```
import Darwin
import Darwin.Mach
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims

/// A namespace for C and platform types
@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
public enum CInterop {

    public typealias Mode = mode_t

    /// The C `char` type
    public typealias Char = CChar

    /// The platform's preferred
character type. On Unix, this is an 8-bit
C
    /// `char` (which may be signed or
unsigned, depending on platform). On
    /// Windows, this is `UInt16` (a
"wide" character).
    public typealias PlatformChar =
CInterop.Char

    /// The platform's preferred Unicode
encoding. On Unix this is UTF-8 and on
    /// Windows it is UTF-16. Native
strings may contain invalid Unicode,
    /// which will be handled by either
error-correction or failing, depending
    /// on API.
    public typealias
PlatformUnicodeEncoding = UTF8
}

/// The C `mode_t` type.
@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
public typealias CModeT = mode_t

/// An error number used by system calls
to communicate what kind of error
/// occurred.
@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
@frozen public struct Errno :
RawRepresentable, Error, Hashable,
Codable {

    /// The raw C error number.
    public let rawValue: CInt

    /// Creates a strongly typed error
number from a raw C error number.
    public init(rawValue: CInt)

    /// Error. Not used.
    public static var notUsed: Errno {
get }

    /// Operation not permitted.
    ///
    /// An attempt was made to perform an
operation
```

```
    /// limited to processes with
appropriate privileges
    /// or to the owner of a file or
other resources.
    ///
    /// The corresponding C error is
`EPERM`.
    public static var notPermitted: Errno
{ get }

    /// No such file or directory.
    ///
    /// A component of a specified
pathname didn't exist,
    /// or the pathname was an empty
string.
    ///
    /// The corresponding C error is
`ENOENT`.
    public static var
noSuchFileOrDirectory: Errno { get }

    /// No such process.
    ///
    /// There isn't a process that
corresponds to the specified process ID.
    ///
    /// The corresponding C error is
`ESRCH`.
    public static var noSuchProcess:
Errno { get }

    /// Interrupted function call.
    ///
    /// The process caught an
asynchronous signal (such as `SIGINT` or
`SIGQUIT`)
    /// during the execution of an
interruptible function.
    /// If the signal handler performs a
normal return,
    /// the caller of the interrupted
function call receives this error.
    ///
    /// The corresponding C error is
`EINTR`.
    public static var interrupted: Errno
{ get }

    /// Input/output error.
    ///
    /// Some physical input or output
error occurred.
    /// This error isn't reported until
    /// you attempt a subsequent
operation on the same file descriptor,
    /// and the error may be lost
(overwritten) by subsequent errors.
    ///
    /// The corresponding C error is
`EIO`.
    public static var ioError: Errno {
get }

    /// No such device or address.
    ///
    /// Input or output on a special file
referred to a device that didn't exist,
    /// or made a request beyond the
limits of the device.
    /// This error may also occur when,
for example,
    /// a tape drive isn't online or when
there isn't a disk pack loaded on a
drive.
    ///
    /// The corresponding C error is
`ENXIO`.
    public static var
noSuchAddressOrDevice: Errno { get }

    /// The argument list is too long.
    ///
    /// The number of bytes
    /// used for the argument and
environment list of the new process
    /// exceeded the limit `NCARGS`, as
defined in `<sys/param.h>`.
    ///
    /// The corresponding C error is
`E2BIG`.
    public static var argListTooLong:
Errno { get }

    /// Executable format error.
    ///
    /// A request was made to execute a
file that,
    /// although it has the appropriate
```

```
permissions,
    /// isn't in the format required for
an executable file.
    ///
    /// The corresponding C error is
`ENOEXEC`.
    public static var execFormatError:
Errno { get }

    /// Bad file descriptor.
    ///
    /// A file descriptor argument was
out of range,
    /// referred to no open file,
    /// or a read (write) request was
made to a file
    /// that was only open for writing
(reading).
    ///
    /// The corresponding C error is
`EBADF`.
    public static var badFileDescriptor:
Errno { get }

    /// No child processes.
    ///
    /// A `wait(2)` or `waitpid(2)`
function was executed
    /// by a process that dosn't have any
existing child processes
    /// or whose child processes are all
already being waited for.
    ///
    /// The corresponding C error is
`ECHILD`.
    public static var noChildProcess:
Errno { get }

    /// Resource deadlock avoided.
    ///
    /// You attempted to lock a system
resource
    /// that would have resulted in a
deadlock.
    ///
    /// The corresponding C error is
`EDEADLK`.
    public static var deadlock: Errno {
get }

    /// Can't allocate memory.
    ///
    /// The new process image required
more memory
    /// than was allowed by the hardware
    /// or by system-imposed memory
management constraints.
    /// A lack of swap space is normally
temporary;
    /// however, a lack of core is not.
    /// You can increase soft limits up
to their corresponding hard limits.
    ///
    /// The corresponding C error is
`ENOMEM`.
    public static var noMemory: Errno {
```

```
get }

    /// Permission denied.
    ///
    /// You attempted to access a file
    /// in a way that's forbidden by the
file's access permissions.
    ///
    /// The corresponding C error is
`EACCES`.
    public static var permissionDenied:
Errno { get }

    /// Bad address.
    ///
    /// An address passed as an argument
to a system call was invalid.
    ///
    /// The corresponding C error is
`EFAULT`.
    public static var badAddress: Errno {
get }

    /// Not a block device.
    ///
    /// You attempted a block device
operation on a nonblock device or file.
    ///
    /// The corresponding C error is
`ENOTBLK`.
    public static var notBlockDevice:
Errno { get }
```

```
    /// Resource busy.
    ///
    /// You attempted to use a system
resource which was in use at the time,
    /// in a manner that would have
conflicted with the request.
    ///
    /// The corresponding C error is
`EBUSY`.
    public static var resourceBusy: Errno
{ get }

    /// File exists.
    ///
    /// An existing file was mentioned in
an inappropriate context;
    /// for example, as the new link name
in a link function.
    ///
    /// The corresponding C error is
`EEXIST`.
    public static var fileExists: Errno {
get }

    /// Improper link.
    ///
    /// You attempted to create a hard
link to a file on another file system.
    ///
    /// The corresponding C error is
`EXDEV`.
    public static var improperLink: Errno
{ get }
```

```
    /// Operation not supported by
device.
    ///
    /// You attempted to apply an
inappropriate function to a device;
    /// for example, trying to read a
write-only device such as a printer.
    ///
    /// The corresponding C error is
`ENODEV`.
    public static var
operationNotSupportedByDevice: Errno {
get }

    /// Not a directory.
    ///
    /// A component of the specified
pathname exists,
    /// but it wasn't a directory,
    /// when a directory was expected.
    ///
    /// The corresponding C error is
`ENOTDIR`.
    public static var notDirectory: Errno
{ get }

    /// Is a directory.
    ///
    /// You attempted to open a directory
with write mode specified.
    /// Directories can be opened only in
read mode.
```

```
    ///
    /// The corresponding C error is
`EISDIR`.
    public static var isDirectory: Errno
{ get }

    /// Invalid argument.
    ///
    /// One or more of the specified
arguments wasn't valid;
    /// for example, specifying an
undefined signal to a signal or kill
function.
    ///
    /// The corresponding C error is
`EINVAL`.
    public static var invalidArgument:
Errno { get }

    /// The system has too many open
files.
    ///
    /// The maximum number of file
descriptors
    /// allowable on the system has been
reached;
    /// requests to open a file can't be
satisfied
    /// until you close at least one file
descriptor.
    ///
    /// The corresponding C error is
`ENFILE`.
```

```
    public static var
tooManyOpenFilesInSystem: Errno { get }

    /// This process has too many open
files.
    ///
    /// To check the current limit,
    /// call the `getdtablesize`
function.
    ///
    /// The corresponding C error is
`EMFILE`.
    public static var tooManyOpenFiles:
Errno { get }

    /// Inappropriate control function.
    ///
    /// You attempted a control function
    /// that can't be performed on the
specified file or device.
    /// For information about control
functions, see `ioctl(2)`.
    ///
    /// The corresponding C error is
`ENOTTY`.
    public static var
inappropriateIOCTLForDevice: Errno {
get }

    /// Text file busy.
    ///
    /// The new process was a pure
procedure (shared text) file,
    /// which was already open for
writing by another process,
    /// or while the pure procedure file
was being executed,
    /// an open call requested write
access.
    ///
    /// The corresponding C error is
`ETXTBSY`.
    public static var textFileBusy: Errno
{ get }

    /// The file is too large.
    ///
    /// The file exceeds the maximum size
allowed by the file system.
    /// For example, the maximum size on
UFS is about 2.1 gigabytes,
    /// and about 9,223 petabytes on HFS-
Plus and Apple File System.
    ///
    /// The corresponding C error is
`EFBIG`.
    public static var fileTooLarge: Errno
{ get }

    /// Device out of space.
    ///
    /// A write to an ordinary file,
    /// the creation of a directory or
symbolic link,
    /// or the creation of a directory
entry failed
```

```
    /// because there aren't any
available disk blocks on the file system,
    /// or the allocation of an inode for
a newly created file failed
    /// because there aren't any inodes
available on the file system.
    ///
    /// The corresponding C error is
`ENOSPC`.
    public static var noSpace: Errno {
get }

    /// Illegal seek.
    ///
    /// An `lseek(2)` function was issued
on a socket, pipe or FIFO.
    ///
    /// The corresponding C error is
`ESPIPE`.
    public static var illegalSeek: Errno
{ get }

    /// Read-only file system.
    ///
    /// You attempted to modify a file or
directory
    /// on a file system that was read-
only at the time.
    ///
    /// The corresponding C error is
`EROFS`.
    public static var readOnlyFileSystem:
Errno { get }
```

```
    /// Too many links.
    ///
    /// The maximum number of hard links
to a single file (32767)
    /// has been exceeded.
    ///
    /// The corresponding C error is
`EMLINK`.
    public static var tooManyLinks: Errno
{ get }

    /// Broken pipe.
    ///
    /// You attempted to write to a pipe,
socket, or FIFO
    /// that doesn't have a process
reading its data.
    ///
    /// The corresponding C error is
`EPIPE`.
    public static var brokenPipe: Errno {
get }

    /// Numerical argument out of domain.
    ///
    /// A numerical input argument was
outside the defined domain of the
    /// mathematical function.
    ///
    /// The corresponding C error is
`EDOM`.
    public static var outOfDomain: Errno
```

```
{ get }

    /// Numerical result out of range.
    ///
    /// A numerical result of the
function
    /// was too large to fit in the
available space;
    /// for example, because it exceeded
a floating point number's
    /// level of precision.
    ///
    /// The corresponding C error is
`ERANGE`.
    public static var outOfRange: Errno {
get }

    /// Resource temporarily unavailable.
    ///
    /// This is a temporary condition;
    /// later calls to the same routine
may complete normally.
    /// Make the same function call again
later.
    ///
    /// The corresponding C error is
`EAGAIN`.
    public static var
resourceTemporarilyUnavailable: Errno {
get }

    /// Operation now in progress.
    ///
    /// You attempted an operation that
takes a long time to complete,
    /// such as `connect(2)` or
`connectx(2)`,
    /// on a nonblocking object.
    /// See also `fcntl(2)`.
    ///
    /// The corresponding C error is
`EINPROGRESS`.
    public static var nowInProgress:
Errno { get }

    /// Operation already in progress.
    ///
    /// You attempted an operation on a
nonblocking object
    /// that already had an operation in
progress.
    ///
    /// The corresponding C error is
`EALREADY`.
    public static var alreadyInProcess:
Errno { get }

    /// A socket operation was performed
on something that isn't a socket.
    ///
    /// The corresponding C error is
`ENOTSOCK`.
    public static var notSocket: Errno {
get }

    /// Destination address required.
    ///
    /// A required address was omitted
from a socket operation.
    ///
    /// The corresponding C error is
`EDESTADDRREQ`.
    public static var addressRequired:
Errno { get }

    /// Message too long.
    ///
    /// A message sent on a socket was
larger than
    /// the internal message buffer or
some other network limit.
    ///
    /// The corresponding C error is
`EMSGSIZE`.
    public static var messageTooLong:
Errno { get }

    /// Protocol wrong for socket type.
    ///
    /// A protocol was specified that
doesn't support
    /// the semantics of the socket type
requested.
    /// For example,
    /// you can't use the ARPA Internet
UDP protocol with type `SOCK_STREAM`.
    ///
    /// The corresponding C error is
`EPROTOTYPE`.
```

```
    public static var
protocolWrongTypeForSocket: Errno { get }

    /// Protocol not available.
    ///
    /// A bad option or level was
specified
    /// in a `getsockopt(2)` or
`setsockopt(2)` call.
    ///
    /// The corresponding C error is
`ENOPROTOOPT`.
    public static var
protocolNotAvailable: Errno { get }

    /// Protocol not supported.
    ///
    /// The protocol hasn't been
configured into the system,
    /// or no implementation for it
exists.
    ///
    /// The corresponding C error is
`EPROTONOSUPPORT`.
    public static var
protocolNotSupported: Errno { get }

    /// Socket type not supported.
    ///
    /// Support for the socket type
hasn't been configured into the system
    /// or no implementation for it
exists.
```

```
    ///
    /// The corresponding C error is
`ESOCKTNOSUPPORT`.
    public static var
socketTypeNotSupported: Errno { get }

    /// Not supported.
    ///
    /// The attempted operation isn't
supported
    /// for the type of object
referenced.
    ///
    /// The corresponding C error is
`ENOTSUP`.
    public static var notSupported: Errno
{ get }

    /// Protocol family not supported.
    ///
    /// The protocol family hasn't been
configured into the system
    /// or no implementation for it
exists.
    ///
    /// The corresponding C error is
`EPFNOSUPPORT`.
    public static var
protocolFamilyNotSupported: Errno { get }

    /// The address family isn't
supported by the protocol family.
    ///
    /// An address incompatible with the
requested protocol was used.
    /// For example, you shouldn't
necessarily expect
    /// to be able to use name server
addresses with ARPA Internet protocols.
    ///
    /// The corresponding C error is
`EAFNOSUPPORT`.
    public static var
addressFamilyNotSupported: Errno { get }

    /// Address already in use.
    ///
    /// Only one use of each address is
normally permitted.
    ///
    /// The corresponding C error is
`EADDRINUSE`.
    public static var addressInUse: Errno
{ get }

    /// Can't assign the requested
address.
    ///
    /// This error normally results from
    /// an attempt to create a socket
with an address that isn't on this
machine.
    ///
    /// The corresponding C error is
`EADDRNOTAVAIL`.
    public static var
```

```
addressNotAvailable: Errno { get }

    /// Network is down.
    ///
    /// A socket operation encountered a
dead network.
    ///
    /// The corresponding C error is
`ENETDOWN`.
    public static var networkDown: Errno
{ get }

    /// Network is unreachable.
    ///
    /// A socket operation was attempted
to an unreachable network.
    ///
    /// The corresponding C error is
`ENETUNREACH`.
    public static var networkUnreachable:
Errno { get }

    /// Network dropped connection on
reset.
    ///
    /// The host you were connected to
crashed and restarted.
    ///
    /// The corresponding C error is
`ENETRESET`.
    public static var networkReset: Errno
{ get }
```

```
    /// Software caused a connection
abort.
    ///
    /// A connection abort was caused
internal to your host machine.
    ///
    /// The corresponding C error is
`ECONNABORTED`.
    public static var connectionAbort:
Errno { get }

    /// Connection reset by peer.
    ///
    /// A connection was forcibly closed
by a peer.
    /// This normally results from a loss
of the connection
    /// on the remote socket due to a
timeout or a reboot.
    ///
    /// The corresponding C error is
`ECONNRESET`.
    public static var connectionReset:
Errno { get }

    /// No buffer space available.
    ///
    /// An operation on a socket or pipe
wasn't performed
    /// because the system lacked
sufficient buffer space
    /// or because a queue was full.
    ///
    /// The corresponding C error is
`ENOBUFS`.
    public static var noBufferSpace:
Errno { get }

    /// Socket is already connected.
    ///
    /// A `connect(2)` or `connectx(2)`
request was made
    /// on an already connected socket,
    /// or a `sendto(2)` or `sendmsg(2)`
request was made
    /// on a connected socket specified a
destination when already connected.
    ///
    /// The corresponding C error is
`EISCONN`.
    public static var socketIsConnected:
Errno { get }

    /// Socket is not connected.
    ///
    /// A request to send or receive data
wasn't permitted
    /// because the socket wasn't
connected and,
    /// when sending on a datagram
socket,
    /// no address was supplied.
    ///
    /// The corresponding C error is
`ENOTCONN`.
    public static var socketNotConnected:
```

```
Errno { get }

    /// Can't send after socket shutdown.
    ///
    /// A request to send data wasn't
permitted
    /// because the socket had already
been shut down
    /// with a previous `shutdown(2)`
call.
    ///
    /// The corresponding C error is
`ESHUTDOWN`.
    public static var socketShutdown:
Errno { get }

    /// Operation timed out.
    ///
    /// A `connect(2)`, `connectx(2)` or
`send(2)` request failed
    /// because the connected party
didn't properly respond
    /// within the required period of
time.
    /// The timeout period is dependent
on the communication protocol.
    ///
    /// The corresponding C error is
`ETIMEDOUT`.
    public static var timedOut: Errno {
get }

    /// Connection refused.
    ///
    /// No connection could be made
    /// because the target machine
actively refused it.
    /// This usually results from trying
to connect to a service
    /// that's inactive on the foreign
host.
    ///
    /// The corresponding C error is
`ECONNREFUSED`.
    public static var connectionRefused:
Errno { get }

    /// Too many levels of symbolic
links.
    ///
    /// A pathname lookup involved more
than eight symbolic links.
    ///
    /// The corresponding C error is
`ELOOP`.
    public static var
tooManySymbolicLinkLevels: Errno { get }

    /// The file name is too long.
    ///
    /// A component of a pathname
exceeded 255 (`MAXNAMELEN`) characters,
    /// or an entire pathname exceeded
1023 (`MAXPATHLEN-1`) characters.
    ///
    /// The corresponding C error is
```

```
`ENAMETOOLONG`.
    public static var fileNameTooLong:
Errno { get }

    /// The host is down.
    ///
    /// A socket operation failed because
the destination host was down.
    ///
    /// The corresponding C error is
`EHOSTDOWN`.
    public static var hostIsDown: Errno {
get }

    /// No route to host.
    ///
    /// A socket operation failed because
the destination host was unreachable.
    ///
    /// The corresponding C error is
`EHOSTUNREACH`.
    public static var noRouteToHost:
Errno { get }

    /// Directory not empty.
    ///
    /// A directory with entries other
than `.` and `..`
    /// was supplied to a `remove(2)`
directory or `rename(2)` call.
    ///
    /// The corresponding C error is
`ENOTEMPTY`.
```

```
    public static var directoryNotEmpty:
Errno { get }

    /// Too many processes.
    ///
    /// The corresponding C error is
`EPROCLIM`.
    public static var tooManyProcesses:
Errno { get }

    /// Too many users.
    ///
    /// The quota system ran out of table
entries.
    ///
    /// The corresponding C error is
`EUSERS`.
    public static var tooManyUsers: Errno
{ get }

    /// Disk quota exceeded.
    ///
    /// A write to an ordinary file,
    /// the creation of a directory or
symbolic link,
    /// or the creation of a directory
entry failed
    /// because the user's quota of disk
blocks was exhausted,
    /// or the allocation of an inode for
a newly created file failed
    /// because the user's quota of
inodes was exhausted.
```

```
    ///
    /// The corresponding C error is
`EDQUOT`.
    public static var diskQuotaExceeded:
Errno { get }

    /// Stale NFS file handle.
    ///
    /// You attempted access an open file
on an NFS filesystem,
    /// which is now unavailable as
referenced by the given file descriptor.
    /// This may indicate that the file
was deleted on the NFS server
    /// or that some other catastrophic
event occurred.
    ///
    /// The corresponding C error is
`ESTALE`.
    public static var staleNFSFileHandle:
Errno { get }

    /// The structure of the remote
procedure call (RPC) is bad.
    ///
    /// Exchange of RPC information was
unsuccessful.
    ///
    /// The corresponding C error is
`EBADRPC`.
    public static var rpcUnsuccessful:
Errno { get }
```

```
    /// The version of the remote
procedure call (RPC) is incorrect.
    ///
    /// The version of RPC on the remote
peer
    /// isn't compatible with the local
version.
    ///
    /// The corresponding C error is
`ERPCMISMATCH`.
    public static var rpcVersionMismatch:
Errno { get }

    /// The remote procedure call (RPC)
program isn't available.
    ///
    /// The requested program isn't
registered on the remote host.
    ///
    /// The corresponding C error is
`EPROGUNAVAIL`.
    public static var
rpcProgramUnavailable: Errno { get }

    /// The version of the remote
procedure call (RPC) program is
incorrect.
    ///
    /// The requested version of the
program
    /// isn't available on the remote
host.
    ///
```

```swift
    /// The corresponding C error is
`EPROGMISMATCH`.
    public static var
rpcProgramVersionMismatch: Errno { get }

    /// Bad procedure for program.
    ///
    /// A remote procedure call was
attempted for a procedure
    /// that doesn't exist in the remote
program.
    ///
    /// The corresponding C error is
`EPROCUNAVAIL`.
    public static var
rpcProcedureUnavailable: Errno { get }

    /// No locks available.
    ///
    /// You have reached the system-
imposed limit
    /// on the number of simultaneous
files.
    ///
    /// The corresponding C error is
`ENOLCK`.
    public static var noLocks: Errno {
get }

    /// Function not implemented.
    ///
    /// You attempted a system call that
isn't available on this system.
```

```
    ///
    /// The corresponding C error is
`ENOSYS`.
    public static var noFunction: Errno {
get }

    /// Inappropriate file type or
format.
    ///
    /// The file was the wrong type for
the operation,
    /// or a data file had the wrong
format.
    ///
    /// The corresponding C error is
`EFTYPE`.
    public static var
badFileTypeOrFormat: Errno { get }

    /// Authentication error.
    ///
    /// The authentication ticket used to
mount an NFS file system was invalid.
    ///
    /// The corresponding C error is
`EAUTH`.
    public static var
authenticationError: Errno { get }

    /// Need authenticator.
    ///
    /// Before mounting the given NFS
file system,
    /// you must obtain an authentication
ticket.
    ///
    /// The corresponding C error is
`ENEEDAUTH`.
    public static var needAuthenticator:
Errno { get }

    /// Device power is off.
    ///
    /// The corresponding C error is
`EPWROFF`.
    public static var devicePowerIsOff:
Errno { get }

    /// Device error.
    ///
    /// A device error has occurred;
    /// for example, a printer running
out of paper.
    ///
    /// The corresponding C error is
`EDEVERR`.
    public static var deviceError: Errno
{ get }

    /// Value too large to be stored in
data type.
    ///
    /// A numerical result of the
function
    /// is too large to be stored in the
space that the caller provided.
```

```
    ///
    /// The corresponding C error is
`EOVERFLOW`.
    public static var overflow: Errno {
get }

    /// Bad executable or shared library.
    ///
    /// The executable or shared library
being referenced was malformed.
    ///
    /// The corresponding C error is
`EBADEXEC`.
    public static var badExecutable:
Errno { get }

    /// Bad CPU type in executable.
    ///
    /// The specified executable doesn't
support the current CPU.
    ///
    /// The corresponding C error is
`EBADARCH`.
    public static var badCPUType: Errno {
get }

    /// Shared library version mismatch.
    ///
    /// The version of the shared library
on the system
    /// doesn't match the expected
version.
    ///
    /// The corresponding C error is
`ESHLIBVERS`.
    public static var
sharedLibraryVersionMismatch: Errno { get
}

    /// Malformed Mach-O file.
    ///
    /// The Mach object file is
malformed.
    ///
    /// The corresponding C error is
`EBADMACHO`.
    public static var malformedMachO:
Errno { get }

    /// Operation canceled.
    ///
    /// The scheduled operation was
canceled.
    ///
    /// The corresponding C error is
`ECANCELED`.
    public static var canceled: Errno {
get }

    /// Identifier removed.
    ///
    /// An IPC identifier was removed
while the current process was waiting on
it.
    ///
    /// The corresponding C error is
```

```
`EIDRM`.
    public static var identifierRemoved:
Errno { get }

    /// No message of desired type.
    ///
    /// An IPC message queue doesn't
contain a message of the desired type,
    /// or a message catalog doesn't
contain the requested message.
    ///
    /// The corresponding C error is
`ENOMSG`.
    public static var noMessage: Errno {
get }

    /// Illegal byte sequence.
    ///
    /// While decoding a multibyte
character,
    /// the function encountered an
invalid or incomplete sequence of bytes,
    /// or the given wide character is
invalid.
    ///
    /// The corresponding C error is
`EILSEQ`.
    public static var
illegalByteSequence: Errno { get }

    /// Attribute not found.
    ///
    /// The specified extended attribute
```

```
doesn't exist.
    ///
    /// The corresponding C error is
`ENOATTR`.
    public static var attributeNotFound:
Errno { get }

    /// Bad message.
    ///
    /// The message to be received is
inappropriate
    /// for the attempted operation.
    ///
    /// The corresponding C error is
`EBADMSG`.
    public static var badMessage: Errno {
get }

    /// Reserved.
    ///
    /// This error is reserved for future
use.
    ///
    /// The corresponding C error is
`EMULTIHOP`.
    public static var multiHop: Errno {
get }

    /// No message available.
    ///
    /// No message was available to be
received by the requested operation.
    ///
    /// The corresponding C error is
`ENODATA`.
    public static var noData: Errno { get
}

    /// Reserved.
    ///
    /// This error is reserved for future
use.
    ///
    /// The corresponding C error is
`ENOLINK`.
    public static var noLink: Errno { get
}

    /// Reserved.
    ///
    /// This error is reserved for future
use.
    ///
    /// The corresponding C error is
`ENOSR`.
    public static var noStreamResources:
Errno { get }

    /// Reserved.
    ///
    /// This error is reserved for future
use.
    ///
    /// The corresponding C error is
`ENOSTR`.
    public static var notStream: Errno {
```

```
get }

    /// Protocol error.
    ///
    /// Some protocol error occurred.
    /// This error is device-specific,
    /// but generally isn't related to a
hardware failure.
    ///
    /// The corresponding C error is
`EPROTO`.
    public static var protocolError:
Errno { get }

    /// Reserved.
    ///
    /// This error is reserved for future
use.
    ///
    /// The corresponding C error is
`ETIME`.
    public static var timeout: Errno {
get }

    /// Operation not supported on
socket.
    ///
    /// The attempted operation isn't
supported for the type of socket
referenced;
    /// for example, trying to accept a
connection on a datagram socket.
    ///
    /// The corresponding C error is
`EOPNOTSUPP`.
    public static var
notSupportedOnSocket: Errno { get }

    /// 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 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
    public typealias RawValue = CInt
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension Errno {

    /// Operation would block.
    ///
    /// The corresponding C error is
`EWOULDBLOCK`.
    public static var wouldBlock: Errno {
get }
```

```
    /// Too many references: can't
splice.
    ///
    /// The corresponding C error is
`ETOOMANYREFS`.
    public static var tooManyReferences:
Errno { get }

    /// Too many levels of remote in
path.
    ///
    /// The corresponding C error is
`EREMOTE`.
    public static var
tooManyRemoteLevels: Errno { get }

    /// No such policy registered.
    ///
    /// The corresponding C error is
`ENOPOLICY`.
    public static var noSuchPolicy: Errno
{ get }

    /// State not recoverable.
    ///
    /// The corresponding C error is
`ENOTRECOVERABLE`.
    public static var notRecoverable:
Errno { get }

    /// Previous pthread mutex owner
died.
    ///
    /// The corresponding C error is
`EOWNERDEAD`.
    public static var previousOwnerDied:
Errno { get }

    /// Interface output queue is full.
    ///
    /// The corresponding C error is
`EQFULL`.
    public static var outputQueueFull:
Errno { get }

    /// The largest valid error.
    ///
    /// This value is the largest valid
value
    /// encountered using the C `errno`
global variable.
    /// It isn't a valid error.
    ///
    /// The corresponding C error is
`ELAST`.
    public static var lastErrnoValue:
Errno { get }
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension Errno :
CustomStringConvertible,
CustomDebugStringConvertible {

    ///  A textual representation of the
```

```
most recent error
    ///  returned by a system call.
    ///
    /// The corresponding C function is
`strerror(3)`.
    public var description: String {
get }

    ///  A textual representation,
    ///  suitable for debugging,
    ///  of the most recent error
returned by a system call.
    ///
    /// The corresponding C function is
`strerror(3)`.
    public var debugDescription: String {
get }
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension Errno {

    public static func ~= (lhs: Errno,
rhs: any Error) -> Bool
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension Errno : BitwiseCopyable {
}

/// An abstract handle to an input or
```

```
output data resource,
/// such as a file or a socket.
///
/// You are responsible for managing the
lifetime and validity
/// of `FileDescriptor` values,
/// in the same way as you manage a raw C
file handle.
@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
@frozen public struct FileDescriptor :
RawRepresentable, Hashable, Codable {

    /// The raw C file handle.
    public let rawValue: CInt

    /// Creates a strongly-typed file
handle from a raw C file handle.
    public init(rawValue: CInt)

    /// 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 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
    public typealias RawValue = CInt
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor {

    /// The standard input file
descriptor, with a numeric value of 0.
    public static var standardInput:
FileDescriptor { get }

    /// The standard output file
descriptor, with a numeric value of 1.
    public static var standardOutput:
FileDescriptor { get }

    /// The standard error file
descriptor, with a numeric value of 2.
    public static var standardError:
FileDescriptor { get }
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor {

    /// The desired read and write access
for a newly opened file.
    @available(macOS 11.0, iOS 14.0,
watchOS 7.0, tvOS 14.0, *)
    @frozen public struct AccessMode :
```

```
RawRepresentable, Sendable, Hashable,
Codable {

        /// The raw C access mode.
        public var rawValue: CInt

        /// Creates a strongly-typed
access mode from a raw C access mode.
        public init(rawValue: CInt)

        /// Opens the file for reading
only.
        ///
        /// The corresponding C constant
is `O_RDONLY`.
        public static var readOnly:
FileDescriptor.AccessMode { get }

        /// Opens the file for writing
only.
        ///
        /// The corresponding C constant
is `O_WRONLY`.
        public static var writeOnly:
FileDescriptor.AccessMode { get }

        /// Opens the file for reading
and writing.
        ///
        /// The corresponding C constant
is `O_RDWR`.
        public static var readWrite:
FileDescriptor.AccessMode { get }
```

```
        /// 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 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
        public typealias RawValue = CInt
    }

    /// Options that specify behavior for
a newly-opened file.
    @available(macOS 11.0, iOS 14.0,
watchOS 7.0, tvOS 14.0, *)
    @frozen public struct OpenOptions :
OptionSet, Sendable, Hashable, Codable {

        /// The raw C options.
        public var rawValue: CInt

        /// Create a strongly-typed
options value from raw C options.
        public init(rawValue: CInt)

        /// Indicates that opening the
```

```
file doesn't
        /// wait for the file or device
to become available.
        ///
        /// If this option is specified,
        /// the system doesn't wait for
the device or file
        /// to be ready or available.
        /// If the
        ///
<doc:FileDescriptor/open(_:_:options:perm
issions:retryOnInterrupt:)-2266j>
        /// call would result in the
process being blocked for some reason,
        /// that method returns
immediately.
        /// This flag also has the effect
of making all
        /// subsequent input and output
operations on the open file nonblocking.
        ///
        /// The corresponding C constant
is `O_NONBLOCK`.
        public static var nonBlocking:
FileDescriptor.OpenOptions { get }

        /// Indicates that each write
operation appends to the file.
        ///
        /// If this option is specified,
        /// each time you write to the
file,
        /// the new data is written at
```

```
the end of the file,
        /// after all existing file data.
        ///
        /// The corresponding C constant
is `O_APPEND`.
        public static var append:
FileDescriptor.OpenOptions { get }

        /// Indicates that opening the
file creates the file if it doesn't
exist.
        ///
        /// The corresponding C constant
is `O_CREAT`.
        public static var create:
FileDescriptor.OpenOptions { get }

        /// Indicates that opening the
file truncates the file if it exists.
        ///
        /// If this option is specified
and the file exists,
        /// the file is truncated to zero
bytes
        /// before any other operations
are performed.
        ///
        /// The corresponding C constant
is `O_TRUNC`.
        public static var truncate:
FileDescriptor.OpenOptions { get }

        /// Indicates that opening the
```

```
file creates the file,
        /// expecting that it doesn't
exist.
        ///
        /// If this option and ``create``
are both specified and the file exists,
        ///
<doc:FileDescriptor/open(_:_:options:perm
issions:retryOnInterrupt:)-2266j>
        /// returns an error instead of
creating the file.
        /// You can use this, for
example,
        /// to implement a simple
exclusive-access locking mechanism.
        ///
        /// If this option and ``create``
are both specified
        /// and the last component of the
file's path is a symbolic link,
        ///
<doc:FileDescriptor/open(_:_:options:perm
issions:retryOnInterrupt:)-2266j>
        /// fails even if the symbolic
link points to a nonexistent name.
        ///
        /// The corresponding C constant
is `O_EXCL`.
        public static var
exclusiveCreate:
FileDescriptor.OpenOptions { get }

        /// Indicates that opening the
```

```
file
        /// atomically obtains a shared
lock on the file.
        ///
        /// Setting this option or the
``exclusiveLock`` option
        /// obtains a lock with
`flock(2)` semantics.
        /// If you're creating a file
using the ``create`` option,
        /// the request for the lock
always succeeds
        /// except on file systems that
don't support locking.
        ///
        /// The corresponding C constant
is `O_SHLOCK`.
        public static var sharedLock:
FileDescriptor.OpenOptions { get }

        /// Indicates that opening the
file
        /// atomically obtains an
exclusive lock.
        ///
        /// Setting this option or the
``sharedLock`` option.
        /// obtains a lock with
`flock(2)` semantics.
        /// If you're creating a file
using the ``create`` option,
        /// the request for the lock
always succeeds
```

```
        /// except on file systems that
don't support locking.
        ///
        /// The corresponding C constant
is `O_EXLOCK`.
        public static var exclusiveLock:
FileDescriptor.OpenOptions { get }

        /// Indicates that opening the
file doesn't follow symlinks.
        ///
        /// If you specify this option
        /// and the file path you pass to
        ///
<doc:FileDescriptor/open(_:_:options:perm
issions:retryOnInterrupt:)-2266j>
        /// is a symbolic link,
        /// then that open operation
fails.
        ///
        /// The corresponding C constant
is `O_NOFOLLOW`.
        public static var noFollow:
FileDescriptor.OpenOptions { get }

        /// Indicates that opening the
file only succeeds if the file is a
directory.
        ///
        /// If you specify this option
and the file path you pass to
        ///
<doc:FileDescriptor/open(_:_:options:perm
issions:retryOnInterrupt:)-2266j>
        /// is a not a directory, then
that open operation fails.
        ///
        /// The corresponding C constant
is `O_DIRECTORY`.
        public static var directory:
FileDescriptor.OpenOptions { get }

        /// Indicates that opening the
file
        /// opens symbolic links instead
of following them.
        ///
        /// If you specify this option
        /// and the file path you pass to
        ///
<doc:FileDescriptor/open(_:_:options:perm
issions:retryOnInterrupt:)-2266j>
        /// is a symbolic link,
        /// then the link itself is
opened instead of what it links to.
        ///
        /// The corresponding C constant
is `O_SYMLINK`.
        public static var symlink:
FileDescriptor.OpenOptions { get }

        /// Indicates that opening the
file monitors a file for changes.
        ///
        /// Specify this option when
opening a file for event notifications,
        /// such as a file handle
returned by the `kqueue(2)` function,
        /// rather than for reading or
writing.
        /// Files opened with this option
        /// don't prevent their
containing volume from being unmounted.
        ///
        /// The corresponding C constant
is `O_EVTONLY`.
        public static var eventOnly:
FileDescriptor.OpenOptions { get }

        /// Indicates that executing a
program closes the file.
        ///
        /// Normally, file descriptors
remain open
        /// across calls to the `exec(2)`
family of functions.
        /// If you specify this option,
        /// the file descriptor is closed
when replacing this process
        /// with another process.
        ///
        /// The state of the file
        /// descriptor flags can be
inspected using `F_GETFD`,
        /// as described in the
`fcntl(2)` man page.
        ///
        /// The corresponding C constant
is `O_CLOEXEC`.
```

```
        public static var closeOnExec:
FileDescriptor.OpenOptions { get }

        /// The type of the elements of
an array literal.
        @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
        public typealias
ArrayLiteralElement =
FileDescriptor.OpenOptions

        /// The element type of the
option set.
        ///
        /// To inherit all the default
implementations from the `OptionSet`
protocol,
        /// the `Element` type must be
`Self`, the default.
        @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
        public typealias Element =
FileDescriptor.OpenOptions

        /// 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 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
        public typealias RawValue = CInt
    }

    /// Options for specifying what a
file descriptor's offset is relative to.
    @available(macOS 11.0, iOS 14.0,
watchOS 7.0, tvOS 14.0, *)
    @frozen public struct SeekOrigin :
RawRepresentable, Sendable, Hashable,
Codable {

        /// The raw C value.
        public var rawValue: CInt

        /// Create a strongly-typed seek
origin from a raw C value.
        public init(rawValue: CInt)

        /// Indicates that the offset
should be set to the specified value.
        ///
        /// The corresponding C constant
is `SEEK_SET`.
        public static var start:
FileDescriptor.SeekOrigin { get }

        /// Indicates that the offset
should be set
```

```
        /// to the specified number of
bytes after the current location.
        ///
        /// The corresponding C constant
is `SEEK_CUR`.
        public static var current:
FileDescriptor.SeekOrigin { get }

        /// Indicates that the offset
should be set
        /// to the size of the file plus
the specified number of bytes.
        ///
        /// The corresponding C constant
is `SEEK_END`.
        public static var end:
FileDescriptor.SeekOrigin { get }

        /// Indicates that the offset
should be set
        /// to the next hole after the
specified number of bytes.
        ///
        /// For information about what is
considered a hole,
        /// see the `lseek(2)` man page.
        ///
        /// The corresponding C constant
is `SEEK_HOLE`.
        public static var nextHole:
FileDescriptor.SeekOrigin { get }

        /// Indicates that the offset
```

```
should be set
        /// to the start of the next file
region
        /// that isn't a hole
        /// and is greater than or equal
to the supplied offset.
        ///
        /// The corresponding C constant
is `SEEK_DATA`.
        public static var nextData:
FileDescriptor.SeekOrigin { get }

        /// 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 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
        public typealias RawValue = CInt
    }
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor {
    /// Runs a closure and then closes
the file descriptor, even if an error
occurs.
    ///
    /// - Parameter body: The closure to
run.
    ///   If the closure throws an error,
    ///   this method closes the file
descriptor before it rethrows that error.
    ///
    /// - Returns: The value returned by
the closure.
    ///
    /// If `body` throws an error
    /// or an error occurs while closing
the file descriptor,
    /// this method rethrows that error.
    public func closeAfter<R>(_ body: ()
throws -> R) throws -> R

    /// Writes a sequence of bytes to the
current offset
    /// and then updates the offset.
    ///
    /// - Parameter sequence: The bytes
to write.
    /// - Returns: The number of bytes
written, equal to the number of elements
in `sequence`.
    ///
    /// This method either writes the
entire contents of `sequence`,
    /// or throws an error if only part
of the content was written.
    ///
    /// Writes to the position associated
with this file descriptor, and
    /// increments that position by the
number of bytes written.
    /// See also ``seek(offset:from:)``.
    ///
    /// If `sequence` doesn't implement
    /// the
<doc://com.apple.documentation/documentat
ion/swift/sequence/3128824-
withcontiguousstorageifavailable> method,
    /// temporary space will be allocated
as needed.
    @discardableResult
    public func writeAll<S>(_ sequence:
S) throws -> Int where S : Sequence,
S.Element == UInt8

    /// Writes a sequence of bytes to the
given offset.
    ///
    /// - Parameters:
    ///   - offset: The file offset where
writing begins.
    ///   - sequence: The bytes to write.
    /// - Returns: The number of bytes
written, equal to the number of elements
in `sequence`.
    ///
    /// This method either writes the
```

```
entire contents of `sequence`,
    /// or throws an error if only part
of the content was written.
    /// Unlike ``writeAll(_:)``,
    /// this method preserves the file
descriptor's existing offset.
    ///
    /// If `sequence` doesn't implement
    /// the
<doc://com.apple.documentation/documentat
ion/swift/sequence/3128824-
withcontiguousstorageifavailable> method,
    /// temporary space will be allocated
as needed.
    @discardableResult
    public func
writeAll<S>(toAbsoluteOffset offset:
Int64, _ sequence: S) throws -> Int where
S : Sequence, S.Element == UInt8
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor {

    /// Opens or creates a file for
reading or writing.
    ///
    /// - Parameters:
    ///   - path: The location of the
file to open.
    ///   - mode: The read and write
access to use.
```

```
    ///   - options: The behavior for
opening the file.
    ///   - permissions: The file
permissions to use for created files.
    ///   - retryOnInterrupt: Whether to
retry the open operation
    ///     if it throws
``Errno/interrupted``.
    ///     The default is `true`.
    ///     Pass `false` to try only once
and throw an error upon interruption.
    /// - Returns: A file descriptor for
the open file
    ///
    /// The corresponding C function is
`open`.
    public static func open(_ path:
FilePath, _ mode:
FileDescriptor.AccessMode, options:
FileDescriptor.OpenOptions =
FileDescriptor.OpenOptions(),
permissions: FilePermissions? = nil,
retryOnInterrupt: Bool = true) throws ->
FileDescriptor

    /// Opens or creates a file for
reading or writing.
    ///
    /// - Parameters:
    ///   - path: The location of the
file to open.
    ///   - mode: The read and write
access to use.
```

```
    ///   - options: The behavior for
opening the file.
    ///   - permissions: The file
permissions to use for created files.
    ///   - retryOnInterrupt: Whether to
retry the open operation
    ///     if it throws
``Errno/interrupted``.
    ///     The default is `true`.
    ///     Pass `false` to try only once
and throw an error upon interruption.
    /// - Returns: A file descriptor for
the open file
    ///
    /// The corresponding C function is
`open`.
    public static func open(_ path:
UnsafePointer<CChar>, _ mode:
FileDescriptor.AccessMode, options:
FileDescriptor.OpenOptions =
FileDescriptor.OpenOptions(),
permissions: FilePermissions? = nil,
retryOnInterrupt: Bool = true) throws ->
FileDescriptor

    /// Deletes a file descriptor.
    ///
    /// Deletes the file descriptor from
the per-process object reference table.
    /// If this is the last reference to
the underlying object,
    /// the object will be deactivated.
    ///
    /// The corresponding C function is
`close`.
    public func close() throws

    /// Repositions the offset for the
given file descriptor.
    ///
    /// - Parameters:
    ///   - offset: The new offset for
the file descriptor.
    ///   - whence: The origin of the new
offset.
    /// - Returns: The file's offset
location,
    ///   in bytes from the beginning of
the file.
    ///
    /// The corresponding C function is
`lseek`.
    @discardableResult
    public func seek(offset: Int64, from
whence: FileDescriptor.SeekOrigin) throws
-> Int64

    /// Reads bytes at the current file
offset into a buffer.
    ///
    /// - Parameters:
    ///   - buffer: The region of memory
to read into.
    ///   - retryOnInterrupt: Whether to
retry the read operation
    ///     if it throws
```

```
``Errno/interrupted``.
    ///     The default is `true`.
    ///     Pass `false` to try only once
and throw an error upon interruption.
    /// - Returns: The number of bytes
that were read.
    ///
    /// The
<doc://com.apple.documentation/documentat
ion/swift/unsafemutablerawbufferpointer/
count-95usp> property of `buffer`
    /// determines the maximum number of
bytes that are read into that buffer.
    ///
    /// After reading,
    /// this method increments the file's
offset by the number of bytes read.
    /// To change the file's offset,
    /// call the ``seek(offset:from:)``
method.
    ///
    /// The corresponding C function is
`read`.
    public func read(into buffer:
UnsafeMutableRawBufferPointer,
retryOnInterrupt: Bool = true) throws ->
Int

    /// Reads bytes at the specified
offset into a buffer.
    ///
    /// - Parameters:
    ///   - offset: The file offset where
```

```
reading begins.
    ///   - buffer: The region of memory
to read into.
    ///   - retryOnInterrupt: Whether to
retry the read operation
    ///     if it throws
``Errno/interrupted``.
    ///     The default is `true`.
    ///     Pass `false` to try only once
and throw an error upon interruption.
    /// - Returns: The number of bytes
that were read.
    ///
    /// The
<doc://com.apple.documentation/documentat
ion/swift/unsafemutablerawbufferpointer/
count-95usp> property of `buffer`
    /// determines the maximum number of
bytes that are read into that buffer.
    ///
    /// Unlike
<doc:FileDescriptor/read(into:retryOnInte
rrupt:)>,
    /// this method leaves the file's
existing offset unchanged.
    ///
    /// The corresponding C function is
`pread`.
    public func read(fromAbsoluteOffset
offset: Int64, into buffer:
UnsafeMutableRawBufferPointer,
retryOnInterrupt: Bool = true) throws ->
Int
```

```
    /// Writes the contents of a buffer
at the current file offset.
    ///
    /// - Parameters:
    ///   - buffer: The region of memory
that contains the data being written.
    ///   - retryOnInterrupt: Whether to
retry the write operation
    ///     if it throws
``Errno/interrupted``.
    ///     The default is `true`.
    ///     Pass `false` to try only once
and throw an error upon interruption.
    /// - Returns: The number of bytes
that were written.
    ///
    /// After writing,
    /// this method increments the file's
offset by the number of bytes written.
    /// To change the file's offset,
    /// call the ``seek(offset:from:)``
method.
    ///
    /// The corresponding C function is
`write`.
    public func write(_ buffer:
UnsafeRawBufferPointer, retryOnInterrupt:
Bool = true) throws -> Int

    /// Writes the contents of a buffer
at the specified offset.
    ///
    /// - Parameters:
    ///   - offset: The file offset where
writing begins.
    ///   - buffer: The region of memory
that contains the data being written.
    ///   - retryOnInterrupt: Whether to
retry the write operation
    ///     if it throws
``Errno/interrupted``.
    ///     The default is `true`.
    ///     Pass `false` to try only once
and throw an error upon interruption.
    /// - Returns: The number of bytes
that were written.
    ///
    /// Unlike
``write(_:retryOnInterrupt:)``,
    /// this method leaves the file's
existing offset unchanged.
    ///
    /// The corresponding C function is
`pwrite`.
    public func write(toAbsoluteOffset
offset: Int64, _ buffer:
UnsafeRawBufferPointer, retryOnInterrupt:
Bool = true) throws -> Int
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FileDescriptor {

    /// Duplicates this file descriptor
```

```
and return the newly created copy.
    ///
    /// - Parameters:
    ///   - `target`: The desired target
file descriptor, or `nil`, in which case
    ///      the copy is assigned to the
file descriptor with the lowest raw value
    ///      that is not currently in use
by the process.
    ///   - retryOnInterrupt: Whether to
retry the write operation
    ///      if it throws
``Errno/interrupted``. The default is
`true`.
    ///      Pass `false` to try only
once and throw an error upon
interruption.
    /// - Returns: The new file
descriptor.
    ///
    /// If the `target` descriptor is
already in use, then it is first
    /// deallocated as if a close(2) call
had been done first.
    ///
    /// File descriptors are merely
references to some underlying system
resource.
    /// The system does not distinguish
between the original and the new file
    /// descriptor in any way. For
example, read, write and seek operations
on
```

```
    /// one of them also affect the
logical file position in the other, and
    /// append mode, non-blocking I/O and
asynchronous I/O options are shared
    /// between the references. If a
separate pointer into the file is
desired,
    /// a different object reference to
the file must be obtained by issuing an
    /// additional call to `open`.
    ///
    /// However, each file descriptor
maintains its own close-on-exec flag.
    ///
    ///
    /// The corresponding C functions are
`dup` and `dup2`.
    @available(macOS 12.0, iOS 15.0,
watchOS 8.0, tvOS 15.0, *)
    public func duplicate(as target:
FileDescriptor? = nil, retryOnInterrupt:
Bool = true) throws -> FileDescriptor
}

@available(macOS 12.3, iOS 15.4, watchOS
8.5, tvOS 15.4, *)
extension FileDescriptor {

    /// Creates a unidirectional data
channel, which can be used for
interprocess communication.
    ///
    /// - Returns: The pair of file
```

```
descriptors.
    ///
    /// The corresponding C function is
`pipe`.
    @available(macOS 12.3, iOS 15.4,
watchOS 8.5, tvOS 15.4, *)
    public static func pipe() throws ->
(readEnd: FileDescriptor, writeEnd:
FileDescriptor)
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension FileDescriptor {

    /// Truncates or extends the file
referenced by this file descriptor.
    ///
    /// - Parameters:
    ///   - newSize: The length in bytes
to resize the file to.
    ///   - retryOnInterrupt: Whether to
retry the write operation
    ///      if it throws
``Errno/interrupted``. The default is
`true`.
    ///      Pass `false` to try only
once and throw an error upon
interruption.
    ///
    /// The file referenced by this file
descriptor will by truncated (or
extended) to `newSize`.
```

```
    ///
    /// If the current size of the file
exceeds `newSize`, any extra data is
discarded. If the current
    /// size of the file is smaller than
`newSize`, the file is extended and
filled with zeros to the
    /// provided size.
    ///
    /// This function requires that the
file has been opened for writing.
    ///
    /// - Note: This function does not
modify the current offset for any open
file descriptors
    /// associated with the file.
    ///
    /// The corresponding C function is
`ftruncate`.
    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    public func resize(to newSize: Int64,
retryOnInterrupt: Bool = true) throws
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor : Sendable {
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor :
```

```
BitwiseCopyable {
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor.AccessMode :
CustomStringConvertible,
CustomDebugStringConvertible {

    /// A textual representation of the
access mode.
    public var description: String {
get }

    /// A textual representation of the
access mode, suitable for debugging
    public var debugDescription: String {
get }
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor.AccessMode :
BitwiseCopyable {
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor.OpenOptions :
CustomStringConvertible,
CustomDebugStringConvertible {

    /// A textual representation of the
open options.
    public var description: String {
get }

    /// A textual representation of the
open options, suitable for debugging.
    public var debugDescription: String {
get }
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor.OpenOptions :
BitwiseCopyable {
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor.SeekOrigin :
CustomStringConvertible,
CustomDebugStringConvertible {

    /// A textual representation of the
seek origin.
    public var description: String {
get }

    /// A textual representation of the
seek origin, suitable for debugging.
    public var debugDescription: String {
get }
}
```

```
@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FileDescriptor.SeekOrigin :
BitwiseCopyable {
}

/// Represents a location in the file
system.
///
/// This structure recognizes directory
separators  (e.g. `/`), roots, and
/// requires that the content terminates
in a NUL (`0x0`). Beyond that, it
/// does not give any meaning to the
bytes that it contains. The file system
/// defines how the content is
interpreted; for example, by its choice
of string
/// encoding.
///
/// On construction, `FilePath` will
normalize separators by removing
/// redundant intermediary separators and
stripping any trailing separators.
/// On Windows, `FilePath` will also
normalize forward slashes `/` into
/// backslashes `\`, as preferred by the
platform.
///
/// The code below creates a file path
from a string literal,
/// and then uses it to open and append
to a log file:
```

```
///
///     let message: String = "This is a
log message."
///     let path: FilePath = "/tmp/log"
///     let fd = try
FileDescriptor.open(path, .writeOnly,
options: .append)
///     try fd.closeAfter { try
fd.writeAll(message.utf8) }
///
/// File paths conform to the
///
<doc://com.apple.documentation/documentat
ion/swift/equatable>
/// and
<doc://com.apple.documentation/documentat
ion/swift/hashable> protocols
/// by performing the protocols'
operations on their raw byte contents.
/// This conformance allows file paths to
be used,
/// for example, as keys in a dictionary.
/// However, the rules for path
equivalence
/// are file-system–specific and have
additional considerations
/// like case insensitivity, Unicode
normalization, and symbolic links.
@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
public struct FilePath : Sendable {

    /// Creates an empty, null-terminated
```

```
path.
    public init()
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath {

    /// Returns true if this path
uniquely identifies the location of
    /// a file without reference to an
additional starting location.
    ///
    /// On Unix platforms, absolute paths
begin with a `/`. `isAbsolute` is
    /// equivalent to `root != nil`.
    ///
    /// On Windows, absolute paths are
fully qualified paths. `isAbsolute` is
    /// _not_ equivalent to `root != nil`
for traditional DOS paths
    /// (e.g. `C:foo` and `\bar` have
roots but are not absolute). UNC paths
    /// and device paths are always
absolute. Traditional DOS paths are
    /// absolute only if they begin with
a volume or drive followed by
    /// a `:` and a separator.
    ///
    /// NOTE: This does not perform shell
expansion or substitute
    /// environment variables; paths
beginning with `~` are considered
```

```
relative.
    ///
    /// Examples:
    /// * Unix:
    ///   * `/usr/local/bin`
    ///   * `/tmp/foo.txt`
    ///   * `/`
    /// * Windows:
    ///   * `C:\Users\`
    ///   * `\\?
\UNC\server\share\bar.exe`
    ///   * `\\server\share\bar.exe`
    public var isAbsolute: Bool { get }

    /// Returns true if this path is not
absolute (see `isAbsolute`).
    ///
    /// Examples:
    /// * Unix:
    ///   * `~/bar`
    ///   * `tmp/foo.txt`
    /// * Windows:
    ///   * `bar\baz`
    ///   * `C:Users\`
    ///   * `\Users`
    public var isRelative: Bool { get }

    /// Returns whether `other` is a
prefix of `self`, only considering
    /// whole path components.
    ///
    /// Example:
    ///
    ///     let path: FilePath =
"/usr/bin/ls"
    ///     path.starts(with: "/")
// true
    ///     path.starts(with: "/usr/bin")
// true
    ///     path.starts(with:
"/usr/bin/ls")    // true
    ///     path.starts(with:
"/usr/bin/ls///") // true
    ///     path.starts(with: "/us")
// false
    public func starts(with other:
FilePath) -> Bool

    /// Returns whether `other` is a
suffix of `self`, only considering
    /// whole path components.
    ///
    /// Example:
    ///
    ///     let path: FilePath =
"/usr/bin/ls"
    ///     path.ends(with: "ls")
// true
    ///     path.ends(with: "bin/ls")
// true
    ///     path.ends(with: "usr/bin/ls")
// true
    ///     path.ends(with:
"/usr/bin/ls///") // true
    ///     path.ends(with: "/ls")
// false
```

```
    public func ends(with other:
FilePath) -> Bool

    /// Whether this path is empty
    public var isEmpty: Bool { get }
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath {

    /// Returns the root of a path if
there is one, otherwise `nil`.
    ///
    /// On Unix, this will return the
leading `/` if the path is absolute
    /// and `nil` if the path is
relative.
    ///
    /// On Windows, for traditional DOS
paths, this will return
    /// the path prefix up to and
including a root directory or
    /// a supplied drive or volume.
Otherwise, if the path is relative to
    /// both the current directory and
current drive, returns `nil`.
    ///
    /// On Windows, for UNC or device
paths, this will return the path prefix
    /// up to and including the host and
share for UNC paths or the volume for
    /// device paths followed by any
```

```
subsequent separator.
    ///
    /// Examples:
    /// * Unix:
    ///   * `/foo/bar => /`
    ///   * `foo/bar  => nil`
    /// * Windows:
    ///   * `C:\foo\bar                => 
C:\`
    ///   * `C:foo\bar                 => 
C:`
    ///   * `\foo\bar                  => 
\ `
    ///   * `foo\bar                   => 
nil`
    ///   * `\\server\share\file       => 
\\server\share\`
    ///   * `\\?\UNC\server\share\file => 
\\?\UNC\server\share\`
    ///   * `\\.\device\folder         => 
\\.\device\`
    ///
    /// Setting the root to `nil` will 
remove the root and setting a new
    /// root will replace the root.
    ///
    /// Example:
    ///
    ///     var path: FilePath = 
"/foo/bar"
    ///     path.root = nil // path is 
"foo/bar"
    ///     path.root = "/" // path is
```

```
"/foo/bar"
    ///
    /// Example (Windows):
    ///
    ///     var path: FilePath =
#"\foo\bar"#
    ///     path.root = nil         //
path is #"foo\bar"#
    ///     path.root = "C:"        //
path is #"C:foo\bar"#
    ///     path.root = #"C:\"#     //
path is #"C:\foo\bar"#
    public var root: FilePath.Root?

    /// Creates a new path containing
just the components, i.e. everything
    /// after `root`.
    ///
    /// Returns self if `root == nil`.
    ///
    /// Examples:
    /// * Unix:
    ///   * `/foo/bar => foo/bar`
    ///   * `foo/bar  => foo/bar`
    ///   * `/        => ""`
    /// * Windows:
    ///   * `C:\foo\bar
=> foo\bar`
    ///   * `foo\bar
=> foo\bar`
    ///   * `\\?\UNC\server\share\file
=> file`
    ///   * `\\?\device\folder\file.exe
```

```
=> folder\file.exe`
    ///   * `\\server\share\file
=> file`
    ///   * `\
=> ""`
    public func removingRoot() ->
FilePath
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath {

    /// Returns the final component of
the path.
    /// Returns `nil` if the path is
empty or only contains a root.
    ///
    /// Note: Even if the final component
is a special directory
    /// (`.` or `..`), it will still be
returned. See `lexicallyNormalize()`.
    ///
    /// Examples:
    /// * Unix:
    ///   * `/usr/local/bin/ => bin`
    ///   * `/tmp/foo.txt    => foo.txt`
    ///   * `/tmp/foo.txt/.. => ..`
    ///   * `/tmp/foo.txt/.  => .`
    ///   * `/               => nil`
    /// * Windows:
    ///   * `C:\Users\
=> Users`
```

```
    ///   * `C:Users\
=> Users`
    ///   * `C:\
=> nil`
    ///   * `\Users\
=> Users`
    ///   * `\\?\UNC\server\share\bar.exe
=> bar.exe`
    ///   * `\\server\share
=> nil`
    ///   * `\\?\UNC\server\share\
=> nil`
    public var lastComponent:
FilePath.Component? { get }

    /// Creates a new path with
everything up to but not including
    /// `lastComponent`.
    ///
    /// If the path only contains a root,
returns `self`.
    /// If the path has no root and only
includes a single component,
    /// returns an empty FilePath.
    ///
    /// Examples:
    /// * Unix:
    ///   * `/usr/bin/ls => /usr/bin`
    ///   * `/foo        => /`
    ///   * `/           => /`
    ///   * `foo         => ""`
    /// * Windows:
    ///   * `C:\foo\bar.exe
```

```
=> C:\foo`
    ///   * `C:\
=> C:\`
    ///   * `\
\server\share\folder\file.txt => \
\server\share\folder`
    ///   * `\\server\share\
=> \\server\share\`
    public func removingLastComponent()
-> FilePath

    /// In-place mutating variant of
`removingLastComponent`.
    ///
    /// If `self` only contains a root,
does nothing and returns `false`.
    /// Otherwise removes `lastComponent`
and returns `true`.
    ///
    /// Example:
    ///
    ///     var path = "/usr/bin"
    ///     path.removeLastComponent() ==
true  // path is "/usr"
    ///     path.removeLastComponent() ==
true  // path is "/"
    ///     path.removeLastComponent() ==
false // path is "/"
    @discardableResult
    public mutating func
removeLastComponent() -> Bool
}
```

```
@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath {

    /// Represents a root of a file path.
    ///
    /// On Unix, a root is simply the
directory separator `/`.
    ///
    /// On Windows, a root contains the
entire path prefix up to and including
    /// the final separator.
    ///
    /// Examples:
    /// * Unix:
    ///   * `/`
    /// * Windows:
    ///   * `C:\`
    ///   * `C:`
    ///   * `\`
    ///   * `\\server\share\`
    ///   * `\\?\UNC\server\share\`
    ///   * `\\?\Volume{12345678-
abcd-1111-2222-123445789abc}\`
    @available(macOS 12.0, iOS 15.0,
watchOS 8.0, tvOS 15.0, *)
    public struct Root : Sendable {
    }

    /// Represents an individual, non-
root component of a file path.
    ///
    /// Components can be one of the
```

```
special directory components (`.` or
`..`)
    /// or a file or directory name.
Components are never empty and never
    /// contain the directory separator.
    ///
    /// Example:
    ///
    ///     var path: FilePath = "/tmp"
    ///     let file: FilePath.Component
= "foo.txt"
    ///     file.kind == .regular
// true
    ///     file.extension
// "txt"
    ///     path.append(file)
// path is "/tmp/foo.txt"
    @available(macOS 12.0, iOS 15.0,
watchOS 8.0, tvOS 15.0, *)
    public struct Component : Sendable {
    }
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath {

    /// The extension of the file or
directory last component.
    ///
    /// If `lastComponent` is `nil` or
one of the special path components
    /// `.` or `..`, `get` returns `nil`
```

```
and `set` does nothing.
    ///
    /// If `lastComponent` does not
contain a `.` anywhere, or only
    /// at the start, `get` returns `nil`
and `set` will append a
    /// `.` and `newValue` to
`lastComponent`.
    ///
    /// Otherwise `get` returns
everything after the last `.` and `set`
will
    /// replace the extension.
    ///
    /// Examples:
    ///   * `/tmp/foo.txt
=> txt`
    ///   * `/Applications/Foo.app/
=> app`
    ///   *
`/Applications/Foo.app/bar.txt => txt`
    ///   * `/tmp/foo.tar.gz
=> gz`
    ///   * `/tmp/.hidden
=> nil`
    ///   * `/tmp/.hidden.
=> ""`
    ///   * `/tmp/..
=> nil`
    ///
    /// Example:
    ///
    ///     var path = "/tmp/file"
    ///     path.extension = "txt" //
path is "/tmp/file.txt"
    ///     path.extension = "o"   //
path is "/tmp/file.o"
    ///     path.extension = nil    //
path is "/tmp/file"
    ///     path.extension = ""     //
path is "/tmp/file."
    public var `extension`: String?

    /// The non-extension portion of the
file or directory last component.
    ///
    /// Returns `nil` if `lastComponent`
is `nil`
    ///
    ///   * `/tmp/foo.txt
=> foo`
    ///   * `/Applications/Foo.app/
=> Foo`
    ///   *
`/Applications/Foo.app/bar.txt => bar`
    ///   * `/tmp/.hidden
=> .hidden`
    ///   * `/tmp/..
=> ..`
    ///   * `/
=> nil`
    public var stem: String? { get }
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
```

```
extension FilePath {

    /// Whether the path is in lexical-
normal form, that is `.` and `..`
    /// components have been collapsed
lexically (i.e. without following
    /// symlinks).
    ///
    /// Examples:
    /// *
`"/usr/local/bin".isLexicallyNormal ==
true`
    /// *
`"../local/bin".isLexicallyNormal   ==
true`
    /// *
`"local/bin/..".isLexicallyNormal   ==
false`
    public var isLexicallyNormal: Bool {
get }

    /// Collapse `.` and `..` components
lexically (i.e. without following
    /// symlinks).
    ///
    /// Examples:
    /// * `/usr/./local/bin/.. =>
/usr/local`
    /// * `/../usr/local/bin   =>
/usr/local/bin`
    /// * `../usr/local/../bin =>
../usr/bin`
    public mutating func
```

```
lexicallyNormalize()

    /// Returns a copy of `self` in
lexical-normal form, that is `.` and `..`
    /// components have been collapsed
lexically (i.e. without following
    /// symlinks). See
`lexicallyNormalize`
    public func lexicallyNormalized() ->
FilePath

    /// Create a new `FilePath` by
resolving `subpath` relative to `self`,
    /// ensuring that the result is
lexically contained within `self`.
    ///
    /// `subpath` will be lexically
normalized (see `lexicallyNormalize`) as
    /// part of resolution, meaning any
contained `.` and `..` components will
    /// be collapsed without resolving
symlinks. Any root in `subpath` will be
    /// ignored.
    ///
    /// Returns `nil` if the result would
"escape" from `self` through use of
    /// the special directory component
`..`.
    ///
    /// This is useful for protecting
against arbitrary path traversal from an
    /// untrusted subpath: the result is
guaranteed to be lexically contained
```

```
    /// within `self`. Since this
operation does not consult the file
system to
    /// resolve symlinks, any escaping
symlinks nested inside of `self` can
still
    /// be targeted by the result.
    ///
    /// Example:
    ///
    ///     let staticContent: FilePath =
"/var/www/my-website/static"
    ///     let links: [FilePath] =
    ///       ["index.html",
"/assets/main.css",
"../../../../etc/passwd"]
    ///     links.map
{ staticContent.lexicallyResolving($0) }
    ///       //
["/var/www/my-website/static/index.html",
    ///       //
"/var/www/my-website/static/assets/main.c
ss",
    ///       //  nil]
    public func lexicallyResolving(_
subpath: FilePath) -> FilePath?
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath {

    /// If `prefix` is a prefix of
```

```
`self`, removes it and returns `true`.
    /// Otherwise returns `false`.
    ///
    /// Example:
    ///
    ///     var path: FilePath =
"/usr/local/bin"
    ///     path.removePrefix("/usr/bin")
// false
    ///     path.removePrefix("/us")
// false
    ///
path.removePrefix("/usr/local") // true,
path is "bin"
    public mutating func removePrefix(_
prefix: FilePath) -> Bool

    /// Append a `component` on to the
end of this path.
    ///
    /// Example:
    ///
    ///     var path: FilePath = "/tmp"
    ///     let sub: FilePath =
"foo/./bar/../baz/."
    ///     for comp in
sub.components.filter({ $0.kind !
= .currentDirectory }) {
    ///       path.append(comp)
    ///     }
    ///     // path is
"/tmp/foo/bar/../baz"
    public mutating func append(_
component: FilePath.Component)

    /// Append `components` on to the end
of this path.
    ///
    /// Example:
    ///
    ///     var path: FilePath = "/"
    ///     path.append(["usr", "local"])
// path is "/usr/local"
    ///     let otherPath: FilePath =
"/bin/ls"
    ///
path.append(otherPath.components) // path
is "/usr/local/bin/ls"
    public mutating func append<C>(_
components: C) where C : Collection,
C.Element == FilePath.Component

    /// Append the contents of `other`,
ignoring any spurious leading separators.
    ///
    /// A leading separator is spurious
if `self` is non-empty.
    ///
    /// Example:
    ///
    ///     var path: FilePath = ""
    ///
path.append("/var/www/website") //
"/var/www/website"
    ///
path.append("static/assets") //
```

```
"/var/www/website/static/assets"
    ///     path.append("/main.css") //
"/var/www/website/static/assets/main.css"
    public mutating func append(_ other:
String)

    /// Non-mutating version of
`append(_:Component)`.
    public func appending(_ other:
FilePath.Component) -> FilePath

    /// Non-mutating version of
`append(_:C)`.
    public func appending<C>(_
components: C) -> FilePath where C :
Collection, C.Element ==
FilePath.Component

    /// Non-mutating version of
`append(_:String)`.
    public func appending(_ other:
String) -> FilePath

    /// If `other` does not have a root,
append each component of `other`. If
    /// `other` has a root, replaces
`self` with other.
    ///
    /// This operation mimics traversing
a directory structure (similar to the
    /// `cd` command), where pushing a
relative path will append its components
    /// and pushing an absolute path will
```

```
first clear `self`'s existing
    /// components.
    ///
    /// Example:
    ///
    ///     var path: FilePath = "/tmp"
    ///     path.push("dir/file.txt") //
path is "/tmp/dir/file.txt"
    ///     path.push("/bin")         //
path is "/bin"
    public mutating func push(_ other:
FilePath)

    /// Non-mutating version of `push()`.
    public func pushing(_ other:
FilePath) -> FilePath

    /// Remove the contents of the path,
keeping the null terminator.
    public mutating func
removeAll(keepingCapacity: Bool = false)

    /// Reserve enough storage space to
store `minimumCapacity` platform
    /// characters.
    public mutating func
reserveCapacity(_ minimumCapacity: Int)
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath {
    /// A bidirectional, range
replaceable collection of the non-root
components
    /// that make up a file path.
    ///
    /// ComponentView provides access to
standard `BidirectionalCollection`
    /// algorithms for accessing
components from the front or back, as
well as
    /// standard
`RangeReplaceableCollection` algorithms
for modifying the
    /// file path using component or
range of components granularity.
    ///
    /// Example:
    ///
    ///     var path: FilePath =
"/./home/./username/scripts/./tree"
    ///     let scriptIdx =
path.components.lastIndex(of: "scripts")!
    ///     path.components.insert("bin",
at: scriptIdx)
    ///     // path is
"/./home/./username/bin/scripts/./tree"
    ///
    ///     path.components.removeAll
{ $0.kind == .currentDirectory }
    ///     // path is
"/home/username/bin/scripts/tree"
    @available(macOS 12.0, iOS 15.0,
watchOS 8.0, tvOS 15.0, *)
```

```
    public struct ComponentView :
Sendable {
    }

    /// View the non-root components that
make up this path.
    public var components:
FilePath.ComponentView
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath {

    /// Create a file path from a root
and a collection of components.
    public init<C>(root: FilePath.Root?,
_ components: C) where C : Collection,
C.Element == FilePath.Component

    /// Create a file path from a root
and any number of components.
    public init(root: FilePath.Root?,
components: FilePath.Component...)

    /// Create a file path from an
optional root and a slice of another
path's
    /// components.
    public init(root: FilePath.Root?, _
components:
FilePath.ComponentView.SubSequence)
}
```

```
@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath {

    /// Creates a file path by copying
bytes from a null-terminated platform
    /// string.
    ///
    /// - Parameter platformString: A
pointer to a null-terminated platform
    ///   string.
    public init(platformString:
UnsafePointer<CInterop.PlatformChar>)

    /// Creates a file path by copying
bytes from a null-terminated platform
    /// string.
    ///
    /// - Note It is a precondition that
`platformString` must be null-terminated.
    /// The absence of a null byte will
trigger a runtime error.
    ///
    /// - Parameter platformString: A
null-terminated platform string.
    @inlinable public
init(platformString:
[CInterop.PlatformChar])

    @available(*, deprecated, message:
"Use FilePath.init(_ scalar:
Unicode.Scalar)")
```

```
    @inlinable public
init(platformString: inout
CInterop.PlatformChar)

    @available(*, deprecated, message:
"Use FilePath(_: String) to create a path
from a String")
    @inlinable public
init(platformString: String)

    /// Calls the given closure with a
pointer to the contents of the file path,
    /// represented as a null-terminated
platform string.
    ///
    /// - Parameter body: A closure with
a pointer parameter
    ///   that points to a null-
terminated platform string.
    ///   If `body` has a return value,
    ///   that value is also used as the
return value for this method.
    /// - Returns: The return value, if
any, of the `body` closure parameter.
    ///
    /// The pointer passed as an argument
to `body` is valid
    /// only during the execution of this
method.
    /// Don't try to store the pointer
for later use.
    public func
withPlatformString<Result>(_ body:
```

```
(UnsafePointer<CInterop.PlatformChar>)
throws -> Result) rethrows -> Result
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FilePath :
ExpressibleByStringLiteral {

    /// Creates a file path from a string
literal.
    ///
    /// - Parameter stringLiteral: A
string literal
    ///   whose Unicode encoded contents
to use as the contents of the path.
    public init(stringLiteral: String)

    /// Creates a file path from a
string.
    ///
    /// - Parameter string: A string
    ///   whose Unicode encoded contents
to use as the contents of the path.
    public init(_ string: String)

    /// A type that represents an
extended grapheme cluster literal.
    ///
    /// Valid types for
`ExtendedGraphemeClusterLiteralType` are
`Character`,
    /// `String`, and `StaticString`.
    @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
    public typealias
ExtendedGraphemeClusterLiteralType =
String

    /// A type that represents a string
literal.
    ///
    /// Valid types for
`StringLiteralType` are `String` and
`StaticString`.
    @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
    public typealias StringLiteralType =
String

    /// A type that represents a Unicode
scalar literal.
    ///
    /// Valid types for
`UnicodeScalarLiteralType` are
`Unicode.Scalar`,
    /// `Character`, `String`, and
`StaticString`.
    @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
    public typealias
UnicodeScalarLiteralType = String
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
```

```
extension FilePath :
CustomStringConvertible,
CustomDebugStringConvertible {

    /// A textual representation of the
file path.
    ///
    /// If the content of the path isn't
a well-formed Unicode string,
    /// this replaces invalid bytes with
U+FFFD. See `String.init(decoding:)`
    public var description: String {
get }

    /// A textual representation of the
file path, suitable for debugging.
    ///
    /// If the content of the path isn't
a well-formed Unicode string,
    /// this replaces invalid bytes with
U+FFFD. See `String.init(decoding:)`
    public var debugDescription: String {
get }
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath {

    /// Creates a string by interpreting
the path’s content as UTF-8 on Unix
    /// and UTF-16 on Windows.
    ///
    /// This property is equivalent to
calling `String(decoding: path)`
    public var string: String { get }
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FilePath {

    /// For backwards compatibility only.
This initializer is equivalent to
    /// the preferred
`FilePath(platformString:)`.
    @available(macOS, introduced: 11.0,
deprecated: 12.0, renamed:
"init(platformString:)")
    @available(iOS, introduced: 14.0,
deprecated: 15.0, renamed:
"init(platformString:)")
    @available(watchOS, introduced: 7.0,
deprecated: 8.0, renamed:
"init(platformString:)")
    @available(tvOS, introduced: 14.0,
deprecated: 15.0, renamed:
"init(platformString:)")
    @available(visionOS, unavailable,
renamed: "init(platformString:)")
    @available(*, deprecated, renamed:
"init(platformString:)")
    public init(cString:
UnsafePointer<CChar>)

    @backDeployed(before: macOS 13.0, iOS
16.0, watchOS 9.0, tvOS 16.0)
    @available(*, deprecated, renamed:
"init(platformString:)")
    @available(visionOS, unavailable,
renamed: "init(platformString:)")
    public init(cString: [CChar])

    @backDeployed(before: macOS 13.0, iOS
16.0, watchOS 9.0, tvOS 16.0)
    @available(*, deprecated, renamed:
"init(platformString:)")
    @available(visionOS, unavailable,
renamed: "init(platformString:)")
    public init(cString: inout CChar)

    @backDeployed(before: macOS 13.0, iOS
16.0, watchOS 9.0, tvOS 16.0)
    @available(*, deprecated, renamed:
"init(platformString:)")
    @available(visionOS, unavailable,
renamed: "init(platformString:)")
    public init(cString: String)

    /// For backwards compatibility only.
This function is equivalent to
    /// the preferred
`withPlatformString`.
    public func withCString<Result>(_
body: (UnsafePointer<CChar>) throws ->
Result) rethrows -> Result
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FilePath {

    /// The length of the file path,
excluding the null terminator.
    public var length: Int { get }
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FilePath : Hashable, Codable {

    /// Creates a new instance by
decoding from the given decoder.
    ///
    /// This initializer throws an error
if reading from the decoder fails, or
    /// if the data read is corrupted or
otherwise invalid.
    ///
    /// - Parameter decoder: The decoder
to read data from.
    public init(from decoder: any
Decoder) throws

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
```

```
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// 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 == (a: FilePath,
b: FilePath) -> Bool
```

```
    /// Encodes this value into the given
encoder.
    ///
    /// If the value fails to encode
anything, `encoder` will encode an empty
    /// keyed container in its place.
    ///
    /// This function throws an error if
any values are invalid for the given
    /// encoder's format.
    ///
    /// - Parameter encoder: The encoder
to write data to.
    public func encode(to encoder: any
Encoder) throws

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
```

```
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Root {

    /// Encodes this value into the given
encoder.
    ///
    /// If the value fails to encode
anything, `encoder` will encode an empty
    /// keyed container in its place.
    ///
    /// This function throws an error if
any values are invalid for the given
    /// encoder's format.
    ///
    /// - Parameter encoder: The encoder
to write data to.
    public func encode(to encoder: any
Encoder) throws

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }

    /// Creates a new instance by
decoding from the given decoder.
    ///
    /// This initializer throws an error
if reading from the decoder fails, or
    /// if the data read is corrupted or
otherwise invalid.
    ///
    /// - Parameter decoder: The decoder
to read data from.
    public init(from decoder: any
Decoder) throws
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Root {

    /// Creates a file path root by
copying bytes from a null-terminated
platform
    /// string.
    ///
    /// Returns `nil` if `platformString`
is empty or is not a root.
    ///
    /// - Parameter platformString: A
```

```
pointer to a null-terminated platform
    ///   string.
    public init?(platformString:
UnsafePointer<CInterop.PlatformChar>)

    /// Creates a file path root by
copying bytes from a null-terminated
platform
    /// string. It is a precondition that
a null byte indicates the end of
    /// the string. The absence of a null
byte will trigger a runtime error.
    ///
    /// Returns `nil` if `platformString`
is empty or is not a root.
    ///
    /// - Note It is a precondition that
`platformString` must be null-terminated.
    /// The absence of a null byte will
trigger a runtime error.
    ///
    /// - Parameter platformString: A
null-terminated platform string.
    @inlinable public init?
(platformString: [CInterop.PlatformChar])

    @available(*, deprecated, message:
"Use FilePath.Root.init(_ scalar:
Unicode.Scalar)")
    @inlinable public init?
(platformString: inout
CInterop.PlatformChar)
```

```
    @available(*, deprecated, message:
"Use FilePath.Root.init(_: String)")
    @inlinable public init?
(platformString: String)

    /// Calls the given closure with a
pointer to the contents of the file path
    /// root, represented as a null-
terminated platform string.
    ///
    /// If the path has a relative
portion, an allocation will occur in
order to
    /// add the null terminator.
    ///
    /// - Parameter body: A closure with
a pointer parameter
    ///   that points to a null-
terminated platform string.
    ///   If `body` has a return value,
    ///   that value is also used as the
return value for this method.
    /// - Returns: The return value, if
any, of the `body` closure parameter.
    ///
    /// The pointer passed as an argument
to `body` is valid
    /// only during the execution of this
method.
    /// Don't try to store the pointer
for later use.
    public func
withPlatformString<Result>(_ body:
```

```
(UnsafePointer<CInterop.PlatformChar>)
throws -> Result) rethrows -> Result
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Root :
ExpressibleByStringLiteral {

    /// Create a file path root from a
string literal.
    ///
    /// Precondition: `stringLiteral` is
non-empty and is a root.
    public init(stringLiteral: String)

    /// Create a file path root from a
string.
    ///
    /// Returns `nil` if `string` is
empty or is not a root.
    public init?(_ string: String)

    /// A type that represents an
extended grapheme cluster literal.
    ///
    /// Valid types for
`ExtendedGraphemeClusterLiteralType` are
`Character`,
    /// `String`, and `StaticString`.
    @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
    public typealias
```

```
ExtendedGraphemeClusterLiteralType =
String

    /// A type that represents a string
literal.
    ///
    /// Valid types for
`StringLiteralType` are `String` and
`StaticString`.
    @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
    public typealias StringLiteralType =
String

    /// A type that represents a Unicode
scalar literal.
    ///
    /// Valid types for
`UnicodeScalarLiteralType` are
`Unicode.Scalar`,
    /// `Character`, `String`, and
`StaticString`.
    @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
    public typealias
UnicodeScalarLiteralType = String
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Root :
CustomStringConvertible,
CustomDebugStringConvertible {
    /// A textual representation of the
path root.
    ///
    /// If the content of the path root
isn't a well-formed Unicode string,
    /// this replaces invalid bytes with
U+FFFD. See `String.init(decoding:)`.
    public var description: String {
get }

    /// A textual representation of the
path root, suitable for debugging.
    ///
    /// If the content of the path root
isn't a well-formed Unicode string,
    /// this replaces invalid bytes with
U+FFFD. See `String.init(decoding:)`.
    public var debugDescription: String {
get }
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Root {

    /// On Unix, this returns `"/"`.
    ///
    /// On Windows, interprets the root's
content as UTF-16 on Windows.
    ///
    /// This property is equivalent to
calling `String(decoding: root)`.
```

```
    public var string: String { get }
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Root : Encodable {
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Root : Decodable {
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Root : Hashable {
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component {

    /// The extension of this file or
directory component.
    ///
    /// If `self` does not contain a `.`
anywhere, or only
    /// at the start, returns `nil`.
Otherwise, returns everything after the
dot.
    ///
    /// Examples:
    ///   * `foo.txt    => txt`
    ///   * `foo.tar.gz => gz`
    ///   * `Foo.app    => app`
    ///   * `.hidden    => nil`
    ///   * `..         => nil`
    public var `extension`: String? { get
}

    /// The non-extension portion of this
file or directory  component.
    ///
    /// Examples:
    ///   * `foo.txt => foo`
    ///   * `foo.tar.gz => foo.tar`
    ///   * `Foo.app => Foo`
    ///   * `.hidden => .hidden`
    ///   * `..      => ..`
    public var stem: String { get }
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component {

    /// Whether a component is a regular
file or directory name, or a special
    /// directory `.` or `..`
    @available(macOS 12.0, iOS 15.0,
watchOS 8.0, tvOS 15.0, *)
    @frozen public enum Kind : Sendable {

        /// The special directory `.`,
representing the current directory.
        case currentDirectory
        /// The special directory `..`,
representing the parent directory.
        case parentDirectory

        /// A file or directory name
        case regular

        /// 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 == (a:
FilePath.Component.Kind, b:
FilePath.Component.Kind) -> Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
```

```
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
```

```
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    /// The kind of this component
    public var kind:
FilePath.Component.Kind { get }
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component {

    /// Encodes this value into the given
encoder.
    ///
    /// If the value fails to encode
anything, `encoder` will encode an empty
    /// keyed container in its place.
    ///
    /// This function throws an error if
any values are invalid for the given
    /// encoder's format.
    ///
    /// - Parameter encoder: The encoder
to write data to.
    public func encode(to encoder: any
Encoder) throws
```

```
    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }

    /// Creates a new instance by
decoding from the given decoder.
    ///
    /// This initializer throws an error
if reading from the decoder fails, or
    /// if the data read is corrupted or
otherwise invalid.
    ///
    /// - Parameter decoder: The decoder
to read data from.
    public init(from decoder: any
Decoder) throws
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component {

    /// Creates a file path component by
copying bytes from a null-terminated
    /// platform string.
    ///
    /// Returns `nil` if `platformString`
is empty, is a root, or has more than
    /// one component in it.
    ///
    /// - Parameter platformString: A
pointer to a null-terminated platform
    ///   string.
    public init?(platformString:
UnsafePointer<CInterop.PlatformChar>)

    /// Creates a file path component by
copying bytes from a null-terminated
    /// platform string. It is a
precondition that a null byte indicates
the end of
    /// the string. The absence of a null
byte will trigger a runtime error.
    ///
    /// Returns `nil` if `platformString`
is empty, is a root, or has more than
    /// one component in it.
    ///
    /// - Note It is a precondition that
`platformString` must be null-terminated.
    /// The absence of a null byte will
trigger a runtime error.
```

```
    ///
    /// - Parameter platformString: A
null-terminated platform string.
    @inlinable public init?
(platformString: [CInterop.PlatformChar])

    @available(*, deprecated, message:
"Use FilePath.Component.init(_ scalar:
Unicode.Scalar)")
    @inlinable public init?
(platformString: inout
CInterop.PlatformChar)

    @available(*, deprecated, message:
"Use FilePath.Component.init(_: String)")
    @inlinable public init?
(platformString: String)

    /// Calls the given closure with a
pointer to the contents of the file path
    /// component, represented as a null-
terminated platform string.
    ///
    /// If this is not the last component
of a path, an allocation will occur in
    /// order to add the null terminator.
    ///
    /// - Parameter body: A closure with
a pointer parameter
    ///   that points to a null-
terminated platform string.
    ///   If `body` has a return value,
    ///   that value is also used as the
```

```
return value for this method.
    /// - Returns: The return value, if
any, of the `body` closure parameter.
    ///
    /// The pointer passed as an argument
to `body` is valid
    /// only during the execution of this
method.
    /// Don't try to store the pointer
for later use.
    public func
withPlatformString<Result>(_ body:
(UnsafePointer<CInterop.PlatformChar>)
throws -> Result) rethrows -> Result
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component :
ExpressibleByStringLiteral {

    /// Create a file path component from
a string literal.
    ///
    /// Precondition: `stringLiteral` is
non-empty, is not a root,
    /// and has only one component in it.
    public init(stringLiteral: String)

    /// Create a file path component from
a string.
    ///
    /// Returns `nil` if `string` is
```

```
empty, a root, or has more than one
component
    /// in it.
    public init?(_ string: String)

    /// A type that represents an
extended grapheme cluster literal.
    ///
    /// Valid types for
`ExtendedGraphemeClusterLiteralType` are
`Character`,
    /// `String`, and `StaticString`.
    @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
    public typealias
ExtendedGraphemeClusterLiteralType =
String

    /// A type that represents a string
literal.
    ///
    /// Valid types for
`StringLiteralType` are `String` and
`StaticString`.
    @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
    public typealias StringLiteralType =
String

    /// A type that represents a Unicode
scalar literal.
    ///
    /// Valid types for
```

```
`UnicodeScalarLiteralType` are
`Unicode.Scalar`,
    /// `Character`, `String`, and
`StaticString`.
    @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
    public typealias
UnicodeScalarLiteralType = String
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component :
CustomStringConvertible,
CustomDebugStringConvertible {

    /// A textual representation of the
path component.
    ///
    /// If the content of the path
component isn't a well-formed Unicode
string,
    /// this replaces invalid bytes with
U+FFFD. See `String.init(decoding:)`.
    public var description: String {
get }

    /// A textual representation of the
path component, suitable for debugging.
    ///
    /// If the content of the path
component isn't a well-formed Unicode
string,
    /// this replaces invalid bytes with
U+FFFD. See `String.init(decoding:)`.
    public var debugDescription: String {
get }
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component {

    /// Creates a string by interpreting
the component’s content as UTF-8 on Unix
    /// and UTF-16 on Windows.
    ///
    /// This property is equivalent to
calling `String(decoding: component)`.
    public var string: String { get }
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component : Encodable
{
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component : Decodable
{
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
```

```
extension FilePath.Component : Hashable {
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.ComponentView :
BidirectionalCollection {

    /// A type representing the
sequence's elements.
    public typealias Element =
FilePath.Component

    /// A type that represents a position
in the collection.
    ///
    /// Valid indices consist of the
position of every element and a
    /// "past the end" position that's
not valid for use as a subscript
    /// argument.
    @available(macOS 12.0, iOS 15.0,
watchOS 8.0, tvOS 15.0, *)
    public struct Index : Sendable,
Comparable, Hashable {

        /// Returns a Boolean value
indicating whether the value of the first
        /// argument is less than that of
the second argument.
        ///
        /// This function is the only
requirement of the `Comparable` protocol.
```

```
The
        /// remainder of the relational
operator functions are implemented by the
        /// standard library for any type
that conforms to `Comparable`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func < (lhs:
FilePath.ComponentView.Index, rhs:
FilePath.ComponentView.Index) -> Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// 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 == (a:
FilePath.ComponentView.Index, b:
FilePath.ComponentView.Index) -> Bool

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
```

```
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    /// The position of the first element
in a nonempty collection.
    ///
    /// If the collection is empty,
`startIndex` is equal to `endIndex`.
    public var startIndex:
FilePath.ComponentView.Index { get }

    /// The collection's "past the end"
position---that is, the position one
    /// greater than the last valid
subscript argument.
    ///
    /// When you need a range that
includes the last element of a
collection, use
    /// the half-open range operator
(`..<`) with `endIndex`. The `..<`
```

```
operator
    /// creates a range that doesn't
include the upper bound, so it's always
    /// safe to use with `endIndex`. For
example:
    ///
    ///     let numbers = [10, 20, 30,
40, 50]
    ///     if let index =
numbers.firstIndex(of: 30) {
    ///         print(numbers[index ..<
numbers.endIndex])
    ///     }
    ///     // Prints "[30, 40, 50]"
    ///
    /// If the collection is empty,
`endIndex` is equal to `startIndex`.
    public var endIndex:
FilePath.ComponentView.Index { get }

    /// Returns the position immediately
after the given index.
    ///
    /// The successor of an index must be
well defined. For an index `i` into a
    /// collection `c`, calling
`c.index(after: i)` returns the same
index every
    /// time.
    ///
    /// - Parameter i: A valid index of
the collection. `i` must be less than
    ///   `endIndex`.
    /// - Returns: The index value
immediately after `i`.
    public func index(after i:
FilePath.ComponentView.Index) ->
FilePath.ComponentView.Index

    /// Returns the position immediately
before the given index.
    ///
    /// - Parameter i: A valid index of
the collection. `i` must be greater than
    ///   `startIndex`.
    /// - Returns: The index value
immediately before `i`.
    public func index(before i:
FilePath.ComponentView.Index) ->
FilePath.ComponentView.Index

    /// Accesses the element at the
specified position.
    ///
    /// The following example accesses an
element of an array through its
    /// subscript to print its value:
    ///
    ///     var streets = ["Adams",
"Bryant", "Channing", "Douglas",
"Evarts"]
    ///     print(streets[1])
    ///     // Prints "Bryant"
    ///
    /// You can subscript a collection
with any valid index other than the
```

```
    /// collection's end index. The end
index refers to the position one past
    /// the last element of a collection,
so it doesn't correspond with an
    /// element.
    ///
    /// - Parameter position: The
position of the element to access.
`position`
    ///   must be a valid index of the
collection that is not equal to the
    ///   `endIndex` property.
    ///
    /// - Complexity: O(1)
    public subscript(position:
FilePath.ComponentView.Index) ->
FilePath.Component { get }

    /// A type that represents the
indices that are valid for subscripting
the
    /// collection, in ascending order.
    @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
    public typealias Indices =
DefaultIndices<FilePath.ComponentView>

    /// A type that provides the
collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms
to the `Sequence` protocol by
```

```
    /// supplying `IndexingIterator` as
its associated `Iterator`
    /// type.
    @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
    public typealias Iterator =
IndexingIterator<FilePath.ComponentView>

    /// A collection representing a
contiguous subrange of this collection's
    /// elements. The subsequence shares
indices with the original collection.
    ///
    /// The default subsequence type for
collections that don't define their own
    /// is `Slice`.
    @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
    public typealias SubSequence =
Slice<FilePath.ComponentView>
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.ComponentView :
RangeReplaceableCollection {

    /// Creates a new, empty collection.
    public init()

    /// Replaces the specified subrange
of elements with the given collection.
    ///
    /// This method has the effect of
removing the specified range of elements
    /// from the collection and inserting
the new elements at the same location.
    /// The number of new elements need
not match the number of elements being
    /// removed.
    ///
    /// In this example, three elements
in the middle of an array of integers are
    /// replaced by the five elements of
a `Repeated<Int>` instance.
    ///
    ///      var nums = [10, 20, 30, 40,
50]
    ///      nums.replaceSubrange(1...3,
with: repeatElement(1, count: 5))
    ///      print(nums)
    ///      // Prints "[10, 1, 1, 1, 1,
1, 50]"
    ///
    /// If you pass a zero-length range
as the `subrange` parameter, this method
    /// inserts the elements of
`newElements` at `subrange.startIndex`.
Calling
    /// the `insert(contentsOf:at:)`
method instead is preferred.
    ///
    /// Likewise, if you pass a zero-
length collection as the `newElements`
    /// parameter, this method removes
the elements in the given subrange
```

```
    /// without replacement. Calling the
`removeSubrange(_:)` method instead is
    /// preferred.
    ///
    /// Calling this method may
invalidate any existing indices for use
with this
    /// collection.
    ///
    /// - Parameters:
    ///   - subrange: The subrange of the
collection to replace. The bounds of
    ///     the range must be valid
indices of the collection.
    ///   - newElements: The new elements
to add to the collection.
    ///
    /// - Complexity: O(*n* + *m*), where
*n* is length of this collection and
    ///   *m* is the length of
`newElements`. If the call to this method
simply
    ///   appends the contents of
`newElements` to the collection, this
method is
    ///   equivalent to
`append(contentsOf:)`.
    public mutating func
replaceSubrange<C>(_ subrange:
Range<FilePath.ComponentView.Index>, with
newElements: C) where C : Collection,
C.Element == FilePath.Component
}
```

```
@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.ComponentView {

    /// Encodes this value into the given
encoder.
    ///
    /// If the value fails to encode
anything, `encoder` will encode an empty
    /// keyed container in its place.
    ///
    /// This function throws an error if
any values are invalid for the given
    /// encoder's format.
    ///
    /// - Parameter encoder: The encoder
to write data to.
    public func encode(to encoder: any
Encoder) throws

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }

    /// Creates a new instance by
decoding from the given decoder.
    ///
    /// This initializer throws an error
if reading from the decoder fails, or
    /// if the data read is corrupted or
otherwise invalid.
    ///
    /// - Parameter decoder: The decoder
to read data from.
    public init(from decoder: any
Decoder) throws
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.ComponentView :
Encodable {
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.ComponentView :
Decodable {
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
```

```
extension FilePath.ComponentView :
Hashable {
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component.Kind :
Equatable {
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component.Kind :
Hashable {
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension FilePath.Component.Kind :
BitwiseCopyable {
}

/// The access permissions for a file.
///
/// The following example
/// creates an instance of the
`FilePermissions` structure
/// from a raw octal literal and compares
it
/// to a file permission created using
named options:
///
///     let perms =
FilePermissions(rawValue: 0o644)
///     perms ==
[.ownerReadWrite, .groupRead, .otherRead]
// true
@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
@frozen public struct FilePermissions :
OptionSet, Sendable, Hashable, Codable {

    /// The raw C file permissions.
    public let rawValue: CModeT

    /// Create a strongly-typed file
permission from a raw C value.
    public init(rawValue: CModeT)

    /// Indicates that other users have
read-only permission.
    public static var otherRead:
FilePermissions { get }

    /// Indicates that other users have
write-only permission.
    public static var otherWrite:
FilePermissions { get }

    /// Indicates that other users have
execute-only permission.
    public static var otherExecute:
FilePermissions { get }

    /// Indicates that other users have
read-write permission.
```

```
    public static var otherReadWrite:
FilePermissions { get }

    /// Indicates that other users have
read-execute permission.
    public static var otherReadExecute:
FilePermissions { get }

    /// Indicates that other users have
write-execute permission.
    public static var otherWriteExecute:
FilePermissions { get }

    /// Indicates that other users have
read, write, and execute permission.
    public static var
otherReadWriteExecute: FilePermissions {
get }

    /// Indicates that the group has
read-only permission.
    public static var groupRead:
FilePermissions { get }

    /// Indicates that the group has
write-only permission.
    public static var groupWrite:
FilePermissions { get }

    /// Indicates that the group has
execute-only permission.
    public static var groupExecute:
FilePermissions { get }
```

```
    /// Indicates that the group has
read-write permission.
    public static var groupReadWrite:
FilePermissions { get }

    /// Indicates that the group has
read-execute permission.
    public static var groupReadExecute:
FilePermissions { get }

    /// Indicates that the group has
write-execute permission.
    public static var groupWriteExecute:
FilePermissions { get }

    /// Indicates that the group has
read, write, and execute permission.
    public static var
groupReadWriteExecute: FilePermissions {
get }

    /// Indicates that the owner has
read-only permission.
    public static var ownerRead:
FilePermissions { get }

    /// Indicates that the owner has
write-only permission.
    public static var ownerWrite:
FilePermissions { get }

    /// Indicates that the owner has
```

```
execute-only permission.
    public static var ownerExecute:
FilePermissions { get }

    /// Indicates that the owner has
read-write permission.
    public static var ownerReadWrite:
FilePermissions { get }

    /// Indicates that the owner has
read-execute permission.
    public static var ownerReadExecute:
FilePermissions { get }

    /// Indicates that the owner has
write-execute permission.
    public static var ownerWriteExecute:
FilePermissions { get }

    /// Indicates that the owner has
read, write, and execute permission.
    public static var
ownerReadWriteExecute: FilePermissions {
get }

    /// Indicates that the file is
executed as the owner.
    ///
    /// For more information, see the
`setuid(2)` man page.
    public static var setUserID:
FilePermissions { get }
```

```
    /// Indicates that the file is
executed as the group.
    ///
    /// For more information, see the
`setgid(2)` man page.
    public static var setGroupID:
FilePermissions { get }

    /// Indicates that executable's text
segment
    /// should be kept in swap space even
after it exits.
    ///
    /// For more information, see the
`chmod(2)` man page's
    /// discussion of `S_ISVTX` (the
sticky bit).
    public static var saveText:
FilePermissions { get }

    /// The type of the elements of an
array literal.
    @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
    public typealias ArrayLiteralElement
= FilePermissions

    /// The element type of the option
set.
    ///
    /// To inherit all the default
implementations from the `OptionSet`
protocol,
    /// the `Element` type must be
`Self`, the default.
    @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
    public typealias Element =
FilePermissions

    /// 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 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
    public typealias RawValue = CModeT
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FilePermissions :
CustomStringConvertible,
CustomDebugStringConvertible {

    /// A textual representation of the
file permissions.
    public var description: String {
```

```
get }

    /// A textual representation of the
file permissions, suitable for debugging.
    public var debugDescription: String {
get }
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension FilePermissions :
BitwiseCopyable {
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
@frozen public enum Mach {

    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    public struct Port<RightType> :
~Copyable where RightType : MachPortRight
{

        /// Transfer ownership of an
existing unmanaged Mach port right into a
        /// `Mach.Port` by name.
        ///
        /// This initializer traps if
`name` is `MACH_PORT_NULL`, or if `name`
is
        /// `MACH_PORT_DEAD` and the
`RightType` is `Mach.ReceiveRight`.
```

```
        ///
        /// If the type of the right does
not match the `RightType` of the
        /// `Mach.Port` being
constructed, behavior is undefined.
        ///
        /// The underlying port right
will be automatically deallocated at the
        /// end of the `Mach.Port`
instance's lifetime.
        ///
        /// This initializer makes a
syscall to guard the right.
        public init(name:
mach_port_name_t)

        /// Borrow access to the port
name in a block that can perform
        /// non-consuming operations.
        ///
        /// Take care when using this
function; many operations consume rights,
        /// and send-once rights are
easily consumed.
        ///
        /// If the right is consumed,
behavior is undefined.
        ///
        /// The body block may optionally
return something, which will then be
        /// returned to the caller of
withBorrowedName.
        @inlinable public func
```

```
withBorrowedName<ReturnType>(body:
(mach_port_name_t) -> ReturnType) ->
ReturnType
    }

    /// Possible errors that can be
thrown by Mach.Port operations.
    public enum PortRightError : Error {

        /// Returned when an operation
cannot be completed, because the Mach
        /// port right has become a dead
name. This is caused by deallocation of
the
        /// receive right on the other
end.
        case deadName

        /// 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 == (a:
Mach.PortRightError, b:
Mach.PortRightError) -> Bool
```

```
        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    /// The MachPortRight type used to
manage a receive right.
    @frozen public struct ReceiveRight :
MachPortRight {
    }

    /// The MachPortRight type used to
manage a send right.
    @frozen public struct SendRight :
MachPortRight {
    }

    /// The MachPortRight type used to
manage a send-once right.
```

```
    ///
    /// Send-once rights are the most
restrictive type of Mach port rights.
    /// They cannot create other rights,
and are consumed upon use.
    ///
    /// Upon destruction a send-once
notification will be sent to the
    /// receiving end.
    @frozen public struct SendOnceRight :
MachPortRight {
    }
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach : Sendable {
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach : BitwiseCopyable {
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.Port where RightType ==
Mach.ReceiveRight {

    /// Transfer ownership of an
existing, unmanaged, but already guarded,
    /// Mach port right into a Mach.Port
by name.
```

```
    ///
    /// This initializer aborts if name
is MACH_PORT_NULL.
    ///
    /// If the type of the right does not
match the type T of Mach.Port<T>
    /// being constructed, the behavior
is undefined.
    ///
    /// The underlying port right will be
automatically deallocated when
    /// the Mach.Port object is
destroyed.
    public init(name: mach_port_name_t,
context: mach_port_context_t)

    /// Allocate a new Mach port with a
receive right, creating a
    /// Mach.Port<Mach.ReceiveRight> to
manage it.
    ///
    /// This initializer will abort if
the right could not be created.
    /// Callers may assert that a valid
right is always returned.
    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    @inlinable public init()

    /// Transfer ownership of the
underlying port right to the caller.
    ///
    /// Returns a tuple containing the
```

```
Mach port name representing the right,
    /// and the context value used to
guard the right.
    ///
    /// This operation liberates the
right from management by the Mach.Port,
    /// and the underlying right will no
longer be automatically deallocated.
    ///
    /// After this function completes,
the Mach.Port is destroyed and no longer
    /// usable.
    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    @inlinable public consuming func
relinquish() -> (name: mach_port_name_t,
context: mach_port_context_t)

    /// Remove guard and transfer
ownership of the underlying port right to
    /// the caller.
    ///
    /// Returns the Mach port name
representing the right.
    ///
    /// This operation liberates the
right from management by the Mach.Port,
    /// and the underlying right will no
longer be automatically deallocated.
    ///
    /// After this function completes,
the Mach.Port is destroyed and no longer
    /// usable.
    ///
    /// This function makes a syscall to
remove the guard from
    /// Mach.ReceiveRights. Use
relinquish() to avoid the syscall and
extract
    /// the context value along with the
port name.
    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    @inlinable public consuming func
unguardAndRelinquish() ->
mach_port_name_t

    /// Borrow access to the port name in
a block that can perform
    /// non-consuming operations.
    ///
    /// Take care when using this
function; many operations consume rights.
    ///
    /// If the right is consumed,
behavior is undefined.
    ///
    /// The body block may optionally
return something, which will then be
    /// returned to the caller of
withBorrowedName.
    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    @inlinable public func
withBorrowedName<ReturnType>(body:
(mach_port_name_t, mach_port_context_t)
-> ReturnType) -> ReturnType

    /// Create a send-once right for a
given receive right.
    ///
    /// This does not affect the
makeSendCount of the receive right.
    ///
    /// This function will abort if the
right could not be created.
    /// Callers may assert that a valid
right is always returned.
    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    @inlinable public func
makeSendOnceRight() ->
Mach.Port<Mach.SendOnceRight>

    /// Create a send right for a given
receive right.
    ///
    /// This increments the makeSendCount
of the receive right.
    ///
    /// This function will abort if the
right could not be created.
    /// Callers may assert that a valid
right is always returned.
    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    @inlinable public func
makeSendRight() ->
Mach.Port<Mach.SendRight>
```

```
    /// Access the make-send count.
    ///
    /// Each get/set of this property
makes a syscall.
    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    @inlinable public var makeSendCount:
mach_port_mscount_t
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.Port where RightType ==
Mach.SendRight {

    /// Transfer ownership of the
underlying port right to the caller.
    ///
    /// Returns the Mach port name
representing the right.
    ///
    /// This operation liberates the
right from management by the Mach.Port,
    /// and the underlying right will no
longer be automatically deallocated.
    ///
    /// After this function completes,
the Mach.Port is destroyed and no longer
    /// usable.
    @inlinable public consuming func
relinquish() -> mach_port_name_t
```

```
    /// Create another send right from a
given send right.
    ///
    /// This does not affect the
makeSendCount of the receive right.
    ///
    /// If the send right being copied
has become a dead name, meaning the
    /// receiving side has been
deallocated, then copySendRight() will
throw
    /// a Mach.PortRightError.deadName
error.
    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    @inlinable public func
copySendRight() throws ->
Mach.Port<Mach.SendRight>
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.Port where RightType ==
Mach.SendOnceRight {

    /// Transfer ownership of the
underlying port right to the caller.
    ///
    /// Returns the Mach port name
representing the right.
    ///
    /// This operation liberates the
right from management by the Mach.Port,
    /// and the underlying right will no
longer be automatically deallocated.
    ///
    /// After this function completes,
the Mach.Port is destroyed and no longer
    /// usable.
    @available(macOS 14.4, iOS 17.4,
watchOS 10.4, tvOS 17.4, *)
    @inlinable public consuming func
relinquish() -> mach_port_name_t
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.PortRightError : Equatable
{
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.PortRightError : Hashable
{
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.ReceiveRight : Sendable {
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.ReceiveRight :
BitwiseCopyable {
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.SendRight : Sendable {
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.SendRight :
BitwiseCopyable {
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.SendOnceRight : Sendable {
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
extension Mach.SendOnceRight :
BitwiseCopyable {
}

@available(macOS 14.4, iOS 17.4, watchOS
10.4, tvOS 17.4, *)
public protocol MachPortRight {
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension String {
    /// Creates a string by interpreting
the null-terminated platform string as
    /// UTF-8 on Unix and UTF-16 on
Windows.
    ///
    /// - Parameter platformString: The
null-terminated platform string to be
    ///  interpreted as
`CInterop.PlatformUnicodeEncoding`.
    ///
    /// If the content of the platform
string isn't well-formed Unicode,
    /// this initializer replaces invalid
bytes with U+FFFD.
    /// This means that, depending on the
semantics of the specific platform,
    /// conversion to a string and back
might result in a value that's different
    /// from the original platform
string.
    public init(platformString:
UnsafePointer<CInterop.PlatformChar>)

    /// Creates a string by interpreting
the null-terminated platform string as
    /// UTF-8 on Unix and UTF-16 on
Windows.
    ///
    /// - Parameter platformString: The
null-terminated platform string to be
    ///  interpreted as
`CInterop.PlatformUnicodeEncoding`.
    ///
    /// - Note It is a precondition that
`platformString` must be null-terminated.
    /// The absence of a null byte will
trigger a runtime error.
    ///
    /// If the content of the platform
string isn't well-formed Unicode,
    /// this initializer replaces invalid
bytes with U+FFFD.
    /// This means that, depending on the
semantics of the specific platform,
    /// conversion to a string and back
might result in a value that's different
    /// from the original platform
string.
    @inlinable public
init(platformString:
[CInterop.PlatformChar])

    @available(*, deprecated, message:
"Use String.init(_ scalar:
Unicode.Scalar)")
    @inlinable public
init(platformString: inout
CInterop.PlatformChar)

    @available(*, deprecated, message:
"Use a copy of the String argument")
    @inlinable public
init(platformString: String)

    /// Creates a string by interpreting
the null-terminated platform string as
```

```
    /// UTF-8 on Unix and UTF-16 on
Windows.
    ///
    /// - Parameter platformString: The
null-terminated platform string to be
    ///  interpreted as
`CInterop.PlatformUnicodeEncoding`.
    ///
    /// If the contents of the platform
string isn't well-formed Unicode,
    /// this initializer returns `nil`.
    public init?(validatingPlatformString
platformString:
UnsafePointer<CInterop.PlatformChar>)

    /// Creates a string by interpreting
the null-terminated platform string as
    /// UTF-8 on Unix and UTF-16 on
Windows.
    ///
    /// - Parameter platformString: The
null-terminated platform string to be
    ///  interpreted as
`CInterop.PlatformUnicodeEncoding`.
    ///
    /// - Note It is a precondition that
`platformString` must be null-terminated.
    /// The absence of a null byte will
trigger a runtime error.
    ///
    /// If the contents of the platform
string isn't well-formed Unicode,
    /// this initializer returns `nil`.
    @inlinable public init?
(validatingPlatformString platformString:
[CInterop.PlatformChar])

    @available(*, deprecated, message:
"Use String(_ scalar: Unicode.Scalar)")
    @inlinable public init?
(validatingPlatformString platformString:
inout CInterop.PlatformChar)

    @available(*, deprecated, message:
"Use a copy of the String argument")
    @inlinable public init?
(validatingPlatformString platformString:
String)

    /// Calls the given closure with a
pointer to the contents of the string,
    /// represented as a null-terminated
platform string.
    ///
    /// - Parameter body: A closure with
a pointer parameter
    ///   that points to a null-
terminated platform string.
    ///   If `body` has a return value,
    ///   that value is also used as the
return value for this method.
    /// - Returns: The return value, if
any, of the `body` closure parameter.
    ///
    /// The pointer passed as an argument
to `body` is valid
```

```
    /// only during the execution of this
method.
    /// Don't try to store the pointer
for later use.
    public func
withPlatformString<Result>(_ body:
(UnsafePointer<CInterop.PlatformChar>)
throws -> Result) rethrows -> Result
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension String {

    /// Creates a string by interpreting
the file path's content as UTF-8 on Unix
    /// and UTF-16 on Windows.
    ///
    /// - Parameter path: The file path
to be interpreted as
    ///
`CInterop.PlatformUnicodeEncoding`.
    ///
    /// If the content of the file path
isn't a well-formed Unicode string,
    /// this initializer replaces invalid
bytes with U+FFFD.
    /// This means that, depending on the
semantics of the specific file system,
    /// conversion to a string and back
to a path
    /// might result in a value that's
different from the original path.
```

```
    public init(decoding path: FilePath)
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension String {

    /// Creates a string from a file
path, validating its contents as UTF-8 on
    /// Unix and UTF-16 on Windows.
    ///
    /// - Parameter path: The file path
to be interpreted as
    ///
`CInterop.PlatformUnicodeEncoding`.
    ///
    /// If the contents of the file path
isn't a well-formed Unicode string,
    /// this initializer returns `nil`.
    @available(macOS 12.0, iOS 15.0,
watchOS 8.0, tvOS 15.0, *)
    public init?(validating path:
FilePath)
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension String {

    /// Creates a string by interpreting
the path component's content as UTF-8 on
    /// Unix and UTF-16 on Windows.
    ///
    /// - Parameter component: The path
component to be interpreted as
    ///
`CInterop.PlatformUnicodeEncoding`.
    ///
    /// If the content of the path
component isn't a well-formed Unicode
string,
    /// this initializer replaces invalid
bytes with U+FFFD.
    /// This means that, depending on the
semantics of the specific file system,
    /// conversion to a string and back
to a path component
    /// might result in a value that's
different from the original path
component.
    public init(decoding component:
FilePath.Component)

    /// Creates a string from a path
component, validating its contents as
UTF-8
    /// on Unix and UTF-16 on Windows.
    ///
    /// - Parameter component: The path
component to be interpreted as
    ///
`CInterop.PlatformUnicodeEncoding`.
    ///
    /// If the contents of the path
component isn't a well-formed Unicode
string,
    /// this initializer returns `nil`.
    public init?(validating component:
FilePath.Component)
}

@available(macOS 12.0, iOS 15.0, watchOS
8.0, tvOS 15.0, *)
extension String {

    /// On Unix, creates the string `"/"`
    ///
    /// On Windows, creates a string by
interpreting the path root's content as
    /// UTF-16.
    ///
    /// - Parameter root: The path root
to be interpreted as
    ///
`CInterop.PlatformUnicodeEncoding`.
    ///
    /// If the content of the path root
isn't a well-formed Unicode string,
    /// this initializer replaces invalid
bytes with U+FFFD.
    /// This means that on Windows,
    /// conversion to a string and back
to a path root
    /// might result in a value that's
different from the original path root.
    public init(decoding root:
FilePath.Root)

    /// On Unix, creates the string `"/"`
    ///
    /// On Windows, creates a string from
a path root, validating its contents as
    /// UTF-16 on Windows.
    ///
    /// - Parameter root: The path root
to be interpreted as
    ///
`CInterop.PlatformUnicodeEncoding`.
    ///
    /// On Windows, if the contents of
the path root isn't a well-formed Unicode
    /// string, this initializer returns
`nil`.
    public init?(validating root:
FilePath.Root)
}

@available(macOS 11.0, iOS 14.0, watchOS
7.0, tvOS 14.0, *)
extension String {

    @available(macOS, introduced: 11.0,
deprecated: 12.0, renamed:
"init(decoding:)")
    @available(iOS, introduced: 14.0,
deprecated: 15.0, renamed:
"init(decoding:)")
    @available(watchOS, introduced: 7.0,
deprecated: 8.0, renamed:
"init(decoding:)")
    @available(tvOS, introduced: 14.0,
deprecated: 15.0, renamed:
```

```
"init(decoding:)")
    @available(visionOS, unavailable,
renamed: "init(decoding:)")
    @available(*, deprecated, renamed:
"init(decoding:)")
    public init(_ path: FilePath)

    @available(macOS, introduced: 11.0,
deprecated: 12.0, renamed:
"init(validating:)")
    @available(iOS, introduced: 14.0,
deprecated: 15.0, renamed:
"init(validating:)")
    @available(watchOS, introduced: 7.0,
deprecated: 8.0, renamed:
"init(validating:)")
    @available(tvOS, introduced: 14.0,
deprecated: 15.0, renamed:
"init(validating:)")
    @available(visionOS, unavailable,
renamed: "init(validating:)")
    @available(*, deprecated, renamed:
"init(validating:)")
    public init?(validatingUTF8 path:
FilePath)
}
```