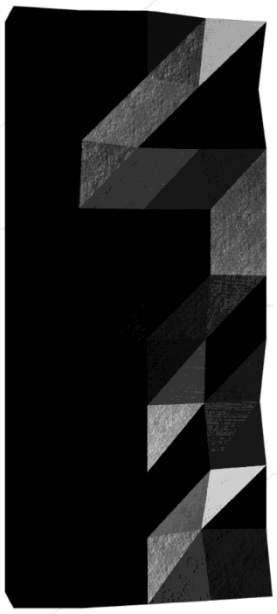

ウェブサーブ

このときようやく、なぜURLが始まるのかが理解できます。 HTTPで

Summa

HTTP サーバーを自作する際の注意点です。実際のブラウザとして使用する必要があります。HTTPは、インターネット上で最も使用されているプロトコルの1つです。その難解さは、セブストで作業するわけではなくても、使用することができます。

Version: 20.7

Contents

章 I

はじめに

HTTP（Hypertext Transfer Protocol）は、分散型、協調型、ハイパーメディア型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することができます。

HTTPは、ハイパーテキストとWorld Wide Webを容易にするために開発されました。

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文書であることが多く、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や、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の実ファイルですが、必ずしもそうではなく、Webサーバーの実装方法によって異なります。

章 II

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信す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。

章 III

一般的なルール

- 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、予期せず終了してもいけません。
  その場合、あなたのプロジェクトは機能しないとみなされ、あなたの成績は0となります。
- ソースファイルをコンパイルするNakef ile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。
- あなたのPaket ileは、少なくともルールを含んでいなければなりません：
  ドル（FACIE）、アル1、cリーン、Iクリーンと再。
- c++ と -Tal l -Hextra -terror フラグを使用してコードを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というフラグを追加しても、コンパイルできるはずです。
- できる限りC++の機能を使った開発を心がけましょう（例えば、string .hâよりも4cstringsを選びましょう）。Cの関数を使っても構いませんが、可能であればC++の関数を使うようにしましょう。
- 外部ライブラリやBoostライブラリは一切禁止です。

章 IV

章 V

必須項目

プログラム名	バブサーブ
ファイルを提出する	Makefile、*.{h、hppl、*.cpp、*.tpp、*.ipp、 設定ファイル
メイクファイル	NAME、All、Clean、Fclean、Re
論考	[ファイルされた構成
外部ファンクション	すべてを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、close、read、write、waitpid、 kill、 signal、 access、 stat、 opendir、 readdir および closedirです。
リブフト認定	n/a
商品説明	C++ 98で作るHTTPサーバー

C++ 98でHTTPサーバ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]を参照してください。

章 VI

また、`poll()`が主題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できる。

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。

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はなくても、読むことで必要な機能を開発するのに役立ちます。

ウェブ　　このときようやく、なぜURLがHTTPで始まるのかが理解でサーブ　　きます。

III.1　必要条件

- プログラムは、設定ファイルを引数に取るか、デフォルトのパスを使用する必要があります。
- 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、クライアントとサーバー間のすべてのI/O操作（リッスンも含む）に対して、1つのポーリング（）（または同等のもの）のみを使用しなければならない。
- poll () (または同等のもの)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 () (または同等のもの)を通さずに、読み出しや書き込みの操作をしてはならない。
- errnoの値を確認することは、読み出しや書き込み操作の後では厳禁です。
- 設定ファイルを読み込む前に、poll（）（または同等のもの）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やwrite/send関数を使用することが可能で、サーバーはブロッキングされないだろう。
しかし、より多くのシステムリソースを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/再送信または書き込み/送信を試みた場合、成績は0となります。

- FD_SET、FD_CLR、FD_I SSET、FD_ZERDなど、あらゆるマクロや定義が使えます（何をどうするのか理解すると、とても便利です）。
- サーバーへのリクエストは、永遠にハングアップすることはありません。
- お客様のサーバーが、お客様の選択したウェブブラウザに対応している必要があ

ウェブ　　このときようやく、なぜURLがHTTPで始まるのかが理解で
サーブ ります。　　きます。

- NGINXはHTTP1.1に準拠しており、ヘッダーやアンサー動作の比較に利用でき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。
- エラーページが提供されていない場合は、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。
- 完全に静的な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が必要です。
- 少なくともGET、PDST、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する必要があります。
- 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（「*Con figura!ion* ale」参照）。

ウェブサーバー　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。

III.2　MacOSのみ

MacOSは他のUnix OSと同じようにwrite()を実装していないため、fcntl()を使うことが許されています。

他のUnix OSと同様の動作をさせるためには、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()の使用は、次のようにのみ許可されています： fcntl(fd, F_SETFL, O_NONBLOCK)；

それ以外の旗は禁止されています。

III.3　コンフィグレーションファイル

NOINXの「サーバー」の部分からヒントを得ることができます。

構成リール。

コンフィギュレーションファイルでは

- 各「サーバー」のポートとホストを選択します。
- サーバーのnanesをセットアップするかどうか。
- ホスト：ポートに対する最初のサーバーは、このホスト：ポートのデフォルトとなります（つまり、他のサーバー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つまたは複数使ってルートを設定します（ルートは正

ウェブ　　このときようやく、なぜURLがHTTPで始まるのかが理解でサーブ　　きます。
規表現を使いません）：

- 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。
- ファイルを検索する元となるディレクトリやファイルを定義する（例．

もし、url /kapouetが/tnp/vwにルートされている場合、url /kapouet /poui c/toto/pouet は。

/tnp/vw/poui c/t ofo/pouet）。

- ディレクトリリストの表示/非表示を切り替えます。

ウェブサーバー　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。

- リクエストがディレクトリの場合、回答するデフォルトのファイルを設定する。
- 特定のファイル拡張子（例：.php）でリースされたCGIを実行することができます。
- POSTメソッドとGETメソッドで動作するようにする。
- アップロードされたファイルを受け付けるようにし、保存先を設定します。

CGIってなんだろうと思いませんか?

CGIを直接呼び出さないので、PATH INF0として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の出力についても同じことが言えます。CGIからコンテンツ長が返されない場合、EOFは返されたデータの終わりを示す。

あなたのプログラムは、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。CGIは、相対パスによるファイルアクセスのために、正しいディレクトリで実行される必要があります。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と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の動作を比較してみるとよいでしょう。
例えば、server_nameがどのように機能するのかを確認します。
私たちはあなたと小さなテスターを共有しました。ブラウザやテストがすべて正常に動作するのであれば、合格することは必須ではありませんが、いくつかのバグを見つけるのに役立つと思います。

ウェブ　　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。
サーブ

大切なのは回復力です。サーバーは絶対に死ぬべきではない。

1つのプログラムだけでテストしないこと。PythonやGolangなど、より便利な言語でテストを書きましょう。CやC++でもOKです。

章 IV

ボーナスパート

ここでは、追加できる機能を紹介します：

- Cookieとセッション管理をサポートする（クイック例を準備する）。
- 複数のCGIを扱う。

ボーナスパートは、必須パートがPERFECTである場合にのみ評価されます。完璧とは、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、ボーナスパートはまったく評価されません。

章 V

章 VI

提出物および相互評価

通常通り、Gigリポジトリに課題を提出してください。あなたのリポジトリ内の作品だけが、ディフェンスの際に評価されます。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。

章 VII