Solaris 10 (2. rész)
Rendszereszközök dióhéjban

A tapasztalt linuxos rendszergazda el tud ugyan boldogulni alapvető esetekben, ha egy Solaris konzoljához viszi a sors szeszélye vagy a saját kíváncsisága, de igazándiból a rendszeradminisztrációhoz használatos eszközök kezelésében több a különbség, mint a hasonlóság. Így aztán annyi felfedezni valót tartogat a rendszerek őrének a Sun operációs rendszere, hogy nem is pazarlom tovább a karaktereket, íme a felhozatal!

Vezérlőpult, nem amatőröknek

A ***Sun Management Console*** (röviden ***smc***) használatát fogjuk most áttekinteni. A vezérlőpult szóhoz talán olyan pejoratív mellékzöngék társultak az idők során, hogy a hozzáértést pótolja egy csillogó-villogó felület.

Bár a hasonló szemlélet miatt egy alcím alá vettem ezt a két hasonló nevű ketyerét, egészen más célt szolgálnak. Az ***smc***-vel elemi rendszeradminisztrátori teendőket tudunk gyorsan és kényelmesen elvégezni, a ***Java Web Console*** a ***Solaris 10***-re telepített különféle alkalmazások, szolgáltatások (már amelyik támogatja ezt) finomhangolását végzi el.

Lépjünk be bármilyen felhasználóval a grafikus felületre, majd indítsuk el az `/usr/sbin/smc` parancsot.

Ez a program ***Java*** alapú, ezért a ***Gnome*** rendszer többi összetevőjéhez mérten kissé lomha. Ezt viszont érdemes elnézni neki, mert az a kis veszteség, ami a program sebességében jelentkezik, többszörösen megtérül a funkcionalitás miatt. Egy rendszert egy úgynevezett ***toolbox*** testesít meg. A ***toolboxok*** lehetnek helyben, vagy más szervereken. Így kényelmesen, egy képernyőről navigálhatunk több szerver beállításai között.

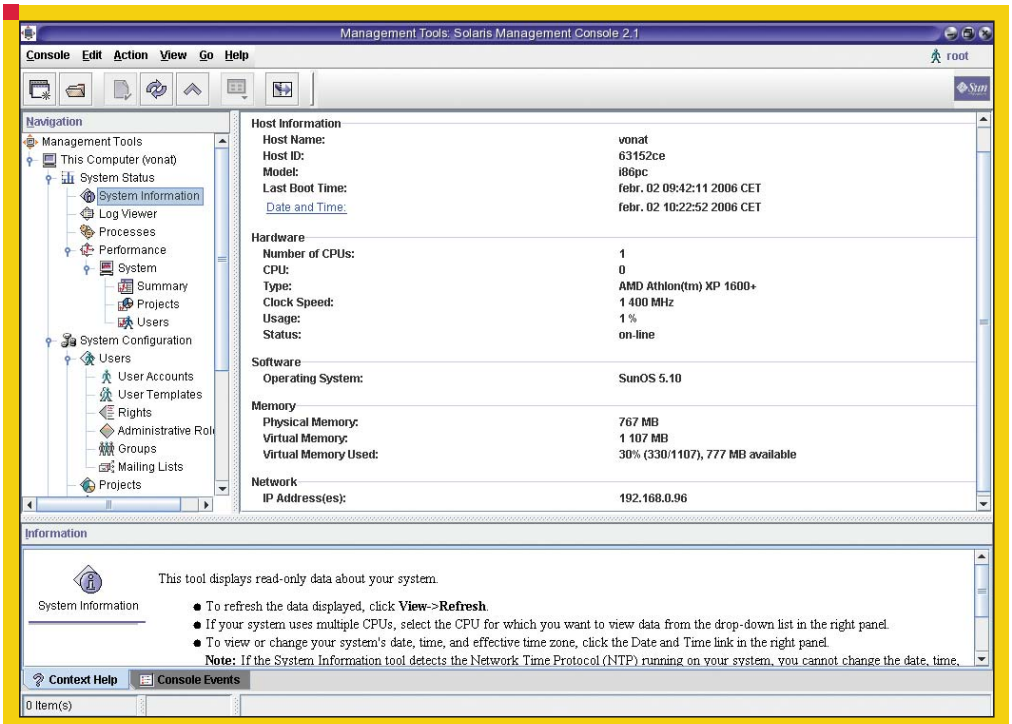
Ahhoz, hogy hozzáférést kaphassunk, be kell jelentkeznünk, hiszen az smc programot bármelyik felhasználó elindíthatja. Mik is azok a beállítások, amiket így elérhetünk? Egyfelől a ***System Status*** alatt az életjeleket figyelhetjük, kicsit kifejezőbben szólva rajta tarthatjuk az ujjunkat a rendszerek ütőerén. A top parancshoz hasonló felületet és a naplókat láthatjuk itt, valamint kapunk egy rövid összefoglalót a rendszer hardverkörnyezetéről a ***System Information*** alatt.

A ***Processes***-t érdemes jól tanulmányozni, a folyamatokat nemcsak listába rendezhetjük itt, hanem egészen mélyre is áshatunk: a környezeti változókat, a fájlleírókat, a használt könyvtárakat (library) és még sok minden mást láthatunk folyamataink magánéletéből.

A ***System Configuration*** választva már mi magunk is aktívan bekapcsolódhatunk: a teljes felhasználókezelést elvégezhetjük, felhasználók, csoportok, jogok (***rights***) és szabályok (***roles***) fölött rendelkezünk itt. A ***Services*** csak az ütemezett feladatokat rejti, pár kattintással összerakhatunk bonyolult `cron` feladatokat. A ***Storage*** az ***NFS***-sel kapcsolatos megosztási munkálatokat egyszerűsíti le, valamint a felcsatolt fájlrendszerek és azok kihasználtsága is megjelenik. A ***RAID*** tömböket is menedzselhetjük egy füst alatt.

A ***Devices and Hardware*** eléggé nagyképű menünév, mivel csak a soros portokat kezelhetjük.

A ***Java Web Console*** ötlete briliáns: az egyes ***Java***-ban írt alkalmazások saját vezérlőpultot készíthetnek, amelyeket aztán egy központi felületről gyorsan elérhetünk, akárhol is vagyunk.



1. ábra Információk a rendszerről

Sajnos még a ***Solaris 10***-be nincsenek beépítve olyan szoftverek, melyek kihasználnák ezt a lehetőséget.

Nyomkövetés vagy új programozási nyelv?

A ***Solaris*** egyik érdekes eszköze a ***DTrace***, mely mind a fejlesztők, mind a rendszergazdák körében nagy sikerre számíthat. Mire vonatkozik tehát az alcím kérdése? Ahhoz, hogy a ***DTrace***-t értelmesen használni tudjuk, alapszinten el kell sajátítanunk a ***D*** programozási nyelvet, melyet a ***Sun*** kimondottan a nyomkövetés céljából hozott létre. Még szerencse, hogy a ***D*** gyakorlatilag a ***C*** nyelv egyszerűsített, módosított változata. A szintaktika például ismerős lesz a ***C*** programozóknak. Szerteágazó eseményeket kérhetünk, hogy figyeljen a `dtrace`, a tesztrendszeren különféle telepítgetések után bő 42 ezer elemet számláló listából válogathatunk. Rendszergazdai jogosultságok kellenek a továbbiak használatához. A `dtrace -l` listázza ki a figyelhető akciókat. Vegyük elő a jól bevált szövegszerkesztőt és hozzuk létre az első ***D*** nyelvű programot:

```
proba.d
proc::exec_common:exec
{
    printf("%d - %s - %d\n",pid,
    ➥execname, uid);
}
syscall::kill:entry
{
    printf("%d - %s - %d\n",pid,
    ➥execname, uid);
}
```

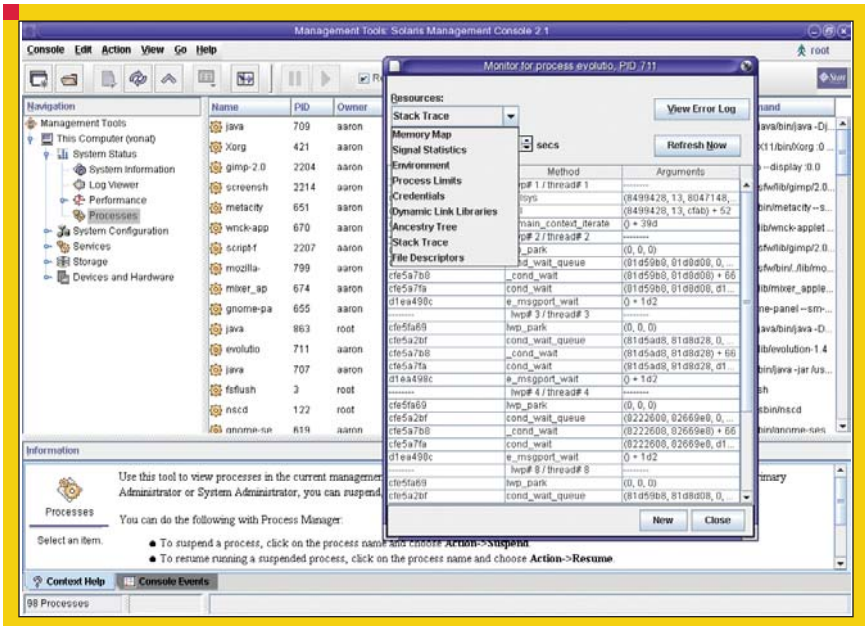
A ***D*** nyelvben is éppúgy létre lehetett volna hozni a klasszikus Helló, világ! példát, de ettől inkább eltekintek, hiszen a ***D*** nem általános célú programozási nyelv. Hogy is lehet megszólaltatni a fenti sorokat? A `dtrace -s proba.d` utasítással, ami nálam egy kis ténykedéssel ezt eredményezte: *1. lista*.

A `dtrace` addig fut, amíg meg nem szakítjuk és gondosan figyel. A ***proba.d*** első blokkja arra ad utasítást, hogy a `proc::exec common:exec` hívásokat kell figyelni és ha egy ilyen érkezett, akkor hajtsa végre a szabványos ***C*** nyelvből jól ismert printf függvényt a megadott formában. A folyamat azonosítója (***process id***), a folyamat neve és a tulajdonos azonosítója (***user id***) kerül ki a képernyőre. A második blokk szinte azonos, csak itt a `syscall::kill:entry`-ket figyeli. Most lássuk ezek után, mi történt, miközben én futtattam a `dtrace`-t. Elindítottam én (100-as ***UID***-em van) a ***Bash***-t, ahol megpróbáltam kilőni a `kill` paranccsal egy folyamatot. Ezután a `su` paranccsal átlényegültem rendszergazdává (0-ás ***UID***), aztán a `df` és az `uptime` utasításokat használtam. Ezután leállítottam a `dtrace`-t, eddig tartott az előadás.

Az, hogy a figyelés lehetőségét és a kapott adatokat mire használjuk fel, már csak a kreativitáson múlik. Egy tipp ***C*** fejlesztőknek: a memóriakezeléshez segítség lehet, ha a lefoglalásokat és felszabadításokat követjük nyomon. Rendszergazdáknak: a ***TCP*** események figyelésével látható, hogy mely folyamatok akarnak hálózati kapcsolatot létesíteni.

1. lista

```
dtrace: script 'proba.d' matched 2 probes
CPU     ID                    FUNCTION:NAME
  0    558           exec_common:exec 4349 - bash - 100
  0     78                kill:entry 3996 - bash - 100
  0    558           exec_common:exec 4350 - bash - 100
  0    558           exec_common:exec 4350 - su - 0
  0    558           exec_common:exec 4351 - bash - 0
  0    558           exec_common:exec 4352 - bash - 0
  0    558           exec_common:exec 4352 - uptime - 0
  0    558           exec_common:exec 4353 - bash - 0
```



2. ábra A folyamatok lelkivilága

Svájci bicska vagy szakbarbár?

Az újonnan telepített rendszerünk az alábbaikban áll a külvilág szolgálatára:

```
Starting nmap 3.81
➥( http://www.insecure.org/
➥nmap/ ) at 2006-02-09
➥19:38 CET
Interesting ports on
➥192.168.0.96:
(The 1642 ports scanned but not
➥shown below are in state:
➥closed)
PORT       STATE SERVICE
21/tcp     open  ftp
22/tcp     open  ssh
23/tcp     open  telnet
25/tcp     open  smtp
79/tcp     open  finger
111/tcp    open  rpcbind
513/tcp    open  login
514/tcp    open  shell
587/tcp    open  submission
898/tcp    open  sun-
➥manageconsole
4045/tcp   open  lockd
6000/tcp   open  X11
7100/tcp   open  font-service
32771/tcp open   sometimes-rpc5
32772/tcp open   sometimes-rpc7
32773/tcp open   sometimes-rpc9
32774/tcp open   sometimes-rpc11
32775/tcp open   sometimes-rpc13
32776/tcp open   sometimes-rpc15
32777/tcp open   sometimes-rpc17
32780/tcp open   sometimes-rpc23
32786/tcp open   sometimes-rpc25
MAC Address: 00:01:02:AA:1B:88
➥(3com)
Device type: general purpose
Running: Sun Solaris 9
OS details: Sun Solaris 9
Uptime 0.345 days (since Thu
➥Feb  9 11:22:38 2006)
```

Az `nmap` ezen verziója elvéti ugyan az operációs rendszer verziószámát, de az kiderül, hogy a rendszer túl sok porton figyel a külvilág felé. A telepítés során számtalan szolgáltatás kerül fel és ezek automatikusan be vannak üzemelve. Ez jó vagy rossz? Attól függ, hogy azt tekintjük kisebb munkának, hogy egy futó szolgáltatást kell kikapcsolni vagy pedig nem beüzemelt szolgáltatásokat bekonfigurálni. Én az előzőre szavazok, pláne, hogy a Solaris oly kényelmes eszközét kínálja a szolgáltatások kezelésére. Az alcím analógiájával élve: a svájci bicskát pillanatok alatt szakbarbárrá tehetjük, hogy kizárólag azt csinálja, amit mi megengedünk neki. Az

```
svcs -a
```

parancs kilistázza a telepített szolgáltatásokat, utána az `svcadm`-mal kezelésbe vehetjük ezeket. Öt perc alatt ki lehet gyomlálni a nem kívánt, külvilág felé portot nyitó fölösleget, ami jóval kevesebb idő, mint a meghagyott szoftverek érintőleges konfigurálása. Az `svcadm` használata sem bonyolult, utasításokat kell használnunk az svcs által kilistázott szolgáltatás-*URL*-ekre. Általános formában így néz ki: `svcadm` parancs szolgáltatás. A parancs lehet `enable`, `disable`, `restart` és `refresh`. Sorrendben: engedélyezi és elindítja, leállítja és letiltja, újraindítja és újra beolvassa a konfigurációját az adott szolgáltatásnak. Gyors munkát teszt lehetővé. Többre vágyunk? A rendszer portjait tűzfal is védheti!

Szögesdróton innen

Mint minden tisztességes és korszerű operációs rendszerhez, ehhez is tartozik tűzfal. A választás a szabadon elérhető ***IP Filter*** megoldásra esett. 2001 májusáig szolgálta például az ***OpenBSD*** felhasználókat ez a tűzfal. A köztudottan a biztonságot szem előtt tartó *BSD*-variáns csupán licensz okok miatt vált meg aztán az IP Filtertől. A Solaris 10-ben nem is olyan egyszerű engedélyezni a csomagok szűrését. Első lépésként a /etc/ipf/pfil.ap fájlban a hálózati kártyánknak megfelelő sorból távolítsuk el a megjegyzés(#) jelét. A hálózati kártyánkról az

```
ifconfig -a
```

parancs ad felvilágosítást. Ezután indítsuk újra a kapcsolódó szolgáltatást:

```
svcadm restart network/pfil
```

Most jön a neheze, állítsuk össze a tűzfal szabályait. A tesztgépen az */etc/ipf/ipf.conf* tartalma ez volt:

```
block out all
block in  all
pass out quick on elxl0 proto
➥tcp from any to any port = 80
➥keep state
pass out quick on elxl0 proto
➥tcp from any to
➥levelezoszerver_ipje port
➥= 995 keep state
pass out quick on elxl0 proto
➥tcp from any to smtp_ipje
➥port = 25 keep state
pass out quick on elxl0 proto
➥udp from any to dns_ipje port
➥= 53 keep state
```

A `to` utáni magyar szavak helyére a szolgáltatónk adatait helyettesítsük be, ha az olvasó is megelégszik azzal, hogy webezni és levelezni szabad, semmi mást nem. A 995-ös portot attól függően, hogy titkosított, illetve titkosítatlan és *POP3* vagy *IMAP* a protokoll, valószínűleg módosítani kell(súgó: */etc/services*). A tűzfal, mint ebből a kis példából kiderül, állapottartó, a kifele menő új kapcsolatok visszafelé jövő ágát engedélyezi, ha utasítjuk erre (`keep state`).
Az *IP Filter* tudásának ez csak egy egészen piciny szelete, érdemes megismerkedni vele.
A ➲ *http://coombs.anu.edu.au/~avalon/* oldalon részletes dokumentációt találunk. De még mindig nem működik a szűrés! A következő lépcső:

```
svcadm enable network/ipfilter
```

Elvileg a *SUN* útmutatása alapján ilyenkor nem kell újraindítani a rendszert, de talán az a legegyszerűbb, hogy végezzünk a munkával. Akik pedig inkább a nehezebb utat választják, ezzel a parancshalmazzal tudják beüzemelni a virtuális szögesdrótot:

```
ifconfig elxl0 unplumb
ifconfig elxl0 plumb ip_cim
➥netmask ide_a_netmask_jon  up
```

Persze az `elxl0`-t mindenki a saját hálózati kártyája szerint módosítsa. Ennyi volt, működik. Az ip címek fordítását (*NAT*) is elvégzi a *Solaris 10*, ha akarjuk, ehhez az *IP Filter* oldalán találunk segítséget.

Rendszerek, amerre a szem ellát

Az, hogy egy szem árva rendszerünk lett a telepítés során, az szép és jó, de mi van, ha éppenséggel tíz rendszert akarok? No miért is kéne egy rendszernél több? Mondjuk azért, hogy külön dobozba tudjuk zárni a sebezhető szoftvereket vagy esetleg az egyes szolgáltatások felügyelete több rendszergazda feladata, így a saját rendszerén mindenki egyedüliként lehet a 0-ás azonosítójú felhasználó. Ezt a képességet itt zónáknak hívják. Megkülönböztetünk globális és nem globális zónákat. A globális zóna tulajdonképpen az a rendszer, ami telepítéskor felkerült. Egyedül a globális zóna tartalmaz indítható rendszermagot (bootable kernel). A nem globális zónák a virtuális rendszereink sokasága. 8192 zóna lehet egy rendszeren elméletileg. Gyakorlatilag pedig annyi, amitől a gép még nem kezd csigalassúsággal vánszorogni. Persze az elvi határt akármilyen erős gép esetén sem léphetjük át. Úgy gondolom, hogy ez a szám nem fog senkit akadályozni a feladatai ellátásában. Hozzunk létre egy új zónát!
Elindítjuk a `proba_zona` nevű zóna beállítását:

```
# zonecfg -z proba_zona
proba_zona: No such zone
➥configured
Use 'create' to begin
➥configuring a new zone.
```

Létrehozzuk:

```
zonecfg:proba_zona> create
```

Beállítjuk a könyvtárát:

```
zonecfg:proba_zona> set
➥zonepath=/zones/proba_zona
```

Ha el akarjuk indítani automatikusan indításakor:

```
zonecfg:proba_zona> set
➥autoboot=true
```

A `tmp` fájlrendszer beállítása:

```
zonecfg:proba_zona> add fs
zonecfg:proba_zona:fs> set
➥dir=/shared/tmp
zonecfg:proba_zona:fs> set
➥special=/tmp
zonecfg:proba_zona:fs> set
➥type=lofs
zonecfg:proba_zona:fs> end
```

A hálózat beállítása:

```
zonecfg:proba_zona> add net
```

Egy külön *IP* címet kap a zóna:

```
zonecfg:proba_zona:net> set
address=192.168.0.5
```

Ide a saját hálózati kártya neve kerül:

```
zonecfg:proba_zona:net> set
➥physical=elxl0
zonecfg:proba_zona:net> end
zonecfg:proba_zona> verify
zonecfg:proba_zona> commit
zonecfg:proba_zona> exit
```

A munka oroszlánrészével már végeztünk is. A `zoneadm -z proba_zona install` elvégzi a fenti utasításokat és telepíti a teljes zónabeli környezetet. Mint egy valódi rendszert, a zónát is el kell indítani a `zoneadm -z proba_zona boot` paranccsal. A bejelentkezés sem marad el, a `zlogin proba_zona` beütésével már be is kerülhetünk az első létrehozott nem globális zónába. Itt aztán szájunk íze szerint konfigurálhatjuk a különféle szolgáltatásokat és az égvilágon mindent, anélkül, hogy ez hatással lenne a globális és a többi általunk létrehozott zónára. Most ugyan külön *IP* címet kapott a létrehozott zóna, de semmi akadálya annak, hogy a legkülönfélébb szolgáltatásokat (*SMTP*, web, *IRC*, stb) szétszedjük mondjuk három zónára három külön ip címmel, aztán a globális zónában az *IP Filtert* úgy állítjuk be, hogy úgy tűnjön, hogy egy rendszer dolgozik összesen. A zónák nagyon hasonlóak a *FreeBSD jail*-jéhez vagy a *chroot*-olt környezetekhez *Linux* alatt.
Az itt leírtak csupán ízelítőül szolgálnak arra, hogy felfedezzük és megismerjük azt, hogy mire képes a Solaris 10 rendszergazdai kezekben. A *Sun* meglepően bőséges és jó minőségű dokumentációt kínál térítés nélkül a ➲ *http://docs.sun.com* weboldalon, valamint létrehozott egy külön weboldalt *BigAdmin*(➲ *http://www.sun.com/bigadmin/home/index.html*) címmel, ahol szintén sok érdekes olvasnivalót találunk.
További élvezetes felfedezést kívánok mindenkinek!

Novák Áron
(aaron@szentimre.hu)

BME-VIK-es hallgató, műkedvelő rendszergazda. Jelenleg leginkább a NetBeans-szel és mindenféle hordozható eszközzel foglalkozik, legalábbis mindazokkal, amelyeket meg lehet szólaltatni Linux alatt.