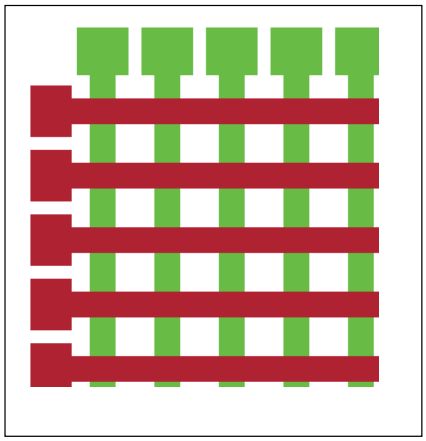
Linux futtatása sérült memóriával

A BadRAM folt segítségével tökéletesen lehet Linuxot futtatni olyan hibás memóriával, ami egyébként sok galibát okozhat.

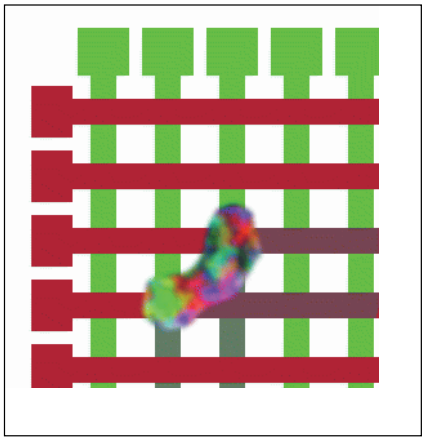
A memóriamodulok meghibásodásának számos oka lehet. Mielőtt orvosolnánk ezeket a hibákat, nézzük meg az okokat. Megértésükhöz vessünk egy pillantást az *1. ábrára*, amelyen a hibátlan memória vázlatos ábráját láthatjuk. A memóriában kiterítve több, majdnem ugyanannyi sor és oszlop található. A memóriacellában minden kereszt általában egy bitet tárol. Ebből következően, egy adott memóriacella címe a szóban forgó sor és oszlop címéből tevődik össze. A memóriamodulokba sorban, egymás után betápláljuk ezt a két „félcímet", mivel a címsín csak fele olyan széles, mint szeretnénk. Ezt a címfelezést egyébként az alaplap végzi.
A memóriamodulok hibáinak okait tekintve a gyártás áll az első helyen. Ez nagyon érzékeny eljárás, főleg a lapka fizikai méretei miatt, mivel a világban egyre nagyobb teljesítőképességű memóriákat igényelnek. A lapka gyártói nem takarékoskodnak a környezeti erőforrásokkal, mivel elég sok erősen tisztított vegyszert használnak fel egyetlen apró kis „homokszármazék" előállításához. Ezenkívül a lapka érzékenysége miatt viszonylag sok már a gyártás során meghibásodik.
Az egyik részleges megoldás az lehet, hogy adatismétléses tárolással látják el a lapkákat, azaz minden harminckét memóriacella-sor tartalmaz egy 33. sort is, így ha egy sor meghibásodik, a jelet átirányítja a külön sorra. Ezt a módszert alkalmazza néhány gyártó, ennek ellenére bizonyos meg nem erősített hírek szerint a selejt aránya még így is hatvan százalék körül mozog. Talán felesleges is mondanom, hogy az ilyen és hasonló okok miatt a memóriagyártás nem olcsó mulatság.
A gyártás során keletkezett hibákat okozhatja egy apró szennyeződés is, amely a levilágítás vagy a fejlesztés során kerülhet az egyik rétegre, ahogy az a *2. ábrán* is látható. Az árnyékolt terület, amely hibás működéshez vezethet, egy apró szennyeződés eredménye.
A másik hibaforrás a kisülés. Ez ugyanaz a jelenség, mint amit egy pamut vagy nejlonpulóver levételekor tapasztalhatunk. A feltöltődés magas feszültségen, egymáshoz közeli felületeknél jöhet létre. Ha a felületek elég közel kerülnek egymáshoz, a töltés hirtelen átugrik, ionizálja a levegőt, kisül, és nagyon rövid ideig igen nagy feszültséget gerjeszt. Teljesen ugyanaz, mint a villámlás, csak természetesen nem olyan erős. A lapka a kis mérete miatt igen érzékeny ezekre az erős kisülésekre. A kisülés a táregységeket, a sorok és oszlopok közötti összeköttetést, valamint a címválasztó logikát teszi tönkre, amint az a *3. ábrán* is látható. Ez a gyakorlatban azt jelenti, hogy az egész sor, az oszlop vagy mindkettő használhatatlanná válik, itt is ezt jelzi az árnyékolt terület. Ezekkel a memóriahibákkal találkozhatunk a leggyakrabban.
Az utolsó hibaforrás, amivel eddig még személyesen nem is találkoztam, a lapka fokozatos elöregedése. Ilyenkor a lapkán az egyébként élesen szétválasztott áramkörök összemosódnak, keverednek. Ez egy természetes folyamat, és ha a lapkát kellően hűvös helyen használjuk, valószínűleg egy-két évtized is kell hozzá, hogy bekövetkezzen. A memória esetében ez talán nem is olyan nagy gond, hiszen ennyi idő alatt már úgyis elavulttá válik a sebessége, illetve a mérete miatt.
Az ilyen sérüléseknél a legfontosabb dolog, hogy nem okoznak véletlenszerű hibákat. Az előfordulhat, hogy a hibás bitek valamilyen minta szerint jelentkeznek, de ha egyszer hibásak, akkor azok is maradnak.
Fontos, hogy a hibák nem keletkeznek csak úgy maguktól. A bajt a hirtelen kicsapódó energia okozza, vagy egy nagyon lassú folyamat. Szóval nem kell arra számítanunk, hogy folyamatosan egymás után jelentkeznek majd a hibák, még akkor sem, ha a modulunk sérült. Mivel a hibák a memóriában nem terjeszkednek, általában kijátszhatók egy ügyes dinamikus memóriafoglalási módszerrel.

A memóriahibák keresése

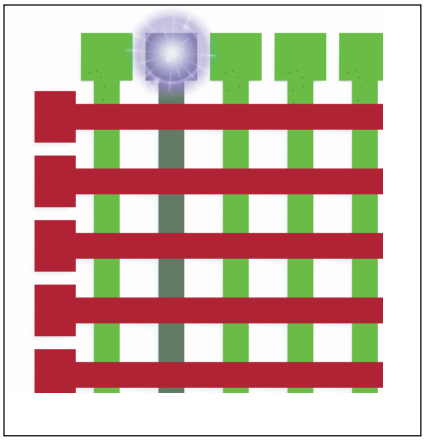
Alapjában véve a memóriahibák keresésére két eljárást használhatunk. Az egyik lehetőség egy program használata, mely átfésüli a teljes memóriát, és jelenti a hibás címeket. Ilyen program az i386-os rendszerekhez készült *memtest86,* mely a mezőnyből hibakeresési képességeivel tűnik ki. Természetesen az ellenőrzés megbízhatósága függ a hiba állandóságától is, de ez általában megfelelő, én már hónapok óta használom a gépem anélkül, hogy változtak volna a hibák a modulomban.



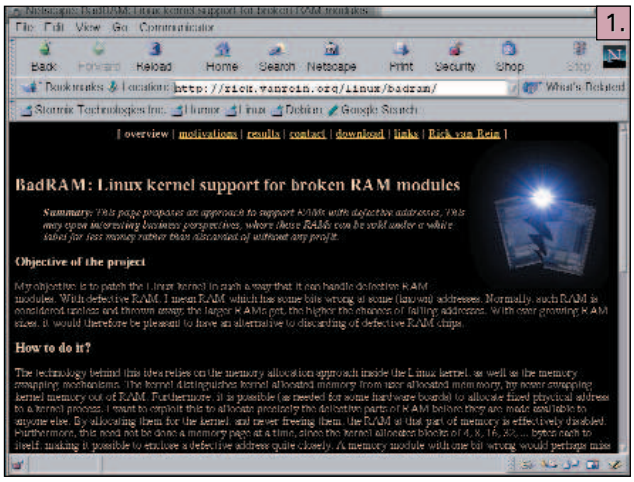
1. ábra Hibátlan memóriaszerkezet

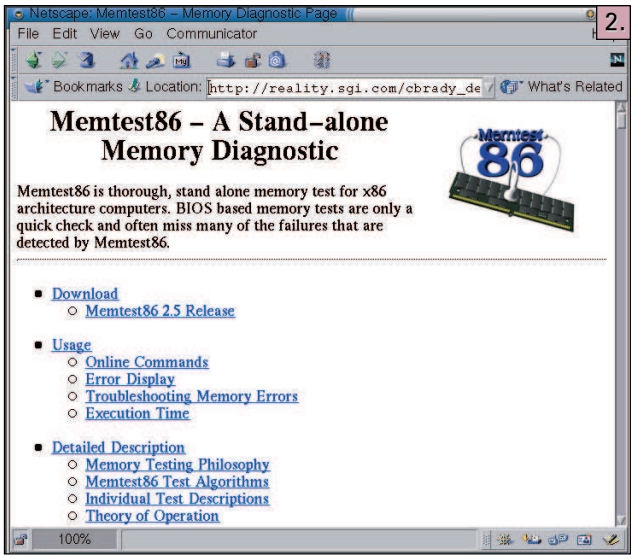


2. ábra Memóriamaratási hiba



3. ábra Az átmeneti tár sérülése kisülés következtében





A másik lehetőség a gépi hibakeresés. A harminctűs SIMM modulok korában elterjedt megoldás volt, hogy használtak egy kilencedik bitet a párosságadat (parity) tárolására. A kilencedik bit lényege, hogy külön helyezkedik el a memóriamodulon, és a tárolt adatok páros (vagy éppen páratlan) voltát jelzi. Íráskor ezt az adatot is újraszámolja a rendszer, olvasáskor pedig ellenőrzi: „párossá” teszi az egész lapkát – vagy páratlanná, ahogy éppen az alaplap kívánja. Íráskor ez a bit megjelenik, és visszaolvasáskor ellenőrzésre kerül. Ha olvasáskor az ellenőrzés hibát jelez, a rendszer párossági hibát vált ki.
A párosságjelző korszerű megfelelője az ECC (Error Correction Code, hibajavító kód). Ez általában valamilyen CRC-jegyzék a tárolt bitekről, és a 32-ből három hibás bit felderítésére alkalmas, valamint jól javít legfeljebb két hibás bitet. (Nem vagyok teljesen biztos a pontos értékekben, de az elmélet a lényeg.) Tehát az ECC RAM-mal lehetséges a hibák felderítése, és jól is javítja a kisebbeket.
Az újabb memóriamodulokban, melyeket a PC munkaállomásokban is használunk, általában nincsen sem párosság-ellenőrzés, sem pedig ECC. Másrészről a csúcskategóriás kiszolgálókban, mint a VA Linux és a Sun, rendszeresen találhatunk ECC-vel ellátott, de legalább párosságjelző memóriákat. Ha jól tudom, a legtöbb PC képes lenne ECC-memóriával működni, de többnyire nem használunk ilyen modulokat, ugyanis ezek drágábbak egyszerűbb testvéreiknél. Igen, az olcsó PC-k világa visszakacsint ránk. A párosságjelző vagy ECC-RAM tulajdonságaira támaszkodva működés közben is felderíthetők lennének a hibák, és nem volna szükség ellenőrző programokra (mint amilyen a memtest86), de a BadRAM foltot, amit ebben a cikkben tárgyalunk, a lehető legegyszerűbb szerkezetben kívántuk megtartani, hátha bekerülhet a rendszermagba, ezért nem tudtunk külön foglalkozni benne az ECC különleges lehetőségeivel.

A hibák megfogalmazása mintákban

Tegyük fel, hogy a memtest86-hoz hasonló memóriaellenőrzők a modulokban keresnek hibákat. Amint azt említettem, a kereső kigyűjti a fellelt hibás címeket. Mivel egy egész sor vagy oszlop is meghibásodhat egy kisülés következtében, ilyenkor bizony hatalmas listák keletkezhetnek. Természetesen az ilyen listák unalmasak, és az esetlegesen fellépő hibák miatt veszélyes volna a rendszermagba gépelni azokat. Sőt, a rendszerindításkor rendelkezésre álló korlátozott erőforrások miatt gyakran használhatatlanok is. Az lenne a tökéletes, ha az összes fellelt hibáról készíthetnénk egy rövid kis leírást. Szerencsére a memóriahibák általában szabályos alakzatokban helyezkednek el, például a `0x1234` címtől kezdve, minden `0x0040` bájtonkénti előfordulással, de ennél bonyolultabb mintázat is kialakulhat. A szabályosságot akkor a legkönnyebb megállapítani, ha binárisan vizsgáljuk az adatokat.
Ennek oka, hogy a sorok és oszlopok címzése úgy történik, hogy címbiteket küldünk e vezetékekre, így a cím egy részének megfejtésével eldönthetjük, hogy a használt címvezetékek megfelelő területre esnek-e vagy sem.
Egy egyszerű hiba tulajdonképpen egy olyan cím, amelynek minden bitje értékes adat. Például a `0x1234,0xffff`, ahol az első szám az alapcím, és a második a maszk. A maszk „1” bitje jelzi, hogy melyek a megfelelő bázisalapcím használható bitjei. Ha ezen a címen kívül a `0x0040`-nel nagyobb cím is rossz, akkor ennek jelzésére csak módosítanunk kell a maszkot. A cím/maszk pár most `0x1234,0xffbf` lett. Láthatjuk, hogy ez helyes, mert a lefedett címek mindegyike A cím, ahol `(A & 0xffbf)=0x1234`, vagy pontosabban `0x1234` és `0x1274`. Ha ott tizenhat hibás címet találunk ezzel a kitöltési tényezővel, és az egész a `0x1234,0xfc3f` címen kezdődik, már meg is vannak a hibás címek: `0x1234, 0x1274, ..., 0x15f4`.
Ez az eljárás jól működik sorok és oszlopok hibáinak keresésénél, vagy ha egy apró szennyeződés kikapcsol néhány szomszédos bitet a lapkán. De mi történik, ha a modulban további hibák is vannak? Vagy ha több modulban is akadnak hibák? A tények feltárásához legtöbbször feljegyezzük az említett cím/maszk párt. Már öt ilyen pár alapján is elindulhatunk. Öt párral általában nem fordulnak elő különleges nehézségek.
A memtest86 a 2.3-as változattól indul, és képes előállítani BadRAM mintákat. Ezeket közvetlenül a parancssorba írhatjuk be, ha a Linux rendszermagba befordítjuk a BadRAM csomagot. A fenti példában a memtest86 jelenti a mintákat, általában így:

```
badram=0x1234,0xfb3f
```

Írjuk ezt le. Ha megvannak az adatok, indítsuk újra a rendszert, és a parancssorba írjuk be a következőket:

```
LILO: linux badram=0x1234,0xfb3f
```

Ha a rendszer gond nélkül betöltődött, tiltsuk le a hibás memóriarészeket. Ezek után az `/etc/lilo.conf` fájlt is bővíthetjük az alábbi sorral:

```
append="badram=0x1234,0xfb3f"
```

A módosítás után futtassuk a LILO-t.! A következő rendszerbetöltéseknél már nem kell beírnunk a cím/maszk párokat.

A rendszermag javítása

Az én régi ZX Spectrum számítógépemet nagyon egyszerűen bővíthettem további 32 kB memóriával. A Sinclair memóriabővítő készlet egy 64 kB tárméretű lapkából állt, amiben mind az alsó, mind pedig a felső egység tovább dolgozhatott, ha az egyik meghibásodott, és az ára is kedvezőbb volt, mint az „igazi” 32 kB-os bővítő lapkának. Az alaplapon kapcsoló segítségével beállíthattam, hogy a modul melyik felét akartam használni. Alapjában véve a BadRAM ugyanezt teszi, csak kicsit ravaszabban.

A memóriakezelő egység, mely minden Linux-változat nélkülözhetetlen része, átirányítja a felhasználó programja által elérhető „felhasználói területet” a megfelelő fizikai memórialapra. Ez nagy, összefüggő memóriaterületnek látszik, de valójában darabokban helyezkedik el a memóriamodulban (és néha az átmeneti tárolóba másolódik). Ha egy felhasználói szintű program elfoglalja a memóriát, a rendszermag egyszerűen oldalakat ad neki a fizikai memóriából.

A Linux egyetlen, fizikai memóriacímzéssel működő része a rendszermag, ez a memória elejére töltődik be – természetesen itt nem lehetnek hibák –, és a felhasználói programok számára fenntartott memóriaterületből foglal el blokkokat.

Ez a memória rendszerbetöltéskor töltődik fel fizikai lapokkal. Valójában a BadRAM azokat a lapokat hagyja ki, amelyek a parancssorban beírt cím/maszk pároknak megfelelnek.

Ez azt jelenti, hogy a BadRAM tulajdonképpen a rendszerbetöltéskor végzi el a feladatát. Tehát egyetlen hatása, hogy a szabad memóriából elfoglal egy csipetnyit. Az alapos ellenőrzőprogramok képesek ezt kimutatni, de a teljesítményre nincs érezhető hatással.

Különböző alkalmazások

Ha ehhez hasonló csomagokat készítesz, és véletlenül a SlashDot is megírja, bizonyára csomó érdekes levelet kapsz, néha érdekfeszítő gondolatokkal. Egyik javaslat, hogy rendszerbetöltés közben végezzünk memóriavizsgálatot. Mókás ötlet, de nem célszerű.

A memtest86 kiváló munkát végez, de elnyammog néhány óráig. Ezt nyilván senki sem szeretné kivárni a rendszer indításakor, de rossz memóriaellenőrzést sem akarunk. (Számos PC BIOS-szal próbálkoztunk, és a BadRAM általában elvégezte az ellenőrzést.)

A memtest86-ot indító LILO-bejegyzés, vagy a memtest86 indítólemez mindig a kedvenc megoldásaim közé tartoztak. Érkezett néhány jelentés olyan alaplapokról, melyek hibáznak egy bizonyos közvetlen memóriacímen, amit egy alaplapra szerelt eszköz rövidzárlata okoz. Az illető azt írta, hogy a hibás címek elkerülésére négy 512 megabájtos modult tett a számítógépébe, de a gondját igazából a BadRAM oldotta meg, egyetlen memórialap kihagyásával. A két gigabájtból letiltott négy kilobájtot, és a gép most gond nélkül működik.

Beszéltem valakivel, akinek valamelyik „PC otthonra!” programból származik a számítógépe. Az átlagos felszereltségű Compaq Deskpránál nem lehet kikapcsolni a néhány régebbi ISA kártya által igényelt 15–16 M közötti kiterjesztett memóriát. Szóval, a 24 megabájtra bővített memóriával nem működik a Linux 2.2 mem=24 M beállítása, mivel a 15–16 MB közötti területet is használná. De miután a `badram=0x00f00000, 0xfff00000` sorokkal a rendszermag tudomására hozta a hibát, a számítógép készen állt a memóriabővítésre, és most jobban megy, mint valaha.

Kaptam visszajelzéseket olyan emberektől, akik régebbi gépeiken párosságellenőrző és ECC RAM-okat is használtak, memóriahibákat tapasztaltak és a hibás lapok megbízhatatlanná váltak. Programkód betöltése közben a memórialap „under evaluation” állapotba került, aztán az az ötletük támadt, hogy a programkód a merevlemezről másolódik be, és hiba esetén onnan visszaállítható. Jól is működött a dolog egy darabig, amint eltűnt a hiba, a lap megint „megbízhatóvá” vált. Ha azonban a program tárolása sem működött, a lap kimaradt a használatból. Ez a megoldás érdekesnek tűnik, de futásidőben követelne processzorteljesítményt, így kénytelenek vagyunk fényűző szolgáltatásnak minősíteni.

Lehetséges bővítések a jövőben

Van néhány eljárás, amellyel növelhető a BadRAM hatékonysága. Az ECC modulok jól használhatók a biztonság növelésére a javítható és a javíthatatlan hibákhoz egyaránt. Mivel nekem sajnos egyik memóriamodulom sincs, a dolog meghaladja lehetőségeimet. Mivel a processzorteljesítményre folyamatosan van szükség, én ezt a szolgáltatást is a fényűzés tárgykörbe sorolnám.

Másik lehetőség a magtáblafoglaló-kezelő (slab allocator) fejlesztése. A táblák kicsi, egyforma és újra felhasználható memóriablokkok, amelyeket a rendszermag tömbökbe rendez. Lehetőség nyílna arra, hogy a hibás címekkel kapcsolatos adatot részletesebben nyerjük ki, ha a táblákhoz BadRAM oldalakat használnánk, és így elkerülhető lenne a hibás címekre eső táblák lefoglalása.

A BadRAM cím hibaadatait lehet bővíteni a lapokkal, megelőzve a hibás címek felülírását.

Eszményi esetben a memóriaveszteséget nullára csökkenthetnénk. A gyakorlatban viszont, a BadRAM így sem végezne jobb munkát, mivel egy átlagos rendszer nem használja egyszerre az összes memórialapot. Éppen ezért kétlem, hogy egyelőre a dolog megérné a processzorteljesítmény és a kódolási képesség ezzel járó növekedését.

Reménykedem abban is, hogy a memóriaforgalmazó vállalatok is magukénak érzik ezt az ötletet, és piacra dobják a hibás (olcsó) memóriamodulokat is. Örülnék, ha lenne modulok „silányságát” jellemző rendszer is, melyet a BadRAM rendszermag leírásában is közölhetnék.

Mindezek után nincs más hátra, mint hogy linuxos barátaimnak sok szerencsét kívánjak a hibás memóriák kereséséhez. Talán egy kevésbé érett operációs rendszer néhány vakbuzgó felhasználójától kapunk majd néhányat...

Rick van Rein
PhD hallgató a Twentei Egyetem informatikai szakán. Közelről figyeli a GNU közösség fejlesztéseit, ebből merít ösztönzést mindennapi munkájához. A BadRAM az ő ajándéka a GNU közösség számára.

Kapcsolódó címek

A BadRAM jelenleg nem a rendszermag része, de megtalálható a
➲ http://home.zonnet.nl/ és a
➲ vanrein/badram/ címen. *(1. kép)*

A BadRAM részletes alkalmazási példái, a rendszermagba fordítás után megtalálhatók a Documentation/badram.txt/ könyvtárban.

A rendszer mérési adatai
➲ http://home.zonnet.nl/vanrein/benchmarks.html.
A Memtest86 program megtalálható a
➲ http://reality.sgi.com/cbrady_denver/memtest86 címen. (*2. kép)*

A táblafoglaló leírása megtalálható *Jeff Bonwick* The Slab Allocator: An Object-Oriented Kernel Memory Allocator című írásában, mely az USENIX kiadásában jelent meg 1994-ben.