A Mars-járművek irányítása

A NASA Spirit és Opportunity nevű Mars-járműveit irányító csoport asztali operációs rendszerként a Linuxot használja.

Amerikai idő szerint 2004 január 3-án reggel 8 óra 30 perckor a *Földről* küldött mintegy féltonnányi fém izzása ragyogta be a *Mars* egét. Hat perccel később az ejtőernyő és légzsákok kinyílása után a fémcsomag elérte a vörös bolygó felszínét, 28 felpattanással megtett mintegy 300 méteres távolságot, majd lassan gurulva megállt. A légzsákok leeresztettek, s ezzel láthatóvá vált az általa óvott gúla alakú fémdoboz, amely fokozatosan szétnyílva utat adott a benne lévő szerkezetnek, egy hatkerekű robot geológusnak, amelyet alkotói *Spirit* névre kereszteltek.
Így kezdődött a *Mars* három hónapig tartó felfedezése. A *NASA Jet Propulsion* laboratóriumának (*JPL*) több száz tudósból és mérnökből álló kutatócsoportja számára a szomszédos bolygónk felfedezésének új fejezete során a *Spirit* szolgáltatja a látást és az összecsukható karjának végén lévő szerszámokból álló eszköztárat. Az *Opportunity*, a *Spirit* ikertestvére három héttel később kezdi meg működését a *Mars* ellenkező oldalán. És vajon ezek a tudósok mit használnak a járművek irányítására? Nem mást, mint a *Linuxot*.
A *MER- küldetés* (*Mars Exploration Rover*, mars-felfedező jármű) fordulópontot jelent a *Linux* űrprogramokban történő felhasználásában. A *Linuxot* korábban is használták már űrkutatási programok során – például az *STS-83* űrrepülőgép fedélzetén már 1997-ben is *Debiannal* felszerelt hordozható gépet láthattunk –, de a *Mars-felfedező jármű* projekt az első *JPL*-küldetés, amely létfontosságú műveletek elvégzésére használja a *Linuxos* rendszereket. A *MER* során a *Linuxot* magasszintű tudományos tervezésre és alacsonyszintű parancselőállítási műveletekre, megjelenítésre és szimulációra egyaránt alkalmazzuk.

Hogyan jutottunk el idáig

Bármilyen furcsának tűnik is, eleinte nem állt szándékunkban a *Linuxot* használni elsődleges fejlesztési felületként. A programunkat eredetileg csak a 2001-es Mars-küldetésben akartuk felhasználni, amely lényegében a JPL 1997-es *Mars Pathfinder* küldetésének megismétlése lett volna. A terv az volt, hogy a jármű irányítását végző parancsokat egyetlen *Silicon Graphics* munkaállomásra bízzuk, ahogy a *Pathfinder* esetén is tettük korábban. Akkoriban a *Linuxos* kísérleteink gyenge próbálkozások voltak az *SGI* lehetőségeihez képest.

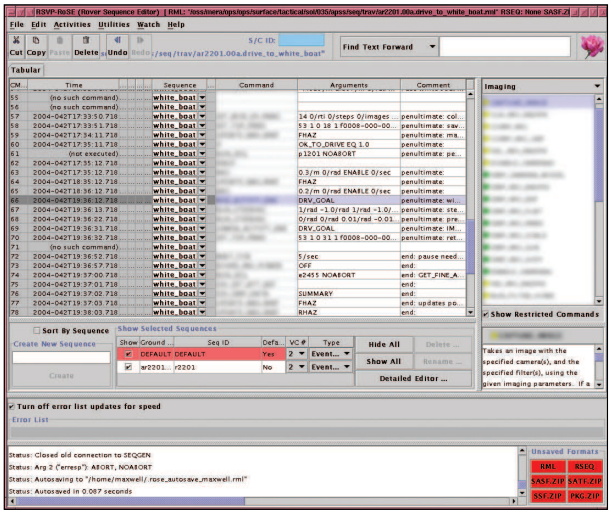
A *JPL* egy 98-as *Mars*-expedíciójának kudarca után azonban újragondolta a *Marssal* kapcsolatos stratégiáját. A 2001-es terveket egy későbbi indítás érdekében elvetettük, s ez a későbbi program lett végső soron a *MER*. Ebből kifolyólag a tervezettnél két évvel több fejlesztési idő állt rendelkezésünkre, jóllehet, nem is arra az űrhajóra vonatkozóan, ami az eredeti tervekben szerepelt. A *MER* járművei nagyobbak, intelligensebbek és összetettebbek, mint a *Pathfinder Sojourner* űrjárója, és mindegyik robotkarral is fel van szerelve.
Azokban az években a *Linux* és a futtatására alkalmazott x86 típusú gépek óriási ütemben fejlődtek mind a processzorsebesség, mind pedig a grafikus teljesítmény területén, s ez nem kis részben az *NVIDIA* grafikus lapkáinak és ezek erőteljes *Linuxos* támogatásának volt köszönhető. Ennek eredményeképpen kezdtünk egyre inkább a gyorsabb, jobb és olcsóbb Linux felé fordulni. A *MER* űrjárműveket így több csoport irányítja, párhuzamosan dolgozva a küldetéshez beszerzett több tucatnyi *Linuxos* gépen.

Az RSVP-rendszer

A használt *RVSP* (*Rover Sequencing and Visualization Program*, űrjármű vezérlő és megjelenítő program) nevű programcsomagunkat *Linux* alatt fejlesztettük, teszteltük és helyeztük üzembe. Az *RVSP* kifinomult eszközöket biztosít a *MER* tudósai és mérnökei számára a *Mars-járművek* irányításához. A nagymértékű fényidő-késleltetés miatt, amely a *Spirit* leszállásának napján közel 20 perces válaszidőt jelentett, lehetetlen a járműveket interaktív módon irányítani. Az *RVSP* ehelyett egy teljesen valósághű környezetet biztosít, amely teljes részlethűséggel jeleníti meg a jármű környezetét és szimulálja annak viselkedését. A marsi éjszaka folyamán az *RVSP* segítségével parancsszinten megtervezzük a napi teendőket, majd a kapott utasításokat műholdas kapcsolaton keresztül küldtük el a járműnek. A kapott parancsokat az űrjármű a következő marsi nap folyamán dolgozta fel. Az *RVSP* két fő alkotórésze a *RoSE* (*Rover Sequence Editor*, az űrjármű műveletszerkesztője), amely szöveges alapú felületet nyújt az űrjármű által végrehajtható parancsokhoz, valamint a *HyperDrive*, amely fejlett háromdimenziós grafikát szolgáltat a vezetési, karmozgatási és képalkotási parancsokhoz. A független programok mindegyike képes önálló működésre is, de igazából együttműködve hatékonyak. Együttes üzemmódban a programok az

Oak Ridge National Laboratories által kifejlesztett *PVM* (paralell virtual machine, párhuzamos virtuális gépek) program vezérlése alatt futnak. A *PVM* egy olyan adatbuszt biztosít, amely lehetőséget ad az *RVSP* komponenseinek a tevékenységük üzenetváltásokon keresztül történő összehangolására.

Az üzeneteink legtöbbjének tartalma az *RML*-re épül, amely az *XML* egy kifejezetten az űrjárművek parancssorozatainak leírására kidolgozott változata. Ennek az üzenetváltó megközelítésnek számos előnye van, amelyek közül nem elhanyagolható az sem, hogy az egyes különálló eszközöket különböző nyelveken valósíthatjuk meg. Amennyiben az eszköz képes *PVM*-üzenetek küldésére, máris a csomag tagjává válhat. A *RoSE*, az üzenettovábbító egység, és az üzenetnaplózó Java nyelven íródott; a HyperDrive, a képnézőnk és a parancsfolyam-böngészőnk forrásnyelve pedig a *C* és a *C++*. A következő küldetésben esetleg olyan parancsnyelveket is ki fogunk próbálni, mint a *Perl* és a *Python*.



1. kép A RoSE parancsszerkesztő (az ITAR-korlátozások miatt homályosítva)

A RoSE

A *RoSE* egy *Java* nyelven írt program, amely a *Sun Java* virtuális gépének 1.3.1 változatán fut, de nem a *Sun* fordítóprogramjával, hanem az *IBM* sokkal gyorsabb *Java* fordítójával, a *Jikes*-szal lett lefordítva. A *RoSE* kihasználja a *Java* jól ismert hordozhatóságát: jó hasznát vettük annak, hogy egy gyors *Linuxos* gépen fordíthattuk és tesztelhettük a programot, és semmi vagy nagyon kevés gondot okozott ennek az *SGI*-re történő telepítése és tesztelése. Még azzal is kísérleteztünk, hogy *Mac OS X*-re átültessük a programot, de ezzel felhagytunk, mivel nem volt arra időnk, hogy egy harmadik platformot is támogassunk. A *Mac OS X*-en sem jöttek elő kézzelfogható problémák, de ha fel is merültek volna, akkor sem jut időnk foglalkozni velük. Igény sem igen mutatkozott a *Mac OS X* támogatására, így teljesen elejtettük ezt a fejlesztési szálat.

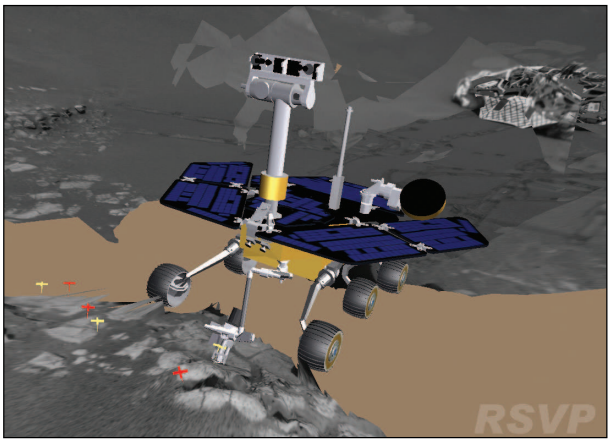
A *RoSE* igen erőteljesen adatvezérelt alkalmazás, amely az űrjárművel kapcsolatos minden tudnivalót *XML* állományokból olvas ki az indítási idő alatt. A Rose fejlesztéséhez használt integrált fejlesztői környezet a jól bevált *GNU Emacs* volt. A *RoSE* grafikus felületének nagy része a szabványos *Java GUI*-eszközkészlettel, a *Swinggel*, kézzel összerakott *Java*-kód. A grafikus felületet a *Swing* segítségével összerakni gyakran unalmas dolog. Ha a *Glade*-nek elérhető lett volna a *Javát* támogató változata, valószínűleg azt használtuk volna. A felhasználói felület fontos részei mégis dinamikusan kerülnek előállításra. Az indítási időben a *RoSE* által beolvasott egyik *XML* beállítófájl a jármű parancskönyvtárának a leírása, ami valami olyasmit jelent, mintha a jármű programozói felülete (API) lenne. A *RoSE* ezt a parancskönyvtárat használja arra, hogy dinamikus párbeszédablakokat állítson elő az egyes parancsok szerkesztésére, s ezzel a megközelítéssel rengeteg munkát takaríthattunk meg, mivel a programkönyvtár a fejlesztés évei során drámai mértékben megváltozott.

A *RoSE* megbízhatóságához jelentős mértékben járult hozzá a *Linux* könnyen automatizálható környezete. Már a kezdeti időszaktól kezdve terjedelmes öntesztelőkódokat írtunk a *RoSE* számára, majd egy egyszerű cron-munkafolyamatot hoztunk létre, amely minden éjjel lefuttatta ezeket a teszteket. Ha valamelyik teszt hibát jelzett, a következő reggel a fejlesztő erről elektronikus levélben kapott értesítést. Ezzel lehetővé vált a programhibák korai észlelése, amikor a hibát okozó változtatások még frissen éltek az emlékezetünkben. A *RoSE*-nak ez az intenzív öntesztje egyfajta kísérlet volt, s az ebbe fektetett energia később busásan megtérült.

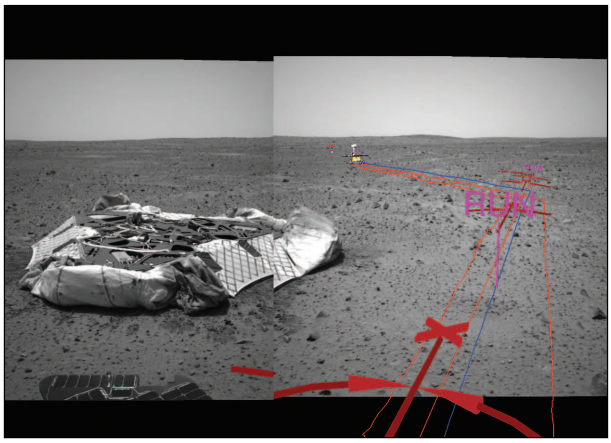
Mellékesen jegyzem meg, hogy a *Java*-t gyakran mondják lassúnak. A tapasztalataink alapján azonban azt kell mondanom, hogy ez a híresztelés teljességgel alaptalan, habár vannak a *RoSE*-nak olyan területei, ahol gyorsabb is lehetne, de ez bármelyik programról elmondható. Ha több időnk lett volna – ez a programfejlesztés állandóan emlegetett problémája – a program összes szűk keresztmetszetű részét sikerült volna kijavítani. A céljainknak a *Java* megfelelő sebességűnek bizonyult, a *Linux* jó *Java*-támogatása pedig nagyon nagy előnyt jelentett számunkra.

A HyperDrive

A *HyperDrive* az *RVSP* programcsomag interaktív megjelenítést végző összetevője, az a rész, amit majd a *CNN*-ben fognak mutogatni. A *HyperDrive* elsődleges célja, hogy lehetővé tegye a jármű irányításának és karmozgatásának biztonságos és hatékony megtervezését oly módon, hogy az irányítónak jól használható információkat ad a jármű pillanatnyi helyzetének és környezetének megértéséhez. Mindezt úgy éri el, hogy a rendelkezésre álló adatforrások egyesítésével egy háromdimenziós képet állít össze, és biztosítja ennek többféle megjelenítését. A *HyperDrive* számára az adatok egyik legfontosabb forrását a sztereó képek alapján előállított terepmodellek jelentik. A két járműre szerelt kamerák nagy része sztereó kamera, ami azt jelenti, hogy az emberi szemhez hasonlóan képesek a mélység érzékelésére. A sztereó összefüggésnek nevezett technika felhasználásával a képek sík, kétdimenziós világát háromdimenzióssá leképezett számítógépes grafikává alakíthatjuk át. A terepmodellek és a jármű által szolgáltatott CAD-adatok együttes felhasználásával előáll annak a rendszernek a magja, amely



2. kép Egy szikla árfúrása a kőkoptató eszközzel



3. kép A Spirit elhagyja a menedékét a kibővített élethűségű nézetben

képes a jármű nézetének leképezésére és lehetővé teszi annak interaktív mozgatását a marsi terepen, vagy a kar megfelelő mozgatását a bolygó felszínén (*2. kép*).
A leképezett képeket egyesítve a jármű kameráival nyert nyers képekkel egy még valósághűbb ábrázolásra nyílik mód, melynek során a jármű által érzékelt képet lefedve a mesterségesen előállított képpel, tanulmányozhatjuk azt, LCD shutter-szemüveggel akár három dimenzióban is (*3. kép*).
A jármű bármely kamerájának képe alapján szimulált nézetek állíthatók elő, segítve a kép alapján történő célkiválasztást. Ehhez a mesterségesen előállított világhoz olyan ikonokat és megjegyzéseket fűzünk, amelyek szemléltetik a végrehajtott parancssorozatot és leírják ennek a világnak a tulajdonságait. Ezeknek az objektumoknak az egyik legfontosabb csoportja egy ikonsorozat, amely a jármű által futtatandó parancsokat mutatja. Míg a ***RoSE*** lehetővé teszi a teljes parancssor szerkesztését, a ***HyperDrive*** csak azokat a parancsokat mutatja, amelyeknek érzékelhető vizuális hatásuk van. A ***HyperDrive***-ban a mozgatással, a robotkarral és a képalkotással kapcsolatos parancsok a terep felett megjelenő ikonsorokat képeznek, mindegyik azon a helyen, ahol végrehajtható, így lehetővé válik a jármű terepviszonyoktól függő mozgásának vizuális programozása és a robotkar nagy pontosságú célravezetése.
A ***Linux*** alatti háromdimenziós grafikai programozás egy addig egyedülálló kalandnak indult és a megbízható illesztőprogramokkal, programkönyvtárakkal egy alkalmas felületté érett. A ***HyperDrive*** a ***Silicon Graphics OpenGL Performer*** leképező programozói felületére épül.
A ***HyperDrive*** felhasználói felületének létrehozásához a GTK+ és libglade eszközöket használtuk. A ***Glade*** és libglade gyors prototípuskészítési lehetősége egyszerűvé és hatékonnyá tette a grafikus felület programozását. Mindenkinek bátran ajánljuk ezt az eszközkészletet, akinek megbízható felületkészítő eszközre van szüksége. Ahogy a fejlesztés célpontja az ***IRIX*** helyett minél inkább a ***Linux*** lett, egyre több nyílt forrású eszközt és programkönyvtárat használhattunk, majd a kompatibilitás megőrzése érdekében gondoskodtunk a létrehozott program visszaültetéséről ***IRIX*** alá.
A ***HyperDrive*** képes továbbá a járművek megjósolt viselkedésének animálására, az elképzelt mozgás valós időben történő megjelenítésére, lehetővé téve a felhasználó számára, hogy különféle nézőpontokból vizsgálja a jármű viselkedését.

Ugyanezt a képességét használhatjuk arra, hogy a marsjármű napi jelentése alapján visszajátsszuk a szerkezet valóságos működését.

Összegzés

Mindent összevetve, a ***Linux*** érett és a várakozásainkat messzemenően felülmúlva teljesítő programfejlesztő felület. A csapatunk tagjai közt a kezdetekkor előfordultak tapasztalt ***Linux***-támogatók és a témában teljesen járatlanok is, de mindnyájan elmondhatjuk, hogy hasznunkra voltak a ***Linuxszal*** és a felhasználói számára elérhető rengeteg nyílt forrású programmal kapcsolatos új tapasztalatok. Különösen az ***OpenGL***-alapú grafika az a terület, amely a drága grafikus szervereink egy alacsony kategóriájú alternatívájaként indulva végül a programjaink szempontjából felülmúlta a legjobb gépeket is.
Ezt a kutatást a ***Jet Propulsion*** laboratóriumban a ***Kaliforniai Műszaki Intézetben*** hajtottuk végre a ***NASA***-val kötött szerződés alapján. Semmilyen ebben a cikkben kereskedelmi termékre, folyamatra vagy szolgáltatásra a márkanévvel, márkajellel, a gyártó nevével vagy egyéb módon történő hivatkozás nem képezi vagy vonja maga után az ***Egyesült Államok*** kormányának, a ***Jet Propulsion*** laboratóriumnak vagy a ***Kaliforniai Műszaki Intézetnek*** a jóváhagyását.

Linux Journal 2004. szeptember, 125. szám

Frank Hartman jelenleg számítógépes grafikával és a Mars-űrjárművek vezérlésével foglalkozó szoftvermérnök a Jet Propulsion Laboratóriumban a kaliforniai Pasadenában. A Philadelphiai Művészeti Egyetemen szobrászatból szerzett képzőművészeti diplomát, és egy repülő- és űrhajózási mérnöki diplomát a Stanford Egyetemen. Frank a kaliforniai Altadenaban él feleségével, Leah-val.

Scott Maxwell programfejlesztő és a Mars-űrjármű irányító a Jet Propulsion Laboratóriumban a kaliforniai Pasadenában. Szeret aikidózni, klasszikus irodalmat olvasni és az életét röviden összefoglalni. Ő a Linux Core Kernel Commentary írója.