dwSize;
        DWORD dwICC;
    }
    alias INITCOMMONCONTROLSEX* LPINITCOMMONCONTROLSEX;
    alias INITCOMMONCONTROLSEX* PINITCOMMONCONTROLSEX;
        enum: DWORD
    {
        // IE3+
        ICC_LISTVIEW_CLASSES = 0x00000001,
        ICC_TREEVIEW_CLASSES = 0x00000002,
        ICC_BAR_CLASSES = 0x00000004, // tool/status/track
        ICC_TAB_CLASSES = 0x00000008,
        ICC_UPDOWN_CLASS = 0x00000010,
        ICC_PROGRESS_CLASS = 0x00000020,
        ICC_HOTKEY_CLASS = 0x00000040,
        ICC_ANIMATE_CLASS = 0x00000080,
```

```
        ICC_WIN95_CLASSES = 0x000000FF,
        ICC_DATE_CLASSES = 0x00000100,
    ICC_USEREX_CLASSES = 0x00000200,
        ICC_COOL_CLASSES = 0x00000400,
        ICC_STANDARD_CLASSES = 0x00004000,
        // IE4+
        ICC_INTERNET_CLASSES = 0x00000800,
        ICC_PAGESCROLLER_CLASS = 0x00001000,
        ICC_NATIVEFNTCTL_CLASS = 0x00002000,
    }

enum : uint
{
MAX_PATH = 260,
HINSTANCE_ERROR = 32,
}

enum
{
ERROR_SUCCESS = 0,
ERROR_INVALID_FUNCTION = 1,
ERROR_FILE_NOT_FOUND = 2,
ERROR_PATH_NOT_FOUND = 3,
ERROR_TOO_MANY_OPEN_FILES = 4,
ERROR_ACCESS_DENIED = 5,
ERROR_INVALID_HANDLE = 6,
ERROR_NO_MORE_FILES = 18,
ERROR_MORE_DATA = 234,
ERROR_NO_MORE_ITEMS = 259,
}

enum
{
DLL_PROCESS_ATTACH = 1,
DLL_THREAD_ATTACH = 2,
DLL_THREAD_DETACH = 3,
DLL_PROCESS_DETACH = 0,
}

enum
{
FILE_BEGIN = 0,
FILE_CURRENT = 1,
FILE_END = 2,
}

enum : uint
{
DELETE = 0x00010000,
READ_CONTROL = 0x00020000,
WRITE_DAC = 0x00040000,
WRITE_OWNER = 0x00080000,
SYNCHRONIZE = 0x00100000,

STANDARD_RIGHTS_REQUIRED = 0x000F0000,
STANDARD_RIGHTS_READ = READ_CONTROL,
STANDARD_RIGHTS_WRITE = READ_CONTROL,
STANDARD_RIGHTS_EXECUTE = READ_CONTROL,
STANDARD_RIGHTS_ALL = 0x001F0000,
SPECIFIC_RIGHTS_ALL = 0x0000FFFF,
ACCESS_SYSTEM_SECURITY = 0x01000000,
MAXIMUM_ALLOWED = 0x02000000,

GENERIC_READ = 0x80000000,
```

```
GENERIC_WRITE = 0x40000000,
GENERIC_EXECUTE = 0x20000000,
GENERIC_ALL = 0x10000000,
}

enum
{
FILE_SHARE_READ = 0x00000001,
FILE_SHARE_WRITE = 0x00000002,
FILE_SHARE_DELETE = 0x00000004,
FILE_ATTRIBUTE_READONLY = 0x00000001,
FILE_ATTRIBUTE_HIDDEN = 0x00000002,
FILE_ATTRIBUTE_SYSTEM = 0x00000004,
FILE_ATTRIBUTE_DIRECTORY = 0x00000010,
FILE_ATTRIBUTE_ARCHIVE = 0x00000020,
FILE_ATTRIBUTE_NORMAL = 0x00000080,
FILE_ATTRIBUTE_TEMPORARY = 0x00000100,
FILE_ATTRIBUTE_COMPRESSED = 0x00000800,
FILE_ATTRIBUTE_OFFLINE = 0x00001000,
FILE_NOTIFY_CHANGE_FILE_NAME = 0x00000001,
FILE_NOTIFY_CHANGE_DIR_NAME = 0x00000002,
FILE_NOTIFY_CHANGE_ATTRIBUTES = 0x00000004,
FILE_NOTIFY_CHANGE_SIZE = 0x00000008,
FILE_NOTIFY_CHANGE_LAST_WRITE = 0x00000010,
FILE_NOTIFY_CHANGE_LAST_ACCESS = 0x00000020,
FILE_NOTIFY_CHANGE_CREATION = 0x00000040,
FILE_NOTIFY_CHANGE_SECURITY = 0x00000100,
FILE_ACTION_ADDED = 0x00000001,
FILE_ACTION_REMOVED = 0x00000002,
FILE_ACTION_MODIFIED = 0x00000003,
FILE_ACTION_RENAMED_OLD_NAME = 0x00000004,
FILE_ACTION_RENAMED_NEW_NAME = 0x00000005,
FILE_CASE_SENSITIVE_SEARCH = 0x00000001,
FILE_CASE_PRESERVED_NAMES = 0x00000002,
FILE_UNICODE_ON_DISK = 0x00000004,
FILE_PERSISTENT_ACLS = 0x00000008,
FILE_FILE_COMPRESSION = 0x00000010,
FILE_VOLUME_IS_COMPRESSED = 0x00008000,
}

enum : DWORD
{
MAILSLOT_NO_MESSAGE = cast(DWORD)-1,
MAILSLOT_WAIT_FOREVER = cast(DWORD)-1,
}

enum : uint
{
FILE_FLAG_WRITE_THROUGH = 0x80000000,
FILE_FLAG_OVERLAPPED = 0x40000000,
FILE_FLAG_NO_BUFFERING = 0x20000000,
FILE_FLAG_RANDOM_ACCESS = 0x10000000,
FILE_FLAG_SEQUENTIAL_SCAN = 0x08000000,
FILE_FLAG_DELETE_ON_CLOSE = 0x04000000,
FILE_FLAG_BACKUP_SEMANTICS = 0x02000000,
FILE_FLAG_POSIX_SEMANTICS = 0x01000000,
}

enum
{
CREATE_NEW = 1,
CREATE_ALWAYS = 2,
OPEN_EXISTING = 3,
OPEN_ALWAYS = 4,
```

```
TRUNCATE_EXISTING = 5,
}



const HANDLE INVALID_HANDLE_VALUE = cast(HANDLE)-1;
const DWORD INVALID_SET_FILE_POINTER = cast(DWORD)-1;
const DWORD INVALID_FILE_SIZE = cast(DWORD)0xFFFFFFFF;


struct OVERLAPPED {
DWORD Internal;
DWORD InternalHigh;
DWORD Offset;
DWORD OffsetHigh;
HANDLE hEvent;
}
alias OVERLAPPED *LPOVERLAPPED;

struct SECURITY_ATTRIBUTES {
DWORD nLength;
void *lpSecurityDescriptor;
BOOL bInheritHandle;
}

alias SECURITY_ATTRIBUTES* PSECURITY_ATTRIBUTES, LPSECURITY_ATTRIBUTES;

struct FILETIME {
DWORD dwLowDateTime;
DWORD dwHighDateTime;
}
alias FILETIME* PFILETIME, LPFILETIME;

struct WIN32_FIND_DATA {
DWORD dwFileAttributes;
FILETIME ftCreationTime;
FILETIME ftLastAccessTime;
FILETIME ftLastWriteTime;
DWORD nFileSizeHigh;
DWORD nFileSizeLow;
DWORD dwReserved0;
DWORD dwReserved1;
char cFileName[MAX_PATH];
char cAlternateFileName[ 14 ];
}

struct WIN32_FIND_DATAW {
DWORD dwFileAttributes;
FILETIME ftCreationTime;
FILETIME ftLastAccessTime;
FILETIME ftLastWriteTime;
DWORD nFileSizeHigh;
DWORD nFileSizeLow;
DWORD dwReserved0;
DWORD dwReserved1;
wchar cFileName[MAX_PATH];
wchar cAlternateFileName[ 14 ];
}

// Critical Section

struct _LIST_ENTRY
{
_LIST_ENTRY *Flink;
```

```
_LIST_ENTRY *Blink;
}
alias _LIST_ENTRY LIST_ENTRY;

struct _RTL_CRITICAL_SECTION_DEBUG
{
WORD Type;
WORD CreatorBackTraceIndex;
_RTL_CRITICAL_SECTION *CriticalSection;
LIST_ENTRY ProcessLocksList;
DWORD EntryCount;
DWORD ContentionCount;
DWORD Spare[ 2 ];
}
alias _RTL_CRITICAL_SECTION_DEBUG RTL_CRITICAL_SECTION_DEBUG;

struct _RTL_CRITICAL_SECTION
{
RTL_CRITICAL_SECTION_DEBUG * DebugInfo;

//
// The following three fields control entering and exiting the critical
// section for the resource
//

LONG LockCount;
LONG RecursionCount;
HANDLE OwningThread; // from the thread's ClientId->UniqueThread
HANDLE LockSemaphore;
ULONG_PTR SpinCount; // force size on 64-bit systems when packed
}
alias _RTL_CRITICAL_SECTION CRITICAL_SECTION;


enum
{
STD_INPUT_HANDLE = cast(DWORD)-10,
STD_OUTPUT_HANDLE = cast(DWORD)-11,
STD_ERROR_HANDLE = cast(DWORD)-12,
}

extern (Windows)
{
BOOL SetCurrentDirectoryA(LPCSTR lpPathName);
DWORD GetCurrentDirectoryA(DWORD nBufferLength, LPSTR lpBuffer);
BOOL CreateDirectoryA(LPCSTR lpPathName, LPSECURITY_ATTRIBUTES
lpSecurityAttributes);
BOOL CreateDirectoryW(LPCWSTR lpPathName, LPSECURITY_ATTRIBUTES
lpSecurityAttributes);
BOOL CreateDirectoryExA(LPCSTR lpTemplateDirectory, LPCSTR lpNewDirectory,
LPSECURITY_ATTRIBUTES lpSecurityAttributes);
BOOL CreateDirectoryExW(LPCWSTR lpTemplateDirectory, LPCWSTR lpNewDirectory,
LPSECURITY_ATTRIBUTES lpSecurityAttributes);
BOOL RemoveDirectoryA(LPCSTR lpPathName);
BOOL RemoveDirectoryW(LPCWSTR lpPathName);

BOOL CloseHandle(HANDLE hObject);

HANDLE CreateFileA(in char* lpFileName, DWORD dwDesiredAccess, DWORD
dwShareMode,
SECURITY_ATTRIBUTES *lpSecurityAttributes, DWORD dwCreationDisposition,
DWORD dwFlagsAndAttributes, HANDLE hTemplateFile);
HANDLE CreateFileW(LPCWSTR lpFileName, DWORD dwDesiredAccess, DWORD
dwShareMode,
```

```
SECURITY_ATTRIBUTES *lpSecurityAttributes, DWORD dwCreationDisposition,
DWORD dwFlagsAndAttributes, HANDLE hTemplateFile);

BOOL DeleteFileA(in char *lpFileName);
BOOL DeleteFileW(LPCWSTR lpFileName);

BOOL FindClose(HANDLE hFindFile);
HANDLE FindFirstFileA(in char *lpFileName, WIN32_FIND_DATA* lpFindFileData);
HANDLE FindFirstFileW(in LPCWSTR lpFileName, WIN32_FIND_DATAW*
lpFindFileData);
BOOL FindNextFileA(HANDLE hFindFile, WIN32_FIND_DATA* lpFindFileData);
BOOL FindNextFileW(HANDLE hFindFile, WIN32_FIND_DATAW* lpFindFileData);
BOOL GetExitCodeThread(HANDLE hThread, DWORD *lpExitCode);
DWORD GetLastError();
DWORD GetFileAttributesA(in char *lpFileName);
DWORD GetFileAttributesW(in wchar *lpFileName);
DWORD GetFileSize(HANDLE hFile, DWORD *lpFileSizeHigh);
BOOL CopyFileA(LPCSTR lpExistingFileName, LPCSTR lpNewFileName, BOOL
bFailIfExists);
BOOL CopyFileW(LPCWSTR lpExistingFileName, LPCWSTR lpNewFileName, BOOL
bFailIfExists);
BOOL MoveFileA(in char *from, in char *to);
BOOL MoveFileW(LPCWSTR lpExistingFileName, LPCWSTR lpNewFileName);
BOOL ReadFile(HANDLE hFile, void *lpBuffer, DWORD nNumberOfBytesToRead,
DWORD *lpNumberOfBytesRead, OVERLAPPED *lpOverlapped);
DWORD SetFilePointer(HANDLE hFile, LONG lDistanceToMove,
LONG *lpDistanceToMoveHigh, DWORD dwMoveMethod);
BOOL WriteFile(HANDLE hFile, in void *lpBuffer, DWORD nNumberOfBytesToWrite,
DWORD *lpNumberOfBytesWritten, OVERLAPPED *lpOverlapped);
DWORD GetModuleFileNameA(HMODULE hModule, LPSTR lpFilename, DWORD nSize);
HANDLE GetStdHandle(DWORD nStdHandle);
BOOL SetStdHandle(DWORD nStdHandle, HANDLE hHandle);
}

struct MEMORYSTATUS {
DWORD dwLength;
DWORD dwMemoryLoad;
DWORD dwTotalPhys;
DWORD dwAvailPhys;
DWORD dwTotalPageFile;
DWORD dwAvailPageFile;
DWORD dwTotalVirtual;
DWORD dwAvailVirtual;
};
alias MEMORYSTATUS *LPMEMORYSTATUS;

HMODULE LoadLibraryA(LPCSTR lpLibFileName);
FARPROC GetProcAddress(HMODULE hModule, LPCSTR lpProcName);
DWORD GetVersion();
BOOL FreeLibrary(HMODULE hLibModule);
void FreeLibraryAndExitThread(HMODULE hLibModule, DWORD dwExitCode);
BOOL DisableThreadLibraryCalls(HMODULE hLibModule);

//
// Registry Specific Access Rights.
//

enum
{
KEY_QUERY_VALUE = 0x0001,
KEY_SET_VALUE = 0x0002,
KEY_CREATE_SUB_KEY = 0x0004,
KEY_ENUMERATE_SUB_KEYS = 0x0008,
KEY_NOTIFY = 0x0010,
```

```
KEY_CREATE_LINK = 0x0020,

KEY_READ = cast(int)((STANDARD_RIGHTS_READ | KEY_QUERY_VALUE |
KEY_ENUMERATE_SUB_KEYS | KEY_NOTIFY) & ~SYNCHRONIZE),
KEY_WRITE = cast(int)((STANDARD_RIGHTS_WRITE | KEY_SET_VALUE |
KEY_CREATE_SUB_KEY) & ~SYNCHRONIZE),
KEY_EXECUTE = cast(int)(KEY_READ & ~SYNCHRONIZE),
KEY_ALL_ACCESS = cast(int)((STANDARD_RIGHTS_ALL | KEY_QUERY_VALUE |
KEY_SET_VALUE | KEY_CREATE_SUB_KEY | KEY_ENUMERATE_SUB_KEYS | KEY_NOTIFY |
KEY_CREATE_LINK) & ~SYNCHRONIZE),
}

//
// Key creation/open disposition
//

enum : int
{
REG_CREATED_NEW_KEY = 0x00000001, // New Registry Key created
REG_OPENED_EXISTING_KEY = 0x00000002, // Existing Key opened
}

//
//
// Predefined Value Types.
//
enum
{
REG_NONE = 0, // No value type
REG_SZ = 1, // Unicode nul terminated string
REG_EXPAND_SZ = 2, // Unicode nul terminated string
// (with environment variable references)
REG_BINARY = 3, // Free form binary
REG_DWORD = 4, // 32-bit number
REG_DWORD_LITTLE_ENDIAN = 4, // 32-bit number (same as REG_DWORD)
REG_DWORD_BIG_ENDIAN = 5, // 32-bit number
REG_LINK = 6, // Symbolic Link (unicode)
REG_MULTI_SZ = 7, // Multiple Unicode strings
REG_RESOURCE_LIST = 8, // Resource list in the resource map
REG_FULL_RESOURCE_DESCRIPTOR = 9, // Resource list in the hardware
description
REG_RESOURCE_REQUIREMENTS_LIST = 10,
REG_QWORD = 11,
REG_QWORD_LITTLE_ENDIAN = 11,
}

/*
* MessageBox() Flags
*/
enum
{
MB_OK = 0x00000000,
MB_OKCANCEL = 0x00000001,
MB_ABORTRETRYIGNORE = 0x00000002,
MB_YESNOCANCEL = 0x00000003,
MB_YESNO = 0x00000004,
MB_RETRYCANCEL = 0x00000005,


MB_ICONHAND = 0x00000010,
MB_ICONQUESTION = 0x00000020,
MB_ICONEXCLAMATION = 0x00000030,
MB_ICONASTERISK = 0x00000040,
```

```
MB_USERICON = 0x00000080,
MB_ICONWARNING = MB_ICONEXCLAMATION,
MB_ICONERROR = MB_ICONHAND,


MB_ICONINFORMATION = MB_ICONASTERISK,
MB_ICONSTOP = MB_ICONHAND,

MB_DEFBUTTON1 = 0x00000000,
MB_DEFBUTTON2 = 0x00000100,
MB_DEFBUTTON3 = 0x00000200,

MB_DEFBUTTON4 = 0x00000300,


MB_APPLMODAL = 0x00000000,
MB_SYSTEMMODAL = 0x00001000,
MB_TASKMODAL = 0x00002000,

MB_HELP = 0x00004000, // Help Button


MB_NOFOCUS = 0x00008000,
MB_SETFOREGROUND = 0x00010000,
MB_DEFAULT_DESKTOP_ONLY = 0x00020000,


MB_TOPMOST = 0x00040000,
MB_RIGHT = 0x00080000,
MB_RTLREADING = 0x00100000,


MB_TYPEMASK = 0x0000000F,
MB_ICONMASK = 0x000000F0,
MB_DEFMASK = 0x00000F00,
MB_MODEMASK = 0x00003000,
MB_MISCMASK = 0x0000C000,
}


int MessageBoxA(HWND hWnd, LPCSTR lpText, LPCSTR lpCaption, UINT uType);
int MessageBoxExA(HWND hWnd, LPCSTR lpText, LPCSTR lpCaption, UINT uType,
WORD wLanguageId);
int MessageBoxW(HWND hWnd, LPCWSTR lpText, LPCWSTR lpCaption, UINT uType);
int MessageBoxExW(HWND hWnd, LPCWSTR lpText, LPCWSTR lpCaption, UINT uType,
WORD wLanguageId);

const HKEY HKEY_CLASSES_ROOT = cast(HKEY)(0x80000000);
const HKEY HKEY_CURRENT_USER = cast(HKEY)(0x80000001);
const HKEY HKEY_LOCAL_MACHINE = cast(HKEY)(0x80000002);
const HKEY HKEY_USERS = cast(HKEY)(0x80000003);
const HKEY HKEY_PERFORMANCE_DATA = cast(HKEY)(0x80000004);
const HKEY HKEY_PERFORMANCE_TEXT = cast(HKEY)(0x80000050);
const HKEY HKEY_PERFORMANCE_NLSTEXT = cast(HKEY)(0x80000060);
const HKEY HKEY_CURRENT_CONFIG = cast(HKEY)(0x80000005);
const HKEY HKEY_DYN_DATA = cast(HKEY)(0x80000006);



enum
{
REG_OPTION_RESERVED = (0x00000000), // Parameter is reserved
```

```
REG_OPTION_NON_VOLATILE = (0x00000000), // Key is preserved
// when system is rebooted

REG_OPTION_VOLATILE = (0x00000001), // Key is not preserved
// when system is rebooted

REG_OPTION_CREATE_LINK = (0x00000002), // Created key is a
// symbolic link

REG_OPTION_BACKUP_RESTORE = (0x00000004), // open for backup or restore
// special access rules
// privilege required

REG_OPTION_OPEN_LINK = (0x00000008), // Open symbolic link

REG_LEGAL_OPTION = (REG_OPTION_RESERVED | REG_OPTION_NON_VOLATILE |
REG_OPTION_VOLATILE | REG_OPTION_CREATE_LINK | REG_OPTION_BACKUP_RESTORE |
REG_OPTION_OPEN_LINK),
}

extern (Windows) LONG RegDeleteKeyA(HKEY hKey, LPCSTR lpSubKey);
extern (Windows) LONG RegDeleteValueA(HKEY hKey, LPCSTR lpValueName);

extern (Windows) LONG RegEnumKeyExA(HKEY hKey, DWORD dwIndex, LPSTR lpName,
LPDWORD lpcbName, LPDWORD lpReserved, LPSTR lpClass, LPDWORD lpcbClass,
FILETIME* lpftLastWriteTime);
extern (Windows) LONG RegEnumValueA(HKEY hKey, DWORD dwIndex, LPSTR
lpValueName, LPDWORD lpcbValueName, LPDWORD lpReserved,
LPDWORD lpType, LPBYTE lpData, LPDWORD lpcbData);

extern (Windows) LONG RegCloseKey(HKEY hKey);
extern (Windows) LONG RegFlushKey(HKEY hKey);

extern (Windows) LONG RegOpenKeyA(HKEY hKey, LPCSTR lpSubKey, PHKEY
phkResult);
extern (Windows) LONG RegOpenKeyExA(HKEY hKey, LPCSTR lpSubKey, DWORD
ulOptions, REGSAM samDesired, PHKEY phkResult);

extern (Windows) LONG RegQueryInfoKeyA(HKEY hKey, LPSTR lpClass, LPDWORD
lpcbClass,
LPDWORD lpReserved, LPDWORD lpcSubKeys, LPDWORD lpcbMaxSubKeyLen, LPDWORD
lpcbMaxClassLen,
LPDWORD lpcValues, LPDWORD lpcbMaxValueNameLen, LPDWORD lpcbMaxValueLen,
LPDWORD lpcbSecurityDescriptor,
PFILETIME lpftLastWriteTime);

extern (Windows) LONG RegQueryValueA(HKEY hKey, LPCSTR lpSubKey, LPSTR
lpValue,
LPLONG lpcbValue);

extern (Windows) LONG RegCreateKeyExA(HKEY hKey, LPCSTR lpSubKey, DWORD
Reserved, LPSTR lpClass,
DWORD dwOptions, REGSAM samDesired, SECURITY_ATTRIBUTES*
lpSecurityAttributes,
PHKEY phkResult, LPDWORD lpdwDisposition);

extern (Windows) LONG RegSetValueExA(HKEY hKey, LPCSTR lpValueName, DWORD
Reserved, DWORD dwType, BYTE* lpData, DWORD cbData);

struct MEMORY_BASIC_INFORMATION {
PVOID BaseAddress;
PVOID AllocationBase;
DWORD AllocationProtect;
DWORD RegionSize;
```

```
DWORD State;
DWORD Protect;
DWORD Type;
}
alias MEMORY_BASIC_INFORMATION* PMEMORY_BASIC_INFORMATION;

enum
{
SECTION_QUERY = 0x0001,
SECTION_MAP_WRITE = 0x0002,
SECTION_MAP_READ = 0x0004,
SECTION_MAP_EXECUTE = 0x0008,
SECTION_EXTEND_SIZE = 0x0010,

SECTION_ALL_ACCESS = cast(int)(STANDARD_RIGHTS_REQUIRED|SECTION_QUERY|
SECTION_MAP_WRITE | SECTION_MAP_READ | SECTION_MAP_EXECUTE |
SECTION_EXTEND_SIZE),
PAGE_NOACCESS = 0x01,
PAGE_READONLY = 0x02,
PAGE_READWRITE = 0x04,
PAGE_WRITECOPY = 0x08,
PAGE_EXECUTE = 0x10,
PAGE_EXECUTE_READ = 0x20,
PAGE_EXECUTE_READWRITE = 0x40,
PAGE_EXECUTE_WRITECOPY = 0x80,
PAGE_GUARD = 0x100,
PAGE_NOCACHE = 0x200,
MEM_COMMIT = 0x1000,
MEM_RESERVE = 0x2000,
MEM_DECOMMIT = 0x4000,
MEM_RELEASE = 0x8000,
MEM_FREE = 0x10000,
MEM_PRIVATE = 0x20000,
MEM_MAPPED = 0x40000,
MEM_RESET = 0x80000,
MEM_TOP_DOWN = 0x100000,
SEC_FILE = 0x800000,
SEC_IMAGE = 0x1000000,
SEC_RESERVE = 0x4000000,
SEC_COMMIT = 0x8000000,
SEC_NOCACHE = 0x10000000,
MEM_IMAGE = SEC_IMAGE,
}

enum
{
FILE_MAP_COPY = SECTION_QUERY,
FILE_MAP_WRITE = SECTION_MAP_WRITE,
FILE_MAP_READ = SECTION_MAP_READ,
FILE_MAP_ALL_ACCESS = SECTION_ALL_ACCESS,
}


//
// Define access rights to files and directories
//

//
// The FILE_READ_DATA and FILE_WRITE_DATA constants are also defined in
// devioctl.h as FILE_READ_ACCESS and FILE_WRITE_ACCESS. The values for these
// constants *MUST* always be in sync.
// The values are redefined in devioctl.h because they must be available to
// both DOS and NT.
//
```

```
enum
{
FILE_READ_DATA = ( 0x0001 ), // file & pipe
FILE_LIST_DIRECTORY = ( 0x0001 ), // directory

FILE_WRITE_DATA = ( 0x0002 ), // file & pipe
FILE_ADD_FILE = ( 0x0002 ), // directory

FILE_APPEND_DATA = ( 0x0004 ), // file
FILE_ADD_SUBDIRECTORY = ( 0x0004 ), // directory
FILE_CREATE_PIPE_INSTANCE = ( 0x0004 ), // named pipe

FILE_READ_EA = ( 0x0008 ), // file & directory

FILE_WRITE_EA = ( 0x0010 ), // file & directory

FILE_EXECUTE = ( 0x0020 ), // file
FILE_TRAVERSE = ( 0x0020 ), // directory

FILE_DELETE_CHILD = ( 0x0040 ), // directory

FILE_READ_ATTRIBUTES = ( 0x0080 ), // all

FILE_WRITE_ATTRIBUTES = ( 0x0100 ), // all

FILE_ALL_ACCESS = cast(int)(STANDARD_RIGHTS_REQUIRED | SYNCHRONIZE | 0x1FF),

FILE_GENERIC_READ = cast(int)(STANDARD_RIGHTS_READ | FILE_READ_DATA |
FILE_READ_ATTRIBUTES | FILE_READ_EA | SYNCHRONIZE),

FILE_GENERIC_WRITE = cast(int)(STANDARD_RIGHTS_WRITE | FILE_WRITE_DATA |
FILE_WRITE_ATTRIBUTES | FILE_WRITE_EA | FILE_APPEND_DATA | SYNCHRONIZE),

FILE_GENERIC_EXECUTE = cast(int)(STANDARD_RIGHTS_EXECUTE |
FILE_READ_ATTRIBUTES | FILE_EXECUTE | SYNCHRONIZE),
}

extern (Windows)
{
BOOL FreeResource(HGLOBAL hResData);
LPVOID LockResource(HGLOBAL hResData);
BOOL GlobalUnlock(HGLOBAL hMem);
HGLOBAL GlobalFree(HGLOBAL hMem);
UINT GlobalCompact(DWORD dwMinFree);
void GlobalFix(HGLOBAL hMem);
void GlobalUnfix(HGLOBAL hMem);
LPVOID GlobalWire(HGLOBAL hMem);
BOOL GlobalUnWire(HGLOBAL hMem);
void GlobalMemoryStatus(LPMEMORYSTATUS lpBuffer);
HLOCAL LocalAlloc(UINT uFlags, UINT uBytes);
HLOCAL LocalReAlloc(HLOCAL hMem, UINT uBytes, UINT uFlags);
LPVOID LocalLock(HLOCAL hMem);
HLOCAL LocalHandle(LPCVOID pMem);
BOOL LocalUnlock(HLOCAL hMem);
UINT LocalSize(HLOCAL hMem);
UINT LocalFlags(HLOCAL hMem);
HLOCAL LocalFree(HLOCAL hMem);
UINT LocalShrink(HLOCAL hMem, UINT cbNewSize);
UINT LocalCompact(UINT uMinFree);
BOOL FlushInstructionCache(HANDLE hProcess, LPCVOID lpBaseAddress, DWORD
dwSize);
LPVOID VirtualAlloc(LPVOID lpAddress, DWORD dwSize, DWORD flAllocationType,
DWORD flProtect);
```

```
BOOL VirtualFree(LPVOID lpAddress, DWORD dwSize, DWORD dwFreeType);
BOOL VirtualProtect(LPVOID lpAddress, DWORD dwSize, DWORD flNewProtect,
PDWORD lpflOldProtect);
DWORD VirtualQuery(LPCVOID lpAddress, PMEMORY_BASIC_INFORMATION lpBuffer,
DWORD dwLength);
LPVOID VirtualAllocEx(HANDLE hProcess, LPVOID lpAddress, DWORD dwSize, DWORD
flAllocationType, DWORD flProtect);
BOOL VirtualFreeEx(HANDLE hProcess, LPVOID lpAddress, DWORD dwSize, DWORD
dwFreeType);
BOOL VirtualProtectEx(HANDLE hProcess, LPVOID lpAddress, DWORD dwSize, DWORD
flNewProtect, PDWORD lpflOldProtect);
DWORD VirtualQueryEx(HANDLE hProcess, LPCVOID lpAddress,
PMEMORY_BASIC_INFORMATION lpBuffer, DWORD dwLength);
}

struct SYSTEMTIME
{
WORD wYear;
WORD wMonth;
WORD wDayOfWeek;
WORD wDay;
WORD wHour;
WORD wMinute;
WORD wSecond;
WORD wMilliseconds;
}

struct TIME_ZONE_INFORMATION {
LONG Bias;
WCHAR StandardName[ 32 ];
SYSTEMTIME StandardDate;
LONG StandardBias;
WCHAR DaylightName[ 32 ];
SYSTEMTIME DaylightDate;
LONG DaylightBias;
}

enum
{
TIME_ZONE_ID_UNKNOWN = 0,
TIME_ZONE_ID_STANDARD = 1,
TIME_ZONE_ID_DAYLIGHT = 2,
}

extern (Windows) void GetSystemTime(SYSTEMTIME* lpSystemTime);
extern (Windows) BOOL GetFileTime(HANDLE hFile, FILETIME *lpCreationTime,
FILETIME *lpLastAccessTime, FILETIME *lpLastWriteTime);
extern (Windows) void GetSystemTimeAsFileTime(FILETIME*
lpSystemTimeAsFileTime);
extern (Windows) BOOL SetSystemTime(SYSTEMTIME* lpSystemTime);
extern (Windows) BOOL SetFileTime(HANDLE hFile, in FILETIME *lpCreationTime,
in FILETIME *lpLastAccessTime, in FILETIME *lpLastWriteTime);
extern (Windows) void GetLocalTime(SYSTEMTIME* lpSystemTime);
extern (Windows) BOOL SetLocalTime(SYSTEMTIME* lpSystemTime);
extern (Windows) BOOL SystemTimeToTzSpecificLocalTime(TIME_ZONE_INFORMATION*
lpTimeZoneInformation, SYSTEMTIME* lpUniversalTime, SYSTEMTIME* lpLocalTime);
extern (Windows) DWORD GetTimeZoneInformation(TIME_ZONE_INFORMATION*
lpTimeZoneInformation);
extern (Windows) BOOL SetTimeZoneInformation(TIME_ZONE_INFORMATION*
lpTimeZoneInformation);

extern (Windows) BOOL SystemTimeToFileTime(in SYSTEMTIME *lpSystemTime,
FILETIME* lpFileTime);
```

```
extern (Windows) BOOL FileTimeToLocalFileTime(in FILETIME *lpFileTime,
FILETIME* lpLocalFileTime);
extern (Windows) BOOL LocalFileTimeToFileTime(in FILETIME *lpLocalFileTime,
FILETIME* lpFileTime);
extern (Windows) BOOL FileTimeToSystemTime(in FILETIME *lpFileTime,
SYSTEMTIME* lpSystemTime);
extern (Windows) LONG CompareFileTime(in FILETIME *lpFileTime1, in FILETIME
*lpFileTime2);
extern (Windows) BOOL FileTimeToDosDateTime(in FILETIME *lpFileTime, WORD*
lpFatDate, WORD* lpFatTime);
extern (Windows) BOOL DosDateTimeToFileTime(WORD wFatDate, WORD wFatTime,
FILETIME* lpFileTime);
extern (Windows) DWORD GetTickCount();
extern (Windows) BOOL SetSystemTimeAdjustment(DWORD dwTimeAdjustment, BOOL
bTimeAdjustmentDisabled);
extern (Windows) BOOL GetSystemTimeAdjustment(DWORD* lpTimeAdjustment, DWORD*
lpTimeIncrement, BOOL* lpTimeAdjustmentDisabled);
extern (Windows) DWORD FormatMessageA(DWORD dwFlags, LPCVOID lpSource, DWORD
dwMessageId, DWORD dwLanguageId, LPSTR lpBuffer, DWORD nSize, void*
*Arguments);extern (Windows) DWORD FormatMessageW(DWORD dwFlags, LPCVOID
lpSource, DWORD dwMessageId, DWORD dwLanguageId, LPWSTR lpBuffer, DWORD
nSize, void* *Arguments);

enum
{
FORMAT_MESSAGE_ALLOCATE_BUFFER = 0x00000100,
FORMAT_MESSAGE_IGNORE_INSERTS = 0x00000200,
FORMAT_MESSAGE_FROM_STRING = 0x00000400,
FORMAT_MESSAGE_FROM_HMODULE = 0x00000800,
FORMAT_MESSAGE_FROM_SYSTEM = 0x00001000,
FORMAT_MESSAGE_ARGUMENT_ARRAY = 0x00002000,
FORMAT_MESSAGE_MAX_WIDTH_MASK = 0x000000FF,
};


//
// Language IDs.
//
// The following two combinations of primary language ID and
// sublanguage ID have special semantics:
//
// Primary Language ID Sublanguage ID Result
// ------------------- --------------- ------------------------
// LANG_NEUTRAL SUBLANG_NEUTRAL Language neutral
// LANG_NEUTRAL SUBLANG_DEFAULT User default language
// LANG_NEUTRAL SUBLANG_SYS_DEFAULT System default language
//

//
// Primary language IDs.
//

enum
{
LANG_NEUTRAL = 0x00,

LANG_AFRIKAANS = 0x36,
LANG_ALBANIAN = 0x1c,
LANG_ARABIC = 0x01,
LANG_BASQUE = 0x2d,
LANG_BELARUSIAN = 0x23,
LANG_BULGARIAN = 0x02,
LANG_CATALAN = 0x03,
LANG_CHINESE = 0x04,
```

```
LANG_CROATIAN = 0x1a,
LANG_CZECH = 0x05,
LANG_DANISH = 0x06,
LANG_DUTCH = 0x13,
LANG_ENGLISH = 0x09,
LANG_ESTONIAN = 0x25,
LANG_FAEROESE = 0x38,
LANG_FARSI = 0x29,
LANG_FINNISH = 0x0b,
LANG_FRENCH = 0x0c,
LANG_GERMAN = 0x07,
LANG_GREEK = 0x08,
LANG_HEBREW = 0x0d,
LANG_HUNGARIAN = 0x0e,
LANG_ICELANDIC = 0x0f,
LANG_INDONESIAN = 0x21,
LANG_ITALIAN = 0x10,
LANG_JAPANESE = 0x11,
LANG_KOREAN = 0x12,
LANG_LATVIAN = 0x26,
LANG_LITHUANIAN = 0x27,
LANG_NORWEGIAN = 0x14,
LANG_POLISH = 0x15,
LANG_PORTUGUESE = 0x16,
LANG_ROMANIAN = 0x18,
LANG_RUSSIAN = 0x19,
LANG_SERBIAN = 0x1a,
LANG_SLOVAK = 0x1b,
LANG_SLOVENIAN = 0x24,
LANG_SPANISH = 0x0a,
LANG_SWEDISH = 0x1d,
LANG_THAI = 0x1e,
LANG_TURKISH = 0x1f,
LANG_UKRAINIAN = 0x22,
LANG_VIETNAMESE = 0x2a,
}
//
// Sublanguage IDs.
//
// The name immediately following SUBLANG_ dictates which primary
// language ID that sublanguage ID can be combined with to form a
// valid language ID.
//
enum
{
SUBLANG_NEUTRAL = 0x00, // language neutral
SUBLANG_DEFAULT = 0x01, // user default
SUBLANG_SYS_DEFAULT = 0x02, // system default

SUBLANG_ARABIC_SAUDI_ARABIA = 0x01, // Arabic (Saudi Arabia)
SUBLANG_ARABIC_IRAQ = 0x02, // Arabic (Iraq)
SUBLANG_ARABIC_EGYPT = 0x03, // Arabic (Egypt)
SUBLANG_ARABIC_LIBYA = 0x04, // Arabic (Libya)
SUBLANG_ARABIC_ALGERIA = 0x05, // Arabic (Algeria)
SUBLANG_ARABIC_MOROCCO = 0x06, // Arabic (Morocco)
SUBLANG_ARABIC_TUNISIA = 0x07, // Arabic (Tunisia)
SUBLANG_ARABIC_OMAN = 0x08, // Arabic (Oman)
SUBLANG_ARABIC_YEMEN = 0x09, // Arabic (Yemen)
SUBLANG_ARABIC_SYRIA = 0x0a, // Arabic (Syria)
SUBLANG_ARABIC_JORDAN = 0x0b, // Arabic (Jordan)
SUBLANG_ARABIC_LEBANON = 0x0c, // Arabic (Lebanon)
SUBLANG_ARABIC_KUWAIT = 0x0d, // Arabic (Kuwait)
SUBLANG_ARABIC_UAE = 0x0e, // Arabic (U.A.E)
SUBLANG_ARABIC_BAHRAIN = 0x0f, // Arabic (Bahrain)
```

```
SUBLANG_ARABIC_QATAR = 0x10, // Arabic (Qatar)
SUBLANG_CHINESE_TRADITIONAL = 0x01, // Chinese (Taiwan)
SUBLANG_CHINESE_SIMPLIFIED = 0x02, // Chinese (PR China)
SUBLANG_CHINESE_HONGKONG = 0x03, // Chinese (Hong Kong)
SUBLANG_CHINESE_SINGAPORE = 0x04, // Chinese (Singapore)
SUBLANG_DUTCH = 0x01, // Dutch
SUBLANG_DUTCH_BELGIAN = 0x02, // Dutch (Belgian)
SUBLANG_ENGLISH_US = 0x01, // English (USA)
SUBLANG_ENGLISH_UK = 0x02, // English (UK)
SUBLANG_ENGLISH_AUS = 0x03, // English (Australian)
SUBLANG_ENGLISH_CAN = 0x04, // English (Canadian)
SUBLANG_ENGLISH_NZ = 0x05, // English (New Zealand)
SUBLANG_ENGLISH_EIRE = 0x06, // English (Irish)
SUBLANG_ENGLISH_SOUTH_AFRICA = 0x07, // English (South Africa)
SUBLANG_ENGLISH_JAMAICA = 0x08, // English (Jamaica)
SUBLANG_ENGLISH_CARIBBEAN = 0x09, // English (Caribbean)
SUBLANG_ENGLISH_BELIZE = 0x0a, // English (Belize)
SUBLANG_ENGLISH_TRINIDAD = 0x0b, // English (Trinidad)
SUBLANG_FRENCH = 0x01, // French
SUBLANG_FRENCH_BELGIAN = 0x02, // French (Belgian)
SUBLANG_FRENCH_CANADIAN = 0x03, // French (Canadian)
SUBLANG_FRENCH_SWISS = 0x04, // French (Swiss)
SUBLANG_FRENCH_LUXEMBOURG = 0x05, // French (Luxembourg)
SUBLANG_GERMAN = 0x01, // German
SUBLANG_GERMAN_SWISS = 0x02, // German (Swiss)
SUBLANG_GERMAN_AUSTRIAN = 0x03, // German (Austrian)
SUBLANG_GERMAN_LUXEMBOURG = 0x04, // German (Luxembourg)
SUBLANG_GERMAN_LIECHTENSTEIN = 0x05, // German (Liechtenstein)
SUBLANG_ITALIAN = 0x01, // Italian
SUBLANG_ITALIAN_SWISS = 0x02, // Italian (Swiss)
SUBLANG_KOREAN = 0x01, // Korean (Extended Wansung)
SUBLANG_KOREAN_JOHAB = 0x02, // Korean (Johab)
SUBLANG_NORWEGIAN_BOKMAL = 0x01, // Norwegian (Bokmal)
SUBLANG_NORWEGIAN_NYNORSK = 0x02, // Norwegian (Nynorsk)
SUBLANG_PORTUGUESE = 0x02, // Portuguese
SUBLANG_PORTUGUESE_BRAZILIAN = 0x01, // Portuguese (Brazilian)
SUBLANG_SERBIAN_LATIN = 0x02, // Serbian (Latin)
SUBLANG_SERBIAN_CYRILLIC = 0x03, // Serbian (Cyrillic)
SUBLANG_SPANISH = 0x01, // Spanish (Castilian)
SUBLANG_SPANISH_MEXICAN = 0x02, // Spanish (Mexican)
SUBLANG_SPANISH_MODERN = 0x03, // Spanish (Modern)
SUBLANG_SPANISH_GUATEMALA = 0x04, // Spanish (Guatemala)
SUBLANG_SPANISH_COSTA_RICA = 0x05, // Spanish (Costa Rica)
SUBLANG_SPANISH_PANAMA = 0x06, // Spanish (Panama)
SUBLANG_SPANISH_DOMINICAN_REPUBLIC = 0x07, // Spanish (Dominican Republic)
SUBLANG_SPANISH_VENEZUELA = 0x08, // Spanish (Venezuela)
SUBLANG_SPANISH_COLOMBIA = 0x09, // Spanish (Colombia)
SUBLANG_SPANISH_PERU = 0x0a, // Spanish (Peru)
SUBLANG_SPANISH_ARGENTINA = 0x0b, // Spanish (Argentina)
SUBLANG_SPANISH_ECUADOR = 0x0c, // Spanish (Ecuador)
SUBLANG_SPANISH_CHILE = 0x0d, // Spanish (Chile)
SUBLANG_SPANISH_URUGUAY = 0x0e, // Spanish (Uruguay)
SUBLANG_SPANISH_PARAGUAY = 0x0f, // Spanish (Paraguay)
SUBLANG_SPANISH_BOLIVIA = 0x10, // Spanish (Bolivia)
SUBLANG_SPANISH_EL_SALVADOR = 0x11, // Spanish (El Salvador)
SUBLANG_SPANISH_HONDURAS = 0x12, // Spanish (Honduras)
SUBLANG_SPANISH_NICARAGUA = 0x13, // Spanish (Nicaragua)
SUBLANG_SPANISH_PUERTO_RICO = 0x14, // Spanish (Puerto Rico)
SUBLANG_SWEDISH = 0x01, // Swedish
SUBLANG_SWEDISH_FINLAND = 0x02, // Swedish (Finland)
}
//
// Sorting IDs.
//
```

```
enum
{
SORT_DEFAULT = 0x0, // sorting default

SORT_JAPANESE_XJIS = 0x0, // Japanese XJIS order
SORT_JAPANESE_UNICODE = 0x1, // Japanese Unicode order

SORT_CHINESE_BIG5 = 0x0, // Chinese BIG5 order
SORT_CHINESE_PRCP = 0x0, // PRC Chinese Phonetic order
SORT_CHINESE_UNICODE = 0x1, // Chinese Unicode order
SORT_CHINESE_PRC = 0x2, // PRC Chinese Stroke Count order

SORT_KOREAN_KSC = 0x0, // Korean KSC order
SORT_KOREAN_UNICODE = 0x1, // Korean Unicode order

SORT_GERMAN_PHONE_BOOK = 0x1, // German Phone Book order
}

// end_r_winnt

//
// A language ID is a 16 bit value which is the combination of a
// primary language ID and a secondary language ID. The bits are
// allocated as follows:
//
// +-----------------------+-------------------------+
// | Sublanguage ID | Primary Language ID |
// +-----------------------+-------------------------+
// 15 10 9 0 bit
//
//
// Language ID creation/extraction macros:
//
// MAKELANGID - construct language id from a primary language id and
// a sublanguage id.
// PRIMARYLANGID - extract primary language id from a language id.
// SUBLANGID - extract sublanguage id from a language id.
//

int MAKELANGID(int p, int s) { return ((cast(WORD)s) << 10) | cast(WORD)p; }
WORD PRIMARYLANGID(int lgid) { return cast(WORD)(lgid & 0x3ff); }
WORD SUBLANGID(int lgid) { return cast(WORD)(lgid >> 10); }


struct FLOATING_SAVE_AREA {
DWORD ControlWord;
DWORD StatusWord;
DWORD TagWord;
DWORD ErrorOffset;
DWORD ErrorSelector;
DWORD DataOffset;
DWORD DataSelector;
BYTE RegisterArea[80 ];
DWORD Cr0NpxState;
}

enum
{
SIZE_OF_80387_REGISTERS = 80,
//
// The following flags control the contents of the CONTEXT structure.
//
CONTEXT_i386 = 0x00010000, // this assumes that i386 and
```

```
CONTEXT_i486 = 0x00010000, // i486 have identical context records

CONTEXT_CONTROL = (CONTEXT_i386 | 0x00000001), // SS:SP, CS:IP, FLAGS, BP
CONTEXT_INTEGER = (CONTEXT_i386 | 0x00000002), // AX, BX, CX, DX, SI, DI
CONTEXT_SEGMENTS = (CONTEXT_i386 | 0x00000004), // DS, ES, FS, GS
CONTEXT_FLOATING_POINT = (CONTEXT_i386 | 0x00000008), // 387 state
CONTEXT_DEBUG_REGISTERS = (CONTEXT_i386 | 0x00000010), // DB 0-3,6,7

CONTEXT_FULL = (CONTEXT_CONTROL | CONTEXT_INTEGER | CONTEXT_SEGMENTS),
}

struct CONTEXT
{

//
// The flags values within this flag control the contents of
// a CONTEXT record.
//
// If the context record is used as an input parameter, then
// for each portion of the context record controlled by a flag
// whose value is set, it is assumed that that portion of the
// context record contains valid context. If the context record
// is being used to modify a threads context, then only that
// portion of the threads context will be modified.
//
// If the context record is used as an IN OUT parameter to capture
// the context of a thread, then only those portions of the thread's
// context corresponding to set flags will be returned.
//
// The context record is never used as an OUT only parameter.
//

DWORD ContextFlags;

//
// This section is specified/returned if CONTEXT_DEBUG_REGISTERS is
// set in ContextFlags. Note that CONTEXT_DEBUG_REGISTERS is NOT
// included in CONTEXT_FULL.
//

DWORD Dr0;
DWORD Dr1;
DWORD Dr2;
DWORD Dr3;
DWORD Dr6;
DWORD Dr7;

//
// This section is specified/returned if the
// ContextFlags word contians the flag CONTEXT_FLOATING_POINT.
//

FLOATING_SAVE_AREA FloatSave;

//
// This section is specified/returned if the
// ContextFlags word contians the flag CONTEXT_SEGMENTS.
//

DWORD SegGs;
DWORD SegFs;
DWORD SegEs;
DWORD SegDs;
```

```
//
// This section is specified/returned if the
// ContextFlags word contians the flag CONTEXT_INTEGER.
//

DWORD Edi;
DWORD Esi;
DWORD Ebx;
DWORD Edx;
DWORD Ecx;
DWORD Eax;

//
// This section is specified/returned if the
// ContextFlags word contians the flag CONTEXT_CONTROL.
//

DWORD Ebp;
DWORD Eip;
DWORD SegCs; // MUST BE SANITIZED
DWORD EFlags; // MUST BE SANITIZED
DWORD Esp;
DWORD SegSs;
}

enum ADDRESS_MODE
{
AddrMode1616,
AddrMode1632,
AddrModeReal,
AddrModeFlat
}

struct ADDRESS
{
DWORD Offset;
WORD Segment;
ADDRESS_MODE Mode;
}

struct ADDRESS64
{
DWORD64 Offset;
WORD Segment;
ADDRESS_MODE Mode;
}

struct KDHELP
{
DWORD Thread;
DWORD ThCallbackStack;
DWORD NextCallback;
DWORD FramePointer;
DWORD KiCallUserMode;
DWORD KeUserCallbackDispatcher;
DWORD SystemRangeStart;
DWORD ThCallbackBStore;
DWORD KiUserExceptionDispatcher;
DWORD StackBase;
DWORD StackLimit;
DWORD[5] Reserved;
}

struct KDHELP64
```

```
{
DWORD64 Thread;
DWORD ThCallbackStack;
DWORD ThCallbackBStore;
DWORD NextCallback;
DWORD FramePointer;
DWORD64 KiCallUserMode;
DWORD64 KeUserCallbackDispatcher;
DWORD64 SystemRangeStart;
DWORD64 KiUserExceptionDispatcher;
DWORD64 StackBase;
DWORD64 StackLimit;
DWORD64[5] Reserved;
}

struct STACKFRAME
{
ADDRESS AddrPC;
ADDRESS AddrReturn;
ADDRESS AddrFrame;
ADDRESS AddrStack;
PVOID FuncTableEntry;
DWORD[4] Params;
BOOL Far;
BOOL Virtual;
DWORD[3] Reserved;
KDHELP KdHelp;
ADDRESS AddrBStore;
}

struct STACKFRAME64
{
ADDRESS64 AddrPC;
ADDRESS64 AddrReturn;
ADDRESS64 AddrFrame;
ADDRESS64 AddrStack;
ADDRESS64 AddrBStore;
PVOID FuncTableEntry;
DWORD64[4] Params;
BOOL Far;
BOOL Virtual;
DWORD64[3] Reserved;
KDHELP64 KdHelp;
}

enum
{
THREAD_BASE_PRIORITY_LOWRT = 15, // value that gets a thread to LowRealtime-1
THREAD_BASE_PRIORITY_MAX = 2, // maximum thread base priority boost
THREAD_BASE_PRIORITY_MIN = -2, // minimum thread base priority boost
THREAD_BASE_PRIORITY_IDLE = -15, // value that gets a thread to idle

THREAD_PRIORITY_LOWEST = THREAD_BASE_PRIORITY_MIN,
THREAD_PRIORITY_BELOW_NORMAL = (THREAD_PRIORITY_LOWEST+1),
THREAD_PRIORITY_NORMAL = 0,
THREAD_PRIORITY_HIGHEST = THREAD_BASE_PRIORITY_MAX,
THREAD_PRIORITY_ABOVE_NORMAL = (THREAD_PRIORITY_HIGHEST-1),
THREAD_PRIORITY_ERROR_RETURN = int.max,

THREAD_PRIORITY_TIME_CRITICAL = THREAD_BASE_PRIORITY_LOWRT,
THREAD_PRIORITY_IDLE = THREAD_BASE_PRIORITY_IDLE,
}

extern (Windows) HANDLE GetCurrentThread();
```

```
extern (Windows) BOOL GetProcessTimes(HANDLE hProcess, LPFILETIME
lpCreationTime, LPFILETIME lpExitTime, LPFILETIME lpKernelTime, LPFILETIME
lpUserTime);
extern (Windows) HANDLE GetCurrentProcess();
extern (Windows) DWORD GetCurrentProcessId();
extern (Windows) BOOL DuplicateHandle (HANDLE sourceProcess, HANDLE
sourceThread,
HANDLE targetProcessHandle, HANDLE *targetHandle, DWORD access,
BOOL inheritHandle, DWORD options);
extern (Windows) DWORD GetCurrentThreadId();
extern (Windows) BOOL SetThreadPriority(HANDLE hThread, int nPriority);
extern (Windows) BOOL SetThreadPriorityBoost(HANDLE hThread, BOOL
bDisablePriorityBoost);
extern (Windows) BOOL GetThreadPriorityBoost(HANDLE hThread, PBOOL
pDisablePriorityBoost);
extern (Windows) BOOL GetThreadTimes(HANDLE hThread, LPFILETIME
lpCreationTime, LPFILETIME lpExitTime, LPFILETIME lpKernelTime, LPFILETIME
lpUserTime);
extern (Windows) int GetThreadPriority(HANDLE hThread);
extern (Windows) BOOL GetThreadContext(HANDLE hThread, CONTEXT* lpContext);
extern (Windows) BOOL SetThreadContext(HANDLE hThread, CONTEXT* lpContext);
extern (Windows) DWORD SuspendThread(HANDLE hThread);
extern (Windows) DWORD ResumeThread(HANDLE hThread);
extern (Windows) DWORD WaitForSingleObject(HANDLE hHandle, DWORD
dwMilliseconds);
extern (Windows) DWORD WaitForMultipleObjects(DWORD nCount, HANDLE
*lpHandles, BOOL bWaitAll, DWORD dwMilliseconds);
extern (Windows) void Sleep(DWORD dwMilliseconds);

// Synchronization

extern (Windows)
{
LONG InterlockedIncrement(LPLONG lpAddend);
LONG InterlockedDecrement(LPLONG lpAddend);
LONG InterlockedExchange(LPLONG Target, LONG Value);
LONG InterlockedExchangeAdd(LPLONG Addend, LONG Value);
PVOID InterlockedCompareExchange(PVOID *Destination, PVOID Exchange, PVOID
Comperand);

void InitializeCriticalSection(CRITICAL_SECTION * lpCriticalSection);
void EnterCriticalSection(CRITICAL_SECTION * lpCriticalSection);
BOOL TryEnterCriticalSection(CRITICAL_SECTION * lpCriticalSection);
void LeaveCriticalSection(CRITICAL_SECTION * lpCriticalSection);
void DeleteCriticalSection(CRITICAL_SECTION * lpCriticalSection);

}



extern (Windows) BOOL QueryPerformanceCounter(long* lpPerformanceCount);
extern (Windows) BOOL QueryPerformanceFrequency(long* lpFrequency);

enum
{
WM_NOTIFY = 0x004E,
WM_INPUTLANGCHANGEREQUEST = 0x0050,
WM_INPUTLANGCHANGE = 0x0051,
WM_TCARD = 0x0052,
WM_HELP = 0x0053,
WM_USERCHANGED = 0x0054,
WM_NOTIFYFORMAT = 0x0055,

NFR_ANSI = 1,
```

```
NFR_UNICODE = 2,
NF_QUERY = 3,
NF_REQUERY = 4,

WM_CONTEXTMENU = 0x007B,
WM_STYLECHANGING = 0x007C,
WM_STYLECHANGED = 0x007D,
WM_DISPLAYCHANGE = 0x007E,
WM_GETICON = 0x007F,
WM_SETICON = 0x0080,



WM_NCCREATE = 0x0081,
WM_NCDESTROY = 0x0082,
WM_NCCALCSIZE = 0x0083,
WM_NCHITTEST = 0x0084,
WM_NCPAINT = 0x0085,
WM_NCACTIVATE = 0x0086,
WM_GETDLGCODE = 0x0087,

WM_NCMOUSEMOVE = 0x00A0,
WM_NCLBUTTONDOWN = 0x00A1,
WM_NCLBUTTONUP = 0x00A2,
WM_NCLBUTTONDBLCLK = 0x00A3,
WM_NCRBUTTONDOWN = 0x00A4,
WM_NCRBUTTONUP = 0x00A5,
WM_NCRBUTTONDBLCLK = 0x00A6,
WM_NCMBUTTONDOWN = 0x00A7,
WM_NCMBUTTONUP = 0x00A8,
WM_NCMBUTTONDBLCLK = 0x00A9,

WM_KEYFIRST = 0x0100,
WM_KEYDOWN = 0x0100,
WM_KEYUP = 0x0101,
WM_CHAR = 0x0102,
WM_DEADCHAR = 0x0103,
WM_SYSKEYDOWN = 0x0104,
WM_SYSKEYUP = 0x0105,
WM_SYSCHAR = 0x0106,
WM_SYSDEADCHAR = 0x0107,
WM_KEYLAST = 0x0108,


WM_IME_STARTCOMPOSITION = 0x010D,
WM_IME_ENDCOMPOSITION = 0x010E,
WM_IME_COMPOSITION = 0x010F,
WM_IME_KEYLAST = 0x010F,


WM_INITDIALOG = 0x0110,
WM_COMMAND = 0x0111,
WM_SYSCOMMAND = 0x0112,
WM_TIMER = 0x0113,
WM_HSCROLL = 0x0114,
WM_VSCROLL = 0x0115,
WM_INITMENU = 0x0116,
WM_INITMENUPOPUP = 0x0117,
WM_MENUSELECT = 0x011F,
WM_MENUCHAR = 0x0120,
WM_ENTERIDLE = 0x0121,

WM_CTLCOLORMSGBOX = 0x0132,
WM_CTLCOLOREDIT = 0x0133,
```

```
WM_CTLCOLORLISTBOX = 0x0134,
WM_CTLCOLORBTN = 0x0135,
WM_CTLCOLORDLG = 0x0136,
WM_CTLCOLORSCROLLBAR = 0x0137,
WM_CTLCOLORSTATIC = 0x0138,



WM_MOUSEFIRST = 0x0200,
WM_MOUSEMOVE = 0x0200,
WM_LBUTTONDOWN = 0x0201,
WM_LBUTTONUP = 0x0202,
WM_LBUTTONDBLCLK = 0x0203,
WM_RBUTTONDOWN = 0x0204,
WM_RBUTTONUP = 0x0205,
WM_RBUTTONDBLCLK = 0x0206,
WM_MBUTTONDOWN = 0x0207,
WM_MBUTTONUP = 0x0208,
WM_MBUTTONDBLCLK = 0x0209,



WM_MOUSELAST = 0x0209,









WM_PARENTNOTIFY = 0x0210,
MENULOOP_WINDOW = 0,
MENULOOP_POPUP = 1,
WM_ENTERMENULOOP = 0x0211,
WM_EXITMENULOOP = 0x0212,


WM_NEXTMENU = 0x0213,
}

enum
{
/*
* Dialog Box Command IDs
*/
IDOK = 1,
IDCANCEL = 2,
IDABORT = 3,
IDRETRY = 4,
IDIGNORE = 5,
IDYES = 6,
IDNO = 7,

IDCLOSE = 8,
IDHELP = 9,


// end_r_winuser



/*
* Control Manager Structures and Definitions
```

```
*/


// begin_r_winuser

/*
* Edit Control Styles
*/
ES_LEFT = 0x0000,
ES_CENTER = 0x0001,
ES_RIGHT = 0x0002,
ES_MULTILINE = 0x0004,
ES_UPPERCASE = 0x0008,
ES_LOWERCASE = 0x0010,
ES_PASSWORD = 0x0020,
ES_AUTOVSCROLL = 0x0040,
ES_AUTOHSCROLL = 0x0080,
ES_NOHIDESEL = 0x0100,
ES_OEMCONVERT = 0x0400,
ES_READONLY = 0x0800,
ES_WANTRETURN = 0x1000,

ES_NUMBER = 0x2000,


// end_r_winuser



/*
* Edit Control Notification Codes
*/
EN_SETFOCUS = 0x0100,
EN_KILLFOCUS = 0x0200,
EN_CHANGE = 0x0300,
EN_UPDATE = 0x0400,
EN_ERRSPACE = 0x0500,
EN_MAXTEXT = 0x0501,
EN_HSCROLL = 0x0601,
EN_VSCROLL = 0x0602,


/* Edit control EM_SETMARGIN parameters */
EC_LEFTMARGIN = 0x0001,
EC_RIGHTMARGIN = 0x0002,
EC_USEFONTINFO = 0xffff,




// begin_r_winuser

/*
* Edit Control Messages
*/
EM_GETSEL = 0x00B0,
EM_SETSEL = 0x00B1,
EM_GETRECT = 0x00B2,
EM_SETRECT = 0x00B3,
EM_SETRECTNP = 0x00B4,
EM_SCROLL = 0x00B5,
EM_LINESCROLL = 0x00B6,
EM_SCROLLCARET = 0x00B7,
```

```
EM_GETMODIFY = 0x00B8,
EM_SETMODIFY = 0x00B9,
EM_GETLINECOUNT = 0x00BA,
EM_LINEINDEX = 0x00BB,
EM_SETHANDLE = 0x00BC,
EM_GETHANDLE = 0x00BD,
EM_GETTHUMB = 0x00BE,
EM_LINELENGTH = 0x00C1,
EM_REPLACESEL = 0x00C2,
EM_GETLINE = 0x00C4,
EM_LIMITTEXT = 0x00C5,
EM_CANUNDO = 0x00C6,
EM_UNDO = 0x00C7,
EM_FMTLINES = 0x00C8,
EM_LINEFROMCHAR = 0x00C9,
EM_SETTABSTOPS = 0x00CB,
EM_SETPASSWORDCHAR = 0x00CC,
EM_EMPTYUNDOBUFFER = 0x00CD,
EM_GETFIRSTVISIBLELINE = 0x00CE,
EM_SETREADONLY = 0x00CF,
EM_SETWORDBREAKPROC = 0x00D0,
EM_GETWORDBREAKPROC = 0x00D1,
EM_GETPASSWORDCHAR = 0x00D2,

EM_SETMARGINS = 0x00D3,
EM_GETMARGINS = 0x00D4,
EM_SETLIMITTEXT = EM_LIMITTEXT, /* ;win40 Name change */
EM_GETLIMITTEXT = 0x00D5,
EM_POSFROMCHAR = 0x00D6,
EM_CHARFROMPOS = 0x00D7,



// end_r_winuser


/*
* EDITWORDBREAKPROC code values
*/
WB_LEFT = 0,
WB_RIGHT = 1,
WB_ISDELIMITER = 2,

// begin_r_winuser

/*
* Button Control Styles
*/
BS_PUSHBUTTON = 0x00000000,
BS_DEFPUSHBUTTON = 0x00000001,
BS_CHECKBOX = 0x00000002,
BS_AUTOCHECKBOX = 0x00000003,
BS_RADIOBUTTON = 0x00000004,
BS_3STATE = 0x00000005,
BS_AUTO3STATE = 0x00000006,
BS_GROUPBOX = 0x00000007,
BS_USERBUTTON = 0x00000008,
BS_AUTORADIOBUTTON = 0x00000009,
BS_OWNERDRAW = 0x0000000B,
BS_LEFTTEXT = 0x00000020,

BS_TEXT = 0x00000000,
BS_ICON = 0x00000040,
BS_BITMAP = 0x00000080,
```

```
BS_LEFT = 0x00000100,
BS_RIGHT = 0x00000200,
BS_CENTER = 0x00000300,
BS_TOP = 0x00000400,
BS_BOTTOM = 0x00000800,
BS_VCENTER = 0x00000C00,
BS_PUSHLIKE = 0x00001000,
BS_MULTILINE = 0x00002000,
BS_NOTIFY = 0x00004000,
BS_FLAT = 0x00008000,
BS_RIGHTBUTTON = BS_LEFTTEXT,



/*
* User Button Notification Codes
*/
BN_CLICKED = 0,
BN_PAINT = 1,
BN_HILITE = 2,
BN_UNHILITE = 3,
BN_DISABLE = 4,
BN_DOUBLECLICKED = 5,

BN_PUSHED = BN_HILITE,
BN_UNPUSHED = BN_UNHILITE,
BN_DBLCLK = BN_DOUBLECLICKED,
BN_SETFOCUS = 6,
BN_KILLFOCUS = 7,

/*
* Button Control Messages
*/
BM_GETCHECK = 0x00F0,
BM_SETCHECK = 0x00F1,
BM_GETSTATE = 0x00F2,
BM_SETSTATE = 0x00F3,
BM_SETSTYLE = 0x00F4,

BM_CLICK = 0x00F5,
BM_GETIMAGE = 0x00F6,
BM_SETIMAGE = 0x00F7,

BST_UNCHECKED = 0x0000,
BST_CHECKED = 0x0001,
BST_INDETERMINATE = 0x0002,
BST_PUSHED = 0x0004,
BST_FOCUS = 0x0008,


/*
* Static Control Constants
*/
SS_LEFT = 0x00000000,
SS_CENTER = 0x00000001,
SS_RIGHT = 0x00000002,
SS_ICON = 0x00000003,
SS_BLACKRECT = 0x00000004,
SS_GRAYRECT = 0x00000005,
SS_WHITERECT = 0x00000006,
SS_BLACKFRAME = 0x00000007,
SS_GRAYFRAME = 0x00000008,
SS_WHITEFRAME = 0x00000009,
SS_USERITEM = 0x0000000A,
```

```
SS_SIMPLE = 0x0000000B,
SS_LEFTNOWORDWRAP = 0x0000000C,

SS_OWNERDRAW = 0x0000000D,
SS_BITMAP = 0x0000000E,
SS_ENHMETAFILE = 0x0000000F,
SS_ETCHEDHORZ = 0x00000010,
SS_ETCHEDVERT = 0x00000011,
SS_ETCHEDFRAME = 0x00000012,
SS_TYPEMASK = 0x0000001F,

SS_NOPREFIX = 0x00000080, /* Don't do "&" character translation */

SS_NOTIFY = 0x00000100,
SS_CENTERIMAGE = 0x00000200,
SS_RIGHTJUST = 0x00000400,
SS_REALSIZEIMAGE = 0x00000800,
SS_SUNKEN = 0x00001000,
SS_ENDELLIPSIS = 0x00004000,
SS_PATHELLIPSIS = 0x00008000,
SS_WORDELLIPSIS = 0x0000C000,
SS_ELLIPSISMASK = 0x0000C000,


// end_r_winuser


/*
* Static Control Mesages
*/
STM_SETICON = 0x0170,
STM_GETICON = 0x0171,

STM_SETIMAGE = 0x0172,
STM_GETIMAGE = 0x0173,
STN_CLICKED = 0,
STN_DBLCLK = 1,
STN_ENABLE = 2,
STN_DISABLE = 3,

STM_MSGMAX = 0x0174,
}


enum
{
/*
* Window Messages
*/

WM_NULL = 0x0000,
WM_CREATE = 0x0001,
WM_DESTROY = 0x0002,
WM_MOVE = 0x0003,
WM_SIZE = 0x0005,

WM_ACTIVATE = 0x0006,
/*
* WM_ACTIVATE state values
*/
WA_INACTIVE = 0,
WA_ACTIVE = 1,
WA_CLICKACTIVE = 2,
```

```
WM_SETFOCUS = 0x0007,
WM_KILLFOCUS = 0x0008,
WM_ENABLE = 0x000A,
WM_SETREDRAW = 0x000B,
WM_SETTEXT = 0x000C,
WM_GETTEXT = 0x000D,
WM_GETTEXTLENGTH = 0x000E,
WM_PAINT = 0x000F,
WM_CLOSE = 0x0010,
WM_QUERYENDSESSION = 0x0011,
WM_QUIT = 0x0012,
WM_QUERYOPEN = 0x0013,
WM_ERASEBKGND = 0x0014,
WM_SYSCOLORCHANGE = 0x0015,
WM_ENDSESSION = 0x0016,
WM_SHOWWINDOW = 0x0018,
WM_WININICHANGE = 0x001A,

WM_SETTINGCHANGE = WM_WININICHANGE,



WM_DEVMODECHANGE = 0x001B,
WM_ACTIVATEAPP = 0x001C,
WM_FONTCHANGE = 0x001D,
WM_TIMECHANGE = 0x001E,
WM_CANCELMODE = 0x001F,
WM_SETCURSOR = 0x0020,
WM_MOUSEACTIVATE = 0x0021,
WM_CHILDACTIVATE = 0x0022,
WM_QUEUESYNC = 0x0023,

WM_GETMINMAXINFO = 0x0024,
}

struct RECT
{
LONG left;
LONG top;
LONG right;
LONG bottom;
}
alias RECT* PRECT, NPRECT, LPRECT;

struct PAINTSTRUCT {
HDC hdc;
BOOL fErase;
RECT rcPaint;
BOOL fRestore;
BOOL fIncUpdate;
BYTE rgbReserved[32];
}
alias PAINTSTRUCT* PPAINTSTRUCT, NPPAINTSTRUCT, LPPAINTSTRUCT;

// flags for GetDCEx()

enum
{
DCX_WINDOW = 0x00000001,
DCX_CACHE = 0x00000002,
DCX_NORESETATTRS = 0x00000004,
DCX_CLIPCHILDREN = 0x00000008,
DCX_CLIPSIBLINGS = 0x00000010,
DCX_PARENTCLIP = 0x00000020,
```

```
DCX_EXCLUDERGN = 0x00000040,
DCX_INTERSECTRGN = 0x00000080,
DCX_EXCLUDEUPDATE = 0x00000100,
DCX_INTERSECTUPDATE = 0x00000200,
DCX_LOCKWINDOWUPDATE = 0x00000400,
DCX_VALIDATE = 0x00200000,
}

extern (Windows)
{
BOOL UpdateWindow(HWND hWnd);
HWND SetActiveWindow(HWND hWnd);
HWND GetForegroundWindow();
BOOL PaintDesktop(HDC hdc);
BOOL SetForegroundWindow(HWND hWnd);
HWND WindowFromDC(HDC hDC);
HDC GetDC(HWND hWnd);
HDC GetDCEx(HWND hWnd, HRGN hrgnClip, DWORD flags);
HDC GetWindowDC(HWND hWnd);
int ReleaseDC(HWND hWnd, HDC hDC);
HDC BeginPaint(HWND hWnd, LPPAINTSTRUCT lpPaint);
BOOL EndPaint(HWND hWnd, PAINTSTRUCT *lpPaint);
BOOL GetUpdateRect(HWND hWnd, LPRECT lpRect, BOOL bErase);
int GetUpdateRgn(HWND hWnd, HRGN hRgn, BOOL bErase);
int SetWindowRgn(HWND hWnd, HRGN hRgn, BOOL bRedraw);
int GetWindowRgn(HWND hWnd, HRGN hRgn);
int ExcludeUpdateRgn(HDC hDC, HWND hWnd);
BOOL InvalidateRect(HWND hWnd, RECT *lpRect, BOOL bErase);
BOOL ValidateRect(HWND hWnd, RECT *lpRect);
BOOL InvalidateRgn(HWND hWnd, HRGN hRgn, BOOL bErase);
BOOL ValidateRgn(HWND hWnd, HRGN hRgn);
BOOL RedrawWindow(HWND hWnd, RECT *lprcUpdate, HRGN hrgnUpdate, UINT flags);
}

// flags for RedrawWindow()
enum
{
RDW_INVALIDATE = 0x0001,
RDW_INTERNALPAINT = 0x0002,
RDW_ERASE = 0x0004,
RDW_VALIDATE = 0x0008,
RDW_NOINTERNALPAINT = 0x0010,
RDW_NOERASE = 0x0020,
RDW_NOCHILDREN = 0x0040,
RDW_ALLCHILDREN = 0x0080,
RDW_UPDATENOW = 0x0100,
RDW_ERASENOW = 0x0200,
RDW_FRAME = 0x0400,
RDW_NOFRAME = 0x0800,
}

extern (Windows)
{
BOOL GetClientRect(HWND hWnd, LPRECT lpRect);
BOOL GetWindowRect(HWND hWnd, LPRECT lpRect);
BOOL AdjustWindowRect(LPRECT lpRect, DWORD dwStyle, BOOL bMenu);
BOOL AdjustWindowRectEx(LPRECT lpRect, DWORD dwStyle, BOOL bMenu, DWORD
dwExStyle);
HFONT CreateFontA(int, int, int, int, int, DWORD,
DWORD, DWORD, DWORD, DWORD, DWORD,
DWORD, DWORD, LPCSTR);
HFONT CreateFontW(int, int, int, int, int, DWORD,
DWORD, DWORD, DWORD, DWORD, DWORD,
DWORD, DWORD, LPCWSTR);
```

```
}

enum
{
OUT_DEFAULT_PRECIS = 0,
OUT_STRING_PRECIS = 1,
OUT_CHARACTER_PRECIS = 2,
OUT_STROKE_PRECIS = 3,
OUT_TT_PRECIS = 4,
OUT_DEVICE_PRECIS = 5,
OUT_RASTER_PRECIS = 6,
OUT_TT_ONLY_PRECIS = 7,
OUT_OUTLINE_PRECIS = 8,
OUT_SCREEN_OUTLINE_PRECIS = 9,

CLIP_DEFAULT_PRECIS = 0,
CLIP_CHARACTER_PRECIS = 1,
CLIP_STROKE_PRECIS = 2,
CLIP_MASK = 0xf,
CLIP_LH_ANGLES = (1<<4),
CLIP_TT_ALWAYS = (2<<4),
CLIP_EMBEDDED = (8<<4),

DEFAULT_QUALITY = 0,
DRAFT_QUALITY = 1,
PROOF_QUALITY = 2,

NONANTIALIASED_QUALITY = 3,
ANTIALIASED_QUALITY = 4,


DEFAULT_PITCH = 0,
FIXED_PITCH = 1,
VARIABLE_PITCH = 2,

MONO_FONT = 8,


ANSI_CHARSET = 0,
DEFAULT_CHARSET = 1,
SYMBOL_CHARSET = 2,
SHIFTJIS_CHARSET = 128,
HANGEUL_CHARSET = 129,
GB2312_CHARSET = 134,
CHINESEBIG5_CHARSET = 136,
OEM_CHARSET = 255,

JOHAB_CHARSET = 130,
HEBREW_CHARSET = 177,
ARABIC_CHARSET = 178,
GREEK_CHARSET = 161,
TURKISH_CHARSET = 162,
VIETNAMESE_CHARSET = 163,
THAI_CHARSET = 222,
EASTEUROPE_CHARSET = 238,
RUSSIAN_CHARSET = 204,

MAC_CHARSET = 77,
BALTIC_CHARSET = 186,

FS_LATIN1 = 0x00000001L,
FS_LATIN2 = 0x00000002L,
FS_CYRILLIC = 0x00000004L,
FS_GREEK = 0x00000008L,
```

```
FS_TURKISH = 0x00000010L,
FS_HEBREW = 0x00000020L,
FS_ARABIC = 0x00000040L,
FS_BALTIC = 0x00000080L,
FS_VIETNAMESE = 0x00000100L,
FS_THAI = 0x00010000L,
FS_JISJAPAN = 0x00020000L,
FS_CHINESESIMP = 0x00040000L,
FS_WANSUNG = 0x00080000L,
FS_CHINESETRAD = 0x00100000L,
FS_JOHAB = 0x00200000L,
FS_SYMBOL = cast(int)0x80000000L,


/* Font Families */
FF_DONTCARE = (0<<4), /* Don't care or don't know. */
FF_ROMAN = (1<<4), /* Variable stroke width, serifed. */
/* Times Roman, Century Schoolbook, etc. */
FF_SWISS = (2<<4), /* Variable stroke width, sans-serifed. */
/* Helvetica, Swiss, etc. */
FF_MODERN = (3<<4), /* Constant stroke width, serifed or sans-serifed. */
/* Pica, Elite, Courier, etc. */
FF_SCRIPT = (4<<4), /* Cursive, etc. */
FF_DECORATIVE = (5<<4), /* Old English, etc. */

/* Font Weights */
FW_DONTCARE = 0,
FW_THIN = 100,
FW_EXTRALIGHT = 200,
FW_LIGHT = 300,
FW_NORMAL = 400,
FW_MEDIUM = 500,
FW_SEMIBOLD = 600,
FW_BOLD = 700,
FW_EXTRABOLD = 800,
FW_HEAVY = 900,

FW_ULTRALIGHT = FW_EXTRALIGHT,
FW_REGULAR = FW_NORMAL,
FW_DEMIBOLD = FW_SEMIBOLD,
FW_ULTRABOLD = FW_EXTRABOLD,
FW_BLACK = FW_HEAVY,

PANOSE_COUNT = 10,
PAN_FAMILYTYPE_INDEX = 0,
PAN_SERIFSTYLE_INDEX = 1,
PAN_WEIGHT_INDEX = 2,
PAN_PROPORTION_INDEX = 3,
PAN_CONTRAST_INDEX = 4,
PAN_STROKEVARIATION_INDEX = 5,
PAN_ARMSTYLE_INDEX = 6,
PAN_LETTERFORM_INDEX = 7,
PAN_MIDLINE_INDEX = 8,
PAN_XHEIGHT_INDEX = 9,

PAN_CULTURE_LATIN = 0,
}

struct RGBQUAD {
BYTE rgbBlue;
BYTE rgbGreen;
BYTE rgbRed;
BYTE rgbReserved;
}
```

```
alias RGBQUAD* LPRGBQUAD;

struct BITMAPINFOHEADER
{
DWORD biSize;
LONG biWidth;
LONG biHeight;
WORD biPlanes;
WORD biBitCount;
DWORD biCompression;
DWORD biSizeImage;
LONG biXPelsPerMeter;
LONG biYPelsPerMeter;
DWORD biClrUsed;
DWORD biClrImportant;
}
alias BITMAPINFOHEADER* LPBITMAPINFOHEADER, PBITMAPINFOHEADER;

struct BITMAPINFO {
BITMAPINFOHEADER bmiHeader;
RGBQUAD bmiColors[1];
}
alias BITMAPINFO* LPBITMAPINFO, PBITMAPINFO;

struct PALETTEENTRY {
BYTE peRed;
BYTE peGreen;
BYTE peBlue;
BYTE peFlags;
}
alias PALETTEENTRY* PPALETTEENTRY, LPPALETTEENTRY;

struct LOGPALETTE {
WORD palVersion;
WORD palNumEntries;
PALETTEENTRY palPalEntry[1];
}
alias LOGPALETTE* PLOGPALETTE, NPLOGPALETTE, LPLOGPALETTE;

/* Pixel format descriptor */
struct PIXELFORMATDESCRIPTOR
{
WORD nSize;
WORD nVersion;
DWORD dwFlags;
BYTE iPixelType;
BYTE cColorBits;
BYTE cRedBits;
BYTE cRedShift;
BYTE cGreenBits;
BYTE cGreenShift;
BYTE cBlueBits;
BYTE cBlueShift;
BYTE cAlphaBits;
BYTE cAlphaShift;
BYTE cAccumBits;
BYTE cAccumRedBits;
BYTE cAccumGreenBits;
BYTE cAccumBlueBits;
BYTE cAccumAlphaBits;
BYTE cDepthBits;
BYTE cStencilBits;
BYTE cAuxBuffers;
BYTE iLayerType;
```

```
BYTE bReserved;
DWORD dwLayerMask;
DWORD dwVisibleMask;
DWORD dwDamageMask;
}
alias PIXELFORMATDESCRIPTOR* PPIXELFORMATDESCRIPTOR, LPPIXELFORMATDESCRIPTOR;


extern (Windows)
{
BOOL RoundRect(HDC, int, int, int, int, int, int);
BOOL ResizePalette(HPALETTE, UINT);
int SaveDC(HDC);
int SelectClipRgn(HDC, HRGN);
int ExtSelectClipRgn(HDC, HRGN, int);
int SetMetaRgn(HDC);
HGDIOBJ SelectObject(HDC, HGDIOBJ);
HPALETTE SelectPalette(HDC, HPALETTE, BOOL);
COLORREF SetBkColor(HDC, COLORREF);
int SetBkMode(HDC, int);
LONG SetBitmapBits(HBITMAP, DWORD, void *);
UINT SetBoundsRect(HDC, RECT *, UINT);
int SetDIBits(HDC, HBITMAP, UINT, UINT, void *, BITMAPINFO *, UINT);
int SetDIBitsToDevice(HDC, int, int, DWORD, DWORD, int,
int, UINT, UINT, void *, BITMAPINFO *, UINT);
DWORD SetMapperFlags(HDC, DWORD);
int SetGraphicsMode(HDC hdc, int iMode);
int SetMapMode(HDC, int);
HMETAFILE SetMetaFileBitsEx(UINT, BYTE *);
UINT SetPaletteEntries(HPALETTE, UINT, UINT, PALETTEENTRY *);
COLORREF SetPixel(HDC, int, int, COLORREF);
BOOL SetPixelV(HDC, int, int, COLORREF);
BOOL SetPixelFormat(HDC, int, PIXELFORMATDESCRIPTOR *);
int SetPolyFillMode(HDC, int);
BOOL StretchBlt(HDC, int, int, int, int, HDC, int, int, int, int, DWORD);
BOOL SetRectRgn(HRGN, int, int, int, int);
int StretchDIBits(HDC, int, int, int, int, int, int, int, int,
void *, BITMAPINFO *, UINT, DWORD);
int SetROP2(HDC, int);
int SetStretchBltMode(HDC, int);
UINT SetSystemPaletteUse(HDC, UINT);
int SetTextCharacterExtra(HDC, int);
COLORREF SetTextColor(HDC, COLORREF);
UINT SetTextAlign(HDC, UINT);
BOOL SetTextJustification(HDC, int, int);
BOOL UpdateColors(HDC);
}

/* Text Alignment Options */
enum
{
TA_NOUPDATECP = 0,
TA_UPDATECP = 1,

TA_LEFT = 0,
TA_RIGHT = 2,
TA_CENTER = 6,

TA_TOP = 0,
TA_BOTTOM = 8,
TA_BASELINE = 24,

TA_RTLREADING = 256,
TA_MASK = (TA_BASELINE+TA_CENTER+TA_UPDATECP+TA_RTLREADING),
```

```
}

struct POINT
{
LONG x;
LONG y;
}
alias POINT* PPOINT, NPPOINT, LPPOINT;


extern (Windows)
{
BOOL MoveToEx(HDC, int, int, LPPOINT);
BOOL TextOutA(HDC, int, int, LPCSTR, int);
BOOL TextOutW(HDC, int, int, LPCWSTR, int);
}

extern (Windows) void PostQuitMessage(int nExitCode);
extern (Windows) LRESULT DefWindowProcA(HWND hWnd, UINT Msg, WPARAM wParam,
LPARAM lParam);

alias LRESULT function (HWND, UINT, WPARAM, LPARAM) WNDPROC;

struct WNDCLASSEXA {
UINT cbSize;
/* Win 3.x */
UINT style;
WNDPROC lpfnWndProc;
int cbClsExtra;
int cbWndExtra;
HINSTANCE hInstance;
HICON hIcon;
HCURSOR hCursor;
HBRUSH hbrBackground;
LPCSTR lpszMenuName;
LPCSTR lpszClassName;
/* Win 4.0 */
HICON hIconSm;
}
alias WNDCLASSEXA* PWNDCLASSEXA, NPWNDCLASSEXA, LPWNDCLASSEXA;


struct WNDCLASSA {
UINT style;
WNDPROC lpfnWndProc;
int cbClsExtra;
int cbWndExtra;
HINSTANCE hInstance;
HICON hIcon;
HCURSOR hCursor;
HBRUSH hbrBackground;
LPCSTR lpszMenuName;
LPCSTR lpszClassName;
}
alias WNDCLASSA* PWNDCLASSA, NPWNDCLASSA, LPWNDCLASSA;
alias WNDCLASSA WNDCLASS;


/*
* Window Styles
*/
enum : uint
{
WS_OVERLAPPED = 0x00000000,
```

```
WS_POPUP = 0x80000000,
WS_CHILD = 0x40000000,
WS_MINIMIZE = 0x20000000,
WS_VISIBLE = 0x10000000,
WS_DISABLED = 0x08000000,
WS_CLIPSIBLINGS = 0x04000000,
WS_CLIPCHILDREN = 0x02000000,
WS_MAXIMIZE = 0x01000000,
WS_CAPTION = 0x00C00000, /* WS_BORDER | WS_DLGFRAME */
WS_BORDER = 0x00800000,
WS_DLGFRAME = 0x00400000,
WS_VSCROLL = 0x00200000,
WS_HSCROLL = 0x00100000,
WS_SYSMENU = 0x00080000,
WS_THICKFRAME = 0x00040000,
WS_GROUP = 0x00020000,
WS_TABSTOP = 0x00010000,

WS_MINIMIZEBOX = 0x00020000,
WS_MAXIMIZEBOX = 0x00010000,

WS_TILED = WS_OVERLAPPED,
WS_ICONIC = WS_MINIMIZE,
WS_SIZEBOX = WS_THICKFRAME,

/*
* Common Window Styles
*/
WS_OVERLAPPEDWINDOW = (WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU |
WS_THICKFRAME | WS_MINIMIZEBOX | WS_MAXIMIZEBOX),
WS_TILEDWINDOW = WS_OVERLAPPEDWINDOW,
WS_POPUPWINDOW = (WS_POPUP | WS_BORDER | WS_SYSMENU),
WS_CHILDWINDOW = (WS_CHILD),
}

/*
* Class styles
*/
enum
{
CS_VREDRAW = 0x0001,
CS_HREDRAW = 0x0002,
CS_KEYCVTWINDOW = 0x0004,
CS_DBLCLKS = 0x0008,
CS_OWNDC = 0x0020,
CS_CLASSDC = 0x0040,
CS_PARENTDC = 0x0080,
CS_NOKEYCVT = 0x0100,
CS_NOCLOSE = 0x0200,
CS_SAVEBITS = 0x0800,
CS_BYTEALIGNCLIENT = 0x1000,
CS_BYTEALIGNWINDOW = 0x2000,
CS_GLOBALCLASS = 0x4000,


CS_IME = 0x00010000,
}

extern (Windows)
{
HICON LoadIconA(HINSTANCE hInstance, LPCSTR lpIconName);
HICON LoadIconW(HINSTANCE hInstance, LPCWSTR lpIconName);
HCURSOR LoadCursorA(HINSTANCE hInstance, LPCSTR lpCursorName);
HCURSOR LoadCursorW(HINSTANCE hInstance, LPCWSTR lpCursorName);
```

```
}


const LPSTR IDI_APPLICATION = cast(LPSTR)(32512);

const LPSTR IDC_ARROW = cast(LPSTR)(32512);
const LPSTR IDC_CROSS = cast(LPSTR)(32515);



/*
* Color Types
*/

const CTLCOLOR_MSGBOX = 0;
const CTLCOLOR_EDIT = 1;
const CTLCOLOR_LISTBOX = 2;
const CTLCOLOR_BTN = 3;
const CTLCOLOR_DLG = 4;
const CTLCOLOR_SCROLLBAR = 5;
const CTLCOLOR_STATIC = 6;
const CTLCOLOR_MAX = 7;

const COLOR_SCROLLBAR = 0;
const COLOR_BACKGROUND = 1;
const COLOR_ACTIVECAPTION = 2;
const COLOR_INACTIVECAPTION = 3;
const COLOR_MENU = 4;
const COLOR_WINDOW = 5;
const COLOR_WINDOWFRAME = 6;
const COLOR_MENUTEXT = 7;
const COLOR_WINDOWTEXT = 8;
const COLOR_CAPTIONTEXT = 9;
const COLOR_ACTIVEBORDER = 10;
const COLOR_INACTIVEBORDER = 11;
const COLOR_APPWORKSPACE = 12;
const COLOR_HIGHLIGHT = 13;
const COLOR_HIGHLIGHTTEXT = 14;
const COLOR_BTNFACE = 15;
const COLOR_BTNSHADOW = 16;
const COLOR_GRAYTEXT = 17;
const COLOR_BTNTEXT = 18;
const COLOR_INACTIVECAPTIONTEXT = 19;
const COLOR_BTNHIGHLIGHT = 20;


const COLOR_3DDKSHADOW = 21;
const COLOR_3DLIGHT = 22;
const COLOR_INFOTEXT = 23;
const COLOR_INFOBK = 24;

const COLOR_DESKTOP = COLOR_BACKGROUND;
const COLOR_3DFACE = COLOR_BTNFACE;
const COLOR_3DSHADOW = COLOR_BTNSHADOW;
const COLOR_3DHIGHLIGHT = COLOR_BTNHIGHLIGHT;
const COLOR_3DHILIGHT = COLOR_BTNHIGHLIGHT;
const COLOR_BTNHILIGHT = COLOR_BTNHIGHLIGHT;


enum : int
{
CW_USEDEFAULT = cast(int)0x80000000
}
```

```
/*
* Special value for CreateWindow, et al.
*/
const HWND HWND_DESKTOP = cast(HWND)0;

extern (Windows) ATOM RegisterClassA(WNDCLASSA *lpWndClass);

extern (Windows) HWND CreateWindowExA(
DWORD dwExStyle,
LPCSTR lpClassName,
LPCSTR lpWindowName,
DWORD dwStyle,
int X,
int Y,
int nWidth,
int nHeight,
HWND hWndParent ,
HMENU hMenu,
HINSTANCE hInstance,
LPVOID lpParam);


HWND CreateWindowA(
LPCSTR lpClassName,
LPCSTR lpWindowName,
DWORD dwStyle,
int X,
int Y,
int nWidth,
int nHeight,
HWND hWndParent ,
HMENU hMenu,
HINSTANCE hInstance,
LPVOID lpParam)
{
return CreateWindowExA(0, lpClassName, lpWindowName, dwStyle, X, Y, nWidth,
nHeight, hWndParent, hMenu, hInstance, lpParam);
}

/*
* Message structure
*/
struct MSG {
HWND hwnd;
UINT message;
WPARAM wParam;
LPARAM lParam;
DWORD time;
POINT pt;
}
alias MSG* PMSG, NPMSG, LPMSG;

extern (Windows)
{
BOOL GetMessageA(LPMSG lpMsg, HWND hWnd, UINT wMsgFilterMin, UINT
wMsgFilterMax);
BOOL TranslateMessage(MSG *lpMsg);
LONG DispatchMessageA(MSG *lpMsg);
BOOL PeekMessageA(MSG *lpMsg, HWND hWnd, UINT wMsgFilterMin, UINT
wMsgFilterMax, UINT wRemoveMsg);
HWND GetFocus();
}
```

```
extern (Windows) DWORD ExpandEnvironmentStringsA(LPCSTR lpSrc, LPSTR lpDst,
DWORD nSize);

extern (Windows)
{
BOOL IsValidCodePage(UINT CodePage);
UINT GetACP();
UINT GetOEMCP();
//BOOL GetCPInfo(UINT CodePage, LPCPINFO lpCPInfo);
BOOL IsDBCSLeadByte(BYTE TestChar);
BOOL IsDBCSLeadByteEx(UINT CodePage, BYTE TestChar);
int MultiByteToWideChar(UINT CodePage, DWORD dwFlags, LPCSTR lpMultiByteStr,
int cchMultiByte, LPWSTR lpWideCharStr, int cchWideChar);
int WideCharToMultiByte(UINT CodePage, DWORD dwFlags, LPCWSTR lpWideCharStr,
int cchWideChar, LPSTR lpMultiByteStr, int cchMultiByte, LPCSTR
lpDefaultChar, LPBOOL lpUsedDefaultChar);
}

extern (Windows) HANDLE CreateFileMappingA(HANDLE hFile,
LPSECURITY_ATTRIBUTES lpFileMappingAttributes, DWORD flProtect, DWORD
dwMaximumSizeHigh, DWORD dwMaximumSizeLow, LPCSTR lpName);
extern (Windows) HANDLE CreateFileMappingW(HANDLE hFile,
LPSECURITY_ATTRIBUTES lpFileMappingAttributes, DWORD flProtect, DWORD
dwMaximumSizeHigh, DWORD dwMaximumSizeLow, LPCWSTR lpName);

extern (Windows) BOOL GetMailslotInfo(HANDLE hMailslot, LPDWORD
lpMaxMessageSize, LPDWORD lpNextSize, LPDWORD lpMessageCount, LPDWORD
lpReadTimeout);
extern (Windows) BOOL SetMailslotInfo(HANDLE hMailslot, DWORD lReadTimeout);
extern (Windows) LPVOID MapViewOfFile(HANDLE hFileMappingObject, DWORD
dwDesiredAccess, DWORD dwFileOffsetHigh, DWORD dwFileOffsetLow, DWORD
dwNumberOfBytesToMap);
extern (Windows) LPVOID MapViewOfFileEx(HANDLE hFileMappingObject, DWORD
dwDesiredAccess, DWORD dwFileOffsetHigh, DWORD dwFileOffsetLow, DWORD
dwNumberOfBytesToMap, LPVOID lpBaseAddress);
extern (Windows) BOOL FlushViewOfFile(LPCVOID lpBaseAddress, DWORD
dwNumberOfBytesToFlush);
extern (Windows) BOOL UnmapViewOfFile(LPCVOID lpBaseAddress);

extern (Windows) HGDIOBJ GetStockObject(int);
extern (Windows) BOOL ShowWindow(HWND hWnd, int nCmdShow);

/* Stock Logical Objects */
enum
{ WHITE_BRUSH = 0,
LTGRAY_BRUSH = 1,
GRAY_BRUSH = 2,
DKGRAY_BRUSH = 3,
BLACK_BRUSH = 4,
NULL_BRUSH = 5,
HOLLOW_BRUSH = NULL_BRUSH,
WHITE_PEN = 6,
BLACK_PEN = 7,
NULL_PEN = 8,
OEM_FIXED_FONT = 10,
ANSI_FIXED_FONT = 11,
ANSI_VAR_FONT = 12,
SYSTEM_FONT = 13,
DEVICE_DEFAULT_FONT = 14,
DEFAULT_PALETTE = 15,
SYSTEM_FIXED_FONT = 16,
DEFAULT_GUI_FONT = 17,
STOCK_LAST = 17,
}
```

```
/*
* ShowWindow() Commands
*/
enum
{ SW_HIDE = 0,
SW_SHOWNORMAL = 1,
SW_NORMAL = 1,
SW_SHOWMINIMIZED = 2,
SW_SHOWMAXIMIZED = 3,
SW_MAXIMIZE = 3,
SW_SHOWNOACTIVATE = 4,
SW_SHOW = 5,
SW_MINIMIZE = 6,
SW_SHOWMINNOACTIVE = 7,
SW_SHOWNA = 8,
SW_RESTORE = 9,
SW_SHOWDEFAULT = 10,
SW_MAX = 10,
}

struct TEXTMETRICA
{
LONG tmHeight;
LONG tmAscent;
LONG tmDescent;
LONG tmInternalLeading;
LONG tmExternalLeading;
LONG tmAveCharWidth;
LONG tmMaxCharWidth;
LONG tmWeight;
LONG tmOverhang;
LONG tmDigitizedAspectX;
LONG tmDigitizedAspectY;
BYTE tmFirstChar;
BYTE tmLastChar;
BYTE tmDefaultChar;
BYTE tmBreakChar;
BYTE tmItalic;
BYTE tmUnderlined;
BYTE tmStruckOut;
BYTE tmPitchAndFamily;
BYTE tmCharSet;
}

extern (Windows) BOOL GetTextMetricsA(HDC, TEXTMETRICA*);

/*
* Scroll Bar Constants
*/
enum
{ SB_HORZ = 0,
SB_VERT = 1,
SB_CTL = 2,
SB_BOTH = 3,
}

/*
* Scroll Bar Commands
*/
enum
{ SB_LINEUP = 0,
SB_LINELEFT = 0,
SB_LINEDOWN = 1,
```

```
SB_LINERIGHT = 1,
SB_PAGEUP = 2,
SB_PAGELEFT = 2,
SB_PAGEDOWN = 3,
SB_PAGERIGHT = 3,
SB_THUMBPOSITION = 4,
SB_THUMBTRACK = 5,
SB_TOP = 6,
SB_LEFT = 6,
SB_BOTTOM = 7,
SB_RIGHT = 7,
SB_ENDSCROLL = 8,
}

extern (Windows) int SetScrollPos(HWND hWnd, int nBar, int nPos, BOOL
bRedraw);
extern (Windows) int GetScrollPos(HWND hWnd, int nBar);
extern (Windows) BOOL SetScrollRange(HWND hWnd, int nBar, int nMinPos, int
nMaxPos, BOOL bRedraw);
extern (Windows) BOOL GetScrollRange(HWND hWnd, int nBar, LPINT lpMinPos,
LPINT lpMaxPos);
extern (Windows) BOOL ShowScrollBar(HWND hWnd, int wBar, BOOL bShow);
extern (Windows) BOOL EnableScrollBar(HWND hWnd, UINT wSBflags, UINT
wArrows);

/*
* LockWindowUpdate API
*/

extern (Windows) BOOL LockWindowUpdate(HWND hWndLock);
extern (Windows) BOOL ScrollWindow(HWND hWnd, int XAmount, int YAmount, RECT*
lpRect, RECT* lpClipRect);
extern (Windows) BOOL ScrollDC(HDC hDC, int dx, int dy, RECT* lprcScroll,
RECT* lprcClip, HRGN hrgnUpdate, LPRECT lprcUpdate);
extern (Windows) int ScrollWindowEx(HWND hWnd, int dx, int dy, RECT*
prcScroll, RECT* prcClip, HRGN hrgnUpdate, LPRECT prcUpdate, UINT flags);

/*
* Virtual Keys, Standard Set
*/
enum
{ VK_LBUTTON = 0x01,
VK_RBUTTON = 0x02,
VK_CANCEL = 0x03,
VK_MBUTTON = 0x04, /* NOT contiguous with L & RBUTTON */

VK_BACK = 0x08,
VK_TAB = 0x09,

VK_CLEAR = 0x0C,
VK_RETURN = 0x0D,

VK_SHIFT = 0x10,
VK_CONTROL = 0x11,
VK_MENU = 0x12,
VK_PAUSE = 0x13,
VK_CAPITAL = 0x14,


VK_ESCAPE = 0x1B,

VK_SPACE = 0x20,
VK_PRIOR = 0x21,
VK_NEXT = 0x22,
```

```
VK_END = 0x23,
VK_HOME = 0x24,
VK_LEFT = 0x25,
VK_UP = 0x26,
VK_RIGHT = 0x27,
VK_DOWN = 0x28,
VK_SELECT = 0x29,
VK_PRINT = 0x2A,
VK_EXECUTE = 0x2B,
VK_SNAPSHOT = 0x2C,
VK_INSERT = 0x2D,
VK_DELETE = 0x2E,
VK_HELP = 0x2F,

/* VK_0 thru VK_9 are the same as ASCII '0' thru '9' (0x30 - 0x39) */
/* VK_A thru VK_Z are the same as ASCII 'A' thru 'Z' (0x41 - 0x5A) */

VK_LWIN = 0x5B,
VK_RWIN = 0x5C,
VK_APPS = 0x5D,

VK_NUMPAD0 = 0x60,
VK_NUMPAD1 = 0x61,
VK_NUMPAD2 = 0x62,
VK_NUMPAD3 = 0x63,
VK_NUMPAD4 = 0x64,
VK_NUMPAD5 = 0x65,
VK_NUMPAD6 = 0x66,
VK_NUMPAD7 = 0x67,
VK_NUMPAD8 = 0x68,
VK_NUMPAD9 = 0x69,
VK_MULTIPLY = 0x6A,
VK_ADD = 0x6B,
VK_SEPARATOR = 0x6C,
VK_SUBTRACT = 0x6D,
VK_DECIMAL = 0x6E,
VK_DIVIDE = 0x6F,
VK_F1 = 0x70,
VK_F2 = 0x71,
VK_F3 = 0x72,
VK_F4 = 0x73,
VK_F5 = 0x74,
VK_F6 = 0x75,
VK_F7 = 0x76,
VK_F8 = 0x77,
VK_F9 = 0x78,
VK_F10 = 0x79,
VK_F11 = 0x7A,
VK_F12 = 0x7B,
VK_F13 = 0x7C,
VK_F14 = 0x7D,
VK_F15 = 0x7E,
VK_F16 = 0x7F,
VK_F17 = 0x80,
VK_F18 = 0x81,
VK_F19 = 0x82,
VK_F20 = 0x83,
VK_F21 = 0x84,
VK_F22 = 0x85,
VK_F23 = 0x86,
VK_F24 = 0x87,

VK_NUMLOCK = 0x90,
VK_SCROLL = 0x91,
```

```
/*
* VK_L* & VK_R* - left and right Alt, Ctrl and Shift virtual keys.
* Used only as parameters to GetAsyncKeyState() and GetKeyState().
* No other API or message will distinguish left and right keys in this way.
*/
VK_LSHIFT = 0xA0,
VK_RSHIFT = 0xA1,
VK_LCONTROL = 0xA2,
VK_RCONTROL = 0xA3,
VK_LMENU = 0xA4,
VK_RMENU = 0xA5,


VK_PROCESSKEY = 0xE5,


VK_ATTN = 0xF6,
VK_CRSEL = 0xF7,
VK_EXSEL = 0xF8,
VK_EREOF = 0xF9,
VK_PLAY = 0xFA,
VK_ZOOM = 0xFB,
VK_NONAME = 0xFC,
VK_PA1 = 0xFD,
VK_OEM_CLEAR = 0xFE,
}

extern (Windows) LRESULT SendMessageA(HWND hWnd, UINT Msg, WPARAM wParam,
LPARAM lParam);

alias UINT function (HWND, UINT, WPARAM, LPARAM) LPOFNHOOKPROC;

struct OPENFILENAMEA {
DWORD lStructSize;
HWND hwndOwner;
HINSTANCE hInstance;
LPCSTR lpstrFilter;
LPSTR lpstrCustomFilter;
DWORD nMaxCustFilter;
DWORD nFilterIndex;
LPSTR lpstrFile;
DWORD nMaxFile;
LPSTR lpstrFileTitle;
DWORD nMaxFileTitle;
LPCSTR lpstrInitialDir;
LPCSTR lpstrTitle;
DWORD Flags;
WORD nFileOffset;
WORD nFileExtension;
LPCSTR lpstrDefExt;
LPARAM lCustData;
LPOFNHOOKPROC lpfnHook;
LPCSTR lpTemplateName;
}
alias OPENFILENAMEA *LPOPENFILENAMEA;

struct OPENFILENAMEW {
DWORD lStructSize;
HWND hwndOwner;
HINSTANCE hInstance;
LPCWSTR lpstrFilter;
LPWSTR lpstrCustomFilter;
DWORD nMaxCustFilter;
```

```
DWORD nFilterIndex;
LPWSTR lpstrFile;
DWORD nMaxFile;
LPWSTR lpstrFileTitle;
DWORD nMaxFileTitle;
LPCWSTR lpstrInitialDir;
LPCWSTR lpstrTitle;
DWORD Flags;
WORD nFileOffset;
WORD nFileExtension;
LPCWSTR lpstrDefExt;
LPARAM lCustData;
LPOFNHOOKPROC lpfnHook;
LPCWSTR lpTemplateName;
}
alias OPENFILENAMEW *LPOPENFILENAMEW;

BOOL GetOpenFileNameA(LPOPENFILENAMEA);
BOOL GetOpenFileNameW(LPOPENFILENAMEW);

BOOL GetSaveFileNameA(LPOPENFILENAMEA);
BOOL GetSaveFileNameW(LPOPENFILENAMEW);

short GetFileTitleA(LPCSTR, LPSTR, WORD);
short GetFileTitleW(LPCWSTR, LPWSTR, WORD);

enum
{
PM_NOREMOVE = 0x0000,
PM_REMOVE = 0x0001,
PM_NOYIELD = 0x0002,
}

/* Bitmap Header Definition */
struct BITMAP
{
LONG bmType;
LONG bmWidth;
LONG bmHeight;
LONG bmWidthBytes;
WORD bmPlanes;
WORD bmBitsPixel;
LPVOID bmBits;
}
alias BITMAP* PBITMAP, NPBITMAP, LPBITMAP;


extern (Windows) HDC CreateCompatibleDC(HDC);

extern (Windows) int GetObjectA(HGDIOBJ, int, LPVOID);
extern (Windows) int GetObjectW(HGDIOBJ, int, LPVOID);
extern (Windows) BOOL DeleteDC(HDC);

struct LOGFONTA
{
LONG lfHeight;
LONG lfWidth;
LONG lfEscapement;
LONG lfOrientation;
LONG lfWeight;
BYTE lfItalic;
BYTE lfUnderline;
BYTE lfStrikeOut;
BYTE lfCharSet;
```

```
BYTE lfOutPrecision;
BYTE lfClipPrecision;
BYTE lfQuality;
BYTE lfPitchAndFamily;
CHAR lfFaceName[32 ];
}
alias LOGFONTA* PLOGFONTA, NPLOGFONTA, LPLOGFONTA;

extern (Windows) HMENU LoadMenuA(HINSTANCE hInstance, LPCSTR lpMenuName);
extern (Windows) HMENU LoadMenuW(HINSTANCE hInstance, LPCWSTR lpMenuName);

extern (Windows) HMENU GetSubMenu(HMENU hMenu, int nPos);

extern (Windows) HBITMAP LoadBitmapA(HINSTANCE hInstance, LPCSTR
lpBitmapName);
extern (Windows) HBITMAP LoadBitmapW(HINSTANCE hInstance, LPCWSTR
lpBitmapName);

LPSTR MAKEINTRESOURCEA(int i) { return
cast(LPSTR)(cast(DWORD)(cast(WORD)(i))); }

extern (Windows) HFONT CreateFontIndirectA(LOGFONTA *);

extern (Windows) BOOL MessageBeep(UINT uType);
extern (Windows) int ShowCursor(BOOL bShow);
extern (Windows) BOOL SetCursorPos(int X, int Y);
extern (Windows) HCURSOR SetCursor(HCURSOR hCursor);
extern (Windows) BOOL GetCursorPos(LPPOINT lpPoint);
extern (Windows) BOOL ClipCursor( RECT *lpRect);
extern (Windows) BOOL GetClipCursor(LPRECT lpRect);
extern (Windows) HCURSOR GetCursor();
extern (Windows) BOOL CreateCaret(HWND hWnd, HBITMAP hBitmap , int nWidth,
int nHeight);
extern (Windows) UINT GetCaretBlinkTime();
extern (Windows) BOOL SetCaretBlinkTime(UINT uMSeconds);
extern (Windows) BOOL DestroyCaret();
extern (Windows) BOOL HideCaret(HWND hWnd);
extern (Windows) BOOL ShowCaret(HWND hWnd);
extern (Windows) BOOL SetCaretPos(int X, int Y);
extern (Windows) BOOL GetCaretPos(LPPOINT lpPoint);
extern (Windows) BOOL ClientToScreen(HWND hWnd, LPPOINT lpPoint);
extern (Windows) BOOL ScreenToClient(HWND hWnd, LPPOINT lpPoint);
extern (Windows) int MapWindowPoints(HWND hWndFrom, HWND hWndTo, LPPOINT
lpPoints, UINT cPoints);
extern (Windows) HWND WindowFromPoint(POINT Point);
extern (Windows) HWND ChildWindowFromPoint(HWND hWndParent, POINT Point);


extern (Windows) BOOL TrackPopupMenu(HMENU hMenu, UINT uFlags, int x, int y,
int nReserved, HWND hWnd, RECT *prcRect);

align (2) struct DLGTEMPLATE {
DWORD style;
DWORD dwExtendedStyle;
WORD cdit;
short x;
short y;
short cx;
short cy;
}
alias DLGTEMPLATE *LPDLGTEMPLATEA;
alias DLGTEMPLATE *LPDLGTEMPLATEW;
```

```
alias LPDLGTEMPLATEA LPDLGTEMPLATE;

alias DLGTEMPLATE *LPCDLGTEMPLATEA;
alias DLGTEMPLATE *LPCDLGTEMPLATEW;


alias LPCDLGTEMPLATEA LPCDLGTEMPLATE;


extern (Windows) int DialogBoxParamA(HINSTANCE hInstance, LPCSTR
lpTemplateName,
HWND hWndParent, DLGPROC lpDialogFunc, LPARAM dwInitParam);
extern (Windows) int DialogBoxIndirectParamA(HINSTANCE hInstance,
LPCDLGTEMPLATEA hDialogTemplate, HWND hWndParent, DLGPROC lpDialogFunc,
LPARAM dwInitParam);

enum : DWORD
{
SRCCOPY = cast(DWORD)0x00CC0020, /* dest = source */
SRCPAINT = cast(DWORD)0x00EE0086, /* dest = source OR dest */
SRCAND = cast(DWORD)0x008800C6, /* dest = source AND dest */
SRCINVERT = cast(DWORD)0x00660046, /* dest = source XOR dest */
SRCERASE = cast(DWORD)0x00440328, /* dest = source AND (NOT dest) */
NOTSRCCOPY = cast(DWORD)0x00330008, /* dest = (NOT source) */
NOTSRCERASE = cast(DWORD)0x001100A6, /* dest = (NOT src) AND (NOT dest) */
MERGECOPY = cast(DWORD)0x00C000CA, /* dest = (source AND pattern) */
MERGEPAINT = cast(DWORD)0x00BB0226, /* dest = (NOT source) OR dest */
PATCOPY = cast(DWORD)0x00F00021, /* dest = pattern */
PATPAINT = cast(DWORD)0x00FB0A09, /* dest = DPSnoo */
PATINVERT = cast(DWORD)0x005A0049, /* dest = pattern XOR dest */
DSTINVERT = cast(DWORD)0x00550009, /* dest = (NOT dest) */
BLACKNESS = cast(DWORD)0x00000042, /* dest = BLACK */
WHITENESS = cast(DWORD)0x00FF0062, /* dest = WHITE */
}

enum
{
SND_SYNC = 0x0000, /* play synchronously (default) */
SND_ASYNC = 0x0001, /* play asynchronously */
SND_NODEFAULT = 0x0002, /* silence (!default) if sound not found */
SND_MEMORY = 0x0004, /* pszSound points to a memory file */
SND_LOOP = 0x0008, /* loop the sound until next sndPlaySound */
SND_NOSTOP = 0x0010, /* don't stop any currently playing sound */

SND_NOWAIT = 0x00002000, /* don't wait if the driver is busy */
SND_ALIAS = 0x00010000, /* name is a registry alias */
SND_ALIAS_ID = 0x00110000, /* alias is a predefined ID */
SND_FILENAME = 0x00020000, /* name is file name */
SND_RESOURCE = 0x00040004, /* name is resource name or atom */

SND_PURGE = 0x0040, /* purge non-static events for task */
SND_APPLICATION = 0x0080, /* look for application specific association */


SND_ALIAS_START = 0, /* alias base */
}

extern (Windows) BOOL PlaySoundA(LPCSTR pszSound, HMODULE hmod, DWORD
fdwSound);
extern (Windows) BOOL PlaySoundW(LPCWSTR pszSound, HMODULE hmod, DWORD
fdwSound);

extern (Windows) int GetClipBox(HDC, LPRECT);
extern (Windows) int GetClipRgn(HDC, HRGN);
```

```
extern (Windows) int GetMetaRgn(HDC, HRGN);
extern (Windows) HGDIOBJ GetCurrentObject(HDC, UINT);
extern (Windows) BOOL GetCurrentPositionEx(HDC, LPPOINT);
extern (Windows) int GetDeviceCaps(HDC, int);

struct LOGPEN
{
UINT lopnStyle;
POINT lopnWidth;
COLORREF lopnColor;
}
alias LOGPEN* PLOGPEN, NPLOGPEN, LPLOGPEN;

enum
{
PS_SOLID = 0,
PS_DASH = 1, /* ------- */
PS_DOT = 2, /* ....... */
PS_DASHDOT = 3, /* _._._._ */
PS_DASHDOTDOT = 4, /* _.._.._ */
PS_NULL = 5,
PS_INSIDEFRAME = 6,
PS_USERSTYLE = 7,
PS_ALTERNATE = 8,
PS_STYLE_MASK = 0x0000000F,

PS_ENDCAP_ROUND = 0x00000000,
PS_ENDCAP_SQUARE = 0x00000100,
PS_ENDCAP_FLAT = 0x00000200,
PS_ENDCAP_MASK = 0x00000F00,

PS_JOIN_ROUND = 0x00000000,
PS_JOIN_BEVEL = 0x00001000,
PS_JOIN_MITER = 0x00002000,
PS_JOIN_MASK = 0x0000F000,

PS_COSMETIC = 0x00000000,
PS_GEOMETRIC = 0x00010000,
PS_TYPE_MASK = 0x000F0000,
}

extern (Windows) HPALETTE CreatePalette(LOGPALETTE *);
extern (Windows) HPEN CreatePen(int, int, COLORREF);
extern (Windows) HPEN CreatePenIndirect(LOGPEN *);
extern (Windows) HRGN CreatePolyPolygonRgn(POINT *, INT *, int, int);
extern (Windows) HBRUSH CreatePatternBrush(HBITMAP);
extern (Windows) HRGN CreateRectRgn(int, int, int, int);
extern (Windows) HRGN CreateRectRgnIndirect(RECT *);
extern (Windows) HRGN CreateRoundRectRgn(int, int, int, int, int, int);
extern (Windows) BOOL CreateScalableFontResourceA(DWORD, LPCSTR, LPCSTR,
LPCSTR);
extern (Windows) BOOL CreateScalableFontResourceW(DWORD, LPCWSTR, LPCWSTR,
LPCWSTR);

COLORREF RGB(int r, int g, int b)
{
return cast(COLORREF)
((cast(BYTE)r|(cast(WORD)(cast(BYTE)g)<<8))|((cast(DWORD)cast(BYTE)b)<<16));
}

extern (Windows) BOOL LineTo(HDC, int, int);
extern (Windows) BOOL DeleteObject(HGDIOBJ);
extern (Windows) int FillRect(HDC hDC, RECT *lprc, HBRUSH hbr);
```

```
extern (Windows) BOOL EndDialog(HWND hDlg, int nResult);
extern (Windows) HWND GetDlgItem(HWND hDlg, int nIDDlgItem);

extern (Windows) BOOL SetDlgItemInt(HWND hDlg, int nIDDlgItem, UINT uValue,
BOOL bSigned);
extern (Windows) UINT GetDlgItemInt(HWND hDlg, int nIDDlgItem, BOOL
*lpTranslated,
BOOL bSigned);

extern (Windows) BOOL SetDlgItemTextA(HWND hDlg, int nIDDlgItem, LPCSTR
lpString);
extern (Windows) BOOL SetDlgItemTextW(HWND hDlg, int nIDDlgItem, LPCWSTR
lpString);

extern (Windows) UINT GetDlgItemTextA(HWND hDlg, int nIDDlgItem, LPSTR
lpString, int nMaxCount);
extern (Windows) UINT GetDlgItemTextW(HWND hDlg, int nIDDlgItem, LPWSTR
lpString, int nMaxCount);

extern (Windows) BOOL CheckDlgButton(HWND hDlg, int nIDButton, UINT uCheck);
extern (Windows) BOOL CheckRadioButton(HWND hDlg, int nIDFirstButton, int
nIDLastButton,
int nIDCheckButton);

extern (Windows) UINT IsDlgButtonChecked(HWND hDlg, int nIDButton);

extern (Windows) HWND SetFocus(HWND hWnd);

extern (Windows) int wsprintfA(LPSTR, LPCSTR, ...);
extern (Windows) int wsprintfW(LPWSTR, LPCWSTR, ...);

enum : uint
{
INFINITE = uint.max,
WAIT_OBJECT_0 = 0,
WAIT_ABANDONED_0 = 0x80,
WAIT_TIMEOUT = 0x102,
WAIT_IO_COMPLETION = 0xc0,
WAIT_ABANDONED = 0x80,
WAIT_FAILED = uint.max,
}

extern (Windows) HANDLE CreateSemaphoreA(LPSECURITY_ATTRIBUTES
lpSemaphoreAttributes, LONG lInitialCount, LONG lMaximumCount, LPCTSTR
lpName);
extern (Windows) HANDLE OpenSemaphoreA(DWORD dwDesiredAccess, BOOL
bInheritHandle, LPCTSTR lpName);
extern (Windows) BOOL ReleaseSemaphore(HANDLE hSemaphore, LONG lReleaseCount,
LPLONG lpPreviousCount);

struct COORD {
SHORT X;
SHORT Y;
}
alias COORD *PCOORD;

struct SMALL_RECT {
SHORT Left;
SHORT Top;
SHORT Right;
SHORT Bottom;
}
alias SMALL_RECT *PSMALL_RECT;
```

```
struct KEY_EVENT_RECORD {
BOOL bKeyDown;
WORD wRepeatCount;
WORD wVirtualKeyCode;
WORD wVirtualScanCode;
union {
WCHAR UnicodeChar;
CHAR AsciiChar;
}
DWORD dwControlKeyState;
}
alias KEY_EVENT_RECORD *PKEY_EVENT_RECORD;

//
// ControlKeyState flags
//

enum
{
RIGHT_ALT_PRESSED = 0x0001, // the right alt key is pressed.
LEFT_ALT_PRESSED = 0x0002, // the left alt key is pressed.
RIGHT_CTRL_PRESSED = 0x0004, // the right ctrl key is pressed.
LEFT_CTRL_PRESSED = 0x0008, // the left ctrl key is pressed.
SHIFT_PRESSED = 0x0010, // the shift key is pressed.
NUMLOCK_ON = 0x0020, // the numlock light is on.
SCROLLLOCK_ON = 0x0040, // the scrolllock light is on.
CAPSLOCK_ON = 0x0080, // the capslock light is on.
ENHANCED_KEY = 0x0100, // the key is enhanced.
}

struct MOUSE_EVENT_RECORD {
COORD dwMousePosition;
DWORD dwButtonState;
DWORD dwControlKeyState;
DWORD dwEventFlags;
}
alias MOUSE_EVENT_RECORD *PMOUSE_EVENT_RECORD;

//
// ButtonState flags
//
enum
{
FROM_LEFT_1ST_BUTTON_PRESSED = 0x0001,
RIGHTMOST_BUTTON_PRESSED = 0x0002,
FROM_LEFT_2ND_BUTTON_PRESSED = 0x0004,
FROM_LEFT_3RD_BUTTON_PRESSED = 0x0008,
FROM_LEFT_4TH_BUTTON_PRESSED = 0x0010,
}

//
// EventFlags
//

enum
{
MOUSE_MOVED = 0x0001,
DOUBLE_CLICK = 0x0002,
}

struct WINDOW_BUFFER_SIZE_RECORD {
COORD dwSize;
}
```

```
alias WINDOW_BUFFER_SIZE_RECORD *PWINDOW_BUFFER_SIZE_RECORD;

struct MENU_EVENT_RECORD {
UINT dwCommandId;
}
alias MENU_EVENT_RECORD *PMENU_EVENT_RECORD;

struct FOCUS_EVENT_RECORD {
BOOL bSetFocus;
}
alias FOCUS_EVENT_RECORD *PFOCUS_EVENT_RECORD;

struct INPUT_RECORD {
WORD EventType;
union {
KEY_EVENT_RECORD KeyEvent;
MOUSE_EVENT_RECORD MouseEvent;
WINDOW_BUFFER_SIZE_RECORD WindowBufferSizeEvent;
MENU_EVENT_RECORD MenuEvent;
FOCUS_EVENT_RECORD FocusEvent;
}
}
alias INPUT_RECORD *PINPUT_RECORD;

//
// EventType flags:
//

enum
{
KEY_EVENT = 0x0001, // Event contains key event record
MOUSE_EVENT = 0x0002, // Event contains mouse event record
WINDOW_BUFFER_SIZE_EVENT = 0x0004, // Event contains window change event
record
MENU_EVENT = 0x0008, // Event contains menu event record
FOCUS_EVENT = 0x0010, // event contains focus change
}

struct CHAR_INFO {
union {
WCHAR UnicodeChar;
CHAR AsciiChar;
}
WORD Attributes;
}
alias CHAR_INFO *PCHAR_INFO;

//
// Attributes flags:
//

enum
{
FOREGROUND_BLUE = 0x0001, // text color contains blue.
FOREGROUND_GREEN = 0x0002, // text color contains green.
FOREGROUND_RED = 0x0004, // text color contains red.
FOREGROUND_INTENSITY = 0x0008, // text color is intensified.
BACKGROUND_BLUE = 0x0010, // background color contains blue.
BACKGROUND_GREEN = 0x0020, // background color contains green.
BACKGROUND_RED = 0x0040, // background color contains red.
BACKGROUND_INTENSITY = 0x0080, // background color is intensified.
}

struct CONSOLE_SCREEN_BUFFER_INFO {
```

```
COORD dwSize;
COORD dwCursorPosition;
WORD wAttributes;
SMALL_RECT srWindow;
COORD dwMaximumWindowSize;
}
alias CONSOLE_SCREEN_BUFFER_INFO *PCONSOLE_SCREEN_BUFFER_INFO;

struct CONSOLE_CURSOR_INFO {
DWORD dwSize;
BOOL bVisible;
}
alias CONSOLE_CURSOR_INFO *PCONSOLE_CURSOR_INFO;

enum
{
ENABLE_PROCESSED_INPUT = 0x0001,
ENABLE_LINE_INPUT = 0x0002,
ENABLE_ECHO_INPUT = 0x0004,
ENABLE_WINDOW_INPUT = 0x0008,
ENABLE_MOUSE_INPUT = 0x0010,
}

enum
{
ENABLE_PROCESSED_OUTPUT = 0x0001,
ENABLE_WRAP_AT_EOL_OUTPUT = 0x0002,
}

BOOL PeekConsoleInputA(HANDLE hConsoleInput, PINPUT_RECORD lpBuffer, DWORD
nLength, LPDWORD lpNumberOfEventsRead);
BOOL PeekConsoleInputW(HANDLE hConsoleInput, PINPUT_RECORD lpBuffer, DWORD
nLength, LPDWORD lpNumberOfEventsRead);
BOOL ReadConsoleInputA(HANDLE hConsoleInput, PINPUT_RECORD lpBuffer, DWORD
nLength, LPDWORD lpNumberOfEventsRead);
BOOL ReadConsoleInputW(HANDLE hConsoleInput, PINPUT_RECORD lpBuffer, DWORD
nLength, LPDWORD lpNumberOfEventsRead);
BOOL WriteConsoleInputA(HANDLE hConsoleInput, in INPUT_RECORD *lpBuffer,
DWORD nLength, LPDWORD lpNumberOfEventsWritten);
BOOL WriteConsoleInputW(HANDLE hConsoleInput, in INPUT_RECORD *lpBuffer,
DWORD nLength, LPDWORD lpNumberOfEventsWritten);
BOOL ReadConsoleOutputA(HANDLE hConsoleOutput, PCHAR_INFO lpBuffer, COORD
dwBufferSize, COORD dwBufferCoord, PSMALL_RECT lpReadRegion);
BOOL ReadConsoleOutputW(HANDLE hConsoleOutput, PCHAR_INFO lpBuffer, COORD
dwBufferSize, COORD dwBufferCoord, PSMALL_RECT lpReadRegion);
BOOL WriteConsoleOutputA(HANDLE hConsoleOutput, in CHAR_INFO *lpBuffer, COORD
dwBufferSize, COORD dwBufferCoord, PSMALL_RECT lpWriteRegion);
BOOL WriteConsoleOutputW(HANDLE hConsoleOutput, in CHAR_INFO *lpBuffer, COORD
dwBufferSize, COORD dwBufferCoord, PSMALL_RECT lpWriteRegion);
BOOL ReadConsoleOutputCharacterA(HANDLE hConsoleOutput, LPSTR lpCharacter,
DWORD nLength, COORD dwReadCoord, LPDWORD lpNumberOfCharsRead);
BOOL ReadConsoleOutputCharacterW(HANDLE hConsoleOutput, LPWSTR lpCharacter,
DWORD nLength, COORD dwReadCoord, LPDWORD lpNumberOfCharsRead);
BOOL ReadConsoleOutputAttribute(HANDLE hConsoleOutput, LPWORD lpAttribute,
DWORD nLength, COORD dwReadCoord, LPDWORD lpNumberOfAttrsRead);
BOOL WriteConsoleOutputCharacterA(HANDLE hConsoleOutput, LPCSTR lpCharacter,
DWORD nLength, COORD dwWriteCoord, LPDWORD lpNumberOfCharsWritten);
BOOL WriteConsoleOutputCharacterW(HANDLE hConsoleOutput, LPCWSTR lpCharacter,
DWORD nLength, COORD dwWriteCoord, LPDWORD lpNumberOfCharsWritten);
BOOL WriteConsoleOutputAttribute(HANDLE hConsoleOutput, in WORD *lpAttribute,
DWORD nLength, COORD dwWriteCoord, LPDWORD lpNumberOfAttrsWritten);
BOOL FillConsoleOutputCharacterA(HANDLE hConsoleOutput, CHAR cCharacter,
DWORD nLength, COORD dwWriteCoord, LPDWORD lpNumberOfCharsWritten);
```

```
BOOL FillConsoleOutputCharacterW(HANDLE hConsoleOutput, WCHAR cCharacter,
DWORD nLength, COORD dwWriteCoord, LPDWORD lpNumberOfCharsWritten);
BOOL FillConsoleOutputAttribute(HANDLE hConsoleOutput, WORD wAttribute, DWORD
nLength, COORD dwWriteCoord, LPDWORD lpNumberOfAttrsWritten);
BOOL GetConsoleMode(HANDLE hConsoleHandle, LPDWORD lpMode);
BOOL GetNumberOfConsoleInputEvents(HANDLE hConsoleInput, LPDWORD
lpNumberOfEvents);
BOOL GetConsoleScreenBufferInfo(HANDLE hConsoleOutput,
PCONSOLE_SCREEN_BUFFER_INFO lpConsoleScreenBufferInfo);
COORD GetLargestConsoleWindowSize( HANDLE hConsoleOutput);
BOOL GetConsoleCursorInfo(HANDLE hConsoleOutput, PCONSOLE_CURSOR_INFO
lpConsoleCursorInfo);
BOOL GetNumberOfConsoleMouseButtons( LPDWORD lpNumberOfMouseButtons);
BOOL SetConsoleMode(HANDLE hConsoleHandle, DWORD dwMode);
BOOL SetConsoleActiveScreenBuffer(HANDLE hConsoleOutput);
BOOL FlushConsoleInputBuffer(HANDLE hConsoleInput);
BOOL SetConsoleScreenBufferSize(HANDLE hConsoleOutput, COORD dwSize);
BOOL SetConsoleCursorPosition(HANDLE hConsoleOutput, COORD dwCursorPosition);
BOOL SetConsoleCursorInfo(HANDLE hConsoleOutput, in CONSOLE_CURSOR_INFO
*lpConsoleCursorInfo);
BOOL ScrollConsoleScreenBufferA(HANDLE hConsoleOutput, in SMALL_RECT
*lpScrollRectangle, in SMALL_RECT *lpClipRectangle, COORD
dwDestinationOrigin, in CHAR_INFO *lpFill);
BOOL ScrollConsoleScreenBufferW(HANDLE hConsoleOutput, in SMALL_RECT
*lpScrollRectangle, in SMALL_RECT *lpClipRectangle, COORD
dwDestinationOrigin, in CHAR_INFO *lpFill);
BOOL SetConsoleWindowInfo(HANDLE hConsoleOutput, BOOL bAbsolute, in
SMALL_RECT *lpConsoleWindow);
BOOL SetConsoleTextAttribute(HANDLE hConsoleOutput, WORD wAttributes);
alias BOOL function(DWORD CtrlType) PHANDLER_ROUTINE;
BOOL SetConsoleCtrlHandler(PHANDLER_ROUTINE HandlerRoutine, BOOL Add);
BOOL GenerateConsoleCtrlEvent( DWORD dwCtrlEvent, DWORD dwProcessGroupId);
BOOL AllocConsole();
BOOL FreeConsole();
DWORD GetConsoleTitleA(LPSTR lpConsoleTitle, DWORD nSize);
DWORD GetConsoleTitleW(LPWSTR lpConsoleTitle, DWORD nSize);
BOOL SetConsoleTitleA(LPCSTR lpConsoleTitle);
BOOL SetConsoleTitleW(LPCWSTR lpConsoleTitle);
BOOL ReadConsoleA(HANDLE hConsoleInput, LPVOID lpBuffer, DWORD
nNumberOfCharsToRead, LPDWORD lpNumberOfCharsRead, LPVOID lpReserved);
BOOL ReadConsoleW(HANDLE hConsoleInput, LPVOID lpBuffer, DWORD
nNumberOfCharsToRead, LPDWORD lpNumberOfCharsRead, LPVOID lpReserved);
BOOL WriteConsoleA(HANDLE hConsoleOutput, in void *lpBuffer, DWORD
nNumberOfCharsToWrite, LPDWORD lpNumberOfCharsWritten, LPVOID lpReserved);
BOOL WriteConsoleW(HANDLE hConsoleOutput, in void *lpBuffer, DWORD
nNumberOfCharsToWrite, LPDWORD lpNumberOfCharsWritten, LPVOID lpReserved);
HANDLE CreateConsoleScreenBuffer(DWORD dwDesiredAccess, DWORD dwShareMode, in
SECURITY_ATTRIBUTES *lpSecurityAttributes, DWORD dwFlags, LPVOID
lpScreenBufferData);
UINT GetConsoleCP();
BOOL SetConsoleCP( UINT wCodePageID);
UINT GetConsoleOutputCP();
BOOL SetConsoleOutputCP(UINT wCodePageID);

enum
{
CONSOLE_TEXTMODE_BUFFER = 1,
}

enum
{
SM_CXSCREEN = 0,
SM_CYSCREEN = 1,
SM_CXVSCROLL = 2,
```

```
SM_CYHSCROLL = 3,
SM_CYCAPTION = 4,
SM_CXBORDER = 5,
SM_CYBORDER = 6,
SM_CXDLGFRAME = 7,
SM_CYDLGFRAME = 8,
SM_CYVTHUMB = 9,
SM_CXHTHUMB = 10,
SM_CXICON = 11,
SM_CYICON = 12,
SM_CXCURSOR = 13,
SM_CYCURSOR = 14,
SM_CYMENU = 15,
SM_CXFULLSCREEN = 16,
SM_CYFULLSCREEN = 17,
SM_CYKANJIWINDOW = 18,
SM_MOUSEPRESENT = 19,
SM_CYVSCROLL = 20,
SM_CXHSCROLL = 21,
SM_DEBUG = 22,
SM_SWAPBUTTON = 23,
SM_RESERVED1 = 24,
SM_RESERVED2 = 25,
SM_RESERVED3 = 26,
SM_RESERVED4 = 27,
SM_CXMIN = 28,
SM_CYMIN = 29,
SM_CXSIZE = 30,
SM_CYSIZE = 31,
SM_CXFRAME = 32,
SM_CYFRAME = 33,
SM_CXMINTRACK = 34,
SM_CYMINTRACK = 35,
SM_CXDOUBLECLK = 36,
SM_CYDOUBLECLK = 37,
SM_CXICONSPACING = 38,
SM_CYICONSPACING = 39,
SM_MENUDROPALIGNMENT = 40,
SM_PENWINDOWS = 41,
SM_DBCSENABLED = 42,
SM_CMOUSEBUTTONS = 43,


SM_CXFIXEDFRAME = SM_CXDLGFRAME,
SM_CYFIXEDFRAME = SM_CYDLGFRAME,
SM_CXSIZEFRAME = SM_CXFRAME,
SM_CYSIZEFRAME = SM_CYFRAME,

SM_SECURE = 44,
SM_CXEDGE = 45,
SM_CYEDGE = 46,
SM_CXMINSPACING = 47,
SM_CYMINSPACING = 48,
SM_CXSMICON = 49,
SM_CYSMICON = 50,
SM_CYSMCAPTION = 51,
SM_CXSMSIZE = 52,
SM_CYSMSIZE = 53,
SM_CXMENUSIZE = 54,
SM_CYMENUSIZE = 55,
SM_ARRANGE = 56,
SM_CXMINIMIZED = 57,
SM_CYMINIMIZED = 58,
SM_CXMAXTRACK = 59,
```

```
SM_CYMAXTRACK = 60,
SM_CXMAXIMIZED = 61,
SM_CYMAXIMIZED = 62,
SM_NETWORK = 63,
SM_CLEANBOOT = 67,
SM_CXDRAG = 68,
SM_CYDRAG = 69,
SM_SHOWSOUNDS = 70,
SM_CXMENUCHECK = 71,
SM_CYMENUCHECK = 72,
SM_SLOWMACHINE = 73,
SM_MIDEASTENABLED = 74,
SM_CMETRICS = 75,
}

int GetSystemMetrics(int nIndex);

enum : DWORD
{
STILL_ACTIVE = (0x103),
}

DWORD TlsAlloc();
LPVOID TlsGetValue(DWORD);
BOOL TlsSetValue(DWORD, LPVOID);
BOOL TlsFree(DWORD);
UINT SetTimer(HWND hWnd, UINT nIDEvent, UINT uElapse, TIMERPROC
lpТаймерFunc);
BOOL KillTimer(HWND hWnd, UINT uIDEvent);



alias UINT SOCKET;
alias int socklen_t;

const SOCKET INVALID_SOCKET = cast(SOCKET)~0;
const int SOCKET_ERROR = -1;

const int WSADESCRIPTION_LEN = 256;
const int WSASYS_STATUS_LEN = 128;

struct WSADATA
{
     WORD wVersion;
     WORD wHighVersion;
     char szDescription[WSADESCRIPTION_LEN + 1];
     char szSystemStatus[WSASYS_STATUS_LEN + 1];
     USHORT iMaxSockets;
     USHORT iMaxUdpDg;
     char* lpVendorInfo;
}
alias WSADATA* LPWSADATA;


const int IOCPARM_MASK = 0x7F;
const int IOC_IN = cast(int)0x80000000;
const int FIONBIO = cast(int)(IOC_IN | ((UINT.sizeof & IOCPARM_MASK) << 16) |
(102 << 8) | 126);


     int WSAStartup(WORD wVersionRequested, LPWSADATA lpWSAData);
     int WSACleanup();
     SOCKET socket(int af, int type, int protocol);
```

```
    int ioctlsocket(SOCKET s, int cmd, uint* argp);
    int bind(SOCKET s, sockaddr* name, int namelen);
    int connect(SOCKET s, sockaddr* name, int namelen);
    int listen(SOCKET s, int backlog);
    SOCKET accept(SOCKET s, sockaddr* addr, int* addrlen);
    int closesocket(SOCKET s);
    int shutdown(SOCKET s, int how);
    int getpeername(SOCKET s, sockaddr* name, int* namelen);
    int getsockname(SOCKET s, sockaddr* name, int* namelen);
    int send(SOCKET s, void* buf, int len, int flags);
    int sendto(SOCKET s, void* buf, int len, int flags, sockaddr* to, int
    tolen);
    int recv(SOCKET s, void* buf, int len, int flags);
    int recvfrom(SOCKET s, void* buf, int len, int flags, sockaddr* from,
    int* fromlen);
    int getsockopt(SOCKET s, int level, int optname, void* optval, int*
    optlen);
    int setsockopt(SOCKET s, int level, int optname, void* optval, int
    optlen);
    uint inet_addr(char* cp);
    int select(int nfds, fd_set* readfds, fd_set* writefds, fd_set* errorfds,
    timeval* timeout);
    char* inet_ntoa(in_addr ina);
    hostent* gethostbyname(char* name);
    hostent* gethostbyaddr(void* addr, int len, int type);
    protoent* getprotobyname(char* name);
    protoent* getprotobynumber(int number);
    servent* getservbyname(char* name, char* proto);
    servent* getservbyport(int port, char* proto);
    int gethostname(char* name, int namelen);
    int getaddrinfo(char* nodename, char* servname, addrinfo* hints,
    addrinfo** res);
    void freeaddrinfo(addrinfo* ai);
    int getnameinfo(sockaddr* sa, socklen_t salen, char* host, DWORD hostlen,
    char* serv, DWORD servlen, int flags);

enum: int
{
    WSAEWOULDBLOCK = 10035,
    WSAEINTR = 10004,
    WSAHOST_NOT_FOUND = 11001,
}

int WSAGetLastError();


enum: int
{
    AF_UNSPEC = 0,
    AF_UNIX = 1,
    AF_INET = 2,
    AF_IMPLINK = 3,
    AF_PUP = 4,
    AF_CHAOS = 5,
    AF_NS = 6,
    AF_IPX = AF_NS,
    AF_ISO = 7,
    AF_OSI = AF_ISO,
    AF_ECMA = 8,
    AF_DATAKIT = 9,
    AF_CCITT = 10,
    AF_SNA = 11,
    AF_DECnet = 12,
    AF_DLI = 13,
```

```
    AF_LAT = 14,
    AF_HYLINK = 15,
    AF_APPLETALK = 16,
    AF_NETBIOS = 17,
    AF_VOICEVIEW = 18,
    AF_FIREFOX = 19,
    AF_UNKNOWN1 = 20,
    AF_BAN = 21,
    AF_ATM = 22,
    AF_INET6 = 23,
    AF_CLUSTER = 24,
    AF_12844 = 25,
    AF_IRDA = 26,
    AF_NETDES = 28,
    AF_MAX = 29,
    PF_UNSPEC = AF_UNSPEC,
    PF_UNIX = AF_UNIX,
    PF_INET = AF_INET,
    PF_IMPLINK = AF_IMPLINK,
    PF_PUP = AF_PUP,
    PF_CHAOS = AF_CHAOS,
    PF_NS = AF_NS,
    PF_IPX = AF_IPX,
    PF_ISO = AF_ISO,
    PF_OSI = AF_OSI,
    PF_ECMA = AF_ECMA,
    PF_DATAKIT = AF_DATAKIT,
    PF_CCITT = AF_CCITT,
    PF_SNA = AF_SNA,
    PF_DECnet = AF_DECnet,
    PF_DLI = AF_DLI,
    PF_LAT = AF_LAT,
    PF_HYLINK = AF_HYLINK,
    PF_APPLETALK = AF_APPLETALK,
    PF_VOICEVIEW = AF_VOICEVIEW,
    PF_FIREFOX = AF_FIREFOX,
    PF_UNKNOWN1 = AF_UNKNOWN1,
    PF_BAN = AF_BAN,
    PF_INET6 = AF_INET6,
    PF_MAX = AF_MAX,
}


enum: int
{
    SOL_SOCKET = 0xFFFF,
}


enum: int
{
    SO_DEBUG = 0x0001,
    SO_ACCEPTCONN = 0x0002,
    SO_REUSEADDR = 0x0004,
    SO_KEEPALIVE = 0x0008,
    SO_DONTROUTE = 0x0010,
    SO_BROADCAST = 0x0020,
    SO_USELOOPBACK = 0x0040,
    SO_LINGER = 0x0080,
    SO_DONTLINGER = ~SO_LINGER,
    SO_OOBINLINE = 0x0100,
    SO_SNDBUF = 0x1001,
    SO_RCVBUF = 0x1002,
    SO_SNDLOWAT = 0x1003,
```

```
    SO_RCVLOWAT = 0x1004,
    SO_SNDTIMEO = 0x1005,
    SO_RCVTIMEO = 0x1006,
    SO_ERROR = 0x1007,
    SO_TYPE = 0x1008,
    SO_EXCLUSIVEADDRUSE = ~SO_REUSEADDR,
    TCP_NODELAY = 1,
    IP_MULTICAST_LOOP = 0x4,
    IP_ADD_MEMBERSHIP = 0x5,
    IP_DROP_MEMBERSHIP = 0x6,
    IPV6_UNICAST_HOPS = 4,
    IPV6_MULTICAST_IF = 9,
    IPV6_MULTICAST_HOPS = 10,
    IPV6_MULTICAST_LOOP = 11,
    IPV6_ADD_MEMBERSHIP = 12,
    IPV6_DROP_MEMBERSHIP = 13,
    IPV6_JOIN_GROUP = IPV6_ADD_MEMBERSHIP,
    IPV6_LEAVE_GROUP = IPV6_DROP_MEMBERSHIP,
}


const uint FD_SETSIZE = 64;


struct fd_set
{
    UINT fd_count;
    SOCKET[FD_SETSIZE] fd_array;
}


// Removes.
void FD_CLR(SOCKET fd, fd_set* set)
{
    uint c = set.fd_count;
    SOCKET* start = set.fd_array.ptr;
    SOCKET* stop = start + c;
    for(; start != stop; start++)
    {
        if(*start == fd)
            goto found;
    }
    return; //not found
    found:
    for(++start; start != stop; start++)
    {
        *(start - 1) = *start;
    }
    set.fd_count = c - 1;
}


// Tests.
int FD_ISSET(SOCKET fd, fd_set* set)
{
    SOCKET* start = set.fd_array.ptr;
    SOCKET* stop = start + set.fd_count;
    for(; start != stop; start++)
    {
        if(*start == fd)
            return true;
    }
    return false;
}
```

```
// Adds.
void FD_SET(SOCKET fd, fd_set* set)
{
     uint c = set.fd_count;
     set.fd_array.ptr[c] = fd;
     set.fd_count = c + 1;
}


// Resets to zero.
void FD_ZERO(fd_set* set)
{
     set.fd_count = 0;
}


struct linger
{
     USHORT l_onoff;
     USHORT l_linger;
}


struct protoent
{
     char* p_name;
     char** p_aliases;
     SHORT p_proto;
}


struct servent
{
     char* s_name;
     char** s_aliases;
     SHORT s_port;
     char* s_proto;
}


/+
union in6_addr
{
     private union _u_t
     {
          BYTE[16] Byte;
          WORD[8] Word;
     }
     _u_t u;
}


struct in_addr6
{
     BYTE[16] s6_addr;
}
+/


version(BigEndian)
{
     uint16_t htons(uint16_t x)
```

```
    {
        return x;
    }
    uint32_t htonl(uint32_t x)
    {
        return x;
    }
}
else version(LittleEndian)
{
    private import stdrus: развербит;
    uint16_t htons(uint16_t x)
    {
        return cast(uint16_t)((x >> 8) | (x << 8));
    }


    uint32_t htonl(uint32_t x)
    {
        return развербит(x);
    }
}
else
{
    static assert(0);
}


uint16_t ntohs(uint16_t x)
{
    return htons(x);
}


uint32_t ntohl(uint32_t x)
{
    return htonl(x);
}


enum: int
{
    SOCK_STREAM = 1,
    SOCK_DGRAM = 2,
    SOCK_RAW = 3,
    SOCK_RDM = 4,
    SOCK_SEQPACKET = 5,
}


enum: int
{
    IPPROTO_IP = 0,
    IPPROTO_ICMP = 1,
    IPPROTO_IGMP = 2,
    IPPROTO_GGP = 3,
    IPPROTO_TCP = 6,
    IPPROTO_PUP = 12,
    IPPROTO_UDP = 17,
    IPPROTO_IDP = 22,
    IPPROTO_IPV6 = 41,
    IPPROTO_ND = 77,
    IPPROTO_RAW = 255,
    IPPROTO_MAX = 256,
```

```
}


enum: int
{
     MSG_OOB = 0x1,
     MSG_PEEK = 0x2,
     MSG_DONTROUTE = 0x4,
MSG_NOSIGNAL = 0x0, /// not supported on win32, would be 0x4000 if it was
}


enum: int
{
     SD_RECEIVE = 0,
     SD_SEND = 1,
     SD_BOTH = 2,
}


enum: uint
{
     INADDR_ANY = 0,
     INADDR_LOOPBACK = 0x7F000001,
     INADDR_BROADCAST = 0xFFFFFFFF,
     INADDR_NONE = 0xFFFFFFFF,
     ADDR_ANY = INADDR_ANY,
}


enum: int
{
     AI_PASSIVE = 0x1,
     AI_CANONNAME = 0x2,
     AI_NUMERICHOST = 0x4,
}


struct timeval
{
     int32_t tv_sec;
     int32_t tv_usec;
}


union in_addr
{
     private union _S_un_t
     {
          private struct _S_un_b_t
          {
               uint8_t s_b1, s_b2, s_b3, s_b4;
          }
          _S_un_b_t S_un_b;
          private struct _S_un_w_t
          {
               uint16_t s_w1, s_w2;
          }
          _S_un_w_t S_un_w;
          uint32_t S_addr;
     }
     _S_un_t S_un;
     uint32_t s_addr;
     struct
```

```
    {
        uint8_t s_net, s_host;
        union
        {
            uint16_t s_imp;
            struct
            {
                uint8_t s_lh, s_impno;
            }
        }
    }
}


union in6_addr
{
    private union _in6_u_t
    {
        uint8_t[16] u6_addr8;
        uint16_t[8] u6_addr16;
        uint32_t[4] u6_addr32;
    }
    _in6_u_t in6_u;
    uint8_t[16] s6_addr8;
    uint16_t[8] s6_addr16;
    uint32_t[4] s6_addr32;
    alias s6_addr8 s6_addr;
}


const in6_addr IN6ADDR_ANY = { s6_addr8: [0] };
const in6_addr IN6ADDR_LOOPBACK = { s6_addr8: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1] };
//alias IN6ADDR_ANY IN6ADDR_ANY_INIT;
//alias IN6ADDR_LOOPBACK IN6ADDR_LOOPBACK_INIT;
    const uint INET_ADDRSTRLEN = 16;
const uint INET6_ADDRSTRLEN = 46;


struct sockaddr
{
    int16_t sa_family;
    ubyte[14] sa_data;
}


struct sockaddr_in
{
    int16_t sin_family = AF_INET;
    uint16_t sin_port;
    in_addr sin_addr;
    ubyte[8] sin_zero;
}


struct sockaddr_in6
{
    int16_t sin6_family = AF_INET6;
    uint16_t sin6_port;
    uint32_t sin6_flowinfo;
    in6_addr sin6_addr;
    uint32_t sin6_scope_id;
}
```

```
struct addrinfo
{
     int32_t ai_flags;
     int32_t ai_family;
     int32_t ai_socktype;
     int32_t ai_protocol;
     size_t ai_addrlen;
     char* ai_canonname;
     sockaddr* ai_addr;
     addrinfo* ai_next;
}


struct hostent
{
     char* h_name;
     char** h_aliases;
     int16_t h_addrtype;
     int16_t h_length;
     char** h_addr_list;
     char* h_addr()
     {
          return h_addr_list[0];
     }
}
}
/************************************* os.win.com
*****************************************************/

//alias WCHAR OLECHAR;
alias OLECHAR *LPOLESTR;
alias OLECHAR *LPCOLESTR;

enum
{
     rmm = 23,     // OLE 2 version number info
     rup = 639,
}

enum : int
{
     S_OK = 0,
     S_FALSE = 0x00000001,
     NOERROR = 0,
     E_NOTIMPL = cast(int)0x80004001,
     E_NOINTERFACE = cast(int)0x80004002,
     E_POINTER = cast(int)0x80004003,
     E_ABORT = cast(int)0x80004004,
     E_FAIL = cast(int)0x80004005,
     E_HANDLE = cast(int)0x80070006,
     CLASS_E_NOAGGREGATION = cast(int)0x80040110,
     E_OUTOFMEMORY = cast(int)0x8007000E,
     E_INVALIDARG = cast(int)0x80070057,
     E_UNEXPECTED = cast(int)0x8000FFFF,
}

struct GUID { // size is 16
align(1):
     DWORD Data1;
     WORD Data2;
     WORD Data3;
     BYTE Data4[8];
}
```

```
alias IID ИИД;
enum
{
     CLSCTX_INPROC_SERVER    = 0x1,
     CLSCTX_INPROC_HANDLER    = 0x2,
     CLSCTX_LOCAL_SERVER    = 0x4,
     CLSCTX_INPROC_SERVER16    = 0x8,
     CLSCTX_REMOTE_SERVER    = 0x10,
     CLSCTX_INPROC_HANDLER16    = 0x20,
     CLSCTX_INPROC_SERVERX86    = 0x40,
     CLSCTX_INPROC_HANDLERX86 = 0x80,

     CLSCTX_INPROC = (CLSCTX_INPROC_SERVER|CLSCTX_INPROC_HANDLER),
     CLSCTX_ALL = (CLSCTX_INPROC_SERVER| CLSCTX_INPROC_HANDLER|
     CLSCTX_LOCAL_SERVER),
     CLSCTX_SERVER = (CLSCTX_INPROC_SERVER|CLSCTX_LOCAL_SERVER),
}

alias GUID IID;
alias GUID CLSID;
/+
extern (C)
{
extern IID IID_IUnknown;
extern IID IID_IClassFactory;
extern IID IID_IMarshal;
extern IID IID_IMallocSpy;
extern IID IID_IStdMarshalInfo;
extern IID IID_IExternalConnection;
extern IID IID_IMultiQI;
extern IID IID_IEnumUnknown;
extern IID IID_IBindCtx;
extern IID IID_IEnumMoniker;
extern IID IID_IRunnableObject;
extern IID IID_IRunningObjectTable;
extern IID IID_IPersist;
extern IID IID_IPersistStream;
extern IID IID_IMoniker;
extern IID IID_IROTData;
extern IID IID_IEnumString;
extern IID IID_ISequentialStream;
extern IID IID_IStream;
extern IID IID_IEnumSTATSTG;
extern IID IID_IStorage;
extern IID IID_IPersistFile;
extern IID IID_IPersistStorage;
extern IID IID_ILockBytes;
extern IID IID_IEnumFORMATETC;
extern IID IID_IEnumSTATDATA;
extern IID IID_IRootStorage;
extern IID IID_IAdviseSink;
extern IID IID_IAdviseSink2;
extern IID IID_IDataObject;
extern IID IID_IDataAdviseHolder;
extern IID IID_IMessageFilter;
extern IID IID_IRpcChannelBuffer;
extern IID IID_IRpcProxyBuffer;
extern IID IID_IRpcStubBuffer;
extern IID IID_IPSFactoryBuffer;
extern IID IID_IPropertyStorage;
extern IID IID_IPropertySetStorage;
extern IID IID_IEnumSTATPROPSTG;
extern IID IID_IEnumSTATPROPSETSTG;
extern IID IID_IFillLockBytes;
```

```
extern IID IID_IProgressNotify;
extern IID IID_ILayoutStorage;
extern IID GUID_NULL;
extern IID IID_IRpcChannel;
extern IID IID_IRpcStub;
extern IID IID_IStubManager;
extern IID IID_IRpcProxy;
extern IID IID_IProxyManager;
extern IID IID_IPSFactory;
extern IID IID_IInternalMoniker;
extern IID IID_IDfReserved1;
extern IID IID_IDfReserved2;
extern IID IID_IDfReserved3;
extern IID IID_IStub;
extern IID IID_IProxy;
extern IID IID_IEnumGeneric;
extern IID IID_IEnumHolder;
extern IID IID_IEnumCallback;
extern IID IID_IOleManager;
extern IID IID_IOlePresObj;
extern IID IID_IDebug;
extern IID IID_IDebugStream;
extern IID IID_StdOle;
extern IID IID_ICreateTypeInfo;
extern IID IID_ICreateTypeInfo2;
extern IID IID_ICreateTypeLib;
extern IID IID_ICreateTypeLib2;
extern IID IID_IDispatch;
extern IID IID_IEnumVARIANT;
extern IID IID_ITypeComp;
extern IID IID_ITypeInfo;
extern IID IID_ITypeInfo2;
extern IID IID_ITypeLib;
extern IID IID_ITypeLib2;
extern IID IID_ITypeChangeEvents;
extern IID IID_IErrorInfo;
extern IID IID_ICreateErrorInfo;
extern IID IID_ISupportErrorInfo;
extern IID IID_IOleAdviseHolder;
extern IID IID_IOleCache;
extern IID IID_IOleCache2;
extern IID IID_IOleCacheControl;
extern IID IID_IParseDisplayName;
extern IID IID_IOleContainer;
extern IID IID_IOleClientSite;
extern IID IID_IOleObject;
extern IID IID_IOleWindow;
extern IID IID_IOleLink;
extern IID IID_IOleItemContainer;
extern IID IID_IOleInPlaceUIWindow;
extern IID IID_IOleInPlaceActiveObject;
extern IID IID_IOleInPlaceFrame;
extern IID IID_IOleInPlaceObject;
extern IID IID_IOleInPlaceSite;
extern IID IID_IContinue;
extern IID IID_IViewObject;
extern IID IID_IViewObject2;
extern IID IID_IDropSource;
extern IID IID_IDropTarget;
extern IID IID_IEnumOLEVERB;
}
+/
extern (System)
{
```

```
extern (Windows)
{
DWORD CoBuildVersion();

int StringFromGUID2(GUID *rguid, LPOLESTR lpsz, int cbMax);

/* init/uninit */

HRESULT CoInitialize(LPVOID pvReserved);
void CoUninitialize();
DWORD CoGetCurrentProcess();


HRESULT CoCreateInstance(CLSID *rclsid, IUnknown UnkOuter,
DWORD dwClsContext, IID* riid, void* ppv);

//HINSTANCE CoLoadLibrary(LPOLESTR lpszLibName, BOOL bAutoFree);
void CoFreeLibrary(HINSTANCE hInst);
void CoFreeAllLibraries();
void CoFreeUnusedLibraries();
}

interface IUnknown
{
HRESULT QueryInterface(IID* riid, void** pvObject);
ULONG AddRef();
ULONG Release();
}

interface IClassFactory : IUnknown
{
HRESULT CreateInstance(IUnknown UnkOuter, IID* riid, void** pvObject);
HRESULT LockServer(BOOL fLock);
}

class ComObject : IUnknown
{
extern (System):
HRESULT QueryInterface(IID* riid, void** ppv)
{
     if (*riid == cast(IID) IID_IUnknown)
     {
     *ppv = cast(void*)cast(IUnknown)this;
     AddRef();
     return S_OK;
     }
     else
     { *ppv = null;
     return E_NOINTERFACE;
     }
}

ULONG AddRef()
{
     return InterlockedIncrement(&count);
}

ULONG Release()
{
     LONG lRef = InterlockedDecrement(&count);
     if (lRef == 0)
     {
     // free object
```

```
    // If we delete this object, then the postinvariant called upon
    // return from Release() will fail.
    // Just let the GC reap it.
    //delete this;

    return 0;
    }
    return cast(ULONG)lRef;
}

LONG count = 0;         // object reference count
}

}

/************************************** os.win.stat
*************************************************/

extern (C):

// linux version is in linux

version (Windows)
{
const S_IFMT = 0xF000;
const S_IFDIR = 0x4000;
const S_IFCHR = 0x2000;
const S_IFIFO = 0x1000;
const S_IFREG = 0x8000;
const S_IREAD = 0x0100;
const S_IWRITE = 0x0080;
const S_IEXEC = 0x0040;
const S_IFBLK = 0x6000;
const S_IFNAM = 0x5000;

int S_ISREG(int m) { return (m & S_IFMT) == S_IFREG; }
int S_ISBLK(int m) { return (m & S_IFMT) == S_IFBLK; }
int S_ISNAM(int m) { return (m & S_IFMT) == S_IFNAM; }
int S_ISDIR(int m) { return (m & S_IFMT) == S_IFDIR; }
int S_ISCHR(int m) { return (m & S_IFMT) == S_IFCHR; }

struct struct_stat
{
short st_dev;
ushort st_ino;
ushort st_mode;
short st_nlink;
ushort st_uid;
ushort st_gid;
short st_rdev;
short dummy;
int st_size;
int st_atime;
int st_mtime;
int st_ctime;
}

int stat(char *, struct_stat *);
int fstat(int, struct_stat *);
int _wstat(wchar *, struct_stat *);
}
```

```
extern(Windows):
     struct SIZE
     {
          LONG cx;
          LONG cy;
     }
     alias SIZE* LPSIZE;
     struct POINTL
     {
          LONG x;
          LONG y;
     }
     alias POINTL* LPPOINTL;
     alias RECT* LPCRECT;
     alias HRESULT THEMEAPI;
     union LARGE_INTEGER
     {
          struct
          {
               DWORD LowPart;
               LONG HighPart;
          }
          private struct _U
          {
               DWORD LowPart;
               LONG HighPart;
          }
          _U u;
          LONGLONG QuadPart;
     }
     union ULARGE_INTEGER
     {
          struct
          {
               DWORD LowPart;
               DWORD HighPart;
          }
     private struct _U
          {
               DWORD LowPart;
               DWORD HighPart;
          }
          _U u;
          DWORDLONG QuadPart;
     }

     enum: UINT
     {
          SWP_NOSIZE = 0x0001,
          SWP_NOMOVE = 0x0002,
          SWP_NOZORDER = 0x0004,
          SWP_NOREDRAW = 0x0008,
          SWP_NOACTIVATE = 0x0010,
          SWP_FRAMECHANGED = 0x0020,
          SWP_SHOWWINDOW = 0x0040,
          SWP_HIDEWINDOW = 0x0080,
          SWP_NOCOPYBITS = 0x0100,
          SWP_NOOWNERZORDER = 0x0200,
          SWP_NOSENDCHANGING = 0x0400,
          SWP_DRAWFRAME = SWP_FRAMECHANGED,
          SWP_NOREPOSITION = SWP_NOOWNERZORDER,
          SWP_DEFERERASE = 0x2000,
          SWP_ASYNCWINDOWPOS = 0x4000,
     }
```

```
enum: UINT
{
	GW_HWNDFIRST = 0,
	GW_HWNDLAST = 1,
	GW_HWNDNEXT = 2,
	GW_HWNDPREV = 3,
	GW_OWNER = 4,
	GW_CHILD = 5,
}
enum: UINT
{
	DI_MASK = 0x0001,
	DI_IMAGE = 0x0002,
	DI_COMPAT = 0x0004,
	DI_DEFAULTSIZE = 0x0008,
	DI_NORMAL = DI_IMAGE | DI_MASK,
}
enum: цел
{
	GCL_MENUNAME = -8,
	GCL_HBRBACKGROUND = -10,
	GCL_HCURSOR = -12,
	GCL_HICON = -14,
	GCL_HMODULE = -16,
	GCL_CBWNDEXTRA = -18,
	GCL_CBCLSEXTRA = -20,
	GCL_WNDPROC = -24,
	GCL_STYLE = -26,
	GCW_ATOM = -32,
	GCL_HICONSM = -34,
}
enum: UINT
{
	SC_SIZE = 0xF000,
	SC_MOVE = 0xF010,
	SC_MINIMIZE = 0xF020,
	SC_MAXIMIZE = 0xF030,
	SC_CLOSE = 0xF060,
	SC_VSCROLL = 0xF070,
	SC_HSCROLL = 0xF080,
	SC_RESTORE = 0xF120,
	SC_SEPARATOR = 0xF00F,
}
enum: цел
{
	GWL_WNDPROC = -4,
	GWL_HINSTANCE = -6,
	GWL_HWNDPARENT = -8,
	GWL_STYLE = -16,
	GWL_EXSTYLE = -20,
	GWL_USERDATA = -21,
	GWL_ID = -12,
	DWL_MSGRESULT = 0,
	DWL_DLGPROC = 4,
	DWL_USER = 8,
}
enum: UINT
{
	WM_SETFONT = 0x0030,
	WM_GETFONT = 0x0031,
	WM_COMPACTING = 0x0041,
	WM_USER = 0x0400,
	WM_NEXTDLGCTL = 0x0028,
	WM_CAPTURECHANGED = 0x0215,
```

```
     WM_WINDOWPOSCHANGING = 0x0046,
     WM_WINDOWPOSCHANGED = 0x0047,
     WM_DRAWITEM = 0x002B,
     WM_DROPFILES = 0x0233,
     WM_PALETTECHANGED = 0x0311,
     WM_CLEAR = 0x0303,
     WM_CUT = 0x0300,
     WM_COPY = 0x0301,
     WM_PASTE = 0x0302,
     WM_MDIACTIVATE = 0x0222,
     WM_MDITILE = 0x0226,
     WM_MDICASCADE = 0x0227,
     WM_MDIICONARRANGE = 0x0228,
     WM_MDIGETACTIVE = 0x0229,
     WM_MOUSEWHEEL = 0x020A,
     WM_MOUSEHOVER = 0x02A1,
     WM_MOUSELEAVE = 0x02A3,
     WM_PRINT = 0x0317,
     WM_PRINTCLIENT = 0x0318,
     WM_MEASUREITEM = 0x002C,
     DM_SETDEFID = WM_USER + 1,
}
enum: UINT
{
BFFM_INITIALIZED = 1,
     BFFM_SETSELECTIONA = WM_USER + 102,
     BFFM_SETSELECTIONW = WM_USER + 103,
}
enum: UINT
{
     NM_FIRST = 0,
     NM_CLICK = NM_FIRST - 2,
     NM_CUSTOMDRAW = NM_FIRST - 12,
}
struct NMMOUSE
{
     NMHDR hdr;
     DWORD dwItemSpec;
     DWORD dwItemData;
     POINT pt;
     LPARAM dwHitInfo;
}
alias NMMOUSE* LPNMMOUSE;
enum: UINT
{
     TTM_ACTIVATE = WM_USER + 1,
     TTM_SETDELAYTIME = WM_USER + 3,
     TTM_ADDTOOLA = WM_USER + 4,
     TTM_DELTOOLA = WM_USER + 5,
     TTM_GETTOOLINFOA = WM_USER + 8,
     TTM_GETTEXTA = WM_USER + 11,
     TTM_UPDATETIPTEXTA = WM_USER + 12,
     TTM_ENUMTOOLSA = WM_USER + 14,
     TTM_GETCURRENTTOOLA = WM_USER + 15,
     TTM_ADDTOOLW = WM_USER + 50,
     TTM_GETTEXTW = WM_USER + 56,
     TTM_UPDATETIPTEXTW = WM_USER + 57,
}
enum: WPARAM
{
     TTDT_AUTOMATIC = 0,
     TTDT_RESHOW = 1,
     TTDT_AUTOPOP = 2,
     TTDT_INITIAL = 3,
```

```
}
// Rich edit.
enum: UINT
{
     ES_DISABLENOSCROLL = 0x00002000,
     EM_CANPASTE = WM_USER + 50,
     EM_EXGETSEL = WM_USER + 52,
     EM_EXLIMITTEXT = WM_USER + 53,
     EM_EXLINEFROMCHAR = WM_USER + 54,
     EM_EXSETSEL = WM_USER + 55,
     EM_GETCHARFORMAT = WM_USER + 58,
     EM_GETSELTEXT = WM_USER + 62,
     EM_PASTESPECIAL = WM_USER + 64,
     EM_SETBKGNDCOLOR = WM_USER + 67,
     EM_SETCHARFORMAT = WM_USER + 68,
     EM_SETEVENTMASK = WM_USER + 69,
     EM_STREAMIN = WM_USER + 73,
     EM_STREAMOUT = WM_USER + 74,
     EM_GETTEXTRANGE = WM_USER + 75,
     // 2.0
     EM_SETUNDOLIMIT = WM_USER + 82,
     EM_REDO = WM_USER + 84,
     EM_CANREDO = WM_USER + 85,
     EM_GETUNDONAME = WM_USER + 86,
     EM_GETREDONAME = WM_USER + 87,
     EM_STOPGROUPTYPING = WM_USER + 88,
     EM_SETTEXTMODE = WM_USER + 89,
     EM_GETTEXTMODE = WM_USER + 90,
     EM_AUTOURLDETECT = WM_USER + 91,
     EM_GETAUTOURLDETECT = WM_USER + 92,
     EM_SETPALETTE = WM_USER + 93,
     EM_GETTEXTEX = WM_USER + 94,
     EM_GETTEXTLENGTHEX = WM_USER + 95,
     EM_SHOWSCROLLBAR = WM_USER + 96,
     EM_SETTEXTEX = WM_USER + 97,
     EN_LINK = 0x070B,
}
// Rich edit.
enum: UINT
{
     SF_TEXT = 0x0001,
     SF_RTF = 0x0002,
     SF_RTFNOOBJS = 0x0003,
     SF_TEXTIZED = 0x0004,
     SFF_SELECTION = 0x8000,
     SFF_PLAINRTF = 0x4000,
     SCF_SELECTION = 0x0001,
     SCF_WORD = 0x0002,
     SCF_ALL = 0x0004,
     CFM_BOLD = 0x00000001,
     CFM_ITALIC = 0x00000002,
     CFM_UNDERLINE = 0x00000004,
     CFM_STRIKEOUT = 0x00000008,
     CFM_PROTECTED = 0x00000010,
     CFM_LINK = 0x00000020,
     CFM_SIZE = 0x80000000,
     CFM_COLOR = 0x40000000,
     CFM_FACE = 0x20000000,
     CFM_OFFSET = 0x10000000,
     CFM_CHARSET = 0x08000000,
     CFM_SMALLCAPS = 0x0040,
     CFM_ALLCAPS = 0x0080,
     CFM_HIDDEN = 0x0100,
     CFM_OUTLINE = 0x0200,
```

```
    CFM_SHADOW = 0x0400,
    CFM_EMBOSS = 0x0800,
    CFM_IMPRINT = 0x1000,
    CFM_DISABLED = 0x2000,
    CFM_REVISED = 0x4000,
    CFM_BACKCOLOR = 0x04000000,
    CFM_LCID = 0x02000000,
    CFM_UNDERLINETYPE = 0x00800000,
    CFM_WEIGHT = 0x00400000,
    CFM_SPACING = 0x00200000,
    CFM_KERNING = 0x00100000,
    CFM_STYLE = 0x00080000,
    CFM_ANIMATION = 0x00040000,
    CFM_REVAUTHOR = 0x00008000,
    CFE_BOLD = 0x0001,
    CFE_ITALIC = 0x0002,
    CFE_UNDERLINE = 0x0004,
    CFE_STRIKEOUT = 0x0008,
    CFE_PROTECTED = 0x0010,
    CFE_LINK = 0x0020,
    CFE_AUTOCOLOR = 0x40000000,
    CFE_AUTOBACKCOLOR    = CFM_BACKCOLOR,
    CFE_SUBSCRIPT = 0x00010000,
    CFE_SUPERSCRIPT = 0x00020000,
    CFM_SUBSCRIPT = CFE_SUBSCRIPT | CFE_SUPERSCRIPT,
    CFM_SUPERSCRIPT = CFM_SUBSCRIPT,
    CFU_UNDERLINE = 1,
    ENM_NONE = 0x00000000,
    ENM_CHANGE = 0x00000001,
    ENM_UPDATE = 0x00000002,
    ENM_LINK = 0x04000000,
    ENM_PROTECTED = 0x00200000,
}
enum: DWORD
{
    PRF_CLIENT = 0x00000004,
}
enum: DWORD
{
    STAP_ALLOW_NONCLIENT = 0x00000001,
    STAP_ALLOW_CONTROLS = 0x00000002,
    STAP_ALLOW_WEBCONTENT = 0x00000004,
}
enum: LPARAM
{
    ENDSESSION_LOGOFF = 0x80000000,
}
enum: цел
{
    BLACKONWHITE = 1,
    WHITEONBLACK = 2,
    COLORONCOLOR = 3,
    HALFTONE = 4,
}
enum: UINT
{
    CDN_FIRST = cast(UINT)-601,
    CDN_LAST = cast(UINT)-699,
    CDN_INITDONE = CDN_FIRST - 0x0000,
    CDN_SELCHANGE = CDN_FIRST - 0x0001,
    CDN_FOLDERCHANGE = CDN_FIRST - 0x0002,
    CDN_SHAREVIOLATION = CDN_FIRST - 0x0003,
    CDN_HELP = CDN_FIRST - 0x0004,
    CDN_FILEOK = CDN_FIRST - 0x0005,
```

```
    CDN_TYPECHANGE = CDN_FIRST - 0x0006,
    CDN_INCLUDEITEM = CDN_FIRST - 0x0007,
}
enum: DWORD
{
    OFN_READONLY = 0x00000001,
    OFN_OVERWRITEPROMPT = 0x00000002,
    OFN_HIDEREADONLY = 0x00000004,
    OFN_NOCHANGEDIR = 0x00000008,
    OFN_SHOWHELP = 0x00000010,
    OFN_ENABLEHOOK = 0x00000020,
    OFN_ENABLETEMPLATE = 0x00000040,
    OFN_ENABLETEMPLATEHANDLE = 0x00000080,
    OFN_NOVALIDATE = 0x00000100,
    OFN_ALLOWMULTISELECT = 0x00000200,
    OFN_EXTENSIONDIFFERENT = 0x00000400,
    OFN_PATHMUSTEXIST = 0x00000800,
    OFN_FILEMUSTEXIST = 0x00001000,
    OFN_CREATEPROMPT = 0x00002000,
    OFN_SHAREAWARE = 0x00004000,
    OFN_NOREADONLYRETURN = 0x00008000,
    OFN_NOTESTFILECREATE = 0x00010000,
    OFN_NONETWORKBUTTON = 0x00020000,
    OFN_NOLONGNAMES = 0x00040000,
    OFN_EXPLORER = 0x00080000,
    OFN_NODEREFERENCELINKS = 0x00100000,
    OFN_LONGNAMES = 0x00200000,
    OFN_ENABLEINCLUDENOTIFY = 0x00400000,
    OFN_ENABLESIZING = 0x00800000,
    OFN_DONTADDTORECENT = 0x02000000,
    OFN_FORCESHOWHIDDEN = 0x10000000,
}
enum: DWORD
{
    CF_SCREENFONTS = 0x00000001,
    CF_PRINTERFONTS = 0x00000002,
    CF_BOTH = CF_SCREENFONTS | CF_PRINTERFONTS,
    CF_SHOWHELP = 0x00000004,
    CF_ENABLEHOOK = 0x00000008,
    CF_ENABLETEMPLATE = 0x00000010,
    CF_ENABLETEMPLATEHANDLE = 0x00000020,
    CF_INITTOLOGFONTSTRUCT = 0x00000040,
    CF_USESTYLE = 0x00000080,
    CF_EFFECTS = 0x00000100,
    CF_APPLY = 0x00000200,
    CF_ANSIONLY = 0x00000400,
    CF_SCRIPTSONLY = CF_ANSIONLY,
    CF_NOVECTORFONTS = 0x00000800,
    CF_NOOEMFONTS = CF_NOVECTORFONTS,
    CF_NOSIMULATIONS = 0x00001000,
    CF_LIMITSIZE = 0x00002000,
    CF_FIXEDPITCHONLY = 0x00004000,
    CF_WYSIWYG = 0x00008000,
    CF_FORCEFONTEXIST = 0x00010000,
    CF_SCALABLEONLY = 0x00020000,
    CF_TTONLY = 0x00040000,
    CF_NOFACESEL = 0x00080000,
    CF_NOSTYLESEL = 0x00100000,
    CF_NOSIZESEL = 0x00200000,
    CF_SELECTSCRIPT = 0x00400000,
    CF_NOSCRIPTSEL = 0x00800000,
    CF_NOVERTFONTS = 0x01000000,
}
enum: UINT
```

```
{
    ODT_MENU = 1,
    ODT_LISTBOX = 2,
    ODT_COMBOBOX = 3,
    ODT_BUTTON = 4,
    ODT_STATIC = 5,
}
enum: цел
{
    HC_ACTION = 0,
}
enum: цел
{
    WH_GETMESSAGE = 3,
    WH_CALLWNDPROC = 4,
    WH_CALLWNDPROCRET = 12,
}
struct CWPSTRUCT
{
    LPARAM lParam;
    WPARAM wParam;
    UINT message;
    HWND hwnd;
}
alias CWPSTRUCT* LPCWPSTRUCT;
struct CWPRETSTRUCT
{
    LRESULT lResult;
    LPARAM lParam;
    WPARAM wParam;
    DWORD message;
    HWND hwnd;
}
alias CWPRETSTRUCT* LPCWPRETSTRUCT;
enum: UINT
{
    MDITILE_VERTICAL = 0x0000,
    MDITILE_HORIZONTAL = 0x0001,
    MDITILE_SKIPDISABLED = 0x0002,
    MDITILE_ZORDER = 0x0004,
}
enum: DWORD
{
    WS_EX_NOPARENTNOTIFY = 0x00000004,
    WS_EX_ACCEPTFILES = 0x00000010,
    WS_EX_TRANSPARENT = 0x00000020,
    WS_EX_RTLREADING = 0x00002000,
    WS_EX_APPWINDOW = 0x00040000,
    WS_EX_DLGMODALFRAME = 0x00000001,
    WS_EX_CONTROLPARENT = 0x00010000,
    WS_EX_WINDOWEDGE = 0x00000100,
    WS_EX_CLIENTEDGE = 0x00000200,
    WS_EX_TOOLWINDOW = 0x00000080,
    WS_EX_STATICEDGE = 0x00020000,
    WS_EX_CONTEXTHELP = 0x00000400,
    WS_EX_MDICHILD = 0x00000040,
    WS_EX_LAYERED = 0x00080000,
    WS_EX_TOPMOST = 0x00000008,
}
enum: DWORD
{
    TTS_ALWAYSTIP = 0x01,
    TTS_NOPREFIX = 0x02,
    TTS_NOANIMATE = 0x10, // IE5+
```

```
     TTS_NOFADE = 0x20, // IE5+
     TTS_BALLOON = 0x40, // IE5+
}
enum
{
     TTF_IDISHWND = 0x0001,
     TTF_CENTERTIP = 0x0002,
     TTF_RTLREADING = 0x0004,
     TTF_SUBCLASS = 0x0010,
     TTF_TRACK = 0x0020, // IE3+
     TTF_ABSOLUTE = 0x0080, // IE3+
     TTF_TRANSPARENT = 0x0100, // IE3+
     TTF_DI_SETITEM = 0x8000, // IE3+
}
enum: WPARAM
{
     SIZE_RESTORED = 0,
     SIZE_MINIMIZED = 1,
     SIZE_MAXIMIZED = 2,
     SIZE_MAXSHOW = 3,
     SIZE_MAXHIDE = 4,
}
enum: DWORD
{
     LWA_COLORKEY = 1,
     LWA_ALPHA = 2,
     AW_HOR_POSITIVE = 0x00000001,
     AW_HOR_NEGATIVE = 0x00000002,
     AW_VER_POSITIVE = 0x00000004,
     AW_VER_NEGATIVE = 0x00000008,
     AW_CENTER = 0x00000010,
     AW_HIDE = 0x00010000,
     AW_ACTIVATE = 0x00020000,
     AW_SLIDE = 0x00040000,
     AW_BLEND = 0x00080000,
}
enum: UINT
{
     MF_STRING = 0x00000000,
     MF_UNCHECKED = 0x00000000,
     MF_BYCOMMAND = 0x00000000,
     MF_GRAYED = 0x00000001,
     MF_CHECKED = 0x00000008,
     MF_POPUP = 0x00000010,
     MF_MENUBARBREAK = 0x00000020,
     MF_MENUBREAK = 0x00000040,
     MF_BYPOSITION = 0x00000400,
MF_SEPARATOR = 0x00000800,
     MF_DEFAULT = 0x00001000,
     MF_SYSMENU = 0x00002000,
     MFT_STRING = MF_STRING,
     MFT_MENUBARBREAK = MF_MENUBARBREAK,
     MFT_MENUBREAK = MF_MENUBREAK,
     MFT_RADIOCHECK = 0x00000200,
     MFT_SEPARATOR = MF_SEPARATOR,
     MFS_UNCHECKED = MF_UNCHECKED,
     MFS_CHECKED = MF_CHECKED,
     MFS_DEFAULT = MF_DEFAULT,
     MFS_GRAYED = MF_GRAYED,
     MIIM_STATE = 0x00000001,
     MIIM_ID = 0x00000002,
     MIIM_SUBMENU = 0x00000004,
     MIIM_TYPE = 0x00000010,
}
```

```
enum: цел
{
     RGN_AND = 1,
     RGN_OR = 2,
     RGN_XOR = 3,
     RGN_DIFF = 4,
     RGN_COPY = 5,
}
//alias UINT CLIPFORMAT; // ?
alias WORD CLIPFORMAT; // ?
// enum can't derive from HWND.
const HWND HWND_TOP = cast(HWND)0;
const HWND HWND_BOTTOM = cast(HWND)1;
const HWND HWND_TOPMOST = cast(HWND)-1;
const HWND HWND_NOTOPMOST = cast(HWND)-2;
enum: UINT
{
     CBS_SIMPLE = 0x0001,
     CBS_DROPDOWN = 0x0002,
     CBS_DROPDOWNLIST = 0x0003,
     CBS_AUTOHSCROLL = 0x0040,
     CBS_OWNERDRAWFIXED = 0x0010,
     CBS_OWNERDRAWVARIABLE = 0x0020,
}
enum: DWORD
{
     TME_HOVER = 1,
     TME_LEAVE = 2,
     TME_QUERY = 0x40000000,
     TME_CANCEL = 0x80000000,
}
const DWORD HOVER_DEFAULT = 0xFFFFFFFF;
enum: UINT
{
     TPM_LEFTBUTTON = 0x0000,
     TPM_RIGHTBUTTON = 0x0002,
     TPM_LEFTALIGN = 0x0000,
     TPM_CENTERALIGN = 0x0004,
     TPM_RIGHTALIGN = 0x0008,
     TPM_TOPALIGN = 0x0000,
     TPM_VCENTERALIGN = 0x0010,
     TPM_BOTTOMALIGN = 0x0020,
     TPM_HORIZONTAL = 0x0000,
     TPM_VERTICAL = 0x0040,
     TPM_NONOTIFY = 0x0080,
     TPM_RETURNCMD = 0x0100,
     TPM_RECURSE = 0x0001,
}
enum
{
     ICON_SMALL = 0,
     ICON_BIG = 1,
}
enum: UINT
{
     SPI_GETNONCLIENTMETRICS = 41,
     SPI_GETWORKAREA = 48,
     SPI_GETANIMATION = 72,
     SPI_GETWHEELSCROLLLINES = 104,
     SPI_GETWHEELSCROLLCHARS = 108,
     // ...
}
enum: DWORD
{
```

```
    ABM_GETTASKBARPOS = 0x00000005,
    // ...
}
enum: UINT
{
    ABE_LEFT = 0,
    ABE_TOP = 1,
    ABE_RIGHT = 2,
    ABE_BOTTOM = 3,
}
const LPSTR IDC_APPSTARTING = cast(LPSTR)32650;
const LPSTR IDC_HAND = cast(LPSTR)32649; // Windows 98+
const LPSTR IDC_HELP = cast(LPSTR)32651;
const LPSTR IDC_IBEAM = cast(LPSTR)32513;
const LPSTR IDC_NO = cast(LPSTR)32648;
const LPSTR IDC_SIZEALL = cast(LPSTR)32646;
const LPSTR IDC_SIZENESW = cast(LPSTR)32643;
const LPSTR IDC_SIZENS = cast(LPSTR)32645;
const LPSTR IDC_SIZENWSE = cast(LPSTR)32642;
const LPSTR IDC_SIZEWE = cast(LPSTR)32644;
const LPSTR IDC_WAIT = cast(LPSTR)32514;
enum: WORD
{
    MK_LBUTTON = 0x0001,
    MK_RBUTTON = 0x0002,
    MK_SHIFT = 0x0004,
    MK_CONTROL = 0x0008,
    MK_MBUTTON = 0x0010,
}
enum: UINT
{
    GMEM_MOVEABLE = 0x0002,
    GMEM_DDESHARE = 0x2000,
    GMEM_SHARE = 0x2000,
}
enum
{
    LOGPIXELSX = 88,
    LOGPIXELSY = 90,
}
enum
{
    MB_SERVICE_NOTIFICATION = 0x00200000,
}
enum
{
    DLGC_WANTARROWS = 0x0001,
    DLGC_WANTTAB = 0x0002,
    DLGC_WANTALLKEYS = 0x0004,
    DLGC_HASSETSEL = 0x0008,
    DLGC_RADIOBUTTON = 0x0040,
    DLGC_WANTCHARS = 0x0080,
    DLGC_STATIC = 0x0100,
}
enum
{
    LB_OKAY = 0,
    LB_ERR = -1,
    LB_ERRSPACE = -2,
}
enum: UINT
{
    LB_GETCOUNT = 0x018B,
    LB_GETITEMDATA = 0x0199,
```

```
    LB_ADDSTRING = 0x0180,
    LB_SETITEMDATA = 0x019A,
    LB_RESETCONTENT = 0x0184,
    LB_INSERTSTRING = 0x0181,
    LB_DELETESTRING = 0x0182,
    LB_GETHORIZONTALEXTENT = 0x0193,
    LB_SETHORIZONTALEXTENT = 0x0194,
    LB_SETITEMHEIGHT = 0x01A0,
    LB_GETITEMHEIGHT = 0x01A1,
    LB_GETSELCOUNT = 0x0190,
    LB_GETSELITEMS = 0x0191,
    LB_SETCURSEL = 0x0186,
    LB_GETCURSEL = 0x0188,
    LB_SETTOPINDEX = 0x0197,
    LB_GETTOPINDEX = 0x018E,
    LB_SELITEMRANGE = 0x0183,
    LB_SETSEL = 0x0185,
    LB_FINDSTRING = 0x018F,
    LB_FINDSTRINGEXACT = 0x01A2,
    LB_GETITEMRECT = 0x0198,
    LB_GETSEL = 0x0187,
    LB_ITEMFROMPOINT = 0x01A9,
    LB_ADDFILE = 0x0196,
    LB_DIR = 0x018D,
}
enum: DWORD
{
    LBS_NOINTEGRALHEIGHT = 0x0100,
    LBS_MULTICOLUMN = 0x0200,
    LBS_DISABLENOSCROLL = 0x1000,
    LBS_NOSEL = 0x4000,
    LBS_EXTENDEDSEL = 0x0800,
    LBS_MULTIPLESEL = 0x0008,
    LBS_SORT = 0x0002,
    LBS_USETABSTOPS = 0x0080,
    LBS_OWNERDRAWVARIABLE = 0x0020,
    LBS_OWNERDRAWFIXED = 0x0010,
    LBS_NOTIFY = 0x0001,
    LBS_HASSTRINGS = 0x0040,
}
enum
{
    LBN_ERRSPACE = -2,
    LBN_SELCHANGE = 1,
    LBN_DBLCLK = 2,
    LBN_SELCANCEL = 3,
    LBN_SETFOCUS = 4,
    LBN_KILLFOCUS = 5,
}
enum
{
    CB_OKAY = 0,
    CB_ERR = -1,
    CB_ERRSPACE = -2,
}
enum: UINT
{
    CB_SETCURSEL = 0x014E,
    CB_GETCURSEL = 0x0147,
    CB_FINDSTRING = 0x014C,
    CB_FINDSTRINGEXACT = 0x0158,
    CB_SETITEMHEIGHT = 0x0153,
    CB_GETITEMHEIGHT = 0x0154,
    CB_ADDSTRING = 0x0143,
```

```
    CB_DELETESTRING = 0x0144,
    CB_DIR = 0x0145,
    CB_INSERTSTRING = 0x014A,
    CB_RESETCONTENT = 0x014B,
    CB_SETITEMDATA = 0x0151,
    CB_GETDROPPEDWIDTH = 0x015f,
    CB_SETDROPPEDWIDTH = 0x0160,
    CB_LIMITTEXT = 0x0141,
    CB_GETEDITSEL = 0x0140,
    CB_SETEDITSEL = 0x0142,
    CB_SHOWDROPDOWN = 0x014F,
    CB_GETDROPPEDSTATE = 0x0157,
}
enum: DWORD
{
    CBS_SORT = 0x0100,
    CBS_HASSTRINGS = 0x0200,
    CBS_NOINTEGRALHEIGHT = 0x0400,
}
enum
{
    CBN_SELCHANGE = 1,
    CBN_SETFOCUS = 3,
    CBN_KILLFOCUS = 4,
    CBN_EDITCHANGE = 5,
}
enum: UINT
{
    TVE_COLLAPSE = 0x0001,
    TVE_EXPAND = 0x0002,
    TVE_TOGGLE = 0x0003,
}
enum: UINT
{
    TVIS_SELECTED = 0x0002,
    TVIS_EXPANDED = 0x0020,
}
enum: UINT
{
    TVIF_TEXT = 0x0001,
    TVIF_IMAGE = 0x0002,
    TVIF_PARAM = 0x0004,
    TVIF_STATE = 0x0008,
    TVIF_HANDLE = 0x0010,
    TVIF_SELECTEDIMAGE = 0x0020,
    TVIF_CHILDREN = 0x0040,
    TVIF_INTEGRAL = 0x0080, // IE4+
}
const цел I_CHILDRENCALLBACK = -1;
enum: UINT
{
    TVGN_FIRSTVISIBLE = 0x0005,
    TVGN_CARET = 0x0009,
}
enum: UINT
{
    TV_FIRST = 0x1100,
    TVM_INSERTITEMA = TV_FIRST + 0,
    TVM_DELETEITEM = TV_FIRST + 1,
    TVM_EXPAND = TV_FIRST + 2,
    TVM_GETITEMRECT = TV_FIRST + 4,
    TVM_GETINDENT = TV_FIRST + 6,
    TVM_SETINDENT = TV_FIRST + 7,
    TVM_SETIMAGELIST = TV_FIRST + 9,
```

```
    TVM_GETNEXTITEM = TV_FIRST + 10,
    TVM_SELECTITEM = TV_FIRST + 11,
    TVM_GETITEMA = TV_FIRST + 12,
    TVM_SETITEMA = TV_FIRST + 13,
    TVM_EDITLABELA = TV_FIRST + 14,
    TVM_GETVISIBLECOUNT = TV_FIRST + 16,
    TVM_HITTEST = TV_FIRST + 17,
    TVM_ENSUREVISIBLE = TV_FIRST + 20,
    TVM_SETITEMHEIGHT = TV_FIRST + 27, // IE4+
    TVM_GETITEMHEIGHT = TV_FIRST + 28, // IE4+
    TVM_INSERTITEMW = TV_FIRST + 50,
    TVM_SETITEMW = TV_FIRST + 63,
    TVN_FIRST = cast(UINT)-400,
    TVN_SELCHANGINGA = TVN_FIRST - 1,
    TVN_SELCHANGEDA = TVN_FIRST - 2,
    TVN_GETDISPINFOA = TVN_FIRST - 3,
    TVN_ITEMEXPANDINGA = TVN_FIRST - 5,
    TVN_ITEMEXPANDEDA = TVN_FIRST - 6,
    TVN_BEGINLABELEDITA = TVN_FIRST - 10,
    TVN_ENDLABELEDITA = TVN_FIRST - 11,
    TVN_SELCHANGINGW = TVN_FIRST - 50,
    TVN_SELCHANGEDW = TVN_FIRST - 51,
    TVN_GETDISPINFOW = TVN_FIRST - 52,
    TVN_ITEMEXPANDINGW = TVN_FIRST - 54,
    TVN_ITEMEXPANDEDW = TVN_FIRST - 55,
    TVN_BEGINLABELEDITW = TVN_FIRST - 59,
    TVN_ENDLABELEDITW = TVN_FIRST - 60,
}
enum: DWORD
{
    TVS_HASBUTTONS = 0x0001,
    TVS_HASLINES = 0x0002,
    TVS_LINESATROOT = 0x0004,
    TVS_EDITLABELS = 0x0008,
    TVS_SHOWSELALWAYS = 0x0020,
    TVS_CHECKBOXES = 0x0100, // IE3+
    TVS_TRACKSELECT = 0x0200, // IE3+
    TVS_FULLROWSELECT = 0x1000, // IE4+
    TVS_NOSCROLL = 0x2000, // IE4+
    TVS_SINGLEEXPAND = 0x0400, // IE4+
}
version(D_Version2)
{
    /+ // DMD 2.012: Error: cannot implicitly convert expression
    (cast(HANDLE)cast(проц*)-65536u) of type const(HANDLE) to цел
    const HTREEITEM TVI_ROOT = cast(HTREEITEM)-0x10000;
    const HTREEITEM TVI_FIRST = cast(HTREEITEM)-0x0FFFF;
    const HTREEITEM TVI_LAST = cast(HTREEITEM)-0x0FFFE;
    const HTREEITEM TVI_SORT = cast(HTREEITEM)-0x0FFFD;
    +/
    enum: HTREEITEM
    {
        TVI_ROOT = cast(HTREEITEM)-0x10000,
        TVI_FIRST = cast(HTREEITEM)-0x0FFFF,
        TVI_LAST = cast(HTREEITEM)-0x0FFFE,
        TVI_SORT = cast(HTREEITEM)-0x0FFFD,
    }
}
else
{
    const HTREEITEM TVI_ROOT = cast(HTREEITEM)-0x10000;
const HTREEITEM TVI_FIRST = cast(HTREEITEM)-0x0FFFF;
    const HTREEITEM TVI_LAST = cast(HTREEITEM)-0x0FFFE;
    const HTREEITEM TVI_SORT = cast(HTREEITEM)-0x0FFFD;
```

```
}
enum: UINT
{
     TVC_UNKNOWN = 0x0000,
     TVC_BYMOUSE = 0x0001,
     TVC_BYKEYBOARD = 0x0002,
}
enum: WPARAM
{
     TVSIL_NORMAL = 0,
     TVSIL_STATE = 2,
}
enum: UINT
{
     SB_SETTEXTA = WM_USER + 1,
     SB_SETPARTS = WM_USER + 4,
     SB_SIMPLE = WM_USER + 9,
     SB_SETTEXTW = WM_USER + 11,
}
enum: DWORD
{
     SBARS_SIZEGRIP = 0x0100,
}
enum: WPARAM
{
     SBT_NOBORDERS = 0x0100,
     SBT_POPOUT = 0x0200,
     SBT_RTLREADING = 0x0400,
     SBT_OWNERDRAW = 0x1000,
}
enum: LRESULT
{
     CDRF_DODEFAULT = 0x0,
     CDRF_NEWFONT = 0x2,
     CDRF_NOTIFYITEMDRAW = 0x20,
     CDRF_NOTIFYITEMERASE = 0x80,
}
enum: DWORD
{
     CDDS_ITEM = 0x00010000,
}
enum: UINT
{
     CDIS_SELECTED = 0x0001,
}
const LPWSTR LPSTR_TEXTCALLBACKW = cast(LPWSTR)-1L;
const LPSTR LPSTR_TEXTCALLBACKA = cast(LPSTR)-1L;
enum: UINT
{
     CCM_FIRST = 0x2000,
CCM_SETVERSION = CCM_FIRST + 0x7,
}
enum: UINT
{
     LVM_FIRST = 0x1000,
     LVM_SETBKCOLOR = LVM_FIRST + 1,
     LVM_SETIMAGELIST = LVM_FIRST + 3,
     LVM_SETITEMA = LVM_FIRST + 6,
     LVM_INSERTITEMA = LVM_FIRST + 7,
     LVM_DELETEITEM = LVM_FIRST + 8,
     LVM_DELETEALLITEMS = LVM_FIRST + 9,
     LVM_SETCALLBACKMASK = LVM_FIRST + 11,
     LVM_GETNEXTITEM = LVM_FIRST + 12,
     LVM_GETITEMRECT = LVM_FIRST + 14,
```

```
    LVM_ENSUREVISIBLE = LVM_FIRST + 19,
    LVM_REDRAWITEMS = LVM_FIRST + 21,
    LVM_ARRANGE = LVM_FIRST + 22,
    LVM_EDITLABELA = LVM_FIRST + 23,
    LVM_GETCOLUMNA = LVM_FIRST + 25,
    LVM_SETCOLUMNA = LVM_FIRST + 26,
    LVM_INSERTCOLUMNA = LVM_FIRST + 27,
    LVM_DELETECOLUMN = LVM_FIRST + 28,
    LVM_SETCOLUMNWIDTH = LVM_FIRST + 30,
    LVM_SETTEXTCOLOR = LVM_FIRST + 36,
    LVM_SETTEXTBKCOLOR = LVM_FIRST + 38,
    LVM_SETITEMSTATE = LVM_FIRST + 43,
    LVM_GETITEMSTATE = LVM_FIRST + 44,
    LVM_SETITEMTEXTA = LVM_FIRST + 46,
    LVM_SORTITEMS = LVM_FIRST + 48,
    LVM_SETEXTENDEDLISTVIEWSTYLE = LVM_FIRST + 54,
    LVM_GETEXTENDEDLISTVIEWSTYLE = LVM_FIRST + 55,
    LVM_INSERTITEMW = LVM_FIRST + 77,
    LVM_SETCOLUMNW = LVM_FIRST + 96,
    LVM_INSERTCOLUMNW = LVM_FIRST + 97,
    LVM_SETITEMTEXTW = LVM_FIRST + 116,
    LVM_EDITLABELW = LVM_FIRST + 118,
}
enum: UINT
{
    LVIS_OVERLAYMASK = 0x0F00,
    LVIS_STATEIMAGEMASK = 0xF000,
}
enum: цел
{
    LVSCW_AUTOSIZE = -1,
    LVSCW_AUTOSIZE_USEHEADER = -2,
}
enum: UINT
{
    LVNI_ALL = 0x0000,
    LVNI_FOCUSED = 0x0001,
    LVNI_SELECTED = 0x0002,
    LVNI_CUT = 0x0004,
    LVNI_DROPHILITED = 0x0008,
    LVNI_ABOVE = 0x0100,
    LVNI_BELOW = 0x0200,
    LVNI_TOLEFT = 0x0400,
    LVNI_TORIGHT = 0x0800,
}
enum: UINT
{
    LVN_FIRST = cast(UINT)-100,
    LVN_ITEMCHANGING = (LVN_FIRST - 0),
    LVN_ITEMCHANGED = (LVN_FIRST - 1),
    LVN_BEGINLABELEDITA = LVN_FIRST - 5,
    LVN_BEGINLABELEDITW = LVN_FIRST - 75,
    LVN_ENDLABELEDITA = LVN_FIRST - 6,
    LVN_ENDLABELEDITW = LVN_FIRST - 76,
    LVN_COLUMNCLICK = LVN_FIRST - 8,
    LVN_GETDISPINFOA = LVN_FIRST - 50,
    LVN_GETDISPINFOW = LVN_FIRST - 77,
}
enum: UINT
{
    LVCF_FMT = 0x0001,
    LVCF_WIDTH = 0x0002,
    LVCF_TEXT = 0x0004,
    LVCF_SUBITEM = 0x0008,
```

```
}
enum: цел
{
     LVCFMT_LEFT = 0x0000,
     LVCFMT_RIGHT = 0x0001,
     LVCFMT_CENTER = 0x0002,
     LVCFMT_JUSTIFYMASK = 0x0003,
}
enum: UINT
{
     LVIF_TEXT = 0x0001,
     LVIF_IMAGE = 0x0002,
     LVIF_PARAM = 0x0004,
     LVIF_STATE = 0x0008,
}
enum: UINT
{
     LVIS_FOCUSED = 0x0001,
     LVIS_SELECTED = 0x0002,
     LVIS_CUT = 0x0004,
     LVIS_DROPHILITED = 0x0008,
}
enum: цел
{
     LVA_DEFAULT = 0x0000,
     LVA_ALIGNLEFT = 0x0001,
     LVA_ALIGNTOP = 0x0002,
     LVA_SNAPTOGRID = 0x0005,
}
enum: цел
{
     LVIR_BOUNDS = 0,
     LVIR_ICON = 1,
     LVIR_LABEL = 2,
     LVIR_SELECTBOUNDS = 3,
}
enum: UINT
{
     LVS_ALIGNTOP = 0x0000,
     LVS_ALIGNLEFT = 0x0800,
LVS_ICON = 0x0000,
     LVS_REPORT = 0x0001,
     LVS_SMALLICON = 0x0002,
     LVS_LIST = 0x0003,
     LVS_SINGLESEL = 0x0004,
     LVS_SHOWSELALWAYS = 0x0008,
     LVS_SORTASCENDING = 0x0010,
     LVS_SORTDESCENDING = 0x0020,
     LVS_SHAREIMAGELISTS = 0x0040,
     LVS_NOLABELWRAP = 0x0080,
     LVS_AUTOARRANGE = 0x0100,
     LVS_EDITLABELS = 0x0200,
     LVS_OWNERDATA = 0x1000,
     LVS_NOSCROLL = 0x2000,
}
enum: DWORD
{
     LVS_EX_GRIDLINES = 0x00000001,
     LVS_EX_SUBITEMIMAGES = 0x00000002,
     LVS_EX_CHECKBOXES = 0x00000004,
     LVS_EX_TRACKSELECT = 0x00000008,
     LVS_EX_HEADERDRAGDROP = 0x00000010,
     LVS_EX_FULLROWSELECT = 0x00000020,
     LVS_EX_ONECLICKACTIVATE = 0x00000040,
```

```
     LVS_EX_TWOCLICKACTIVATE = 0x00000080,
     // IE4+
     LVS_EX_FLATSB = 0x00000100,
     LVS_EX_REGIONAL = 0x00000200,
     LVS_EX_INFOTIP = 0x00000400,
     LVS_EX_UNDERLINEHOT = 0x00000800,
     LVS_EX_UNDERLINECOLD = 0x00001000,
     LVS_EX_MULTIWORKAREAS = 0x00002000,
}
enum
{
     LVSIL_NORMAL = 0,
     LVSIL_SMALL = 1,
     LVSIL_STATE = 2,
}
enum
{
     I_IMAGECALLBACK = -1,
}
enum: UINT
{
     TCM_FIRST = 0x1300,
     TCM_SETITEMA = TCM_FIRST + 6,
     TCM_INSERTITEMA = TCM_FIRST + 7,
     TCM_DELETEITEM = TCM_FIRST + 8,
     TCM_DELETEALLITEMS = TCM_FIRST + 9,
     TCM_GETITEMRECT = TCM_FIRST + 10,
     TCM_GETCURSEL = TCM_FIRST + 11,
     TCM_SETCURSEL = TCM_FIRST + 12,
     TCM_SETITEMEXTRA = TCM_FIRST + 14,
     TCM_ADJUSTRECT = TCM_FIRST + 40,
     TCM_SETITEMSIZE = TCM_FIRST + 41,
     TCM_SETPADDING = TCM_FIRST + 43,
     TCM_GETROWCOUNT = TCM_FIRST + 44,
     TCM_SETTOOLTIPS = TCM_FIRST + 46,
     TCM_SETITEMW = TCM_FIRST + 61,
     TCM_INSERTITEMW = TCM_FIRST + 62,
}
enum: UINT
{
     TCIF_TEXT = 0x0001,
     TCIF_IMAGE = 0x0002,
     TCIF_RTLREADING = 0x0004,
     TCIF_PARAM = 0x0008,
}
enum: DWORD
{
     TCS_FORCEICONLEFT = 0x0010,
     TCS_FORCELABELLEFT = 0x0020,
     TCS_TABS = 0x0000,
     TCS_BUTTONS = 0x0100,
     TCS_SINGLELINE = 0x0000,
     TCS_MULTILINE = 0x0200,
     TCS_RIGHTJUSTIFY = 0x0000,
     TCS_FIXEDWIDTH = 0x0400,
     TCS_RAGGEDRIGHT = 0x0800,
     TCS_FOCUSONBUTTONDOWN = 0x1000,
     TCS_OWNERDRAWFIXED = 0x2000,
TCS_TOOLTIPS = 0x4000,
     TCS_FOCUSNEVER = 0x8000,
     // IE3+
     TCS_SCROLLOPPOSITE = 0x0001,
     TCS_BOTTOM = 0x0002,
     TCS_RIGHT = 0x0002,
```

```
    TCS_MULTISELECT = 0x0004,
    TCS_HOTTRACK = 0x0040,
    TCS_VERTICAL = 0x0080,
    // IE4+
    TCS_FLATBUTTONS = 0x0008,
}
enum: UINT
{
    TCN_FIRST = cast(UINT)-550,
    TCN_SELCHANGE = TCN_FIRST - 1,
    TCN_SELCHANGING = TCN_FIRST - 2,
}
enum
{
    HTERROR = -2,
    HTTRANSPARENT = -1,
    HTNOWHERE = 0,
    HTCLIENT = 1,
    HTCAPTION = 2,
    HTSYSMENU = 3,
    HTGROWBOX = 4,
    HTMENU = 5,
    HTHSCROLL = 6,
    HTVSCROLL = 7,
    HTMINBUTTON = 8,
    HTMAXBUTTON = 9,
    HTLEFT = 10,
    HTRIGHT = 11,
    HTTOP = 12,
    HTTOPLEFT = 13,
    HTTOPRIGHT = 14,
    HTBOTTOM = 15,
    HTBOTTOMLEFT = 16,
    HTBOTTOMRIGHT = 17,
    HTBORDER = 18,
    HTOBJECT = 19,
    HTCLOSE = 20,
    HTHELP = 21,
    HTSIZE = HTGROWBOX,
    HTREDUCE = HTMINBUTTON,
    HTZOOM = HTMAXBUTTON,
    HTSIZEFIRST = HTLEFT,
    HTSIZELAST = HTBOTTOMRIGHT,
}
enum
{
    WVR_VALIDRECTS = 0x0400,
}
enum: UINT
{
    NIF_MESSAGE = 0x00000001,
    NIF_ICON = 0x00000002,
    NIF_TIP = 0x00000004,
}
enum: DWORD
{
    NIM_ADD = 0x00000000,
    NIM_MODIFY = 0x00000001,
    NIM_DELETE = 0x00000002,
}
enum: DWORD
{
    VER_PLATFORM_WIN32s = 0,
    VER_PLATFORM_WIN32_WINDOWS = 1,
```

```
    VER_PLATFORM_WIN32_NT = 2,
}
enum: UINT
{
    SIF_RANGE = 0x0001,
    SIF_PAGE = 0x0002,
    SIF_POS = 0x0004,
    SIF_DISABLENOSCROLL = 0x0008,
    SIF_ALL = 23,
}
enum: UINT
{
    DFC_SCROLL = 3,
}
enum: UINT
{
    DFCS_SCROLLSIZEGRIP = 0x0008,
}
enum: UINT
{
    LR_LOADFROMFILE = 0x0010,
    LR_DEFAULTSIZE = 0x0040,
    LR_COPYFROMRESOURCE = 0x4000,
    LR_SHARED = 0x8000,
}
enum: COLORREF
{
    CLR_INVALID = 0xFFFFFFFF,
    CLR_NONE = CLR_INVALID,
}
enum: UINT
{
    DT_TOP = 0x00000000,
    DT_LEFT = 0x00000000,
    DT_CENTER = 0x00000001,
    DT_RIGHT = 0x00000002,
    DT_VCENTER = 0x00000004,
    DT_BOTTOM = 0x00000008,
    DT_WORDBREAK = 0x00000010,
    DT_SINGLELINE = 0x00000020,
    DT_EXPANDTABS = 0x00000040,
    DT_TABSTOP = 0x00000080,
    DT_NOCLIP = 0x00000100,
    DT_EXTERNALLEADING = 0x00000200,
    DT_CALCRECT = 0x00000400,
    DT_NOPREFIX = 0x00000800,
    DT_INTERNAL = 0x00001000,
    DT_EDITCONTROL = 0x00002000,
    DT_PATH_ELLIPSIS = 0x00004000,
    DT_END_ELLIPSIS = 0x00008000,
    DT_MODIFYSTRING = 0x00010000,
    DT_RTLREADING = 0x00020000,
    DT_WORD_ELLIPSIS = 0x00040000,
}
enum: UINT
{
    CF_TEXT = 1,
    CF_BITMAP = 2,
    CF_METAFILEPICT = 3,
    CF_SYLK = 4,
    CF_DIF = 5,
    CF_TIFF = 6,
    CF_OEMTEXT = 7,
    CF_DIB = 8,
```

```
    CF_PALETTE = 9,
    CF_PENDATA = 10,
    CF_RIFF = 11,
    CF_WAVE = 12,
    CF_UNICODETEXT = 13,
    CF_ENHMETAFILE = 14,
    CF_HDROP = 15,
    CF_LOCALE = 16,
}
enum: UINT
{
    BIF_RETURNONLYFSDIRS = 0x0001,
    BIF_NEWDIALOGSTYLE = 0x0040,
    BIF_NONEWFOLDERBUTTON = 0x0200, // shell32.dll 6.0+
}
enum
{
    TRANSPARENT = 1,
    OPAQUE = 2,
}
enum: UINT
{
    ETO_OPAQUE = 0x0002,
    ETO_CLIPPED = 0x0004,
}

enum: UINT
{
    IMAGE_BITMAP = 0,
    IMAGE_ICON = 1,
    IMAGE_CURSOR = 2,
}
const LPCSTR IDI_HAND = cast(LPCSTR)32513;
const LPCSTR IDI_QUESTION = cast(LPCSTR)32514;
const LPCSTR IDI_EXCLAMATION = cast(LPCSTR)32515;
const LPCSTR IDI_ASTERISK = cast(LPCSTR)32516;
const LPCSTR IDI_INFORMATION = IDI_ASTERISK;

//private import os.win.base.native: RT_STRING;//

enum: LONG
{
    HS_HORIZONTAL = 0,
    HS_VERTICAL = 1,
    HS_FDIAGONAL = 2,
    HS_BDIAGONAL = 3,
    HS_CROSS = 4,
    HS_DIAGCROSS = 5,
}
enum: DWORD
{
    LOAD_LIBRARY_AS_DATAFILE = 0x00000002,
}
enum: UINT
{
    PBM_SETRANGE = WM_USER + 1,
    PBM_SETPOS = WM_USER + 2,
    PBM_DELTAPOS = WM_USER + 3,
    PBM_SETSTEP = WM_USER + 4,
    PBM_STEPIT = WM_USER + 5,
}
const DWORD MAX_COMPUTERNAME_LENGTH = 15;
const DWORD LF_FACESIZE = 32;
typedef HANDLE HIMAGELIST;
```

```
enum: UINT
{
     ILD_NORMAL = 0,
}
enum: UINT
{
     //ILC_COLOR = ,
     ILC_COLOR4 = 0x0004,
     ILC_COLOR8 = 0x0008,
     ILC_COLOR16 = 0x0010,
     ILC_COLOR24 = 0x0018,
     ILC_COLOR32 = 0x0020,
     ILC_MASK = 0x0001,
}
// Rich edit.
alias DWORD function(/+ DWORD_PTR +/ DWORD dwCookie, LPBYTE pbBuff, LONG
cb, LONG* pcb) EDITSTREAMCALLBACK;
alias DWORD LCID;
struct WINDOWPOS
{
     HWND hwnd;
     HWND hwndInsertAfter;
     цел x;
     цел y;
     цел cx;
     цел cy;
     UINT флаги;
}
alias WINDOWPOS* LPWINDOWPOS;
alias WINDOWPOS* PWINDOWPOS;
struct WNDCLASSW
{
     UINT стиль;
     WNDPROC lpfnWndProc;
     цел cbClsExtra;
     цел cbWndExtra;
     HANDLE hInstance;
     HICON hIcon;
     HCURSOR hCursor;
     HBRUSH hbrBackground;
     LPCWSTR lpszMenuName;
     LPCWSTR lpszClassName;
}
alias WNDCLASSW* LPWNDCLASSW;
struct OSVERSIONINFOA
{
     DWORD dwOSVersionInfoSize;
     DWORD dwMajorVersion;
     DWORD dwMinorVersion;
     DWORD dwBuildNumber;
     DWORD dwPlatformId;
     CHAR[128] szCSDVersion;
}
alias OSVERSIONINFOA* LPOSVERSIONINFOA;
const HWND HWND_MESSAGE = cast(HWND)-3; // Win2000/XP only.
struct NOTIFYICONDATA
{
     DWORD cbSize;
     HWND hWnd;
     UINT uID;
     UINT uFlags;
     UINT uCallbackMessage;
     HICON hIcon;
     char[64] szTip;
```

```
}
alias NOTIFYICONDATA* PNOTIFYICONDATA;
// Unaligned!
struct SHITEMID
{
     align(1):
     USHORT cb; // Размер including cb.
     BYTE[1] abID;
}
alias SHITEMID* PSHITEMID;
alias SHITEMID* LPSHITEMID;
alias SHITEMID* LPCSHITEMID;
struct ITEMIDLIST
{
     SHITEMID mkid;
}
alias ITEMIDLIST* PITEMIDLIST;
alias ITEMIDLIST* LPITEMIDLIST;
alias ITEMIDLIST* LPCITEMIDLIST;
alias цел function(HWND hwnd, UINT uMsg, LPARAM lParam, LPARAM lpData)
BFFCALLBACK;
struct BROWSEINFOA
{
     HWND hwndOwner;
     LPCITEMIDLIST pidlRoot;
     LPSTR pszDisplayName;
     LPCSTR lpszTitle;
     UINT ulFlags;
     BFFCALLBACK lpfn;
     LPARAM lParam;
     цел iImage;
}
alias BROWSEINFOA* PBROWSEINFOA;
alias BROWSEINFOA* LPBROWSEINFOA;
struct BROWSEINFOW
{
     HWND hwndOwner;
     LPCITEMIDLIST pidlRoot;
     LPWSTR pszDisplayName;
     LPCWSTR lpszTitle;
     UINT ulFlags;
     BFFCALLBACK lpfn;
     LPARAM lParam;
     цел iImage;
}
alias BROWSEINFOW* PBROWSEINFOW;
alias BROWSEINFOW* LPBROWSEINFOW;
struct LOGBRUSH
{
     UINT lbStyle;
     COLORREF lbColor;
LONG lbHatch;
}
alias LOGBRUSH* LPLOGBRUSH;
struct DRAWTEXTPARAMS
{
     UINT cbSize;
     цел iTabLength;
     цел iLeftMargin;
     цел iRightMargin;
     UINT uiLengthDrawn;
}
alias DRAWTEXTPARAMS* LPDRAWTEXTPARAMS;
struct NMHDR
```

```
{
     HWND hwndFrom;
     UINT idFrom;
     UINT code;
}
alias NMHDR* LPNMHDR;
struct NMCUSTOMDRAW
{
     NMHDR hdr;
     DWORD dwDrawStage;
     HDC hdc;
     RECT rc;
     /+ DWORD_PTR +/ DWORD dwItemSpec;
     UINT uItemState;
     LPARAM lItemlParam;
}
alias NMCUSTOMDRAW* LPNMCUSTOMDRAW;
struct NMTVCUSTOMDRAW
{
     NMCUSTOMDRAW nmcd;
     COLORREF clrText;
     COLORREF clrTextBk;
     цел iLevel; // IE4+
}
alias NMTVCUSTOMDRAW* LPNMTVCUSTOMDRAW;
struct NM_LISTVIEW
{
     NMHDR hdr;
     цел iItem;
     цел iSubItem;
     UINT uNewState;
UINT uOldState;
     UINT uChanged;
     POINT ptAction;
     LPARAM lParam;
}
struct LVITEMA
{
     UINT mask;
     цел iItem;
     цел iSubItem;
     UINT state;
     UINT stateMask;
     LPSTR pszText;
     цел cchTextMax;
     цел iImage;
     LPARAM lParam;
}
alias LVITEMA* LPLVITEMA;
alias LVITEMA* PLVITEMA;
alias LVITEMA LV_ITEMA;
alias LVITEMA* LPLV_ITEMA;
alias LVITEMA* PLV_ITEMA;
struct LVITEMW
{
     UINT mask;
     цел iItem;
     цел iSubItem;
     UINT state;
     UINT stateMask;
     LPWSTR pszText;
     цел cchTextMax;
     цел iImage;
     LPARAM lParam;
```

```
}
alias LVITEMW* LPLVITEMW;
alias LVITEMW* PLVITEMW;
alias LVITEMW LV_ITEMW;
alias LVITEMW* LPLV_ITEMW;
alias LVITEMW* PLV_ITEMW;
struct LVDISPINFOA
{
     NMHDR hdr;
     LVITEMA item;
}
alias LVDISPINFOA* LPLVDISPINFOA;
alias LVDISPINFOA* PLVDISPINFOA;
alias LVDISPINFOA LV_DISPINFOA;
alias LVDISPINFOA* LPLV_DISPINFOA;
alias LVDISPINFOA* PLV_DISPINFOA;
struct LVDISPINFOW
{
     NMHDR hdr;
     LVITEMW item;
}
alias LVDISPINFOW* LPLVDISPINFOW;
alias LVDISPINFOW* PLVDISPINFOW;
alias LVDISPINFOW LV_DISPINFOW;
alias LVDISPINFOW* LPLV_DISPINFOW;
alias LVDISPINFOW* PLV_DISPINFOW;
struct LVCOLUMNA
{
     UINT mask;
     цел fmt;
     цел cx;
     LPSTR pszText;
     цел cchTextMax;
     цел iSubItem;
}
alias LVCOLUMNA* LPLVCOLUMNA;
alias LVCOLUMNA* PLVCOLUMNA;
alias LVCOLUMNA LV_COLUMNA;
alias LVCOLUMNA* LPLV_COLUMNA;
alias LVCOLUMNA* PLV_COLUMNA;
struct LVCOLUMNW
{
     UINT mask;
     цел fmt;
     цел cx;
     LPWSTR pszText;
     цел cchTextMax;
     цел iSubItem;
}
alias LVCOLUMNW* LPLVCOLUMNW;
alias LVCOLUMNW* PLVCOLUMNW;
alias LVCOLUMNW LV_COLUMNW;
alias LVCOLUMNW* LPLV_COLUMNW;
alias LVCOLUMNW* PLV_COLUMNW;
struct TBBUTTON
{
     цел iBitmap;
     цел idCommand;
     BYTE fsState;
     BYTE fsStyle;
     BYTE[2] bReserved;
DWORD dwData;
     цел iString;
}
```

```
alias TBBUTTON* PTBBUTTON, LPTBBUTTON, LPCTBBUTTON;
struct NMTOOLBARA
{
    NMHDR hdr;
    цел iItem;
    TBBUTTON tbButton;
    цел cchText;
    LPSTR pszText;
    RECT rcButton;
}
alias NMTOOLBARA* LPNMTOOLBARA;
struct NMTOOLBARW
{
    NMHDR hdr;
    цел iItem;
    TBBUTTON tbButton;
    цел cchText;
    LPWSTR pszText;
    RECT rcButton;
}
alias NMTOOLBARW* LPNMTOOLBARW;
enum: BYTE
{
    TBSTYLE_BUTTON = 0x00,
    TBSTYLE_SEP = 0x01,
    TBSTYLE_CHECK = 0x02,
    TBSTYLE_GROUP = 0x04,
    TBSTYLE_DROPDOWN = 0x08,
    TBSTYLE_AUTOSIZE = 0x10,
    /+
    // The following are too big for TBBUTTON.fsStyle
    TBSTYLE_TOOLTIPS = 0x0100,
    TBSTYLE_WRAPABLE = 0x0200,
    TBSTYLE_ALTDRAG = 0x0400,
    +/
}
enum: BYTE
{
    //BTNS_AUTOSIZE = TBSTYLE_AUTOSIZE,
    BTNS_WHOLEDROPDOWN = 0x80,
}
enum: BYTE
{
    TBSTATE_CHECKED = 0x01,
    TBSTATE_PRESSED = 0x02,
    TBSTATE_ENABLED = 0x04,
    TBSTATE_HIDDEN = 0x08,
    TBSTATE_INDETERMINATE = 0x10,
    TBSTATE_WRAP = 0x20,
    TBSTATE_ELLIPSES = 0x40,
    TBSTATE_MARKED = 0x80,
}
/*enum: LRESULT
{
    TBDDRET_DEFAULT = 0,
    TBDDRET_NODEFAULT = 1,
    TBDDRET_TREATPRESSED = 2,
}*/
enum: UINT
{
    TB_SETSTATE = WM_USER + 17,
    TB_ADDBUTTONSA = WM_USER + 20,
    TB_INSERTBUTTONA = WM_USER + 21,
    TB_DELETEBUTTON = WM_USER + 22,
```

```
     TB_GETITEMRECT = WM_USER + 29,
     TB_BUTTONSTRUCTSIZE = WM_USER + 30,
     TB_SETBUTTONSIZE = WM_USER + 31,
     TB_AUTOSIZE = WM_USER + 33,
     TB_SETIMAGELIST = WM_USER + 48,
     TB_INSERTBUTTONW = WM_USER + 67,
     TB_ADDBUTTONSW = WM_USER + 68,
     TB_SETPADDING = WM_USER + 87,
}
enum: UINT
{
     TBN_FIRST = cast(UINT)-700,
     TBN_DROPDOWN = TBN_FIRST - 10,
}
struct TVITEMA
{
     UINT mask;
     HTREEITEM hItem;
     UINT state;
     UINT stateMask;
     LPSTR pszText;
цел cchTextMax;
     цел iImage;
     цел iSelectedImage;
     цел cChildren;
     LPARAM lParam;
}
alias TVITEMA* LPTVITEMA;
alias TVITEMA* PTVITEMA;
alias TVITEMA TV_ITEMA;
alias TVITEMA* LPTV_ITEMA;
alias TVITEMA* PTV_ITEMA;
struct TVITEMW
{
     UINT mask;
     HTREEITEM hItem;
     UINT state;
     UINT stateMask;
     LPWSTR pszText;
     цел cchTextMax;
     цел iImage;
     цел iSelectedImage;
     цел cChildren;
     LPARAM lParam;
}
alias TVITEMW* LPTVITEMW;
alias TVITEMW* PTVITEMW;
alias TVITEMW TV_ITEMW;
alias TVITEMW* LPTV_ITEMW;
alias TVITEMW* PTV_ITEMW;
struct TVHITTESTINFO
{
     POINT pt;
     UINT флаги;
     HTREEITEM hItem;
}
alias TVHITTESTINFO* LPTVHITTESTINFO;
struct TVINSERTSTRUCTA
{
     HTREEITEM hParent;
     HTREEITEM hInsertAfter;
     TV_ITEMA item;
}
alias TVINSERTSTRUCTA* LPTVINSERTSTRUCTA;
```

```
alias TVINSERTSTRUCTA TV_INSERTSTRUCTA;
alias TVINSERTSTRUCTA* LPTV_INSERTSTRUCTA;
struct NMTREEVIEWA
{
    NMHDR hdr;
    UINT действие;
    TVITEMA itemOld;
    TVITEMA itemNew;
    POINT ptDrag;
}
alias NMTREEVIEWA* LPNMTREEVIEWA;
alias NMTREEVIEWA NM_TREEVIEW;
alias NMTREEVIEWA* LPNM_TREEVIEW;
struct NMTVDISPINFOA
{
    NMHDR hdr;
    TVITEMA item;
}
alias NMTVDISPINFOA* LPNMTVDISPINFOA;
alias NMTVDISPINFOA TV_DISPINFOA;
alias NMTVDISPINFOA* LPTV_DISPINFOA;
struct NMTVDISPINFOW
{
    NMHDR hdr;
    TVITEMW item;
}
alias NMTVDISPINFOW* LPNMTVDISPINFOW;
alias NMTVDISPINFOW TV_DISPINFOW;
alias NMTVDISPINFOW* LPTV_DISPINFOW;
struct TCITEMA
{
    UINT mask;
    UINT lpReserved1;
    UINT lpReserved2;
    LPSTR pszText;
    цел cchTextMax;
    цел iImage;
    LPARAM lParam;
}
alias TCITEMA* LPTCITEMA;
alias TCITEMA TC_ITEMA;
alias TCITEMA* LPTC_ITEMA;
struct TCITEMW
{
    UINT mask;
    UINT lpReserved1;
    UINT lpReserved2;
    LPWSTR pszText;
    цел cchTextMax;
    цел iImage;
    LPARAM lParam;
}
alias TCITEMW* LPTCITEMW;
alias TCITEMW TC_ITEMW;
alias TCITEMW* LPTC_ITEMW;
// Rich edit.
struct CHARRANGE
{
    LONG cpMin;
    LONG cpMax;
}
// Rich edit.
struct EDITSTREAM
{
```

```
    /+ DWORD_PTR +/ DWORD dwCookie;
    DWORD dwError;
    EDITSTREAMCALLBACK pfnCallback;
}
// Rich edit.
struct CHARFORMAT2A
{
    UINT cbSize;
    DWORD dwMask;
    DWORD dwEffects;
    LONG yHeight;
    LONG yOffset;
    COLORREF crTextColor;
    BYTE bCharSet;
    BYTE bPitchAndFamily;
    char[LF_FACESIZE] szFaceName;
    WORD wWeight;
    SHORT sSpacing;
    COLORREF crBackColor;
    LCID lcid;
    DWORD dwReserved;
    SHORT sStyle;
    WORD wKerning;
    BYTE bUnderlineType;
    BYTE bAnimation;
    BYTE bRevAuthor;
    BYTE bReserved1;
}
static assert(CHARFORMAT2A.sizeof == 84);
// Rich edit.
struct ENLINK
{
    NMHDR nmhdr;
    UINT cooб;
    WPARAM wParam;
    LPARAM lParam;
    CHARRANGE chrg;
}
struct TEXTRANGEA
{
    CHARRANGE chrg;
    LPSTR lpstrText;
}
struct MENUITEMINFOA
{
    UINT cbSize;
    UINT fMask;
    UINT fType;
    UINT fState;
    UINT wID;
    HMENU hSubMenu;
    HBITMAP hbmpChecked;
    HBITMAP hbmpUnchecked;
    DWORD dwItemData;
    LPSTR dwTypeData;
    UINT cch;
    //HBITMAP hbmpItem;
}
alias MENUITEMINFOA* LPMENUITEMINFOA;
struct MENUITEMINFOW
{
    UINT cbSize;
    UINT fMask;
    UINT fType;
```

```
    UINT fState;
    UINT wID;
    HMENU hSubMenu;
    HBITMAP hbmpChecked;
    HBITMAP hbmpUnchecked;
    DWORD dwItemData;
    LPWSTR dwTypeData;
    UINT cch;
    //HBITMAP hbmpItem;
}
alias MENUITEMINFOW* LPMENUITEMINFOW;
struct SCROLLINFO
{
    UINT cbSize;
    UINT fMask;
    цел nMin;
    цел nMax;
    UINT nPage;
    цел nPos;
    цел nTrackPos;
}
alias SCROLLINFO* LPSCROLLINFO;
alias UINT function(HWND hdlg, UINT uiMsg, WPARAM wParam, LPARAM lParam)
LPCCHOOKPROC;
alias UINT function(HWND hdlg, UINT uiMsg, WPARAM wParam, LPARAM lParam)
LPCFHOOKPROC;
alias BOOL function(HDC hdc, LPARAM lpData, цел cchData) GRAYSTRINGPROC;
enum: DWORD
{
    CC_RGBINIT = 0x00000001,
    CC_FULLOPEN = 0x00000002,
    CC_PREVENTFULLOPEN = 0x00000004,
    CC_SHOWHELP = 0x00000008,
    CC_ENABLEHOOK = 0x00000010,
    CC_ENABLETEMPLATE = 0x00000020,
    CC_ENABLETEMPLATEHANDLE = 0x00000040,
    CC_SOLIDCOLOR = 0x00000080,
    CC_ANYCOLOR = 0x00000100,
}
struct CHOOSECOLORA
{
    DWORD lStructSize;
    HWND hwndOwner;
    HWND hInstance;
    COLORREF rgbResult;
    COLORREF* lpCustColors;
    DWORD Flags;
    LPARAM lCustData;
    LPCCHOOKPROC lpfnHook;
    LPCSTR lpTemplateName;
}
alias CHOOSECOLORA* PCHOOSECOLORA;
alias CHOOSECOLORA* LPCHOOSECOLORA;
struct LOGFONTW
{
    LONG lfHeight;
    LONG lfWidth;
    LONG lfEscapement;
    LONG lfOrientation;
    LONG lfWeight;
    BYTE lfItalic;
    BYTE lfUnderline;
    BYTE lfStrikeOut;
    BYTE lfCharSet;
```

```
    BYTE lfOutPrecision;
    BYTE lfClipPrecision;
    BYTE lfQuality;
    BYTE lfPitchAndFamily;
    WCHAR[32] lfFaceName;
}
alias LOGFONTW* PLOGFONTW;
alias LOGFONTW* LPLOGFONTW;
struct NONCLIENTMETRICSA
{
    UINT cbSize;
    цел iBorderWidth;
    цел iScrollWidth;
    цел iScrollHeight;
    цел iCaptionWidth;
    цел iCaptionHeight;
    LOGFONTA lfCaptionFont;
    цел iSmCaptionWidth;
    цел iSmCaptionHeight;
    LOGFONTA lfSmCaptionFont;
    цел iMenuWidth;
    цел iMenuHeight;
    LOGFONTA lfMenuFont;
    LOGFONTA lfStatusFont;
    LOGFONTA lfMessageFont;
}
alias NONCLIENTMETRICSA LPNONCLIENTMETRICSA;
struct CHOOSEFONTW
{
    align(1):
    DWORD lStructSize;
    HWND hwndOwner;
    HDC hDC;
    LPLOGFONTW lpLogFont;
    INT iPointSize;
    DWORD Flags;
    DWORD rgbColors;
    LPARAM lCustData;
    LPCFHOOKPROC lpfnHook;
    LPCWSTR lpTemplateName;
    экз hInstance;
    LPWSTR lpszStyle;
    WORD nFontType;
    WORD ___MISSING_ALIGNMENT__;
    INT nSizeMin;
    INT nSizeMax;
}
alias CHOOSEFONTW* PCHOOSEFONTW;
alias CHOOSEFONTW* LPCHOOSEFONTW;
struct CHOOSEFONTA
{
    align(1):
    DWORD lStructSize;
    HWND hwndOwner;
    HDC hDC;
    LPLOGFONTA lpLogFont;
    INT iPointSize;
    DWORD Flags;
    DWORD rgbColors;
    LPARAM lCustData;
    LPCFHOOKPROC lpfnHook;
    LPCSTR lpTemplateName;
    экз hInstance;
    LPSTR lpszStyle;
```

```
    WORD nFontType;
    WORD ___MISSING_ALIGNMENT__;
    INT nSizeMin;
    INT nSizeMax;
}
alias CHOOSEFONTA* PCHOOSEFONTA;
alias CHOOSEFONTA* LPCHOOSEFONTA;
struct ICONINFO
{
    BOOL fIcon;
    DWORD xHotspot;
    DWORD yHotspot;
    HBITMAP hbmMask;
    HBITMAP hbmColor;
}
alias ICONINFO* LPICONINFO;
alias ICONINFO* PICONINFO;
struct MINMAXINFO
{
    POINT ptReserved;
    POINT ptMaxSize;
    POINT ptMaxPosition;
    POINT ptMinTrackSize;
    POINT ptMaxTrackSize;
}
alias MINMAXINFO* LPMINMAXINFO;
alias MINMAXINFO* PMINMAXINFO;
struct NCCALCSIZE_PARAMS
{
    RECT[3] rgrc;
    PWINDOWPOS lppos;
}
alias NCCALCSIZE_PARAMS* LPNCCALCSIZE_PARAMS;
struct CREATESTRUCTA
{
    LPVOID lpCreateParams;
    экз hInstance;
    HMENU hMenu;
    HWND hwndParent;
    цел cy;
    цел cx;
    цел y;
    цел x;
    LONG стиль;
    LPCSTR lpszName;
    LPCSTR lpszClass;
    DWORD dwExStyle;
}
alias CREATESTRUCTA* LPCREATESTRUCTA;
struct ACTCTXW
{
    ULONG cbSize;
    DWORD dwFlags;
    LPCWSTR lpSource;
    USHORT wProcessorArchitecture;
    LANGID wLangId;
    LPCWSTR lpAssemblyDirectory;
    LPCWSTR lpResourceName;
    LPCWSTR lpApplicationName;
    HMODULE hModule;
}
alias ACTCTXW* PACTCTXW;
alias ACTCTXW* LPACTCTXW;
struct HELPINFO
```

```
{
     UINT cbSize;
     цел iКонтекстType;
     цел iCtrlId;
     HANDLE hItemHandle;
     DWORD dwКонтекстId;
     POINT MousePos;
}
alias HELPINFO* LPHELPINFO;
struct TOOLINFOA
{
     UINT cbSize;
     UINT uFlags;
     HWND hwnd;
     UINT uId;
     RECT rect;
     экз hinst;
     LPSTR lpszText;
}
alias TOOLINFOA* PTOOLINFOA;
alias TOOLINFOA* LPTOOLINFOA;
struct STYLESTRUCT
{
     DWORD styleOld;
     DWORD styleNew;
}
alias STYLESTRUCT* LPSTYLESTRUCT;
extern(C) DWORD MAKELONG(WORD wLow, WORD wHigh)
{
     return wLow | (wHigh << 16);
}
alias MAKELONG MAKELPARAM;
alias MAKELONG MAKEWPARAM;
alias MAKELONG MAKELRESULT;
const цел DLGWINDOWEXTRA = 30;
extern(C) COLORREF RGB(цел r, цел g , цел b)
{
     return cast(COLORREF)(cast(BYTE)r |
          cast(WORD)(cast(BYTE)g << 8) |
          cast(DWORD)(cast(BYTE)b << 16));
}
struct DRAWITEMSTRUCT
{
     UINT CtlType;
     UINT CtlID;
     UINT itemID;
     UINT itemAction;
     UINT itemState;
     HWND hwndItem;
     HDC hDC;
     RECT rcItem;
     DWORD itemData;
}
alias DRAWITEMSTRUCT* LPDRAWITEMSTRUCT;
struct MEASUREITEMSTRUCT
{
     UINT CtlType;
     UINT CtlID;
     UINT itemID;
     UINT itemWidth;
     UINT высотаПункта;
     DWORD itemData;
}
alias MEASUREITEMSTRUCT* LPMEASUREITEMSTRUCT;
```

```
struct ANIMATIONINFO
{
     UINT cbSize;
     цел iMinAnimate;
}
struct APPBARDATA
{
     DWORD cbSize;
     HWND hWnd;
     UINT uCallbackMessage;
     UINT uEdge;
     RECT rc;
     LPARAM lParam; // message specific
}
alias APPBARDATA* PAPPBARDATA;
struct CLIENTCREATESTRUCT
{
     HANDLE hWindowMenu;
     UINT idFirstChild;
}
alias CLIENTCREATESTRUCT* LPCLIENTCREATESTRUCT;
struct MDICREATESTRUCTA
{
     LPCSTR szClass;
     LPCSTR szTitle;
     HANDLE hOwner;
     цел x;
     цел y;
     цел cx;
     цел cy;
     DWORD стиль;
     LPARAM lParam;
}
alias MDICREATESTRUCTA* LPMDICREATESTRUCTA;
struct DROPFILES
{
     DWORD pFiles;
     POINT pt;
     BOOL fNC;
     BOOL fWide;
}
alias DROPFILES* LPDROPFILES;
alias HANDLE HHOOK;
alias HANDLE HTHEME;
alias HANDLE HTREEITEM;
alias HANDLE HDROP;
HCURSOR CopyCursor(HCURSOR pcur)
{
     return cast(HCURSOR)CopyIcon(cast(HICON)pcur);
}
BOOL DrawIconEx(HDC hdc, цел xLeft, цел yTop, HICON hIcon, цел cxWidth,
цел cyWidth, UINT istepIfAniCur, HBRUSH hbrFlickerFreeDraw, UINT
diFlags);
BOOL DrawIcon(HDC hDC, цел X, цел Y, HICON hIcon);
BOOL SetWindowPos(HWND hWnd, HWND hWndInsertAfter, цел X, цел Y, цел cx,
цел cy, UINT uFlags);
HWND GetCapture();
HWND SetCapture(HWND hWnd);
BOOL ReleaseCapture();
HMENU GetMenu(HWND hWnd);
BOOL IsChild(HWND hWndParent, HWND hWnd);
BOOL IsWindow(HWND hWnd);
```

```
HWND CreateWindowExW(DWORD dwExStyle, LPCWSTR lpClassName, LPCWSTR
lpWindowName, DWORD dwStyle, цел x, цел y, цел nWidth, цел nHeight, HWND
hWndParent, HMENU hMenu, экз hInstance, LPVOID lpParam);
LRESULT SendMessageW(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam);
DWORD SetClassLongA(HWND hWnd, цел nIndex, LONG dwNewLong);
DWORD GetClassLongA(HWND hWnd, цел nIndex);
LONG SetWindowLongA(HWND hWnd, цел nIndex, LONG dwNewLong);
LONG GetWindowLongA(HWND hWnd, цел nIndex);
DWORD GetSysColor(цел nIndex);
BOOL EnableWindow(HWND hWnd, BOOL bEnable);
BOOL IsWindowEnabled(HWND hWnd);
COLORREF GetTextColor(HDC hdc);
//COLORREF SetTextColor(HDC hdc, COLORREF crColor);
HWND GetWindow(HWND hWnd, UINT uCmd);
DWORD GetWindowThreadProcessId(HWND hWnd, LPDWORD lpdwProcessId);
SHORT GetKeyState(цел nVirtKey);
SHORT GetAsyncKeyState(цел vKey);
HWND SetParent(HWND hWndChild, HWND hWndNewParent);
цел CombineRgn(HRGN hrgnDest, HRGN hrgnSrc1, HRGN hrgnSrc2, цел
fnCombineMode);
BOOL EnumWindows(WNDENUMPROC lpEnumFunc, LPARAM lParam);
BOOL EnumChildWindows(HWND hWndParent, WNDENUMPROC lpEnumFunc, LPARAM
lParam);
BOOL SetWindowTextA(HWND hWnd, LPCSTR lpString);
BOOL SetWindowTextW(HWND hWnd, LPCWSTR lpString);
цел GetWindowTextLengthA(HWND hWnd);
цел GetWindowTextLengthW(HWND hWnd);
цел GetWindowTextA(HWND hWnd, LPSTR lpString, цел nMaxCount);
цел GetWindowTextW(HWND hWnd, LPWSTR lpString, цел nMaxCount);
BOOL IsWindowVisible(HWND hWnd);
BOOL WaitMessage();
BOOL BringWindowToTop(HWND hWnd);
UINT RegisterWindowMessageA(LPCSTR lpString);
HWND GetParent(HWND hWnd);
HWND GetDesktopWindow();
HWND GetNextDlgTabItem(HWND hDlg, HWND hCtl, BOOL bPrevious);
HBRUSH CreateSolidBrush(COLORREF crColor);
HBRUSH CreateHatchBrush(цел fnStyle, COLORREF clrref);
проц InitCommonControls();
BOOL DestroyWindow(HWND hwnd);
ATOM RegisterClassExA(WNDCLASSEXA* lpwcx);
ATOM RegisterClassW(WNDCLASSW* lpWndClass);
HWND GetActiveWindow();
LRESULT DefDlgProcA(HWND hDlg, UINT Msg, WPARAM wParam, LPARAM lParam);
LRESULT DefDlgProcW(HWND hDlg, UINT Msg, WPARAM wParam, LPARAM lParam);
BOOL IsDialogMessageA(HWND hDlg, LPMSG lpMsg);
BOOL IsDialogMessageW(HWND hDlg, LPMSG lpMsg);
HBRUSH GetSysColorBrush(цел nIndex);
BOOL PostMessageA(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam);
UINT SetТаймер(HWND hWnd, UINT nIDEvent, UINT uElapse, TIMERPROC
lpТаймерFunc);
BOOL KillТаймер(HWND hWnd, UINT uIDEvent);
LPSTR GetCommandLineA();
LPWSTR GetCommandLineW();
BOOL SetCurrentDirectoryW(LPCWSTR lpPathName);
DWORD GetCurrentDirectoryW(DWORD nBufferLength, LPWSTR lpBuffer);
BOOL GetComputerNameA(LPSTR lpBuffer, LPDWORD nSize);
BOOL GetComputerNameW(LPWSTR lpBuffer, LPDWORD nSize);
BOOL GetVersionExA(LPOSVERSIONINFOA lpVersionInformation);
UINT GetSystemDirectoryA(LPSTR lpBuffer, UINT uSize);
UINT GetSystemDirectoryW(LPWSTR lpBuffer, UINT uSize);
BOOL GetUserNameA(LPSTR lpBuffer, LPDWORD nSize); // advapi32.lib
BOOL GetUserNameW(LPWSTR lpBuffer, LPDWORD nSize); // advapi32.lib
```

```
DWORD GetEnvironmentVariableA(LPCSTR lpName, LPSTR lpBuffer, DWORD
nSize);
DWORD GetEnvironmentVariableW(LPCWSTR lpName, LPWSTR lpBuffer, DWORD
nSize);
DWORD ExpandEnvironmentStringsW(LPCWSTR lpSrc, LPWSTR lpDst, DWORD
nSize);
DWORD GetLogicalDrives();
BOOL SetMenu(HWND hWnd, HMENU hMenu);
//BOOL win32.winuser.SetLayeredWindowAttributes(HWND hwnd, COLORREF
crKey, BYTE bAlpha, DWORD dwFlags);
BOOL SystemParametersInfoA(UINT uiAction, UINT uiParam, PVOID pvParam,
UINT fWinIni);
BOOL TrackMouseEvent(LPTRACKMOUSEEVENT lpEventTrack);
BOOL GetClassInfoA(экз hInstance, LPCSTR lpClassName, LPWNDCLASSA
lpWndClass);
BOOL GetClassInfoW(экз hInstance, LPCWSTR lpClassName, LPWNDCLASSW
lpWndClass);
LRESULT CallWindowProcA(WNDPROC lpPrevWndFunc, HWND hWnd, UINT Msg,
WPARAM wParam, LPARAM lParam);
LRESULT CallWindowProcW(WNDPROC lpPrevWndFunc, HWND hWnd, UINT Msg,
WPARAM wParam, LPARAM lParam);
BOOL OpenClipboard(HWND hWndNewOwner);
BOOL EmptyClipboard();
HGLOBAL GlobalAlloc(UINT uFlags, DWORD dwBytes);
BOOL CloseClipboard();
HANDLE SetClipboardData(UINT uFormat, HANDLE hMem);
HANDLE GetClipboardData(UINT uFormat);
//HGLOBAL GlobalFree(HGLOBAL hMem);
LPVOID GlobalLock(HGLOBAL hMem);
//BOOL GlobalUnlock(HGLOBAL hMem);
BOOL DrawFocusRect(HDC hDC, RECT* lprc);
LRESULT CallNextHookEx(HHOOK hhk, цел nCode, WPARAM wParam, LPARAM
lParam);
HHOOK SetWindowsHookExA(цел idHook, HOOKPROC lpfn, экз hMod, DWORD
dwThreadId);
BOOL UnhookWindowsHookEx(HHOOK hhk);
//цел GetSystemMetrics(цел nIndex);
BOOL DestroyMenu(HMENU hMenu);
BOOL SetMenuItemInfoA(HMENU hMenu, UINT uItem, BOOL fByPosition,
LPMENUITEMINFOA lpmii);
BOOL SetMenuItemInfoW(HMENU hMenu, UINT uItem, BOOL fByPosition,
LPMENUITEMINFOW lpmii);
BOOL InsertMenuItemA(HMENU hMenu, UINT uItem, BOOL fByPosition,
LPMENUITEMINFOA lpmii);
BOOL InsertMenuItemW(HMENU hMenu, UINT uItem, BOOL fByPosition,
LPMENUITEMINFOW lpmii);
BOOL RemoveMenu(HMENU hMenu, UINT uPosition, UINT uFlags );
UINT GetMenuItemID(HMENU hMenu, цел nPos);
BOOL DrawMenuBar(HWND hWnd);
HMENU CreatePopupMenu();
HMENU CreateMenu();
BOOL Shell_NotifyIconA(DWORD dwMessage, NOTIFYICONDATA* pnid);
LONG RegQueryValueExA(HKEY hKey, LPCSTR lpValueName, LPDWORD lpReserved,
LPDWORD lpType, LPBYTE lpData, LPDWORD lpcbData);
LONG RegQueryValueExW(HKEY hKey, LPCWSTR lpValueName, LPDWORD lpReserved,
LPDWORD lpType, LPBYTE lpData, LPDWORD lpcbData);
LONG RegConnectRegistryA(LPCSTR lpMachineName, HKEY hKey, PHKEY
phkResult);
UINT RegisterClipboardFormatA(LPCSTR lpszFormat);
UINT RegisterClipboardFormatW(LPCWSTR lpszFormat);
цел GetClipboardFormatNameA(UINT format, LPSTR lpszFormatName, цел
cchMaxCount);
цел GetClipboardFormatNameW(UINT format, LPWSTR lpszFormatName, цел
cchMaxCount);
```

```
DWORD GlobalSize(HGLOBAL hMem);
VOID ExitProcess(UINT uExitCode);
BOOL DrawAnimatedRects(HWND hwnd, цел idAni, RECT* lprcFrom, RECT*
lprcTo);
HWND FindWindowExA(HWND hwndParent, HWND hwndChildAfter, LPCSTR
lpszClass, LPCSTR lpszWindow);
UINT SHAppBarMessage(DWORD dwMessage, PAPPBARDATA pData);
BOOL SetPropA(HWND hWnd, LPCSTR lpString, HANDLE hData);
HANDLE GetPropA(HWND hWnd, LPCSTR lpString);
HANDLE RemovePropA(HWND hWnd, LPCSTR lpString);
DWORD CommDlgExtendedError();
LRESULT DefFrameProcA(HWND hWnd, HWND hWndMDIClient, UINT uMsg, WPARAM
wParam, LPARAM lParam);
LRESULT DefFrameProcW(HWND hWnd, HWND hWndMDIClient, UINT uMsg, WPARAM
wParam, LPARAM lParam);
LRESULT DefMDIChildProcA(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM
lParam);
LRESULT DefMDIChildProcW(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM
lParam);
VOID SetLastError(DWORD dwErrCode);
HWND CreateMDIWindowA(LPSTR lpClassName, LPSTR lpWindowName, DWORD
dwStyle, цел X, цел Y, цел nWidth, цел nHeight, HWND hWndParent, экз
hInstance, LPARAM lParam);
цел MulDiv(цел nNumber, цел nNumerator, цел nDenominator);
BOOL FillRgn(HDC hdc, HRGN hrgn, HBRUSH hbr);
COLORREF GetNearestColor(HDC hdc,COLORREF crColor);
цел DrawTextA(HDC hDC, LPCSTR lpString, цел nCount, LPRECT lpRect, UINT
uFormat);
цел DrawTextExA(HDC hdc, LPSTR lpchText, цел cchText, LPRECT lprc, UINT
dwDTFormat, LPDRAWTEXTPARAMS lpDTParams);
цел DrawTextExW(HDC hdc, LPWSTR lpchText, цел cchText, LPRECT lprc, UINT
dwDTFormat, LPDRAWTEXTPARAMS lpDTParams);
HANDLE LoadImageA(экз hinst, LPCSTR lpszName, UINT uType, цел cxDesired,
цел cyDesired, UINT fuLoad);
HANDLE LoadImageW(экз hinst, LPCWSTR lpszName, UINT uType, цел cxDesired,
цел cyDesired, UINT fuLoad);
HANDLE CopyImage(HANDLE hImage, UINT uType, цел cxDesired, цел cyDesired,
UINT fuFlags);
цел WSACancelAsyncRequest(HANDLE hAsyncTaskHandle);
HANDLE WSAAsyncGetHostByName(HWND hWnd, бцел wMsg, PCSTR имя, char* buf,
цел buflen);
HANDLE WSAAsyncGetHostByAddr(HWND hWnd, бцел wMsg, PCSTR addr, цел len,
цел type, char* buf, цел buflen);
BOOL ExtTextOutA(HDC hdc, цел X, цел Y, UINT fuOptions, RECT* lprc,
LPCSTR lpString, UINT cbCount, INT* lpDx);
BOOL Arc(HDC hdc, цел nLeftRect, цел nTopRect, цел nRightRect, цел
nBottomRect, цел nXStartArc, цел nYStartArc, цел nXEndArc, цел nYEndArc);
BOOL PolyBezier(HDC hdc, POINT* lppt,DWORD cPoints);
BOOL Ellipse(HDC hdc, цел nLeftRect, цел nTopRect, цел nRightRect, цел
nBottomRect);
BOOL Polygon(HDC hdc, POINT* lpPoints,цел nCount);
BOOL Rectangle(HDC hdc, цел nLeftRect, цел nTopRect, цел nRightRect, цел
nBottomRect);
BOOL GdiFlush();
LONG RegSetValueExW(HKEY hKey, LPCWSTR lpValueName, DWORD Reserved, DWORD
dwType, BYTE* lpData, DWORD cbData);
LONG RegCreateKeyExW(HKEY hKey, LPCWSTR lpSubKey, DWORD Reserved, LPWSTR
lpClass, DWORD dwOptions, REGSAM samDesired, LPSECURITY_ATTRIBUTES
lpSecurityAttributes, PHKEY phkResult, LPDWORD lpdwDisposition);
LONG RegOpenKeyExW(HKEY hKey, LPCWSTR lpSubKey, DWORD ulOptions, REGSAM
samDesired, PHKEY phkResult);
LONG RegDeleteKeyW(HKEY hKey, LPCWSTR lpSubKey);
```

```
LONG RegEnumKeyExW(HKEY hKey, DWORD dwIndex, LPWSTR lpName, LPDWORD
lpcbName, LPDWORD lpReserved, LPWSTR lpClass, LPDWORD lpcbClass,
PFILETIME lpftLastWriteTime);
LONG RegEnumValueW(HKEY hKey, DWORD dwIndex, LPTSTR lpValueName, LPDWORD
lpcbValueName, LPDWORD lpReserved, LPDWORD lpType, LPBYTE lpData, LPDWORD
lpcbData);
BOOL DrawFrameControl(HDC hdc, LPRECT lprc, UINT uType, UINT uState);
BOOL GetTextExtentPoint32A(HDC hdc, LPCSTR lpString, цел cbString, LPSIZE
lpSize);
BOOL GetTextExtentPoint32W(HDC hdc, LPCWSTR lpString, цел cbString,
LPSIZE lpSize);
экз ShellExecuteA(HWND hwnd, LPCSTR lpOperation, LPCSTR lpFile, LPCSTR
lpParameters, LPCSTR lpDirectory, INT nShowCmd);
HANDLE CreateActCtxW(PACTCTXW pActCtx);
BOOL ActivateActCtx(HANDLE hActCtx, ULONG** lpCookie);
UINT GetTempFileNameW(LPCWSTR lpPathName, LPCWSTR lpPrefixString, UINT
uUnique, LPWSTR lpTempFileName);
DWORD GetTempPathW(DWORD nBufferLength, LPWSTR lpBuffer);
VOID OutputDebugStringA(LPCSTR lpOutputString);
VOID DebugBreak();
BOOL BitBlt(HDC hdcDest, цел nXDest,цел nYDest, цел nWidth, цел nHeight,
HDC hdcSrc, цел nXSrc, цел nYSrc, DWORD dwRop);
BOOL GetIconInfo(HICON hIcon, PICONINFO piconinfo);
BOOL DestroyIcon(HICON hIcon);
BOOL DestroyCursor(HCURSOR hCursor);
LPITEMIDLIST SHBrowseForFolderA(LPBROWSEINFOA lpbi);
LPITEMIDLIST SHBrowseForFolderW(LPBROWSEINFOW lpbi);
HRESULT SHGetMalloc(LPMALLOC* ppMalloc);
BOOL SHGetPathFromIDListA(LPCITEMIDLIST pidl, LPSTR pszPath);
BOOL SHGetPathFromIDListW(LPCITEMIDLIST pidl, LPWSTR pszPath);
BOOL InitCommonControlsEx(LPINITCOMMONCONTROLSEX lpInitCtrls);
цел GetDlgCtrlID(HWND hwndCtl);
HWND GetDlgItem(HWND hDlg, цел nIDDlgItem);
BOOL ShowOwnedPopups(HWND hWnd, BOOL fShow);
UINT GetWindowsDirectoryA(LPSTR lpBuffer, UINT uSize);
UINT GetWindowsDirectoryW(LPWSTR lpBuffer, UINT uSize);
//HCURSOR CopyCursor(HCURSOR pcur);
экз LoadLibraryExA(LPCSTR lpLibFileName, HANDLE hFile, DWORD dwFlags);
экз LoadLibraryExW(LPCWSTR lpLibFileName, HANDLE hFile, DWORD dwFlags);
HICON CopyIcon(HICON hIcon);
BOOL ChooseColorA(LPCHOOSECOLORA lpcc);
UINT DragQueryFileA(HDROP hDrop, UINT iFile, LPSTR lpszFile, UINT cch);
UINT DragQueryFileW(HDROP hDrop, UINT iFile, LPWSTR lpszFile, UINT cch);
VOID DragFinish(HDROP hDrop);
BOOL DragQueryPoint(HDROP hDrop, LPPOINT lppt);
BOOL GrayStringA(HDC hDC, HBRUSH hBrush, GRAYSTRINGPROC lpOutputFunc,
LPARAM lpData, цел nCount, цел X, цел Y, цел nWidth, цел nHeight);
BOOL IsWindowUnicode(HWND hWnd);
BOOL ChooseFontA(LPCHOOSEFONTA lpcf);
BOOL ChooseFontW(LPCHOOSEFONTW lpcf);
HBITMAP CreateCompatibleBitmap(HDC hdc, цел nWidth, цел nHeight);
LONG DispatchMessageW(MSG* lpmsg);
BOOL PeekMessageW(LPMSG lpMsg, HWND hWnd, UINT wMsgFilterMin, UINT
wMsgFilterMax, UINT wRemoveMsg);
LRESULT DefWindowProcW(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM
lParam);
HWND GetNextDlgGroupItem(HWND hDlg, HWND hCtl, BOOL bPrevious);
HANDLE FindFirstChangeNotificationA(LPCSTR lpPathName, BOOL
bWatchSubtree, DWORD dwNotifyFilter);
HANDLE FindFirstChangeNotificationW(LPCWSTR lpPathName, BOOL
bWatchSubtree, DWORD dwNotifyFilter);
BOOL FindCloseChangeNotification(HANDLE hChangeHandle);
BOOL FindNextChangeNotification(HANDLE hChangeHandle);
```

```
     DWORD GetFullPathNameA(LPCSTR lpFileName, DWORD nBufferLength, LPSTR
     lpBuffer, LPSTR *lpFilePart);
     DWORD GetFullPathNameW(LPCWSTR lpFileName, DWORD nBufferLength, LPWSTR
     lpBuffer, LPWSTR *lpFilePart);
     SHORT VkKeyScanA(char ch);
     SHORT VkKeyScanW(wchar ch);
     HRSRC FindResourceExA(HMODULE hModule, LPCSTR lpType, LPCSTR lpName, WORD
     wLanguage);
     HRSRC FindResourceExW(HMODULE hModule, LPCWSTR lpType, LPCWSTR lpName,
     WORD wLanguage);
     HGLOBAL LoadResource(HMODULE hModule, HRSRC hResInfo);
     DWORD SizeofResource(HMODULE hModule, HRSRC hResInfo);
     BOOL EnableMenuItem(HMENU hMenu, UINT uIDEnableItem, UINT uEnable);
     BOOL IsMenu(HMENU hMenu);
     HMENU GetSystemMenu(HWND hWnd, BOOL bRevert);
     DWORD GetModuleFileNameW(HMODULE hModule, LPWSTR lpFilename, DWORD
     nSize);
     HBITMAP CreateBitmap(цел nWidth, цел nHeight, UINT cPlanes, UINT
     cBitsPerPel, VOID *lpvBits);
     BOOL SetBrushOrgEx(HDC hdc, цел nXOrg, цел nYOrg, LPPOINT lppt);
     BOOL PatBlt(HDC hdc, цел nXLeft, цел nYLeft, цел nWidth, цел nHeight,
     DWORD dwRop);
     HTHEME GetWindowTheme(HWND hWnd);
     THEMEAPI SetWindowTheme(HWND hwnd, LPCWSTR pszSubAppName, LPCWSTR
     pszSubIdList);
     цел SetScrollInfo(HWND hwnd, цел fnBar, LPSCROLLINFO lpsi, BOOL fRedraw);
     BOOL GetScrollInfo(HWND hwnd, цел fnBar, LPSCROLLINFO lpsi);
     BOOL DragDetect(HWND hwnd, POINT pt);
     HFONT CreateFontIndirectW(LOGFONTW *lplf);
     DWORD GetThemeAppProperties();
     BOOL IsAppThemed();
     HTHEME OpenThemeData(HWND hwnd, LPCWSTR pszClassList);
     HRESULT CloseThemeData(HTHEME hTheme);
     HRESULT GetThemeColor(HTHEME hTheme, цел iPartId, цел iStateId, цел
     iPropId, COLORREF *pColor);
     HIMAGELIST ImageList_Create(цел cx, цел cy, UINT флаги, цел cInitial, цел
     cGrow);
     BOOL ImageList_Destroy(HIMAGELIST himl);
     BOOL ImageList_Draw(HIMAGELIST himl, цел i, HDC hdcDst, цел x, цел y,
     UINT fStyle);
     BOOL ImageList_DrawEx(HIMAGELIST himl, цел i, HDC hdcDst, цел x, цел y,
     цел dx, цел dy, COLORREF rgbBk, COLORREF rgbFg, UINT fStyle);
     цел ImageList_Add(HIMAGELIST himl, HBITMAP hbmImage, HBITMAP hbmMask);
     цел ImageList_AddIcon(HIMAGELIST himl, HICON hicon);
     цел ImageList_AddMasked(HIMAGELIST himl, HBITMAP hbmImage, COLORREF
     crMask);
     BOOL ImageList_Remove(HIMAGELIST himl, цел i);
HMODULE GetModuleHandleA(LPCSTR lpModuleName);
////////////////////////////////////////////
/+

extern(C)
{
     extern IID IID_IPicture;
     version(REDEFINE_UUIDS)
     {
          // These are needed because uuid.lib is broken in DMC 8.46.
          IID _IID_IUnknown= { 0, 0, 0, [ 192, 0, 0, 0, 0, 0, 0, 70] };
          IID _IID_IDataObject = { 270, 0, 0, [192, 0, 0, 0, 0, 0, 0, 70 ] };
          IID _IID_IPicture = { 2079852928, 48946, 4122, [139, 187, 0, 170, 0,
          48, 12, 171] };
          IID _IID_ISequentialStream = { 208878128, 10780, 4558, [ 173, 229,
          0, 170, 0, 68, 119, 61 ] };
          IID _IID_IStream = { 12, 0, 0, [ 192, 0, 0, 0, 0, 0, 0, 70 ] };
```

```
            IID _IID_IDropTarget = { 290, 0, 0, [ 192, 0, 0, 0, 0, 0, 0, 70 ] };
            IID _IID_IDropSource = { 289, 0, 0, [ 192, 0, 0, 0, 0, 0, 0, 70 ] };
            IID _IID_IEnumFORMATETC = { 259, 0, 0, [ 192, 0, 0, 0, 0, 0, 0, 70 ]
            };
        }
        else
        {
            alias IID_IUnknown _IID_IUnknown;
            alias IID_IDataObject _IID_IDataObject;
            alias IID_IPicture _IID_IPicture;
            alias IID_ISequentialStream _IID_ISequentialStream;
            alias IID_IStream _IID_IStream;
            alias IID_IDropTarget _IID_IDropTarget;
            alias IID_IDropSource _IID_IDropSource;
            alias IID_IEnumFORMATETC _IID_IEnumFORMATETC;
        }
}
+/

//alias IID_IUnknown _IID_IUnknown;
//alias IID_IDataObject _IID_IDataObject;
//alias IID_IPicture _IID_IPicture;
         //alias IID_ISequentialStream _IID_ISequentialStream;
//alias IID_IStream _IID_IStream;
     //alias IID_IDropTarget _IID_IDropTarget;
     //alias IID_IDropSource _IID_IDropSource;
     //alias IID_IEnumFORMATETC _IID_IEnumFORMATETC;

extern(Windows):

interface ISequentialStream: IUnknown
{
     extern(Windows):
     HRESULT Read(ук pv, ULONG cb, ULONG* pcbRead);
     HRESULT Write(ук pv, ULONG cb, ULONG* pcbWritten);
}


/// STREAM_SEEK
enum: DWORD
{
     STREAM_SEEK_SET = 0,
     STREAM_SEEK_CUR = 1,
     STREAM_SEEK_END = 2,
}
alias DWORD STREAM_SEEK;


// TODO: implement the enum`s used here.
struct STATSTG
{
     LPWSTR pwcsName;
     DWORD тип;
     ULARGE_INTEGER cbSize;
     FILETIME mtime;
     FILETIME ctime;
     FILETIME atime;
     DWORD grfMode;
     DWORD grfLocksSupported;
     CLSID clsid;
     DWORD grfStateBits;
     DWORD reserved;
}
```

```
interface IStream: ISequentialStream
{
    extern(Windows):
    HRESULT Seek(LARGE_INTEGER dlibMove, DWORD dwOrigin, ULARGE_INTEGER*
    plibNewPosition);
    HRESULT SetSize(ULARGE_INTEGER libNewSize);
    HRESULT CopyTo(IStream pstm, ULARGE_INTEGER cb, ULARGE_INTEGER* pcbRead,
    ULARGE_INTEGER* pcbWritten);
    HRESULT Commit(DWORD grfCommitFlags);
    HRESULT Revert();
    HRESULT LockRegion(ULARGE_INTEGER libOffset, ULARGE_INTEGER cb, DWORD
    dwLockType);
    HRESULT UnlockRegion(ULARGE_INTEGER libOffset, ULARGE_INTEGER cb, DWORD
    dwLockType);
    HRESULT Stat(STATSTG* pstatstg, DWORD grfStatFlag);
    HRESULT Clone(IStream* ppstm);
}
alias IStream* LPSTREAM;


alias UINT OLE_HANDLE;

alias LONG OLE_XPOS_HIMETRIC;

alias LONG OLE_YPOS_HIMETRIC;

alias LONG OLE_XSIZE_HIMETRIC;

alias LONG OLE_YSIZE_HIMETRIC;


interface IPicture: IUnknown
{
    extern(Windows):
    HRESULT get_Handle(OLE_HANDLE* phandle);
    HRESULT get_hPal(OLE_HANDLE* phpal);
    HRESULT get_Type(short* ptype);
    HRESULT get_Width(OLE_XSIZE_HIMETRIC* pwidth);
    HRESULT get_Height(OLE_YSIZE_HIMETRIC* pheight);
    HRESULT Render(HDC hdc, цел x, цел y, цел cx, цел cy, OLE_XPOS_HIMETRIC
    xSrc, OLE_YPOS_HIMETRIC ySrc, OLE_XSIZE_HIMETRIC cxSrc,
    OLE_YSIZE_HIMETRIC cySrc, LPCRECT prcWBounds);
    HRESULT set_hPal(OLE_HANDLE hpal);
    HRESULT get_CurDC(HDC* phdcOut);
    HRESULT SelectPicture(HDC hdcIn, HDC* phdcOut, OLE_HANDLE* phbmpOut);
    HRESULT get_KeepOriginalFormat(BOOL* pfkeep);
    HRESULT put_KeepOriginalFormat(BOOL keep);
    HRESULT PictureChanged();
    HRESULT SaveAsFile(IStream pstream, BOOL fSaveMemCopy, LONG* pcbSize);
    HRESULT get_Attributes(DWORD* pdwAttr);
}

struct DVTARGETDEVICE
{
    DWORD tdSize;
    WORD tdDriverNameOffset;
    WORD tdDeviceNameOffset;
    WORD tdPortNameOffset;
    WORD tdExtDevmodeOffset;
    BYTE[1] tdData;
}
```

```
struct FORMATETC
{
     CLIPFORMAT cfFormat;
     DVTARGETDEVICE* ptd;
     DWORD dwAspect;
     LONG lindex;
     DWORD tymed;
}
alias FORMATETC* LPFORMATETC;


struct STATDATA
{
     FORMATETC formatetc;
     DWORD grfAdvf;
     IAdviseSink pAdvSink;
     DWORD dwConnection;
}


struct STGMEDIUM
{
     DWORD tymed;
     union //u
     {
          HBITMAP hBitmap;
          //HMETAFILEPICT hMetaFilePict;
          HENHMETAFILE hEnhMetaFile;
          HGLOBAL hGlobal;
          LPOLESTR lpszFileName;
          IStream pstm;
          //IStorage pstg;
     }
     IUnknown pUnkForRelease;
}
alias STGMEDIUM* LPSTGMEDIUM;


interface IDataObject: IUnknown
{
     extern(Windows):
     HRESULT GetData(FORMATETC* pFormatetc, STGMEDIUM* pmedium);
     HRESULT GetDataHere(FORMATETC* pFormatetc, STGMEDIUM* pmedium);
     HRESULT QueryGetData(FORMATETC* pFormatetc);
     HRESULT GetCanonicalFormatEtc(FORMATETC* pFormatetcIn, FORMATETC*
     pFormatetcOut);
     HRESULT SetData(FORMATETC* pFormatetc, STGMEDIUM* pmedium, BOOL
     fRelease);
     HRESULT EnumFormatEtc(DWORD dwDirection, IEnumFORMATETC*
     ppenumFormatetc);
     HRESULT DAdvise(FORMATETC* pFormatetc, DWORD advf, IAdviseSink pAdvSink,
     DWORD* pdwConnection);
     HRESULT DUnadvise(DWORD dwConnection);
     HRESULT EnumDAdvise(IEnumSTATDATA* ppenumAdvise);
}


interface IDropSource: IUnknown
{
     extern(Windows):
     HRESULT QueryContinueDrag(BOOL fEscapePressed, DWORD grfKeyState);
     HRESULT GiveFeedback(DWORD dwEffect);
}
```

```
interface IDropTarget: IUnknown
{
    extern(Windows):
    HRESULT DragEnter(IDataObject pDataObject, DWORD grfKeyState, POINTL pt,
    DWORD* pdwEffect);
    HRESULT DragOver(DWORD grfKeyState, POINTL pt, DWORD* pdwEffect);
    HRESULT DragLeave();
    HRESULT Drop(IDataObject pDataObject, DWORD grfKeyState, POINTL pt,
    DWORD* pdwEffect);
}


interface IEnumFORMATETC: IUnknown
{
    extern(Windows):
    HRESULT Next(ULONG celt, FORMATETC* rgelt, ULONG* pceltFetched);
    HRESULT Skip(ULONG celt);
    HRESULT Reset();
    HRESULT Clone(IEnumFORMATETC* ppenum);
}


interface IEnumSTATDATA: IUnknown
{
    extern(Windows):
    HRESULT Next(ULONG celt, STATDATA* rgelt, ULONG* pceltFetched);
    HRESULT Skip(ULONG celt);
    HRESULT Reset();
    HRESULT Clone(IEnumSTATDATA* ppenum);
}


interface IAdviseSink: IUnknown
{
    // TODO: finish.
}


interface IMalloc: IUnknown
{
    extern(Windows):
    ук Alloc(ULONG cb);
    ук Realloc(ук pv, ULONG cb);
    проц Free(ук pv);
    ULONG GetSize(ук pv);
    цел DidAlloc(ук pv);
    проц HeapMinimize();
}

// Since an interface is a pointer..
alias IMalloc PMALLOC;
alias IMalloc LPMALLOC;


LONG MAP_LOGHIM_TO_PIX(LONG x, LONG logpixels)
{
    return MulDiv(logpixels, x, 2540);
}


enum: DWORD
{
    DVASPECT_CONTENT = 1,
```

```
     DVASPECT_THUMBNAIL = 2,
     DVASPECT_ICON = 4,
     DVASPECT_DOCPRINT = 8,
}
alias DWORD DVASPECT;


enum: DWORD
{
     TYMED_HGLOBAL = 1,
     TYMED_FILE = 2,
     TYMED_ISTREAM = 4,
     TYMED_ISTORAGE = 8,
     TYMED_GDI = 16,
     TYMED_MFPICT = 32,
     TYMED_ENHMF = 64,
     TYMED_NULL = 0
}
alias DWORD TYMED;


enum
{
     DATADIR_GET = 1,
}


enum: HRESULT
{
     DRAGDROP_S_DROP = 0x00040100,
     DRAGDROP_S_CANCEL = 0x00040101,
     DRAGDROP_S_USEDEFAULTCURSORS = 0x00040102,
     V_E_LINDEX = cast(HRESULT)0x80040068,
     STG_E_MEDIUMFULL = cast(HRESULT)0x80030070,
     STG_E_INVALIDFUNCTION = cast(HRESULT)0x80030001,
     DV_E_TYMED = cast(HRESULT)0x80040069,
     DV_E_DVASPECT = cast(HRESULT)0x8004006B,
     DV_E_FORMATETC = cast(HRESULT)0x80040064,
     DV_E_LINDEX = cast(HRESULT)0x80040068,
     DRAGDROP_E_ALREADYREGISTERED = cast(HRESULT)0x80040101,
}


alias HRESULT WINOLEAPI;


WINOLEAPI OleInitialize(LPVOID pvReserved);
WINOLEAPI DoDragDrop(IDataObject pDataObject, IDropSource pDropSource, DWORD
dwOKEffect, DWORD* pdwEffect);
WINOLEAPI RegisterDragDrop(УОК hwnd, IDropTarget pDropTarget);
WINOLEAPI RevokeDragDrop(УОК hwnd);
WINOLEAPI OleGetClipboard(IDataObject* ppDataObj);
WINOLEAPI OleSetClipboard(IDataObject pDataObj);
WINOLEAPI OleFlushClipboard();
WINOLEAPI CreateStreamOnHGlobal(HGLOBAL hGlobal, BOOL fDeleteOnRelease,
LPSTREAM ppstm);
WINOLEAPI OleLoadPicture(IStream pStream, LONG lSize, BOOL fRunmode, IID*
riid, проц** ppv);


enum : DWORD
{
CP_ACP = (0),
CP_MACCP = (2),
```

```
CP_OEMCP = (1),
CP_UTF8 = 65001
}
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
18:03

```
module com;
public import tpl.com, sys.com, sys.COM.all;

бул КОМАктивен;

extern(C) extern бул комАктивен(бул данет = нет);

static this()
{
if(!комАктивен)
     {
          откройКОМ();
          комАктивен(да);
          КОМАктивен = да;
     }
}

static ~this()
{
if(!комАктивен)
     {
     закройКОМ();
     комАктивен(нет);
     КОМАктивен = нет;
     }
}
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
18:05

```
module dinrus;

/** Динамическая версия русского диалекта языка программирования Ди.
**********************************************************
ПРОЕКТ "ДИНРУС"

Идея данного проекта заключается в создании универсального
языка программирования на основе языка D, разрабатываемого
американской компанией Digital Mars.

Автором языка D является Уолтер Брайт (Walter Bright), известный
как разработчик серии компиляторов языка C для компании Simantec (SC).
```

Опыт Брайта по созданию компиляторов выразился в намерении
создать новый, более мощный язык системного программирования.

Брайт создал две версии компилятора DMD.

Проект Динрус рассчитан на использование первой из них (v 1.065).

Динрус основан на собственной переработанной версии рантайма,
в которой основные элементы библиотеки существенно отличаются, и
несовместимы с другими версиями.

Задача Динрус – обеспечить возможность одновременного
программирования как на английском, так и на русском языках.

Русский язык является основным приоритетом.

Дальнейшая переработка системных библиотек позволит обеспечить
быстрое и доступное системное программирование на родном
(русском) языке.

Автор и Разработчик : Виталий Кулич
**************************************/
//==
/*************************
Конфигурационный файл основной библиотеки языка программирования Динрус.
Целевое назначение: сообщать системе о том, где располагаются перечни
констант, структур, функций и других операционных элементов для той или
иной системы, в зависимости от вида трансляции и вида системных устройств
и самой ОС(если ОС = Windows, ОС = Linux и т.п.)

Это единственный файл, который импортируется модулем object.
Так как указанный модуль содержит важные сведения о типах языка и загружается
компилятором ранее других, данные системные настройки становятся общими для
всех модулей языка и поэтому отпадает необходимость переопределения констант
или структур в явном виде, либо в затруднительном поиске таковых по всем
модулям.

Кроме того, этим обеспечивается главная цель: единство языковых определений и
отсутствие захламляющих языковую среду переопределений одних и тех же
элементов.
Таким образом, язык становится более прогрессивным в плане чистоты, качества,
единства и скорости своего развития.

Так как язык Динрус в своём будущем нацелен на портируемость и компактность,
в нём предусмотрена конфигурационная версиональность, обеспечиваемая
статическими
если(static if) и переключателями версии (version). Главные определения и
переключатели
должны всегда располагаться в этом модуле.

Пакеты для той или иной системы могут распространятся отдельно или
добавляться в
последующем; они должны так же строго структурироваться и содержать такие же
основные модули, как в данном случае, предназначенном для ОС i386 (Windows).

Конфигурированные здесь настройки могут включаться при импорте тех или иных
модулей,
например, когда импортируется модуль sys.com, он устанавливает version = OMO.
Так как он импортирует dinrus, то с импортом модуля sys.com или с установкой
версии OMO
автоматически происходит активация и подключение (инициализация) к системным
библиотекам, и программисту более не требуется об этом заботиться вообще.

```
Одним словом, вам остаётся лишь выбрать здесь версию и написать version = X в
вашем файле,
ознакомиться с тем, что предоставляется данной версией, чтобы ... продолжить
развитие данного языка или написать соответствующий модуль для своей
собственной
программы.
************************/
version = Dinrus;

public import object,
gc,
//base,
     //В этом модуле находятся важнейшие настройки для языка Динрус:
     //например, определения основных глобальных типов или
     //список импортируемых языковой средой функций и классов.
sync, thread, stdrus, tpl.all, runtime, exception, global, win;
/+
Модуль win содержит открытый доступ к модулям:

     sys.DConsts,
     //В этом модуле: константы (их перчни) для API ОС.
     sys.uuid,
     sys.DIfaces,
     //Интерфейсы API данной операционной системы.
     sys.DStructs,
     //Здесь искать: структуры API для данной системы.
     sys.DFuncs;
     //Здесь: функции и процедуры, предоставляемые API ОС.
     Кроме того, в нём находятся основные рычаги управления консольным вводом-
     выводом.
+/

/*****************************************************************
Модуль cidrus содержит руссифицированный интерфейс к функциям языка Си,
которые переработаны в модуле stdrus и других под Динрус.
При использовании этого модуля появляются накладки.
*****************************************************************/
version(PlusC)
{
public import cidrus;
}

version(COM)//ОБЩАЯ МОДЕЛЬ ОБЪЕКТА (COM)
{
     public import com;
}
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>

19 декабря 2016 г.
18:08

```
module win;
public import sys.DConsts, sys.DIfaces, sys.DStructs, sys.DFuncs, sys.uuid;
/+
public static
{
ук КОНСВВОД;
```

```
ук КОНСВЫВОД;
ук КОНСОШ;
//бцел ИДПРОЦЕССА;
//ук УКНАПРОЦЕСС;
//ук УКНАНИТЬ;
}

static this()
{
//ИДПРОЦЕССА = GetCurrentProcessId();
//УКНАПРОЦЕСС = cast(ук)
OpenProcess(0x000F0000|0x00100000|0x0FFF,false,ИДПРОЦЕССА);
//УКНАНИТЬ = GetCurrentThread();
              КОНСВВОД = ДайСтдДескр(ПСтд.Ввод);
              //КОНСВЫВОД = ДайСтдДескр(cast(ПСтд) 0xfffffff5);
              КОНСВЫВОД = ДайСтдДескр(ПСтд.Вывод);
              КОНСОШ = ДайСтдДескр(ПСтд.Ошибка);
}
+/
extern(C)
{
     проц перейдиНаТочкуКонсоли( цел аХ, цел аУ);
     проц установиАтрыКонсоли(ПТекстКонсоли атр);
     цел гдеИксКонсоли();
     цел гдеИгрекКонсоли();
     ПТекстКонсоли дайАтрыКонсоли();
     проц сбросьЦветКонсоли();
     фук консВход();
     фук консВыход();
     фук консОш();

     struct Console
     {
              alias newline opCall;
              alias emit opCall;
              /// emit a utf8 string to the console
              Console emit(char[] s);
              Console err(char[] s);
              /// emit an unsigned integer to the console
              Console emit(ulong i);
              /// emit a newline to the console
     Console newline();
              alias newline нс;
     }
}

extern(D):

     проц скажи(ткст ткт);
     проц скажи(бдол ткт);
     проц скажинс(ткст ткт);
     проц скажинс(бдол ткт);
     проц ошибнс(ткст ткт);
     проц нс();
     проц таб();
```

Источник <file:///D:\dinrus\help\ModStructDinrus.docx>