A Perl és a hálózat

Mint más nyelvekben, a Perlben is számos lehetőség nyílik a folyamatok közötti kapcsolattartás (IPC) megvalósítására.

Lássuk, miként beszélgethet egymással két különböző számítógépen csücsülő folyamat! Megoldásunkhoz használjuk az egyik legelterjedtebb programozási nyelvet, a Perlt.

Az IPC (interprocess communication)

Ugyanazon a számítógépen futó két folyamat rengeteg módon cserélhet adatot. Használhatnak szignálokat, fifókat, csővezetéket, és még számtalan más eszköz áll a programozó rendelkezésére (lásd man 1 perlipc). Minden megoldásnak megvan a maga előnye és hátránya. Az egyik természetesen hatékonyabban láthat el egyes feladatokat a másiknál, és bizonyos helyzetekben nem is mind használható. Létezik viszont egy olyan kialakítás, amelyet én még minden gond esetén használni tudtam, továbbá egyszerű és megbízható: ez pedig a foglalatok (socketek) használata. Foglalatokat nemcsak a helyi gépen futó folyamatok közötti adatcserére, hanem távoli számítógépek közötti kapcsolattartásra is fel lehet használni. Nekem azért tetszett meg nagyon, mert általa a Perlben objektumközpontú felületet kapunk, és nem (feltétlenül) kell az alacsony szintű függvényekkel bajlódnunk.

Foglalatok – miképpen épül föl egy kapcsolat?

Amikor két folyamat beszélgetni akar, először el kell dönteniük, hogy milyen tartományban találhatók. Ha a Unix-tartományt választjuk, csak a helyi számítógépen futó folyamatok kapcsolódhatnak egymáshoz. Az INET-tartomány használata teszi lehetővé a távoli gépek közötti adatcserét. Ezután mindkét folyamatnak létre kell hoznia egy-egy foglalatot. Ennek a foglalatnak be kell állítani a tulajdonságait, és ezután kaput kell neki foglalni. Az ügyfélnél ez a lépés elhagyható, a rendszermag megteszi helyettünk. A kiszolgálónál azért kötelező, mert tudnunk kell a folyamat kapuszámát, máskülönben nem tudjuk elérni. Ezt követően az ügyfél a megadott kapun már kapcsolódhat is a kiszolgálóhoz, és megkezdődhet a „beszélgetés”. Ez valóban beszélgetés, ugyanis ezt követően az ügyfél foglalata állományleíróként viselkedik, és ugyanúgy lehet bele írni és olvasni, mint egy szabályos fájlba. A beszélgetés továbbá kétirányú: a kiszolgáló ír valamit, amit az ügyfél olvas; ezután az ügyfél ír és a kiszolgáló olvas stb. Arra azonban vigyázni kell, hogy ez a foglalat egy olyan „állomány” leírója, amelynek nincsen vége. Pontosabban akkor van vége, amikor a kapcsolat lezárult. Ezért a kétirányú kapcsolattartáshoz ki kell találni egy „vége” jelet. Hasonlóan az indiánfőnök és a kisindán beszélgetéséhez, itt is, ha az egyik folyamat befejezte a mondanivalóját, uff-ot mond. Ebből tudja a másik, hogy ő következik, olvasás helyett írni kezd a foglalatra.

Perl – ahol mindez egyszerűbb

A Perl – mint fentebb már említettem – objektumközpontú felülettel segíti a programozók munkáját. Természetesen itt is léteznek azok az alacsonyszintű függvények, mint C-ben, tehát ha valaki ragaszkodik hozzá, használhatja őket.
A `Socket` modul alkalmazásával viszont mindez sokkal egyszerűbb. Amennyiben még esetleg nem foglalkoztál objektumközpontú programozással, akkor se ijedj meg. Az objektumközpontú programozás filozófiája csak egy, a strukturált nyelveknél az emberi gondolkodáshoz közelebb álló elképzelés. Eszerint először általánosítasz (osztályok), aztán leírod, hogy az eset miben különbözik az átlagostól (objektumok). A Perl-kézikönyvoldalak nagyon jó bevezetőt tartalmaznak az objektumközpontú programozáshoz. Ha valamennyire ismered már a Perlt, de nem tudod, mi az objektumközpontú programozás, feltétlenül olvasd el a perlboot oldalt (1. fejezet).

És ahogyan mindez a gyakorlatban fest

Ha Unix-tartományban akarsz dolgozni, a `Socket::UNIX`-osztályt kell használnod (man 3perl `IO::Socket::UNIX`), INET-tartomány esetén pedig – ki gondolta volna – a `Socket::INET`-osztályt (man 3perl `IO::Socket::INET`). Az osztályból készítesz egy objektumot és már kész is a foglalat. Tudnod kell, hogy Perlben nincs külön kinevezett objektum-függvénylétrehozó (constructor). Egy osztály bármely eljárása lehet függvénylétrehozó, ha meghívja a `bless()` függvényt, csupán megfelelően le kell írnia az osztályt. Természetesen mindenki az egységes felület kialakítására törekszik, ezért a függvénylétrehozót többnyire `new()`-nak hívják. Ebben az esetben nem kell az `IO::Socket::INET->new()` formához ragaszkodnod, használhatod a jobban olvasható `new IO::Socket::INET` kifejezést. A függvény egy asszociatív tömböt vár. Mivel ennek elemeit a programban sehol máshol nem használjuk, nyugodtan lehet anonim. A szükséges kulcsok egy kiszolgáló esetén a következők lehetnek:

LocalPort

Annak a kapunak a száma, amelyen a szolgáltatást indítani akarod. Ne felejtsd el, hogy az 1024 alatti kapuk használatához rendszergazdai jogok szükségesek!

Proto

A használni kívánt protokoll. Ez TCP vagy UDP lehet attól függően, hogy kapcsolat- vagy datagramközpontú protokollal akarsz dolgozni. Ha nem tudod, hogy mik ezek, használj TCP-t. A TCP teszi lehetővé a programozók számára, hogy minden csomag megérkezzen, és abban a sorrendben tegyék, ahogy elküldte őket.

Listen

Ez a várakozási sor hossza. Amíg egy ügyféllel foglalkozol, nem tudsz többet kiszolgálni (kivéve, ha a kiszolgálót többszálasra készíted). A többiek ezalatt a várakozási sorba kerülnek, és a kapcsolatra várnak. Ha a várakozási sor is betelik, a további