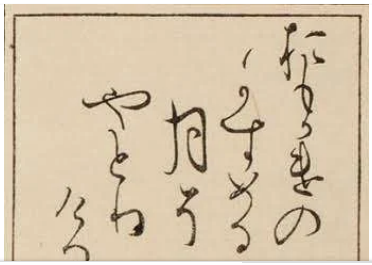
網受解説物語

Webservとは



昔の人

今は昔。平安の時代、俳句や短歌が恋する男女の繊細な思ひを伝える手段として用いられていた。

それはまるで、今日のブラウザとwebサーバーが情報を交換する方法に似ている。

しかし、このデジタルな時代において、それはどのように機能するのか？

webservがクライアントとサーバーの間でどのように通信を可能にするかを解説する。

これはそんな物語。

目次

- **Configuration File (設定ファイル)**
  - つまりどういうこと？
- 全体的な流れ
  - configファイル関係
  - イベント
- 用語説明
  - 課題用語
  - 非ブロックであり、すべてのI/O操作のために1つのpoll()またはその同等のもののみを使用するべき。
  - GET、POST、DELETE
- イベント
- CGI

課題要件

Mandatory Part (必須部分)

- - HTTPサーバをC++ 98で書くこと。
- - サーバは設定ファイルを使用して実行されるべき。
- - さまざまな関数とメソッドがリストされており、それらを実装する必要がある。

Requirements (要件)

- プログラムは引数として設定ファイルを取るべき。

- - プログラムは引数として設定ファイルを取るべき。
- - サーバは絶対にブロックしてはならない。
- - 非ブロックであり、すべてのI/O操作のために1つのpoll()またはその同等のもののみを使用するべき。
- - サーバへのリクエストは永遠にハングアップしてはならない。
- - サーバは選択したウェブブラウザと互換性があるべき。
- - サーバは静的なウェブサイトを提供し、ファイルのアップロードを処理できるべき。
- - 少なくともGET、POST、DELETEメソッドが必要。

Configuration File (設定ファイル)

- - 設定ファイルは、ポート、ホスト、サーバ名、エラーページ、ルートなど、さまざまなサーバパラメータの設定を可能にするべき。

つまりどういうこと？

むずいね…

横文字多すぎて意味わからん。。。そんな人もいるでしょう。（己棚上）
つまり

『HTTPサーバをC＋＋で書いて！！』

ってことです。
まあ、全体の流れに沿って順に説明できればなと思いますので、特に気にせずそのまま、GO！！

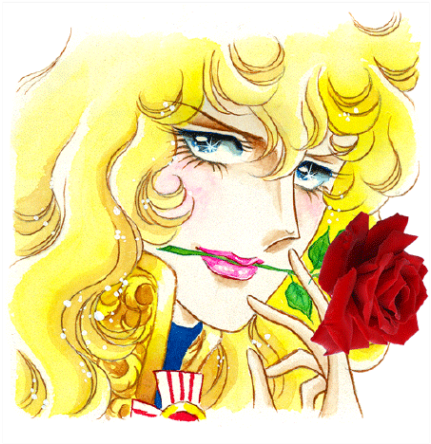
全体的な流れ

以下に大雑把なコードの流れを書きますが、この課題は

『イベント駆動型プログラム』

となっているので、最初から最後まで綺麗な一本道になっているのではなく、while でループし常に何かしらのイベントが起こることを待っている感じなので、少しわかりずらいかもです。

まあ、恋する乙女や中二病の方も日頃から、「王子様来ないかなー」とか「クラスにテロリストが入ってきたら俺がやっつけるのに！」などと妄想して、イベントを待っているので、それと同じです（違うだろ）。



ベルサイユのばら

ではでは流れへ！

configファイル関係

1. 課題に指定された、さまざまなサーバパラメータの設定ができるような、汎用性の高いconfigファイル生成機を作る。
2. できたconfigファイルからサーバの設定や動作を定義する。

イベント

- wait !!
- listen !!
- write !!
- read !!
- connect !!
- requests !!
- cull !!

以上！！

(´ө`)は？

キャプションを入力

お、落ち着いてください！本当にこういう感じなんですよ！

まあ一個ずつやりましょ。まずは用語の説明から(❁´ω`❁)

用語説明

課題用語

- HTTPサーバってそもそも何？

　HTTP は「HyperText Transfer Protocol（ハイパーテキスト・トランスファー・プロトコル）」の略です。

　直訳すると、すごい文　を　転送する　約束事、です。

(´ө`)は？(part2)

　まあ要するに、「ホームページのファイルとかを受け渡しするときに使うお約束事」です。

　詳しくは下記を見ればだいたいわかります。

https://wa3.i-3-i.info/word110370.html

　因みにwebserver と　HTTPserver はほぼ同じです。

　実は皆さんもよくやっている行為なんですよね。
　GoogleとかのwebブラウザにURLを入力して、調べごとしたりしますよね。

https://wa3.i-3-i.info/word110370.html

　と言うかさっき貼ったこれも
http の一つですね。htmlがHyperTextに当たる部分です。（https なのは見逃して）

　そもそもclient とserver があって、
電話で例えると

clientが電話をかける人
serverが電話を受け取る人

って感じです。

今回で言うと、
webブラウザがclient、
webserverがserverです。

webserverがURLを受け取って、
webブラウザがファイルを画面に出す！

こんな感じ

なんかそんなのやっているんだなあと思っていただければ良いと思います。

非ブロックであり、すべてのI/O操作のために1つのpoll()またはその同等のもののみを使用するべき。

(´･ω･`)は？(part3)
知らない方はこの部分で発狂しそうになったのではないでしょうか。

大丈夫です。一つずつやっていきましょう。

- 非ブロックとは？

「非ブロック」とは、プログラムが特定の操作（通常はI/O操作）を待機する際に、その操作が完了するのを待たずに次の処理に進むことを指します。これは、操作が即座に完了しない場合でも、プログラムが停止することなく他のタスクを実行できるようにするためのものです。（by gpt）

(´･ω･`)だからI/O操作がわからんのだが

- I/O操作とは　input output の操作で、データの入力、出力の操作です。

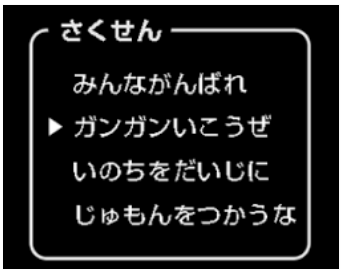
キーボードやマウスからの入力、ccへの出力、ディスクへの読み書き、ネットワークを介したデータの送受信など、さまざまな形態のデータのやり取りが含まれます。

画面に出力

　スマホのアプリのローディングで、時間がかかったことありませんか？
ネットワーク通信やファイルI/Oなどの操作でも同じように時間がかかることがあるのです。

　ブロッキングモードでは、これらの操作が完了するまでプログラムは待機状態となり、他のタスクを実行しない。

　逆に、非ブロッキングモードでは、操作が即座に完了しない場合でも、プログラムは待機せずに次の処理に進みます。
　ガンガンいこうぜ！



ドラゴンクエスト

- poll()とは？

pollは、複数のファイルディスクリプタ(ソケット)の状態を、同時に、監視するためのシステムコール関数です。

pollは、監視するファイルディスクリプタのリスト（通常は`pollfd`構造体の配列）と、タイムアウト期間（ミリ秒単位）を引数として取ります。

しかし、poll自身で何か実行する、というわけではなく、監視しているfdの状態が変われば、イベントが起こったら、フラグを立てる。そんな感じの関
数です。
ワンピースで例えると見聞色の覇気のようなもので、見聞色の覇気は、覇気で敵の動きを察知でき、それによって敵の攻撃を先読みして回避したりできます。見聞色の覇気がないとカタクリはルフィの拳の一つにしか対応できず、他の拳にはあってしまいますもんね。（知らない）

ギア！INT_MAX!!

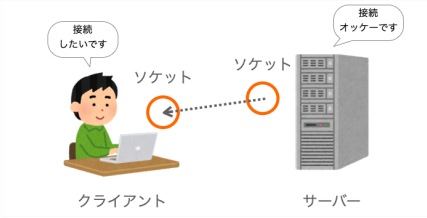
同じようにpoll()も複数のソケットやファイルディスクリプタを同時に監視することで、接続を保ちつつ、イベントハンドラ関数へ、タイムラグなく渡すことができるのです。

下記のサイトもオススメです。

【C言語】ソケット通信について解説

このページにはプロモーションが含まれていますこのページではC言語でのソケット通信の仕方について解説していきたいと思いま...

daeudaeu.com



キャプションを入力

GET、POST、DELETE

GET、POST、DELETEはHTTPメソッドと呼ばれるもので、クライアントがWebサーバーとの間でリソースに対する特定の操作を要求するために使用されます。

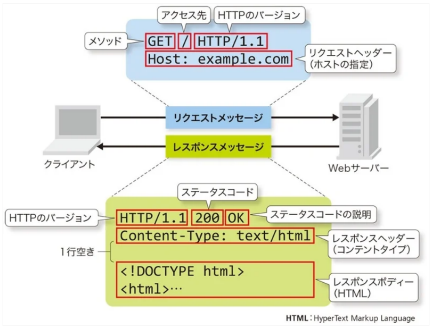
先ほどはあまり深掘りしなかったですが、クライアントとサーバーはメッセージを使ってやり取りをします。

クライアントから HTTP サーバーに送るデータが「リクエストメッセージ」
HTTP サーバーからクライアントに送るデータが「レスポンスメッセージ」

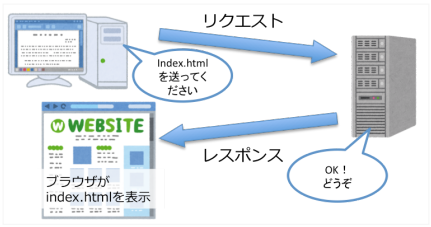
と呼ばれています。
HTTPメソッドはこの**リクエストメッセージ**においての要求の種類を表すもので、「どんな要求をしたいか」を表す情報になります。

詳しくは下記サイトがオススメです。

https://wa3.i-3-i.info/word11405.html



キャプションを入力



キャプションを入力

以下はそれぞれの説明です。

GETは中身がなくてPOSTはある！DELETEはそれを消す！それだけ覚えておけばほぼOKです（たぶん）。

1. **GET**:
   - **目的**: サーバー上のリソースの情報を取得するために使用されます。
   - **特性**:
     - データはURLのクエリパラメータとして送信される。
     - イデンポテント（同じリクエストを何度実行しても結果が変わらない）。
     - 安全（サーバー上のリソースを変更しない）。
   - **使用例**: Webページの表示、データのクエリなど。
2. **POST**:
   - **目的**: サーバー上のリソースを作成または更新するために使用されます。
   - **特性**:
     - データはリクエストボディに含まれ

- イデンポテントではない（同じリクエストを複数回実行すると異なる結果が得られることがある）。
- **使用例**: フォームの送信、新しいデータの作成、データの更新など。

3. **DELETE**:
   - **目的**: サーバー上のリソースを削除するために使用されます。
   - **特性**:
     - イデンポテント（同じリクエストを何度実行しても結果が変わらない）。
   - **使用例**: データの削除、アカウントの削除など。

イベント

webserv のもっとも根幹とも言われるイベント。while で回しているのですが、中で何が行われているのでしょうか。

- wait !!

- listen !!
- write !!
- read !!
- connect !!
- requests !!
- cull !!

1 .wait

waitでは主にpoll()をします。以下の四つのソケットを監視し、発生したイベントを適切なカテゴリに分類します。

1. **listen_sockets**: これはリスニングソケットの集合で、新しいクライアントからの接続要求を待機しています。新しいクライアントが接続を試みると、このソケット上でイベントが発生します。
2. **connections**: これは既存のクライアントとの接続を表すソケットの集合です。クライアントとサーバー間のデータの送受信や接続の終了など、各接続に関連するイベントがこのソケット上で発生します。
3. **read_resources**: これは読み取り操作を待機しているリソースやソケットの集合です。データが読み取り可能になったとき、これらのリソース上でイベントが発生します。
4. **write_resources**: これは書き込み操作を待機しているリソースやソケットの集合です。データが書き込み可能になったとき、これらのリソース上でイベントが発生します。

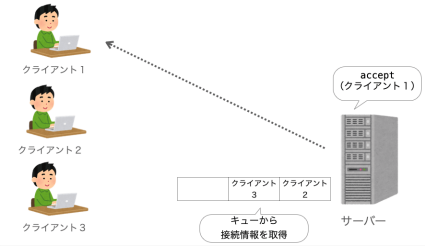
2 .listen

listen では主にaccept()をします。while に入る前にprepare_listeners()でlisten()、bind() しているのですが、そこではソケットで指定した特定のIPアドレスとポート番号に、アクセスするファイルに接続しようとしたクライアントがあるかどうかを待機しています。こ

の時点で、ソケットは「リスニングソケット」として動作します。

クライアントがサーバーに接続を試みると、リスニングソケットが新しいクライアントからの接続要求を受け入れ、リスニングソケット上の新しい接続ソケット」を作成し、このソケットを使用してクライアントとの通信を行います。リスニングソケット自体は、新しい接続要求を待機し続けます。

要するに**accept**は「クライアントからの接続要求を受け入れる」ことです。



キャプションを入力

３.read/write

readと**write**ではその名前の通り、readとwriteします。以上。
(y ゜ロ゜) y もっと説明しろ！

すみません。怒られたので少しだけ詳しく。
本課題ではreadまたはrecv(レシーブ的な)、とwriteまたはsend(送る関数)が使用可能なのですが、これらは『HTTPリクエスト』『HTTPリスポンス』に関係しています。

HTTPリクエスト：「このページを下さい」　要求
HTTPレスポンス：「はい、言われてたページだよ」　返答

readと**write**はソケット上でのデータの受信と送信を行う関数であり、これを使用してHTTPリクエストやHTTPレスポンスのデータを受信・送信することができます。

4.connect

connectでは接続ソケット上で、ソケット上に読み取り可能なデータが存在するか、またはソケットがデータの書き込みを受け付ける準備ができているか、等を検出しそれに応じて適切な処理を行っています。

先ほどの**read**と**write**と違う点はデータのソースと目的にあります。

- **read**と**write**はリソース（例：ファイルやデータベース）からの読み取り、また書き込みを処理しています。これは、データの取得や変換などの特定のタスクに関連しています。
- **具体例**: サーバーがクライアントからのHTTPリクエストを受け取った際、**read**関数を使用してリクエストデータを受信します。また、サーバーがHTTPレスポンスをクライアントに送信する際には、**write**関数を使用してレスポンスデータを送信します。

- **connect** では接続（クライアントとの通信）からの読み取り、書き込みを処理しています。これは、メッセージの受信や応答の生成など、通信に関連するタスクに関連しています。
- **具体例**: クライアントがサーバーに接続を試みた際、**connect**はその接続要求を検出し、書き込み可能、読み取り可能とフラグを立てます。

5.requests

requestsではクライアントからのHTTPリクエストを直接処理しております。

ん？？(°ω°υ)

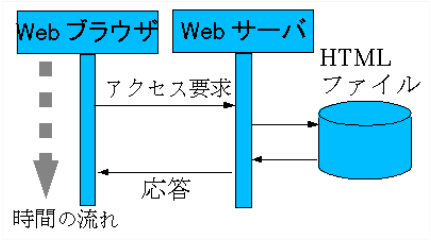
なんか3 のreadでもHTTPリクエストの処理をやっていたような。。。

すみません！！実はreadだけ**CGI**というプログラムを実行するかしないかで分岐するのです。

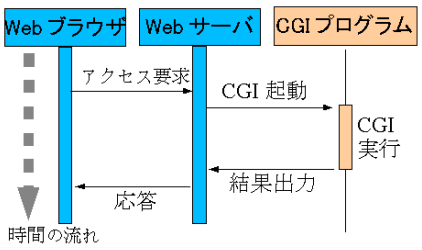
CGIに関しては後ほど説明しますが、**read/write**での挙動を改めてまとめると

1. **HTTPリクエストの受信 (read)**:
   - クライアントからのHTTPリクエストがサーバーに到着すると、サーバーはまずそのリクエストを**read**します。
   - リクエストに基づいて、サーバーは外部のリソース（例：CGIスクリプトやデータベース）からデータを取得する必要がある場合があります。このとき、サーバーは**CGI** プログラムを実行し、その結果を**read**します。
   - このプロセスは、サーバーがクライアントのリクエストに応じて必要なデータを取得するためのものです。
2. **HTTPレスポンスの送信 (write)**:
   - サーバーが必要なデータを取得した後、次のステップはクライアントにレスポンスを送信することです。
   - このレスポンスは、先ほど取得したデータを基に生成されます。
   - このステップでは、サーバーはレスポンスデータを**write**してクライアントに送信します。
   - 通常、このステップでは**CGI**の実行は行われません。なぜなら、**CGI**の実行とデータの取得は既に前のステップで完了しているからです。

つまり**requests**では**CGI**を介さずにHTTPリクエストを直接処理しています。



通常のHTMLファイルにアクセスした場合



CGIにアクセスした場合

6.cull

cull ではサーバーが管理している接続の中で、一定時間アクティブでない接続を特定し、それらの接続を適切に処理しています。これにより、サーバーのリソースを効率的に使用し、無駄な接続を削除することができます。

1. **DuringResponseSending**: この状態は、サーバーがクライアントにレスポンスを送信中であることを示しています。この状態では、接続がまだアクティブであり、データの送信が完了するまで待機する必要があるため、何も行われません。
2. **Closed**: この状態は、接続が既に閉じられていることを示しています。この場合、接続は不要であり、リソースを解放するために接続を削除します。
3. **それ以外の状態**: この場合、接続はアクティブではないが、まだ完全に閉じられていないか、または何らかのエラー状態にある可能性があります。このような状態の接続は、サーバーのリソースを無駄に消費する可能性があるため、読み取りと書き込みのリソースを削除して、接続をクリーンアップします。さらに、クライアントにはHTTPエラーコード408（Request Timeout）を返して、リクエストがタイムアウトしたことを通知します。

例えば「Closed」という状態は、クライアント側が接続を閉じた、または何らかの理由で接続が切断されたことを示しています。
この場合、サーバー側にはその接続を維持する理由がなくなります。したがって、サーバー側も接続を削除して、使用していたリソース（メモリ、ソケットなど）を解放します。

因みに**cull**は「取り除く」という意味の単語で、アクティブでない、または不要になった接続を「取り除く」または「削除する」という意味になります。

いらないものをカール（刈ーる）... ...はい、すみません。

CGI

最後にCGIです！
「Common Gateway Interface」の略で、直訳すると「共通の出入口の接点」みたいな意味になります。

普通のホームページは静的ファイル、予め用意されているファイルを渡します。

個人のブログやPDFファイルのリンクなどがそれにあたります。

阿部寛のサイトを開くのが速いのもそのお陰！！(それ以外の努力もしてそう)

インド人

では逆にCGIでは何を取り扱うのか。それは動的ファイルと呼ばれるもので、現代の殆どのサイトがこの動的ファイルを使っています。

例えば天気予報や株価チャート、ネット掲示板など、リアルタイムで変更するものは動的ファイルにあたります。

さて、CGIではどのように動的ファイルを作っているのでしょうか。

CGIでの作業があるのですが、大きく分けて2種類あります。

1. **外部プログラムとの連携**: CGIは、ウェブサーバーが外部のプログラムやスクリプトを実行するためのインターフェースとして機能します。例えば、データベースへのクエリやシステムコマンドの実行など、サーバー外部のリソースとの連携が必要な場合に使用されます。この場合、CGIは外部プログラムの実行結果を受け取り、それをHTTPレスポンスとしてクライアントに返します。
2. **CGI自体の計算**: CGIスクリプト自体が計算や処理を行い、その結果をクライアントに返す場合もあります。この場合、外部のプログラムやリソースとの連携は必要ありません。CGIスクリプトは、クライアントからのリクエストに基づいて動的にコンテンツを生成し、それをHTTPレスポンスとして返します。