Ördögűzés

Sorozatunk előző részében betekintést nyerhettünk a Linux alatti fájlkezelés rejtelmeibe. Most a háttérben futó programok kezelési lehetőségeivel ismerkedünk meg.

A Linux többfeladatos operációs rendszer. Ez mindössze annyit jelent, hogy egyszerre több programot, alkalmazást képes futtatni. Valójában szó sincs valódi párhuzamos futtatásról, egyszerre csak egy program fut, de egy ütemező algoritmus segítségével a rendszer – bizonyos idő elteltével – mindig egy másik, szintén futásra kijelölt alkalmazásnak adja át a futási lehetőséget.

Ez az ütemező eljárás meglehetősen bonyolult, működésének ismertetése meg is haladná e cikk kereteit. Ezért legyen elég annyi, hogy ezt a feladatot is a rendszermag látja el. Természetesen, mint minden valamire való többfeladatos rendszerben, a Linux sem hagyja szabadon garázdálkodni a futó programokat. Az alkalmazás futásának teljes ideje alatt komoly megszorítások vonatkoznak rá, és ha csak az egyiket is áthágja, egyből kinn találhatja magát a rendszer memóriájából.

Írásomban a párhuzamosan futtatott alkalmazások irányításával kezelésével, foglalkozunk, végezetül pedig pár szót ejtünk egy a Unix világában kulcsfontosságú fogalomról: a démonokról.

A Linux többfeladatos, úgynevezett multitaszkos operációs rendszer. Ez mindössze annyit jelent, hogy egyszerre több programot és alkalmazást képes futtatni. Valójában mindig csak egyetlen program fut, de a rendszer egy ütemező algoritmus segítségével – bizonyos idő elteltével – mindig egy másik, szintén futásra kijelölt alkalmazásnak adja át a futási lehetőséget.

Azt a legkisebb egységet, amely egy ilyen rendszerben párhuzamos feldolgozásra kerülhet, folyamatnak (process), illetve feladatnak (task) hívjuk. Egyszerűbben: ha bármilyen programot elindítunk, a rendszer létrehoz egy új folyamatot, amelyben az alkalmazás a többi, már meglévő folyamattal együtt fut majd.

Az éppen futó folyamatok listáját a `ps` parancs segítségével kaphatjuk meg. Ha ezt az utasítást kapcsolók nélkül adjuk ki, csak az általunk elindított programok kerülnek listázásra. Ha kicsit bővebb adatokra vágyunk, például az összes folyamatot meghatározott adataival együtt szeretnénk látni – például ki és melyik terminálon futtatja és a többi –, a `ps aux` parancsot használjuk. Az így kapott, általában többképernyős listában az utolsó oszlop tartalmazza a futó programok neveit. Az első oszlop azt a felhasználót jelöli, aki az alkalmazást elindította. Számunkra a második oszlop lesz még igazán fontos, ugyanis ez tartalmazza az úgynevezett PID-et (Process ID), ami nem más, mint egy azonosítókulcs, amely az adott folyamatot azonosítja. Minden folyamatnak egyedi azonosítója van. Ha egy folyamat egy kódszámot kapott, azt a kódszámot innentől kezdve egyik folyamat sem kaphatja meg (egészen a rendszer újraindításáig), még akkor sem, ha az adott program esetleg régen befejezte a futását.

Sajnos, kevés a tökéletes program, ezért előbb-utóbb számíthatunk rá, hogy valamelyik le fog fagyni. Az ilyen helyzetek eléggé kellemetlenek az olyan „egyfeladatos” rendszerekben, mint amilyen például az MS-DOS: ebben az esetben kizárólag a számítógép újraindításával kezelhetjük a gondot a leghatékonyabban.

Egy valódi többfeladatos rendszernél azonban nem kell ilyen mélyreható eszközökhöz folyamodnunk, mivel egy folyamat nem veszélyeztetheti a többi futó alkalmazást, illetve magát az operációs rendszert. Ha valamelyik programunk netán lefagy vagy rendellenesen működik, egyszerűen csak meg kell kérnünk a rendszermagot, hogy küldjön neki egy üzenetet (signal), amelyben a futás befejezésére szólítja fel. Ezt a `kill` utasítással tehetjük meg. Kapcsolóként a leállítandó folyamat PID-jét kell megadnunk.

Egyes esetekben akkora gubanc is keletkezhet, hogy az adott alkalmazás nem képes az általunk küldött utasítást „kezelni”. Ilyenkor erőszakosabb fellépésre van szükség: a `kill` utasítás után írjunk egy `-9` kapcsolót, például: `kill -9 192`. A `-9` hatására a rendszermag se szó, se beszéd a futó alkalmazást eltávolítja a memóriából. Ez ellen egy felhasználói program sem képes védekezni, tehát jaj annak az alkalmazásnak, amelyik erre a sorsra jut! A `kill` utasítás eredeti feladata egyébként nem az, hogy az éppen futó programjainkat megsemmisítsük a memóriából, hanem az, hogy segítségével különböző rendszerüzeneteket (signal) küldhessünk a folyamatoknak. A `kill` parancs hivatalos írásmódja: `kill -`*`jel`* `PID`.

A `-signal`-hoz a jel nevét (például `KILL`, `TERM` stb.) vagy számát írhatjuk. Amennyiben nem adunk meg jelet, akkor a 15-ös számú `TERM` jel kerül elküldésre. Ez felszólítja az alkalmazást, hogy azonnal zárja be az összes megnyitott állományt és fejezze be a futását.

A `-9` a `KILL` jel küldését teszi lehetővé. Ezt a lehetőséget csakugyan a legutolsó esetben használjuk, ugyanis egészségtelen, ha a nyitott állományokat nem zárják le megfelelően.

A `kill` mellett meg kell említenünk a `killall` utasítást is.

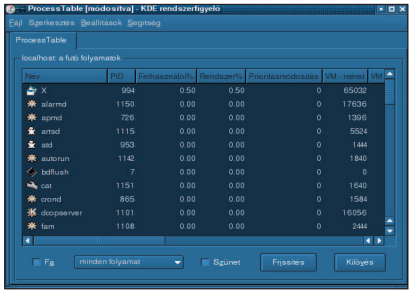
Itt kapcsolóként nem a PID-et, hanem az alkalmazás nevét szükséges megadnunk. Eredményeként az összes ilyen nevű futó program befejeződik. Természetesen, amennyiben ezt az utasítást nem rendszergazdaként adtuk ki, csak az általunk elindított alkalmazásokra lesz hatással.

A folyamatok kezelésére másfajta és egyszerűbb módszer is létezik. Ilyen például a KDE-hez tartozó ***KDE rendszerfigyelő***, de kötelességünk megemlíteni a konzolos `top` nevű programot is. Mélyedjünk el egy kicsit az éppen futó folyamatok listájában! Már a lista méretéből is arra következtethetünk, hogy a háttérben sokkal több minden fut, mint ahány alkalmazást indítottunk.

Észrevehetjük, hogy viszonylag sok a rendszergazda által indított, általában `d` betűre végződő program, például a `kerneld`, `syslogd`, `inetd`, `nfsd` stb. Ezek nem mások, mint az összes Unix-alapú rendszer alappilléreit képző démonok (daemon). A démon görög eredetű szó, eredeti jelentése „közvetítő”. Az elnevezés olyan programokat takar, amelyek a rendszer indulásától egészen az újraindításig futnak, ám idejük nagy részét általában semmittevéssel töltik, csak egy meghatározott esemény bekövetkeztekor lendülnek munkába.

Minden démon kizárólag egy egyéni fe-

ladatot lát el. Például a `syslogd` feladata a rendszerben történő dolgok naplózása. Ha nincs semmi naplóznivaló, „mély álomba szenderül”, de amint újabb feljegyezni való érkezik, azonnal munkába áll. Kiegészítésként megjegyezzük, hogy a



Linuxban (és számos más, többfeladatos operációs rendszerben) a legalapvetőbb műveletek elvégzése is ezen az elven alapszik. Például a lemezre való írásért felelős eljárások is folyamatosan külön feladatként (task) futnak, de érdemi tevékenységet csak akkor végeznek, ha valamit írni kell a merevlemezre. Ezek a feladatok mélyen a rendszermagban, a felhasználói folyamatok alatti szinteken helyezkednek el, és egy felhasználó sem bolygathatja meg őket. A démonok azonban a „felhasználói rétegben” futnak, tehát a rendszergazda bármikor leállíthatja, illetve újra elindíthatja őket.
A Unix-rendszerekben szinte az összes feladatot a démonok látják el. Például a különböző hálózati kiszolgálók (web, FTP stb.) is démonként futnak.
A démonok lelkivilágának jobb megértéséhez a dolgot egy kézzelfogható példával szemléltetem, mégpedig a `cron` démonnal (senkit ne tévesszen meg, hogy a név végén nincs `d` betű!). E démon segítségével a felhasználók által előre meghatározott feladatok elvégzését tehetjük önműködővé. Például beállíthatjuk, hogy a program minden hajnalban 5 órakor hívja meg a `halt` parancsot, amely a rendszer leállását eredményezi (éjszakai letöltögetőknek roppant hasznos szolgáltatás!).
A `cron` démon tehát a Linux elindítását követően betöltődik, ettől kezdve minden percben megnézi az adatbázisában, hogy akad-e valamilyen tennivalója. Ha van, elvégzi a kijelölt feladatot, ha nincs, a következő percig „hibernálja” magát.
Most nézzük meg, hogyan is működik ez a gyakorlatban! Ahhoz, hogy egy felhasználó kiaknázhassa a `cron` démont által nyújtott szolgáltatásokat, azonosítójának szerepelnie kell a */etc/cron.allow* állományban. Ezt a fájlt csak a rendszergazda írhatja! A `cron` démon feladatait az úgynevezett ***crontab*** állomány tárolja. Minden felhasználóhoz külön crontab-fájl tartozik, amely a */var/spool/cron* könyvtár alatt található. A crontab felépítése nagyon egyszerű: minden bejegyzést külön sorba kell írnunk, és egy bejegyzés hat oszlopból áll. Az első öt a feladat elvégzésére kijelölt időpontot jelzi (perc, óra, nap, hónap, a hét napja), az utolsó pedig magát a feladatot. Ha valamelyik időpont helyére *-ot (csillagot) írunk, az minden lehetséges értéket helyettesíteni fog. A „hét napja” oszlopnál a nulla jelenti a vasárnapot, egy a hétfőt és így tovább. Eredeti példánkhoz visszakanyarodva az ötórai gépleállítás crontabja valahogy így nézne ki:
`5 0 * * * /sbin/halt`.
Biztonsági okokból crontab-állományunkat nem szerkeszthetjük csak úgy kézzel. Erre a feladatra a `crontab` parancsot kell használnunk. Ha új bejegyzést szeretnénk létrehozni, hozzunk létre egy új állományt a saját könyvtárunkban, amelybe az új feladatot az előbb ismertetett formátumban írjuk be. Ezután adjuk ki a `crontab fÆjlnØv` utasítást, és máris bekerült a cron adatbázisába. Bejegyzést törölni a `-r` kapcsolóval tudunk, egy már meglévő átszerkesztéséhez pedig a `-e`-t kell használnunk.
A démonok tehát a rendszer indításától fogva betelepszenek a memóriába. De vajon a rendszer melyik része tölti be őket? Ezt a feladatot az úgynevezett indító parancsfájlok végzik el, amelyek minden rendszerinduláskor lefutnak (előző számunkban már említettük a héjakat, amelyek hasonlóak a DOS-os kötegelt (batch) állományokhoz, csak sokkal fejlettebbek). Ezeknek a héjaknak előre „megmondják”, hogy melyik démonokat kell betölteniük.
Az indító parancsfájlok valójában egyetlen démont sem tölt be, hanem csak a saját indítóhéjaikat hívja meg. Ezek az indítóhéjak általában a */etc/rc.d/init.d* könyvtár alatt találhatók.
Hogy a rendszer indulásakor melyik indítóhéjat használjuk, az úgynevezett futási szintek (runlevels) segítségével adhatjuk meg. A Linux-rendszerekben hét futási szint van megadva (0–6-ig). Minden egyes futási szintben leolvasható, hogy milyen démonokat indítsunk el, illetve állítsunk le.
A nullás és hatos a rendszerleállításra szolgál, tehát értelemszerűen az összes démon leállítását tartalmazza. Az 1-es általában csak a legszükségesebb összetevőket tartalmazza, a 2-es és a 3-as a standard beállítást. A 4-esnek és az 5-ösnek tulajdonképpen nincs meghatározott szerepe, esetükben általában a grafikus bejelentkezők is elindulnak.
Ezt a rendszert, ha gondoljuk, kedvünkre megváltoztathatjuk, de nincs sok értelme.
Hogy a rendszer melyik futási szinttel induljon, a */etc/inittab “id:* kezdetű sora írja le. Ha úgy látjuk jónak, változtassuk meg nyugodtan, csak arra vigyázunk, nehogy 0-ra vagy 6-ra állítsuk, mert kellemetlen meglepetésben lehet részünk (a rendszer elindulása után egyből a leállítási szintbe lép át).
Miután rendszerünk teljesen betöltődött, a különböző futási szintek között kedvünkre váltogathatunk, erre használható az `init <œj futÆsi szint>` utasítás. Ha az új futási szinthez 6-ot írunk, a kiadott parancs a `halt` utasítással lesz egyenértékű.
Végezetül nézzük meg, hogyan is szerkeszthetjük a futási szinteket! A démonok inicializációs scriptjeit a */etc/rc.d/init.d* könyvtár alatt lelhetjük meg. Minden futási szinthez külön könyvtár tartozik, amelynek elnevezése: */etc/rc.d/rc.x* (az x a futási szint számát jelöli). Kukkantsunk be az egyik ilyen könyvtárba! Mint láthatjuk, a démonok incializációs scriptjeire mutató közvetett hivatkozásokkal van tele. A hivatkozás nevének utolsó része megegyezik a démonindító parancsfájlok nevével (ez általában a démon nevével is azonos). Az első karakter csak `S` vagy `K` lehet. Az első esetben a héj a `start` értékkel hívható meg, tehát az adott démon elindul. Az utóbbi esetben pedig a `stop` érték lesz átadva, tehát a démon befejezi a futását.
Ha nem akarjuk, hogy egy démon például a 3-as futási szinten betöltődjön, egyszerűen távolítsuk el a közvetett hivatkozást. Az ***init.d*** könyvtár alatti héjakat kézzel is meghívhatjuk, így mi magunk a futási szintektől függetlenül is démonokat tölthetünk be, illetve távolíthatunk el a memóriából.
A következő részben már teljesen más vizekre evezünk: a Linux alatti fájlrendszerkezelést vesszük nagyító alá, tehát mindenféle lemezterületet és egyéb cserélhető lemezegységet fogunk fájlrendszerünkhöz be- és kifűzni.

Garzó András
(garzoand@interware.hu)
Körülbelül három éve foglalkozik Linux- és más Unix-rendszerekkel. Legjobban az operációs rendszerek lelkivilága érdekli, de nyitott egyéniség. Kedvenc étele a palacsinta, és van egy Richard nevű macskája. Minden észrevételt, megjegyzést, levelet szívesen fogad.