ahtppd v0.5b **Fejlesztői dokumentáció**

Ez a dokumentáció az ahtppd webszerver továbbfejlesztéséhez, a kód tanulmányozásához ad segítséget, hogy bemutatja a program felépítését. Alapvető Unix programozási ismeretek szükségesek a program megértéséhez.

Specifikáció:

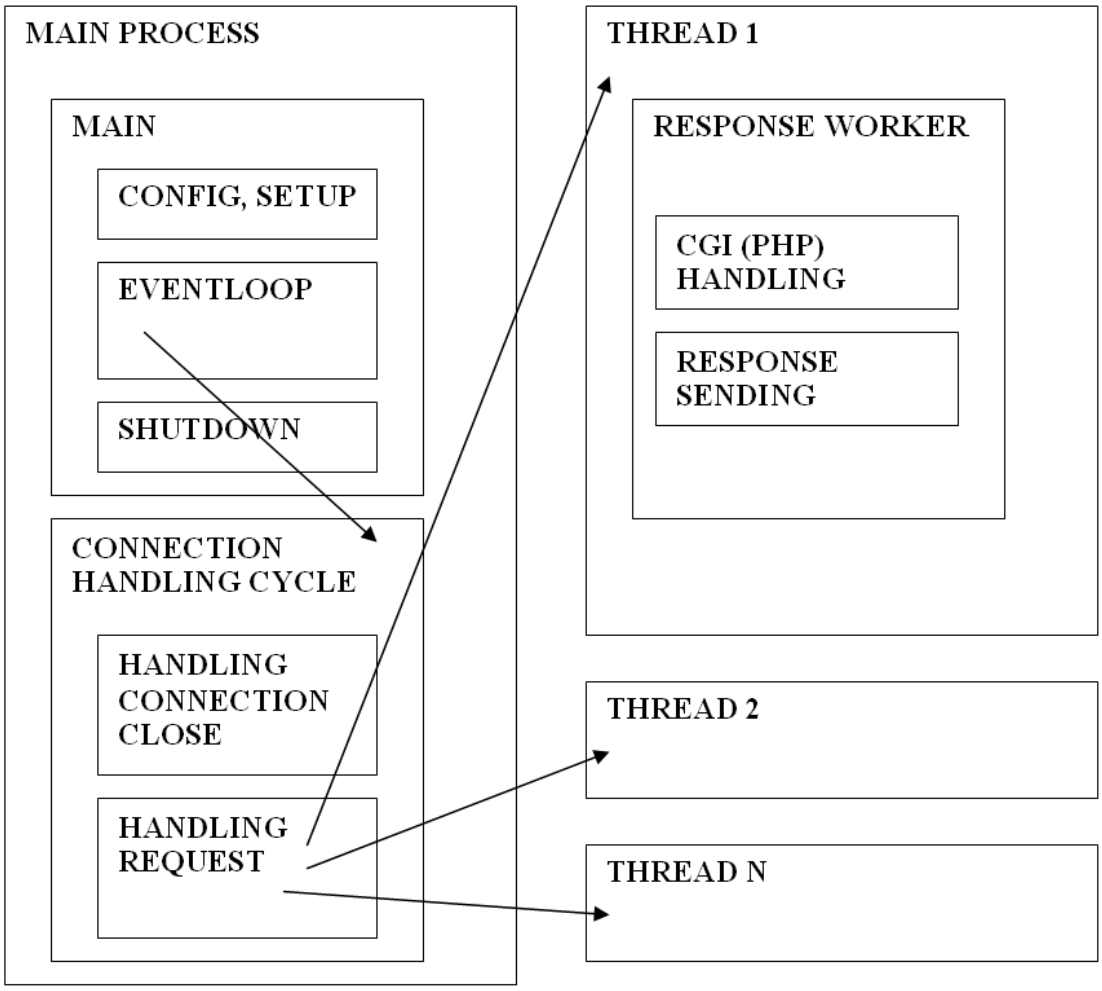
Webszerver a következő funkcionalitási körrel:

- *Háttérben daemonként futó alkalmazás*
- *Konfigurációs fájllal konfigurálható (port, user, htdocs)*
- *Alapvető biztonsági elemek: (effektív user váltás, URL parsing)*
- *Egyszerre több kliens párhuzamos kiszolgálása (szálkezeléssel)*
- *Beépített mini-statisztika (elérhető: http://address:port/_stat_)*

Fejlesztéshez szükséges információk:

A program nem igényel különleges fejlesztői környezetet, EditPlus szövegszerkesztővel készült, akármilyen kis funkcionalitású szerkesztővel módosítható. A forráskód C++-ban íródott, az egyszerű fordítás érdekében makefile is készült a projekthez.
Ha tehát módosítást eszközöltünk a forráskódban, azt egy egyszerű *make* paranccsal újrafordíthatjuk.
Fordítás előtt bizonyosodjunk meg, hogy a gépen megtalálható a g++ csomag, mely a fordításhoz szükséges, de sok Linux disztibúciónak nem alaptartozéka.

Felépítési diagram:



MAIN PROCESS
A program indítás után fork()-ol, és a parent exitel, a child folyamat tekinthető tehát a MAIN PROCESS-nek. A shellt visszakapjuk indítás után, a program a háttérben képes futni ezáltal.

MAIN
A program main() függvénye sorban az alábbi funckiókat valósítja meg:

CONFIG, SETUP
A konfigurációs fájl felolvasása, a beolvasott értékek eltárolása. Számlálók alapállapotba állítása, a megadott címre bindolás, és a listenelés megkezdése mind a SETUP hatásköre.
Ezen kívül az effektív uid és guid is megváltoztatásra kerül a konfigurációs fájlnak megfelelően.

EVENTLOOP
Ez a fő ciklusunk, melynek elején egy poll() hívás található, mely az aktív connectionöket monitorozza, és POLLIN eventeket vár minden csatornán.
Amint a poll() visszatért, megvizsgáljuk, hogy a listenelő socket-re érkezett-e POLLIN esemény, ebben az esetben acceptelünk, és létrehozzuk az új klienssel a kapcsolatot.
Ezek után meghívjuk a CONNECTION HANDLING CYCLE függvényt.

SHUTDOWN
Miután KILL signalt kaptunk, megszakítjuk az EVENTLOOP futtatását, és bezárjuk a függőben maradt kapcsolatokat, hogy aztán befejeződhessen a program futása.

CONNECTION HANDLING CYCLE
Ez a függvény a vector típusú struktúrában tárolt kapcsolatainkat görgeti végig, és megnézi, érkezett-e valamelyikre POLLHUP vagy POLLIN event.

HANDLE CONNECTION CLOSE
POLLHUP esetén lezárjuk a kapcsolatot, majd töröljük a nyilvántartó rendszerünkből.

HANDLING REQUEST
POLLIN esetén olvasunk a hálózatról, és a kapcsolatot leíró struktúrában egy bufferben gyűjtjük a fogadott adatot.
Amikor \r\n\r\n végződésű adatot olvastunk, megbizonyosodhatunk, hogy a teljes request megérkezett hozzánk, egy új threaded hozunk létre tehát, ami elvégzi a response elkészítését, és elküldését.

RESPONSE WORKER

Ez az a függvény, mely pointerként a megválaszolandó kérést küldő kapcsolat struktúráját kapja meg, és külön szálon indul el. Ebben dolgozzuk fel a requestet, parseoljuk az URL-t, megvizsgáljuk valós-e, majd *php* vagy *php5* kiterjesztés esetén a CGI HANDLING programkód segítségével, különben pedig a fájlrendszerből olvasással kinyerjük a response contentet.

CGI HANDLING

A CGI interfész referenciája alapján történik a request átadása a CGI scriptnek, ami esetünkben leimplementálva a PHP, de természetesen tetszőleges program lehet.

A CGI scripttel történő kommunikáció a következőképpen történik: pipe()-al két fd-t hozunk létre, majd fork()-olunk. A child process beköti az fd-ket az STDIN és STDOUT fd-k helyére, a parent process pedig az fd-kre történő olvasás ill. írás használatával a child process standard ki-és bemenetével kommunikálhat.

A child process az execle hívás segítségével cseréli le a process imaget a hívandó CGI scriptre, a függvény paramétereiben adhatjuk meg a szükséges környezeti változókat, amikben a request részletei találhatóak. (A CGI referencia szerint.)

A parent ezután streamként EOF-ig olvas a child kimenetéről, és eltárolja azt, ezáltal megszereztük a kimenetet, amit a kliensnek el kell juttatnunk.

RESPONSE SENDING

Miután vagy CGI-n kersztül, vagy a fájlrendszerről megszereztük a küldendő adatot, azt egyszerűen elküldjük a megfelelő response headerekkel (beleértve a Content-type kiterjesztés alapján történő beállítását) együtt a socketen.

STATISTICS

Amennyiben egy különleges request érkezik *(http://address:port/_stat_* címre), akkor a program belső statisztikáját adjuk vissza: küldött/fogadott bájtok száma, összes létrehozott kapcsolat. Ezek a program egyes részeiben mindig frissítésre kerülnek.