1 Asynchronous I/O

1.1 Asynchronous I/O

Asynchronous I/O is a major part of the difficulty in creating a server implemenation in C, particularly when using threads is not a viable option. We don't want the server to be stuck while we wait for it to perform some I/O operation like reading or writing to an existing connection when the server has other tasks it could be doing, like establishing new client connections or parsing an already-received message. In C, one of the primary ways of performing asynchronous I/O is using the `epoll` API. Using this API, we are able to keep a list of file descriptors we want the current process to monitor, as well as a list of file descriptors that are ready for I/O. However, this process is extraordinarily tedious, requiring many expensive system calls to set up and maintain the `epoll` instance. When compared with other, more modern languages, the `epoll` API is both more verbose and more difficult to use. Consider a more modern language like Go or Javascript. Both of these languages have their own facilities for asynchronous operation. In Go, we use several constructs, including the `go` and `select` keywords, to implement various aspects of I/O. In Javascript, we utilize both the `async` and `await` keywords, as well as the idea of *promises* in order to achieve some measure of asynchronous operation. The goal of our extension is to include these same facilities in `ABLE`C, allowing for a programmer to more easily write and understand the code performing asynchronous I/O operations.

1.2 Asynchronous I/O Extension

In this section, we discuss the mechanics of the Asynchronous I/O extension, including the specifics of the translation from `ABLE`C code (`.xc` files) to plain C code.

1.3 Asynchronous I/O Implementation