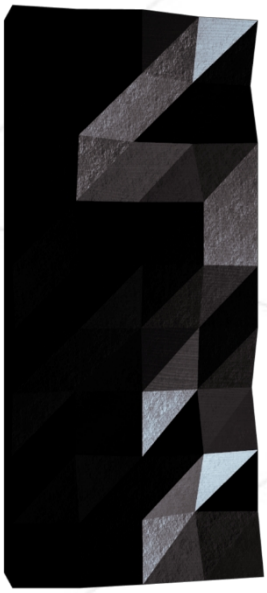

Webserv

URL이 HTTP로 시작하는 이유를 마침내 이해하게 되는 순간입니다.

요약:

이 프로젝트는 ow HTTP 서버를 작성하는 것입니다. 실제

브라우저로 테스트해 볼 수 있습니다.

HTTP는 인터넷에서 가장 많이 사용되는 프로토콜 중 하나입니다.

웹 사이트에서 작업하지 않더라도 그 비밀을 아는 것은 유용할 것입니다.

버전: 21.2

콘텐츠

1장 소개

HTTP(**하이퍼텍스트 전송 프로토콜**)는 분산형 협업 하이퍼미디어 정보 시스템을 위한 애플리케이션 프로토콜입니다.

HTTP는 하이퍼텍스트 문서에 사용자가 쉽게 액세스할 수 있는 다른 리소스에 대한 하이퍼링크가 포함된 월드와이드웹 데이터 통신의 기반입니다. 예를 들어, 마우스 클릭이나 웹 브라우저에서 화면을 탭하는 등의 방법으로 하이퍼텍스트 문서에 액세스할 수 있습니다.

HTTP는 하이퍼텍스트와 월드와이드웹을 용이하게 하기 위해 개발되었습니다.

웹 서버의 주요 기능은 웹 페이지를 저장, 처리 및 클라이언트에 전달하는 것입니다. 클라이언트와 서버 간의 통신은 하이퍼텍스트 전송 프로토콜(HTTP)을 사용하여 이루어집니다.

전달되는 페이지는 텍스트 콘텐츠 외에 이미지, 스타일시트, 스크립트 등이 포함된 HTML 문서가 대부분입니다.

트래픽이 많은 웹사이트에는 여러 웹 서버를 사용할 수 있습니다.

사용자 에이전트(일반적으로 웹 브라우저 또는 웹 크롤러)는 HTTP를 사용하여 특정 리소스를 요청하여 통신을 시작하고 서버는 해당 리소스의 콘텐츠 또는 응답할 수 없는 경우 오류 메시지로 응답합니다. 리소스는 일반적으로 서버의 보조 저장소에 있는 실제 파일이지만 반드시 그런 것은 아니며 웹 서버가 구현된 방식에 따라 달라집니다.

주요 기능은 콘텐츠를 제공하는 것이지만, HTTP의 완전한 구현에는 클라이언트로부터 콘텐츠를 수신하는 방법도 포함됩니다. 이 기능은 파일 업로드를 포함한 웹 양식 제출에 사용됩니다.

2장 일반 규칙

- 프로그램이 어떤 상황에서도(메모리가 부족한 경우에도) 충돌해서는 안 되며, 예기치 않게 종료되어서는 안 됩니다.
  이 경우 프로젝트는 작동하지 않는 것으로 간주되며 성적은 다음과 같이 표시됩니다.
  0.
- 소스 파일을 컴파일할 메이크파일을 제출해야 합니다. 다시 링크해서는 안 됩니다.
- 메이크파일에는 최소한 규칙이 포함되어야 합니다:
  (이름), all, clean, fclean, re.
- c++와 -Wall -Wextra -Werror 플래그를 사용하여 코드를 컴파일합니다.
- 코드는 **C++ 98 표준을** 준수해야 합니다. 그런 다음 -std=c++98 플래그를 추가하면 여전히 컴파일됩니다.
- 항상 가능한 한 많은 C++ 기능을 사용하여 개발하도록 노력하세요(예를 들어 <string.h>보다 <cstring>). C 함수를 사용할 수 있지만 가능하면 항상 C++ 버전을 선호하세요.
- 외부 라이브러리 및 부스트 라이브러리는 금지되어 있습니다.

제3장 필수 부분

프로그램 이름	웹서버
파일 제출	메이크파일, *.{h, hpp}, *.cpp, *.tpp, *.ipp, 구성 파일
메이크파일	이름, 모두, 깨끗하다, 깨끗하다, 다시
인수	[구성 파일]
외부 함수.	C++ 98의 모든 것. execve, dup, dup2, pipe, strerror, gai_strerror, errno, dup, dup2, fork, 소켓페어, htons, htonl, ntohs, ntohl, select, poll, epoll(epoll_create, epoll_ctl, epoll_wait), kqueue(kqueue, 이벤트), socket, accept, listen, send, recv, chdir bind, connect, getaddrinfo, freeaddrinfo, setsockopt, getsockname, getprotobyname, fcntl, close, read, write, waitpid, kill, signal, access, stat, open, opendir, readdir 및 closedir.
Libft 인증	n/a
설명	C++ 98의 HTTP 서버

C++ 98로 HTTP 서버를 작성해야 합니다. 실행 파일

은 다음과 같이 실행됩니다:

./웹서버 [구성 파일]

제목과 평가 척도에 poll()이 언급되어 있더라도 select(), kqueue() 또는 epoll()과 같은 동등한 함수를 사용할 수 있습니다.

이 프로젝트를 시작하기 전에 RFC를 읽고 텔넷과 NGINX로 몇 가지 테스트를 해보세요.
모든 RFC를 구현할 필요는 없더라도 이를 읽어보면 필요한 기능을 개발하는 데 도움이 될 것입니다.

Webserv URL이 HTTP로 시작하는 이유를 마침내 이해하게 되는 순간입니다.

III.1 요구 사항

- 프로그램에서 구성 파일을 인수로 받거나 기본 경로를 사용해야 합니다.
- 다른 웹 서버를 실행할 수 없습니다.
- 서버는 절대로 차단해서는 안 되며 필요한 경우 클라이언트를 적절히 바운스할 수 있습니다.
- 비차단 방식이어야 하며 클라이언트와 서버 간의 모든 I/O 작업(수신 포함)에 대해 **1개의** poll()(또는 이에 상응하는)만 사용해야 합니다.
- poll()(또는 이에 상응하는 함수)는 읽기와 쓰기를 동시에 확인해야 합니다.
- poll()(또는 이와 동등한 함수)를 거치지 않고 읽기 또는 쓰기 작업을 수행해서는 안 됩니다.
- 읽기 또는 쓰기 작업 후 errno 값을 확인하는 것은 엄격히 금지되어 있습니다.
- 구성 파일을 읽기 전에 poll()(또는 이와 동등한 함수)를 사용할 필요는 없습니다.

비차단 파일 기술자를 사용해야 하므로 poll()(또는 이와 동등한 함수)가 없는 읽기/받기 또는 쓰기/보내기 함수를 사용할 수 있으며 서버가 차단되지 않습니다.
하지만 시스템 리소스를 더 많이 소모하게 됩니다.
따라서 poll()(또는 이와 동등한 함수)를 사용하지 않고 파일 설명자로 읽기/받기 또는 쓰기/전송을 시도하면 성적이 0점이 됩니다.

- 모든 매크로를 사용할 수 있으며 FD_SET, FD_CLR, FD_ISSET, FD_ZERO와 같이 정의할 수 있습니다(매크로가 무엇을 어떻게 수행하는지 이해하지 못하면 매우 유용합니다).
- 서버에 대한 요청이 영원히 중단되는 일은 절대 없어야 합니다.
- 서버는 선택한 **웹 브라우저와** 호환되어야 합니다.
- NGINX는 HTTP 1.1을 준수하며 헤더와 응답 동작을 비교하는 데 사용될 수 있다고 간주합니다.
- HTTP 응답 상태 코드는 정확해야 합니다.
- **기본 오류 페이지가** 제공되지 않는 경우 서버에 **기본 오류 페이지가** 있어야 합니다.
- 포크는 CGI가 아닌 다른 것(예: PHP, Python 등)에는 사용할 수 없습니다.

- **완전히 정적인 웹사이트를 제공할** 수 있어야 합니다.
- 클라이언트는 **파일을 업로드할** 수 있어야 합니다.
- 최소한 GET, POST, DELETE 메서드가 필요합니다.
- 서버 스트레스 테스트. 어떤 대가를 치르더라도 서버를 계속 사용할 수 있어야 합니다.
- 서버는 여러 포트를 수신할 수 있어야 합니다(*구성 파일* 참조).

III.2 MacOS 전용

MacOS는 다른 유닉스 OS와 같은 방식으로 write()를 구현하지 않으므로 fcntl()을 사용할 수 있습니다.
다른 유닉스 OS와 유사한 동작을 얻으려면 비차단 모드에서 파일 기술자를 사용해야 합니다.

그러나 다음 플래그가 있는 경우에만 fcntl()을 사용할 수 있습니다: F_SETFL, O_NONBLOCK 및 FD_CLOEXEC.
그 외의 깃발은 금지됩니다.

III.3 구성 파일

NGINX 구성 파일의 '서버' 부분에서 영감을 얻을 수 있습니다.

구성 파일에서 다음을 수행할 수 있어야 합니다:

- 각 '서버'의 포트와 호스트를 선택합니다.
- 서버 이름을 설정할지 여부를 설정합니다.
- 호스트:포트의 첫 번째 서버가 이 호스트:포트의 기본값이 됩니다(즉, 다른 서버에 속하지 않은 모든 요청에 응답한다는 의미).
- 기본 오류 페이지를 설정합니다.
- 클라이언트 본문 크기를 제한합니다.
- 다음 규칙/구성 중 하나 또는 여러 개를 사용하여 경로를 설정합니다(경로에 정규식을 사용하지 않음):
    - 경로에 허용되는 HTTP 메서드 목록을 정의합니다.

- HTTP 리디렉션을 정의합니다.
- 파일을 검색할 디렉토리 또는 파일을 정의합니다(예: url /kapouet이 /tmp/www에 루팅 된 경우, url /kapouet/pouic/toto/pouet은 /tmp/www/pouic/toto/pouet).
- 디렉토리 목록을 켜거나 끕니다.

Webserv URL이 HTTP로 시작하는 이유를 마침내 이해하게 되는 순간입니다.

- 요청이 디렉터리인 경우 응답할 기본 파일을 설정합니다.
- 특정 파일 확장자(예: .php)를 기반으로 CGI를 실행합니다.
- POST 및 GET 메서드와 함께 작동하도록 설정합니다.
- 경로가 업 로 드 된 파일을 수락하고 저장할 위치를 구성할 수 있도록 설정하세요.

  * CGI가 무엇인지 궁금하신가요?
  * CGI를 직접 호출하지 않으므로 전체 경로를 PATH_INFO로 사용하세요.
  * 청크된 요청의 경우 서버가 청크를 해제해야 하며, CGI는 본문 끝을 EOF로 예상한다는 점을 기억하세요.
  * CGI의 출력도 마찬가지입니다. CGI에서 content_length가 반환되지 않으면 EOF는 반환된 데이터의 끝을 표시합니다.
  * 프로그램은 요청된 파일을 첫 번째 인수로 사용하여 CGI를 호출해야 합니다.
  * CGI는 상대 경로 파일 액세스를 위해 올바른 디렉토리에서 실행해야 합니다.
  * 서버는 하나의 CGI(php-CGI, Python 등)로 작동해야 합니다.

평가 중에 모든 기능이 작동하는지 테스트하고 시연하려면 몇 가지 구성 파일과 기본 기본 파일을 제공해야 합니다.

한 가지 동작에 대한 질문이 있는 경우 프로그램 동작과 NGINX의 동작을 비교해야 합니다.
예를 들어 server_name의 작동 방식을 확인하세요.
간단한 테스터를 공유해 드렸습니다. 브라우저와 테스트에서 모든 것이 정상적으로 작동한다면 반드시 통과해야 하는 것은 아니지만 버그를 찾는 데 도움이 될 수 있습니다.

중요한 것은 복원력입니다. 서버는 절대 죽지 않아야 합니다.

하나의 프로그램으로만 테스트하지 마세요. Python이나 Golang 등 더 편리한 언어로 테스트를 작성하세요. 원한다면 C나 C++로도 가능합니다.

4장 보너스 부분

추가할 수 있는 추가 기능은 다음과 같습니다:

- 쿠키 및 세션 관리 지원(간단한 예제 준비).
- 여러 CGI를 처리합니다.

보너스 부분은 필수 부분이 완벽한 경우에만 평가됩니다. 완벽하다는 것은 필수 부분을 완벽하게 완료하고 오작동 없이 작동한다는 의미입니다. 모든 필수 요건을 통과하지 못한 경우 보너스 부분은 전혀 평가되지 않습니다.

제5장

제출 및 동료 평가

평소처럼 Git 저장소에 과제를 제출하세요. 방어 기간 동안에는 저장소 내의 작업만 평가됩니다. 파일 이름이 올바른지 다시 한 번 확인하여 주저하지 마세요.

16D85ACC441674FBA2DF65190663F42A3832CEA21E024516795E1223BBA77916734D1
26120A16827E1B16612137E59ECD492E46EAB67D109B142D49054A7C281404901890F
619D682524F5