```
import Darwin.AssertMacros
import Darwin.C
import Darwin.ConditionalMacros
import Darwin.MacTypes
import Darwin.Mach
import Darwin.POSIX
import Darwin.aliasdb
import Darwin.ar
import Darwin.architecture
import Darwin.bank
import Darwin.bitstring
import Darwin.block
import Darwin.bootparams
import Darwin.bsm
import Darwin.crt_externs
import Darwin.device
import Darwin.execinfo
import Darwin.fstab
import Darwin.fts
import Darwin.getopt
import Darwin.hfs
import Darwin.libc
import Darwin.libkern
import Darwin.libproc
import Darwin.machine
import Darwin.malloc
import Darwin.membership
import Darwin.ncurses
import Darwin.net
import Darwin.netinet
import Darwin.netinet6
import Darwin.ntsid
import Darwin.os
```

```swift
import Darwin.paths
import Darwin.printerdb
import Darwin.sys
import Darwin.sysdir
import Darwin.sysexits
import Darwin.utmp
import Darwin.uuid
import _Builtin_float
import _errno
import _math
import _signal
import _stdio
import _time
import sys_time
import unistd

/// The `Boolean` type declared in
MacTypes.h and used throughout Core
/// Foundation.
///
/// The C type is a typedef for `unsigned
char`.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
@frozen public struct DarwinBoolean :
ExpressibleByBooleanLiteral, Sendable {

    public init(_ value: Bool)

    /// The value of `self`, expressed as
a `Bool`.
    public var boolValue: Bool { get }
    /// Create an instance initialized to
`value`.
    public init(booleanLiteral value:
Bool)

    /// A type that represents a Boolean
literal, such as `Bool`.
    @available(iOS 7.0, tvOS 9.0, watchOS
2.0, visionOS 1.0, macOS 10.9, bridgeOS
7.0, *)
    public typealias BooleanLiteralType =
Bool
}

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
extension DarwinBoolean :
CustomReflectable {

    /// Returns a mirror that reflects
`self`.
    public var customMirror: Mirror { get
}
}

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
extension DarwinBoolean :
CustomStringConvertible {
    /// A textual representation of
`self`.
    public var description: String {
get }
}

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
extension DarwinBoolean : Equatable {

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (lhs:
DarwinBoolean, rhs: DarwinBoolean) ->
Bool
}

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
extension DarwinBoolean : BitwiseCopyable
{
}

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
nonisolated(unsafe) public let
MAP_FAILED: UnsafeMutableRawPointer!

/// Enumeration describing Mach error
codes.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
@objc public enum MachErrorCode : Int32,
Sendable {

    case success = 0

    /// Specified address is not
currently valid.
    case invalidAddress = 1

    /// Specified memory is valid, but
does not permit the required
    /// forms of access.
    case protectionFailure = 2

    /// The address range specified is
already in use, or no address
    /// range of the size specified could
be found.
    case noSpace = 3
    /// The function requested was not
applicable to this type of
    /// argument, or an argument is
invalid.
    case invalidArgument = 4

    /// The function could not be
performed.  A catch-all.
    case failure = 5

    /// A system resource could not be
allocated to fulfill this
    /// request.  This failure may not be
permanent.
    case resourceShortage = 6

    /// The task in question does not
hold receive rights for the port
    /// argument.
    case notReceiver = 7

    /// Bogus access restriction.
    case noAccess = 8

    /// During a page fault, the target
address refers to a memory
    /// object that has been destroyed.
This failure is permanent.
    case memoryFailure = 9

    /// During a page fault, the memory
object indicated that the data
    /// could not be returned.  This
```

```
failure may be temporary; future
    /// attempts to access this same data
may succeed, as defined by the
    /// memory object.
    case memoryError = 10

    /// The receive right is already a
member of the portset.
    case alreadyInSet = 11

    /// The receive right is not a member
of a port set.
    case notInSet = 12

    /// The name already denotes a right
in the task.
    case nameExists = 13

    /// The operation was aborted.  Ipc
code will catch this and reflect
    /// it as a message error.
    case aborted = 14

    /// The name doesn't denote a right
in the task.
    case invalidName = 15

    /// Target task isn't an active task.
    case invalidTask = 16

    /// The name denotes a right, but not
an appropriate right.
    case invalidRight = 17
    /// A blatant range error.
    case invalidValue = 18

    /// Operation would overflow limit on
user-references.
    case userReferencesOverflow = 19

    /// The supplied (port) capability is
improper.
    case invalidCapability = 20

    /// The task already has send or
receive rights for the port under
    /// another name.
    case rightExists = 21

    /// Target host isn't actually a
host.
    case invalidHost = 22

    /// An attempt was made to supply
"precious" data for memory that is
    /// already present in a memory
object.
    case memoryPresent = 23

    /// A page was requested of a memory
manager via
    /// memory_object_data_request for an
object using a
    /// MEMORY_OBJECT_COPY_CALL strategy,
with the VM_PROT_WANTS_COPY
```

```
    /// flag being used to specify that
the page desired is for a copy
    /// of the object, and the memory
manager has detected the page was
    /// pushed into a copy of the object
while the kernel was walking
    /// the shadow chain from the copy to
the object. This error code is
    /// delivered via
memory_object_data_error and is handled
by the
    /// kernel (it forces the kernel to
restart the fault). It will not
    /// be seen by users.
    case memoryDataMoved = 24

    /// A strategic copy was attempted of
an object upon which a quicker
    /// copy is now possible.  The caller
should retry the copy using
    /// vm_object_copy_quickly. This
error code is seen only by the
    /// kernel.
    case memoryRestartCopy = 25

    /// An argument applied to assert
processor set privilege was not a
    /// processor set control port.
    case invalidProcessorSet = 26

    /// The specified scheduling
attributes exceed the thread's limits.
    case policyLimit = 27
    /// The specified scheduling policy
is not currently enabled for the
    /// processor set.
    case invalidPolicy = 28

    /// The external memory manager
failed to initialize the memory object.
    case invalidObject = 29

    /// A thread is attempting to wait
for an event for which there is
    /// already a waiting thread.
    case alreadyWaiting = 30

    /// An attempt was made to destroy
the default processor set.
    case defaultSet = 31

    /// An attempt was made to fetch an
exception port that is
    /// protected, or to abort a thread
while processing a protected
    /// exception.
    case exceptionProtected = 32

    /// A ledger was required but not
supplied.
    case invalidLedger = 33

    /// The port was not a memory cache
control port.
    case invalidMemoryControl = 34
    /// An argument supplied to assert
security privilege was not a host
    /// security port.
    case invalidSecurity = 35

    /// thread_depress_abort was called
on a thread which was not
    /// currently depressed.
    case notDepressed = 36

    /// Object has been terminated and is
no longer available.
    case terminated = 37

    /// Lock set has been destroyed and
is no longer available.
    case lockSetDestroyed = 38

    /// The thread holding the lock
terminated before releasing the lock.
    case lockUnstable = 39

    /// The lock is already owned by
another thread.
    case lockOwned = 40

    /// The lock is already owned by the
calling thread.
    case lockOwnedSelf = 41

    /// Semaphore has been destroyed and
is no longer available.
```

```
    case semaphoreDestroyed = 42

    /// Return from RPC indicating the
target server was terminated
    /// before it successfully replied.
    case rpcServerTerminated = 43

    /// Terminate an orphaned activation.
    case rpcTerminateOrphan = 44

    /// Allow an orphaned activation to
continue executing.
    case rpcContinueOrphan = 45

    /// Empty thread activation (No
thread linked to it).
    case notSupported = 46

    /// Remote node down or inaccessible.
    case nodeDown = 47

    /// A signalled thread was not
actually waiting.
    case notWaiting = 48

    /// Some thread-oriented operation
(semaphore_wait) timed out.
    case operationTimedOut = 49

    /// During a page fault, indicates
that the page was rejected as a
    /// result of a signature check.
    case codesignError = 50
    /// The requested property cannot be
changed at this time.
    case policyStatic = 51

    /// Creates a new instance with the
specified raw value.
    ///
    /// If there is no value of the type
that corresponds with the specified raw
    /// value, this initializer returns
`nil`. For example:
    ///
    ///     enum PaperSize: String {
    ///         case A4, A5, Letter,
Legal
    ///     }
    ///
    ///     print(PaperSize(rawValue:
"Legal"))
    ///     // Prints
"Optional("PaperSize.Legal")"
    ///
    ///     print(PaperSize(rawValue:
"Tabloid"))
    ///     // Prints "nil"
    ///
    /// - Parameter rawValue: The raw
value to use for the new instance.
    public init?(rawValue: Int32)

    /// The raw type that can be used to
represent all values of the conforming
```

```
    /// type.
    ///
    /// Every distinct value of the
conforming type has a corresponding
unique
    /// value of the `RawValue` type, but
there may be values of the `RawValue`
    /// type that don't have a
corresponding value of the conforming
type.
    @available(iOS 7.0, tvOS 9.0, watchOS
2.0, visionOS 1.0, macOS 10.9, bridgeOS
7.0, *)
    public typealias RawValue = Int32

    /// The corresponding value of the
raw type.
    ///
    /// A new instance initialized with
`rawValue` will be equivalent to this
    /// instance. For example:
    ///
    ///     enum PaperSize: String {
    ///         case A4, A5, Letter,
Legal
    ///     }
    ///
    ///     let selectedSize =
PaperSize.Letter
    ///     print(selectedSize.rawValue)
    ///     // Prints "Letter"
    ///
    ///     print(selectedSize ==
PaperSize(rawValue:
selectedSize.rawValue)!)
    ///     // Prints "true"
    public var rawValue: Int32 { get }
}

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
extension MachErrorCode : Equatable {
}

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
extension MachErrorCode : Hashable {
}

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
extension MachErrorCode :
RawRepresentable {
}

/// The value returned by `sem_open()` in
the case of failure.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var SEM_FAILED: Semaphore? { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IEXEC: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IFBLK: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IFCHR: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IFDIR: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IFIFO: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IFLNK: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IFMT: mode_t { get }
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IFREG: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IFSOCK: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IFWHT: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IREAD: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IRGRP: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IROTH: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IRUSR: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IRWXG: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IRWXO: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IRWXU: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_ISGID: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_ISTXT: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_ISUID: mode_t { get }
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_ISVTX: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IWGRP: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IWOTH: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IWRITE: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IWUSR: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IXGRP: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public var S_IXOTH: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var S_IXUSR: mode_t { get }

public typealias Semaphore =
UnsafeMutablePointer<sem_t>

/// Clear break bit.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCCBRK: UInt { get }

/// Clear data terminal ready.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCCDTR: UInt { get }

/// Become virtual console.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCCONS: UInt { get }

/// Enable/get timestamp of last DCd
rise.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public var TIOCDCDTIMESTAMP: UInt { get }

/// Wait till output drained.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCDRAIN: UInt { get }

/// Download microcode to DSI Softmodem.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCDSIMICROCODE: UInt { get }

/// Set exclusive use of tty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCEXCL: UInt { get }

/// Pty: external processing.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCEXT: UInt { get }

/// Flush buffers.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCFLUSH: UInt { get }

/// Get ttywait timeout.
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCGDRAINWAIT: UInt { get }

/// Get termios struct.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCGETA: UInt { get }

/// Get special characters.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCGETC: UInt { get }

/// Get line discipline.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCGETD: UInt { get }

/// Get parameters -- gtty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCGETP: UInt { get }

/// Get local special chars.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public var TIOCGLTC: UInt { get }

/// Get pgrp of tty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCGPGRP: UInt { get }

/// Get window size.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCGWINSZ: UInt { get }

/// Hang up on last close.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCHPCL: UInt { get }

/// Internal input VSTOP.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCIXOFF: UInt { get }

/// Internal input VSTART.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCIXON: UInt { get }

/// Bic local mode bits.
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCLBIC: UInt { get }

/// Bis local mode bits.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCLBIS: UInt { get }

/// Get local modes.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCLGET: UInt { get }

/// Set entire local mode word.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCLSET: UInt { get }

/// Bic modem bits.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCMBIC: UInt { get }

/// Bis modem bits.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public var TIOCMBIS: UInt { get }

/// Modem: get wait on close.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCMGDTRWAIT: UInt { get }

/// Get all modem bits.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCMGET: UInt { get }

/// Get modem control state.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCMODG: UInt { get }

/// Set modem control state.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCMODS: UInt { get }

/// Modem: set wait on close.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCMSDTRWAIT: UInt { get }

/// Set all modem bits.
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCMSET: UInt { get }

/// Void tty association.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCNOTTY: UInt { get }

/// Reset exclusive use of tty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCNXCL: UInt { get }

/// Output queue size.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCOUTQ: UInt { get }

/// Pty: set/clear packet mode.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCPKT: UInt { get }

/// Ptsname(3).
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```swift
public var TIOCPTYGNAME: UInt { get }

/// Grantpt(3).
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCPTYGRANT: UInt { get }

/// Unlockpt(3).
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCPTYUNLK: UInt { get }

/// Remote input editing.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCREMOTE: UInt { get }

/// Set break bit.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSBRK: UInt { get }

/// 4.2 compatibility.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSCONS: UInt { get }

/// Become controlling tty.
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSCTTY: UInt { get }

/// Set ttywait timeout.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSDRAINWAIT: UInt { get }

/// Set data terminal ready.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSDTR: UInt { get }

/// Set termios struct.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSETA: UInt { get }

/// Drn out, fls in, set.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSETAF: UInt { get }

/// Drain output, set.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public var TIOCSETAW: UInt { get }

/// Set special characters.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSETC: UInt { get }

/// Set line discipline.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSETD: UInt { get }

/// As above, but no flushtty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSETN: UInt { get }

/// Set parameters -- stty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSETP: UInt { get }

/// Pty: generate signal.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSIG: UInt { get }

/// Set local special chars.
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSLTC: UInt { get }

/// Set pgrp of tty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSPGRP: UInt { get }

/// Start output, like `^Q`.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSTART: UInt { get }

/// Simulate `^T` status message.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSTAT: UInt { get }

/// Simulate terminal input.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSTI: UInt { get }

/// Stop output, like `^S`.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public var TIOCSTOP: UInt { get }

/// Set window size.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCSWINSZ: UInt { get }

/// Enable/get timestamp of last input
event.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCTIMESTAMP: UInt { get }

/// Pty: set/clr usr cntl mode.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var TIOCUCNTL: UInt { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var environ:
UnsafeMutablePointer<UnsafeMutablePointer
<CChar>?> { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func fcntl(_ fd: Int32, _ cmd:
Int32) -> Int32
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func fcntl(_ fd: Int32, _ cmd:
Int32, _ value: Int32) -> Int32

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func fcntl(_ fd: Int32, _ cmd:
Int32, _ ptr: UnsafeMutableRawPointer) ->
Int32

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func ioctl(_ fd: CInt, _ request:
UInt, _ value: CInt) -> CInt

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func ioctl(_ fd: CInt, _ request:
UInt, _ ptr: UnsafeMutableRawPointer) ->
CInt

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func ioctl(_ fd: CInt, _ request:
UInt) -> CInt
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public var noErr: OSStatus { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func open(_ path:
UnsafePointer<CChar>, _ oflag: Int32) ->
Int32

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func open(_ path:
UnsafePointer<CChar>, _ oflag: Int32, _
mode: mode_t) -> Int32

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func openat(_ fd: Int32, _ path:
UnsafePointer<CChar>, _ oflag: Int32) ->
Int32

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func openat(_ fd: Int32, _ path:
UnsafePointer<CChar>, _ oflag: Int32, _
mode: mode_t) -> Int32
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func sem_open(_ name:
UnsafePointer<CChar>, _ oflag: Int32) ->
Semaphore?

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public func sem_open(_ name:
UnsafePointer<CChar>, _ oflag: Int32, _
mode: mode_t, _ value: CUnsignedInt) ->
Semaphore?

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
extension extern_proc {

    public var p_starttime: timeval
}

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let environ:
UnsafeMutablePointer<UnsafeMutablePointer
<CChar>?> { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public let noErr: OSStatus { get }

/// The value returned by `sem_open()` in
the case of failure.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let SEM_FAILED: Semaphore? { get }

/// Hang up on last close.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCHPCL: UInt { get }

/// Get parameters -- gtty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCGETP: UInt { get }

/// Set parameters -- stty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSETP: UInt { get }

/// As above, but no flushtty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSETN: UInt { get }
```

```swift
/// Set special characters.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSETC: UInt { get }

/// Get special characters.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCGETC: UInt { get }

/// Bis local mode bits.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCLBIS: UInt { get }

/// Bic local mode bits.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCLBIC: UInt { get }

/// Set entire local mode word.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCLSET: UInt { get }

/// Get local modes.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
```

```
1.0, *)
public let TIOCLGET: UInt { get }

/// Set local special chars.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSLTC: UInt { get }

/// Get local special chars.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCGLTC: UInt { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
nonisolated(unsafe) public var
MAP_FAILED: UnsafeMutableRawPointer!

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IFMT: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IFIFO: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
```

```
1.0, *)
public let S_IFCHR: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IFDIR: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IFBLK: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IFREG: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IFLNK: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IFSOCK: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IFWHT: mode_t { get }
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IRWXU: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IRUSR: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IWUSR: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IXUSR: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IRWXG: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IRGRP: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public let S_IWGRP: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IXGRP: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IRWXO: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IROTH: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IWOTH: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IXOTH: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_ISUID: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_ISGID: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_ISVTX: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_ISTXT: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IREAD: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IWRITE: mode_t { get }

@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let S_IEXEC: mode_t { get }

/// Get modem control state.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public let TIOCMODG: UInt { get }

/// Set modem control state.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCMODS: UInt { get }

/// Set exclusive use of tty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCEXCL: UInt { get }

/// Reset exclusive use of tty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCNXCL: UInt { get }

/// Flush buffers.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCFLUSH: UInt { get }

/// Get termios struct.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCGETA: UInt { get }

/// Set termios struct.
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSETA: UInt { get }

/// Drain output, set.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSETAW: UInt { get }

/// Drn out, fls in, set.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSETAF: UInt { get }

/// Get line discipline.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCGETD: UInt { get }

/// Set line discipline.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSETD: UInt { get }

/// Internal input VSTART.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public let TIOCIXON: UInt { get }

/// Internal input VSTOP.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCIXOFF: UInt { get }

/// Set break bit.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSBRK: UInt { get }

/// Clear break bit.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCCBRK: UInt { get }

/// Set data terminal ready.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSDTR: UInt { get }

/// Clear data terminal ready.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCCDTR: UInt { get }

/// Get pgrp of tty.
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCGPGRP: UInt { get }

/// Set pgrp of tty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSPGRP: UInt { get }

/// Output queue size.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCOUTQ: UInt { get }

/// Simulate terminal input.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSTI: UInt { get }

/// Void tty association.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCNOTTY: UInt { get }

/// Pty: set/clear packet mode.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public let TIOCPKT: UInt { get }

/// Stop output, like `^S`.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSTOP: UInt { get }

/// Start output, like `^Q`.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSTART: UInt { get }

/// Set all modem bits.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCMSET: UInt { get }

/// Bis modem bits.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCMBIS: UInt { get }

/// Bic modem bits.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCMBIC: UInt { get }

/// Get all modem bits.
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCMGET: UInt { get }

/// Remote input editing.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCREMOTE: UInt { get }

/// Get window size.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCGWINSZ: UInt { get }

/// Set window size.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSWINSZ: UInt { get }

/// Pty: set/clr usr cntl mode.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCUCNTL: UInt { get }

/// Simulate `^T` status message.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
```

```
public let TIOCSTAT: UInt { get }

/// 4.2 compatibility.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSCONS: UInt { get }

/// Become virtual console.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCCONS: UInt { get }

/// Become controlling tty.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSCTTY: UInt { get }

/// Pty: external processing.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCEXT: UInt { get }

/// Pty: generate signal.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSIG: UInt { get }

/// Wait till output drained.
```

```
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCDRAIN: UInt { get }

/// Modem: set wait on close.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCMSDTRWAIT: UInt { get }

/// Modem: get wait on close.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCMGDTRWAIT: UInt { get }

/// Enable/get timestamp of last input
event.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCTIMESTAMP: UInt { get }

/// Enable/get timestamp of last DCd
rise.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCDCDTIMESTAMP: UInt { get }

/// Set ttywait timeout.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCSDRAINWAIT: UInt { get }

/// Get ttywait timeout.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCGDRAINWAIT: UInt { get }

/// Download microcode to DSI Softmodem.
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCDSIMICROCODE: UInt { get }

/// Grantpt(3).
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCPTYGRANT: UInt { get }

/// Ptsname(3).
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCPTYGNAME: UInt { get }

/// Unlockpt(3).
@available(macOS 10.9, iOS 7.0, watchOS
2.0, tvOS 9.0, bridgeOS 7.0, visionOS
1.0, *)
public let TIOCPTYUNLK: UInt { get }
```

```
// MARK: - xlocale Additions

import Darwin.POSIX.langinfo
import Darwin.POSIX.langinfo.xlocale
import Darwin.POSIX.monetary
import Darwin.POSIX.monetary.xlocale
import Darwin.POSIX.regex
import Darwin.POSIX.regex.xlocale
import xlocale

// Available when xlocale is imported
with Darwin
public typealias __NSConstantString =
__NSConstantString_tag

// Available when xlocale is imported
with Darwin
public typealias __builtin_ms_va_list =
UnsafeMutablePointer<CChar>

// Available when xlocale is imported
with Darwin
public typealias __builtin_va_list =
UnsafeMutablePointer<CChar>
```