ウェブサーブ

このときようやく、なぜURLがHTTPで始まるのか が理解できます。

概要

このプロジェクトは、あなた自身のHTTPサーバを書くことです。実際のブラウザでテストすることができます。

HTTPは、インターネット上で最も利用されているプロトコルの一つです。その難解さを知っておくと、たとえWebサイトを担当することがなくても役に立つでしょう。

バージョン：20.2

内容

第一章 はじめに

HTTP（Hypertext **Transfer Protocol**）は、分散、協調、ハイパーメディア情報システムのためのアプリケーションプロトコルである。

HTTPはWorld Wide Webのデータ通信の基盤であり、ハイパーテキスト文書には、ユーザーが簡単にアクセスできる他のリソースへのハイパーリンクが含まれています。例えば、マウスのクリックやウェブブラウザの画面をタップすることでアクセスできる。

HTTPは、ハイパーテキストとWWWを促進するために開発された。

Webサーバーの主な機能は、Webページを保存、処理し、クライアントに配信することです。クライアントとサーバーの間の通信は、HTTP（Hypertext Transfer Protocol）を使って行われます。

配信されるページはHTML文書が多く、テキストコンテンツに加えて画像、スタイルシート、スクリプトを含む場合があります。

アクセス数の多いウェブサイトでは、複数のウェブサーバーを使用することがあります。

ユーザーエージェント（一般的にはWebブラウザやWebクローラ）は、HTTPを使って特定のリソースを要求することで通信を開始し、サーバーはそのリソースの内容、またはそれができない場合はエラーメッセージを応答する。リソースは通常、サーバーの二次記憶装置上にある実ファイルであるが、必ずしもそうではなく、ウェブサーバーの実装方法によって異なる。

主な機能はコンテンツを提供することですが、HTTPの完全な実装には、クライアントからコンテンツを受け取る方法も含まれています。この機能は、ファイルのアップロードを含むウェブフォームの送信に使用されます。

第II章 総則

- プログラムはどんな状況でも（たとえメモリ不足でも）クラッシュしてはいけませんし、予期せず終了してはいけません。
  もしそうなった場合、あなたのプロジェクトは機能しないものとみなされ、あなたの成績は
  0.
- ソースファイルをコンパイルするMakefileを提出する必要があります。再リンクはしてはいけません。
- Makefileには少なくともルールが含まれている必要があります。
  $(NAME)、all、clean、fclean、re。
- c++と-Wall -Wextra -Werrorフラグでコンパイルしてください。
- あなたのコードは**C++ 98標準に**準拠している必要があります。そして、-std=c++98フラグを追加しても、コンパイルできるはずです。
- 常にできる限りC++の機能を使って開発するように心がける（例えば <cstring> over <string.h>) を参照してください。Cの関数を使うこともできますが、可能であれば常にC++版を使うようにしましょう。
- 外部ライブラリやBoostライブラリの使用は禁止されています。

第三章 必須項目

プログラム名	ウェブサーブ
ファイルを提出する	Makefile、*.{h, hpp}、*.cpp、*.tpp、*.ipp。 設定ファイル
メイクファイル	NAME、All、 Clean、Fclean、Re
論証	[設定ファイル]をクリックします。
外部機能。	すべてC++ 98で。 execve、dup、dup2、pipe、strerror、gai_strerror、errno、dup、dup2、fork、htons、htonl、ntohs、ntohl、select、poll、epoll (epoll_create, epoll_ctl, epoll_wait),kqueue (kqueue、 kevent), socket, accept, listen, send, recv, bind, connect, getaddrinfo, freeaddrinfo, setsockopt, getsockname, getprotobyname, fcntl
リブート認可	非対称性
商品説明	C++で作るHTTPサーバー 98

C++98でHTTPサーバーを書く必要があります。

あなたの実行ファイルは、次のように実行されます。

./webserv [設定ファイル]です。

なお、poll()はサブジェクトや評価尺度で言及されていても、select()、kqueue()、epoll()など、同等のものを使用することが可能です。

このプロジェクトを始める前に、RFCを読み、telnetとNGINXでいくつかのテストを行ってください。
RFCをすべて実装する必要がなくても、読むことで必要な機能を開発することができます。

III.1　必要条件

- プログラムは、設定ファイルを引数として受け取るか、デフォルトのパスを使用しなければなりません。
- 他のWebサーバーをexecveすることはできません。
- サーバーは決してブロックしてはならず、必要であればクライアントを適切にバウンスすることができます。
- ノンブロッキングでなければならず、クライアントとサーバ間のすべてのI/O操作（リッスンを含む）に対して**1つのpoll()**（または同等のもの）しか使用してはならない。
- poll()（または同等）は、読み込みと書き込みを同時にチェックする必要があります。
- poll()（またはそれに相当するもの）を通さずに読み書きの操作をすることは、決してあってはなりません。
- 読み取りまたは書き込み操作の後に errno の値を確認することは厳禁です。
- 設定ファイルを読み込む前に poll() (または同等) を使用する必要はありません。

ノンブロッキングのファイルディスクリプタを使用しなければならないので、 poll()（または同等のもの）を使用しないread/recvまたはwrite/send関数を使用することが可能で、サーバーはブロックされないでしょう。
しかし、システムリソースをより多く消費することになります。
したがって、poll()（または同等のもの）を使用せずに、任意のファイル記述子で読み取り/再読み取りまたは書き込み/送信を試みた場合、 あなたの成績は0になります。

- FD_SET, FD_CLR, FD_ISSET, FD_ZERO など、あらゆるマクロや定義を利用することができます（何をどのように行うかを理解することは非常に有効です）。
- サーバーへのリクエストは、永遠にハングアップしてはならないのです。
- サーバーは、お客様が選択された**ウェブブラウザに**対応している必要があります。
- NGINXはHTTP 1.1に準拠しており、ヘッダーとアンサーの動作を比較するために使用される可能性があることを考慮します。
- HTTPレスポンスのステータスコードは正確でなければなりません。
- **エラーページが**ない場合は、サーバーに**デフォルトのエラーページを用意する**必要があります。
- CGI以外のもの（PHPとかPythonとか）にはforkは使えません。
- **完全な静的ウェブサイトを提供**できること。
- クライアントが**ファイルをアップロード**できること。
- 少なくともGET, POST, DELETEメソッドが必要です。

- サーバーのストレステスト。何が何でも利用可能な状態を維持しなければならない。
- サーバーは複数のポートをリッスンできる必要があります（*設定ファイル*参照）。

III.2　MacOS用 のみ

MacOSは他のUnix系OSと同じようにwrite()を実装していないので、fcntl()を使うことが許されているのです。
他のUnix系OSと同様の挙動を得るためには、ファイルディスクリプタをノンブロッキングモードで使用する必要があります。

ただし、fcntl()は次のようにしか使用できません。 fcntl(fd, F_SETFL, O_NONBLOCK);
それ以外の旗は禁止されています。

III.3　コンフィギュレーション ファイル

NGINXの設定ファイルの'server'の部分から、いくつかのヒントを得ることができます。

設定ファイルでは

- 各「サーバー」のポートとホストを選択します。
- server_namesを設定するかしないか。
- host:port の最初のサーバは、この host:port のデフォルトとなります (つまり、他のサーバに属さないすべてのリクエストに応答します)。
- デフォルトのエラーページを設定する。
- クライアントのボディサイズを制限する。
- 以下のルールや設定を1つまたは複数組み合わせて、ルートを設定します（ルートには正規表現を使用しません）。
    - ルートに受け入れられる HTTP メソッドのリストを定義します。
    - HTTPリダイレクトを定義する。
    - ファイルを検索すべきディレクトリまたはファイルを定義する（例えば、url /kapouetが/tmp/wwwにルートされている場合、url /kapouet/pouic/toto/pouetは、/tmp/wwwにルートされている場合。/tmp/www/pouic/toto/pouet)です。
    - ディレクトリ掲載のオン／オフを切り替える。

- 要求がディレクトリの場合、回答するデフォルトファイルを設定します。
- 特定のファイル拡張子（例えば.php）に基づきCGIを実行する。
- アップロードされたファイルを受け入れることができるようにし、その保存先を設定します。
    * CGIってなんだろう？
    * CGIを直接呼び出さないので、PATH_INFOにはフルパスを使用します。
    * ただ、チャンクされたリクエストの場合、サーバーはチャンクを解除する必要があり、CGIはボディの終わりとしてEOFを期待することを忘れないでください。
    * CGI の出力も同様です。CGIからcontent_lengthが返されない場合、EOF は返されたデータの終わりを示します。
    * プログラムは、要求されたファイルを第一引数としてCGIを呼び出す必要があります。
    * 相対パスでファイルにアクセスするため、CGIは正しいディレクトリで実行する必要があります。
    * サーバーは1つのCGI（php-CGI、Pythonなど）で動作する必要があります。

評価中にすべての機能が動作することをテストし実証するために、いくつかの設定ファイルやデフォルトの基本ファイルを提供する必要があります。

ある動作について疑問がある場合は、自分のプログラムの動作とNGINXの動作を比較するとよいでしょう。
例えば、server_nameがどのように機能するかを確認します。
私たちはあなたと小さなテスターを共有しました。 ブラウザとテストですべてがうまくいくなら、それを通過することは必須ではありませんが、いくつかのバグを狩るのに役立ちます。

重要なのは回復力です。 サーバーは絶対に死んではいけない。

1つのプログラムだけでテストを行わないこと。 PythonやGolangなど、より便利な言語を使ってテストを書きましょう。 必要であればCやC++でも。

第四章 ボーナス

パート

ここでは、追加できる機能を紹介します。

- クッキーとセッション管理に対応（簡単な例を用意）。
- 複数のCGIを扱う。

ボーナスパーツは、必須パーツがPERFECTである場合にのみ査定されます。パーフェクトとは、必須パートが統合的に行われ、誤動作することなく動作することを意味します。　　必須条件をすべてクリアしていない場合、ボーナスパーツの評価は一切行われません。

第五章

提出と相互評価

通常通り、Gitリポジトリに課題を提出する。防衛戦では、あなたのリポジトリ内の作品だけが評価されます。ファイル名が正しいかどうか、遠慮なく再確認してください。