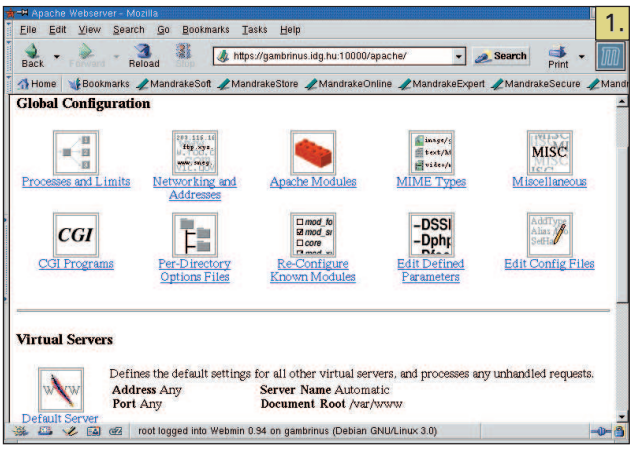
Webkiszolgálók harca

A cikkben a Linux alatt is futó webkiszolgálókat veszem górcső alá, ezek közül is azokat, amelyek szabadon elérhetők. Lássuk, miből gazdálkodhatunk, és melyik kiszolgálónak mi az erénye, illetve a hátránya.

Rögtön tisztázzuk, hogy bizonyos alkalmazások nemcsak webkiszolgálóként, hanem alkalmazáskiszolgálóként is működnek, esetleg saját beépített webfejlesztési eszközeik vannak. Az ismerkedést kezdjük a jó öreg Apache-csal, és tekintettel a népszerűségére, egy kicsit többet foglalkozunk vele.

Apache

Az Apache a világ legtöbbet használt webkiszolgálója, az összes kiszolgálóprogram körülbelül 55–60 százalékát ez a program teszi ki. Jelenleg a 2.0-s sorozat a legújabb, ami jelentős megújuláson ment keresztül az előző 1.3-as sorozathoz képest. Teherbírása és bővíthetősége legendás. Bővíthetőségét modulalapú felépítésének köszönheti, és él is ezzel a lehetőséggel. A program fordításakor mi dönthetjük el, hogy mi kerüljön magába a webkiszolgáló szerepét ellátó alkalmazásba, és mi modulba. Ha úgy kívánjuk, az összes modult belefordíthatjuk a programba, ennek azonban semmi értelme, mert csak lassabban fog elindulni, és egy csomó olyan modul lesz benne, ami felesleges. Milyen újdonságokkal rendelkezik, illetve hogyan is működik a 2.0-s Apache? Nos, az előző sorozathoz képest előrelépés, hogy már nem folyamat- (process), hanem szálalapú (thread) futtatásra is lehetőség nyílik. Miért előnyös ez? A weblapok lekérésekor legalább egy új folyamatnak el kellett indulnia, ilyenkor természetesen újra be kellett tölteni a démont a memóriába, következésképpen pazarolta az erőforrásokat, és lassabb volt az indulás. Ezzel ellentétben a szálak közös memóriaterületet használnak, és egy folyamaton belül több kiszolgálódémon is elindulhat. Ekkor tehát az Apache kevesebb memóriát használ, és gyorsabban szolgálja ki a kéréseket. A szálak programozásánál ugyanakkor körültekintően kell eljárnunk, mivel súlyos gondok adódhatnak, ha elrontja az ember, és ezt nem mindegyik felület támogatja megfelelő módon. Érdemes hát ezt a módszert használni a webkiszolgálónál. A fejlesztők viszont a választás lehetőségét meghagyják a rendszergazdáknak, hogy a megbízhatóságot és a kipróbált módszert választják-e – hiszen az Apache ebben is nagyon jól teljesített –, esetleg megkockáztatják a még kevesebb ideje használatos kódot, és valóban nagyobb teljesítményt érnek el. Hogyan lehetséges ez? Az 1.3-as ág fejlesztésénél a fejlesztők a POSIX-szabványokon alapuló megoldást használták, ahol pedig ezt megvalósító réteg nem állt rendelkezésre az operációs rendszerben, ott egy POSIX-emulációs réteget húztak fel a rendszerre. Ez néhol kevésbé megbízhatóvá és lassabbá tette a rendszert. A 2.0-s ág a hálózati kapcsolatok kezeléséhez és a fájlrendszer eléréséhez már a futtató operációs rendszer által is használt natív rendszerhívásokat használ. A natív rendszerhívások lehetőségét a fent említett MPM-ek (Multi-processing Modules) és az APR (Apache Portable Runtime) biztosítja. Az MPM-ek teszik lehetővé azt is, hogy a rendszergazda kiválaszthassa, milyen módon használja a kiszolgálót – a régi, folyamatalapú, vagy az új, kevert, több folyamatot és több

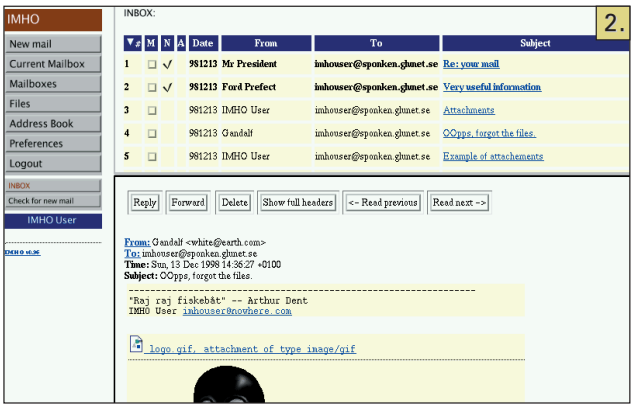


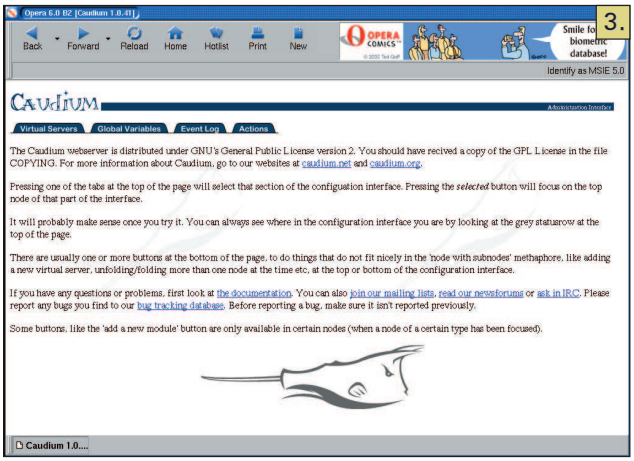
1.

szálat használó, vagy egy harmadik, úgynevezett Perchild modult használva. A Worker MPM az a modul, ami a kevert megoldást alkalmazza. Elindul egy folyamat, és az ügyfelek egy bizonyos kérésszámig ezen belül szolgálják ki a rendszer. Ha a kérések túllépik ezt a számot, egy újabb folyamat indul, és kéréseket azon belül lehet ismét a meghatározott számig kiszolgálni. Mind a folyamatok, mind a folyamaton belüli szálak legnagyobb mérete megadható. A Preforking MPM a régi Apache-megközelítést használja, a már fent leírt egy démon folyamatmegközelítést. Itt a kiszolgálható ügyfelek legnagyobb számát az engedélyezett folyamatok száma határozza meg. Az erőforrások tekintetében a drágább indulásért és a nagyobb igényért cserébe nagyobb megbízhatóságot kapunk. A Perchild MPM pedig egy olyan új, szintén kevert megoldást alkalmazó rendszer, amelynél folyamatonként beállítható, hogy milyen felhasználói azonosító alatt működjön. Ezzel a felhasználókat teljesen el tudjuk határolni egymástól. Ez a modul nem minden felületen érhető el, és a fejlesztők is felhívják rá a figyelmet, hogy bár már üzembiztos, még fejlesztés alatt álló modulról van szó.

Az Apache felügyeleti eszközökkel is jól el van látva, amelyek közül a Webmin – mint nyílt forrású megoldás – külön kiemelhető (*1. kép*).

Az Apache felhasználási szerződése mind az üzleti felhasználást, mind az olyan megoldások fejlesztését lehetővé teszi, amelyek már nem lesznek nyílt forrásúak. A világon a legismertebb cég, amely Apache-csal foglalkozik és kereskedelmi tevékenységet folytat, a Covalent. Nemrég olyan megállapodást kötöttek a Microsofttal, amelynek értelmében az ASP.Net-alapú alkalmazások Apache alatt is futtathatók lesznek. A kiadott leírás csak a Windows operációs rendszerről szólt, de ne feledjük, hogy egy modullal már most is lehetőség nyílik ***.asp*** kiterjesztésű weblapokat futtatni Apache alatt, bár ez tényleg nem rendelkezik a .NET lehetőségeivel. Ez az Apache legnagyobb erőssége:





a modulokon keresztül nemcsak az igényeinknek megfelelően finomhangolható, de gyakorlatilag nincs olyan parancsnyelv, amit ne támogatna. Az Apache projekt azonban ennél sokkal szélesebb körű: szerepel benne Java-alapú webkiszolgáló, illetve alkalmazáskiszolgáló, XML-RPC- és SOAP-megvalósítás is – ezekről egy másik számban bővebben írok majd.

Boa

Ha valakinek egyszerű, gyors, CGI-ket is futtatni tudó webkiszolgálóra lenne szüksége, a Boa a számára megfelelő kiszolgáló. Futása közben csak akkor indít magából újabb példányokat, ha CGI-t kell futtatnia, máskülönben a már futó bináris belül oldja meg a kért oldalak kiszolgálását. A már futó folyamat önmagán belül többszörözi a HTTP-csatornákat. Sebessége is lenyűgöző: a tesztek alapján egy 300 MHz-es gépen is több ezer kapcsolatot képes kezelni! Fő célja a sebesség, és nem utolsósorban a biztonság. Fejlesztése már 1991-ben elkezdődött, ezenkívül beállítófájlja nagyon hasonlít az Apache-éhoz. Ha ennyi mindent tud egy kis bináris, akkor miért nem ismert és népszerű? Például azért, mert nem támogatja a kiszolgálóoldali parancsnyelveket, például a PHP-t. A legritkább esetben gyorsabb vagy könnyebb C-alapú CGI-ben írni meg egy dinamikus weboldalt, mint PHP-ban, nem meglepő tehát, hogy a rendszergazdák legtöbbször más megoldásokhoz fordulnak.

Roxen

A Roxen Webchallenger a második legnépszerűbb nyílt forrású program a világon. Megbízhatóságáról sok mindent elárul, hogy a ➲ http://www.bsd.hu is ezt használja (az élet furcsa fintora: a Roxen GPL felhasználási szerződésű, a fejlesztők ugyanis nem szeretik a BSD felhasználási szerződést). Különlegessége, hogy nagy részét a Pike nevű parancsnyelvben írták – erre sok más tekintetben is támaszkodik –, és folyamatok helyett a kezdetektől fogva szálakat használt. Ami számos rendszergazda szemében vonzóvá teheti, az a weben keresztüli felügyelhetőség. Ez egyben legnagyobb hátránya is: amikor először nézünk bele a beállítófájlba, valószínűleg még a lélegzetünk is eláll, és nem kis időt vesz igénybe, mire elsajátítjuk a felület kezelését. Az XML ugyanis nagyon szép lehetőségeket kínál az adatok megjelenítésére, de ha egy beállítófájlt ebben tárolnak, és azt nekünk kézzel kell szerkeszteni, nos… Én inkább maradok az Apache szemléleténél – egyszerű beállítófájl, ha valaki nem ismeri, grafikus beállítóprogramot kap. Nos, a Roxennél meg sem próbálkoznék kézzel beállítani a kiszolgálót. A Pike nyelven keresztül kitűnően és gyorsan lehet olyan weboldalakat összeütni, amelyek ***.rxml*** kiterjesztéssel rendelkeznek. Figyelem! Csak a Roxen Webchallenger tartozik GPL felhasználási szerződés alá, és a Roxen honlapján (➲ http://www.roxen.com) szinte csak a Roxen CMS-ről esik szó – ez teljes fejlesztési és vezérlőfelületet ad az oldalhoz, de kereskedelmi termék. A letöltéseket a Download menüpont alól tehetjük meg, a legutolsó változat a 2.2-es. Érdekessége, hogy egy hozzá fejlesztett webmail tartozik mellé, ami a legjobbak egyike, az IMHO (*2. kép*).

Caudium

A Caudium a Roxen 1.3-as sorozatából indult ki mint külön fejlesztési ág, kihasználva a GPL adta lehetőségeket. A történet valamikor még az 1.3-as Roxen idejében kezdődött, amikor is a fejlesztők úgy gondolták, hogy nem raknak vissza minden foltot (patch). Elég sok külső és néhány belső – értsd: céges – fejlesztő úgy gondolta, hogy külön utakra lépnek, és nem követik a cég elképzeléseit és jövőbeni terveit. Így született meg a Caudium. Sok mindenben hasonlít a Roxenre, abban is, hogy saját webmail programmal rendelkezik, amelynek neve CAMAS. Ez is külön fejlesztési ág, az IMHO-ból jött létre a Caufiumhoz igazítva. További megegyező tulajdonságuk, hogy csak webfelületen keresztül érdemes beállítani őket (*3. kép*). Ha telepítjük a támogatást, PHP-oldalaink is működni fognak alatta.

Zope

A Linuxvilágban már sorozat is megjelent róla, így sokaknak ismerős lehet. Beépített webkiszolgálóval rendelkezik, a fő hangsúly azonban mégiscsak a tartalomszolgáltatáson van, amit rendkívül magas szintű nyelven lát el. Már készül hozzá a sablonértelmező, amelynek segítségével a kiszolgált oldalak küllemének cseréje könnyedén megoldhatóvá válik, és a megjelenítés sokkal jobban elválasztható lesz magától az alkalmazástól. Ehhez részletesebben lapozzuk fel a Linuxvilág régebbi számait (15–19. szám).

Deim Ágoston (ago@lsc.hu)
Kedveli a sört, szereti a futást és imádja Szabó Lőrinc verseit. Nem hisz vakon egyik rendszerben sem. Vonzódik a BSD-hez is. Tagja az LME-nek és a MBE-nek. Mottója: a gép nem lehet fontosabb az embernél.

➲ httpd.apache.org; ➲ http://www.boa.org
➲ http://www.roxen.com; ➲ http://www.caudium.org
➲ http://www.lysator.liu.se/~stewa/IMHO/