

## Popište rozdíly HTTP/1.0 a 1.1

"Host" hlavička pro name-based vhosty  
keepalive  
chunked encoding.  
(chce to malinko rozvést).

## Struktura HTTP požadavku a odpovědi

Požadavek:  
Method RequestUri HTTPVersion CRLF  
[Header CRLF]\*  
[CRLF body]  
Příklad:  
GET /index.html HTTP/1.1  
Host: [www.example.com](http://www.example.com)

## Napište třídy stavových kódů protokolu HTTP.

1xx information  
2xx success  
3xx redirection  
4xx client error  
5xx server error

## Co je URI a z čeho se skládá, uveďte schéma pro http.

URI (celým názvem Uniform Resource Identifier – „jednotný identifikátor zdroje“) je řetězec znaku s definovanou strukturou, který slouží k přesné specifikaci zdroje informací (ve smyslu dokument nebo služba), hlavně za účelem jejich použití pomocí počítačové sítě, zejména Internetu.  
protokol://subdomena.subdomena.domena.toplevel:port/cesta/k/souboru/nazev?parametr1=hodnota1&parametr2=hodnota2

## Co je name-based virtual hosting, jak funguje a proč nefunguje v HTTP/1.0

Name based webhosting umožňuje běh více virtuálních hostů na jedné IP adrese. V HTTP 1.1 prohlížeč posílá hlavičku Host, ve které je zadána jakou URL uživatel zadal. Podle DNS se požadavek pošle na server, který podle hlavičky Host rozhodne, který virtuální host použije. V HTTP 1.0 není hlavička Host, proto nefunguje.

## Popsat algoritmus pro IP-based virtual hosting

Požadavky se rozdělují podle cílové IP adresy požadavku. Pokud pro danou adresu neexistuje nakonfigurovaný virtuální host, použije se virtuální host "\_default\_". Neexistuje-li ani ten, použije se globální server config.

## Uveďte vztah mezi webspace a filesystemem.

Filesystem je umístění souboru v souborovém systému operačního systému serveru.  
Webspace je systém souboru, jak ho vidí klient - adresuje se pomocí URL.  
Webspace se konfigurací serveru mapuje na filesystem.

## Co je to MPM a k čemu slouží?

Multi Processing Modules - moduly pro souběžné zpracování požadavku slouží pro kontrolu vytváření potomků (procesu a vláken) k obsluze požadavku na určených portech.

Existují druhy:

Unix  
prefork - obsluha pomocí procesu  
worker - obsluha pomocí vláken (v procesech)  
další platform-specific mpm

## Co je a jak funguje MPM Prefork modul

Tento modul používá pro obsluhu požadavku **procesy**. Na začátku vytvoří hlavní proces StartServers obslužných procesů, které čekají na požadavky. V průběhu činnosti serveru jejich počet upravuje tak, aby počet čekajících procesů ležel mezi hodnotami *MinSpareServers* a *MaxSpareServers*. Maximální počet současně

bežících procesu udává hodnota MaxClients. Procesy běží maximálně tak dlouho, dokud neobslouží *MaxRequestsPerChild* požadavku.

### **Popište vlastnosti souboru .htaccess. Jaké jsou jejich výhody a nevýhody.**

Upravují volby povolené přes AllowOverride v konfiguraci. Usnadňují konfiguraci přímo v adresáři. Nacítají se při každém dotazu.

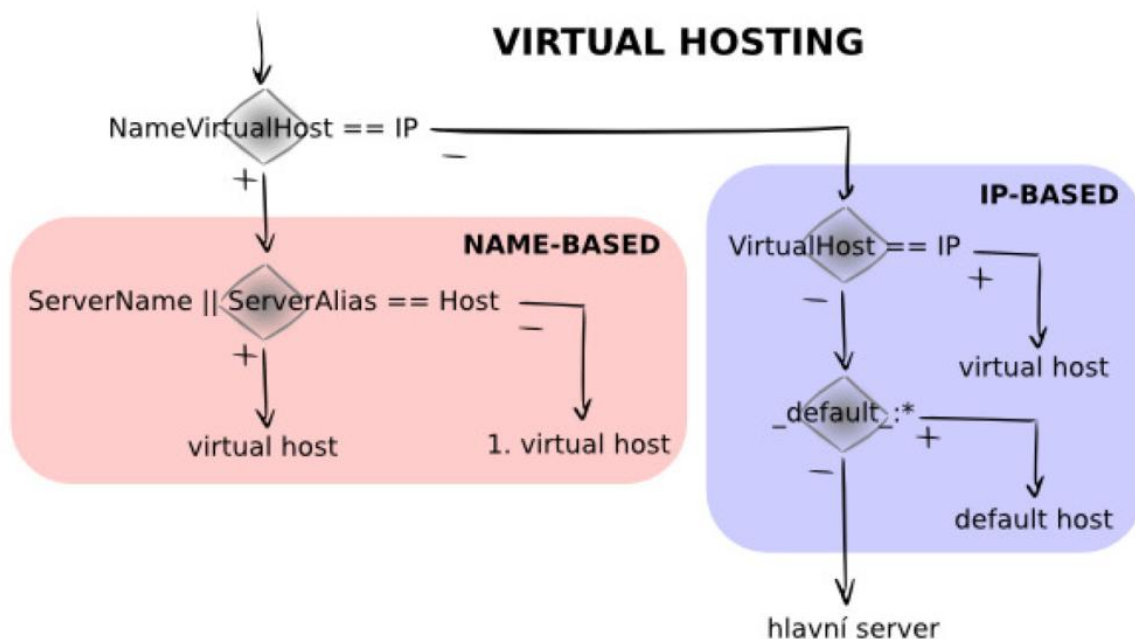
### **Když je nastaveno Order allow, deny uveďte, jestli bude přístup odepřen nebo povolen, když je shoda**

jen allow - **Povolen**  
jen deny - **Zakázán**  
oba - **Zakázán**  
žádný - **Zakázán**

### **Bude přístup odepřen či povolen v případě, že je Order Deny, Allow a nastane shoda:**

jen allow - **Povolen**  
jen deny - **Zakázán**  
oba - **Povolen**  
žádný - **Povolen**

**Uveďte algoritmus výběru hostitele v případě, že je použit zároveň IP a name based virtual hosting.**



**prava pristupu.. byla tabulka kdo se kam snazi prihlasit a konfigurace nejak takhle:**

Location adresar  
basic autorizace  
restriction valid-user  
order deny,allow  
allow \*.mycorp.k328  
match any

Location podadresar  
match all  
restrict user admin nekdo

v tabulce byly snad vsechny moznosti z adresar/podadresar, prihlasen/neprihlasen, z povolene domeny/z nepovolene domeny

## **K čemu slouží konfigurační sekce(kontejnery), jaké existují a v čem se liší?**

Kontejnery umožňují podmíněnou konfiguraci a omezení platnosti direktiv.

Existují dva druhy podle způsobu vyhodnocování:

při startu serveru - `<IfDefine>`, `<IfModule>` a `<IfVersion>`

při požadavku na zdroj - `<Directory>`, `<DirectoryMatch>`, `<Files>`, `<FilesMatch>`, `<Location>`,  
`<LocationMatch>`, `<VirtualHost>`, `<Proxy>`, ...

## **Co je to handler (např. u SetHandler) a jak je typicky implementován.**

Handler je funkce nějakého modulu, která zpracovává některý soubor. Nastavuje se buď pro všechny soubory v nějakém adresáři (SetHandler název) nebo pro všechny soubory s danou příponou daného serveru/vhosta (AddHandler název přípona). Je tedy typicky implementován jako funkce v modulu serveru.

## **Vysvětli HTTP Basic Authentication**

Požadavek neautorizovaného navštěvníka je odmítnut s "401 Authorization required". Klient posle vhlavice v plaintextu (kodovaným base64) uživatelské jméno a heslo. Pokud odpovídá, zobrazí se mu požadovaná stránka. Pokud ne, je odmítnut s "403 Forbidden" (nebo teda může dostat další výzvu).

## **K čemu jsou a jak fungují type maps.**

Type Map je soubor (obvykle s příponou var), ve kterém je uloženo jak postupovat při vyjednávání obsahu.

Pokud je např. zavolána stránka index.html, která neexistuje a existuje soubor index.var, bude se postupovat podle něj. Zde je uloženo při jakém jazyku, content-type atd. se zavola jako jiná stránka.

## **Popište jak server vyhodnocuje prepisu URL pomocí mod\_rewrite**

Viz minulá otázka plus:

mod\_rewrite provádí dva druhy náhrad. První hned po přijetí požadavku - to pokud je RewriteRule v konfiguraci přímo serveru nebo virtual hosta. Pokud je ale RewriteRule v .htaccess souboru, náhrada se posílá až v pozdější fázi - když je původní URL pretransformována na cestu ve filesystému. Pokud je v souboru RewriteRule, provede se substituce a je znovu zavolána nová URL (interne). To zpomaluje běh serveru.

## **Pro požadavek http://www.example.org/old se aplikuje rewrite "RewriteRule /old(.\*)/new\$1 [R]", co se stane a jakými dalšími metodami toto jde zaradit?**

Provede HTTP redirect na adresu http://www.example.org/new. Lze přes "RedirectMatch /old(.\*)/new\$1". Take přes nějakou server-side aplikaci.

## **Vysvětli princip fungování suexec**

SuEXEC umožňuje spustit CGI aplikace pod uživatelem a skupinou která je vlastní, namísto uživatele/skupiny pod kterou běží webserver. Funguje tak, že webserver namísto CGI aplikace pustí SuEXEC wrapper, který je nastaven SUID root (tedy bude běžet jako root, nezávisle na tom kdo ho spustil), a předá mu v rámci parametru cestu k CGI aplikaci. SuEXEC wrapper udělá hofomoc bezpečnostních kontrol a pak, projdou-li, spustí tu aplikaci pod uživatelem a skupinou která je vlastní.

## **Napište alespoň tři body kontrolního algoritmu SUEXEC**

cgi script nevlastní root a nikdo jiný krom uživatele do něj nemůže zapisovat

nikdo jiný nemůže zapisovat ani do nadřazeného adresáře

uživatel splňuje požadavky na minimální uid zvolené při kompilaci

cgi script nesmí mít nastaven suid nebo sgid bit

cgi script musí existovat a být citelný

## **Podrobně popište forward proxy**

Dopředný proxy server je umístěn mezi uživateli a vzdálenými servery.

Více požadavků na stejný zdroj může být obslouženo z cache (vyšší rychlost).

Uživatelé musí mít pro používání proxy serveru správně nastavené aplikace.

Všechen provoz jde zkrz proxy server, proto je možné i řízení přístupu a monitorování dat.

## Vysvětlit princip fungování reverzní proxy

Reverzní proxy funguje na straně poskytovatele obsahu (= webserveru). Funguje tak, že jí dorazí HTTP požadavek, ona si ho přečte, zpracuje a preposle na některý z webserveru co sedí za ní. O nicem z tohohle nemá klient ponětí. Reverzní proxy umožňuje loadbalancing, cachování, SSLko, komprimaci dat atd. Výhodou je možnost hodně jemné konfigurace (podle hlaviček požadavku atd.) oproti loadbalancingu na nižších vrstvách OSI a spousta pěkných featur, nevýhodou je pak o dost větší náročnost (vzít celý požadavek, přečíst ho, zpracovat, upravit a poslat na další stroj je náročná činnost).

## Výhody a nevýhody implementace vlastního modulu

Moduly pro https se píšou v jazyce C. Existují ale moduly, které zpřístupňují metody apache přes API dalším jazykům. Moduly tedy lze psát v téměř libovolném jazyce.

Hlavní nevýhoda přímého přístupu k apachím proměnným je v tom, že snadno můžeme podelat celý server :) Problematické je také ladení.

Psát vlastní moduly se tedy vyplatí pouze v případě, že vyžadujeme co nejrychlejší zpracování na straně serveru.

## Co je to AP, APR?

**AP** je Apache High-level function, funkce vysokoúrovňového API Apache. Z začínají "ap\_"

**APR** je Apache Portable Runtime, API nezávislé na operačním systému. Funkce začínají na "apr\_"

## Výhody a nevýhody u php jako modul, CGI a CLI.

**PHP jako modul** je nejpoužívanější možnost, slouží k dynamickému generování webového obsahu.

**PHP jako CGI** umožňuje psát CGI-BIN skripty v syntaxi PHP. Slouží rovněž ke generování webového obsahu.

**PHP jako CLI** umožňuje psát shellové skripty pomocí PHP. Výstup ale není primárně pro web, tzn. Nemusí být na prvním řádku Content-type atd. Skript může pracovat se standardním vstupem, výstupem a err výstupem. Na rozdíl od CGI může zpracovávat parametry z příkazové řádky. Nicméně pro psaní shellových skriptů se PHP kvůli rychlosti moc nehodí.

## Jaký je nejjednodušší výstup CGI skriptu?

Content-type: <mime-ty><CRLF><CRLF>

## Popsat obecně syntax SSI

```
<!--#element argument="value" argument2="othervalue" ... -->
```

Argumenty jsou odděleny whitespacem a mohou být v uvozovkách.

## K čemu jsou filtry a jaký jsou jejich typy

Vstupní a výstupní. Umožňují upravovat data, jako např. přidávat něco před nebo za ně, měnit velikost písmen, komprimovat je, zpracovávat SSI apod.

## Jak zpracovává požadavek FastCGI

Ve FastCGI životní cyklus procesu nekončí s ukončením obsluhy požadavku, ale proces je recyklován a znovu použit pro obsluhu dalších požadavků. FastCGI aplikace mohou být jedno- i více-vláknové! Webserver (FastCGI process manager) vytvoří procesy aplikace, aby mohl obsluhovat požadavky.

při startu serveru - statické aplikace,

při příchodu požadavku - dynamické aplikace.

1.

FastCGI aplikace provede inicializaci a čeká 2. na požadavky - spojení - od webserveru.

Při příchodu požadavku na webserver je vytvořeno spojení s FastCGI procesem a jsou mu odeslány potřebné informace (ty z proměnných prostředí).

3.

4. FastCGI proces přes stejné spojení odešle odpověď (standardní výstup a standardní chybový výstup).

Uzavřením spojení mezi webserverem a FastCGI aplikací je obsluha kompletní. FastCGI proces ale nekončí, čeká na další požadavky.

5.

## Popište podrobne handshake SSLv3/TLSv1

1. dohoda o šifrách
2. vytvorenie session
3. voliteľne autentizace serveru a klienta

Konkrétne:

klient pošle úvodnú zprávu Hello a seznam podporovaných šifrovacích algoritmu

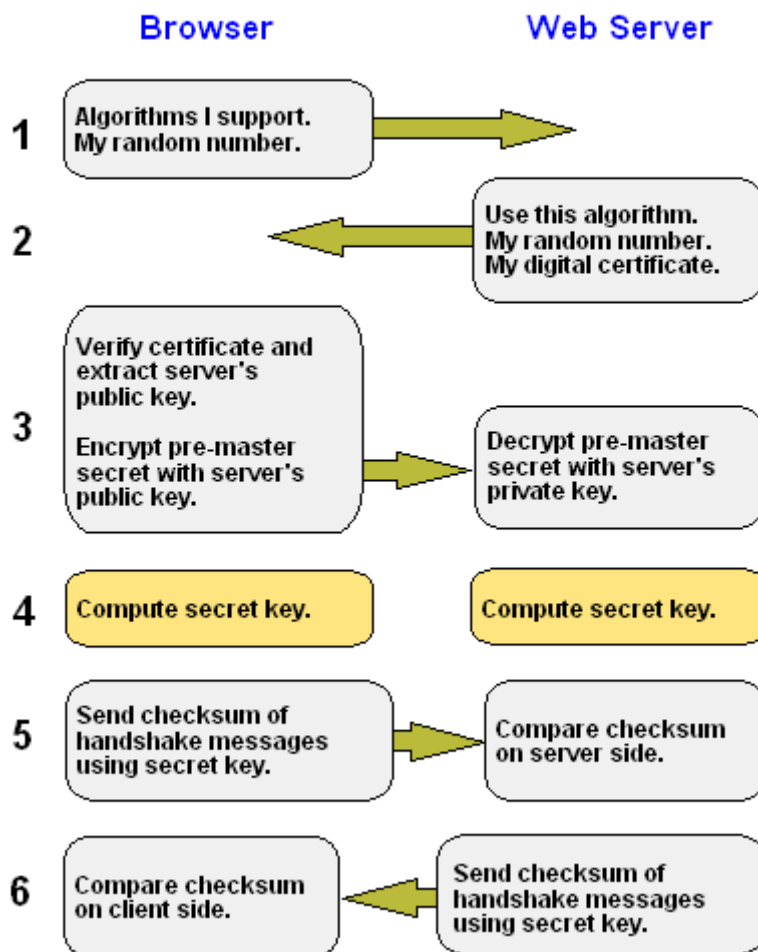
server pošle certifikát obsahujúci verejný kľuč serveru, šifrovací algoritmus z klientského seznamu, ktorý bude používať klient overí certifikát pomocou root certifikátu CA

pomocou náhodných čísel a šifry klient vygeneruje kľuč pro symetrické šifry a zašifruje ju verejným kľučom serveru

server si kľuč rozšifruje svým privátnym kľučom a už vesele šifrovane komunikujú viz obrázek

(<http://www.yourdictionary.com/images/computer/SSL.GIF>)

From Computer Desktop Encyclopedia  
© 2005 The Computer Language Co. Inc.



## Další otázky:

multiviews

Popište, jaký je rozdíl mezi zpracováním TCP komunikace na Apache httpd a nginx

Popište jak webserver zpracovává požadavek