01 - HTML, HTTP, webový server, Apache httpd

Základní pojmy – X/HTML

Hypertext

Hypertext je informační systém, který zobrazuje informace v textu, který obsahuje návěstí odkazující na upřesnění nebo zdroje uváděných informací tzv. hyperlinky neboli česky (hypertextové) odkazy. Rovněž odkazuje i na jiné informace v systému a umožňuje snadné publikování, údržbu a vyhledávání těchto informací. Nejznámějším takovým systémem je World Wide Web.

- **HTML** – HyperText Markup Language
  - **Deklarace** DTD – je povinná pro 4.01+, direktiva `<!DOCTYPE`
  - značkovací jazyk vycházející ze SGML, publikace na webu
  - definice dokumentu
  - **Kořenovy element** – element html reprezentuje celý dokument. Kořenový element je povinný, ale otevírací a ukončovací značka samotná povinná není (pokud tyto značky nebudou v těle dokumentu uvedeny, prohlížeč si je sám doplní podle kontextu).
  - **Hlavička dokumentu** – obsahuje metadata, která se vztahují k celému dokumentu. Definují např. název dokumentu, jazyk, kódování, klíčová slova, popis, použitý styl zobrazení. Element head je opět povinný, ale jeho otevírací a koncová značka povinná není, prohlížeč ji sám doplní podle kontextu.
  - **Tělo dokumentu** – obsahuje vlastní text dokumentu. Element body je povinný, ale jeho otevírací a koncová značka povinná není, prohlížeč ji sám doplní podle kontextu.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transit;ional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
          <title>HTML 4.01 Transitional</title>
    </head>
    <body>
          <a href="http://www.example.com">
                <img src="link.gif" alt="Link na www.example.com"><a>
    </body>
</html>
```

- XHTML je XML verze jazyka HTML
  - *Měl by* to být validní XML dokument
  - *Měl by* se „dobře“ zobrazit ve webovém prohlížeči

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
```

Některé webové prohlížeče mají problém se zobrazením, pokud není uvedena deklarace <!DOCTYPE jako první. Bez deklarace <?xml encoding=““?> se předpokládá UTF-8/16 kódování.

(X)HTML

- Deklarace DTD – je povinná až ve verzi 4.01, je uvedena direktivou <!DOCTYPE.
- Kořenový element – element html reprezentuje celý dokument. Kořenový element je povinný, ale otevírací a ukončovací značka samotná povinná není (pokud tyto značky nebudou v těle dokumentu uvedeny, prohlížeč si je sám doplní podle kontextu).
- Hlavička dokumentu – obsahuje metadata, která se vztahují k celému dokumentu. Definují např. název dokumentu, jazyk, kódování, klíčová slova, popis, použitý styl zobrazení. Element head je opět povinný, ale jeho otevírací a koncová značka povinná není, prohlížeč ji sám doplní podle kontextu.
- Tělo dokumentu – obsahuje vlastní text dokumentu. Element body je povinný, ale jeho otevírací a koncová značka povinná není, prohlížeč ji sám doplní podle kontextu.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
      "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>HTML 4.01 Transitional</title>
  </head>
  <body>
    <a href="http://www.example.com"><img src="link.gif" alt="Link na www.example.com"></a>
  </body>
```

</html>
HTML 4.01 Transitional

- XML verze jazyka HTML je XHTML (http://cs.wikipedia.org/wiki/Extensible_HyperText_Markup_Language)

Základní pojmy – URL, URI, IRI

● URL = Uniform Resource Locator
  ● řetězec znaků s definovanou strukturou
  ● specifikující **umístěni zdrojů** na Internetu
  ● je podmnožinou URI

● URI = Uniform Resource Identifier
  ● **pojmenovava zdroje** na Internetu
  ● syntaxe je specifická podle zvoleného schématu

● IRI = Internationalized Resource Identifier
  ● zobecněním URI
  ● umožnuje využívat Unicode místo ASCII (resp. Latin1)

Synataxe URL

● Zjednodušená struktura

```
generic-url ::= scheme:scheme-specific
scheme ::= http | ftp ...
scheme-specific ::= common-inet | ...
common-inet ::= //user:password@host:port/url-path
```

● Pro schéma HTTP může URL obsahovat #fragment
  ● není součástí URL podle RFC
  ● parser by měl být schopen jej rozpoznat

● V URL lze bezpečně (bez zakódování) používat pouze
  ● alfanumerické znaky ([0-9a-zA-Z])
  ● speciální znaky $ - _ . + ! * ' ()
  ● ostatní musí být kódovány pomocí %xx (kód je podle ISO 8859-1)

● Nebezpečné (unsafe) znaky: <> “ # % {} [] | \ ^ ~ `
  ● mohou být považovány za speciální při přenosu
  ● mohou být změněny, proto by měly být kódovány

● Vyhrazené (reserved) znaky: ; / ? : @ = &
  ● speciální význam v URL
  ● pro jiné použití musí být kódovány

● Ukázka kódování znaků

Znak	Kod	Znak	Kod	Znak	Kod
&	%2B	/	%2F	:	%3A
;	%3B	=	%3D	?	%3F
@	%40				

Protokol HTTP

● Základní vlastnosti
  ● HyperText Transfer Protocol
  ● Textový protokol (raw data)
  ● Bezestavový protokol
  ● Fungování způsobem: požadavek – odpověď
  ● Protokol aplikační vrstvy (ISO OSI)
  ● Nad TCP/IP protokoly (obvykle port 80)
  ● Verze
  ● 1.0 RFC1945 http://tools.ietf.org/html/rfc1945
  ● 1.1 RFC2616, RFC2068 http://tools.ietf.org/html/rfc2616
  ● HTTPS – šifrovaná verze protokolu (HTTP + SSL/TLS)

Protokol HTTP/1.1

● Vlastnosti z HTTP/1.0 zůstávají
● Rozšíření oproti HTTP/1.0
  ● Hlavička `Host:` u každého požadavku (možnost používat virtální „servery“ – virtual hosts)

● Možnost poslat odpověď ve více částech (chunked encoding)
● Podpora trvalého (persistent) spojení se serverem (případně hlavička `Connection: close` při ukončení spojení)
● Správné zpracování odpovědi `100 Continue` (obvykle se ignoruje)
● Vylepšená podpora Cache-Control

Protokol HTTP – Pojmy (terminologie)
● **Zpráva (message)** - Základní jednotka HTTP komunikace skládající se ze strukturované sekvence bytů (oktetů) a přenášená spojením.
● **Požadavek (request)** - HTTP zpráva od klienta serveru
● **Odpověď (response)** - HTTP zpráva od serveru klientovi
● **Zdroj (resource)** - Datový o objekt nebo služba na síti, která může být identifikována pomocí URI. Zdroje mohou být k dispozici ve více reprezentacích (jazykové varianty, datové formáty atd.)
● **Klient (client)** - Program, který navazuje spojení za účelem odeslání požadavku.
● **Uživatelský agent (user agent)** - Klient, který inicioval požadavek. Typicky jimi jsou prohlíže, editory, roboti atd.
● **Server** - Aplikace, která přijímá spojení a obsluhuje přes něj přijaté požadavky a odesílá zpět odpovědi. Program může být schopen chovat se jako klient i server, proto je označení vždy vztaženo ke konkrétnímu spojení.
● **Origin server** - Server, na kterém se zdroj nachází nebo bude vytvořen.
● **Spojení (connection)** - Virtuální cesta mezi dvěma aplikacemi vytvořená za účelem komunikace.
● **Entita (entity)** - Informace přenášená jako obsah *požadavku* nebo *odpovědi*. *Entita* se skládá z *metainformace* ve formě hlavičky entity a obsahu - těla *entity*
● **Reprezentace (representation)** - *Entita* v *odpovědi* podléhá *dojednání obsahu* (podrobněji viz sekce 12 RFC) a mohou existovat různé *reprezentace entity* odpovídající různým stavům *odpovědi*.
● **Dojednání obsahu (content negotiation)** - Mechanismus umožňující výběr vhodné *reprezentace entity* pro obsluhu *požadavku* (podrobně RFC sekce 12). Reprezentace *entit* v *odpovědích* může být dojednána vč. chybových *odpovědí*.
● **Varianta (variant)** - *Zdroj* může mít jednu nebo více *reprezentací* s ním spojených v libovolném okamžiku. Každá tato *reprezentace* se nazývá *varianta*, to však nutně neznamená, že *zdroj* podléhá *dojednání obsahu*.
● **Proxy** - Program, který se chová jako *klient* i *server* aby svým klientům zprostředkoval podání *požadavků*. Tyto jsou buď obslouženy interně nebo po případných transformacích předány dalším *serverům*.
● **Transparent proxy** - *Proxy*, které nemodifikuje *požadavky* ani *odpovědi*, krom toho, co je nutné pro úspěšnou autentizaci a identifikaci.
● **Non-transparent proxy** - *Proxy*, které, pro poskytnutí dalších služeb *uživatelskému agentovi*, modifikuje *požadavek* a/nebo *odpověď*.
● **Brána (gateway)** - *Server*, který je zprostředkovatelem pro další *servery*. Narozdíl od *proxy brána* přijímá *požadavky* jako by se jednalo o *origin server* pro požadovaný *zdroj*. *Klient* neví, že komunikuje pouze s *bránou*.
● **Tunel (tunel)** - Zprostředkující program, který umožňuje předání dat mezi dvěma *spojeními* bez jejich další analýzy. Jakmile je *tunel* vytvořen (aktivní), není považován za část *HTTP komunikace*. Vytvoření *tunelu* tak mohlo být iniciováno libovolným HTTP *požadavkem*. Tunel přestává existovat, když obě strany přenášeného *spojení* jsou uzavřeny.
● **Keš (cache)** - Lokální úložiště programu pro ukládání *odpovědí* a subsystém, který kontroluje ukládání, získávání a mazání *zpráv*. *Keš* ukládá *kešovatelné odpovědi* za účelem urychlení odpovědi a snížení zátěže sítě. *Keš* implementuje *klient* nebo *server* (a nikoliv *tunel*).
● **Kešovatelný (cacheable)** - *Odpověď* je *kešovatelná*, pokud je povoleno uložit ji do *keše* jako odpověď na následující požadavky. Pravidla pro určení, zda je *zdroj kešovatelný* jsou v sekci 13 RFC.
● **Z první ruky (first-hand)** - *Odpověď* je *z první ruky*, pokud přišla přímo z *origin serveru* (případně přes *proxy*) nebo byla
● **Příchozí/odchozí zpráva** - *Příchozí* resp. *odchozí zpráva* je taková, která putuje *spojením* směrem k *origin serveru* resp. ke *klientovi* (směr je určován vždy vzhledem k *origin serveru*).

HTTP – Syntaxe
● Požadavek - zjednodušená struktura požadavku

```
Request ::= RequestLine CRLF [Header CRLF]* [body]
RequestLine ::= Method RequestUri HTTPVersion
CRLF ::= CR LF
Header ::= Host | ...
Method ::= GET | HEAD | ...
```

● Server vždy musí podporovat metody GET a HEAD, ostatní jsou volitelné.

● Odpověď - zjednodušená struktura odpovědi

```
Response ::= Status-Line
                (header CRLF)*
                CRLF
                [ message-body ]
Status-Line ::= HTTP-Version Status-Code Reason-Phrase CRLF
```

HTTP – Stavové kódy

- Určují výsledek požadavku
  - číselně (pro automatické zpracování)
  - textově (pro snažší porozumění člověkem)
- První číslice kódu určuje třídu:
  1xx - Informational (informace) - požadavek přijat, zpracování pokračuje
  2xx - Success (úspěch) - požadavek přijat a plně zpracován
  3xx - Redirection (přesměrování) - další akce musí být provedeny pro dokončení požadavku
  4xx - Client Error (chyba klienta) - požadavek nemůže být zpracován (špatná syntaxe, …)
  5xx - Server Error (chyba serveru) - serveru se nepodařilo zpracovat validní požadavek

- Nejčastější kódy
- 100 – Continue - zpracování pokračuje
- 200 – OK
- 300 – Multiple Choices - výběr z možností (např. při dojednávání obsahu)
- 301 – Moved Permanently - trvale přesunuto
- 400 – Bad Request - neplatný požadavek
- 401 – Unauthorized - neautorizovaný přístup
- 403 – Forbidden - přístup zakázán
- 404 – Not Found - zdroj nenalezen
- 500 – Internal Server Error - vnitřní chyba serveru
- Pokud klient některému kódu nerozumí, považuje jej za kód x00

HTTP – Virtualizace

- Virtual hosting (VH) je technika umožňující na jednom serveru (např. webserveru) hostit více domén
- IP-based virtual hosting
  - Pro každou hostovanou doménu je vyhrazena jedna **nesdilena** IP adresa.
  - Lze použít s HTTP/1.0 i HTTP/1.1
- Name-based virtual hosting
- Na **jedne** IP adrese je provozováno více domén
- Používané protokoly pak musejí implementovat mechanismus umožňující rozlišení domén
- Lze použít pouze s HTTP/1.1
- *Obě techniky lze kombinovat*

HTTP – Určení hostitele

- HTTP 1.1

1. pokud server dostane absolutní URI, musí hlavičku Host ignorovat a použít název hostitele z URI
2. pokud dostane relativní URI a požadavek obsahuje hlavičku Host, použije název hostitele z hlavičky
3. jinak odpovídá: 400 Bad Request

- HTTP 1.0

1. pokud požadavek neobsahuje hlavičku Host, může se pokusit určit hostitele jinak

- Při směřování požadavku na proxy, je nutné zadávat absolutní URI
- Absolutní cesta v URI nesmí být prázdná, musí být alespoň /

HTTP – Autentizace

- Pokud jde požadavek přes proxy server, kde je třeba být autorizován, je potřeba do hlavičky požadavku přidat položku **Proxy-Authorization** , která bude obsahovat přihlašovací údaje. např. ve formátu

```
Basic base64[user:pass]
```

- *Příklad v shellu:*

```
# echo -n 'y36aws:tajneheslo' | uuencode -m -
begin-base64 644 -
eTM2YXdzOnRham5laGVzbG8=
====
```

  - Položka hlavičky by v tomto případě vypadala násldovně:
    Proxy-Autorization: Basic eTM2YXdzOnRham5laGVzbG8=

- Stejně funguje také HTTP Basic autorizace

Příklad - HTTP/1.1 - Relativní URI

```
 # telnet webing 80
Trying 147.32.80.114...
Connected to webing.
Escape character is '^]'.
```

```
GET / HTTP/1.1
Host: webing

HTTP/1.1 200 OK
Date: Tue, 09 Sep 2008 11:07:47 GMT
Server: Apache
Last-Modified: Sat, 10 Nov 2007 15:28:58 GMT
ETag: "1c09169a-303-43e94c2048280"
Accept-Ranges: bytes
Content-Length: 771
Content-Type: text/html

...

Connection closed by foreign host.
```

Příklad - HTTP/1.1 - Absolutní URI

```
 # telnet webing 80
Trying 147.32.80.114...
Connected to webing.
Escape character is '^]'.
GET http://webing.felk.cvut.cz/ HTTP/1.1
Host: webing.felk.cvut.cz

HTTP/1.1 200 OK
Date: Tue, 09 Sep 2008 11:17:44 GMT
Server: Apache
Last-Modified: Sat, 10 Nov 2007 15:28:58 GMT
ETag: "1c09169a-303-43e94c2048280"
Accept-Ranges: bytes
Content-Length: 771
Content-Type: text/html

...

Connection closed by foreign host.
```

Příklad - HTTP/1.0 - Relativní požadavek přes proxy - chyba HTTP 400

```
 # telnet proxy 80
Trying 147.32.80.13...
Connected to proxy.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.0 400 Bad Request
Server: squid
Mime-Version: 1.0
Date: Tue, 09 Sep 2008 11:43:16 GMT
Content-Type: text/html
Content-Length: 1420
Expires: Tue, 09 Sep 2008 11:43:16 GMT
X-Squid-Error: ERR_INVALID_URL 0
X-Cache: MISS from proxy.felk.cvut.cz
Via: 1.0 proxy.felk.cvut.cz (squid)
Proxy-Connection: close

...

Connection closed by foreign host.
```

HTTP – Přenos dat

- Spojení klienta se serverem může být
  - Separátní
  - Perzistentní

- Data mohou být obsažena v jedné odpovědi, nebo rozdělena do několika odpovědí (chunked), např. při:
  - Generování obsahu (ještě není známa velikost)
  - Pro více různých obsahů (např. obrázky)

- Chunked encoding
  - Data jsou rozdělena do několika bloků (chunks)
  - V hlavičce odpovědi musí být Transfer-Encoding: chunked
  - Tělo obsahuje jednotlivé bloky
  - Každý blok začíná svoji délkou zapsanou hexadecimálně.

- Chunked encoding

```
Chunked-Body ::=  chunk*
                  last-chunk
                  trailer
                  CRLF
chunk ::= chunk-size CRLF
          chunk-data CRLF
last-chunk ::= "0" CRLF
trailer ::= (entity-header CRLF)*
```

- Každá HTTP/1.1 aplikace musí být schopna přijmout a dekódovat chunks encoding.
  - Proxy server může spojit jednotlivé části do jedné

HTTP – Metody

- **GET** (bezpečná)

Požadavek o zdroj reprezentován URI (bez vedlejších efektů). V případě žádosti o zpracování – pouze výsledek, nikoli zdrojový kód. Požadavek může být podmíněný (cache) nebo částečný (partial)

- **HEAD** (bezpečná)

Identická metoda jako GET, server odesílá stejné informace, avšak bez těla odpovědi (slouží především pro testování)

- **POST**

Odeslání informací (v těle požadavku) na server ke zpracování

- **OPTIONS** (bezpečná)

Požadavek na informace o možnostech zdroje (URI) nebo serveru *

- **TRACE** (bezpečná)

Kopie požadavku (informace o změnách provedených na serverech vedoucích k cíli)

- **PUT** (idempotentní)

Nahrává data zdroje na serveru

- **DELETE** (idempotentní)

Maže zdroj na serveru

- **CONNECT**

Překládá požadavek na spojení na trasparentní TCP/IP tunel (obvykle při SSL komunikaci přes proxy server)

OPTIONS

Metoda umožňuje získat informace o zdroji nebo serveru.

- Dotaz na celý server místo na zdroj

```
 # telnet ewait.felk.cvut.cz 80

OPTIONS * HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 10 Sep 2008 13:13:05 GMT
Server: Apache
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Length: 0
Connection: close
Content-Type: text/plain
```

- Dotaz na zdroj, který má více variant

```
 # telnet ewait.felk.cvut.cz 80

OPTIONS http://ewait.felk.cvut.cz/y36aws/examples/03-mime/manual/manual/ HTTP/1.1
Host: ewait.felk.cvut.cz

HTTP/1.1 200 OK
Date: Wed, 10 Sep 2008 13:11:45 GMT
```

```
Server: Apache
Vary: accept-language,accept-charset
Allow: GET,HEAD,POST,OPTIONS,TRACE
Cache-Control: no-cache, no-store
Content-Length: 0
Content-Type: text/html
Content-Language: en
```

Webový server – implementace
Dostupná řešení
Implementací protokolu HTTP je k dispozici celá řada. Nejpoužívanější řešení podle počtu aktivních serverů sleduje Web Server Survey firmy Netcraft.

Mezi nejpoužívanější implementace patří:
- Apache httpd, Apache Tomcat
- Microsoft IIS
- Lighttpd

- **Apache httpd**
  - největší poddíl na trhu
  - modulární, portabilní (APR), rozšiřitelný
- **Apache Tomcat**
  - implmentuje specifikace Java Servlet a Java Server Pages (JSP) specifikace
  - poskytuje čistě „javovské“ prostředí HTTP serveru
- **Microsoft Internet Information Service (IIS)**
  - sada internetových služeb - HTTP(s), FTP, SMTP, NNTP server
  - druhý nejrozšířenější webserver
  - modularizace, XML konfigurace
  - aplikační servery a servlet containery pro Java Enterprise technologii,
  - HTTP moduly pro různé programovací jazyky
- **Lighttpd**
  - malý, vyhovující standardům, bezpečný, flexibilní, výkonný server, modulární
  - virtualhosting, CGI, SSI, SSL/TLS, autentizace proti LDAP, WebDAV

Existuje řada dalších řešení, jak obecných, tak zaměřených určitým směrem.
- aplikační servery a servlet containery pro Java Enterprise technologii,
- embedded servery s nízkými nároky na zdroje,
- HTTP moduly pro různé programovací jazyky,
- …

Apache httpd – Historie
- 1993: NCSA HTTPd (National Center for Supercomputing Applications)
- 1994: Odchod hlavního programátora, založení emailové konference, koordinace patchů
- 1995: 1. veřejná verze 0.6.2
- 1996: Nejúspěšnější web server
- 2002: Verze 2.0 - přepsání kódu verze 1.3 s ohledem na modularizaci a nezávislou přenositelnou vrstvu (Apache Portable Runtime), UNIX vlákna, podpora ne-UNIXových systémů , nové Apache API a podpora IPv6

Apache httpd – Vlastnosti
- Open source
- Dostupnost na mnoha systémech (APR)
- Velká konfigurovatelnost pomocí souborů
- Podpora jazyků (PHP, Perl, Python, TCL, JSP, SSI,…)
- Podpora rozhraní CGI / FastCGI
- Virtualhosting
- HTTP autentizace
- Proxy server
- Přizpůsobitelné logování
- přepisování URL (rewrite)
- filtrování vstupu/výstupu
- Dojednávání obsahu
- Podpora SSL, TLS a komprese

Apache httpd – Architektura
- Prostředí pro běh serveru (Apache Portable Runtime)
- Moduly souběžného zpracování (Multi-Processing Module)
  - Prefork - Procesy bez vláken (pro vláknově ne-bezpečné moduly – např. PHP)
  - Worker - Procesy a vlákna (vyšší výkon)
- Core
- Další moduly (Auth, Autoindex, Alias, CGI, Include, Info, Mime, Status, Userdir, ...)
  - Filtry – vstupní / výstupní
  - Handlery – zpracování požadavků

Apache httpd – Zpracování dotazu
- Překlad URI na jméno souboru
- Kontrola přístupu
  - autorizace jména uživatele
  - kontrola práv pro přístup uživatele
  - kontrola přístupu založená na jiných kritériích (např. podle IP adresy)
- Zjištění MIME typu dotazu
- Fixups – co se doposud nestihlo udělat
- Zaslání požadovaných dat klientovi
- Zalogování přístupu

Apache httpd – Instalace
- **Konfigurace → Kompilace → Instalace**
  - `./configure && make && make install`
- Konfigurace
  - `./configure --help`
  - `--enable-layout=LAYOUT`
  - `--prefix=DIR`
  - `--enable-FEATURE / --disable-FEATURE`
    zahrne/nezahrne danou vlastnost do instalace
  - `--enable-modules=MODULE-LIST`
  - `all | most | module1,..,moduleN`
    mastavení modulů
  - `--with-PACKAGE / --without-PACKAGE`
    použije/nepoužije balík

Apache httpd – Adresářová struktura
/etc/apache2/ - globální konfigurace
/etc/apache2/modules.d - konfigurace modulů serveru
/etc/apache2/ssl - klíče/certifikáty
/etc/apache2/vhosts.d - konfigurace virtuálních hostů
/usr/lib64/apache2/ - zkompilované knihovny
/usr/lib64/apache2/modules - zkompilované moduly
/usr/sbin - binární soubory
/usr/share, - dokumentace, manuálové
/usr/share/man stránky, příklady
/var/log/apache2 - logy
/var/www - webové stránky
/var - v dalších podadresářích další soubory, zámky ...

Moduly
Apache httpd lze rozšiřovat pomocí modulů. Nejběžnější moduly (autentizace, autorizace, SSL atp.) jsou přímo součástí archivu se zdrojovými kódy. Pomocí voleb *configure skriptu* lze tyto zařadit/vyřadit z kompilace a instalace.
Moduly mohou být linkovány
- staticky se základní instalací nebo
- linkovány dynamicky (DSO Dynamic Shared Objects) - tímto způsobem lze snadno přidávat další moduly vč. modulů třetích stran. Instalace obsahuje podůrné nástroje pro podporu tohoto procesu.

Nastavení volitelných modulů a jejich vlastností:
- `–enable-`*`FEATURE`*`, –disable-`*`FEATURE`* - zahrne/nezahrne danou vlastnost do instalace
- `–enable-modules=`*`MODULE-LIST`* `- all | most | module1,..,moduleN -` Nastavení MPM (Multi-processing moduly) a modulů třetích stran:
- `–with-`*`PACKAGE`*`, –without-`*`PACKAGE`* - použije/nepoužije balík (MPM, SSL …)

Po kompilaci a instalaci lze zjistit jaké staticky linkované moduly jsou k dispozici:

```
 y36aws ~ # apachectl -l
Compiled in modules:
  core.c
  worker.c
  http_core.c
  mod_so.c
```

Zjištění modulů načítaných do serveru lze zjistit pomocí přepínače -M. POZOR, bere v úvahu pouze konfigurační soubory, nikoliv běžící instanci serveru.

```
 y36aws ~ # apachectl -M
Loaded Modules:
 core_module (static)
 mpm_worker_module (static)
 ...
 alias_module (shared)
 auth_basic_module (shared)
```

V distribuci Gentoo GNU/Linux s balíkem =www-servers/apache-2.2.9-r1 není příkaz apachectl k dispozici v běžné podobě (apache2ctl je link na init skript), místo něj lze stejným způsobem přímo použít příkaz apache2 (pozor, bez parametrů spustí www server!).

Apache httpd – Spuštění

Start a zastavení serveru se provádí nástrojem apachectl s parametrem `-k AKCE`. `AKCE` může být `START, STOP, RESTART` a další (viz man). Pokud konfigurace serveru obsahuje podmíněné bloky `<IfDefine>`, které mají být zpracovány, je třeba parametrem `-D` tyto definovat a tak povolit.

● Manuální ovládání
- `apache2ctl -k start -D PHP -D SSL -D DAV -D SVN`
- `apache2ctl -k stop -D PHP -D SSL -D DAV -D SVN`
- `apache2ctl -k test -D PHP -D SSL -D DAV -D SVN`

● Kontrola konfigurace

Pomocí příkazu `apachectl` lze též ověřit, zda zamýšlená konfigurace neobsahuje syntaktické chyby.
- `apache2ctl -t`
- `apache2ctl -t -D SSL -D PHP5 -D USERDIR -D INFO`

```
y36aws ~ # apachectl -t
Sytax OK
```

Konfigurace může obsahovat podmíněné bloky `<IfDefine NAZEV_BLOKU>`, které se provedou pouze v případě, že při startu serveru byl pomocí přepínače `-D` definován název bloku. Tuto skutečnost musíme zohlednit i při testování konfigurace a pokud chceme testovat podmíněné bloky, je nutné přepínačem `-D` též definovat příslušné názvy.

```
y36aws ~ # apachectl -t -D SSL -D PHP5 -D USERDIR -D INFO
[Mon May 26 15:48:49 2008] [warn] The Alias directive in
     /etc/apache2/modules.d/00_mod_autoindex.conf
at line 5 will probably never match because it overlaps an earlier Alias.
```

● Kontrola log souborů
- `tail -f /var/log/apache2/{error,access}_log`

Důležité informace o startu, běhu a zastavení serveru jsou zapisovány do logu. Ten se typicky nalézá v adresáři `/var/log/[httpd,apache2]` a je rozdělen do několika souborů, z nichž nejdůležitější je:

- `error.log` - informace o startu a zastavení serveru, závažné chyby; lze rozdělit na obecné informace a log hlavního serveru a *virtual hostů*
- `access.log`, `error.log` a další - nastavitelný formát, uživatelsky definovány

```
y36aws ~ # tail -f /var/log/apache2/{error,access}_log
==> /var/log/apache2/error_log <==
[Wed Sep 24 15:58:27 2008] [error] [client 147.32.80.98] client denied by server
configuration: /var/www/y36aws/03-moduly/total.png
[Wed Sep 24 15:59:10 2008] [error] [client 147.32.80.98] client denied by server
configuration: /var/www/y36aws/03-moduly/total.png
```

```
==> /var/log/apache2/access_log <==
80.154.42.59 - - [09/Sep/2008:17:56:31 +0200] "GET /admin/phpMyAdmin-2.6.4-
rc1/main.php HTTP/1.0" 403 1015
147.32.80.13 - - [10/Sep/2008:13:06:56 +0200] "GET /favicon.ico HTTP/1.0" 403 1024
147.32.80.13 - - [10/Sep/2008:13:06:59 +0200] "GET /favicon.ico HTTP/1.0" 403 1024
147.32.80.13 - - [10/Sep/2008:13:07:11 +0200] "GET /y36aws/examples/03-log/
HTTP/1.0" 403 1061
```

● Automatické ovládání
  ● **Init skripty – volání příkazu apache2ctl ...**

UNIXové operační systémy podporují automatické spouštění a zastavování démonů (služeb) pomocí *init skriptů*. Pro ovládání *httpd Apache* lze použít třeba následující. Typicky se využívá nástroj apachectl s přepínačem `-k AKCE`, kde `AKCE` může být `start, stop, restart` (další viz man).

```
#!/bin/bash
# Zakladni initscript

### zpracovani pozadavku
case $1 in
    start)
        apachectl -k start -D PHP -D SSL -D DAV -D SVN -D CGI
        return $?
        ;;
    stop)
        apachectl -k stop -D PHP -D SSL -D DAV -D SVN -D CGI
        return $?
        ;;
    restart)
        apachectl -k restart -D PHP -D SSL -D DAV -D SVN -D CGI
        return $?
        ;;
    *)
        echo "Pouziti: $0 start|stop|restart"
        return 2
esac
```

Virtual hosting
Virtual hosting je technika, která umožnuje na jednom serveru (např. webserveru) hostit více domén. V určitých případech je postačující i jedna IP adresa. Možné realizace této techniky jsou:
● ***IP-based virtual hosting*** - pro každou hostovanou doménu je vyhrazena jedna **nesdílená** IP adresa - server je vybaven více fyzickými, virtuálními rozhraními nebo jejich kombinací.
Techniku lze použít s HTTP/1.0 i HTTP/1.1
● ***Name-based virtual hosting*** - na jedné IP adrese je provozováno více domén. Používané protokoly pak musejí implementovat mechanismus umožňující rozlišení domén.

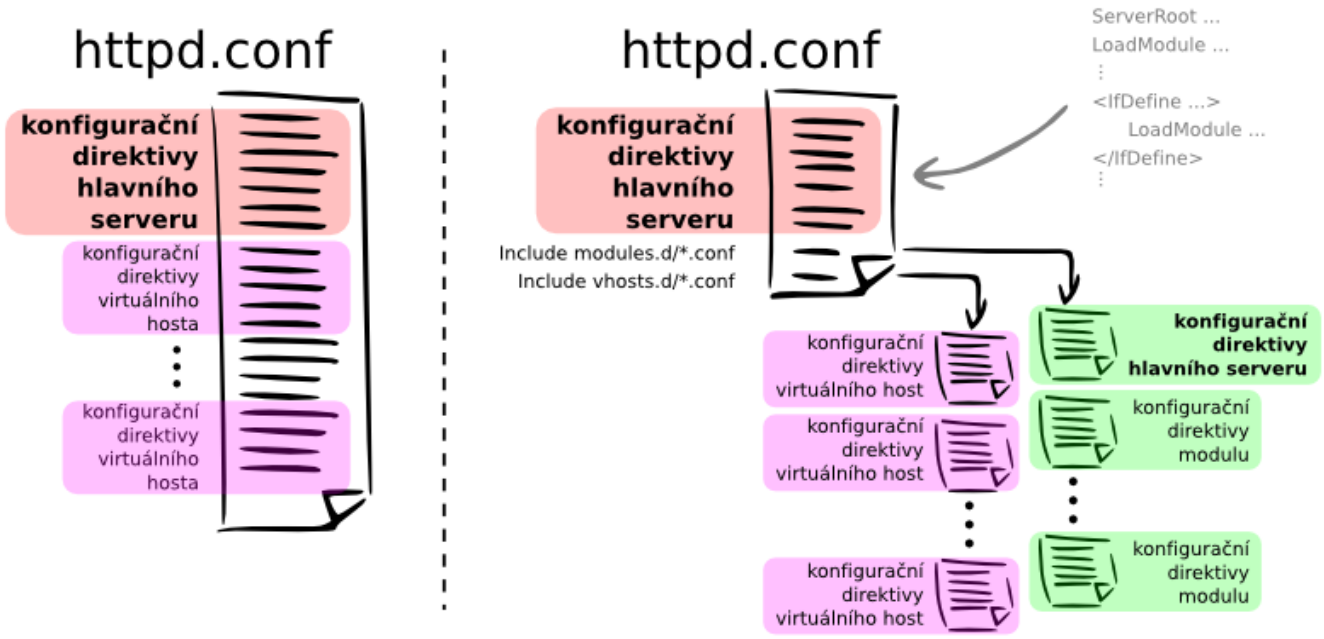
- Techniku **nelze** použít s HTTP/1.0 ale pouze s HTTP/1.1, kde je podpora rozlišení domén realizována Hlavičkou `Host:`.
- Nelze použít SSL pro více domén - SSL handshake předchází určení domény a nelze určit jaký certifikát použít.

02 - Konfigurace serveru Apache, testování

Obsah
- Struktura konfiguračnich souborů
- Syntaxe
- Webspace → Filesystem
- Konfiguračni minimum
- Další možnosti konfigurace
- Konfigurace virtual hosts

Apache httpd se konfiguruje pomocí textových konfiguračních souborů, do kterých se umísťují *direktivy*. Typicky je hlavním konfiguračním souborem `httpd.conf`. Cestu k a název tohoto souboru lze nastavit při kompilaci (`-sysconfdir`), případně pomocí parametru `-f` při startu serveru. Konfigurace se načítá pouze při startu serveru, proto je třeba server pro provedení změn třeba restartovat.
Před úpravou konfiguračních souborů se doporučuje provést jejich zálohu.
Konfiguraci lze uložit do více souborů a propojit je direktivou `Include` (viz dále). Tato direktiva podporuje *wildcards* (např. *conf označuje všechny soubory, jejichž název končí na conf).

Struktura konfiguračnich souborů



Je možné mít celou konfiguraci serveru uloženou v jediném konfiguračním souboru (viz obrázek vlevo). Jelikož neexistuje vnitřní dělení souboru, je možné že se časem stane tento soubor nepřehledným (s množstvím přibývajících nastavení a virtuálních hostů). Proto je vhodné rozdělit konfiguraci mezi více menších souborů.

V distribuci Gentoo Linux je rozdělení následující (viz obrázek vpravo):
- hlavní soubor `httpd.conf` obsahuje pouze:
  1. direktivu `ServerRoot`
  2. načítání (podmíněné pomocí Define -D xxx) modulů
  3. vložení dalších konfiguračních souborů (pomocí `Include`)
- adresář `modules.d` obsahuje soubory s příponou `.conf`, které se načtou v abecedním pořadí (proto jsou na začátku názvu souboru čísla) a obsahují globální nastavení serveru i konfiguraci jednotlivých modulů (podmíněnou jejich přítomností)
- adresář vhosts.d obsahuje soubory s příponou `.conf`, které se načtou v abecedním pořadí (proto jsou na začátku názvu souboru čísla) a obsahují nastavení virtual hostů

Syntaxe konfiguračních souborů
- Jedna direktiva na řadce
- Při rozděleni direktivy na vice řadků – posledni znak ****
- Direktivy – case insensitive, jejich parametry (např. cesty) – mohou byt case sensitive
- Komentař – řadek začinajici znakem #
- Komentaře by neměly byt spolu s direktivami
- Prazdne řadky a bile znaky před direktivami – ignorovany

Platnost direktiv (scope)

● Direktivy mimo kontejner – tj. v hlavním konfiguračním souboru (resp. v připojených souborech) - plati pro cely server
● Platnost direktivy lze omezit vložením do sekci (kontejnerů), ktere omezi jejich platnost na
  ● Část souboroveho systemu (cesty):
    `<Directory> <DirectoryMatch> <Files> <FilesMatch>`
  ● Zdroje (URL):
    `<Location> <LocationMatch> <Proxy>`
  ● Website (pokud je server s podporou virtual hostingu):
    `<VirtualHost>`
● Některe direktivy nemaji smysl v některych sekci (např. nastaveni obsluhy požadavků, vytvařeni vlaken/procesů)
● Je nutne dodržovat kontext direktiv (podle dokumentace)

Soubor .htaccess

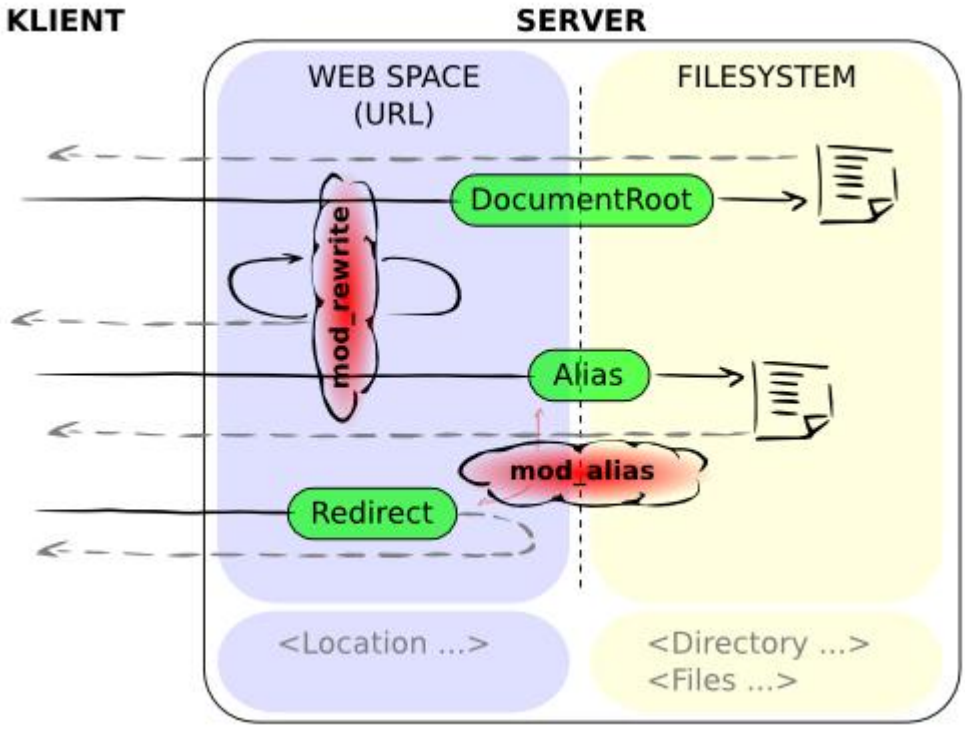
Apache httpd umožňuje *direktivy* vložit do speciálních souborů, které se umístí do adresářů exportovaných na web. Nastavení v těchto souborech platí pro daný adresář a všechny jeho podadresáře. *Direktivy* musí být možné použít v kontextu sekce `<Directory>` a podobných.

● Umistěni v adresařich exportovanych na web
● Konfigurace plati pro dany adresař a jeho podadresaře
● Direktivy, použitelne v kontejneru (např. `<Directory>`)
● Vyhodnocovani při každem požadavku do adresaře
● Nastaveni použiti `.htaccess` souborů
  ● direktiva `AccessFileName` – název souboru, ve kterém se bude hledat, kontext: cely server, vhost
    `Default: .htaccess`
  ● direktiva `AllowOverride` – určuje, jaké skupiny direktiv je možné používat, kontext: `<Directory>` apod.
    `None, All, Typ_direktivy...`
  ● Soubory se načitaji postupně od */* (pokud to neni zakazano)

Výhodou použití `.htaccess` souborů je decentralizace konfigurace, možnost zásahu do konfigurace (např. v případě webhostingu). Nevýhodou může být zvýšená zátěž serveru, jelikož se musí příslušné soubory načíst a zpracovat.

Webspace → Filesystem

Možnosti mapování mezi *webspace* a filesystémem a zároveň direktivy, kterými lze příslušné kontejnery nastavit ukazuje následující obrázek.



Webspace vs. Filesystem
Nejčastěji používané *kontejnery* jsou ty, které upravují nastavení *filesystemu* nebo *webspace*.

- ***Filesystem*** - adresářová struktura tak, jak ji vidí operační systém serveru.
- ***Webspace*** - struktura webu taková, jakou ji prezentuje *webový server klientovi*. Obsah *webspace* se nutně nemusí mapovat na *filesystém*, může se jednat o dynamicky vytvářené zdroje.

Kontejnery - konfigurační sekce kontejnery podminek
Direktivy mohou platit pro celý sever, nebo být omezeny na určité sekce. Pro omezení platnosti se využívají *direktivy* označované jako konfigurační sekce - *kontejnery*. Rozlišují se dva základní typy *kontejnerů* podle toho, zda se vyhodnocují při každém požadavku a mezi ně patří většina *kontejnerů*, nebo pouze při startu (a restartu) serveru.

`<IfDefine>, <IfModule>, <IfVersion>`
Pouze tyto tři *kontejnery* jsou vyhodnocovány při startu nebo restartu serveru. Konfigurační direktivy v nich obsažené budou použity pouze v případě, že podmínka je v době startu (restartu) pravdivá.

● Omezuji platnost direktiv
● Vyhodnocuji se při re/startu

● `<IfDefine` *`název`*`> ... </IfDefine>`
  ● Při definici nazvu pomoci přepinače `-D` *`název`*
  ● Povoluje / zakazuje vlastnosti serveru
  Konfigurace uvnitř *kontejneru* `<IfDefine nazev>…</IfDefine>` bude načtena pouze v případě, že při startu serveru byl příslušný název definován přepínačem -D. Tímto způsobem lze snadno povolit/zakázat některé vlastnosti serveru atp.
● `<IfModule` *`jméno_modulu`*`> ... </IfModule>` - Při načteni modulu do serveru
Apache httpd je *modulární server* a jeho moduly lze načítat dynamicky. Tyto moduly rozšiřují jeho funkcionalitu, nicméně *direktivy*, které poskytují, nemusí vždy být k dispozici.
Kontejner `<IfModule module_name>…</IfModule>` umožňuje načíst část konfigurace pouze v případě, že požadovaný *modul* určený svým názvem module_name je načtěn v serveru.

● `<IfVersion` *`[<>=]x.y.z`*`> ... </IfVersion>` - Podle verze serveru
Pomocí *kontejneru* `<IfVersion [<>=]x.y.z>…</IfVersion>` lze omezit direktivy tak, aby se provedly pouze pokud je verze serveru menší, větší nebo rovna zadané verzi.

Filesystémové kontejnery `Directory, DirectoryMatch`
Kontejnery `<Directory>` resp. `Files` omezují platnost direktiv na zvolené adresáře resp. soubory. V názvech lze používat shellové vzory (*, ?, [ ] apod.). Stejného efektu lze dosáhnout pomocí `.htaccess` souborů.

Ukázka výpisu obsahu adresáře (`Option +Indexes`) pro adresář `/var/www/htdocs` a jeho podadresáře.
```
<Directory /var/www/htdoc>
  Options +Indexes
</Directory>
```

Ukázka zakazuje přístup k souborům končícím na `txt`.
```
<Files *txt>
  Order allow,deny
  Deny from all
</Files>
```

1. Všechny kontejnery `<Directory vzor>` v pořadi od nejkratši shody po nejdelši
● Např. pro adresař `/home/*/public_html`
  1. `/`
  2. `/home`
  3. `/home/user`
  4. `/home/user/public_html`
2 .Soubory .htaccess
3. Všechny kontejnery `<Directory ~ ERE>` a `<DirectoryMatch ERE>` v pořadi podle uvedeni v konfiguraci

Kontejner `<Directory />` je povolen po všechny, dostupny je tudiž cely filesystem – obvykle nutno zakazat !
Existují direktivy `<DirectoryMatch>` a `<FilesMatch>`, kde název může být regulární výraz (PCRE - Perl Compatible Regular Expression).

Webspace kontejnery `Files, FilesMatch, Location`
Kontejner `<Location>` umožňuje použít dané direktivy pouze pro určitou část *webspace* určenou relativní URL vzhledem ke kořenu webu. Opět v URL lze použít shellové vzory a ve variantě `<LocationMatch>` regulární výrazy (PCRE).

Konfigurační sekce jsou aplikovány v přesně daném pořadí, které je dobré znát, aby se zamezilo nechtěným vedlejším efektům. Pořadí zpracování:

1. `<Directory>` a *`.htaccess`* - má přednost (překrývá <Direcotry>
2. `<DirectoryMatch>` a `<Directory ~>`
3. `<Files>` a `<FilesMatch>`
4. `<Location>` a `<LocationMatch>`

Pořadí zpracování direktiv kromě `<Directory>` se dále řídí pořadím, ve kterém se vyskytly v konfiguračním souboru. U `<Directory>` zpracování probíhá od nejkratší cesty k nejdelší (`/ → /var/ → /var/www/…`).V případě, že *kontejner* `<Directory>` vyskytne vícekrát pro jeden a tentýž adresář, zpracování probíhá v pořadí výskytu v konfiguračním souboru.

1. Po zpracovani kontejnerů `<Directory...>`
2. Všechny kontejnery `<Files...>` v pořadi podle uvedeni v konfiguraci
   - `<Files vzor>`
   - `<Files ~ ERE>`
   - `<FilesMatch ERE>`

   Může byt současti kontejneru `<Directory>`

3. Aplikuje v připadně shody vzoru (ERE) s URL
   - `<Location vzor>`
   - `<Location ~ ERE>`
   - `<LocationMatch ERE>`

   Ukázka - uživateli není povoleno (kód 403) ke zdroji pomocí URL http://server/dir. V případě, že *filesystém* je case-insensitive (typická situace v OS MS Windows), názvy dir a DIR jsou shodné. Když tedy uživatel zadá URL http://server/DIR, přístup mu bude povolen (kód 200).

```
<Location /dir>
  Order allow,deny
  Deny from all
</Location>
```

Vzory, vyrazy, ...
- Vzory – wildcards (shellove)
  - **?**, *****, **[ ]**
  - Znak / se nenahrazuje, musi by explicitně uveden
- Vyrazy – extended regular expressions - **.** , **?**, *****, **+**, **|**, **^**, **$**, ...
- Cesty obsahujici znaky // mohou, ale nemuseji byt shodne s /
  - `/adr1//adr2 → /adr1/adr2`
  - `//adr !~ ^/adr`

Možnosti – Options
Povoluje / zakazuje možnosti v určenych adresařich
- Syntaxe: `Options [+|-]option...`
  - **+** přidava, **–** odebira
  - Bez znamenka nastavuje pouze uvedene vlastnosti
  - Vlastnosti s/bez znamenka se nesmi kombinovat
- Default: `Options All`
- Kontext: server, virtual host, directory, .htaccess

Konfiguračni minimum
- ServerRoot (core) = kořenovy adresař s instalaci serveru (--prefix)

  `ServerRoot "/usr/lib64/apache2"`
  - Kořenový adresář s instalací serveru
  - Při konfiguraci lze nastavit parametrem –prefix
  - Syntaxe: `ServerRoot` *`cesta_k_adresáři`*
  - Default: `ServerRoot /usr/local/apache`
  - Kontext: `server`

● Listen (mpm) = IP adresa/jmeno, TCP port serveru (je možno použit vicekrat)

```
Listen 80
```

- IP adresa/jméno a TCP port, na kterém server přijímá požadavky
- Při konfiguraci lze nastavit parametrem –port
- Syntaxe: `Listen` [*adresa:/ /*]*port* [*protokol*]
- Kontext: `server`

● ServerName (core) = jmeno serveru pro odpovědi (např. při přesměrovani)

```
ServerName www.example.com
```

- Jméno serveru (a port), kterým se identifikuje klientovi
- Pokud se jméno neuvede, provede se reverzní dotaz na DNS
- Pokud se neuvede port, použije se stejný jako u příchozího požadavku
- Syntaxe: `ServerName` [*scheme:*]*doménové_jméno*[*:port*]
- Kontext: `server, virtual host`

● User / Group (mpm) = Indetita uživatele / skupiny pro běh serveru v připadě, že je server spuštěn rootem

```
User apache
Group apache
```

- Identita uživatele/skupiny, procesů obsluhujících požadavky v případě, že server byl spuštěn rootem
- Syntaxe: `User` *uživatelské_jméno*
- Syntaxe: `User` *#UID*
- Default: `User` #-1
- Kontext: `server`

● DocumentRoot (core) = mapovani kořene webu na filesystem

```
DocumentRoot "/var/www/localhost/htdocs"
```

- Mapování kořene webu na filesystém (bez posledního /)
- Pokud neobsahuje absolutní cestu, je vztažena k `ServerRoot`
- Netýká se direktivy `Alias`
- Syntaxe: `DocumentRoot` *adresář*
- Default: `DocumentRoot /usr/local/apache/htdocs`
- Kontext: `server, virtual host`

<Directory> resp. **<DirectoryMatch>**

- omezení platnosti direktiv na adresář
- absolutní cesta nebo lze využít shellových vzorů resp. regulárních výrazů (PCRE)
- Syntaxe: `<Directory /cesta/k/adresari> … </Directory>` resp. `<DirectoryMatch regexp> … </DirectoryMatch>`
- příklad - zakázání přístupu do kořenového adresáře *filesystemu*

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>
```

<Files> resp. **<FilesMatch>**

- omezení platnosti direktiv soubor resp. soubory
- název jako shellový vzor resp. regulární výraz (PCRE)
- Syntaxe: `<Files /cesta/k/souboru> … </Files>` resp. `<FilesMatch regexp> … </FilesMatch>`
- příklad - zakázání přístupu k souborům začínajícím na .ht

```
<FilesMatch "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</FilesMatch>
```

<Location> resp. **<LocationMatch>**

- omezení platnosti direktiv na URL - relativní URL vzhledem ke kořenu webu
- Syntaxe: `<Location URL> … </Location>` resp. `<LocationMatch regexp> … </LocationMatch>`

Options

- direktiva udává vlastnosti, které budou k dispozici
- Core; kontext: server, vhost, directory, .htaccess
- pro snažší „dědičnost“ nastavení má modifikátory + resp. -, které umožňují přidat resp. odebrat vlastnost.
- některé vlastnosti: All (všechny vlastnosti), ExecGCI, Indexes (listování adresáře), Includes (SSI)
- příklady

```
# pridani a odebrani vlastnosti z konfigurace
Options +Indexes -FollowSymlinks
# pevne nastaveni vlastnosti
Options Indexes FollowSymlinks
```

● Načitani modulů (mod_so)

```
LoadModule dir_module modules/mod_dir.so
```

- načtení rozšiřujícího modulu
- Syntax: `LoadModule nazev_modulu cesta/k/modulu`
- cesta může být buď absolutní nebo relativní vůči ServerRoot

● Vychozi stranka (mod_dir)

```
DirectoryIndex index.html index.htm
```

- nastavení výchozí stránky (resp. souboru). Lze zadat seznam oddělený mezerou, soubory jsou hledány v pořadí v seznamu.
- kontext: `server, vhost, directory, .htaccess`

● Automaticky generovany index (mod_autoindex)

```
Options +Indexes
IndexOptions FoldersFirst IgnoreCase
IndexIgnore README .htaccess *.bak *~
AddIcon (IMG,/icons/image.xbm) .gif .jpg .xbm
```

● Nastaveni znakove sady a jazyků (core)

```
AddDefaultCharset utf-8
```

- nastavení znakové sady a jazyků (core)
- Core, kontext: `server, vhost, directory, .htaccess`
- Syntax: `AddDefaultCharset charset`

● Logovani (core)

```
ErrorLog /var/log/apache2/error_log
LogLevel error
HostnameLookups off
```

- **ErrorLog**
  - standardní formát, vazba na analyzátory, relativní cesta vůči ServerRoot
  - Core, zajišťuje základní logování
  - lze nastavit pro každý vhost, zprávy pak již nejsou logovány do logu hlavního serveru
  - Syntax: `ErrorLog /cesta/k/logu | syslog[:facility]`

● Logovani – vlastni (mod_log_config)

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
```

● Autorizace (mod_authz_host) = Určuje pořadi zakazů a povoleni

Modul poskytující autorizaci na základě IP adresy resp. doménového jména klienta.

```
Order Deny, Allow
Allow From ...
Deny From ...
```

Order

- určuje pořadí, v jakém se budou aplikovat *direktivy* `Allow from` a `Deny from`
- kontext: `directory, .htaccess`
- Syntax: `Order allow,deny` (nejdříve všechny `allow` pak `deny`) nebo `Order deny,allow` (naopak).

Allow from resp. **Deny from**

- Určuje, ktery klient je resp. neni autorizovan
- IP adresa - `147.32.80.90`
- část IP adresy - `147.32.80`

- síť (CIDR) - `10.0.0.0/8`
- domenove jmeno - `mypc.example.org`
- část domenoveho jmena - `example.org .net`
- všichni - vyhrazeno slovo `all`

mod_alias

Nastavení mapování mezi *webspace* a *filesytemem* - není pak nutné, aby všechny dokumenty byly uloženy v DocumentRoot.

- **Alias** resp. **AliasMatch**
  - namapuje zvolený adresář na URL
  - Syntax: `Alias /relativni/URL /cesta/na/filesystemu`
  - příklad - `Alias /image /ftp/pub/image`

- **Redirect** resp. **RedirectMatch**
  - umožňuje přesměrování požadavku na jinou URL
  - Syntax: `Redirect [status] /relativni/URL /cesta/na/filesystemu`
    - status: permanent (301) | temp (302, výchozí) | seeother (303) | gone (410)
  - příklad - `Redirect /service http://foo2.example.com/service`

mod_mime

Modul umožňuje nastavit mapování MIME typů podle přípon souborů a výchozí MIME typ.

- **DefaultType**
  - určuje, jaký MIME typ bude vložen do hlavičky `Content-type` v případě, že se jej nepodaří určit
  - od verze 2.2.7 je k dispozici volba none, tj. určení typu obsahu je na *klientovi* - hlavička Content-type nebude vložena
  - Syntax: `DefaultType MIME | none`

Konfigurace virtual hosts

● Kontejner `<VirtualHost>`
- určuje nastavení jednotlivých vhostů
- Syntax: `<VirtualHost addr[:port] [addr[:port]] ... > ... </VirtualHost>`
- addr může být IP adresa nebo DNS název, což se nedoporučuje

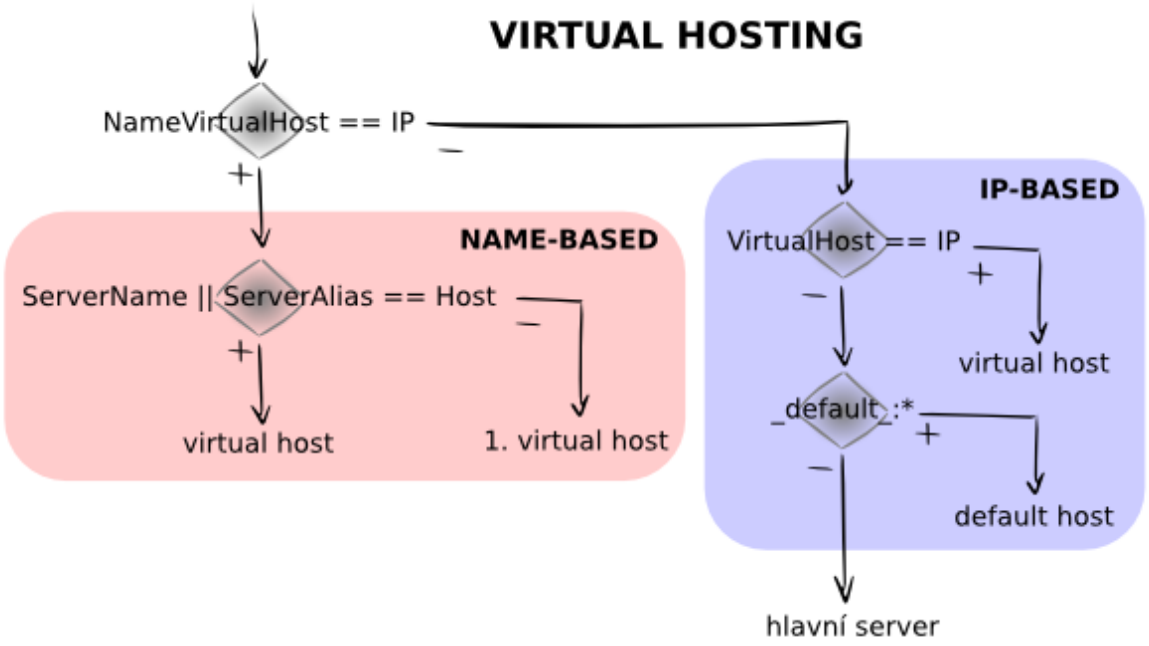
● IP-based virtual hosting
- `ServerName`
- `ServerAlias`
- `DocumentRoot`
- pokud nebude nalezen odpovidajici ip-vh, použije se _default_, pokud neni ani ten, použije se nastaveni hlavniho serveru

● Name-based virtual hosting
- určuje, které IP adresy, na kterých server poslouchá jsou určeny pro name-based virtual hosting
- nutno dopředu definovat IP adresy použite pro name-based vh
- `NameVirtualHost IP[:port]`
- je nutne specifikovat jmeno serveru
- jinak shodne s ip-vh

Rozlišení jednotlivých `name-based` virtuálních hostitelů se provádí podle direktivy `ServerName` v konfiguraci každého vhosta (viz příklad).
Výběr vhosta probíhá podle algoritmu na následujícím obrázku (předpokládá se, že server na požadované IP adrese a portu poslouchá).

Pokud převedeme celý server na name-based virtual hosting (`NameVirtualHost *:80`), je vhodné jako prvního virtuálního hostitele uvést `<VirtualHost *:80> … </VirtualHost>`.

Vyběr virtualniho hostitele




```
# --- IP BASED VH ---
# MY CORP.
<VirtualHost 10.0.254.1:80>
   ServerName www.mycorp.k328
   DocumentRoot "/var/www/vhost/mycorp"
   <Directory "/var/www/vhost/mycorp">
      Options FollowSymLinks
      AllowOverride None
      Order deny,allow
      Allow from all
   </Directory>
</VirtualHost>

#<VirtualHost _default_:80>
#</VirtualHost>

# --- NAME BASED VH ---
# rikame, ze IP se pouzije pro n-vh
NameVirtualHost 10.0.254.2:80
# OTHER CORP. 1
<VirtualHost 10.0.254.2:80>
   # teprve tady rikam, o jaky vhost se jedna!
   ServerName www.othercorp1.k328
   DocumentRoot "/var/www/vhost/othercorp1"
   <Directory "/var/www/vhost/othercorp1">
      Options FollowSymLinks
      AllowOverride None
         Order deny,allow
         Allow from all
   </Directory>
</VirtualHost>

# OTHER CORP. 2
<VirtualHost 10.0.254.2:80>
   ServerName www.othercorp2.k328
   DocumentRoot "/var/www/vhost/othercorp2"
   <Directory "/var/www/vhost/othercorp2">
      Options FollowSymLinks
      AllowOverride None
      Order deny,allow
      Allow from all
```

```
   </Directory>
</VirtualHost>

10.0.254.1      www.mycorp.k328
10.0.254.1      www.aliascorp.k328
10.0.254.2      www.othercorp1.k328
10.0.254.2      www.othercorp2.k328
10.0.254.2      www.othercorp3.k328
10.0.254.3      www.nocorp.k328
```
/etc/hosts

V případě nastavení výše budou výsledky algoritmu výběru virtuálního hostitele k následující:

- požadavek na www.mycorp.k328 a www.aliascorp.k328 povedou na IP-based virtual hosting a bude zvolen hostitel 10.0.254.1
- požadavek na www.othercorp[12].k328 povede na name-based virtual hosting a podle hlavičky Host: se zvolí odpovídající *kontejner*
- požadavek na www.othercorp[12].k328 povede na name-based virtual hosting, ale jelikož neexistuje odpovídající kontejner, vybere se první, tj. se ServerName www.othercorp1.k328
- požadavku www.nocorp.k328 neodpovídá žádný z virtuálních hostitelů, nicméně server na dané IP adrese poslouchá a proto odpoví podle nastavení hlavního serveru

Apache httpd – Moduly - Obsah
- Přehled typů modulů
- Práce s moduly
- Moduly pro souběžné zpracování (MPM)
- Moduly pro logování a informace o serveru
- Modul pro práci s MIME informacemi
- Vyjednávání obsahu

Přehled typů modulů
- Systémového prostředí
- Autentizace a řízení přístupu
- Dynamického generování obsahu
- Konfigurace typu obsahu
- Adresářových výpisů
- Hlaviček odpovědí
- Informací o serveru a logování
- Mapování URL

Práce s moduly
- Dynamické načítání modulů (mod_so)

LoadModule identifikátor cesta_k_modulu
- Identifikátor je definovaný uvnitř každého modulu
- Cesta je relativně k ServerRoot, název modulu obvykle s příponou .so
- Kontext: server, obvykle následuje použití kontejneru `<IfModule...>`
- Příklad: `LoadModule alias_module modules/mod_alias.so`

```
<IfModule alias_module>
   Alias /vhosts /var/www/vhosts
   <Directory /var/www/vhosts>
      DefaultType text/html
      AddDefaultCharset utf-8
      Order allow,deny
      Allow from all
   </Directory>
</IfModule>
```

- Kontrola (testování) konfigurace
  - Je nutné uvést používané definice (-D def)

  apache2ctl -t -D SSL -D PHP5 -D USERDIR -D INFO

  [time] [warn] The Alias directive in .../00_mod_autoindex.conf at line 5 will probably never match because it overlaps an earlier Alias.

- Zjištění modulů načítaných do serveru.
  - Bere v úvahu pouze konfigurační soubory, nikoliv běžící server

  apache2ctl -M

  Loaded Modules:
  core_module (static)
  mpm_worker_module (static)
  ... alias_module (shared) auth_basic_module (shared) ...

- List staticky zkompilovaných modulů

  apache2ctl -l

  Compiled in modules: core.c worker.c http_core.c mod_so.c

Moduly pro souběžné zpracování
- Kontrola vytváření potomků (procesů a vlákan) k obsluze požadavků na určených portech
- Volba MPM při konfiguraci kompilace (--with-mpm=jméno)
- Použití MPM je výlučné a obvykle podle použitého OS

UNIX	BeOS
prefork – procesy	beos
worker – vlákna	
event – experimentální (vlákna)	Netware
itk – experimentální (procesy)	mpm_netware

Windows	OS/2
mpm_winnt	mpmt_os2

MPM – prefork
- Obsluhu požadavků zajišťují procesy

MPM – worker
- Obsluhu požadavků zajišťují vlákna procesů

Moduly pro informace o serveru a logování
- mod_status
  - Umožňuje kontrolovat výkon serveru
  - Poskyuje informace o jeho běhu
- mod_info
  - Poskytuje informace o nastavení serveru
- mod_log_config
  - Poskytuje prostředky pro logování požadavků
  - Umožňuje nastavení různých formátů logů a jejich úložišť
- mod_logio
  - Přidává možnost logovat množství příchozích a odchozích dat

Status module
- počet worker procesů/vláken
  - Obsluhujících
  - Idle
- stav a statistika worker každého procesu/vlákna
  - počet požadavků
  - počet bytů
- celkový počet požadavků a přenesených dat a průměrné hodnoty
- doba běhu serveru, vytížení CPU
- aktuálně obsluhované požadavky

```
<Location /server-status>
   SetHandler server-status
   Order Deny,Allow
   Deny from all
   Allow from .admin.company.com
</Location>
```

- Direktivy
  - ExtendedStatus Off|On - umožňuje logovat jednotlivé požadavky
  - SeeRequestTail Off|On - ovládá záznam prvních/posledních 63 znaků z požadavku

- URL
  - /server-status
  - /server-status?refresh=N automatická obnova po N sekundách
  - /server-status?auto strojově zpracovatelný výstup
- Příklad

`http://ewait.felk.cvut.cz/y36aws/examples/03-status/total/total.cgi`

Info module
- Výpis obsahuje
  - seznam použitých modulů
  - výpis možných direktiv modulu
  - výpis použitých direktiv modulu
- Poskytované informace mohou být citlivé, je vhodné k nim omezit přístup.

```
<Location /server-info>
   SetHandler server-info
   Order allow,deny
   # Allow access from server itself
   Allow from 127.0.0.1
</Location>
```

- URL
  - /server-info - komplexní informace

- /server-info?název_modulu - informace o zvoleném modulu (např. mod_alias.c)
- /server-info?config - konfigurace bez komentářů a bílých znaků
- /server-info?list - seznam použitých modulů
- /server-info?server - základní informace o serveru

Moduly pro logování
- Základní logování zajišťuje jádro serveru (core)
- Direktiva **ErrorLog** file-path|syslog[:facility]
  - file-path – logování do souboru
  - syslog[:facility] - využít logovacího démona (pouze UNIX)
    - syslogd
    - syslog-ng
- Formát záznamu nelze změnit
- Obsahuje i ladící výstup z CGI skriptů (stderr)

Logování – formátování
- Formátovací řetězce
- %h remote host
- %r první řádek z požadavku
- %u remote user - z autentizace (! stavový kód HTTP 401)
- %l remote logname (vlastník procesu, který inicioval spojení)
  vyžaduje modul mod_ident a na vzdáleném počítači běžící démon (authd, pidentd atp.), jinak bude místo hodnoty –
- %b velikost odpovědi bez http hlavičky v bytech, místo 0 je –
- %B velikost odpovědi bez http hlavičky v bytech
- %t čas požadavku (std. formát, %{format}t podle strftime(3))
- %s HTTP status požadavku
- %I/O počet přijatých/odeslaných bytů vč. HTTP hlaviček (vyžaduje modul mod_logio)

access_log
- Modifikátor < / > určuje první/poslední požadavek
  - V případě interních přesměrování
  - Většina direktiv ve výchozím nastavení uvažuje původní požadavek
- Záznam všech požadavků na server
- Formát logu lze konfigurovat stejně jako jeho úložiště

LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common

```
...
147.32.80.98 - - [15/Jul/2008:14:45:49 +0200] "GET /y36aws-0 3/total.png HTTP/1.1"
200 10099
127.0.0.1 - - [15/Jul/2008:14:46:01 +0200] "GET /server-stat us?auto HTTP/1.0" 200
1201
...
```

MIME module
- Svázání meta-informace se souborem podle přípony
  - mime-type (HTTP> Content-Type, př. Content-Type: image/gif)
  - jazyk (HTTP> Content-Language, př. Content-Language: cs, en)
  - znaková sada (HTTP> Content-Type, př. Content-Type: text/html; charset=ISO-8859-2)
  - kódování (HTTP> Content-Encoding, př. Content-Encoding: x-gzip)
- Změna mime-type neovlivňuje informaci Last-Modified
  - hlavička odpovědi zůstane stejná (a změny se neprojeví)
  - při změně mime-type změnit čas poslední modifikace (touch)
- Přípona s tečkou i bez tečky
- Nezáleží na velikosti písmen (case-insensitive)
  - Pokud má soubor více přípon (multiple extensions)
  - Mime-type podle poslední přípony
  - Další metainformace podle ostatních přípon (jazyk, kódování, ...)
  - Porovnání přípon se provádí s každou zvlášť (může vést k neočekávaným výsledkům při použití handleru!)
  - Omezení určení vlastností pouze na poslední příponu:

```
<FilesMatch \.cgi$>
    SetHandler cgi-script
</FilesMatch>
```

- Příklad: Soubor s názvem info.cs.utf8.html
  Content-Type: **text/html**; **charset=UTF-8**
  Content-Language: **cs**
- Umístění souboru mime-type konfigurace
- **TypesConfig** /etc/mime.types
  - Výchozí soubor s mime-typy needitovat (možný přepis při upgrade)
  - Přípony jsou case-insensitive
- **egrep '(zip|html|pdf|jpg|png|mp3)' /etc/mime.types**

```
...
application/pdf          pdf
application/xhtml+xml    xhtml xht
application/zip          zip
audio/mpeg               mpga mpega mp2 mp3 m4a
image/jpeg               jpeg jpg jpe
image/png                png
text/html                html htm shtml
...
```

MIME module – mime-type

- **AddType** *MIME-type extension...*
  - Přidává vazbu mezi příponou (extension) a mime-typem
  - AddType image/gif .gif
- **DefaultType** *MIME-type*|None
  - Určuje implicitní mime-type (core)
  - DefaultType text/plain
- **ForceType** *MIME-type*|None
  - Vynucuje nastavení MIME-type (core)
  - ForceType None
- **RemoveType** *extension...*
  - Odstraňuje vazby na příponou (extension)

MIME module – mime-*

- Jazyk
  - **AddLanguage** *MIME-lang extension...*
    - AddLanguage cs .cz .cs
  - **DefaultLanguage** *MIME-lang*
    - DefaultLanguage en
  - **RemoveLanguage** *extension...*

- Znaková sada
  - **AddCharset** *charset extension...*
    - AddCharset ISO-8859-2 .iso8859-2 .latin2 .cen
  - **AddDefaultCharset** *charset*
    - **Aplikuje se na text/plain, text/html** (core)
    - Má přednost přad ostatními (META tag), používat s rozmyslem!
  - **RemoveCharset** *extension...*

- Kódování
  - **AddEncoding** *MIME-enc extension...*
    - AddEncoding x-gzip .gz
  - **RemoveEncoding** *extension...*

- **mod_mime_magic**
  - Určuje mime-typ na základně prvních několika bajtů obsahu souboru (jako příkaz file)
  - Použití pokud není možné určit mime-typ pomocí mod_mime
  - Direktiva MimeMagicFile povoluje použití modulu
  - Použití modulu znamená další zátěž serveru
  - **MimeMagicFile** *file*
    - MimeMagicFile conf/magic

MIME module – Handlery a filtry

- Handler
  - **AddHandler** *handler extension...*
    - AddHandler cgi-script .cgi
  - **SetHandler** *handler* (core)

    - **RemoveHandler** *extension...*
- Vstupní filtr
    - **AddInputFilter** *filter[;filter…] extension...*
        - Zpracovává příchozí požadavek včetně vstupu metody POST
        - Přidává/přepisuje nastavení pomocí direktivy SetInputFilter
    - **SetInputFilter** *filter[;filter…]* (core)
    - **RemoveInputFilter** *extension...*

- Výstupní filtr
    - **AddOutputFilter** *filter[;filter…] extension...*
        - Zpracovává odchozí odpověď
        - Přidává/přepisuje nastavení pomocí direktivy SetOutputFilter.
        - AddOutputFilter INCLUDES;DEFLATE shtml
    - **SetOutputFilter** *filter[;filter…]* (core)

```
<Directory /www/data/>
   SetOutputFilter INCLUDES
</Directory>
```

    - **RemoveOutputFilter** *extension...*

Vyjednávání (výběr) obsahu
- Umožňuje reagovat na preference uživatele
- Možnosti protokolu HTTP/1.1
    - Hlavičky Accept*
        - Accept-Language: fr; q=1.0, en; q=0.5
        - Accept: text/html; q=1.0, text/*; q=0.8, image/gif; q=0.6, image/jpeg; q=0.6, image/*; q=0.5, */*; q=0.1
- Modul mod_negotiation

Negotiation module – Způsob práce
- Vyjednávání obsahu
    - Multiviews
    - Type-map
- Algoritmus pro vyjednání obsahu
    1. vynásobení kvality zdroje a Accept
    2. nejlepší jazyk
    3. pořadí jazyka v Accept-Language nebo LanguagePriority
    4. nejlepší media type
    5. nejlepší znaková sada (příp. Latin1, non Latin1)
    6. nejlepší kódování
    7. nejmenší velikost obsahu
    8. první zbývající možnost (podle type-map, nebo ASCII pořadí)

Negotiation module – Multiviews
- Povolení nastavením volby Multiviews
    - Options +Multiviews
- Nastavení činnosti direktivou MultiviewsMatch (mod_mime)
    - MultiviewsMatch Any | NegotiatedOnly | Filters | Handlers
        - NegotiatedOnly – musí odpovídat Content-Type/Language/Encoding (default)
        - Filters, Handlers – zpracovává přípony asociované s filtry / handlery
        - Any – zpracovává soubory s libovolnou příponou
- Nastavení preferencí jazyka
    - **LanguagePriority** *MIME-lang [MIME-lang] ...*
        - LanguagePriority cs en de fr
    - **ForceLanguagePriority** None | Prefer | Fallback
        - LanguagePriority cs en de fr

Negotiation module – Type-map
- Soubor obsahující varianty jména souborů (obvykle .var)
- Oddělovač položek – prázdný řádek
    Kvalita zdroje – parametr qs=<0.0,1.0>
    Položka – Content-Type/Language/Encoding
- Zpracování handlerem type-map
    - AddHandler **type-map** .var

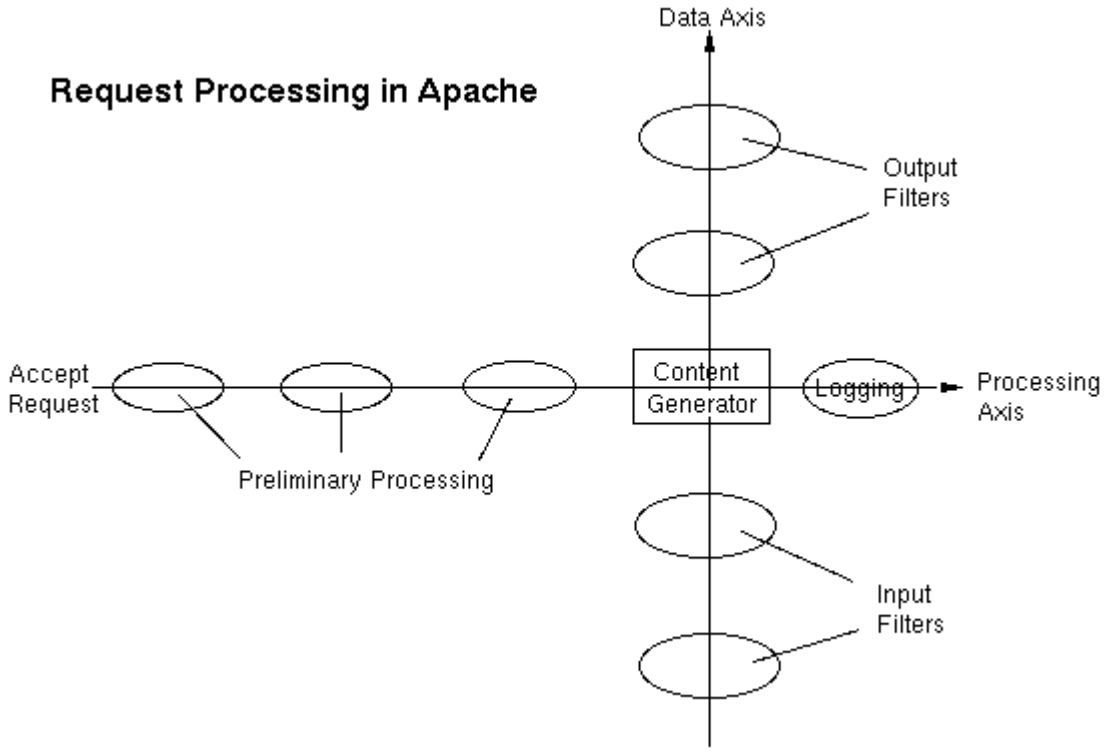
```
URI: text.html

URI: text.html.cs
```

Content-Language: cs-cz
Content-Type: text/html

URI: text.html.en
Content-Language: en-us
Content-Type: text/html

Dynamické generování obsahu



mod_ext_filter

Zajišťuje filtrování požadavku - odpovědi externím programem.

```
ExtFilterDefine lower-to-upper mode=output \
intype=text/plain outtype=text/plain
cmd="/usr/bin/tr '[:lower:]' '[:upper:]'"

<Directory "/tmp/aws">
SetOutputFilter lower-to-upper
AddType text/plain .txt
</Directory>
```

CGI (Common Gateway Interface)

- umožňuje vykonávat uživatelské programy
- moduly cgi, cgid (pro vláknové zpracování), adresář se pak nastaví běžným způsobem
- nutno nastavit ScriptAlias - určuje adresáře, kde se mohou vykonávat CGI skripty
- základní konfigurace

```
LoadModule cgid_module libexec/mod_cgid.so
Scriptsock /var/run/cgisock

ScriptAlias /cgi-bin/ "/www/cgi-bin/"
<Directory "/www/cgi-bin/">
    ...
</Directory>
```

- konfigurace pomocí povolení CGI v adresáři

```
<DirectoryMatch "/home/[a-z0-9]+/www/cgi-bin">
    Options +ExecCGI
    AddHandler cgi-script .cgi .sh ...
</DirectoryMatch>
```

- totéž lze pomocí URL
  `ScriptAliasMatch "^~([a-z0-9]+)/cgi-bin/(.*)" "/home/$1/www/cgi-bin/$2"`

Proměnné

Proměnné nesou informaci o prostředí, mohou přes ně být předávana data (QUERY_STRING, metoda GET). Webový server lze nakonfigurovat tak, aby nastavil i další proměnné (mod_setenv). Poku jsou data odesílána metodou POST, server je předá CGI aplikaci přes standardní vstup a nastaví proměnné REQUEST_METHOD a CONTENT_LENGTH, které je nutné pak testovat!

- DOCUMENT_ROOT (/var/www/localhost/htdocs)
- PATH (/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/sbin…)
- PWD (/var/www/y36aws/03-moduly/cgi-bin)
- SCRIPT_FILENAME (/var/www/y36aws/03-moduly/cgi-bin/df.sh)
- REQUEST_URI (/y36aws-03/cgi-bin/df.sh?dir=/tmp)
- SCRIPT_NAME (/y36aws-03/cgi-bin/df.sh)
- REQUEST_METHOD (POST)
  - určuje, jaká metoda byla použita
- CONTENT_LENGTH (328)
  - server není povinen ukončit předání parametrů přes standardní vstup metodou POST znakem EOF - velikost je nutná k ukončení čtení vstupu
- QUERY_STRING (dir=/tmp)
- SERVER_NAME (ewait)
- SERVER_ADDR (147.32.80.97)
- SERVER_SIGNATURE (Apache Server at ewait Port 80)
- REMOTE_ADDR (147.32.80.98)
- HTTP_USER_AGENT (Mozilla/5.0 (X11; U; Linux i686 (x86_64); cs-CZ; rv:1.8.1.14) Gecko/20080404 Firefox/2.0.0.14)
- HTTP_ACCEPT_CHARSET (ISO-8859-2,utf-8;q=0.7,*;q=0.7)

Příklady

Jednoduchý BASH skript vypisující datum

`date.sh - http://server/cgi-bin/date.sh`

```
#!/bin/bash
echo "Content-type: text/html"
echo ""
/usr/bin/date
```

Využití parametrů předaných skrze *query string*. Pokud se používá shell, lze pomocí běžných utilit zjistit proměnné prostředí (/usr/bin/env atp.).

`df.sh - http://server/cgi-bin/df.sh?dir=/tmp`

```
#!/bin/bash
echo "Content-type: text/html"
echo ""
echo "<pre>"
df -h "${QUERY_STRING/dir=/}"
echo "</pre>"
```

Výpis hodnot polí odeslaných formulářem (GET nebo POST) realizovaný v jazyku C.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int main(int argc, char** argv) {
        char* queryString = NULL;
        char* postData = NULL;
        long contentLength = 0;
```

```
        printf("Content-type: text/plain\n\n");

        printf("QUERY_STRING: %s\n", getenv("QUERY_STRING"));

        // zpracovavame pozadavek POST
        if (strncmp(getenv("REQUEST_METHOD"), "POST", 4*sizeof(char)) == 0) {
                contentLength = atoi(getenv("CONTENT_LENGTH"));
                postData = malloc((contentLength + 1) * sizeof(char));

                printf("contentLength: %ld \n", contentLength);

                read(STDIN_FILENO, postData, contentLength);

                postData[contentLength] = (char) 0;

                printf("POST: %s\n", postData);
        }
        free(postData);
        return 0;
}
```

Vstupní data jsou URL encoded (procento + hodnota), při dalším zpracování je nutno provést dekódování.

Ladění CGI

POZOR, možnosti ladění jsou velmi omezené.

- nastavení logovacího souboru, lze v VirtualHost a globální konfiguraci, cesta je buď absolutní nebo relativní vůči ServerRoot

  ```
  ScriptLog cesta/k/log/souboru

  # ScriptLogLength bytes
  # delka protokolovanych dat POST, PUT
  # ScriptLogBuffer bytes
  ```

Do logovacího souboru se vypisuje:

- vždy
  ```
  %% [time] request-line
  %% HTTP-status CGI-script-filename
  ```

- pokud se CGI nepodařilo spustit
  ```
  %%error
  error-message
  ```

- v případě chyby
  ```
  %request
  All HTTP request headers received
  POST or PUT entity (if any)
  %response
  All headers output by the CGI script
  %stdout
  CGI standard output
  %stderr
  CGI standard error
  ```

  Vypisuje se pouze poslední chyba, pokud nedošlo k výstupu na STDOUT nebo STDERR, je příslušná část vynechána. V hlavičkách se kontroluje pouze syntax a nikoliv obsah. Je nutné pro ladění mít i příslušnou chybu, kterou zaslal server klientovi nebo přístup k ErrorLog, obsahuje přesný popis důvodu chyby (typicky HTTP 500 Internal Server Error - premature end of script headers, malformed headers atp.).

Server Side Includes (SSI)

- modul umožňuje vkládání dynamicky generovaného obsahu do HTML souborů
- technologie SSI není vhodná v případě, že většina stránky je generována dynamicky - hodí se pouze pro vkládání malých částí (např. datum atp.)

Direktivy

- Options +IncludesNoExec - povolí SSI
- Options +Includes - povolí SSI vč. vykonávání externích příkazů

Aby v souboru byly hledány a zpracovány SSI příkazy je nutné takový soubor zpracovat výstupním filtrem INCLUDES. Máme tři možnosti:

- zaregistrováním MIME typu a nastavením výstupního filtru jsme schopni omezit použití SSI na určité soubory:
  ```
  <IfModule mime_module>
      AddType text/html .shtml
      AddOutputFilter INCLUDES .shtml
  </IfModule>
  ```

- použít direktivu XBitHack v adresáři - není pak nutné přejmenovat existující soubory před použitím SSI (v případě předchozí možnosti s příponou .shtml). Zpracování SSI pak bude provedeno u souborů s nastaveným právem *execute (x)*.
  ```
  XBitHack On
  ```

- zpracovat všechny soubory bez výjimky výstupním filtrem:
  ```
  SetOutputFilter INCLUDES
  ```

Použití poslední možnosti je třeba důkladně zvážit, není doporučeno provádět výstupní filtr ani na .html soubory, ale pouze na .shtml nebo použít XBitHack. Zamezí se tak zbytečnému zpracování souborů, které neobsahují SSI.

načtení modulu a povolení SSI pro adresář

```
LoadModule include_module modules/mod_include.so

<Directory "...">
    ...
    <IfModule mime_module>
        AddType text/html .shtml
        AddOutputFilter INCLUDES .shtml
    </IfModule>
    ...
</Directory>
```

Syntaxe SSI

- příkazy se píší jako HTML komentáře
  ```
  <!--#element attribute=value attribute=value ... -->
  ```

Elementy

- echo - výpis
- set - nastavení proměnné

```
<!--#include file|virtual="" -->
```

Umožňuje vložit obsah jiného dokumentu/souboru do zpracovávaného, jsou zde 2 možné atributy (přístupy)

- file - soubor, který je ve stejném adresáři jako zpracovávaný soubor nebo podadresáři, nelze použít absolutní cestu nebo se dostat do nadřazené úrovně
- virtual - dokument, ktetý je k dispozici ve virtuální adresářové struktuře webserveru (aby předchozí omezení nebylo tak silné), musí být povoleno použití PATH_INFO - AcceptPathInfo On|Default - požadavku se totiž nezmění REQUEST_URI jen PATH_INFO.

Použití virtual vést k nečekanému výsledku - dojde ke znovu zpracování požadavku, ale změní se jen PATH_INFO a nikoliv REQUEST_URI. Proto pokud budeme mít všechny soubory (např .ssi, .s/html, .sh) v jednom adresáři a bude nastaven *mod_actions*, může dojít k rekurzivnímu volání akce (Action) díky stále stejné REQUEST_URI! Řešením je v takovém případě použít file, který je zde postačující a nevyvolá opakování požadavku.

Základní proměnné

K dispozici jsou tytéž proměnné prostředí jako pro CGI skripty.
Vložení aktuálního data.

```
<!--#echo var="DATE_LOCAL" -->
```

Actions

- modul umožňuje podle MIME typu vykonat při požadavku určitý CGI skript
- vyžaduje, aby zvolený CGI skript byl v adresáři, na který vede ScriptAlias a měl všechny vlastnosti CGI skriptu

Konfigurace

```
LoadModule actions_module modules/mod_actions.so

<Directory "...">
    # textove soubory dostanou specialni hlavicku a paticku
    Action text/plain "/y36aws-03/actions/txt.sh" virtual
</Directory>
```

- slovo virtual určuje, zda soubor musí existovat
    - pokud není uvedeno a soubor neexistuje, dostaneme *404 Not Found*
    - pokud je uvedeno, skript se vykoná a s existencí/neexistencí souboru se musí vyrovnat sám

txt.sh

```
#!/bin/bash
echo "Content-Type: text/html"
echo ""

echo "<h1>Soubor: ${REDIRECT_URL}</h1>"
echo "<hr>"

if [ "$REQUEST_METHOD" == "POST" ]; then
        #QUERY_STRING="$QUERY_STRING&`cat`"
        QUERY_STRING="$QUERY_STRING&`head -c $CONTENT_LENGTH`"
        QUERY_STRING=`echo "$QUERY_STRING" | sed 's/^&//'`
fi

if [ ! -f "${PATH_TRANSLATED}" ]; then
        echo "Soubor neexistuje."
elif [ `echo "${QUERY_STRING}" | tr '&' '\n'| grep  "^full=" | tail -1` == "full=yes"
] ; then
        echo '<pre>'
        cat "${PATH_TRANSLATED}"
        echo "$QUERY_STRING"
        echo '</pre>'
        echo "<hr>"
        echo "<a href='${REDIRECT_URL}'>zobrazit pouze ukazku (10 radek)</a>"
        echo '<form action="#" method="POST"><input type="submit" name="full"
value="no"><input type="hidden" name="a" value="a"></form>'
else
        echo "<pre>"
        head "${PATH_TRANSLATED}"
        echo "$QUERY_STRING"
        echo "</pre>"
        echo "<hr>"
        echo "<a href='${REDIRECT_URL}?full=yes'>zobrazit cely soubor</a>"
        echo '<form action="#" method="POST"><input type="submit" name="full"
value="yes"><input type="hidden" name="a" value="a"></form>'
fi

echo "<hr>"
stat ${PATH_TRANSLATED} | egrep "Modify:"
```

Komprese posílaných dat

- prohlížeč (klient) musí podporovat kódování (Accept Encoding: gzip,deflate), jinak server pošle nekomprimovaná data
- pokud máme mod_mime a mod_filter lze např.

```
AddType text/plain .log
AddOutputFilter DEFLATE .log
```

Příklad - simulace podpory komprese

plain text

```
~ $ wget ewait/y36aws-03/logs/20.log
--2008-07-02 15:34:28--  http://ewait/y36aws-03/logs/20.log
Překládám ewait… 147.32.80.97
Navazuje se spojení s ewait|147.32.80.97|:80… spojeno.
HTTP požadavek odeslán, program čeká na odpověď… 200 OK
Délka: 11942240 (11M) [text/plain]
Ukládám do: „20.log“.

100%[ ... ] 11 942 240  9,09M/s   za 1,3s

2008-07-02 15:34:29 (9,09 MB/s) – „20.log“ uloženo [11942240/11942240]

~ $ file 20.log
20.log: ASCII text
```

komprese

```
~ $ wget --header='Accept-Encoding: gzip,deflate' ewait/y36aws-03/logs/20.log
--2008-07-02 15:34:37--  http://ewait/y36aws-03/logs/20.log
Překládám ewait… 147.32.80.97
Navazuje se spojení s ewait|147.32.80.97|:80… spojeno.
HTTP požadavek odeslán, program čeká na odpověď… 200 OK
Délka: neudáno [text/plain]
Ukládám do: „20.log“.

    [ ... ] 390 514     1,39M/s   za 0,3s

2008-07-02 15:34:38 (1,39 MB/s) – „20.log“ uložen [390514]

~ $ file 20.log
20.log: gzip compressed data, from Unix
```

Programování vlastních modulů

Modularita SW

Modularita SW se s úspěchem používá téměř u všech velkých projektů k oddělení základní a rozšiřující funkčnosti. Modularita umožňuje vznik a vývoj funkčnosti samotnými uživateli přesně podle jejich potřeb, čímž poskytuje velkou flexibilitu používaného SW.
Z pohledu programátora jedině díky modularitě lze také zachovat přijatelnou míru udržovatelnosti programového kódu, neboť k práci s modulem v zásadě není třeba podrobná znalost ostatních částí programu.
Rozdělení na základní funkčnost v hlavním programu httpd a doplňující funkce v modulech bylo provedeno i v případě Apache při jednom z prvních celkových přepisů (zpřehlednění) zdrojového kódu.
Modularita httpd umožňuje používat základní funkce webového serveru a vložení modulů s právě takovými vlastnostmi, které potřebujeme k dosažení cíle, vše zbytečné lze vynechat. Tím snížíme riziko výskytu chyby v serveru a případně též HW nároky (paměť a další prostředky OS).

Moduly httpd

httpd je, stejně jako drtivá většina dalších programů, které mají vykonávat své úkoly s minimální režií a maximální rychlostí na dané platformě, napsán v jazyce C, včetně všech standardně přidávaných (distribučních) modulů.
httpd je však natolik rafinovaný SW, že existují speciální moduly zpřístupňující jeho API i v jiných jazycích, a dokonce skriptovacích, jako je např.
Perl (**mod_perl**), takže se dá říci, že moduly httpd lze napsat v libovolném programovacím jazyce. (*mod_perl je ale především velmi rychlý interpret perlovského kódu, slouží jako účinná náhrada spouštění perlovských CGI skriptů.*)

Poměrně významnou nevýhodou je naopak obtížný vývoj modulů, protože přímý přístup k vnitřním datovým strukturám httpd pochopitelně může vést a v průběhu ladění často vede k ovlivnění správného běhu serveru a případně k jeho předčasnému ukončení (*segfault*).

Obecně se tedy dá říci, že moduly se píší pro tu funkčnost, jež vyžaduje maximální výkon na straně serveru, a kde se tedy relativně větší pracnost vývoje vyplatí.

Základní elementy API httpd modulů

Při psaní modulů máme k dispozici vedle obecně neportabilních prostředků OS především dvě sub-API:

1. Apache High-Level function: ap_*().
2. Apache Portable Runtime (APR) library function: apr_*().

Základní definice modulu se provede deklarací struktury **module_struct**.

SRC/include/http_config.h

```
  struct module_struct
```

SRC/modules/experimental/mod_example.c

```
module AP_MODULE_DECLARE_DATA example_module =
{
    STANDARD20_MODULE_STUFF,
    x_create_dir_config,    /* per-directory config creator */
    x_merge_dir_config,     /* dir config merger */
    x_create_server_config, /* server config creator */
    x_merge_server_config,  /* server config merger */
    x_cmds,                 /* command table */
    x_register_hooks,       /* set up other request processing hooks */
};
```

SRC/modules/experimental/mod_example.c

```
static void x_register_hooks(apr_pool_t *p)
{
    ap_hook_pre_config(x_pre_config, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_post_config(x_post_config, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_open_logs(x_open_logs, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_child_init(x_child_init, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_handler(x_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_quick_handler(x_quick_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_pre_connection(x_pre_connection, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_process_connection(x_process_connection, NULL, NULL, APR_HOOK_MIDDLE);
    /* [1] post read_request handling */
    ap_hook_post_read_request(x_post_read_request, NULL, NULL,
                              APR_HOOK_MIDDLE);
    ap_hook_log_transaction(x_logger, NULL, NULL, APR_HOOK_MIDDLE);
#if 0
    ap_hook_http_scheme(x_http_scheme, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_default_port(x_default_port, NULL, NULL, APR_HOOK_MIDDLE);
#endif
    ap_hook_translate_name(x_translate_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_map_to_storage(x_map_to_storage_handler, NULL,NULL, APR_HOOK_MIDDLE);
    ap_hook_header_parser(x_header_parser_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_check_user_id(x_check_user_id, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_fixups(x_fixer_upper, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_type_checker(x_type_checker, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_access_checker(x_access_checker, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_auth_checker(x_auth_checker, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_insert_filter(x_insert_filter, NULL, NULL, APR_HOOK_MIDDLE);
}
```

Příklad vytvoření elementárního funkčního modulu

Vlastní modul můžeme napsat zcela od začátku ručně, nebo si můžeme nechat vytvořit vzorový příklad příkazem
`$ apxs -n <jmeno_modulu> -g`

K funkčnímu modulu je potřeba provést vedle jeho správného napsání ještě další kroky:

1. Překlad sdílené knihovny (.so, případně .dll nebo podobné) a její umístění do adresáře s moduly httpd.
2. Úprava konfigurace httpd:
    1. načtení modulu (direktiva LoadModule), a
    2. (typicky) i jeho další konfigurace.
3. Restart daemona httpd.

Překlad můžeme provést opět ručně, což vyžaduje znalost parametrů překladače a linkeru a cest k hlavičkovým souborům httpd a knihovny APR. Alternativně lze využít možností skriptu apxs. Přepínač -c zajistí překlad a přípravu sdílené knihovny, -i pak její instalaci.
`$ apxs -c -i <jmeno_modulu>.c`

Konfigurace serveru a modulu
Dalšími parametry skriptu apxs se dá dosáhnout přidání konfiguračních řádek do httpd.conf. Úspěšnost tohoto kroku ale závisí na parametrech a proměnných nastavených při instalaci (např. /usr/lib64/apache2/build/config_vars.mk) a na současném obsahu tohoto konfiguračního souboru a případně vše potřebné přenastavit parametry -S resp. -D skriptu apxs.
Proto tento krok bude rychlejší provést ručně.

1. Nadefinovat -D <muj define> do proměnné APACHE2_OPTS (např. v /etc/conf.d/apache2).
2. Vytvořit konfiguraci modulu:

```
#Obvykle v httpd.conf
<IfDefine <muj define>>
        LoadModule   <jmeno_modulu>_module   modules/mod_<jmeno_modulu>.so
</IfDefine>

# Případné konfigurační volby modulu, jsou-li zde povoleny ("struct command_rec").

# Obvykle v modules.d/NN_mod_<jmeno_modulu>.conf
<IfDefine <muj define>>
        <IfModule <jmeno_modulu>_module>
                <Location <cesta>>
                    SetHandler <jmeno_definovane_v_modulu>
                    Order allow,deny
                    Allow from all
                        # Případné konfigurační volby modulu, jsou-li zde povoleny.
                </Location>
        </IfModule>
</IfDefine>
```

Po restartu serveru bychom na adrese http://localhost/<cesta> měli vidět vše, co modul ve spolupráci s případnými dalšími moduly a filtry vyprodukuje.

Ladění kódu
Ladění kódu libovolného (síťového) daemona je přímo velmi obtížné, protože se z definice odpojuje od řídicího terminálu a navíc typicky běží ve více vláknech/procesech.
Proto httpd, stejně jako řada dalších podobných programů, poskytuje možnost být spuštěn jednovláknově a na popředí.
V našem případě jde o přepínač -X.
 # httpd -X <other options>

- mod_userdir
- Možnosti integrace
- suEXEC
- PHP
- Možne způsoby integrace
- Zakladni nastaveni
- PHP + suEXEC
- suPHP

Uživatelské adresáře

mod_userdir

dává možnost exportovat na web dané adresáře uživatelů serveru. Uživatelské adresáře jsou přístupné pod URL http://server/**~uzivatel**.

- **UserDir**
  - Direktiva má několik použití a znamená:
    - název adresáře, který se má exportovat,
    - absolutní cesta k adresáři, ve kterém jsou exportované uživatelské adresáře (aby nemusely být součástí domovského adresáře),
    - URL (→ redirect),
    - enabled nebo disabled povolení nebo nepovolení pro vybrané uživatele.
  - Syntaxe: UserDir *název* | *absolutní cesta* | *URL* | *enabled [seznam uživatelů]* | *disabled [seznam uživatelů]*
  - Default: UserDir public_html
  - Kontext: server, vhost

Příklady
Předpokládejme že, domovské adresáře uživatelů jsou umístěny v /home a existuje uživatel franta.
Požadavek http://www.mycorp.k328/~franta/index.html → uživatel se jmenuje franta.

UserDir www
<DirectoryMatch /home/*/www>
Order deny,allow
</DirecotryMatch>
//Požadavek povede na soubor /home/franta/www/index.html.

UserDir /var/www/users
<Directory /var/www/users>
Order deny,allow
</Direcotry>
//Požadavek povede na soubor /var/www/users/franta/index.html.
UserDir http://www.mycorp.k328/users/*/
Požadavek bude přesměrován na URL http://www.mycorp.k328/users/franta/index.html.
UserDir enabled
UserDir disabled root sysadm
Uživatelské adresáře jsou povoleny pro všechny kromě uživatelů root a sysadm.
UserDir disabled
UserDir enabled franta pepa
Uživatelské adresáře jsou zakázány všem kromě uživatelů franta a pepa.

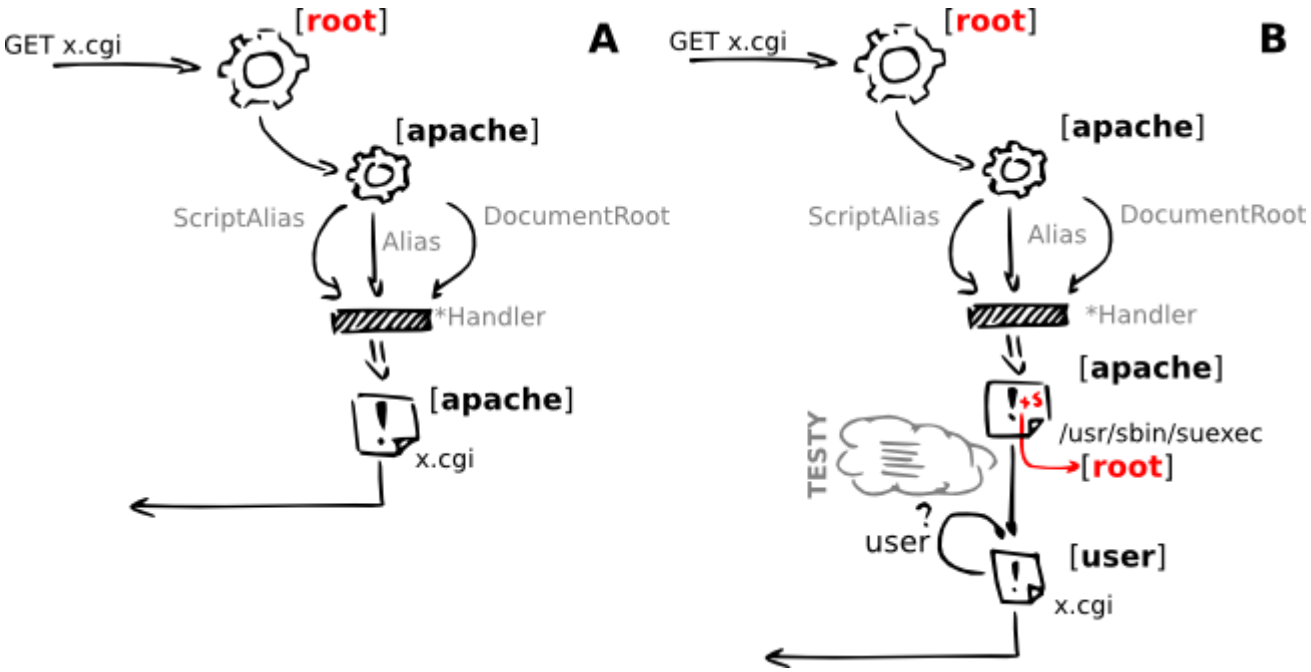
<Directory /home/*/www/cgi-bin>
Options ExecCGI
SetHandler cgi-script
</Directory>
Povolení zpracování CGI pro uživatelské adresáře.

SuEXEX
Jednou z možností, jak postavit webovou aplikaci je použití CGI a SSI (vizte předchozí přednášky). Tento přístup s sebou přináší komplikace při vytváření aplikace i bezpečnostní rizika v následujících případech:

- uživatelům je na web exportován podadresář jejich domovského adresáře (např. www nebo public_html),
- na serveru je provozován virtualhosting a správa jednotlivých hostitelů je delegována na některého z uživatelů (např. zákazníka).

V těchto případech veškeré procesy spouštěné serverem běží pod stejným uživatelem a skupinou jako původní proces webserveru, který požadavek obsluhoval (typicky apache, apache). Toto chování nemusí být výhodné z mnoha důvodů, pro příklad uvádíme alespoň některé:

- soubory vytvářené aplikací nebudou pro uživatele čitelné/zapisovatelné,
- omezit snadno pohyb po filesystému serveru (pokud server využívá více uživatelů, musí data exportovaná na web být alespoň čitelná webserverem, což může vést k neoprávněnému přístupu k nim),
- nelze snadno vynutit dodržení kvót (např. maximální obsazené místo na pevném disku),
- a další.



Řešením je využití mechanismu *suEXEC*, který je součástí *Apache httpd*.
V případě, že přijde požadavek na zdroj, kterým je CGI skript nebo stránka s SSI, je spuštěn přes wrapper tak, aby běžel buď s identitou nastavenou v konfiguraci nebo s identitou majitele spouštěného souboru.
K tomu je potřeba **mod_suexec.so a *wrapperu*** - aplikace suexec - která zařídí změnu identity.

Jelikož wrapper suexec musí mít nastaven setuid bit a musí být vlastněn uživatelem root, je zde riziko zneužití. Z toho důvodu je potřeba vhodně omezit přístup k tomuto *wrapperu*. Navíc je většina konfigurace určena již při kompilaci a je implementován algoritmus, který zabraňuje pokusům o zneužití. To samozřejmě znamená omezení pro existující CGI aplikace a ty, pokud pro použití s *suEXEC* nejsou připraveny, nemusí fungovat.

Kompilace suEXEC
Nastavení *suEXEC* se z velké části provádí už při kompilaci. Možnosti skriptu configure následují i s krátkým popisem.

- –enable-suexec - povolí kompilaci a instalaci *suEXEC*, jelikož *suEXEC* není součástí výchozího nastavení balíku.
- –with-suexec-bin=PATH - cesta k *wrapperu*, z bezpečnostních důvodů je určena při kompilaci.
- –with-suexec-caller=UID - kdo smí *wrapper* spouštět, tj. identita, pod kterou běží webserver (jinému UID nebude povoleno spuštění).
- –with-suexec-userdir=DIR - název adresáře z direktivy UserDir, pokud chceme povolit *suEXEC* pro uživatelské adresáře. Pokud máme více virtuálních hostitelů a exportujeme různé uživatelské adresáře, je nutné zde uvést cestu k adresáři, který je jim nadřazen.
- –with-suexec-docroot=DIR - název adresáře, ve kterém je, kromě UserDirs povolen *suEXEC*, pokud není uveden, použije se takový, který byl zadán přepínačem –datadir=DATA.
- –with-suexec-uidmin=UID, –with-suexec-gidmin=GID - minimální UID, GID, pod kterým může program běžet - cílem je zabránit běhu jako systémový uživatel nebo skupina.
- –with-suexec-logfile=FILE - cesta k log souboru.
- –with-suexec-safepath=PATH - určuje bezpečnou hodnoutu proměnné prostředí PATH, výchozí je /usr/local/bin:/usr/bin:/bin

Po instalaci je vhodné ověřit, jaká práva jsou nastavena pro *wrapper* suexec. Pokud budeme předpokládat, že webserver má efektivní název skupiny apache, pak by práva měla vypadat následovně.

y36aws ~ # ll /path/to/suexec
-rws--x--- 1 root apache 14K 2. lis 10.44 /path/to/suexec

Pokud by byla nastavena jinak, lze to napravit následovně.
y36aws ~ # chown root:apache /cesta/k/suexec
y36aws ~ # chmod 4750 /cesta/k/suexec

Parametry kompilace suEXEC v Gentoo GNU/Linux

Pokud *Apache httpd* nekompilujeme ručně ale používáme balíčkovací systém, nastavení kompilačních parametrů se liší. Pro balíčkovací systém Portage v distribuci Gentoo GNU/Linux se nastavení kompilačních parametrů suEXEC se provádí přes proměnné prostředí před instalací balíku.

- SUEXEC_SAFEPATH: Default PATH for suexec (default: /usr/local/bin:/usr/bin:/bin)
- SUEXEC_LOGFILE: Path to the suexec logfile (default: /var/log/apache2/suexec_log)
- SUEXEC_CALLER: Name of the user Apache is running as (default: apache)
- SUEXEC_DOCROOT: Directory in which suexec will run scripts (default: /var/www)
- SUEXEC_MINUID: Minimum UID, which is allowed to run scripts via suexec (default: 1000)
- SUEXEC_MINGID: Minimum GID, which is allowed to run scripts via suexec (default: 100)
- SUEXEC_USERDIR: User subdirectories (like /home/user/html) (default: public_html)
- SUEXEC_UMASK: Umask for the suexec process (default: 077)

Použití suEXEC

suEXEC lze použít dvěma způsoby:

1. nastavit pro každého virtualního hostitele a hlavní server uživatelské jméno a skupinu,
2. povolit použití pro uživatelské adresáře.

V prvním případě se (typicky uvnitř kontejneru <VirtualHost>) použije **jediná** direktiva modulu mod_suexec.so

- ***SuexecUserGroup***
  - Direktiva nastavuje uživatele a skupinu - identitu, pod kterou poběží CGI programy.
  - SuexecUserGroup User Group
  - Kontext: server, virtual host.
  - K dispozici od Apache httpd 2.0 a pozdějších.

Druhý případ vyžaduje pouze konfiguraci webových domovských adresářů uživatelů, použití *suEXEC* už bude dále zajištěno automaticky.

Zda je *suEXEC* v serveru podporován (tj. je načten modul a je k dispozici *wrapper*) lze zjistit ze souboru error_log následovně.
y36aws ~ # grep -i 'suexec' /cesta/k/error_log
[Sun Nov 02 10:50:00 2008] [notice] suEXEC mechanism enabled (wrapper: /cesta/k/suexec)

suEXEC loguje do souboru, který byl určen při kompilaci, typicky se jedná o soubor suexec_log, který je umístěn v adresáři, kam loguje server.

Bezpečnostní algoritmus

suEXEC je postaven na *wrapper*u, který je volán při každém požadavku na CGI program nebo SSI a musí mít nastavený *setuid* bit. *Wrapper* proto implementuje algoritmus, který se snaží zabránit jeho případnému zneužití.

Aby byl CGI program spuštěn pomocí suEXEC musí všechny podmínky (kroky algoritmu) být splňené. Libovolná jedna chyba znamená celkový neúspěch a vede na chybu HTTP 500 (Internal Server Error).
Algoritmus je pokud možno co nejpřesněji přeložen z oficiální dokumentace (http://httpd.apache.org/docs/2.2/suexec.html#model).

1. Existuje uživatel, který se pokouší spustit *wrapper*, na serveru?
2. Byl *wrapper* zavolán se správným počtem argumentů?
   - Může být např. poškozen *suEXEC* sám o sobě nebo binárka *Apache httpd*.
3. Má uživatel, který se pokouší *wrapper* spustit k tomu právo?
   - Kontroluje se, zda UID odpovídá tomu, které bylo nastaveno při kompilaci (–with-suexec-caller=UID).
4. Obsahuje cílový CGI/SSI program nebezpečné hierarchické reference?

    - Reference / a .. nejsou dovoleny, program musí být umístěn v adresáři nastaveném při kompilaci (–with-suexec-docroot=DIR) nebo v uživatelských adresářích.
5. Existuje cílové uživatelské jméno na serveru?
    - Např. pomocí direktivy SuexecUserGroup, lze nastavit neplatnou identitu.
6. Existuje cílový název skupiny?
    - Např. pomocí direktivy SuexecUserGroup, lze nastavit neplatnou identitu.
7. Není cílový uživatel superuživatel?
    - Uživatel root nesmí vykonávat CGI/SSI programy.
8. Je UID alespoň takové jako minimální zadané při kompilaci?
    - Tímto způsobem lze zabránit vykonání programu pod identitou systémových uživatelů.
9. Není cílová skupina superuživatelská?
    - Skupina root nesmí vykonávat CGI/SSI programy.
10. Je GID alespoň takové jako minimální zadané při kompilaci?
    - Tímto způsobem lze zabránit vykonání programu pod identitou systémových skupin.
11. Je *wrapper* schopen změnit identitu na cílové UID a GID.
    - Zde se *wrapper* stane cílovým uživatelem a skupinou (všech skupin, kterých je uživatel členem).
12. Je možné přejít do adresáře ve kterém je požadovaný CGI/SSI program?
    - Nutnost.
13. Je adresář obsahující požadovaný CGI/SSI program součástí webspace webserveru?
    - Kontroluje se, zda je adresář obsažen ve webspace a buď je v při kompilaci nastaveném suEXEC document root nebo je v UserDir se jménem nastaveným při kompilaci.
14. Je adresář obsahující požadovaný CGI/SSI program zapisovatelný pouze cílovým uživatelem?
    - Nechceme povolit měnit obsah adresáře nikomu dalšímu.
15. Existuje požadovaný CGI/SSI program?
    - Jinak není co vykonávat.
16. Je CGI/SSI program zapisovatelný pouze cílovým uživatelem?
    - Nechceme povolit měnit program nikomu dalšímu.
17. Nemá požadovaný CGI/SSI program nastaven *setuid* nebo *setgid* bit?
    - Nespouštíme programy, které mohou opět změnit svoji identitu.
18. Vlastní požadovaný CGI/SSI program cílový uživatel a skupina?
    - Nutnost (vše se loguje, tak ať máme jistotu v tom, o co se kdo pokouší).
19. Je *wrapper* schopen úspěšně vyčistit prostředí procesu a zajistit tak bezpečné fungování?
    - Nastavení bezpečného obsahu proměnné PATH a odstranění ostatních proměnných prostředí kromě těch, které nejsou v seznamu bezpečných proměnných (nastaveno při kompilaci).
20. Je *wrapper* schopen stát se požadovaným CGI/SSI programem a spustit se?
21. Pokud vše dopadne dobře, zde končí činnost *wrapper*u a začíná se vykonávat požadovaný CGI/SSI program.

Příklady

Instalace *Apache httpd* s podporou *suEXEC* (postup je určen pro Gentoo GNU/Linux).

```
y36aws ~ # echo "www-servers/apache suexec" >> /etc/portage/package.use
y36aws ~ # export SUEXEC_USERDIR='www'
y36aws ~ # emerge -avnN apache
```

Načtení modulu mod_suexec.so.

```
<IfDefine SUEXEC>
    LoadModule suexec_module modules/mod_suexec.so
</IfDefine>
```

Nastavení uživatelských adresářů pro zpracování CGI programů pomocí *suEXEC*. Podle předchozí konfigurace se předpokládá, že uživatelské webové adresáře odpovídají shellovému vzoru /home/*/www.

```
# povoleni uzivatelskych adresaru
<IfDefine USERDIR>
    LoadModule userdir_module modules/mod_userdir.so
    <IfModule userdir_module>

    # nastaveni nazvu uzivatelskych adresaru
    UserDir www

    # nastaveni pristupu k adresarum
    <Directory /home/*/www>
        AllowOverride FileInfo AuthConfig Limit Indexes
        Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
        <Limit GET POST OPTIONS>
```

```
                Order allow,deny
                Allow from all
            </Limit>
            <LimitExcept GET POST OPTIONS>
                Order deny,allow
                Deny from all
            </LimitExcept>
        </Directory>

        # beh CGI/SSI programu pouze pres suEXEC
        <IfDefine SUEXEC>
            <IfModule suexec_module>
                <Directory /home/*/www/cgi-bin>
                    Options ExecCGI
                    SetHandler cgi-script
                </Directory>
            </IfModule>
        </IfDefine>
        </IfModule>
</IfDefine>
```

Nastavení virtuálních hostitelů pro zpracování CGI programů pomocí *suEXEC*.

```
<VirtualHost ...>
  ...

        # nastaveni suEXEC + CGI
        <IfDefine SUEXEC>
            <IfModule suexec_module>
                # urceni uzivatele a skupiny
                SuexecUserGroup user group

                    # povoleni CGI
                    <Directory /var/www/path/to/cgi-bin>
                        Options +ExecCGI
                        AddHandler cgi-script .cgi
                    </Directory>
            </IfModule>
        </IfDefine>
 </VirtualHost>
```

Jednoduchý CGI program pro zjištění identity, pod kterou proces běží.

```
#!/bin/bash
cat << --HTTP--
Content-type: text/plain
Connection: close

`id`
--HTTP--
whoami.cgi
```

Příklad obsahu logu *suEXEC* - chyba - identita vlastník souboru byla odlišná od požadované identity.

```
[2008-11-02 17:33:54]: uid: (1004/pepa) gid: (100/users) cmd: whoami.cgi
[2008-11-02 17:33:54]: target uid/gid (1004/100) mismatch with directory (0/0) or program (0/0)
```

Příklad obsahu logu *suEXEC* - úspěch - obsahuje pouze identitu majitele spouštěného CGI programu.

```
[2008-11-02 18:58:00]: uid: (1001/mycorp) gid: (100/users) cmd: whoami.cgi
```

FastCGI

Rozhraní CGI má pro provoz webových aplikací několik zásadních výhod proti implementaci aplikace přímo pomocí API webového serveru (v případě Apache httpd modulu). Jsou to:

- jednoduchost,
- nezávislost na jazyku,

- nezávislost na architektuře,
- izolace procesů,
- standard atd.

Klíčovým problémem aplikací využívajících CGI rozhraní je **výkon**. Režie nutná pro vytvoření nového procesu, ve kterém aplikace běží, je vysoká.

Výhodou implementace aplikace přímo pomocí API webového serveru je hlavně **výkon** a obsluha požadavku rovnou ve webserveru. Řešení má ale též nevýhody:

- komplexnost,
- závislost na jazyku,
- svázanost s architekturou serveru,
- komplikované ne-li nemožné dosáhnout izolace procesů.

Řešením se snaží být **FastCGI** (www.fastcgi.com) - specifikace firmy OpenMarket Inc. Přichází s konceptem perzistence procesů webové aplikace. Mezi hlavní výhody patří:

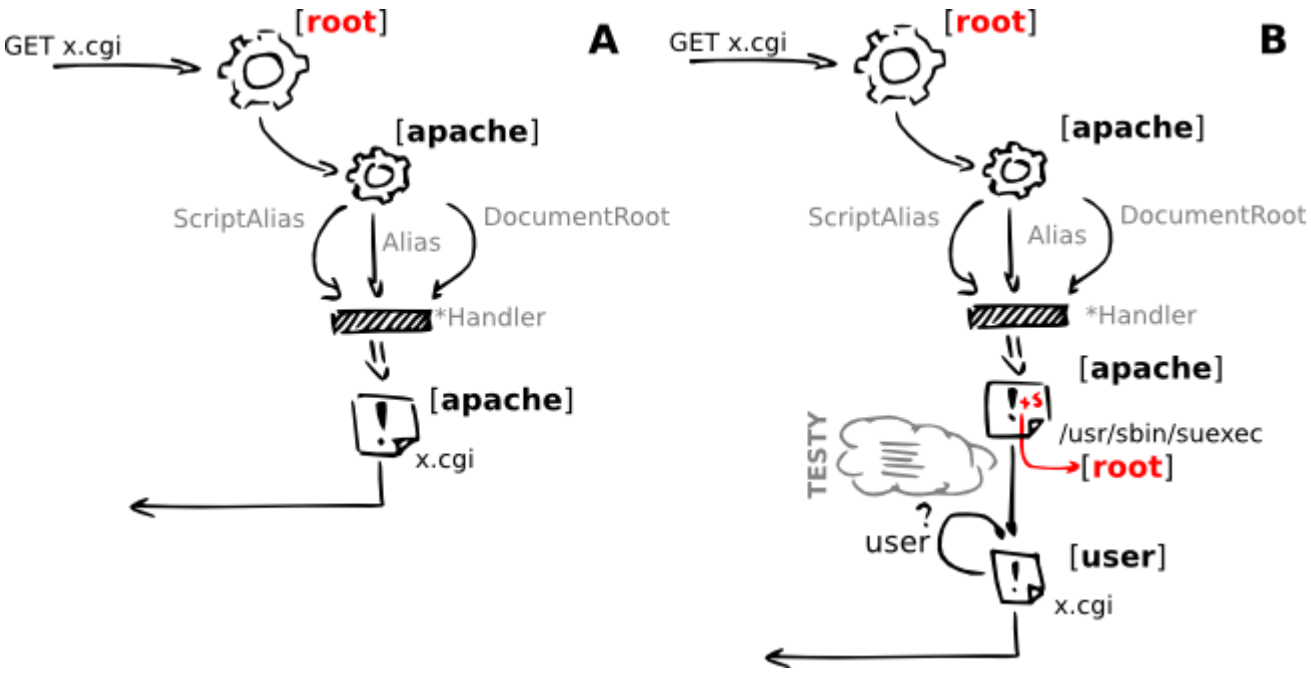
- perzistentní procesy - vyšší výkon,
- izolace procesů,
- podpora distribuovaného výpočtu - je možné se k aplikaci připojit i pomocí TCP spojení,
- „jednoduchost“, nezávislost na jazyku a architektuře, relativně snadná migrace z CGI.

Způsob obsluhy požadavků

CGI

Požadavky obsluhuje webserver následujícím způsobem:

1. pro každý požadavek vytvoří nový proces a provede jeho inicializaci,
2. do proměnných prostředí uloží informace z požadavku,
3. vstup od klienta (POST) pošle na standardní vstup procesu,
4. CGI program zapisuje výstup na svůj standardní výstup (odesílá se klientovi) a chyby na standardní chybový výstup (error log a script log),
5. CGI program končí s ukončením obsluhy požadavku.



FastCGI

Životní cyklus procesu nekončí s ukončením obsluhy požadavku, ale proces je recyklován a znovupoužit pro obsluhu dalších požadavků. FastCGI aplikace mohou být jedno- i více-vláknové!

1. Webserver (FastCGI process manager) vytvoří procesy aplikace, aby mohl obsluhovat požadavky.
    - při startu serveru - statické aplikace,
    - při příchodu požadavku - dynamické aplikace.
2. FastCGI aplikace provede inicializaci a čeká na požadavky - **spojení** - od webserveru.

3. Při příchodu požadavku na webserver je vytvořeno spojení s FastCGI procesem a jsou mu odeslány potřebné informace (ty z proměnných prostředí).
4. FastCGI proces přes stejné spojení odešle odpověď (standardní výstup a standardní chybový výstup).
5. Uzavřením spojení mezi webserverem a FastCGI aplikací je obsluha kompletní. FastCGI proces ale nekončí, čeká na další požadavky.

Základní rozdíly:

- perzistence procesů (obsluhují více požadavků),
- full-duplex spojení, do kterého se multiplexují proměnné prostředí, standardní vstup, standardní výstup a standardní chybový výstup,
    1. AF_UNIX - unixový socket - lokálně spuštěné aplikace,
    2. AF_INET - TCP spojení - vzdáleně spouštěné aplikace,
- lze spouštět i více-vláknové aplikace - obsluhu provádí jeden proces, ve kterém běží vlákna (např. javovská aplikace)

Vzdálené spouštění FastCGI aplikací umožňuje exportovat některé intranetové aplikace po vhodném nastavení zabezpečení do internetu, rozložení zátěže a další. Nutností je dobré zabezpečení → firewall!

Protokol FastCGI

Komunikace mezi webserverem a aplikací probíhá pomocí speciálního protokolu. Existuje jeden formát paketu pro komunikaci oběma směry, rozlišují se typy:

- FCGI_PARAMS - posílání proměnných prostředí ve tvaru promenna=hodnota,
- FCGI_STDIN - standardní vstup z webserveru odesílaný aplikaci,
- FCGI_DATA - data pro aplikaci pokud je v roli *filter*,
- FCGI_STDOUT - standardní výstup z FastCGI aplikace webserveru,
- FCGI_STDERR - standardní chybový výstup z FastCGI aplikace webserveru,
- FCGI_END_REQUEST - konec obsluhy požadavku (mohou poslat aplikace i server).

Role FastCGI aplikací

FastCGI aplikaci lze provozovat v několika rolích:
Responder -odpovídá na požadavky (stejný režim jako klasické CGI).
Filter - zpracovává požadovaný soubor předtím, než je odeslán klientovi, neobchází se kontrola oprávnění webserverem.
Authorizer - provádí autorizace před provedením požadavku, návratový kód HTTP musí být 200 OK, jinak autorizace neproběhla s kladným výsledkem. Výhody:

- kontrola oprávnění v externích databázích,
- komplexní ACL a další.

(Konfigurace rolí Filter a Authorizer pro mod_fastcgi nejsou v záběru přednášky a je ponecháno na čtenáři zda-li se s nimi seznámí blíže).

Vývoj FastCGI aplikací

Pro usnadnění vývoje FastCGI aplikací se pro různé jazyky používají knihovny, které zakrývají většinu implementačních detailů protokolu FastCGI. Pro vývoj v jazyce C lze použít *FastCGI Developer's Kit* (viz www.fastcgi.com). Program potom vypadá velmi podobně jako pro klasické CGI, přibyla pouze smyčka pro přijetí obsluhy požadavku. Kit předefinovává standardní rutiny pro I/O.

```
/* FastCGI Developer's Kit */
#include <fcgi_stdio.h>

int main(void) {
  /* prijimaci smycka */
  while(FCGI_Accept() >= 0) {
    printf("Content-type: text/html\r\n");
    printf("\r\n");
    printf("Hello world!<br>\r\n");
  }
  /* ukonceni */
  return (0);
}
```

HelloWorld.c

Překlad FastCGI aplikace s použitím kitu.

```
y36aws /var/www/main/fcgi # gcc HelloWorld.c -o HelloWorld.fcgi -lfcgi
```

mod_fastcgi

Modul *mod_fastcgi* poskytuje FastCGI rozhraní pro server *Apache httpd*, rozhraní FastCGI není implementováno 100% podle specifikace! Součástí modulu je handler a manažer procesů FastCGI aplikací.

Modul poskytuje handler fastcgi-script, registrace se provádí direktivami

- SetHandler fastcgi-script,
- AddHandler fastcgi-script .fcg .fcgi.

FastCGI aplikace může být *statická*, *dynamická*, nebo *externí*. Podle toho se na ní vztahují konfigurační volby. Pro spuštění *dynamické* FastCGI aplikace se vyžaduje alespoň Options ExecCGI.
Modul podporuje pouze role *Responder* a *Authorizer*. Pro roli *Authorizer* existují 3 variace - Apache httpd rozlišuje autentizaci, autorizaci a kontrola přístupu.

Process manager

Process manager program, součást FastCGI, který zajišťuje kontrolu vykonání FastCGI aplikací, tj. jejich

- spouštění (na základě požadavku - dynamicky, podle konfigurace - staticky),
- restartování v případě ukončení aplikace (důvod není podstatný),
- ukončování.

Statické aplikace jsou nastartovány po startu serveru, případně po prodlevě nastavené v konfiguraci.

Dynamické aplikace

Process manager získá po přijetí požadavku informace od webserveru, že má spustit aplikaci. Od jejího spuštění se k ní chová jako ke *statické* FastCGI aplikaci. Aby byl běh *dynamických* FastCGI aplikací efektivní, musí *process manager* implementovat sadu zásad, které mu umožní rozhodovat, zda vytvářet další procesy aplikace, či aplikaci ukončit.
Základním sledovaným ukazatelem je aktivita (počet obsloužených požadavků) v posledním časovém období (-updateInterval).
Aby nedocházelo ke zbytečnému ukončování procesů, které byly velmi aktivní nedávno v minulosti, ale ne v posledním sledovaném časovém období, je potřeba sledovat aktivitu i v předchozích obdobích. Výsledná popularita aplikace je pak určena pomocí speciální funkce (viz http://www.fastcgi.com/drupal/node/6?q=node/24).
Výsledek je porovnán s konfiguračními parametry -multiTreshold, -singleTreshold a pokud je nižší po uplynutí -killInterval jsou nadbytečné procesy ukončeny.

Konfigurační direktivy
(Výběr. Popis všech direktiv je k dispozici v dokumentaci modulu - je součástí instalačního balíku.)

FastCgiServer

- Konfigurace statické FastCGI aplikace
- FastCgiServer filename [option …]
- Kontext: server
- option (výběr):
  - -appConnTimeout n - výchozí: 0 (blokující) - blokující/neblokující connect()
  - -user username|uid - výchozí: - uživatel pro suEXEC
  - -group groupname|gid - výchozí: - skupina pro suEXEC
  - -idle-timeout n - výchozí: 30 seconds - timeout, po kterém bude obsluha požadavku přerušena (chyba)
  - -init-start-delay n - výchozí: 1 second - prodleva mezi startem serveru a startem FastCGI aplikací
  - -flush - vypne výstupní cache
  - -min-server-life n - výchozí: 30 seconds - minimální doba po kterou musí proces běžet, aby byl restartován,
  - -nph - neparsovat hlavičky
  - -priority - výchozí: 0 - priorita procesu aplikace
  - -processes - výchozí: 1 - počet procesů aplikace spuštěných při startu serveru
  - -restart-delay n - výchozí: 5 seconds - minimální prodleva mezi obnovením havarovaných procesů

FastCgiConfig

- Výchozí konfigurace pro dynamické FastCGI aplikace
- Kontext: pouze **server**
- option (výběr), platí volby výše a navíc:
  - -killInterval - výchozí: 300 seconds - frekvence odstraňování dynamických FastCGI aplikací
  - -maxClassProcesses n - výchozí: 10 - maximální počet procesů jedné FastCGI aplikace
  - -maxProcesses n - výchozí: 50 - maximální počet procesů (musí být větší než voba výše)

- -minProcesses n - výchozí: 5 - minimální počet procesů aplikace
- -multiThreshold n - výchozí: 50 - číslo mezi 0 - 100, pokud je vytížení nižší než toto číslo a běží více procesů aplikace, je některý z nich ukončen
- -singleThreshold n - výchozí: 50 - číslo mezi 0 - 100, pokud je vytížení nižší než toto číslo a běží už pouze jeden proces aplikace, je ukončen
- -processSlack n - výchozí: 5 - počet procesů, který má zůstat z celkového limitu volný (přiblížení se k němu vede k ukončování málo vytížených procesů)

FastCgiIpcDir

- Cesta k adresáři se sockety. Direktiva, pokud je explicitně uvedena, musí předcházet ostatním FastCgi.* direktivám.
- FastCgiIpcDir directory
- Kontext: server
- Výchozí hodnota: FastCgiIpcDir RUNTIMEDIR/fastcgi

FastCgiWrapper

- Podpora pro suEXEC.
- FastCgiWrapper On | Off | filename
- Kontext: server
- Výchozí hodnota: Off

Základní konfigurace

Načtení modulu, povolení kontroly aktuálnosti dynamicky spouštěné aplikace (z důvodů změn). Povolení klasického CGI i FastCGI.

```
LoadModule fastcgi_module modules/mod_fastcgi.so
# nastaveni FastCGI musi vetsinou byt v hlavnim serveru, nelze jej umistit jinam!
FastCgiConfig -autoUpdate
...
<IfModule fastcgi_module>
  Alias /fcgi /var/www/main/fcgi
  <Directory /var/www/main/fcgi>
    Options +ExecCGI
    AddHandler fastcgi-script .fcgi
    AddHandler cgi-script .cgi
    Order Deny,Allow
  </Directory>
</IfModule>
```

FastCGI + suEXEC

Pro dynamické FastCGI aplikace je potřeba následující.

1. Povolit suEXEC (+ pro virtuálního hostitele nastavit identitu pomocí direktivy SuexecUserGroup user group)
2. Povolit suEXEC wrapper v mod_fastcgi direktivou FastCgiWrapper On, případně nastavit cestu k wrapperu.

FastCGI + PHP

Konfigurace PHP-CGI se neliší - používá se redirect přes direktivu Action. Interpret podporuje současně CGI i FastCGI.
Porovnání rychlosti získání výpisu funkce phpinfo() pomocí nástroje ab.

FastCGI, PHP-CGI - *dynamická aplikace*

```
y36aws ~ #  ab -c 100 -n 1500 http://localhost/php-fcgi/index.php-fcgi
...
Document Path:          /php-fcgi/index.php-fcgi
Document Length:        50935 bytes
...
Requests per second:    144.10 [#/sec] (mean)
Time per request:       693.975 [ms] (mean)
Time per request:       6.940 [ms] (mean, across all concurrent requests)
Transfer rate:          7188.97 [Kbytes/sec] received
...
```

FastCGI, PHP-CGI - *statická aplikace* - 5 dopředu spuštěných procesů.

```
FastCgiServer /var/www/main/fcgi/php.fcgi -processes 5
y36aws ~ #  ab -c 100 -n 1500 http://localhost/php-fcgi/index.php-fcgi
...
Document Path:          /php-fcgi/index.php-fcgi
Document Length:        50935 bytes
...
Requests per second:    469.71 [#/sec] (mean)
Time per request:       212.897 [ms] (mean)
Time per request:       2.129 [ms] (mean, across all concurrent requests)
Transfer rate:          23433.72 [Kbytes/sec] received
...
```

CGI, PHP-CGI

```
y36aws ~ #  ab -c 100 -n 1500 http://localhost/php-cgi/index.php-cgi
...
Document Path:          /php-cgi/index.php-cgi
Document Length:        53386 bytes
...
Requests per second:    64.11 [#/sec] (mean)
Time per request:       1559.780 [ms] (mean)
Time per request:       15.598 [ms] (mean, across all concurrent requests)
Transfer rate:          3352.25 [Kbytes/sec] received
...
```

mod_php

```
y36aws ~ #  ab -c 100 -n 1500 http://localhost/php/index.php
Document Path:          /php/index.php
Document Length:        90992 bytes
...
Requests per second:    388.69 [#/sec] (mean)
Time per request:       257.278 [ms] (mean)
Time per request:       2.573 [ms] (mean, across all concurrent requests)
Transfer rate:          34821.67 [Kbytes/sec] received
...
```

FastCGI + suEXEC + PHP

Tato konfigurace vyžaduje opět wrapper - CGI skript, který bude odpovídat požadavkům na zabezpečení a na něj se budou pomocí direktivy Action přeposílat požadavky.

```
#!/bin/bash
export PHP_FCGI_CHILDREN=0
export PHP_FCGI_MAX_REQUESTS=1000
exec /usr/bin/php-cgi
```

Pomocí wrapperů lze nastavit některé proměnné prostředí, které řídí chování PHP. V režimu FastCGi má PHP zabudovanou **vlastní** správu spouštěných procesů. Proměnné ovlivňují:

- PHP_FCGI_CHILDREN - počet obslužných procesů, které má PHP vytvořit. Pozor na http://bugs.php.net/bug.php?id=40286 (ve verzi 5.3.0 by mělo být opraveno)
- PHP_FCGI_MAX_REQUESTS - počet požadavků, které PHP může obsloužit. Pozor na http://bugs.php.net/bug.php?id=27802

Použití nedokumentovaných proměnných je rizikové i když může takové nastavení být výkonnější, spolehlivější je použít process manager z mod_fastcgi (tj. tyto proměnné prostředí vůbec nenastavovat).

mod_fcgi

Oblíbená alternativa k *mod_fastcgi*. Řeší některé problémy, např. procesům umožňuje komunikaci přes sdílenou paměť a další.

05 – Bezpečnost

Autentizace, autorizace, řízení přístupu

Apache httpd rozděluje proces kontroly oprávnění na 3 části:
- **autentizace** klienta - ověření, že klient je tím, za koho se vydává,
- **autorizace** klienta - ověření, že klient má na zdroj práva,
- **kontrola přístupu** - ověření, dalších podmínek omezujících použití zdroje.

Od verze 2.2 *Apache httpd* se používá nové rozdělení modulů podle toho, v jaké části se používají.
- auth_ - typ autentizace - HTTP Basic nebo HTTP Digest,
- auth**n**_ - poskytovatel autentizace - provádí vlastní ověření identity klienta,
- auth**z**_ - poskytovatel autorizace - rozhoduje o právu klienta na přístup k požadovanému zdroji,
- auth**nz**_ - poskytovatel autentizace i autorizace.

Ve starších verzích *Apache httpd* toto nebylo rozlišováno!

Autentizace

Metody HTTP autentizace definuje RFC 2617 *HTTP Authentication: Basic and Digest Access Authentication*.
- ***AuthType***
  - Určuje typ - basic / digest
  - AuthType Basic|Digest
  - kontext: directory, .htaccess
- ***AuthName***
  - Určuje realm - název oblasti, pro kterou je vyžadována autentizace
  - AuthName realm
  - kontext: directory, .htaccess

HTTP Basic

http://en.wikipedia.org/wiki/Basic_access_authentication
Informace pro autentizaci jsou odeslány spolu s požadavkem v HTTP hlavičce. Hlavička je ve formátu uzivatel:heslo a je zakódována ve formátu *base64*. Kódování se používá aby nedošlo k porušení dat protokolu HTTP a **neznamená** žádnou bezpečnost!

```
y36aws ~ # echo -n 'y36aws:tajneheslo' | uuencode -m -
begin-base64 644 -
eTM2YXdzOnRham5laGVzbG8=
====
```

Průběh autentizace:
1. Klient požádá o zdroj bez toho, aby uvedl přihlašovací údaje.
2. GET /private/index.html HTTP/1.0
   Host: www.mycorp.k328
3. Dostane odpověď HTTP 401 UNAUTHORIZED.
4. HTTP/1.0 401 UNAUTHORIZED
5. Server: Apache
6. Date: Sat, 24 Nov 2008 9:18:15 CET
7. WWW-Authenticate: Basic realm="Zabezpeceno"
   ...
8. Klient zobrazí realm a pole pro vložení přihlašovacích údajů.
9. Po zadání přihlašovacích údajů klient odešle odpověď.
10. GET /private/index.html HTTP/1.0
11. Host: www.mycorp.k328
    Authorization: Basic eTM2YXdzOnRham5laGVzbG8=
12. Podle konfigurace server odpoví HTTP 200 OK, nebo opět HTTP 401 UNAUTHORIZED.

Direktivy
- ***AuthBasicAuthoritative***
  - Určuje chování, pokud uživatele nebude schopen ověřit žádný nakonfigurovaný poskytovatel - pokud je On, uživatel v takovém případě není ověřen.
  - AuthBasicAuthoritative On|Off
  - výchozí: AuthBasicAuthoritative On
  - kontext: directory, .htaccess
- ***AuthBasicProvider***

- Seznam poskytovatelů oddělených mezerou, kteří se mohou použít pro ověření uživatele.
- AuthBasicProvider Directive provider [provider] …
- výchozí: AuthBasicProvider file
- kontext: directory, .htaccess

HTTP Digest Authentication

Použití HTTP Digest Authentication je problematické, nemusí být podporováno prohlížeči (alespoň ve starších verzích). S HTTP Digest též často neumí spolupracovat poskytovatelé autentizace! Stejně tak metoda neřeší např. útok man-in-the-middle. V důsledku se tak spíše používá HTTP Basic na zabezpečeném spojení nebo autentizace v rámci webové aplikace.

Metoda se snaží o větší zabezpečení, neodesílá heslo jako plaintext jako v případě HTTP Basic Authentication, ale využívá MD5 funkce a nonce (number used once) pro zabránění např. útokům, které opakují komunikaci nebo kryptoanalýze.

Ověření se provádí tak, že se pomocí hashovací funkce vypočítá:
- H1 = md5(username:realm:password),
- H2 = md5(method:digestURI),
- response = md5(H1:nonce:H2).

Číslo nonce určuje webserver a zasílá ho v odpovědi HTTP 401 na úvodní požadavek jako obsah hlavičky WWW-Authenticate.

Postup autentizace je dále podobný HTTP Basic Authentication, pouze jsou vyměňovány jiné informace v hlavičkách.
1. Klient požádá o zdroj bez toho, aby uvedl přihlašovací údaje.
2. GET /private-digest/index.html HTTP/1.0
   Host: www.mycorp.k328
3. Dostane odpověď HTTP 401 UNAUTHORIZED.
4. HTTP/1.0 401 UNAUTHORIZED
5. Server: Apache
6. Date: Sat, 24 Nov 2008 9:18:15 CET
7. WWW-Authenticate: Digest realm="Zabezpeceno",
   nonce="rCILfGxcBAA=75f715e2f17a607a2d5760692c755c1d142a06a7", algorithm=MD5,
   domain="/private"
   ...
8. Klient zobrazí realm a pole pro vložení přihlašovacích údajů.
9. Po zadání přihlašovacích údajů klient odešle odpověď.
10. GET /private-digest/index.html HTTP/1.0
11. Host: www.mycorp.k328
    Authorization: Digest username="y36aws", realm="Zabezpeceno",
    nonce="rCILfGxcBAA=75f715e2f17a607a2d5760692c755c1d142a06a7", uri="/private-digest/index.html",
    algorithm=MD5, response="cbcb5ab6b72ff3f8a9cff3073fccafd3", qop=auth, nc=00000001,
    cnonce="4c45d62bd34bb841"
12. Server ověří response (dokáže dopředu spočítat výsledek) a odpoví HTTP 200 OK, nebo opět HTTP 401 UNAUTHORIZED.

Direktivy

Direktivy jsou podobné HTTP Basic Authentication, podrobně viz dokumentace.
- ***AuthDigestDomain***
  - Umožňuje nastavit více domén (přístup přes různé DNS názvy) - DNS název je součástí výpočtu!
  - AuthDigestDomain name name …
  - kontext: directory, .htaccess

mod_authn_default

Fall-back modul, který umožňuje odmítnout autentizaci uživatele v případě, že není nakonfigurován žádný modul pro autentizaci (tj. mod_auth_{basic,digest}) nebo ji předat dále do nižších vrstev.
- ***AuthDefaultAuthoritative***
  - Určuje, zda se má autentizace uživatele odmítnout, nebo má být její zpracování předáno do nižších vrstev.
  - AuthDefaultAuthoritative On|Off
  - výchozí:AuthDefaultAuthoritative On

- kontext: directory, .htaccess

mod_authn_file

Modul umožňuje autentizaci podle textového souboru. Formát souboru se liší pro HTTP Basic a Digest autentizaci. Soubor pro HTTP Basic autentizaci obsahuje na každé řádce záznam o jednom uživateli ve tvaru uzivatel:heslo. Heslo je uloženo jako MD5, SHA hash nebo výsledek volání funkce crypt(). V souboru lze míchat různé typy uložení hesla. Soubor lze udržovat pomocí nástroje htpasswd.
Soubor pro HTTP Digest autentizaci obsahuje obdobné údaje. Řádek má tvar uziv_jmeno:realm:H1, kde H1 = md5(uziv_jmeno:realm:heslo). V jednom souboru lze uchovávat údaje pro různé hodnoty realm. Soubor je možné udržovat nástrojem htdigest.

- ***AuthUserFile***
  - Nastavuje cestu k souboru s uživateli.
  - AuthUserFile /cesta/k/.htpasswd
  - kontext: directory, .htaccess

Příklad konfigurace

```
Alias /private /var/www/main/private
<Directory /var/www/main/private>
  <IfModule auth_basic_module>
  <IfModule authn_file_module>
    AuthType basic
    AuthName "Zabezpeceno"
    AuthBasicProvider file
    AuthBasicAuthoritative On
    AuthUserFile /var/www/main/private/.htpasswd
    Require valid-user
  </IfModule>
  </IfModule>

  Order Deny,Allow
</Directory>
```

HTTP Basic Authentication & mod_authn_file

```
Alias /private-digest /var/www/localhost/private-digest
<Directory /var/www/localhost/private-digest>
  <IfModule auth_digest_module>
  <IfModule authn_file_module>
    AuthType digest
    AuthName "Zabezpeceno"
    AuthDigestProvider file
    AuthUserFile /var/www/localhost/private-digest/.htdig
    Require valid-user
  </IfModule>
  </IfModule>

  Order Deny,Allow
</Directory>
```

HTTP Digest Authentication & mod_authn_file

Autorizace

Autorizaci zajišťují direktivy z core.

- ***Require***
  - Vyžaduje splnění kritéria - user, valid-user, group.
  - Require entity-name,
  - kontext: directory, .htaccess
- ***Satify***
  - Vyžaduje splnění alespoň nějakého kritéria
  - Satify Any|All
  - výchozí: Satify All
  - kontext: directory, .htaccess

Pro ověřování uživatelů a skupin je potřeba dodat příslušné databáze - třeba soubory.

mod_authz_groupfile

Modul umožňuje autorizaci podle členství ve skupině - uvedeny v textovém souboru. Formát skupina: uzivatel uzivatel ….

- ***AuthGroupFile***
  - Nastavuje cestu k souboru se skupinami.
  - AuthUserFile /cesta/k/.htgroup
  - kontext: directory, .htaccess

Kontrola přístupu

Základní direktivy pro kontrolu přístupu poskytuje modul *mod_authz_host* - tj. autorizace na základě informací o klientově IP adrese a DNS jménu.

- ***Allow from***
  - Povolí přístup z vybraných IP adres, DNS jmen, může být použita vícekrát.
  - Allow from all|host
  - kontext: directory, .htaccess
- ***Deny from***
  - Zakáže přístup z vybraných IP adres, DNS jmen, může být použita vícekrát.
  - Deny from all|host
  - kontext: directory, .htaccess
- ***Order***
  - Nastavuje pořadí použití direktiv Allow a Deny, pokud není shoda, výchozí je vždy ta akce, která je uvedena jako druhá!
  - Order ordering
  - výchozí: Order Deny,Allow - tj. **povoluje** přístup kamkoliv!
  - kontext: server, vhost, directory, location, .htaccess

Výsledné oprávnění ke zdroji je určeno podle následující tabulky - záleží, v jakých částech dojde ke shodě a v jakém pořadí se budou aplikovat.

Shoda	Allow,Deny - výsledek	Deny,Allow - výsledek
pouze Allow	povoleno	povoleno
pouze Deny	zakázáno	zakázáno
neshoda	zakázáno	povoleno
obojí	zakázáno	povoleno

Secure Sockets Layer (SSL), Transport Layer Security (TLS)

Secure Sockets Layer (vrstva bezpečných socketů) a *Transport Layer Security* (zabezpečení transportní vrstvy) jsou *protokoly* (resp. vrstvy), které lze vložit mezi transportní a aplikační vrstvu *TCP/IP stacku*. Taková vrstva pak poskytuje zabezpečení komunikace autentizací a šifrováním.

Klíče, certifikáty, certifikační autority

Asymetrická kryptografie je založena na problémech, které jsou efektivně řešitelné (v polynomiálním čase) v případě že je známa tajné informace, ale prakticky neřešitelné v opačném případě. Účastníci kominikace disponují:

- *privátním klíčem*, který slouží k dekódování dat zakodovaných veřejným klíčem, digitálnímu podpisu,
- *veřejným klíčem*, který dávají veřejně k dispozici, aby jím mohla být zakódována zpráva nebo ověřen digitální podpis
- *certifikátem*, který obsahuje veřejný klíč a další informace umožňující identifikovat majitele a případně ověřit, že certifikát je pravý (digitálně podepsaný certifikfační autoritou). *Certifikát* obsahuje mimo jiné: komu byl vydán, vydavatele (certifikační autorita), dobu platnosti, administrativní informace. Informace o vlastníkovi a certifikační autoritě jsou uloženy ve formě *Distinguished Name*

DN pole	zkratka	popis	příklad
Common Name	CN	certifikované jméno	CN=server.felk.cvut.cz
Organization or Company	O	jméno organizace	O=FEE CTU in Prague
Organizational Unit	OU	jméno organizační jednotky	OU=Department of Computer Science and Engineering
City/Locality	L	jméno města	L=Prague
State/Province	ST	jméno kraje	ST=Prague
Country	C	jméno země (ISO code)	C=CZ

Formát uložení

Klíče a certifikáty se ukládají binárně a to ve 2 formátech. Formát DER, obsahuje přímo nezakódovaná binární data a používá se tam, kde přenos libovolných binárních dat není problematický. Formát PEM kóduje binární data pomocí Base64 tak, aby se zobrazila do tisknutelných znaků ASCII z důvodů bezpečného přenosu.
OpenSSL implicitně používá formát PEM (stejně jako Apache httpd a další serverový software), ale pro některé aplikace (např. MS Outlook) může být vyžadován formát DER. Pro převod lze použít opět *OpenSSL*.

y36aws ~ # openssl x509 -in cert.pem -out cert.der -outform DER

Certifikační autorita

Certifikační autorita (CA) je objekt, který vydává *certifikáty* ostatním účastníkům. V *Public Key Infrastructure* (PKI) má z toho důvodu zásadní roli - zajišťuje ověření pravosti certifikátů.

Certificate Chain

Řetězec certifikátů. Certifikát certifikační autority mohl být vydán jinou CA. Při ověření certifikátu se pak postupuje hierarchií certifikátu tak dlouho, než se nalezne certifikát důvěryhodné CA - všechny podřazené CA jsou brány jako důvěryhodné.

Root-level CA

Kořenová CA má *self-signed* certifikát - není zde nikdo, kdo by jej mohl ověřit! Zabezpečení těchto certifikátů proti zneužití je zajištěno jejich publikací na více místech, čímž se sníží riziko, že by se někdo mohl vydávat za danou CA.
Zdarma poskytuje podepsané certifikáty pro servery a osoby CAcert.

Příklady

Self-signed certifikát

Vygenerování klíče (RSA) a certifikátu (X.509)

```
 y36aws /etc/apache2/ssl # openssl req -new -x509 -nodes -out server.crt -keyout server.key -days 365
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'server.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CZ
State or Province Name (full name) [Some-State]:Prague
Locality Name (eg, city) []:Prague
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FEE CTU in Prague
Organizational Unit Name (eg, section) []:DCSE
Common Name (eg, YOUR name) []:localhost
Email Address []:local@local.local
```

Ověření informací v certifikátu.

```
 y36aws /etc/apache2/ssl # openssl x509 -noout -text -in server.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            81:65:f8:69:62:31:7c:c8
        Signature Algorithm: sha1WithRSAEncryption
```

```
        Issuer: C=CZ, ST=Prague, L=Prague, O=FEE CTU in Prague, OU=DCSE,
CN=localhost/emailAddress=local@local.local
        Validity
            Not Before: Dec  1 09:19:53 2008 GMT
            Not After : Dec  1 09:19:53 2009 GMT
        Subject: C=CZ, ST=Prague, L=Prague, O=FEE CTU in Prague, OU=DCSE,
CN=localhost/emailAddress=local@local.local
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:a7:47:12:64:62:23:41:88:ce:9e:7a:5c:50:f9:
                    1b:3e:3e:86:ca:57:45:34:84:2b:69:66:1a:9a:07:
                    18:2d:00:ea:08:10:9a:08:2d:c7:f9:fe:4d:5c:63:
                    8e:de:a2:c8:e1:df:73:e3:0a:ea:ea:8f:9e:d5:ae:
                    54:e0:c1:27:a2:9a:af:18:4d:7a:26:e8:66:6a:e7:
                    53:ae:4e:57:d1:ab:1a:a6:55:1f:58:8f:3a:6e:b6:
                    ee:a3:ca:ce:b1:b6:39:e9:8a:c8:34:d4:7a:db:51:
                    a8:f7:18:f3:73:2d:28:f0:2e:80:05:db:ad:f1:07:
                    10:c4:85:fd:5a:78:35:0e:9d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                02:BB:22:B4:21:94:D3:B1:A7:EF:1B:59:78:0E:1C:6A:5F:01:3D:F7
            X509v3 Authority Key Identifier:
                keyid:02:BB:22:B4:21:94:D3:B1:A7:EF:1B:59:78:0E:1C:6A:5F:01:3D:F7
                DirName:/C=CZ/ST=Prague/L=Prague/O=FEE CTU in
Prague/OU=DCSE/CN=localhost/emailAddress=local@local.local
                serial:81:65:F8:69:62:31:7C:C8

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
        a1:08:f2:6a:da:5f:1e:bd:79:5d:bc:17:59:8f:02:b6:5b:79:
        12:23:da:c5:33:68:e4:df:c2:ac:95:c1:d8:f1:93:ac:ef:d0:
        bb:02:40:9b:3d:0f:f7:e7:e7:88:30:1a:e6:ea:b4:83:6b:b6:
        83:f1:6c:ef:33:fb:a4:a1:8a:70:53:f3:0d:2e:b1:0a:fa:cb:
        bc:a0:b3:a1:97:6d:2f:fd:44:52:6a:e2:3d:52:1c:a5:80:9d:
        85:ef:00:34:9a:1f:02:24:05:74:97:bd:57:16:86:c9:69:0c:
        ee:dd:ec:d5:58:be:a4:82:ed:16:04:df:e8:ac:9d:ae:77:33:
        d1:f6
```

Vlastní CA

Konfigurace OpenSSL

```
[ ca ]
default_ca      = CA_y36aws             # The default ca section

####################################################################
[ CA_y36aws ]

dir             = ./y36awsCA            # Where everything is kept
certs           = $dir/certs            # Where the issued certs are kept
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt        # database index file.
#unique_subject = no                    # Set to 'no' to allow creation of
                                        # several ctificates with same subject.
new_certs_dir   = $dir/newcerts         # default place for new certs.

certificate     = $dir/cacert.pem       # The CA certificate
serial          = $dir/serial           # The current serial number
crlnumber       = $dir/crlnumber        # the current crl number
                                        # must be commented out to leave a V1 CRL
crl             = $dir/crl.pem          # The current CRL
private_key     = $dir/private/cakey.pem# The private key
```

RANDFILE = $dir/private/.rand # private random number file
/etc/ssl/openssl.conf
Vytvoření adresářů a souborů.
y36aws ~ # cd /etc/ssl/y36awsCA
y36aws /etc/ssl/y36awsCA # mkdir certs crl newcerts private
y36aws /etc/ssl/y36awsCA # touch index.txt
y36aws /etc/ssl/y36awsCA # echo 01 > serial
y36aws /etc/ssl/y36awsCA # cd ..
Vygenerování klíče a certifikátu CA - self-signed.
y36aws /etc/ssl # openssl req -new -x509 -nodes -out y36awsCA/cacert.pem -keyout
y36awsCA/private/cakey.pem -days 1098

Žádost o certifikát

Vytvoření žádosti.
y36aws /etc/ssl # openssl req -new -nodes -out /etc/apache2/ssl/server12.req -keyout
etc/apache2/ssl/server12.key
Podepsání žádosti.
y36aws /etc/ssl # openssl ca -in /etc/apache2/ssl/server12.req -out /etc/apache2/ssl/server12.crt

Protokol SSL/TLS

Protokol SSL/TLS umožňuje aplikacím komunikovat po síti způsobem, který zabraňuje odposlouchávání či falšování zpráv. Pomocí kryptografie poskytuje SSL/TLS svým koncovým bodům autentizaci a soukromí při komunikaci přes nezabezpečené sítě. Typicky je autentizován pouze server - jeho totožnost je zaručena. Klient zůstává neautentizován, ale má jistotu o identitě protistrany. Další úroveň zabezpečení – při níž oba konce „konverzace“ mají jistotu s kým komunikují – je označována jako vzájemná autentizace. Vzájemná autentizace vyžaduje nasazení infrastruktury veřejných klíčů (PKI) pro klienty.

Protokol SSL navrhl Netscape, v současnosti existují 2 používané verze - SSLv2 a SSLv3. SSLv3 v podstatě odpovídá TLS 1.0 protokolu navrženém IETF - RFC 4346.

SSL/TLS využívá asymetrické kryptografie - každá z komunikujících stran má dvojici šifrovacích klíčů - veřejný a soukromý. Veřejný klíč je možné zveřejnit a pokud tímto klíčem kdokoliv zašifruje nějakou zprávu, je zajištěno, že ji bude moci rozšifrovat jen majitel použitého veřejného klíče svým soukromým klíčem.

Komunikace pomocí SSL/TLS vyžaduje vytvoření *session* (sezení) mezi komunikujícími stranami.
1. Handshake (podání rukou).
2. Komunikace - šifrování provozu symetrickou šifrou (nižší nároky na HW).

Handshake:
1. Dohoda o šifrách, které so budou používat v průběhu přenostu.
2. Vytvoření *session* mezi komunikujícími stranami.
3. [volitelně] Autentizace serveru (aby si byl klient jist, s kým komunikuje).
4. [volitelně] Autentizace klienta (aby měl server jistotu, kdo je klient).

Mezi kryptografické algoritmy, které lze v dnešní implementaci protokolů běžně použít jsou:
- pro výměnu klíčů: RSA, Diffie-Hellman, DSA nebo Fortezza;
- pro symetrickou šifru: RC2, RC4, IDEA, DES, 3DES nebo AES;
- pro jednocestné hašovací funkce: MD5 nebo SHA-1, SHA-2.

Podrobný popis protokolů SSL a TLS je mimo rámec tohoto textu, případné zájemce odkazujeme na zdroje uvedené pod textem.

Bezpečnost

Pro zajištění bezpečnosti a omezení možných útoků jako např. man-in-the-middle SSL/TLS zahrnuje řadu bezpečnostních opatření:
- Klient používá veřejný klíč certifikační autority (CA) k ověření jejího digitálního podpisu v serverovém certifikátu. Lze-li digitální podpis CA ověřit, klient přijme serverový certifikát jako platný certifikát vydaný důvěryhodnou CA.
- Klient ověřuje, zda je vydávající certifikační autorita na seznamu důvěryhodných CA.
- Klient kontroluje dobu životnosti serverového certifikátu. Autentizační proces se zastaví, pokud doba jeho platnosti vypršela.

- K ochraně před útoky typu Man-in-the-Middle porovnává klient aktuální DNS jméno serveru se jménem z certifikátu.
- Ochrana před několika známými útoky (včetně Man-in-the-Middle), jako je snaha o použití nižší (méně bezpečné) verze protokolu nebo slabšího šifrovacího algoritmu.
- Opatření všech aplikačních záznamů pořadovými čísly a používání těchto čísel v MAC.
- Používání ověřovacího kódu zprávy rozšířeného o klíč, takže jen vlastník klíče dokáže MAC ověřit. Definováno v RFC 2104. Jen v TLS.
- Zpráva ukončující inicializaci (Finished) obsahuje hash všech zpráv vyměněných v rámci inicializace oběma stranami.
- Pseudonáhodná funkce rozděluje vstupní data na poloviny a zpracovává každou z nich jiným hashovacím algoritmem (MD5 a SHA-1), pak je XORuje dohromady. To poskytuje ochranu, pokud by byla nalezena slabina jednoho z algoritmů. Jen v TLS.
- SSL v3 je proti SSL v2 vylepšeno přidáním šifer založených na SHA-1 a podporou autentizace certifikáty. Další vylepšení SSL v3 zahrnují lepší inicializační protokol a vyšší odolnost proti útokům typu man-in-the-middle.

HTTPs

Je URI schéma označující zabezpečenou komunikaci protokolem HTTP. HTTPs vlastně není protokol, označuje pouze kombinaci HTTP a SSL nebo TLS. Tato kombinaci zajišťuje postačující bezpečnost proti odposlechnutí, není však zcela bezpečná proti útoku man-in-the-middle. K zabránění takovému útoku se používá např. digitální podpis nebo uveřejnění otisku certifikátu.
Bezpečnost je značně závislá na kryptografických algoritmech podporovaných serverem a prohlížečem.

SSL a virtual hosting

SSL nemá žádnou znalost o protokolech vyších vrstev. Z toho důvodu SSL servery mohou používat pouze jeden certifikát pro IP adresu a port. Z toho plyne, že typicky není možné použít SSL a name-based virtual hosting.
Apache httpd 2.2 SSL FAQ.
RFC 3546 rozšiřuje protokol TLS o řešení, které umožní identifikovat server už na úrovni TLS vrstvy - *Server Name Indication* - je podporováno v prohlížečích Mozilla Firefox 2.0+, MS Internet Explorer 7 ve Windows Vista a Opera 8. Pro *Apache httpd* existuje zatím podpora SNI pouze ve formě patche pro *mod_ssl*.

Konfigurace Apache httpd

Implementací SSL pro *Apache httpd* je k dispozici několik. Pro účely předmětu byl vybrán *mod_ssl*, který je standardní součástí *Apache httpd* 2.2. Modul využívá OpenSSL.
Modul je před použitím potřeba načíst.

```
<IfDefine SSL>
  LoadModule ssl_module modules/mod_ssl.so
</IfDefine>
```

Direktivy mod_ssl

Pro CGI/SSI mohou být potřeba proměnné prostředí, ty je nutné povolit v direktivě SSLOptions - z výkonnostních důvodů standardně nejsou nastaveny. Seznam proměnných je k dispozici např. na http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#envvars.

- ***SSLEngine***
  - Zapne/vypne SSL, typicky se používá ve virtuálních hostitelích. Hodnota optional umožňuje přejít na TLS, pokud jej klient podporuje (zatím žádný takový není).
  - SSLEngine On|Off|optional
  - Kontext: server, vhost
- ***SSLOptions***
  - Nastavuje různé možnosti SSL,
  - SSLOptions [+|-] option …
  - Kontext: server, vhost, directory, .htaccess
  - Možnosti (výběr):
    - StdEnvVars - exportuje standardní proměnné prostředí spojené se SSL,
    - ExportCertData - do proměnných prostředí uloží certifikáty klienta a serveru,
    - FakeBasicAuth - použije certifikát přijatý od klienta k napodobení HTTP Basic autentizace.
- ***SSLProtocol***
  - Určuje verzi protokolu - *SSLv2, SSLv3, TLSv1, All*.
  - SSLProtocol [+|-]protocol
  - Výchozí: SSLProtocol All

    - Kontext: server, vhost
    - SSLProtocol all -SSLv2
- ***SSLCipherSuite***
    - Určuje šifry, které se mohou použít při dojednání s klientem.
    - SSLCipherSuite cipher-spec
    - Výchozí: SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
    - Kontext: server, vhost, directory, .htaccess
- ***SSLCertificateFile***
    - Cesta k souboru s certifikátem serveru v PEM formátu. Může obsahovat rovnou i klíč, to se ale nedoporučuje.
    - SSLCertificateFile /cesta/k/certifikatu.crt
    - Kontext: server, vhost
- ***SSLCertificateKeyFile***
    - Cesta k souboru s privátním klíčem serveru v PEM formátu.
    - SSLCertificateKeyFile /cesta/k/certifikatu.key
    - Kontext: server, vhost
- ***SSLCertificateChainFile***
    - Cesta k souboru, který obsahuje řetězec certifikátů CA, které vydaly certifikát serveru.
    - SSLCertificateChainFile /cesta/k/ca.crt
    - Kontext: server, vhost
- ***SSLRequireSSL***
    - Direktiva vynucuje SSL přístup ke zdroji.
    - SSLRequireSSL
    - Kontext: directory, .htaccess
    -
- ***SSLRequire***
    - Udává požadavky, které musí být splněny při komunikaci - boolovská podmínka - lze používat proměnné prostředí.
    - SSLRequire true | false | ! expr | expr && expr | expr || expr …
    - Kontext: directory, .htaccess
    - Příklad: SSLRequire %{SSL_CIPHER_USEKEYSIZE} >= 128

Logování

Povolení *mod_ssl* přináší další možnosti pro formát direktivy CustomLog.

Možnost	Popis
%…{version}c	verze protokolu SSL
%…{cipher}c	šifra
%…{subjectdn}c	DN klienta z certifikátu klienta
%…{issuerdn}c	DN vydavatele z certifikátu klienta
%…{errcode}c	Certificate Verification Error (numerical)
%…{errstr}c	Certificate Verification Error (string)

Příklady

Základní konfigurace
Základní konfigurace je stejná, pokud používáme self-signed certifikát nebo certifikát podepsaný CA. Certifikát, resp. klíč je uložen v souboru server.crt, resp. server.key.

```
<VirtualHost IPadresa:443>
  ...
  # SSL povoleno
  SSLEngine on
  # SSL Cipher Suite:
  SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
  # Server Certificate (RSA):
  SSLCertificateFile /etc/apache2/ssl/server.crt
  # Server Private Key (RSA):
  SSLCertificateKeyFile /etc/apache2/ssl/server.key
  # Nektere aplikace mohou vyzadovat nastaveni promennych prostredi
  <FilesMatch "\.(cgi|shtml|phtml|php)$">
      SSLOptions +StdEnvVars
```

```
    </FilesMatch>
    <Directory "/var/www/localhost/cgi-bin">
        SSLOptions +StdEnvVars
    </Directory>
    # SSL Protocol Adjustments:
    <IfModule setenvif_module>
        BrowserMatch ".*MSIE.*" nokeepalive ssl-unclean-shutdown downgrade-1.0 force-response-1.0
    </IfModule>
    # Per-Server Logging:
    <IfModule log_config_module>
        CustomLog /var/log/apache2/ssl_1_1_request_log "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\"
%b"
    </IfModule>
</VirtualHost>
```

Proxy

- Použití proxy
- Typy proxy serverů
- Základní direktivy
- Konfigurace
- Zabezpečení
- Loadbalancing
- Odkládání

Server proxy je systém, který je jakýmsi prostředníkem mezi klientskými hostiteli a servery, ke kterým přistupují. Když například klientský hostitel požádá o obsah nějakého vzdáleného serveru, server proxy přijme tento požadavek a sám obsah získá, aby ho pak následně předal klientovi, který o něj požádal. Server proxy může zároveň tento požadavek uložit do paměti a až bude opět požadován od nějakého klientského hostitele, tak tento požadavek server proxy předá ze své paměti. Tím může výrazně snížit využitou šířku pásma. Toto je nejzákladnější funkčnost serveru proxy. Jelikož však veškerý obsah, který je požadován ze vzdálených zdrojů, musí jít přes server proxy, můžeme tak velice efektivně řídit, zaznamenávat a vyhodnocovat síťový provoz. A k tomu je server proxy určen.

Využití proxy serveru

- Zastupování - Zpřístupnění nesměrovatelných adres
- Odkládání - Zrychlení přístupových dob
  - Snížení použité šířky pásma
- Monitorování - Monitorování využití Internetu
- Modifikace obsahu - Přidání reklamy
  - Filtrování
- Řízení přístupu - Omezení zneužívání pracovní doby
  - Omezení přístupu na určené servery
  - Autorizace přistupu
- Přesměrování - Nahrazení požadavků
  - Anonymizer

Forward proxy

1. Uživatel předá požadavek na hostitele proxy serveru
2. Proxy server získá zamýšlenou odpověď od hostitele
3. Proxy server předá odpověď uživateli

Reverse proxy

1. Proxy serveru přijde požadavek (žadatel neví, že jde o proxy server, myslí že žádá přímo zdroj)
2. Proxy server obslouží uživatele z paměti, nebo požádá hostitele o odpověď

Další typy proxy serverů

- Intercepting proxy (Transparent)
  - Uživatel odešle požadavek na server skrz gateway, která se chová jako proxy (není nutné konfigurovat prohlížeč)
- transparent a nontransparent

  transparent proxy je proxy, která nemodifikuje http request, nebo response

  nontransparent proxy je proxy, která za nějakým účelem modifikuje obsah http request, nebo response (poskytování služeb, komprimace, filtrování)

- Anonymizing proxy server , Open proxy server, Hostile proxy, Circumventor

Tyto typy proxy serverů slouží zejména k anonymnímu využívání internetu, jsou často zneužívány pro nelegální činnost (distribuce software, spam).
Circumventor je metoda jak obejít přístupová omezení. Circumventor může být webová stránka, která zpřístupňuje jinou blokovanou stránku. Např. v číně elgooG, který tak dovoluje čínským uživatelům používat Google.

mod_proxy

- mod_proxy_http - proxying HTTP requests (HTTP/0.9, HTTP/1.0, HTTP/1.1.)
- mod_proxy_ftp - proxying FTP sites
- mod_proxy_ajp - podpora pro Apache JServ Protocol
- mod_proxy_balancer -podpora pro load balancing
- mod_proxy_connect - podpora pro CONNECT HTTP (tunelování ssl)

Forward proxy – Direktivy

● **ProxyRequest** - Aktivuje nebo deaktivuje službu proxy
- **`ProxyRequest On | Off`**

● **ProxyVia** - Slouží k modifikaci HTTP hlavičky via
- **`ProxyVia On|Off|Full|Block`**
- Pokud hlavičky chceme modifikovat zapíšeme On (default Off)
- Parametr full navíc do hlavičky doplní verzi Apache
- Parametr Block všechny Via hlavičky odstraní

● **NoProxy** - Nastavuje přístup bez použití proxy **`NoProxy host`**
- NoProxy .othercompany2.k328 192.168.112.0/24

● **ProxyBlock** - Blokuje přístup ke specifikovanému hostiteli, doméně
- Např.: ProxyBlock idnes.cz super.cz
- **`ProxyBlock *|word|host|domain`**

● **ProxyDomain** - Názvy, které se nepodaří vyhodnotit se doplní o ProxyDomain
- **`ProxyDomain Domain`**

PŘ

```
ProxyDomain .cvut.cz
# pozadavek http://feld/departments/departments.html preozi na
http://feld.cvut.cz/departments/departments.html
```

● Kontejner **<Proxy>** - aplikuje direktivy na danou proxy
- <Proxy url> ...</Proxy>

Forward proxy – konfigurace - Příklad

Ke zprovoznění forward proxy nám v podstatě stačí jen nastavit direktivu ProxyRequests On, neměli bychom však naší proxy nikdy nechávat nezabezpečenou!

```
ProxyRequests On
ProxyVia On
<Proxy *>
      Order deny,allow
      Deny from all
      Allow from 192.168.1
</Proxy>
```

● Zabezpečení přístupu je možné také pomocí mod_auth

Reverse proxy – Direktivy

Není potřeba nastavovat ProxyRequest na On
Pokud používáme pouze reverse proxy měla by být nastavena na Off

● **ProxyPass** - mapuje adresáře vzdáleného serveru na lokální adresu
- **`ProxyPass /mirror http://www.mirror.server.com`**

- užitečné volby ProxyPass
  - min - minimální počet otevřených spojení
  - max - maximální počet připojení
  - smax - soft maximum
  - acquire - maximální doba v ms, kdy se čeká na volné připojení
  - connectiontimeout - timeout v sekundách

● **ProxySet**

- direktiva umožňuje nastavit parametry pro proxy balancer, které se normálně nastavují pomocí ProxyPass

```
<Proxy balancer://hotcluster>
  BalancerMember http://www2.example.com:8009 loadfactor=1
  BalancerMember http://www3.example.com:8009 loadfactor=2
  ProxySet lbmethod=bytraffic
</Proxy>

ProxySet balancer://foo lbmethod=bytraffic timeout=15
```

● **ProxyPassMatch** - stejná funkcionalita jako ProxyPass, využívá však regulární výrazy

```
# všechny obrazky .jpg bude stahovat ze serveru images.server.com
ProxyPassMatch ^(/.*\.jpg)$ http://images.server.com$1
```

● **ProxyPassReverse** - upraví hlavičku HTTP, aby vypadala, že opravdu pochází ze vzdáleného serveru
- Stejný syntax jako ProxyPass
- PŘ **`ProxyPassReverse /mirror http://www.mirror.server.com`**

Reverse proxy – konfigurace - Příklad

Není potřeba nastavovat ProxyRequest na On
Pokud používáme pouze reverse proxy měla by být nastavena na Off

Pokud chceme používat apache jen jako reverse proxy, měli bychom nastavit ProxyRequests Off (zakážeme forward proxy), pak ke zprovoznění stačí nastavit ProxyPass a případně ProxyPassReverse.

ProxyRequests Off

```
<Proxy *>
      Order deny,allow
      Allow from all
</Proxy>

ProxyPass /foo http://foo.example.com
ProxyPassReverse /foo http://foo.example.com
```

Load Balancing = Rozkládání zátěže - mod_proxy_balancer

load balancing je technika umožňující rozložení zátěže na více serverů. Toto rozložení nám umožňuje zvýšení kvality služeb zejména dostupnosti a času odezvy. Proxy server přímá požadavky od externích klientů a dle aktuálního zatížení si vyžádá obsah od jednoho z redundatních backend serverů. Load balancer je tedy varianta reverse proxy, klient o této proxy neví a z jeho hlediska žádá přímo jediný zdroj, narozdíl od klasické reverse proxy však load balancer nezískává obsah od backend serverů podle url požadavku, ale pomocí plánovacích algoritmů dle zatížení jednotlivých redundatních backend serverů.

`<Proxy balancer://>…</Proxy>`

- Algoritmy pro plánování rozložení zátěže
  - Request Counting
  - Weighted Traffic Counting
  - Pending Request Counting

```
lbmethod = byrequests | bytraffic | bybusyness
```

ProxyPass

- pomocí této direktivy v kombinaci s kontejnerem <proxy> nastavujeme balancery

```
ProxyPass /special-area http://special.example.com/ smax=5 max=10
ProxyPass / balancer://hotcluster/
```

ProxySet

- umožňuje nastavit parametry pro proxy balancer, které se normálně nastavují pomocí ProxyPass
- kontext: directory

```
<Proxy balancer://hotcluster>
  BalancerMember http://www2.example.com:8009 loadfactor=1
  BalancerMember http://www3.example.com:8009 loadfactor=2
  ProxySet lbmethod=bytraffic
</Proxy>

<Proxy http://backend>
  ProxySet keepalive=On
</Proxy>
ProxySet balancer://foo lbmethod=bytraffic timeout=15
```

Příklad konfigurace

```
<Proxy balancer://twoServers>
    BalancerMember http://192.168.1.10:80 smax=10
    BalancerMember http://192.168.1.20:80 smax=5 loadfactor=2
</Proxy>

ProxyPass /www balancer://twoServers/
```

Odkládání (caching)

- **mod_cache -** odkládání obsahu
  - využívá služeb dalších modulů (storage management modules)
- **mod_mem_chache -** odkládání do paměti
- **mod_disk_chache -** odkládání na disk

Odkládání - Direktivy

- **CacheEnable -** cache_type urlstring
  - **`CacheEnable cache_type url-str`**

- **CacheDefaultExpire -** nastavuje implicitní dobu v milisekundach, za kterou vyprší platnost obsahu není-li známá doba poslední modifikace
  - **`CacheDefaultExpire 3600`**

- **CacheMaxExpire -** Výchozí a maximální doba, za kterou vyprší platnost obsahu
  - maxilní doba platnosti obsahu v milisekundách
  - i když má dokument dobu platnosti delší, než je maximum specifikované touto direktivou, bude odstraněn
  - **`CacheMaxExpire 86400`**

- **CacheDisable -** zakáže odkládání pro specifikovanou url
  - **`CacheDisable wwww.example.com`**

- **CacheIgnoreCacheControl -** umožňuje ukládat do i dokumenty s nastavenými hlavičkami nocache
  - **`CacheIgnoreCacheControl On|Off`**

- **CacheLastModifiedFactor -** určuje faktor pro výpočet času, za jaký má být stránka považovaná za neplatnou
  - výpočet probíhá podle vzorce
    čas poslední modifikace * CacheLastModifiedFactor
  - výsledek se příčte k aktuálnímu času a podle toho se rozhodne jestli je stránka v cachi strále platná, nebo se má použít originální verze
  - **`CacheLastModifiedFactor 0.1`**

Directivy mod_mem_cache = Modul umožňující odkládání do paměti

MCacheSize = maximální velikost použité paměti v kB
MCacheMaxObjectCount = maximální počet objektů uložených v paměti
MCacheMinObjectSize = minimální velikost objektu ukládaného do cache
MCacheMaxObjectSize = maximální velikost objektu ukládaného do cache
MCacheRemovalAlgorithm = umožňuje nastavit algoritmus odstraňování dokumentů z paměti

- LRU (Least Recently Used)
- GDSF (GreadyDual-Size)

Odkládání do paměti - Příklad

```
<IfModule mod_mem_cache.c>
        CacheEnable mem /
        MCacheSize 4096
        MCacheMaxObjectCount 100
        MCacheMinObjectSize 1
        MCacheMaxObjectSize 2048
</IfModule>
```

Directivy mod_disk_cache = Modul umožňuje odkládání na disk.

CacheRoot = mapuje adresář kam se bude odkládat obsah
CacheMaxFileSize = maximální velikost dokumentu, který se bude ukládat do paměti
CacheDirLength

- délka názvu každého adresáře v CacheRoot
- apache používá pro vytváření jmen souborů hashovací funkci, délka názvu se nastaví touto direktivou, počet vnořených sdresářů pak pomocí direktivy CacheDirLevels, pokud se tedy žádá vícekrát o stejný dokument, apache získá cestu opětovným spočtením hashovací funkce

CacheDirLevels = nastavuje počet vnořených adresářů v CacheRoot

Odkládání na disk - Příklad

```
<IfModule mod_disk_cache.c>
        CacheRoot /var/cache/apache2/mod_disk_cache
        CacheEnable disk /
        CacheDirLevels 5
        CacheDirLength 3
</IfModule>
```