# QSocket: Socket Options

This thread has been locked by a moderator.

34

**IMPORTANT** If you haven't yet read Calling BSD Sockets from Swift, do that first.

Here are some straightforward wrappers for the `getsockopt` and `setsockopt` calls:

```
extension FileDescriptor {

    /// Gets a socket option.
    ///
    /// Equivalent to the `getsockopt` BSD Sockets call.
    ///
    /// For simple socket options, consider using
    /// ``getSocketOption(_:_:as:retryOnInterrupt:)``.

    public func getSocketOption(_ level: CInt, _ name: CInt, _ optionValue: UnsafeMutableRawPointer, optionLen: inout Int,
retryOnInterrupt: Bool = true) throws {
        guard var optionSockLen = socklen_t(exactly: optionLen), optionLen >= 0 else { fatalError() }
        try errnoQ(retryOnInterrupt: retryOnInterrupt) {
            Foundation.getsockopt(self.rawValue, level, name, optionValue, &optionSockLen)
        }
        optionLen = Int(optionSockLen)
    }

    /// Sets a socket option.
    ///
    /// Equivalent to the `setsockopt` BSD Sockets call.
    ///
    /// For simple socket options, consider using
    /// ``setSocketOption(_:_:_:retryOnInterrupt:)``.

    public func setSocketOption(_ level: CInt, _ name: CInt, _ optionValue: UnsafeRawPointer, _ optionLen: Int, retryOnInterrupt:
Bool = true) throws {
        guard let optionSockLen = socklen_t(exactly: optionLen), optionLen >= 0 else { fatalError() }
        try errnoQ(retryOnInterrupt: retryOnInterrupt) {
            Foundation.setsockopt(self.rawValue, level, name, optionValue, optionSockLen)
        }
    }
}
```

These work fine but they're a little primitive. I like adding a layer that makes it easier to work with standard types:

```
extension FileDescriptor {

    /// Gets a simple socket option.
    ///
    /// This allows you to get a simple socket option without messing around
    /// with the unsafe pointer malarkey involved in
    /// ``getSocketOption(_:_:_:optionLen:retryOnInterrupt:)``.  See
    /// `QSocketOptionConvertible` for more about how this works.

    public func getSocketOption<T>(_ level: CInt, _ name: CInt, as: T.Type, retryOnInterrupt: Bool = true) throws -> T
        where T: QSocketOptionConvertible
    {
        var result = T()
        try withUnsafeMutableBytes(of: &result) { buf in
            var bufCount = buf.count
            try self.getSocketOption(level, name, buf.baseAddress!, optionLen: &bufCount, retryOnInterrupt: retryOnInterrupt)
            guard bufCount == buf.count else {
                throw Errno.noBufferSpace
            }
        }
        return result
    }

    /// Sets a simple socket option.
    ///
    /// This allows you to set a simple socket option without messing around
    /// with the unsafe pointer malarkey involved in
    /// ``setSocketOption(_:_:_:_:retryOnInterrupt:)``.  See
    /// ``QSocketOptionConvertible`` for more about how this works.

    public func setSocketOption<T>(_ level: CInt, _ name: CInt, _ optionValue: T, retryOnInterrupt: Bool = true) throws
        where T: QSocketOptionConvertible
    {
        // Can't use `&value` because of a new compiler warning.  We work around
        // that per the [docs][ref].  One day there may be a 'bitwise copyable'
        // protocol that we can add to `QSocketOptionConvertible` to actually
        // expression what's going on here at the type layer.
        //
        // [ref]: <https://github.com/atrick/swift-evolution/blob/diagnose-implicit-raw-bitwise/proposals/nnnn-implicit-raw-
bitwise-conversion.md#workarounds-for-common-cases>
        var value = optionValue
        try withUnsafeBytes(of: &value) { buf in
            try self.setSocketOption(level, name, buf.baseAddress!, buf.count, retryOnInterrupt: retryOnInterrupt)
        }
    }
}

/// Indicates that a type can be used as a socket option.
///
/// This has one true constraint, namely that the type has a default value.
/// There are, however, two implicit constraints:
///
/// * The type must be just data. If, for example, the type contains an object
///   reference, bad things would happen.
///
/// * The type's size is compatible with `socklen_t`.
///
/// We specifically conform various types, like `CInt` and `timeval`, to this
/// protocol but you can add to that list if necessary.

public protocol QSocketOptionConvertible {
    init()
}

extension UInt8: QSocketOptionConvertible { }
extension CInt: QSocketOptionConvertible { }
extension CUnsignedInt: QSocketOptionConvertible { }
extension timeval: QSocketOptionConvertible { }
```

Share and Enjoy

—

Quinn "The Eskimo!" @ Developer Technical Support @ Apple

`let myEmail = "eskimo" + "1" + "@" + "apple.com"`

Network

Reply

Posted 5 days ago by eskimo

Add a Comment





 