A magmódú Linux

Modulkészítés nélküli programfuttatás a magtérben.

A Kernel Mode Linux (KML) olyan módszer, ami az egyszerű felhasználói térben futó programok számára lehetővé teszi, hogy a magtérben fussanak. Írásunk ennek a hátterét világítja meg.

A hagyományos rendszermagok a processzor beépített szolgáltatásai révén védik magukat. A Linux-rendszermag például a saját védelmére a processzor legkiváltságosabb szintjét és memóriavédelmi szolgáltatásait használja. A rendszermag saját magának a legkiváltságosabb szintet tartja fenn: a magmódot. Ezzel szemben a felhasználói folyamatok a legkevésbé kiváltságos szinten, felhasználói üzemmódban futnak. Ilyeténképpen maga a processzor védi a rendszermagot, minthogy a felhasználói módban végrehajtott programok nem férhetnek hozzá olyan tárterületekhez, amelyek a magmódban végrehajtott programokhoz tartoznak.

Mivel a felhasználói folyamatok nem képesek rendszerhívásokat kiadni a rendszermagban, ezért e nehézség feloldásához a hagyományos rendszermagok a korszerű processzorok nyújtotta szolgáltatásokat aknázzák ki, biztonságosan, de korlátozott mértékben növelve a rendszer kiváltságos üzemmódjának a szintjét. Vegyük példaként az IA-32 felület számára készült Linux-rendszermag programvezérelt megszakításkezelő (software interrupt) eljárását.

A programvezérelt megszakítás különleges ugróutasításnak tekintendő, aminek a célcímét a rendszermag korlátozza.

A kezdeti értékadás során a programvezérelt megszakítás célcímét a rendszerhívásokat kezelő különleges eljárás címére állítja be. Rendszerhívások igénybevételéhez a felhasználói program egy különleges utasítást, az `int 0x80`-at hajtja végre, ekkor a rendszermagban lévő rendszerhívásokat kezelő eljárás magmódban lesz végrehajtva. Az eljárás környezetet vált, azaz menti a felhasználói program regisztereinek tartalmát. Végül meghívja azt a magszolgáltatást, ami a felhasználói program által kért rendszerszolgáltatást kivitelezi.

Egy mostani Pentium 4-es gépen a programvezérelt megszakításkezelés és környezetváltás mintegy 132-szer lassabb, mint egy puszta függvényhívás.

Mellesleg az IA-32 számára készült, 2.5.53 és nagyobb változatszámú Linux-rendszermagok néhány különleges utasítást használnak a rendszerhívásokhoz: a `sysenter`-t és a `sysexit`-et. Ez azonban még mindig 36-szor lassabb, mint egy egyszerű függvényhívás.

A rendszerhívások felgyorsításának kézenfekvő módja a felhasználói folyamatok magmódban történő végrehajtása. Ekkor a rendszerhívások kezelése felgyorsul, hiszen a programvezérelt megszakítások és környezetváltások szükségtelenek. Ezek ettől kezdve már nem többek egyszerű függvényhívásoknál, mert a felhasználói program most már közvetlenül hozzáfér a rendszermaghoz. Úgy tűnik, hogy ez a megközelítés biztonsági réshez vezet: a magmódban végrehajtott felhasználói programok a rendszermag tetszőleges részéhez hozzáférhetnek.

A statikus programelemzés területén elért eredmények, például a típuselmélet (type theory) felhasználhatók a rendszermag felhasználói programokkal szembeni védelmére. Számos módszer támogatja ezt a programalapú védelmi megközelítést, többek között a Java bytecode, a .NET CIL, a O'Caml, a Typed Assembly Language és a Proof-Carrying Code.

KML: felhasználói folyamatok végrehajtása magmódban

A rendszermag által védett program megalkotásához tett első lépésként kidolgoztam a KML-t. A KML tulajdonképpen a felhasználói folyamatokat magmódban futtató módosított mag, amit azután a magmódú felhasználói folyamatok hívnak meg. A magmódú felhasználói folyamatok közvetlen kölcsönhatásban állnak a rendszermaggal. Emiatt a rendszerhívásokkal kapcsolatos többletráfordítás elhanyagolható.

A KML az eredeti Linux-rendszermaghoz foltként áll rendelkezésre, emiatt a rendszermagot újra össze kell építeni. A KML használatához szükséges a folt, valamint a rendszermag beállításakor be kell kapcsolni a Kernel Mode Linuxot (KML). Építsük újra a rendszermagot, majd telepítsük, ezt követően indítsuk újra a gépet. A KML-folt a ➲ http://www.yl.is.s.u-tokyo.ac.jp/~tosh/kml címről tölthető le, illetve megtalálható a 48. CD Magazin/Magmod könyvtárában.

A legfrissebb KML-ben a */**trusted*** könyvtárban található programok magmódú felhasználói folyamatokként futnak; maga a rendszermag nem végez semmiféle biztonsági ellenőrzést. Vegyük például az alábbi parancsot:

```
% cp /bin/bash /trusted/bin &&
➥/trusted/bin/bash
```

Ez a bash parancsértelmezőt magmódban futtatja.

Mire képesek a magmódú felhasználói folyamatok?

A magmódú felhasználói folyamatok egyszerű felhasználói folyamatok, kivéve természetesen a kiváltsági szintjüket (privilege level). Éppen ezért ezek mindent meg tudnak tenni, amire az egyszerű felhasználói folyamatok képesek, például valamennyi rendszerhívásfajtát igénybe tudják venni, még a `fork`-ot (elágazás), a `clone`-t és az `mmap`-et is. Ezenkívül ha a legfrissebb GNU C könyvtárat – ezen a 2.3.2-es vagy ennél frissebb változatot vagy a CVS-ből származó fejlesztői változatot kell érteni – a magmódú felhasználói programokban a rendszerhívások önműködően függvényhívásokká alakítják, akkor akad néhány kivétel is, ilyen a `clone`. Ezért a felhasználói programodban levő rendszerhívásokból eredő többletráfordítás úgy lett felszámolva, hogy magát a programot nem is kellett hozzá módosítani.

A lapozási eljárás az egyszerű felhasználói folyamatokkal megegyező módon működik, vagyis a magmódú felhasználói folyamatok mindegyike saját címterülettel rendelkezik. Azon kívül, ha a magmódú felhasználói folyamatok feleslegesen sok memóriát foglalnának is le, a rendszermag önműködően kilapozza a tárból, ahogyan ezt az egyszerű felhasználói folyamatok esetében is tenné.

A kivételek, tehát a szakaszolási hibák (segmentation faults), érvénytelen műveleti kivételek (illegal operation exceptions) a felhasználói folyamatokkal megegyező módon kezelhetők, hacsak a program nem jogosulatlanul szeretne hozzáférni a

1. táblázat **Kísérleti környezet**

Processzor	Pentium 4 2,533 GHz (L2 gyorstár 512 KB)
Tár	1 GB (PC 2100 DDR SDRAM)
Merevlemez	80 GB
Operációs rendszer	Linux-rendszermag 2.5.59 (KML_2.5.59_001)
Libc	alfaváltozat glibc-2.3.2 a CVS-fából

2. táblázat **A rendszerhívások várakozási ideje (processzorciklusokban kifejezve)**

	Eredeti Linux	Magmódú Linux (KML)
getpid	432	12
gettimeofday	820	404

3. táblázat
Az újraolvasás átviteli sebessége: az átmeneti tár mérete = 8 KB

Áll. méret (KB)	Eredeti Linux (sysenterrel)	Magmódú Linux (KML)
16	2675	3223
32	2918	3188
64	3056	3365
128	2906	3205
256	2327	2486

(iozone -S 512 k -s 16 k -s 32 k -s 64 k -s 128 k -s 256 k -r 8 k) (Egység: MB/sec)

magból a tárterülethez, vagy hibásan hajt végre kiváltságokhoz kötött utasításokat (privileged instructions).
A példa kedvéért végezzük el a következő program összeépítését, és hajtsuk végre magmódú folyamatként:

```
int main(int argc, char* argv[])
{
    *(int*)0 = 1;

    return 0;
}
```

A folyamat végrehajtása szakaszolási hibával zárul, ez azonban nem jár együtt magpánikkal (kernel panic). Ez a példa egyúttal azt is megmutatja, hogy a jelző eljárás működik. A második példánkban építsük össze az alábbi programot és hajtsuk végre magmódú felhasználói folyamatként:

```
int main(int argc, char* argv[])
{
    for (;;);

    return 0;
}
```

Használjuk a CTRL-C billentyűket a folyamatnak való `SIGINT` küldéséhez. Figyeljük meg, hogy a folyamat meg is kapja a jelet, és rendesen befejeződik a végrehajtása. A második példánk azt szemlélteti, hogy a folyamatütemezés működik – vagyis még akkor is, ha egy magmódú felhasználói folyamat végtelen ciklusba kerül, az előjogait érvényesíti a folyamaton (leállítja), és más folyamatokat hajt végre. Talán már észre is vetted, hogy a rendszer továbbra is rendesen működik, még a példában levő végtelen ciklus ellenére is.

Milyen feladatok elvégzésére képtelenek a magmódú felhasználói folyamatok?

Annak ellenére, hogy a magmódú felhasználói folyamatok egyszerű felhasználói folyamatok, akad néhány őket érintő korlátozás is. Ha a magmódú felhasználói folyamatok megsértik ezeket a korlátozásokat, a rendszer meghatározatlan állapotba kerül. Ez a legrosszabb forgatókönyv szerint rendszerösszeomlást jelent. Lássuk az első korlátot!
Ne módosítsd a CS, DS, SS és FS szakaszregiszterek tartalmát. Az IA-32 számára készült KML feltételezi, hogy ezeket a regisztereket a magmódú felhasználói folyamatok nem módosítják; belül használja őket. A második korlát pedig az, hogy kiváltsághoz kötött műveleteket (privileged actions) ne hajts végre helytelen módon. Magmódban a programoknak bármilyen kivételezett műveletet jogukban áll végrehajtani. Abban az esetben azonban, ha ez a művelet az adott rendszermaggal nincs engedélyezve, a rendszer meghatározatlan állapotba kerül. Erre az esetre mutat példát a következő magmódú felhasználói folyamat:

```
int main(int argc, char* argv[])
{
    /* disable hardware interrupts */
    __asm__ __volatile__ ("cli");

    for (;;);

    return 0;
}
```

Ekkor a rendszer működése megakad. Tapasztalatom szerint csak kevés alkalmazás sérti meg ezeket a korlátokat. A korlátok megsértői között találjuk a WINE-t és a VMware-t is. Ezek a korlátok csak a magmódú felhasználói folyamatok számára állnak fenn. Az egyszerű felhasználói folyamatokat sohasem sújtják ezek a korlátok, még akkor sem, ha KML-képes rendszermagot futtatunk.

KML-belügyek

Az IA-32 processzorokban a futó program kiváltsági szintjét annak a kódrészletnek a kiváltsági szintje határozza meg, ahol a program végrehajtódik. Emlékezzünk csak vissza, hogy az IA-32 processzorok egyrészt egy szakaszból állnak, amelyet a CS elnevezésű szakaszregiszter jelöl ki, másrészt a szakaszba mutató eltolásból (offset), amit az EIP regiszter határoz meg. A kódszakasz kiváltsági szintjét tehát a szakaszleíró határozza meg. A szakaszleírónak van egy mezője, ami a szakasz kiváltsági szintjét határozza meg.
A Linux-rendszermag alapvetően két szakaszt hoz létre: a rendszermag kódszakaszát és a felhasználói kódszakaszt. A rendszermagkódszakaszt maga a rendszermag használja, kiváltsági szintje pedig magmódra (kernel mode) van állítva. A felhasználói kódszakasz egyszerű felhasználói folyamatoknak van fenntartva, kiváltsági szintje pedig felhasználói módra van beállítva (user mode). Amikor az `execve`-t felhasználói folyamat futtatására használjuk, az eredeti Linux-rendszermag a szakaszregisztert a felhasználói kódszakaszra állítja be. Így aztán a felhasználói folyamat valódi felhasználói üzemmódban

lesz végrehajtva. Ahhoz viszont, hogy a felhasználói folyamatot magmódú felhasználói folyamatként hajthassuk végre, a KML a folyamathoz tartozó CS regisztert – a felhasználói kódszakasz helyett – rendszermag kódszakaszra állítja. Ezután a folyamat magmódban hajtódik végre. A KML egyszerű megközelítése miatt a magmódú felhasználói folyamat átlagos felhasználói folyamat lehet.

A vereméséghiba és megoldása

A KML alapvető megközelítése egyszerű, a benne rejlő legnagyobb hibát vereméségnek nevezik. Mindenekelőtt azt fejtem ki, hogy az eredeti Linux-rendszermag miként kezeli a kivételeket – a lapozási hibákat és a megszakításokat időzítő megszakításokat – az IA 32 processzorokon. Ezt követően ismertetem a vereméséget, végül pedig az általam kidolgozott megoldást mutatom be. Az eredeti Linux-rendszermagban a megszakításkezelést a megszakításleíró táblában (Interrupt Descriptor Table, azaz IDT) kapuként meghatározott megszakításkezelő eljárások végzik. Ha megszakítási kérelem érkezik, az IA-32 felfüggeszti a program futását, menti a program végrehajtási környezetét, és végrehajtja a megszakításkezelő eljárást. A program kiváltsági szintjétől függ, hogy az IA-32 a megszakítások érkezésekor milyen módon menti a futó program végrehajtási környezetét. Ha az IA-32 a programot felhasználói módban hajtja végre, a memória vermét önműködően átkapcsolja a magveremre, azután a végrehajtási környezetet menti a magverembe. Amennyiben a program végrehajtása magmódban történik, az IA-32 processzor nem kapcsol át a saját tárvermére és nem menti a környezetet (az EIP, EFLAGS, ESP, és az SS regiszter tartalmát) a futó program tárvermébe. De vajon mi történik akkor, ha a KML magmódú felhasználói folyamata hozzáfér a tárveremhez, amit a processzor laptáblái még ki sem osztottak? Először laphiba keletkezik, ekkor a processzor megpróbálja megszakítani a folyamatot, majd megkísérel elugrani az IDT-ben meghatározott hibakezelőhöz. Ezt a feladatot azonban képtelen végrehajtani, hiszen nincs verem, ahová a végrehajtási környezetet menteni tudná, ugyanis a folyamat végrehajtása magmódban történt, így a processzor sohasem képes a tárveremből a magverembe kapcsolni. A processzor e végzetes hiba jelzésére egy különleges védelmi hibát próbál meg létrehozni, az úgynevezett kettős hibát. Újra csak a régi hibával állunk szemben: a processzor nem tud kettős védelmi hibát létrehozni, mert nincs verem, ahová a futó folyamat végrehajtási környezetét menthetné. Végül a processzor feladja, és újraindítja a gépet.

A vereméség megoldásához a KML az IA-32 processzor feladatkezelő szolgáltatását használja. A szolgáltatás használatával a mag egyetlen utasítás segítségével képes váltogatni a folyamatok között. A jelenlegi rendszermagok azonban már nem használják ezt a szolgáltatást, mert a kizárólag programvezérelt megoldásoknál lassabb, így szinte feledésbe merült.

Az IA-32 processzorok esetében e szolgáltatás ereje abban rejlik, hogy a megszakítások és a kivételek kezelésére egyaránt használható. Az IA-32 processzor által kezelt feladatokat be lehet állítani az IDT-re. Amikor megszakítás keletkezik, egy feladat rendelődik hozzá, ez kezeli a megszakítást – a processzor a megszakított program végrehajtási környezetét a tárvermek helyett először a feladat adatszerkezetébe menti. Ezt követően a processzor az IDT-ben meghatározott feladat-adatszerkezetből helyreállítja a környezetet.

A legfontosabb az, hogyha a feladatkezelő szolgáltatást használjuk a megszakítások kezelésére, akkor a tárveremben nincs szükség kapcsolgatásokra, vagyis ha a laphibákat ezzel a programmal kezeljük, akkor a magmódú felhasználói folyamat képes biztonságosan elérni a tárvermet. Ha viszont az összes lapozási hibát ezzel a szolgáltatással kezeljük, akkor romlik a rendszer működőképessége, mivel a feladatalapú megszakításkezelés lassabb, mint az egyszerű megszakításkezelés. Ezért csak a kettős kivétel hibákat kezeljük ilyen módon, más szóval kizárólag a tárverem hiányból adódó hibák kezelését végzi ez a szolgáltatás. Tapasztalatom szerint a tárvermek csak ritkán okoznak lapozási hibákat, és a működéscsökkenés is elhanyagolható.

A működés mérése

A működés javulásának mérésére két kísérletet végeztem. Mindkét mérés az eredeti és KML rendszermag működését vette össze. Az eredeti Linux-rendszermag működésének mérésére a `sysenter/sysexit` módszert választottam a `0x80`-as utasítás helyett. A kísérleti környezet az *1. táblázatban* látható. Az első kísérletben a `getpid` és `gettimeofday` rendszerhívások várakozási idejét mértem, a rendszerhívásokat a felhasználói programok közvetlenül kezdeményezték a mérésekben, a libc nélkül. A várakozási időt az `rdtsc` utasítással mértem, az eredményeket pedig a *2. táblázatban* foglaltam össze.

A végeredmény azt tükrözi, hogy a `getpid` a KML-ben 36-szor volt gyorsabb, mint az eredeti Linux-rendszermagban; a `gettimeofday` pedig kétszer olyan gyors volt a KML-ben, mint az eredeti Linux-magban.

A második kísérlet állomány B/K-mérés volt, amit az Iozone állományrendszer-mérővel folytattam le. A be- és kivitelt négy fájlművelettípusra nézve vizsgáltam meg: írás, újraírás, olvasás és újraolvasás. A mérést különböző méretű állományokkal végeztem, aminek a mérete 16 KB-tól 256-KB-ig terjedt, az átmeneti tár méretét pedig 8 KB-ban rögzítettem.

A rendszer hátterét az `ext3` állományrendszer adta. Minden egyes Iozone-mérést harmincszor hajtottam végre, és a legjobb átviteli eredményt választottam.

Az újraolvasás átviteli teljesítménye a *3. táblázatban* található. Helyhiány miatt az írás, újraírás, olvasás és újraolvasás részletes eredményeinek bemutatásától eltekintünk.

Az eredmény azt mutatja, hogy KML-ben az újraolvasás sebessége 6,8–21 százalékkal javult. Ezenkívül az írás 0,6–3,2%-kal, az újraírás 3,3–5,3%-kal és az olvasás 3,1–15%-kal lett gyorsabb. Ezek a kísérleti eredmények azt jelzik, hogy a KML képes javítani a rendszerhívásokat gyakran igénybevevő alkalmazások működésén, amik kisebb állományokat olvasnak vagy írnak. Például a webkiszolgálók, adatbázisok hatékonyan üzemeltethetők KML-ben.

A megelőző kísérletekből érdemes megemlíteni, hogy KML kiiktatta a rendszerhívásokkal járó többletterhet. Az alkalmazáson végzett kevés finomítással a KML további teljesítményjavulást mutathat. Például a magmódú felhasználói folyamatok a további javulás érdekében képesek lesznek közvetlenül hozzáférni az átmeneti tárakhoz.

A kapcsolódó címek megtalálhatóak a 48. CD Magazin/Magmod könyvtárában.

Linux Journal 2003. május, 109. szám

Toshiyuki Maeda

A Tokiói Egyetemen számítógéptudományból készül PhD fokozatot szerezni. Kedvenc képregényei a Hikaru no GO (Hikaru go-játéka), Jojo no Kimio na Boken (Jojo bizarr kalandja), és Runatikku Zatsugidan (Bolondos akrobata-színtársulat).