Linux/ppc64 IBM pSeries-en

Vincze Dávid
G3S7T

Készült a „Linux rendszergazdai ismeretek” c. tárgy keretein belül

Miskolc, 2006

Bevezető

Az Általános Informatika Tanszék 2001 óta rendelkezik egy IBM pSeries 620 7025-6F1 típusú szerverrel, amin üzembe helyezése óta AiX 4.3.3 futott, feladata többek között webszerver, DNS, FTP, levelezés, levelező lista üzemeltetés és SSH volt, illetve a tanszék felhasználói, oktatók, dolgozók, hallgatók futtathatták rajta saját programjaikat. Több évnyi használat után tapasztaltuk, hogy sok probléma van vele, mind használhatóságban, mind teljesítményben. Amikor megszűnt a támogatás a 4.3.3-as verziójú AiX-hoz, akkor döntöttünk úgy, hogy nem AiX 5.2-re frissítünk tovább, hanem megpróbálkozunk Linux-ot varázsolni rá, gondoltuk így sokkal több feladatot tud majd ellátni, mindezt megbízhatóbban és gyorsabban.

Ekkor a Linux kernel már elég jól támogatta a powerpc64-es architektúrát. A támogatást az architektúrához egy kis csapat fejlesztette és fejleszti még most is az IBM-nél, igaz mások nem is tudtak volna segédkezni nekik, mivel az IBM dokumentumai a gépek részletes működéséről nem publikusak. Manapság már csak az újabb pSeries (régi nevén RS/6000) és iSeries (régi nevén AS/400) gépekhez fejlesztik a kernelt, de a POWER3 és RS64 processzorokon alapuló gépekhez is teljes körű a támogatás. Igaz a teszteléshez használt gép nem a legújabb modell, de a megszerzett tapasztalatokat és információkat hasznosíthatjuk az legújabb generációs POWER5+, PowerPC G5 és Cell processzor alapú mostani csúcsmodelleken.

Komplett Linux disztribúciók is léteznek már PowerPC-re, igaz ezek többnyire nem 64, hanem 32-bitesek, de mint azt látni fogjuk ez nem igazán befolyásolja a dolgokat.

A dolgozatban bemutatásra kerül az architektúra, annak sajátosságai, és, hogy hogyan sikerült egy működő Linux kernelt és az ehhez szükséges toolchain-t készíteni, azt használni, és egy működőképes Linux disztribúciót telepíteni, illetve tippeket, tanácsokat a rendszer használatához.

1. Linux és pSeries

avagy "Farewell, AiX!"

Aki ismeri az IBM RS/6000-es (napjainkban már pSeries néven futó) számítógépcsaládját, valószínűleg a natív operációs rendszere, az AiX felől is hallott már. Az AiX nem rossz operációs rendszer, de meggyűlhet vele a bajunk, ha más rendszerekkel való együttműködésre szeretnénk rábírni, illetve Linux-ról ismert open source alkalmazásokat szeretnénk rajta használni. Leginkább ez a régebbi verzióknál jön elő, a legújabb verziói már szerencsére egyre jobban a Linux felé húznak (AiX 5L - L, mint Linux Affinity), de még ezekkel is akadhatnak problémák. Másfelől az AiX pénzbe kerül, illetve nem nyílt forrású, és különben is, a Linux szimpatikusabb. Óriási előny még az is, hogy Linux-szal lehetőségünk nyílik szinte bármilyen PCI-os kiegészítő hardver használatára is, nincs szükség a méregdrága IBM termékekre, például egy AiX által is támogatott rezes gigabit IBM hálózati interfésznek $1440 a piaci ára, ezzel szemben egy ugyanolyan, vagy még talán jobb minőségű PC szerverekbe szánt felső kategóriás gigabites hálózati kártya ennek kb. tizedébe kerül. Egy mezei 10/100 Mbit-es IBM Ethernet kártya $445.50, egy PC-be való minőségi kártya $50 körül mozog.

Természetesen egy ideje létezik ppc64 kernelre alapuló disztribúció ezekre a gépekre is: *SuSE Enterprise Server 8 for IBM iSeries/pSeries, Red Hat Enterprise Linux Advanced Server 3, Turbolinux Enterprise Server 8*. Mivel ezek sem ingyenesek, ezekhez nem volt hozzáférésem, ezért kénytelen voltam saját magam barkácsolni valamit. Elegendő dokumentáció hiányában sok minden az itt leírtak közül kísérletezgetések eredménye, illetve a kernel forráskód behatóbb tanulmányozása alapján jött létre.

1.1. Az első gép

Típusjelzése pSeries 620, pontosabban 7025-6F1, egy jól megtermett szép fekete kocka. Paraméterei:

- 2 db RS64-III 'Pulsar' processzor
  - 128 KB L1 instruction cache, 128 KB L1 data cache
  - On-chip L2 cache jegyzék, amely maximum 8 MB off-chip L2 cache-t támogat
  - 14.4 GB/s L2 cache átviteli sebesség
  - 32-byte (256 bit) széles on-chip buszok
  - 450 MHz működési frekvencia
  - 4 utas szuperskalár
  - 5 szint mélységű pipeline
  - 22 Watt maximális fogyasztás
- CHRP (Common Hardware Reference Platform) architektúra

- szerviz processzor, oppanel (status LCD es különböző status LED-ek)
- 10 db 64-bites hot-plug PCI slot (6db 66MHz/3.3V, 4db 33MHz/5V)
- 2x 575W tápegység, egy kivezetéssel
- 1 GB memória (max. 32 GB)
- 2x 36.4G SCSI diszk (max. 12 hot-swap disk hely)
- Beépített AMD PCnet hálózati kártya (100 Mbit full duplex Ethernet)
- GXT130P (Matrox G200) framebuffer, max. 256 szín

1.2. Első lépések a Linux felé

1.2.1. Cross toolchain

Valahogyan egy kernel image-t kell készítenünk először, ehhez szükség van egy ppc64, illetve egy sima ppc toolchain-re. Történelmi okok miatt bemutatok egy mára már elavultnak mondható módszert.

1.2.2. Binutils

Első lépésben szerezzük be a forrást, penguinppc64.org javaslata szerint használjuk a legutóbbi cvs snapshotot, amit a következő címen lelhetünk: *ftp://sources.redhat.com/pub/binutils/snapshots/*. Teszteltem több verzióval is, egyikkel sem volt probléma.

32-bites binutils-ra is szükség lesz, így célszerű egy bi-arch binutils-t fordítani PPC, illetve PPC64 támogatással.

```
./configure --target=powerpc64-linux --enable-targets=powerpc-linux \
            --disable-nls --disable-shared -prefix=$HOME/ppc64tools
make
make install
```

A `make install`-nak van egy kisebb szépséghibája, mégpedig az, hogy a kész binárisokra nem csinálja meg a symbolic linkeket a többi architektúrákhoz, jelen esetben nem csinál powerpc-linux- kezdetű symlinkeket a powerpc64-linux- kezdetű fileokra. Ezt az hiányosságot az alábbi utasítássorozattal pótolhatjuk, vagy akár forgathatunk két lépésben is az --enable-targets kapcsolót elhagyván, egyszer --target=powerpc64-linux, másszor --target=powerpc-linux opcióval.

```
cd $HOME/ppc64tools
mkdir -p powerpc-linux/bin
cd powerpc64-linux/bin
for i in `/bin/ls`
do
  ln -s $HOME/ppc64tools/powerpc64-linux/bin/${i} \
        $HOME/ppc64tools/powerpc-linux/bin/${i}
done
cd $HOME/ppc64tools/powerpc-linux/bin/
```

```
rm -f as ld
cd $HOME/ppc64tools/bin
for i in `/bin/ls`
do
  ln -s $i powerpc-linux-`echo $i |cut -d"-" -f3`
done
rm -f powerpc-linux-ld powerpc-linux-as
```

Az utolsó paranccsal letörölt két programot természetesen pótolnunk kell. Az assemblernek, illetve linkernek paramétereket kell átpasszolnunk, hogy 32 bites binárisokat hozzanak létre, a többi a saját nevéből (0. argumentum) tudja, hogy 32 bitesként kell viselkednie.

powerpc-linux-ld:

```
#!/bin/sh
$HOME/ppc64tools/bin/powerpc64-linux-ld -m elf32ppc $*
```

powerpc-linux-as:

```
#!/bin/sh
$HOME/ppc64tools/bin/powerpc64-linux-as -a32 $*
```

Ne felejtsük el ezeket is belinkelni a powerpc-linux/bin jegyzékbe, illetve futtathatóvá tenni őket:

```
cd $HOME/ppc64tools/powerpc-linux/bin
ln -s $HOME/ppc64tools/bin/powerpc-linux-as as
ln -s $HOME/ppc64tools/bin/powerpc-linux-ld ld
chmod +x $HOME/ppc64tools/bin/powerpc-linux-as \
         $HOME/ppc64tools/bin/powerpc-linux-ld
```

Természetesen ha már rendelkezünk egy meglévő PPC-n futó Linux disztribúcióval, akkor könnyebb dolgunk van, mivel ebben az esetben 32-bitessel már rendelkezünk, így elég 64-bitest fordítanunk, az `--enable-targets=powerpc-linux` opciót el is hagyhatjuk, és a symlinkekkel, illetve wrapper scriptekkel sem kell bajlódnunk.

1.2.3. GCC

Következő lépésben essünk neki a GCC-nek, szerencsére ezzel sem lesz különösebb probléma. A *http://gcc.gnu.org/gcc-3.2/* címről töltsünk le egy vanilla 3.2.1-es verziójú GCC-t, és a hozzá való patch-et innen: *ftp://ftp.linuxppc64.org/pub/people/amodra/gcc-3.2/gcc-3.2.1-ppc64.diff.gz* A configure scriptnek megadott --prefix opció értéke ugyanaz legyen, mint amit a binutils-nak adtunk meg prefixet. Kicsomagolás és patchelés után adjuk hozzá a PATH környezeti változóhoz még azt az elérési útvonalat is, ahová a binutils-t tettük, mivel az autoconf script hajlamos nem megtalálni néhány binutils alkalmazást (pl. nm, objdump).

```
export PATH=${PATH}:$HOME/ppc64tools/bin
./configure --target=powerpc64-linux -enable-languages=c \
            --disable-shared --disable-nls --disable-threads \
            --enable-__cxa_atexit -prefix=$HOME/ppc64tools
make
make install
```

A betöltő, illetve kernel kitömörítő kód lefordításához kell egy 32-bites is, abból a 3.3.1-es verziójú lesz a megfelelő. Természetesen, ahogy a binutils-nál is említettem, ha ezt ppc platformon csináljuk és rendelkezésünkre áll már egy 32 bites gcc, akkor ezt a lépést elhagyhatjuk. Egy apró patch-et is applikálnunk kell a fordítás előtt: *http://penguinppc.org/embedded/cross-compiling/gcc-3.3.1-crossppc.diff*

Végül, a következőképpen fordíthatjuk le:

```
./configure --target=powerpc-linux --disable-shared -with-newlib \
            --enable-languages=c --disable-threads \
            --prefix=$HOME/ppc64tools
make
make install
```

Ellenőrizni a `gcc -dumpmachine` paranccsal tudjuk.

1.3. Cross-toolchain egy új módszerrel

Forradalmi jelentőségű a CrossTool nevű segédprogram megjelenése. Ez egy szkriptgyűjtemény, ami automatikusan elkészíti a választott architektúrára a cross-toolchain-t, sőt szükség esetén még a forráskódokat is letöltögeti és az esetlegesen hozzájuk való patch-eket is. Ezzel az eszközzel gyerekjátékká válik a cross-toolchain-ek készítése bármelyik ismertebb architektúrára. A mi esetünkben van egy jól működő példa szkript is mellékelve: demo-ppc970.sh, ezt futtatva a számítógép teljesítményétől függően néhány órán belül egy használható toolchain-t kapunk.

2. Kernel

A ppc64 Linux kernel a már meglévő 32 bites ppc kernelen alapszik, nagyfokú eltérések a két kernel között többek közt a memória menedzsment alrendszerben vannak, mint például egy új memória térkép, nem folytonos fizikai memória támogatás, ezenfelül LPAR támogatás, és különböző hardware specifikus driverek, pl. RTAS, NVRAM, real-time clock, és két platform specifikus vezérlő blokk jelent meg a kernelben: NACA (Node Address Communications Area / Node Architected Control Array), PACA (Processor Address Communications Area / Processor Architected Control Array). Ezek az iSeries-eken használt OS/400 rendszerekre származtathatóak vissza.

Az első, a NACA az egész rendszerre kiterjedő információk tárolására szolgál, mint például a gépben/logikai partícióban lévő processzorok száma, a kernel számára elérhető valós memória mérete, adatok a cache-ről, az interrupt controller típusa, mutató a

PACA-kat tartalmazó tömbre. Tartalmaz még ezeken felül egy megszakítás vektor táblát, azoknak a program rutinoknak a címeivel, amelyeket kivételek bekövetkezésekor a hypervisor meghívhat.

A másik, a PACA processzoronkénti információkat tartalmaz, így logikusan minden logikai processzorhoz tartozik egy PACA. Minden processzor SPRG1 (Special Purpose Register 1 - Különleges célú regiszter 1) regisztere mindig az aktuális processzorhoz tartozó PACA virtuális címére mutat. Tartalmazza többek között a CPU állapotát, logikai, illetve fizikai számát, verzióját, egy RTAS struktúrát, és a kernel TOC címét. Legnagyobb haszna a megszakítások feldolgozásakor van, ugyanis ide mentődnek le különböző adatok az interrupt feldolgozása előtt.

Régebben a penguinppc64.org oldalon egy elég régi, 2.4.21-es kernelhez készült patch volt található. Ha csak nem valamilyen különleges oknál fogva erre a kernelre volt szükségünk, akkor célszerű volt az akkori legfrissebb kernel forrást a ppc64 bitkeeper repository-ból leszedni. Mára már odáig jutott a fejlesztés, hogy az alap Linux kernel is megfelelően támogatja a ppc64-es architektúrát.

Még mielőtt reflexből írnánk, hogy make menuconfig, vessünk egy gyors pillantást néhány Makefile-ra. Először nézzük azt, ami a kernel forrás gyökerében van. Ebben módosítanunk kell néhány dolgot, a cross-compile-hoz, elsőként meg kell adnunk milyen architektúrára szeretnénk fordítani: az `ARCH:=` kezdetű sort módosítsuk egyszerűen erre: `ARCH:=ppc64`. Egy picit lentebb találunk még egy változtatásra szoruló változót, `CROSS_COMPILE`-hoz adjuk meg a frissen fordított toolchain előtagját: `CROSS_COMPILE = powerpc64-linux-`. Ugyanezt a hatást érhetjük el persze a következő paranccsal is:

```
make ARCH=ppc64 CROSS_COMPILE=powerpc64-linux-,
```

de az előbbi megoldással biztosan nem felejti el az ember a paramétereket egy esetleges újrafordításnál.

Az arch/ppc64/boot directory-ban nézzük meg a Makefile-t. Itt a betöltő kód fordításához adjuk meg a 32 bites fordítót (ha ppc-n fordítunk nem kell!):

```
CROSS32_COMPILE = powerpc-linux-
```

Az arch/ppc64/configs directory-ban található egy pSeries_defconfig nevű file, célszerű ezt használni, esetleg néhány dolgot érdemes módosítani benne, erre rögvest ki is térünk.

Ha a fentiekkel megvolnánk, akkor mehet a kernel fordítás a megszokott módon, config, menuconfig, xconfig, ahogy jólesik. Személy szerint nekem a menuconfig szimpatikus, így ezt fogom bemutatni. Tehát:

```
make menuconfig
```

Természetesen egyelőre egy olyan kernelre van szükség, amit csak az installálás erejéig fogunk használni. Így pár dolgot ki lehet belőle venni, ezzel is csökkentve a hibalehetőségek számát, példának okáért: *Loadable Module support*, hálózati opciók, filerendszerek között is lehet szortírozni, 2.4-es sorozatból való kernel esetén a *Kernel*

hacking alatt viszont a *"Magic SysRq key"*-t véletlenül se vegyük ki, mivel valamilyen oknál fogva, a *pSeries Hypervisor Console* (Hardware Management Console) függ tőle, és arra pedig szükség lehet ha LPAR (Logical PARtitioning) rendszerünk van.

És most nézzük meg részletesebben mi micsoda a 2.4-es kernelben:

```
Platform support -->
  (pSeries) Machine Type
```

Ez egyértelmű, az al-architektúrát választhatjuk ki: pSeries (ex RS/6000), illetve iSeries (ex AS/400)

```
  [ ] VMX ( AltiVec ) Support (NEW)
```

Támogatás egyes PowerPC processzorokban lévő SIMD utasításkészlethez (hasonló mint az MMX/SSE x86-os architektúrán), nyugodtan kikapcsolhatjuk ugyanis ezekben a processzorokban nincs jelen ez a feature.

```
  [*] Symmetric multi-processing support
```

Manapság az egy processzoros RS6K ritka, mint a fehér holló, de ha épp olyan gép az áldozat, akkor is engedélyezhetjük nyugodt szívvel, mivel SMP támogatással is remekül fut a kernel egy processzorral, sőt érdekességképpen említem, hogy a 2.4.21-es kernel, amihez van a külön ppc64 patch, SMP támogatás nélkül le sem fordul, igaz ezt a hibát azóta már az újabb verziókban javították.

```
  [ ]   Distribute interrupts on all CPUs by default
```

Ezeken a gépeken jelenleg nem támogatott, ezzel a kernel verzióval.

```
  (2) Maximum number of CPUs (2-32) (NEW)
```

Célszerű ezt annyira állítani ahány processzor van a gépben, spórolhatunk egy kevés memóriát.

```
  [ ]   Hardware multithreading
```

Ezt a feature-t az RS64-IV 'SStar' előtti processzorok nem tartalmazzák, így ez a mi esetünkben lényegtelen, bár léteznek olyan pSeries 620-asok amikbe már RS64-IV processzor került. Hasonló mint az Intel Xeon processzoraiban bevezetett Hyper-Threading technológia. Dióhéjban annyi a dolog lényege, hogy egy fizikai processzor egyszerre több task/thread állapotait tárolja, és amikor egy thread megakad egy hosszabb késleltetésű esemény miatt (például egy Level 2 cache miss), történik egy hardveres kontextus váltás, és így egy másik thread igénybe veheti a processzor erőforrásait. Mivel

ezek a processzorok ennek engedélyezésével logikailag két processzornak látszódnak, az előző opciónál megadott értéket is növeljük a duplájára. Érdekes, hogy a mi gépünkben lévő RS64-III processzorokat is valamiért HMT-t támogatónak ismerte fel a kernel, és négy processzort vélt látni.

```
[*] MsChunks Physical to Absolute address translation support
```

Tévedés ne essék, az ms most nem Microsoft-ot takar. Az MsChunks rövidítés "Main Store Chunks"-ot jelent, ami a címtranszformációk gyorsítására szolgál. Eme opció bekapcsolásával engedélyezzük a kernelnek, hogy nem folyamatos fizikai memóriával rendelkező gépekre optimizálja magát. LPAR rendszereken ez elengedhetetlen a dinamikus particionálás miatt, LPAR nélküli pSeries-en nem szükséges.

```
[*] Firmware flash interface
```

Ha engedélyezzük akkor Linux alól is tudunk majd firmware-t frissíteni. Később szó lesz még erről.

```
[*] Scanlog Dump interface
```

Amikor a ppc64 hardver meghibásodik és a szerviz processzor fel is ismeri a hibát, elmenti a rendszer belső állapotát az NVRAM-ba (ez a scanlog dump). Következő bootoláskor ezen driveren keresztül érhetjük el ezeket az adatokat, mégpedig a /proc/ppc64/scan-log-dump speciális fileon keresztül. Csak szekvenciális olvasást támogat a driver, illetve három kulcsszóra figyel ha írunk bele, ezek a következőek: reset, debugon, debugoff. Feltételezem, hogy ezek nem szorulnak további magyarázatra.

```
[*] Support for RTAS (RunTime Abstraction Services) in /proc
```

Az RTAS egy firmware interface az operációs rendszer és a ppc64 hardware között. Hasznos dolog ha /proc-on keresztül el tudjuk érni, engedélyezzük. A későbbiekben lesz még róla szó bővebben.

```
[ ] RTAS Error Inject
```

Ha ezt engedélyezzük, akkor lehetőségünk nyílik manuálisan hardware hibákat generálni a /proc/ppc64/errinjct bejegyzésen keresztül. A hibakezelési kódok tesztelésére szolgál csak, éles rendszeren ne használjuk.

```
[ ] Shared kernel/user space addressing
```

Ennek segítségével user space-ből is elérhetjük a kernel memória részeit. Személy szerint erre nekem nem volt szükségem.

```
  [*] LPAR Configuration Data (NEW)
```

Ezt akkor is engedélyezhetjük ha nem tud LPAR-t a masinánk, ilyenkor egy logikai partició van. A /proc/ppc64/lparcfg fileban jelenik meg pár információ a logikai partici(ók)ról.

```
General setup ->
[*] Support for frame buffer devices
```

Ha nincs framebuffer kártya a gépünkben, akkor természetesen erre nincs szükségünk, ehelyett majd soros konzolt kell használnunk.

```
[*] Support for Open Firmware device tree in /proc
```

Ezt engedélyezve az OF promptból is elérhető device-tree másolatát érhetjük el egy /proc bejegyzésen keresztül. A device-tree nem más, mint egy hierarchikus adatstruktúra, ami a rendszer hardvereit, illetve konfigurációs lehetőségeit írja le. Későbbiekben még lesz róla szó.

```
Block devices ->
<*> RAM disk support
(8192)   Default RAM disk size
```

Célszerű nagyobbra venni, mint az alap 4 Mbyte, de ezt akár később kernel paraméterrel (ramdisk_size) is felülbírálhatjuk bootoláskor.

```
[*]   Initial RAM disk (initrd) support
```

Initrd támogatásra szükségünk lesz természetesen, ebben lesz majd valamelyik disztribúció telepítőjének a ramdisk-je.

```
SCSI support  --->
SCSI low-level drivers  --->
<*> SYM53C8XX SCSI support
```

Általában ilyenek, vagy ezzel kompatibilisek a beépített SCSI vezérlők ezekben a gépekben.

```
Network device support  --->
Ethernet (10 or 100Mbit)  --->
[*]   EISA, VLB, PCI and on board controllers
<*>     AMD PCnet32 PCI support
```

Ez a driver való a beépített hálózati interfész(ek)hez.

```
< >     EtherExpressPro/100 support
```

POWER4-es konfigurációkba már ez a vezérlő került.

```
Ethernet (1000 Mbit)  --->
<*> Alteon AceNIC/3Com 3C985/NetGear GA620 Gigabit support

[*]   Omit support for old Tigon I based AceNICs
```

Nyugodtan engedélyezhetjük, mivel ezek az "IBM Gigabit PCI-SX" kártyák és valószínűleg a többi IBM által feliratozott kártya is Tigon II, így spórolhatunk néhány kilobyteot.

```
< > IBM PowerPC Virtual Ethernet driver support (NEW)
```

Ez is csak LPAR rendszereknél szükséges, a partíciók közti Ethernet interface megosztáshoz, így nyugodtan nemmel felelhetünk.

Token Ring drivereket vegyük ki ha nem tartunk rájuk igényt, mivel benne vannak ebben a defconfig-ban.

```
Console drivers  --->

Frame-buffer support  --->
[*] Support for frame buffer devices (EXPERIMENTAL)
```

Ha van valamiféle videóvezérlő a gépben és szeretnénk is használni, akkor itt igennel feleljünk.

```
[*]   Open Firmware frame buffer device support
```

Elméletileg ez a driver képes az OF-en keresztül kezelni a videóvezérlőt, és így használható framebuffer-ként, gyakorlatilag azt tapasztaltam, hogy ez nem működik ezen a gépen, sőt még hibákat is generált, sok kellemetlenséget okozva, de valószínűleg ez is gépenként változik.

```
<*>   Matrox acceleration (EXPERIMENTAL)
[*]     G100/G200/G400/G450/G550 support
```

A GXT130P kártyákhoz ez a driver való, valószínűleg a GXT135P-k is ezzel működnek, de nem volt ilyen kártya a birtokomban, így nem tudtam kipróbálni.

```
Character devices  --->
```

```
<*> Standard/generic (8250/16550 and compatible UARTs) serial support
[*]   Support for console on serial port
```

Jól jöhet, ha valami gond lenne a framebuffer-rel (illetve ha nincs framebuffer kártya a gépben), paraméternek kell majd megadnunk bootoláskor, hogy melyik portot/ portokat használja console-nak, pl. console=ttyS0,9600 console=ttyS1.

```
[ ] pSeries Hypervisor Virtual Console support
```

Erre csak LPAR rendszerek esetén van szükség, ilyenkor ennek a drivernek a közreműködésével minden logikai partíciónak lesz egy konzolja, amit a HMC-n (Hardware Management Console) keresztül lehet elérni.

```
< > /dev/nvram support
< > Enhanced Real Time Clock Support
```

Őket nem szükséges külön engedélyezni, ugyanis alapból benne vannak a ppc64 kernelben.

Évekig a 2.6-os sorozatú kernelekben sem volt ez máshogy, de nemrégiben a ppc64 architektúrát elkezdték egybeolvasztani a powerpc architektúrával. Ez a lépés, és az évek jó néhány változást hoztak magukkal. Ezeket az alábbiakban láthatjuk:

```
[*] 64-bit kernel
```

Már a gyökér menüben szembetalálkoztunk a választási lehetőséggel, hogy 32 vagy 64-bites kernel legyen-e.

```
Platform support  --->
      Machine type (Generic desktop/server/laptop)  --->
            (X) Generic desktop/server/laptop
            ( ) IBM Legacy iSeries
      [*]   IBM pSeries & new (POWER5-based) iSeries
      [ ]   Cell Broadband Processor Architecture
```

Az alrendszer kiválasztása. Az iSeries és Cell is POWER alapú, de eltérnek a pSeries architektúrától, ezért a konfigurációban is meg kell őket különböztetni.

```
Kernel options  --->
      [*] Support for enabling/disabling CPUs
```

Menetközbeni processzor engedélyezés és letiltás támogatása, hotplug CPU-khoz, illetve így lehet tiltani az SMT-t is, ugyanis SMT-s gépen minden páratlan számú processzor a második szálat jelenti. Az SMT tiltása ritka esetekben hasznos lehet.

```
      [*] Distribute interrupts on all CPUs by default
```

A megszakítások szétosztása a processzorokon. Alaphelyzetben a boot processzor kap meg és dolgoz fel minden megszakítást. Ha bekapcsoljuk, akkor bizonyos időközönként váltogatni (round-robin „algoritmussal”) fogja az operációs rendszer, hogy melyik processzor kapja és dolgozza fel az interruptokat.

```
      [ ] Support for shared-processor logical partitions
```

Mikroparticionálishoz szükséges, amikor is nem egy egész, hanem a processzor tört részét osztjuk ki egy LPAR-hoz.

```
Bus options  --->
      <*> Support for PCI Hotplug (EXPERIMENTAL)
      <*>   RPA PCI Hotplug driver
      <*>     RPA Dynamic Logical Partitioning for I/O slots
```

Ha szeretnénk használni a hardver PCI hotplug képesség, akkor ezt feltétlenül engedélyezni kell. Természetesen ez önmagában még nem elég a menetközbeni PCI eszköz hozzáadásához, vagy eltávolításához. Ez csak egy interfészt biztosít az userspace-ben futó menedzselő programoknak.

```
Device Drivers  --->
      Character devices  --->
            <*> IBM Hypervisor Virtual Console Server support
```

Az újabb LPAR-t tudó gépeken a soros portot nem lehetséges közvetlenül elérni, a firmware multiplexeli az operációs rendszerek számára. Ez ahhoz a multiplexelt interfészhez egy driver. Fontos megjegyezni, hogy ezek a soros portok nem a szabvány /dev bejegyzéseken keresztül lesznek elérhetőek, hanem hvsiX neveken (major 226, minor 128-tól).

```
      SCSI low-level drivers  --->
            <*> IBM Power Linux RAID adapter support
```

Az OpenPower gépekben ilyen típusú SCSI vezérlő van (ipr).

Újabb keletű 2.6-os kernelnél, crosstool-al készített toolchain-nel a következőképpen kell paraméterezni a make-et.

```
make ARCH=powerpc CROSS_COMPILE=powerpc64-unknown-linux-gnu-
```

A fordítást még most se kezdjük el, ehelyett szerezzünk be egy initrd image-t. Ha Debian-t szeretnénk, akkor könnyeden hozzájuthatunk egyhez:
http://ftp.debian.org/debian/dists/sarge/main/installer-powerpc/current/images/power3/cdrom/initrd.gz

Illetve újabb gépekre a power4 image való, bár nem igazán van különbség a kettő között:

http://ftp.debian.org/debian/dists/sarge/main/installer-powerpc/current/images/power4/cdrom/initrd.gz

Ezek 32-bites ppc kernelekhez lettek készítve, de a célnak tökéletesen megfelelnek, csak hozzá kell fűzni a zImage-hez. Ezt a következőképpen tehetjük meg, az előbb emlegetett /arch/ppc64/boot (vagy /arch/powerpc/boot) directory-ban lévő Makefile ramdisk.image.gz néven keresi az initrd-t, tehát egyszerűen csak nevezzük át a initrd.gz-t ramdisk.image.gz-re, és helyezzük az /arch/ppc64/boot-ba (vagy /arch/powerpc/boot -ba). Ezek után egyszerűen:

`make dep && make zImage.initrd` (2.6-os kernelnél a *dep* nem szükséges)

Ekkor szükségünk lesz még a sima powerpc Sarge telepítő CD-re, ilyenkor miután bebootoltuk az image-t, vegyük ki a kernelt tartalmazó CD-t, és cseréljük ki erre, amikor a telepítő keresni kezdi. Természetesen összehozhatjuk egy lemezzel is, ekkor az eredeti CD image-t kell módosítani a későbbiekben leírt instrukciók alapján. Vagy választhatjuk a netboot-os initrd-t, ha nincs CD olvasó a gépben: *http://ftp.debian.org/debian/dists/sarge/main/installer-powerpc/current/images/power4/netboot/initrd.gz*

Személy szerint a Sarge-t ajánlanám, többek között azért is, mert az IBM Service Aids for Hardware Diagnostics (lásd még a későbbiekben) Linuxos verziójának libstdc++5-re van szüksége, illetve a Woody-s csomagból feltett rpm ezeknek az rpm-eknek a kibontásakor illegal instruction-el szállt el, és még a Woody-s setserial is hibás, OOPS-ot generál, és ha a setserial minden bootkor lefut, akkor okozhat némi kellemetlenséget.

Fontos, hogy ha a Sarge mellett döntünk, akkor a kernelben szükség lesz Compressed ROM file system, /dev file system és Virtual memory file system támogatásra, mivel az initrd egy cramfs image, illetve a telepítő devfs-t es tmpfs-t használ.

Nos, minden elkészült, /arch/powerpc/boot/zImage.initrd néven ott mosolyog a kernelünk az initrd-vel együtt.

3. Boot

3.1. BOOTP/TFTP

Elsőre a hálózatról való bootolás tűnt a legkényelmesebbnek. Szükség lesz hozzá természetesen egy másik számítógépre, azon DHCP kiszolgálóra, illetve TFTP daemon-ra.

Egy egyszerű config részlet a dhcpd-hez:

```
subnet 10.0.0.0 netmask 255.255.0.0 {
        next-server 10.0.0.1;

        host pseries620 {
          hardware ethernet 00:06:29:B9:20:EB;
          fixed-address 10.0.6.20;
          filename "zImage";
        }
}
```

Ebben az esetben 00:06:29:B9:20:EB MAC című hálókártyáról ha jön egy request, akkor ő a 10.0.6.20-as IP címet fogja megkapni.

Egyes források szerint, ha menet közben a firmware kap egy bármilyen nem TFTP csomagot, megáll a TFTP transfer, ez vonatkozik az ARP-re is, így előnyös lehet ha a gépen amiről bootolni fogjuk az image-t, beledrótozzuk az ARP táblájába a 6F1-es MAC címét, így nem fog ARP request-eket küldeni neki. Ezt a következőképpen tehetjük meg:

```
arp -s 10.0.6.20 00:06:29:B9:20:EB
```

Saját tapasztalat viszont az volt, hogy nem szakad meg a TFTP, hanem éppenséggel el sem indul, mivel a firmware nem válaszol az ARP kérésekre. Tehát ezzel mindenképpen jól járunk, még jobb ha csak simán egy cross kábellel van összekötve a két gép.

Mikor a gép bootol csaljuk elő az OpenFirmware (továbbiakban OF) prompt-ot. A 8-as (illetve F8 is néhány modellen) billentyű lenyomásával hívhatjuk elő, miután az OF inicializálta a keyboard-ot. Soros terminálról is megtehetjük természetesen. Miután megkaptuk a promptot, írjuk be:

```
boot net:kliens_ip,image_neve,kiszolgalo_ip
```

Ha nem adunk meg image nevet, akkor a dhcpd adja meg, persze csak ha annak a konfigurációs filejában megadtunk egy filenevet. Például:

```
boot net:10.0.6.20,zImage,10.0.0.1
```

Sajnos az első néhány száz csomag után mindig megállt az átvitel. Egy próba erejéig 10 Mbit-re lett állítva a link, de úgy is sikertelen volt. Nincs mit tenni, más módszert kell választani.

3.2. FLOPPY

Az OpenFirmware ismeri a FAT12-t, így nem kell semmi speciális dolog, elég ha szimplán rámásolja az ember a kernel image-t egy floppy lemezre, az egyetlen baj, hogy a lefordított kernel közel sem fér rá egy 1.44-es lemezre, főleg nem initrd-vel együtt. Viszont létezik egy kernel, ami hivatalosan CHRP PowerPC-hez van debian.org-on, ott ahol a fentebb említett initrd is található. Ez egy 900 kbyte körüli 32-bites kernel, de próbának megfelel. Ez bőven ráfér egy standard 1.44-es floppy lemezre. Próba szerencse, boot a következőképpen lehetséges OF prompt-ból:

```
boot floppy:,\linux.bin
```

Szépen lassan beolvasta, kitömörítette; tehát az a kódrész működik, de sajnos csak eddig jut el. Nincs más választás, újabb kernelforgatás, hogy ráférjen egy lemezre. Nem könnyű feladat, mivel még bzip2-t sem lehet használni, csakis gzip-et, mivel csak ehhez van loader kód. Minél minimálisabbra kell konfigurálni, filesystem-ek között lehet szortírozni, azok sokat foglalnak. Az arch/ppc64 jegyzék alatt lévő Makefile-ból kivehetjük az mtraceback=full gcc paramétert, ez méretben jelent elég sokat, a fordítási időt nem befolyásolja. Initrd-t pedig hanyagoljuk, majd rátesszük egy másik lemezre, ezért csak simán `make zImage`-el fordítsuk a kernelt. Közben volt kéznél egy soros terminál, ami rákerült a kísérleti alany 1-es serial port-jára (hátulról nézve a jobb alsó port). Ezután boot újra hajlékonylemezről console=ttyS0,9600 argumentummal: betölti, kitömöríti, elindítja, terminálra írja a kernel message-ket, és közben az oppanel LCD-re is írja, hogy éppen mit csinál. Megjelenik a Linux-ppc64, felirat, de miután már látta az egyik processzort, és a PCI scan-nel is végzett, egy hibakód jött a szerviz processzortól. A hibakód szerint firmware upgrade-et szeretne. De, mint később kiderült csak a framebuffer-rel volt probléma. A firmware upgrade-t mindenesetre érdemes megtenni, csak javíthat a helyzeten, lásd később.

3.3. CD-ROM

A sorozatos csalódások után fény derült arra a tényre, hogy az OF ismeri a cdfs-t is, tehát nem kell a floppykkal trükközni, lehet akármekkora a kernel, sőt még az initrd-t is könnyedén hozzá lehet csatolni! A következő paranccsal lehet megkezdeni a CD-ről való bootolást:

```
boot cdrom:particio,\kernel parameterek
```

Az AiX install CD-k behatóbb tanulmányozása után megtudtam, hogyan lehet automatizálni a CD-ről való bootolást. Szükség van a CD-n egy látszólag DOS kompatibilis partíciós táblára, illetve egy speciális bootinfo filera, aminek a helyét és nevét a 'boot-device' nevű OF változóban lehet megadni, aminek a default értéke

'ppc/bootinfo.txt'. A partíciós táblában a CHS-ben megadott mezőket úgy tűnik nem veszi figyelembe a firmware, elég ha csupa 1-es bittel töltjük fel (mind a négy partíciónál!), a kezdő szektor száma számít csak, illetve a méretét is meg kell adni, bár valószínűleg nem sok mindent befolyásol, típushoz pedig 0x96-ot adjuk meg (lehetséges, hogy 0x96 mint ISO9660), illetve az 55AA magic string-et se felejtsük el a szektor végéről.

AiX-ot futtató gépek diszkjeinek partíciós táblája szintén egy magic string-el kezdődik, első 4 byte: 0xC9 0xC2 0xD4 0xC1. Ez az AiX-os Logical Volume Manager-t azonosítja, elméletileg nincs rá szükség, de lehetséges, hogy néhány firmware-nél jól jön, mindenesetre ártani nem árthat. Egy példa CD első szektoráról található egy hexdump a függelékben (Függelék A).

Az említtet bootinfo.txt file egyszerű felépítésű, íme egy példa:

```
<chrp-boot>
<description>Acelvaros Linux</description>
<os-name>Linux</os-name>
<boot-script>boot &device;:\ppc\kamm2</boot-script>
</chrp-boot>
```

Igazából az egyetlen hasznos tag egyedül a <boot-script>, itt adható meg a file, hogy mit bootoljon tovább. Az eredeti AiX CD-n lévő bootinfo.txt-ben van még egy <icon>, illetve <bitmap> tag, ahol azt a névből is kitalálhatjuk egy ikont definiálhatunk a CD-hez, ugyanis egyes modellek a firmware menüjükben szép ikonokat rajzolnak a bootolható eszközök mellé, így a CD-ROM mellett ez az ikon fog megjelenni, ügyes kezű emberek készíthetnek ppc64 Tux ikont például.

Az ilyen formátumú fileokat a firmware automatikusan értelmezi, a fentebb említett módszerrel ilyen fileokat is bootolhatunk OF-ből, illetve a bootinfo.txt helyére is rakhatunk pl. egy image-t, de az nem túlzottan elegáns megoldás.

3.4. Yaboot

Minden egyes kernelhez nem lenne az igazi másik CD-t írni, bár szerencsére CD-RW-t olvas a gépben lévő drive, mégis kényelmesebb lenne, ha egy disc-en több variáció is lehetne. Erre a problémára a megoldást a yaboot szolgáltatja (http://www.penguinppc.org/projects/yaboot), ami egy bootloader PowerPC platformra.

Szerencsére van letölthető yaboot bináris, ennek a cross-fordításával nem kell időznünk, bár megtehetjük, hasznos lehet pl. debug opcióval fordítani, ha hibakeresésre kényszerülünk, vagy csak behatóbban szeretnénk tanulmányozni a bootolási folyamatot. Egy ilyen debuggolt bootolási folyamatnak a kimenete megtalálható a függelékben (Függelék B.).

Mielőtt ezt megpróbálnánk bebootolni OF promptból, futtassuk le rajta az addnote programot (ami egy speciális PowerPC headert tesz az ELF binárishoz, olyat amilyet az AiX-os bosboot segédprogram is generál), különben nem hajlandó bebootolni (papírforma szerint, gyakorlatilag igen, legalábbis a kipróbált gépeken, feltehetőleg gép és firmware függő). Egyébként a yaboot már be tudja bootolni a 64-bit ELF vmlinux-ot is. A

CD-re helyezzük el a konfigurációs fileját: /etc/yaboot.conf . A konfig hasonlít a LILO szerű bootloaderekéhez, lássunk egy egyszerű konfigurációt:

```
init-message = "Acelvaros Linux by Kamm\n\n"
timeout=30
default=fb

image=/ppc/chrp/knew
        label=fb
        append="console=ttyS0 video=matrox console=tty1"
        initrd-size=8192
        read-only

image=/ppc/chrp/knew
        label=serial
        append="console=ttyS0 nofb noinitrd"
        read-only
```

A frissen sült CD-t bootolván, yaboot panaszolja, hogy hibás/ismeretlen filerendszer van a lemezen, így a /etc/yaboot.conf-ot nem tudja olvasni.

Arra tippeltem, hogy valószínűleg a partíció típusa (0x96) miatt nem hajlandó felismerni a filerendszert. Ezért hexeditorral egy második partíciót (0x83) adtam hozzá az iso image-ben, ez egy apró 2 megabyte méretű partíció volt, amiben kizárólag az /etc/yaboot.conf file szerepelt, ezután cat-el összefűztem az eredeti iso image-t és az ext2 image-t. Így már hajlandó volt megtalálni a yaboot a konfigurációs file-ját. Később kisebb nyomozás után derült ki, hogy nem a partíciós táblában lévő típussal van a probléma, hanem a firmware egyszerűen nem ismer semmilyen iso9660 extension-t, ezért nem találja a yaboot.conf nevű file-t, mivel az a CD-n elsődlegesen "YABOOT.CON;1" néven szerepel. A probléma áthidalására az mkisofs -U kapcsolója szolgáltatja a megoldást, ami a szabványtól eltérő bejegyzéseket fog generálni a CD-re, vagy alternatív megoldásként, ha ragaszkodunk a filerendszer szabványosságához, yaboot forrásában átírhatjuk, hogy ne yaboot.conf-ot hanem pl. yaboot.cfg nevű config file-t keressen, amit a -U használata nélkül is képes megtalálni a firmware. Újabb mkisofs-ekkel szerencsére már hexeditort és számológépet sem kell használnunk az első szektorban lévő partíció manuális létrehozása miatt, mivel támogatnak egy -chrp-boot kapcsolót, ami mindezt megteszi helyettünk. Tehát a következőképpen generálhatjuk az iso image-t.

```
mkisofs -U -chrp-boot -o pseries_linux.iso ~/pseries_cd
```

Írjuk fel egy CD-re az elkészült image-t, helyezzük be, és bootoljunk be róla. Ügyeljünk rá, hogy a multiboot-ban benne legyen a CD-ROM is, és az első helyen legyen. A multiboot menüt ugyanúgy tudjuk előcsalni, mint az OF promptot, annyi különbséggel, hogy nem 8/F8-at kell lenyomni, hanem 1/F1-et. Itt egy mini menürendszer fogad minket, ahol számokkal lehet navigálni, és tudjuk megváltoztatni a bootolási eszközöket (Multiboot menüpont), illetve azok sorrendjét. Ugyanezt megtehetjük AiX alól is a

bootlist programmal, először nézzük meg mi az aktuális bootolási sorrend, ha már eleve a CD-ROM van az első helyen, nyert ügyünk van, másképpen tegyük a CD-t az első helyre, például:

```
# bootlist -m normal -o
hdisk1
# bootlist -m normal -o cd0 hdisk1
cd0
hdisk1
```

Ha ezzel is megvolnánk, várjuk meg míg bebootol a yaboot, aztán a kernel, és ha minden jól ment, akkor már egy Debian Woody/Sarge installer mosolyog vissza a képernyőről, innentől már a szokásos módon telepíthetünk. Egyetlen dologra kell odafigyelni a particionálásnál: az első számú partíciónak PReP boot (0x41) típusúnak kell lennie, mivel a firmware ilyen partíciót keres a diszkeken bootoláskor, innen próbálja meg betölteni a kernelt. Ez legyen mondjuk 4 megabyte méretű, ebben bőven elfér egy kernel.

Ha ez a partíció nem létezik, akkor Multiboot-nal nem is jelenik meg a disk, mint bootolható eszköz. A kernel betöltését ugyanúgy végzi a firmware, mintha OF promptból utasítottuk volna, tehát egyszerűen idemásolhatjuk a kernelt pl. dd-vel:

```
dd if=zImage of=/dev/sda1
```

Érdemesebb a kernel helyett yaboot-ot tenni ebbe a partícióba, mégiscsak jobb egy bootloader, mint egy fix kernel fix argumentumokkal. A yaboot a rootfs-ról olvassa be a konfigurációs file-ját, illetve az abban megadott kernel image-ket. Ismeri az Ext2/3-at, ReiserFS-t, XFS-t, illetve azokat amiket az OpenFirmware is ismer: FAT, CDFS.

Ha LPAR rendszeren tesszük mindezeket, akkor igénybe vehetjük a HMC-t (HMC = Hardware Management Console) is, soros terminál vagy framebuffer helyett, csak a megfelelő /dev bejegyzést kell létrehoznunk:

```
mknod 226 0 /dev/hvc0
```

Célszerű inittab-ból indítani rá egy getty-t, vagy ha elszántabbak vagyunk, akár már az initrd-t is módosíthatjuk, hogy HMC-re dolgozzon, így már installáláskor használhatjuk a HVC-t, igaz a kernel üzeneteket valószínűleg nem fogjuk látni, ami pedig debuggolásnál nem hátrány.

LPAR képes gépeken a soros port nem érhető el közvetlenül, hanem a szerviz processzor multiplexeli az operációs rendszerek számára. Ehhez kell a fentebb említett Hypervisor Virtual Console Server opció a kernelbe. A Hypervisor Virtual Serial Interface-nek szintén létre kell hozni /dev bejegyzéseket, ha használni szertnénk őket:

```
mknod 226 128 /dev/hvsi0
mknod 226 129 /dev/hvsi1
```

Az architektúra specifikus dolgokról elérhető információkat méretük miatt a függelékben (Függelék C.) kaptak helyet.

4. Más RS/6000 / pSeries gépek

4.1. Második gép (pSeries 640-B80)

Hasonló az előző géphez, azonos időszakból való. A processzora más, POWER3-II -es, de binárisan kompatibilisek egymással. Típusjelzése pSeries 640, pontosabban 7026-B80. Paraméterei:

- 2 db POWER3-II processzor 1 kártyán
  - 32 KB L1 instruction cache, 64 KB L1 data cache
  - 4MB L2 cache
  - 4x 512 bit széles memória busz
  - 375 MHz működési frekvencia
- CHRP (Common Hardware Reference Platform) architektúra
- szerviz processzor, oppanel (status LCD és különböző status LED-ek)
- 5 db 64-bites hot-plug PCI slot (2db 50MHz/3.3V, 2db 33MHz/5V, 1db 32-bit 33MHz/5V)
- 2x 540W tápegység, két kivezetéssel
- 2 GB ECC memória (max. 16 GB)
- 2x 18.2 G SCSI diszk (max. 4 hot-swap disk hely)
- 2 db beépített AMD PCnet hálózati kártya (100 Mbit full duplex Ethernet)
- Framebuffer kártya nélkül

Tapasztalatok szerint sokkal stabilabban fut rajta a Linux operációs rendszer, mint az előbb bemutatott gépen, stock kernelt is hiba nélkül bootol, így különálló kernel patcheket is használhatunk rajta, természetesen csak olyanokat, amik platform függetlenek. Maga a firmware is gyorsabban végez az inicializációkkal (2-3 perc).

Mivel két merevlemez van a gépben, lehetőség van egymás mellett AiX-ot és Linux-ot használni, ehhez az OF Multiboot funkcióját célszerű használni, amivel egyszerűen lehet választani rendszerindításkor, hogy melyik diszkről történjen a bootolás.

4.2. Harmadik gép (OpenPower 720)

Két generációval újabb processzorokat tartalmaz, mint ez előbbiekben bemutatott gépek.

Típusjelzése OpenPower 720, pontosabban 9124-720. Paraméterei:

- 1 db kétmagos POWER5 processzor 1 kártyán
  - 32 KB L1 instruction cache, 64 KB L1 data cache
  - 1.875 MB L2 cache, 36 MB L3 cache

    - 16 byte széles memória busz
    - 1.65 GHz működési frekvencia
- CHRP (Common Hardware Reference Platform) architektúra
- szerviz processzor, oppanel (status LCD és különböző status LED-ek)
- 5 db 64-bites hot-plug PCI-X slot (133MHz 3.3V/5V)
- 1x 540W tápegység, két kivezetéssel
- 8 GB ECC DDR memória (max. 32 GB)
- 2x 72G SCSI diszk (max. 8 hot-swap disk hely)
- 2 db beépített Intel E1000 hálózati kártya (1000 Mbit full duplex Ethernet)
- 2 db HMC port
- 2 db RIO-2 és SPCN port
- Framebuffer kártya nélkül

A marketing szerint „Linux-ra hangolt” gépről van szó. Ez lényegében igaz is, de igazából csak az a két Linux disztribúció támogatja, amiket az IBM favorizál. A többi disztribúcióval apró hackelések szükségesek, bár mind a Gentoo, mind a Debian tűrhetőnek minősíthető install CD-ket nyújt.

A szerviz processzor üzemkész állapotban elérhető a soros portokon, illetve a HMC portokon keresztül HTTPS kapcsolaton keresztül. Igazából a HMC portok is Ethernet portok. A régebbi speciális HMC portokat váltotta le, amiknek a „hálózatba kötése” is speciális eszközt igényelt, ebben az esetben viszont elég egy sima Ethernet hub vagy switch. A HMC portot nem látja az operációs rendszer, ezen keresztül a szerviz processzor mindig elérhető.

A gép LPAR képes, tehát több operációs rendszer párhuzamos futtatását teszi lehetővé. Ehhez szükség egy Hardware Management Console, ami egy másik különálló gépet jelent, célszerűen egy desktop PC-t, HMC szoftver is csak x86 architektúrára van. Viszont ennek a szoftverek külön díja van, így nem áll rendelkezésre. Lehet még a szerviz processzoron keresztül is közvetlenül, de ehhez egy firmware kiegészítés szükséges.

A diszkek felosztását egy külön LPAR-ban futó ún. Virtual I/O Server végzi, ami konfigurálható nagyságú virtuális diszket szolgáltat a többi LPAR számára. A hálózat felosztása egyszerű és nagyszerű: minden LPAR egyedi azonosítóval taggelt VLAN-ba kerül, így egyszerűen illeszthetőek be a hálózatba.

A géphez a belső diszken kívül, külső diszktömb is van, ez egy IBM TotalStorage

DS4100, két RAID vezérlővel. Minden vezérlőn 256 MB battery-backed cache van (közönséges PC2700 ECC DDR RAM), 2 Fibre Channel host port, 1 Fibre Channel expansion port, 1 Ethernet, 1 soros van. A host portokon csatlakoznak a gépek, vagy SAN eszközök, az expansion portokra lehet még további diszk fiókokat tenni, az Ethernet és a soros port menedzselésre, illetve diagnosztikára szolgál. Maximum 14 darab 7200/perc fordulatszámú 250GB vagy 400GB kapacitású SATA diszket lehet beletenni.

Minden vezérlőre 2 host csatlakoztat, tehát 4 host csatlakozása megoldott SAN switch nélkül. A mi esetünkben két node csatlakozik, így célszerű egyiket az egyik vezérlőre, másikat a másik vezérlőre kötni. A diszkekből több különálló RAID tömb alakítható ki, 0,1,3,5 és 10-es szintűek. Figyelemre méltó képessége, hogy már definiált tömböt képes menet közben adatvesztés nélkül kibővíteni, sőt akár átalakítani is RAID szintek között.

A gépekben Emulex LightPulse FC adapterek vannak, 2Gbit/sec sebességűek. A nodeok ugyanazt a fizikai diszkterületet is láthatják, így ha több gépről szeretnénk ugyanazokat az adatokat elérni, akkor valamilyen cluster filerendszer megoldást kell használni.

Szóba jöhetnek: GFS2, OCFS2, GPFS, CXFS

GFS2: Global File System 2, eredetileg a Sistina Software fejlesztette, később megvette a RedHat. Minden szükséges szolgáltatást biztosít. Hátulütője, hogy erősen development fázisban van, a Linux kernelnek nem része, külön patch létezik, de DLM nélkül, ami szintén nincs az alap kernelben, csak egy fejlesztői ágban, legegyszerűbben úgy érhető el a legfrissebb kód, hogy a saját GIT tree-jéből checkout-oljuk. Ezt a saját tree-t nemrégiben kapta csak, ebben benne vannak a mainstream kernel frissítései is, és a DLM+GFS2 frissítések is.

OCFS2: Oracle Cluster File System 2, Oracle fejlesztés. Hasonló az Ext3 filerendszerhez. Része a Linux kernelnek, stabilnak mondható a kód. Sajnos kvótázást nem támogat.

GPFS: General Parallel File System, IBM fejlesztette ki még az első RS/6000 SP sorozathoz AiX alá, később elkészült Linux alá is, mind x86 mint ppc64 platformokra. A kernellel való kommunikációhoz szükséges modul GPL licenszű, „csak” a GPFS library és a segédprogramok zárt forrásúak. Nem ingyenes!!

CXFS: Clustered XFS, SGI fejlesztés. Nagy teljesítményű filerendszer, ez sem ingyenes, sajnos nem áll rendelkezésre.

5. IBM Service aids for hardware diagnostics

Az IBM kiadott néhány speciális programot ezekhez a gépekhez, ezeket a *http://techsupport.services.ibm.com/server/lopdiags* címről tölthetjük le. Az oldalon írtakkal ellentétben nem csak a felsorolt, általuk előnyben részesített disztribúciókkal (SuSE, RedHat, TurboLinux) kompatibilisek a felajánlott csomagok, remekül futnak Debian Sarge alatt is.

ppc64utils/ nvram	Mint a neve is sugallja, az NVRAM partícióinak a tartalmát lehet vele értelmezni, amit a /dev/nvram-on keresztül ér el: OF változókat, szerviz processzor változóit, error logokat és a VPD-t (Vital Product Data).
ppc64utils/ snap	System Snapshot: snap.tar.gz fileba ment le bizonyos konfigurációs fileokat, illetve állapotinformációkat a rendszerről. IBM hibabejelentéskor kérhet egy ilyet snapshotot, a könnyebb diagnosztika érdekében.
ppc64utils/ update_flash	Ezzel a szkripttel lehet firmware-t frissíteni az RTAS-on keresztül, természetesen csak akkor, ha a kernel is támogatja. Lásd a következő fejezetet.
diagela	AiX alatt használatos diagnosztikai program Linuxos verziója, az esetleges hibák felderítésében nyújthat segítséget. Ez tartalmazza az rtas_errd-t is, amiről az RTAS fejezetben még szó lesz.
src	Az AiX-ról ismert System Resource Controller Linuxos implementációja, aki már hozzászokott az SRC nyújtotta kényelemhez érdemes lehet feltennie.
lsvpd	Szintén az AiX-on használt lsvpd, lsmcode, lscfg parancsok Linuxos megfelelői. Igaz Linux alatt jelentősen kevesebb információt közöl a gépről, mint AiX alatt. Lásd a függeléket.

A többi elérhető RPM AiX-os cluster management programoknak a linuxosított verziói: rsct, inventory scout, csm. Ezeket nyugodtan hanyagolhatjuk, nélkülük is teljes a rendszerünk.

6. Firmware upgrade

Az aktuális firmware verziót az lsmcode nevű segédprogrammal kérdezhetjük le. A frissítés pedig AiX-ból ezzel a paranccsal hajtható végre:

```
/usr/lpp/diagnostics/bin/update_flash -f firmware_image_file
```

A legfrissebb microcode-okat az IBM honlapjáról tölthetjük le: *http://techsupport.services.ibm.com/server/mdownload*

Leírás szerint akár egy óráig is eltarthat a procedúra, de tipikusan 10 és 30 perc között végez. A tesztgépünkben (620) 2001 májusi platform firmware (CL010507) volt gyárilag, frissítve lett egy 2003 februárban kiadottra (CL021213). A dokumentációban

leírtakkal ellentétben néhány percig tartott csak. Az update_flash programot az IBM kiadta Linux alá is, így Linux alól is frissíthetünk firmware-t a későbbiekben, természetesen csak ha engedélyeztük a kernelben a *Firmware Flash Interface*-t.

Kipróbáltam működőképes-e, frissítettem CL040712-re Linux alól a 620-as gépen. Rögvest a következő üzenetet kaptam a kerneltől:

```
FLASH: flash image with 3291758 bytes stored for hardware flash on
reboot
```

Ezután automatikusan rebootol, de mikor elér ahhoz a ponthoz, ahol normális esetben újraindítja a gépet, ahelyett, hogy restartolná, az RTAS elvégzi a firmware upgrade-et. Ilyenkor még véletlenül se kapcsoljuk ki, bár vészhelyzet eseten egy soros terminállal és a firmware-t tartalmazó floppy lemezekkel felszerelkezve van esély a visszaállításra a szervíz processzoron keresztül.

```
Restarting system.
FLASH: preparing saved firmware image for flash
FLASH: flash image is 3291758 bytes
FLASH: performing flash and reboot
FLASH: this will take several minutes. Do not power off!
```

7. RTAS

A Run-Time Abstraction Service (RTAS) egy firmware interfész, aminek segítségével az operációs rendszer platform specifikus funkciókhoz férhet hozzá anélkül, hogy platform függő kódra lenne szükség. A ppc64-es Linux kernel használ néhány RTAS funkciót, mint például hozzáférés az NVRAM-hoz, reboot, shutdown, firmware frissítés, belső érzékelők kezelése, operator panel vezérlés, kommunikáció a szerviz processzorral, heartbeat, stb.

A Linux kernel automatikusan indít egy rtasd nevű daemon-t, aminek az a dolga, hogy az esetleges hibákat loggolja az NVRAM-ba, illetve bootoláskor kiolvassa onnan, hogy az előző leállásnál történt-e hiba, mivel fatális hiba esetén csakis az NVRAM-ban marad nyoma a rendellenes működésnek. Ezenfelül még user space programok számára is elérhetővé teszi az error log-ot, ez a daemon felelős a /proc/ppc64/error_log-ért.

Nézzük meg milyen bejegyzéseket tartalmaz a /proc/ppc64/rtas/:

clock	Interface a hardveres órához, az Epoch óta eltelt másodpercek számát olvashatjuk ki belőle, illetve be is állíthatjuk ezen keresztül az órát.
error_log	A szerviz processzor által érzékelt hibákról szóló jelentést lehet elérni ezen a bejegyzésen keresztül. A fentebb említett diagela csomagban található rtas_errd daemon folyamatosan figyeli, és rögzíti a hibákat a filerendszerre is, mégpedig a /var/log/platform fileba, illetve értesít is az eseményekről. Egy ilyen log található a függelékben (Függelék E.).
frequency	A beépített hanggenerátort (mint a PC Speaker) vezérli, a kívánt frekvenciát adhatjuk meg Hz-ben.
poweron	Automatikus bekapcsolási időpontot lehet ezen keresztül beállítani.
sensors	A gépben lévő érzékelők által szolgáltatott adatokat olvashatjuk ki innen:

hőérzékelők, feszültségmérők, ventilátor fordulatszámmérők, stb.

progress — Az oppanel LCD-re írhatunk ezen keresztül szöveget, például egy kis szkripttel ki lehet rá iratni időközönként a terhelést, uptime-ot, és más hasznos információt.

volume — Az említett hanggenerátor erősségét szabályozhatjuk elméletileg százalékban megadva, viszont a skála logaritmikus, nem lineáris. Gyakorlatilag ezeken a gépeken a 1-100 intervallumba eső számok között nincs különbség, mind ugyanolyan hangosan szól, ha 0-t írunk bele, akkor hallgat el.

Ha a kernelben engedélyeztük a firmware frissítés lehetőségét, akkor ezeken felül még négy bejegyzést hoz létre a kernel: firmware_flash, firmware_update, manage_flash, validate_flash. A service aids programok között található update_firmware szkript ezeken keresztül adja át az RTAS-nak a firmware image-t és utasítja a gépet a shutdown-ra és a shutdown utáni véglegesítésre.

7.1. RTAS programozása

Az RTAS elérése egyetlen híváson keresztül történik, ennek a függvénynek a neve: *rtas-call*. Ennek a címét bootoláskor szerzi meg a kernel, és egy kisebb memória részt is lefoglal az RTAS-sal való kommunikáció számára ("instantiating rtas..." üzenet, lásd a dmesg-ben). Változó paraméterszámú függvény, első paramétere egy RTAS-token elnevezésű azonosító, amivel a konkrét RTAS eljárást adjuk meg. Következő két paraméter a kimenő illetve bemenő paraméterek számát adja meg.

RTAS token-ek például:

get-time-of-day/ set-time-of-day — Mint a nevükből is kitalálható, az idő lekérdezésére, és beállítására szolgálnak, a /proc/ppc64/clock-ot olvasva, írva ezekkel az token-ekkel hívódik meg az rtas-call függvény.

display-character — Ezt hasznosítja a /proc/ppc64/progress, lekérdezni nem lehet, hogy éppen mi van az LCD-n, bár a progress bejegyzést olvashatjuk, annak a kezelője úgy hidalja át a problémát, hogy mindig eltárolja a memóriában, hogy mi lett utoljára beleírva.

nvram-fetch/ nvram-store — Az NVRAM olvasására és írására szolgáló token-ek, paraméterként megadott memóriacímről olvassa/írja az adatokat.

set-time-for-power-on — Fentebb a /proc/ppc64/rtas/poweron-nál említett automatikus bekapcsolási időt állítja be.

event-scan — A szerviz processzor által detektált hibák kiolvasására szolgál. Az operációs rendszer periodikusan hívogatja az esetleges hibákért.

read-pci-config/ write-pci-config — A PCI busz, és PCI eszközök paramétereinek lekérdezésére vagy módosítására szolgálnak.

power-off — Kikapcsolja a gépet.

get-sensor-state — Ezzel a token-nel olvassa ki a kernel a sensor-okra vonatkozó adatokat, amikor a /proc/ppc64/sensors tartalmát nézzük meg.

set-indicator — Ez pedig az előző token-nek a másik fele, igaz itt nem sensor-nak,

hanem indicator-nak nevezik. Ahogyan a nevéből is látszik, ezzel beállítani tudjuk a különböző indicator-okat, mint pl. LED-eket az oppanel-en, ha vannak.

Az utóbbi kettő felhasználásával készítettem egy kernel kiegészítést, ami a pSeries 640 7026-B80 típusú gépek oppanel-jén található LED-et tudja kezelni. Kétféle módon tud működni: folyamatosan világít, illetve villog. A folyamatos világítást valamilyen hiba esetén szokta beállítani a szervíz processzor, ez jelenti a System Fault állapotát a LED-nek, és AiX alól csak kioltani lehet a /usr/lpp/diagnostics/bin/usysfault segédprogrammal.

A villogó állapot a System Identification, ezt AiX alól is szabadon kapcsolhatjuk ki-be (/usr/lpp/diagnostics/bin/usysident), hasznos ha sok ugyanolyan gépünk van egy helyen, így megkülönböztető jelzésként szolgál.

Linux alóli vezérlést a /proc/ppc64/rtas alá tenni találtam a legcélszerűbbnek, mivel ott van már a többi RTAS-hoz kapcsolódó dolog is. A két állapothoz két bejegyzést csináltam: sysfault, sysident. Ha olvassuk őket akkor az aktuális állapotot adják vissza (0: kikapcsolva, 1: bekapcsolva), ha írunk bele, akkor 1-re bekapcsol, 1-től különböző értékre pedig kikapcsol. Az indicator-ok token-jeit egyrészt az OF device-tree-ben kerestem meg (/proc/device-tree/rtas/ibm,indicator-*), másrészt a sensors bejegyzés olvasásakor is jelen volt két Unkown Sensor (9006 es 9007), és sejtettem, hogy ezek lehetnek azok, de a biztonság kedvéért még megnéztem az említett AiX-os segédprogramokat is, hogy milyen stringek szerepelnek a bináris file-ban, illetve a libdiag-ban. Ezekben is megtaláltam a 'set-indicator', 'ibm,indicator-9006', 'ibm,indicator-9007' stringeket, igy már biztos lehettem benne, hogy jó úton járok.

Első lépésben a sensors handler-ét bővítettem ki, hogy ismerje a 9006-as és 9007-as számú indicator-okat, mint 'System Fault', és 'System Indicator'. Így az állapotukat még lekérdezhetjük a sensors bejegyzés olvasásával is.

Az arch/ppc64/kernel/rtas-proc.c fileban a következő módosításokat végeztem el:

```
...
/* Az allapotokat ember altal is ertelmezheto formaban
   leiro stringek */
const char * sysfault_indicator[]    = { "Normal", "Fault" };
const char * sysident_indicator[]    = { "Normal", "On" };
...

...
/* A sensorok adatainak ertelmezesekor egy switch-case
   szerkezeten megy vegig, ehhez adtam hozza meg a ket
   indicatort */
case IBM_SYSFAULT_INDICATOR:
               label_string = "System Fault indicator:";
               value_arrsize = sizeof(sysfault_indicator)/
                               sizeof(char *);
               value_arr = sysfault_indicator;
               break;
case IBM_SYS_IDENTIFICATION:
               label_string = "Identification light:";
               value_arrsize = sizeof(sysident_indicator)/
```

```
                                sizeof(char *);
                value_arr = sysident_indicator;
                break;
...
```

Második lépésben ugyanezt a filet bővítettem tovább. Definiáltam két makrót a két indicator számának:

```
...
#define IBM_SYSFAULT_INDICATOR 0x232e /* 9006 */
#define IBM_SYS_IDENTIFICATION 0x232f /* 9007 */
...
```

Ezután definiáltam a függvény prototípusokat, illetve egy-egy tömböt kreáltam, amik a proc bejegyzésen való fileműveletek handler-eit tartalmazzák:

```
...
/* Az olvasas/iras fuggvenyek protipusai */
static ssize_t ppc_rtas_sysfault_read(struct file * file,
               char * buf, size_t count, loff_t *ppos);
static ssize_t ppc_rtas_sysfault_write(struct file * file,
               const char * buf, size_t count, loff_t *ppos);

static ssize_t ppc_rtas_sysident_read(struct file * file,
               char * buf, size_t count, loff_t *ppos);
static ssize_t ppc_rtas_sysident_write(struct file * file,
               const char * buf, size_t count, loff_t *ppos);
...

...
/* Megadjuk az olvasashoz es irashoz tartozo handlereket */
struct file_operations ppc_rtas_sysfault_operations = {
            .read =         ppc_rtas_sysfault_read,
            .write =        ppc_rtas_sysfault_write
};

struct file_operations ppc_rtas_sysident_operations = {
            .read =         ppc_rtas_sysident_read,
            .write =        ppc_rtas_sysident_write
};
...
```

Miután ez megvolt egy a sysfault, sysident bejegyzéseket létrehozó kódot adtam hozzá:

```
...
/* Ha ervenyes a mar fentebb ellenorzott get_sensor_state es
   set_indicator RTAS token, akkor jegyzi be csak */
if ((get_sensor_state != RTAS_UNKNOWN_SERVICE) ||
    (set_indicator != RTAS_UNKNOWN_SERVICE)) {
        entry=create_proc_entry("sysfault",S_IRUGO|
```

```
                                        S_IWUSR,rtas_proc_dir);
                if (entry)
                        entry->proc_fops = &ppc_rtas_sysfault_operations;
        }

        if ((get_sensor_state != RTAS_UNKNOWN_SERVICE) ||
            (set_indicator != RTAS_UNKNOWN_SERVICE)) {
                entry=create_proc_entry("sysident",S_IRUGO|
                                        S_IWUSR,rtas_proc_dir);
                if (entry)
                        entry->proc_fops = &ppc_rtas_sysident_operations;
        }
        ...
```

Végül megírtam a hozzájuk való handler-eket, amik olvasáskor, illetve íráskor hívódnak meg.

1/A. *sysfault* írása:

```
static long ppc_rtas_sysfault_write(struct file * file,
            const char * buf, size_t count, loff_t *ppos)
{
       char state = 0;
       int error;

       /* userspacebol atmasoljuk a beirt adatot */
       if (copy_from_user(&state, buf, 2))
                return -EFAULT;
       /* Ha az 1 karakter (0x31) lett beleirva, akkor az allapot
          legyen 1, ellenkezo esetben 0 */
       if (state == '1')
                state = 1;
       else
                state = 0;

       /* az RTAS meghivasa - set_indicator paranccsal,
                3 bemeno es 1 kimeno parameter lesz,
                a kimenore nem vagyunk kivancsiak ezert NULL,
                az elso  bemeno az indicator azonositoja,
                masodik a megjelolt indicatoron beluli index,
                es vegul, hogy milyen ertek allitodjon be */
       error = rtas_call(set_indicator, 3, 1, NULL,
                         IBM_SYSFAULT_INDICATOR, 0, state);

       /* hibakezeles */
       if(error != 0)
       {
                printk(KERN_WARNING "rtas-proc.o: sysfault write \
                                     failed\n");
                return -EFAULT;
       }
       return count;
}
```

1/B. *sysfault* olvasása:

```
static ssize_t ppc_rtas_sysfault_read(struct file * file, char * buf,
                size_t count, loff_t *ppos)
{
        int n, sn;
        unsigned long ret;
        int state, error;
        char tmpbuf[4];

        /* az RTAS hivas, amivel kiolvassuk az erteket,
             get_sensor_state - a parancs,
             2 - kimeno parameterek szama,
             2 - bemeno parameterek szama,
             &ret - erre a cimre fogja irni a kert adatokat,
             IBM_SYSFAULT_INDICATOR - a kivant sensor szama
             0 - a megjelolt indicator indexe
         */
        error = rtas_call(get_sensor_state, 2, 2, &ret,
                          IBM_SYSFAULT_INDICATOR, 0);
        state = (int) ret;

        /* az ertek atadasa userspace-be */
        n = snprintf(tmpbuf, 4, "%d\n", state);
        sn = n + 1;
        if (*ppos >= n)
                return 0;

        if (n > count)
                n = count;
        if (n > sn - *ppos)
                n = sn - *ppos;
        if (copy_to_user(buf, tmpbuf + (*ppos), n))
                return -EFAULT;

        *ppos += n;
        return n;
}
```

2/A. *sysident* írása:

```
static ssize_t ppc_rtas_sysident_write(struct file * file, const char *
buf,
                size_t count, loff_t *ppos)
{
        char state = 0;
        int error;

        if (copy_from_user(&state, buf, 2))
                return -EFAULT;
        if (state == '1')
                state = 1;
        else
                state = 0;
        error = rtas_call(set_indicator, 3, 1, NULL,
                          IBM_SYS_IDENTIFICATION, 0, state);
        if(error != 0)
        {
                printk(KERN_WARNING "rtas-proc.o: sysident write \
```

```
                                        failed\n");
                return -EFAULT;
        }

        return count;
}
```

2/B. *sysident* olvasása:

```
static ssize_t ppc_rtas_sysident_read(struct file * file, char * buf,
                size_t count, loff_t *ppos)
{
        int n, sn;
        unsigned long ret;
        int state, error;
        char tmpbuf[4];

        error = rtas_call(get_sensor_state, 2, 2, &ret,
                          IBM_SYS_IDENTIFICATION, 0);
        state = (int) ret;
        n = snprintf(tmpbuf, 4, "%d\n", state);
        sn = n + 1;
        if (*ppos >= n)
                return 0;

        if (n > count)
                n = count;
        if (n > sn - *ppos)
        n = sn - *ppos;
        if (copy_to_user(buf, tmpbuf + (*ppos), n))
                return -EFAULT;
        *ppos += n;
        return n;
}
```

A bitkeeper-ből való 2.4.27-es verziójú kernelhez a teljes patchet lásd a függelékben (Függelék F.).

8. OpenFirmware device-tree

Az Open Firmware Device Tree egy hierarchikus adat struktúra, ami a rendszer hardvereit és konfigurálhatóságát írja le. Használható a különböző hardverek reprezentációjára az operációs rendszer számára. Mint minden fa, ez is csomópontokból áll, melyek közül néhány levél. Minden csomópontnak van szülője, kivéve a legfelső csomópontot. A csomópontok névvel vannak ellátva, és tartozik hozzájuk egy lista, amiben az adott csomópont tulajdonságai vannak eltárolva. Igazából a név is egy tulajdonság, abban speciális, hogy mindenképpen léteznie kell. Ezzel a névvel, az ún. device specifier-rel lehet azonosítani a csomópontokat, tehát a neveknek egyedieknek kell lenniük. A bejegyzések lehetnek int, bytes, string típusúak.

OF prompt-ból a következőképpen tekinthetjük meg a tree-t:

```
0> dev /
0> ls
```

Linux alól ezt a /proc/device-tree alatt érhetjük el. Nézzük is meg a fontosabb bejegyzéseket:

aliases/	- a definiált OF aliasok
pci/	- a PCI buszon található eszközök listája és azok tulajdonságai
memory-controller/	- memóriavezérlő tulajdonságai és típusa
memory/	- a gépben lévő memória modulok adatai
interrupt-controller/	- a megszakítás vezérlő típusa és tulajdonságai
cpus/	- információk a használható processzorokról
nvram/	- adatok az NVRAM-ról, többek között tpus és méret
options/	- OF változók, amiket pl. az OF prompt-ból tudunk változtatni.
rtas/	- az RTAS képességeit felsoroló csomópont: milyen hívásokat támogat; általa ismert token-ek
device_type	- a gép al-architektúrája, jelen esetben CHRP.
model + name	- a gép típusjelzését tartalmazó string: pl: IBM,7025-6F1, 9124-720
ibm,vpd	- a Vital Product Data-t tartalmazó levél. Ezt használják az említett lsvpd, lsmcode, lscfg szkriptek.

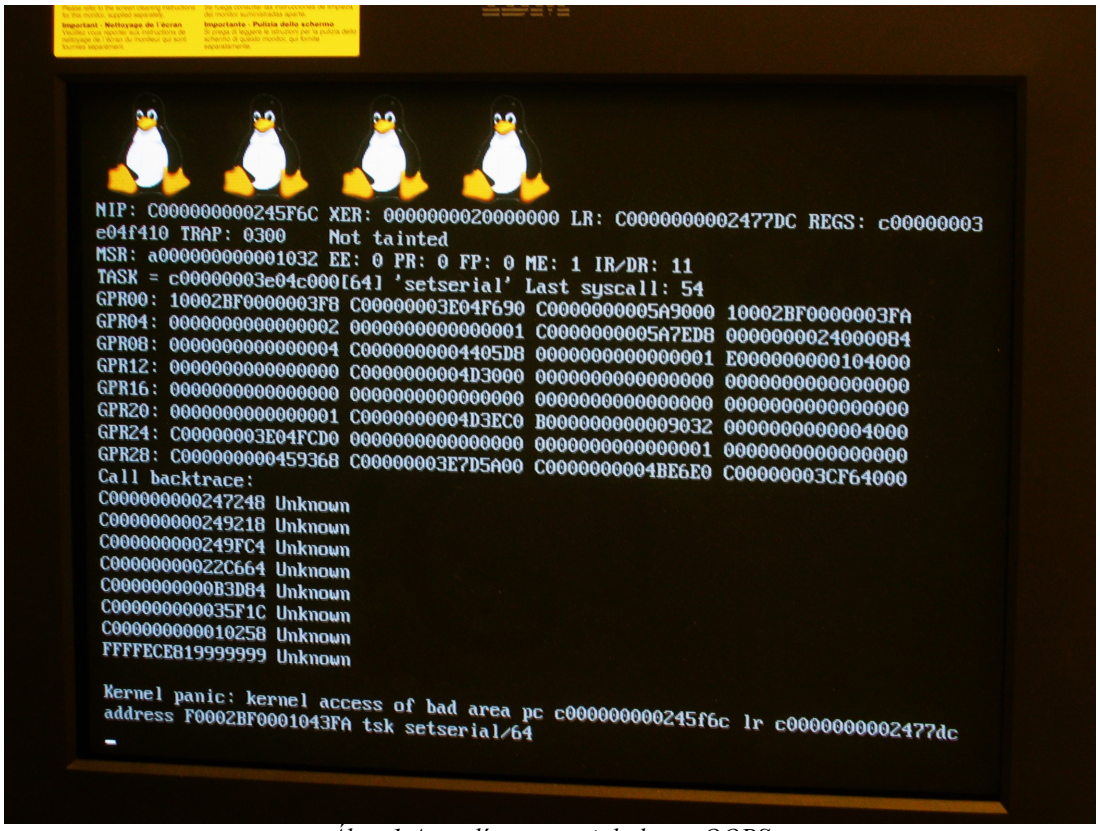
A komplett device-tree-ért lásd a függeléket (Függelék G.), mind az OF prompt-ból elérhetőt, mind a Linux alól elérhetőt.

9. 32 vs. 64-bites alkalmazások

Az IBM 64-bites PowerPC processzor implementációja teljes mértékű 32-bites végrehajtási kompatibilitást biztosít teljesítmény csökkenés nélkül. Két végrehajtási módja van a processzornak: 32-bites, és 64-bites; 32-bites módban az instrukciók és a címzés pont úgy viselkednek, mint egy 32-bites processzoron, 64-bites módban pedig egy igazi 64-bites környezet áll rendelkezésére. A hardveres 32-bites bináris kompatibilitás natív, nem támaszkodik semmilyen veszteséges emulációs technikára, így kiválóan futnak a 32-bites Linuxos alkalmazások, sőt a 64-bites környezetre épített általános célú szoftverek, melyeknek nincs szüksége a megnövekedett címtartomány használatára csekély teljesítmény veszteséget szenvedhetnek néhány dolog többek között a 64-bites cimkék kezelése miatt. Mivel az instrukciók fix 32-bit hosszúságúak, amik egyszerre csak 16-bit hasznos adatot hordozhatnak maximum. Így egy 64-bites regiszter feltöltéséhez 64 / 16 = 4 instrukció szükséges. Viszont nincs olyan utasítás ami közvetlenül egy 64-bites általános célú regiszter magas helyiértékű doubleword-jébe tudna írni, így először az alacsony helyiértékűt kell feltölteni, és azt shiftelni 32-bittel, és ezután kerülhet az alsó 4 byte-ba az oda szánt adat.

32-bites alkalmazások használatakor is van némi overhead, mégpedig a rendszerhívások bekövetkeztekor, de ennek hatása minimálisra tehető. Mivel nem jár számottevő teljesítmény veszteséggel a programok 32-bites módban való futtatása, és már rengeteg 32-bites ppc linux disztribúció, és alkalmazás érhető el, így a tipikus végrehajtási környezet ppc64-en is 32-bites. A RedHat, SuSE, TurboLinux fizetős disztribúciói is mind 32-bitesek, Debian-nak sincsenek még hivatalos ppc64 csomagjai (remélhetőleg a nem túl távoli jövőben lesznek, az Apple PowerPC G5 alapú termékeinek terjedésének köszönhetően, egyelőre csak egy erősen tesztverzió érhető el, nem hivatalos forrásból), viszont Fedora-nak folyamatosan frissülő ppc64 csomagjai vannak, de ezekről semmilyen információ, dokumentáció nem érhető el, pusztán maguk a csomagok érhetőek el, valószínűleg automatizáltan generálódnak, és nincsenek tesztelve, illetve meg kell még említeni a Gentoo-t, ahol saját magunk építhetjük lépésről lépésre Linux rendszerünket, így persze fordíthatjuk 64-bitesre a library-ket és az alkalmazásokat.

Végül azt is meg kell említeni, hogy a 32 és 64-bites alkalmazások között formailag is van különbség, a Linux/ppc32-es binárisoknak SVR4 az ABI-ük, a Linux/ppc64 pedig az AIX-ot követve PowerOpen ABI-val dolgozik.



Ábra 1 Az említett setserial okozta OOPS

Függelék A. - hexdump a CD első szektoráról

```
00000000   C9 C2 D4 C1  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000010   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000020   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000030   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000040   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000050   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000060   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000070   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000080   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000090   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000000A0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000000B0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000000C0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000000D0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000000E0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000000F0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000100   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000110   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000120   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000130   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000140   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000150   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000160   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000170   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000180   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000190   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000001A0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000001B0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 80 FF  ................
000001C0   FF FF 96 FF  FF FF 00 00  00 00 9C 6B  04 00 00 FF  ...........k....
000001D0   FF FF 00 FF  FF FF 00 00  00 00 00 00  00 00 00 FF  ................
000001E0   FF FF 00 FF  FF FF 00 00  00 00 00 00  00 00 00 FF  ................
000001F0   FF FF 00 FF  FF FF 00 00  00 00 00 00  00 00 55 AA  ..............U.
```

Függelék B. - bootolási folyamat yaboot debuggal

```
Adding OF methods...
    prom_init - OF interface initialized.
    yaboot_start - Malloc buffer allocated at 00300000 (1048576 bytes)
    yaboot_start - reloc_offset :  0         (should be 0)
    yaboot_start - test_bss     :  0         (should be 0)
    yaboot_start - test_data    :  0         (should be 0)
    yaboot_start - &test_data   :  00221110
    yaboot_start - &test_bss    :  0022111c
    yaboot_start - linked at    :  0x00200000
    yaboot_start - Running on _machine = 4
    yaboot_main - /chosen/bootpath = /pci@fff7f09000/pci@b/scsi@1/sd@4,0
    yaboot_main - After parse_device_path:
dev=/pci@fff7f09000/pci@b/scsi@1/sd@4,0, part=-1, file=
    yaboot_main - After pmac path kludgeup:
dev=/pci@fff7f09000/pci@b/scsi@1/sd@4,0, part=-1, file=
    open_file - dev_path = /pci@fff7f09000/pci@b/scsi@1/sd@4,0
file_name = /etc/yaboot.conf
partition = -1
    open_file - device is a block device
    partitions_lookup - block size of device is 512
partitions:
    file_block_open - number: 03, start: 0x001df000, length: 0x041ee800
--> ext2_open
    ext2_open - dev=/pci@fff7f09000/pci@b/scsi@1/sd@4,0, part=0x00308060 (3),
file_name=/etc/yaboot.conf
    ext2_open - partition offset: 1004535808
    ext2_open - </pci@fff7f09000/pci@b/scsi@1/sd@4,0:0>
    ext2_open - file->of_device = 01a85380
<-- ext2_open - FILE_ERR_OK
    ext2_read - ext_read() from pos 0x0, size: 0x32768x
Config file read, 4096 bytes
    ext2_close - ext2_close called
    load_config_file - Config file successfully parsed, 4096 bytes
Acelvaros Linux by Kamm


Welcome to yaboot version 1.3.10
Enter "help" to get some basic usage information
boot:
* fb                          serial
boot: fb
    get_params - after parse_device_path:
dev=/pci@fff7f09000/pci@b/scsi@1/sd@4,0 part=-1 file=/fb
Please wait, loading kernel...
    open_file - dev_path = /pci@fff7f09000/pci@b/scsi@1/sd@4,0
file_name = /fb24_kamm2
partition = -1
    open_file - device is a block device
    partitions_lookup - block size of device is 512
partitions:
    file_block_open - number: 03, start: 0x001df000, length: 0x041ee800
--> ext2_open
    ext2_open - dev=/pci@fff7f09000/pci@b/scsi@1/sd@4,0, part=0x00310740 (3),
file_name=/fb24_kamm2
    ext2_open - partition offset: 1004535808
    ext2_open - </pci@fff7f09000/pci@b/scsi@1/sd@4,0:0>
    ext2_open - file->of_device = 01a85380
<-- ext2_open - FILE_ERR_OK
```

```
    ext2_read - ext_read() from pos 0x0, size: 0x20x
    ext2_read - ext_read() from pos 0x14, size: 0x32x
    load_elf32 - Elf32 header:
    load_elf32 -  e.e_type      = 2
    load_elf32 -  e.e_machine   = 20
    load_elf32 -  e.e_version   = 1
    load_elf32 -  e.e_entry     = 0x00400000
    load_elf32 -  e.e_phoff     = 0x00000034
    load_elf32 -  e.e_shoff     = 0x0029b0c8
    load_elf32 -  e.e_flags     = 0
    load_elf32 -  e.e_ehsize    = 0x00000034
    load_elf32 -  e.e_phentsize = 0x00000020
    load_elf32 -  e.e_phnum     = 2
    ext2_read - ext_read() from pos 0x34, size: 0x64x
    load_elf32 - Before prom_claim, mem_sz: 0x00400000
    load_elf32 - After ELF parsing, load base: 00400000, mem_sz: 0x00400000
    load_elf32 -     wanted load base: 0x00400000, mem_sz: 0x00400000
    ext2_read - ext_read() from pos 0x10000, size: 0x2666453x
   Elf32 kernel loaded...
    ext2_close - ext2_close called
    yaboot_text_ui - setting kernel args to: root=/dev/sdb3 ro console=ttyS0
video=matrox console=tty1
    yaboot_text_ui - flushing icache...    yaboot_text_ui -  done
    yaboot_text_ui - Kernel entry point = 00400000
    yaboot_text_ui - kernel: arg1 = 00400000,
        arg2 = 0x00000000,
        prom = 00c1e160,
        arg4 = 0,
        arg5 = 0

    yaboot_text_ui - Entering kernel...

zImage starting: loaded at 0x400000
    trying: 0x01400000
    trying: 0x01500000
    trying: 0x01600000
    trying: 0x01700000
    trying: 0x01800000
    trying: 0x01900000
    trying: 0x01a00000
    trying: 0x01b00000
    trying: 0x01c00000
    trying: 0x01d00000
gunzipping (0x1d00000 <- 0x407000:0x64b1d9)...done 7429691 bytes
57160 bytes of heap consumed, max in use 41492
... skipping 0x10000 bytes of ELF header
kernel:
        entry addr = 0x1d10000
        a1         = 0x400000,
        a2         = 0x0,
        prom       = 0xc1e160,
        bi_recs    = 0x23b0000,
opening display /pci@fff7f0a000/pci@b,6/display@1... ok
instantiating rtas at 0x000000003ff59000... done
0000000000000000 : booting  cpu /cpus/PowerPC,RS64-III@0
0000000000000001 : starting cpu /cpus/PowerPC,RS64-III@2 ... ok
    starting secondary threads
opened /pci@fff7f09000
open success
opened /pci@fff7f0a000
open success
 (translate ok) returning from prom_init
---- start early boot console ----
```

```
firmware_features = 0x0
-----------------------------------------------------
naca                        = 0xc000000000004000
naca->pftSize               = 0x18
naca->paca                  = 0xc0000000004dd000

systemcfg                   = 0xc000000000005000
systemcfg->platform         = 0x100
systemcfg->processor        = 0x340001
systemcfg->processorCount   = 0x4
systemcfg->physicalMemorySize = 0x40000000
systemcfg->dCacheL1LineSize   = 0x80
systemcfg->iCacheL1LineSize   = 0x80
htab_data.htab              = 0xc00000003d000000
htab_data.num_ptegs         = 0x20000
-----------------------------------------------------
Starting Linux PPC64 2.4.27
idle = idle_default
[boot]0010 Setup System
Linux version 2.4.27 (kamm@suti) (gcc version 3.2.1) #19 SMP Fri Oct 8 19:52:16
CEST 2004
[boot]0012 Setup Arch
On node 0 totalpages: 262144
zone(0): 262144 pages.
zone(1): 0 pages.
zone(2): 0 pages.
[boot]0015 Setup Done
Kernel command line: root=/dev/sdb3 ro console=ttyS0 video=matrox console=tty1
[boot]0020 XICS Init
[boot]0021 XICS Done
time_init: decrementer frequency = 451.182086 MHz
time_init: processor frequency   = 451.200000 MHz
---- end early boot console ----
```

Függelék C/1. - dmesg

```
firmware_features = 0x0
-----------------------------------------------------
naca                         = 0xc000000000004000
naca->pftSize                = 0x18
naca->paca                   = 0xc0000000004dd000

systemcfg                    = 0xc000000000005000
systemcfg->platform          = 0x100
systemcfg->processor         = 0x340001
systemcfg->processorCount    = 0x4
systemcfg->physicalMemorySize = 0x40000000
systemcfg->dCacheL1LineSize  = 0x80
systemcfg->iCacheL1LineSize  = 0x80
htab_data.htab               = 0xc00000003d000000
htab_data.num_ptegs          = 0x20000
-----------------------------------------------------
Starting Linux PPC64 2.4.27
idle = idle_default
Linux version 2.4.27 (kamm@suti) (gcc version 3.2.1) #19 SMP Fri Oct 8 19:52:16
CEST 2004
On node 0 totalpages: 262144
zone(0): 262144 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: root=/dev/sdb3 ro console=ttyS0  video=matrox console=tty1
time_init: decrementer frequency = 451.182086 MHz
time_init: processor frequency   = 451.200000 MHz
Console: colour dummy device 80x25
Calibrating delay loop... 1117.38 BogoMIPS
Memory: 985612k available (3792k kernel code, 2724k data, 388k init)
[c000000000000000,c000000040000000]
Dentry cache hash table entries: 131072 (order: 9, 2097152 bytes)
Inode cache hash table entries: 65536 (order: 8, 1048576 bytes)
Mount cache hash table entries: 256 (order: 0, 4096 bytes)
Buffer cache hash table entries: 65536 (order: 7, 524288 bytes)
Page-cache hash table entries: 262144 (order: 9, 2097152 bytes)
proc_ppc64: Creating /proc/ppc64/pmc
PCI: Creating ../proc/ppc64/pcifr
PCI: Creating ../proc/ppc64/pci
POSIX conformance testing by UNIFIX
Entering SMP Mode...
Probe found 4 CPUs
Waiting for 3 CPUs
Processor 1 found.
Processor 2 found.
Processor 3 found.
Waiting on wait_init_idle (map = 0x0)
All processors have done init_idle
[boot]0040 PCI Probe
PCI: Probing PCI hardware
ISA bridge at 00:10.0
PCI: Probing PCI hardware done
[boot]0041 PCI Done
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
i/pSeries Real Time Clock Driver v1.1
VIO: missing or empty /vdevice node; no virtual IO devices present.
```

```
PPC64 nvram contains 262144 bytes
--------NVRAM Partitions---------
indx            sig     chks    len     name
0       02      9e      8192    ibm,scan-log
131072          02      26      512     ibm,es-logs
139264          02      a2      128     ibm,err-log
141312          52      11      3584    ibm,vpd
198656          70      fd      512     common
206848          71      bc      384     ibm,setupcfg
212992          72      62      64      post-err-log
214016          50      f0      37      of-config
214608          53      dd      2049    ibm,chkpt
247392          72      e5      7       ibm,diag-log
247504          a0      d2      74      IBM,AIX
248688          a0      9e      22      IBM,AIX
249040          a0      30      130     ppc64,linux
251120          7f      cc      689     wwwwwwwwwww
RTAS daemon started
Starting kswapd
Journalled Block Device driver loaded
devfs: v1.12c (20020818) Richard Gooch (rgooch@atnf.csiro.au)
devfs: boot_options: 0x0
Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
udf: registering filesystem
SGI XFS with no debug enabled
SGI XFS Quota Management subsystem
matroxfb: Matrox MGA-G200 (PCI) detected
matroxfb: BIOS on your Matrox device does not contain powerup info
matroxfb: 640x480x8bpp (virtual: 640x3270)
matroxfb: framebuffer at 0xFF18000000, mapped to 0xe00000000040f000, size
2097152
Console: switching to colour frame buffer device 80x30
fb0: MATROX VGA frame buffer device
Detected PS/2 Mouse Port.
pty: 256 Unix98 ptys configured
Serial driver version 5.05c (2001-07-08) with MANY_PORTS SHARE_IRQ SERIAL_PCI
enabled
ttyS00 at 0x03f8 (irq = 4) is a 16550A
ttyS01 at 0x02f8 (irq = 3) is a 16550A
ttyS02 at 0x0898 (irq = 14) is a 16550A
ttyS03 at 0x0890 (irq = 15) is a 16550A
hvc_count: couldn't find a 'vty' node
Floppy drive(s): fd0 is 2.88M
FDC 0 is a National Semiconductor PC87306
RAMDISK driver initialized: 16 RAM disks of 16384K size 1024 blocksize
loop: loaded (max 8 devices)
SCSI subsystem driver Revision: 1.00
PCI: Enabling device 01:01.0 (0140 -> 0143)
sym53c8xx: at PCI bus 1, device 1, function 0
sym53c8xx: setting PCI_COMMAND_MASTER...(fix-up)
sym53c8xx: setting PCI_COMMAND_INVALIDATE (fix-up)
sym53c8xx: 53c896 detected
PCI: Enabling device 01:01.1 (0140 -> 0143)
sym53c8xx: at PCI bus 1, device 1, function 1
sym53c8xx: setting PCI_COMMAND_MASTER...(fix-up)
sym53c8xx: setting PCI_COMMAND_INVALIDATE (fix-up)
sym53c8xx: 53c896 detected
sym53c896-0: rev 0x7 on pci bus 1 device 1 function 0 irq 35
sym53c896-0: ID 7, Fast-40, Parity Checking
sym53c896-0: handling phase mismatch from SCRIPTS.
sym53c896-1: rev 0x7 on pci bus 1 device 1 function 1 irq 34
sym53c896-1: ID 7, Fast-40, Parity Checking
sym53c896-1: handling phase mismatch from SCRIPTS.
```

```
scsi0 : sym53c8xx-1.7.3c-20010512
scsi1 : sym53c8xx-1.7.3c-20010512
  Vendor: IBM       Model: CDRM00203     !K  Rev: 1_05
  Type:   CD-ROM                             ANSI SCSI revision: 02
sym53c896-0-<2,*>: FAST-20 WIDE SCSI 40.0 MB/s (50.0 ns, offset 31)
  Vendor: IBM       Model: DDYS-T36950M      Rev: S9RA
  Type:   Direct-Access                      ANSI SCSI revision: 03
sym53c896-0-<4,*>: FAST-20 WIDE SCSI 40.0 MB/s (50.0 ns, offset 31)
  Vendor: IBM       Model: DDYS-T36950M      Rev: S9RA
  Type:   Direct-Access                      ANSI SCSI revision: 03
Attached scsi disk sda at scsi0, channel 0, id 2, lun 0
Attached scsi disk sdb at scsi0, channel 0, id 4, lun 0
sym53c896-0-<2,*>: FAST-20 WIDE SCSI 40.0 MB/s (50.0 ns, offset 31)
SCSI device sda: 71096640 512-byte hdwr sectors (36401 MB)
Partition check:
 /dev/scsi/host0/bus0/target2/lun0: p2 p3 p4
sym53c896-0-<4,*>: FAST-20 WIDE SCSI 40.0 MB/s (50.0 ns, offset 31)
SCSI device sdb: 71096640 512-byte hdwr sectors (36401 MB)
 /dev/scsi/host0/bus0/target4/lun0: p1 p2 p3
Attached scsi CD-ROM sr0 at scsi0, channel 0, id 1, lun 0
sym53c896-0-<1,*>: FAST-20 WIDE SCSI 40.0 MB/s (50.0 ns, offset 15)
sr0: scsi-1 drive
Uniform CD-ROM driver Revision: 3.12
md: raid0 personality registered as nr 2
md: raid1 personality registered as nr 3
md: md driver 0.90.0 MAX_MD_DEVS=256, MD_SB_DISKS=27
md: Autodetecting RAID arrays.
md: autorun ...
md: ... autorun DONE.
LVM version 1.0.8(17/11/2003)
Initializing Cryptographic API
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 4096 buckets, 64Kbytes
TCP: Hash tables configured (established 131072 bind 65536)
ip_conntrack version 2.1 (4096 buckets, 32768 max) - 248 bytes per conntrack
ip_tables: (C) 2000-2002 Netfilter core team
arp_tables: (C) 2002 David S. Miller
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
802.1Q VLAN Support v1.8 Ben Greear <greearb@candelatech.com>
All bugs added by David S. Miller <davem@redhat.com>
kjournald starting.  Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Freeing unused kernel memory: 388k init
```

Függelék C/2a. - /proc/cpuinfo (hibásan felismert SMT)

```
processor       : 0
cpu             : Pulsar
clock           : 451MHz
revision        : 0.1

processor       : 1
cpu             : Pulsar
clock           : 451MHz
revision        : 0.1

processor       : 2
cpu             : Pulsar
clock           : 451MHz
revision        : 0.1

processor       : 3
cpu             : Pulsar
clock           : 451MHz
revision        : 0.1

timebase        : 451183691
machine         : CHRP IBM,7025-6F1
```

Függelék C/2b. - /proc/cpuinfo (B80)

```
processor       : 0
cpu             : POWER3 (630+)
clock           : 375.000000MHz
revision        : 1.4

processor       : 1
cpu             : POWER3 (630+)
clock           : 375.000000MHz
revision        : 1.4

timebase        : 93749632
machine         : CHRP IBM,7026-B80
```

Függelék C/2c. - /proc/cpuinfo (OpenPower 720)

```
processor       : 0
cpu             : POWER5 (gr)
clock           : 1654.344000MHz
revision        : 2.2 (pvr 003a 0202)

processor       : 1
cpu             : POWER5 (gr)
clock           : 1654.344000MHz
revision        : 2.2 (pvr 003a 0202)

processor       : 2
cpu             : POWER5 (gr)
```

```
clock           : 1654.344000MHz
revision        : 2.2 (pvr 003a 0202)

processor       : 3
cpu             : POWER5 (gr)
clock           : 1654.344000MHz
revision        : 2.2 (pvr 003a 0202)

timebase        : 207053000
machine         : CHRP IBM,9124-720
```

Függelék C/3. - lspci -v

```
0000:00:00.0 Host bridge: IBM: Unknown device 0102 (rev 05)
        Flags: bus master, 66MHz, medium devsel, latency 0

0000:00:0b.0 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at fffffff0 (32-bit, prefetchable) [size=32K]
        Bus: primary=00, secondary=01, subordinate=03, sec-latency=248
        I/O behind bridge: 00010000-0001ffff
        Memory behind bridge: c0000000-c7ffffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:00:0b.2 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at <unassigned> (32-bit, prefetchable)
        Bus: primary=00, secondary=04, subordinate=06, sec-latency=248
        I/O behind bridge: 00020000-0002ffff
        Memory behind bridge: c8000000-cfffffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:00:0b.4 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at <unassigned> (32-bit, prefetchable)
        Bus: primary=00, secondary=07, subordinate=09, sec-latency=248
        I/O behind bridge: 00030000-0003ffff
        Memory behind bridge: d0000000-d7ffffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:00:0b.6 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at <unassigned> (32-bit, prefetchable)
        Bus: primary=00, secondary=0a, subordinate=0c, sec-latency=248
        I/O behind bridge: 00040000-0004ffff
        Memory behind bridge: d8000000-dfefffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:00:10.0 ISA bridge: Symphony Labs W83C553 (rev 10)
        Flags: bus master, medium devsel, latency 0

0000:00:11.0 Co-processor: IBM: Unknown device 00e0 (rev 01) (prog-if ff)
        Subsystem: IBM: Unknown device 00e1
        Flags: bus master, 66MHz, medium devsel, latency 72, IRQ 39
        Memory at fffffff0 (32-bit, prefetchable) [size=4K]
        Memory at fffffff0 (32-bit, prefetchable) [size=512K]
        Memory at fffffff0 (32-bit, prefetchable) [size=16M]
        Memory at fffffff0 (32-bit, prefetchable) [size=8M]

0000:01:01.0 SCSI storage controller: LSI Logic / Symbios Logic 53C896/897 (rev 07)
```

```
        Subsystem: LSI Logic / Symbios Logic LSI53C896/7 PCI to Dual Channel Ultra2 SCSI
Multifunction Controller
        Flags: bus master, medium devsel, latency 74, IRQ 35
        I/O ports at 1e800 [size=256]
        Memory at fffffff0 (64-bit, non-prefetchable) [size=1K]
        Memory at fffffff0 (64-bit, non-prefetchable) [size=8K]
        Capabilities: [40] Power Management version 2

0000:01:01.1 SCSI storage controller: LSI Logic / Symbios Logic 53C896/897 (rev 07)
        Subsystem: LSI Logic / Symbios Logic LSI53C896/7 PCI to Dual Channel Ultra2 SCSI
Multifunction Controller
        Flags: bus master, medium devsel, latency 74, IRQ 34
        I/O ports at 1ec00 [size=256]
        Memory at fffffff0 (64-bit, non-prefetchable) [size=1K]
        Memory at fffffff0 (64-bit, non-prefetchable) [size=8K]
        Capabilities: [40] Power Management version 2

0000:10:00.0 Host bridge: IBM: Unknown device 0102 (rev 05)
        Flags: bus master, 66MHz, medium devsel, latency 0

0000:10:0b.0 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at fffffff0 (32-bit, prefetchable) [size=32K]
        Bus: primary=00, secondary=11, subordinate=13, sec-latency=248
        I/O behind bridge: 00000000-0000ffff
        Memory behind bridge: c0000000-c7ffffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:10:0b.2 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at <unassigned> (32-bit, prefetchable)
        Bus: primary=00, secondary=17, subordinate=19, sec-latency=248
        I/O behind bridge: 00020000-0002ffff
        Memory behind bridge: c8000000-cfffffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:10:0b.4 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at <unassigned> (32-bit, prefetchable)
        Bus: primary=00, secondary=1a, subordinate=1c, sec-latency=248
        I/O behind bridge: 00030000-0003ffff
        Memory behind bridge: d0000000-d7ffffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:10:0b.6 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at <unassigned> (32-bit, prefetchable)
        Bus: primary=00, secondary=1d, subordinate=1f, sec-latency=248
        I/O behind bridge: 00040000-0004ffff
        Memory behind bridge: d8000000-dfffffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:10:0c.0 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at <ignored> (32-bit, prefetchable)
        Bus: primary=00, secondary=21, subordinate=23, sec-latency=248
        I/O behind bridge: 00050000-0005ffff
        Memory behind bridge: e0000000-e7ffffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:10:0c.2 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at <unassigned> (32-bit, prefetchable)
        Bus: primary=00, secondary=24, subordinate=26, sec-latency=248
        I/O behind bridge: 00060000-0006ffff
```

```
        Memory behind bridge: e8000000-effffffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:10:0c.4 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at <unassigned> (32-bit, prefetchable)
        Bus: primary=00, secondary=27, subordinate=29, sec-latency=248
        I/O behind bridge: 00070000-0007ffff
        Memory behind bridge: f0000000-f7ffffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:10:0c.6 PCI bridge: IBM: Unknown device 008b (rev 02) (prog-if 0f)
        Flags: bus master, 66MHz, slow devsel, latency 248
        BIST result: 00
        Memory at <unassigned> (32-bit, prefetchable)
        Bus: primary=00, secondary=2a, subordinate=2c, sec-latency=248
        I/O behind bridge: 00080000-0008ffff
        Memory behind bridge: f8000000-ffefffff
        Prefetchable memory behind bridge: 0000000000000000-0000000000000000

0000:11:01.0 Ethernet controller: Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
(rev 26)
        Flags: bus master, medium devsel, latency 74, IRQ 51
        I/O ports at 10ec00 [size=32]
        Memory at fffffff0 (32-bit, non-prefetchable) [size=32]
        Expansion ROM at fffff800 [size=1M]

0000:1d:01.0 VGA compatible controller: Matrox Graphics, Inc. MGA G200 (rev 01) (prog-if
00 [VGA])
        Flags: medium devsel, IRQ 54
        Memory at fffffff0 (32-bit, prefetchable) [size=16M]
        Memory at fffffff0 (32-bit, non-prefetchable) [size=16K]
        Memory at fffffff0 (32-bit, non-prefetchable) [size=8M]
        Expansion ROM at fffff800 [size=64K]
        Capabilities: [dc] Power Management version 1
```

Függelék C/4a. /proc/ppc64/lparcfg – 6F1

```
serial_number=IBM,016571FAA
system_type=IBM,7025-6F1
partition_id=0
system_active_processors=4
system_potential_processors=4
partition_max_entitled_capacity=400
partition_entitled_capacity=400
partition_active_processors=2
partition_potential_processors=4
shared_processor_mode=0
```

Függelék C/4b. /proc/ppc64/lparcfg – B80

```
lparcfg 1.5
serial_number=IBM,01445D3FA
system_type=IBM,7026-B80
partition_id=0
system_active_processors=2
system_potential_processors=2
partition_max_entitled_capacity=200
partition_entitled_capacity=200
partition_active_processors=2
partition_potential_processors=2
shared_processor_mode=0
```

Függelék D. - /proc/ppc64/sensors - B80

```
RTAS (RunTime Abstraction Services) Sensor Information
Sensor          Value          Condition       Location
*****************************************************
Key switch:     Normal         (read ok)       ---
Power source:   AC             (read ok)       ---
EPOW Sensor:    EPOW Reset     (read ok)       ---
Temp. (C/F):      21 /  69     (normal)        Planar #2
Surveillance:            0     (read ok)       ---
Fan (rpm):            2040     (normal)        Fan #1
Fan (rpm):            2040     (normal)        Fan #2
Fan (rpm):            2370     (normal)        Fan #3
Fan (rpm):            2340     (normal)        Fan #4
Voltage (mv):         5032     (normal)        Planar #2
Voltage (mv):         3321     (normal)        Planar #2
Voltage (mv):         5096     (normal)        Planar #2
Voltage (mv):        11922     (normal)        Planar #2
Voltage (mv):         1816     (normal)        Planar #1 at CPU #1
Voltage (mv):         2515     (normal)        Planar #1 at CPU #1
Voltage (mv):         5232     (normal)        Planar #1 at CPU #1
Voltage (mv):         5177     (normal)        Planar #1 at CPU #1
Powersupply:             3     (normal)        Voltage #1
Unknown sensor (type 9006), ignoring it
Unknown sensor (type 9007), ignoring it
```

Függelék E. - /var/log/platform

```
RTAS: 92 -------- RTAS event begin --------
RTAS 0: 03440040 00000082 96008508 00020600
RTAS 1: 19600101 00000002 40200000 00000000
RTAS 2: 00000000 00000000 00000000 00000000
RTAS 3: 49424d00 56320000 00503032 10111521
RTAS 4: 02a0005d 10112014 00000000 00000700
RTAS 5: 00000000 00000000 00000000 00000000
RTAS 6: 01000000 00000000 31303131 31353231
RTAS 7: 20202020 20202020 56322020 20202020
RTAS 8: 20202020 20202020 0002
RTAS: 92 -------- RTAS event end ----------
RTAS: 93 -------- RTAS event begin --------
RTAS 0: 03440040 00000082 96008508 00000100
RTAS 1: 19600101 00000002 40200000 00000000
RTAS 2: 00000000 00000000 00000000 00000000
RTAS 3: 49424d00 56320000 00503032 10111521
RTAS 4: 02a0005d 10112001 00000000 00000000
RTAS 5: 00000000 00000000 00000000 00000000
RTAS 6: 01000000 00000000 31303131 31353231
RTAS 7: 20202020 20202020 56322020 20202020
RTAS 8: 20202020 20202020 0002
RTAS: 93 -------- RTAS event end ----------
RTAS: 94 -------- RTAS event begin --------
RTAS 0: 03440040 00000082 96008508 00020900
RTAS 1: 19600101 00000002 40200000 00000000
RTAS 2: 00000000 00000000 00000000 00000000
RTAS 3: 49424d00 56320000 00503032 10111521
RTAS 4: 02a0005d 10112014 00000000 00000700
RTAS 5: 00000000 00000000 00000000 00000000
RTAS 6: 01000000 00000000 31303131 31353231
RTAS 7: 20202020 20202020 56322020 20202020
RTAS 8: 20202020 20202020 0002
RTAS: 94 -------- RTAS event end ----------
```

Függelék F. - patch

```
diff -Nur lin24-original/arch/ppc64/kernel/rtas-proc.c
lin24/arch/ppc64/kernel/rtas-proc.c
--- lin24-original/arch/ppc64/kernel/rtas-proc.c        2004-09-17
09:53:09.000000000 +0200
+++ lin24/arch/ppc64/kernel/rtas-proc.c 2004-10-03 00:13:24.000000000 +0200
@@ -50,6 +50,8 @@
 #define IBM_DRCONNECTOR                0x232b /* 9003 */
 #define IBM_POWERSUPPLY                0x232c /* 9004 */
 #define IBM_INTQUEUE           0x232d /* 9005 */
+#define IBM_SYSFAULT_INDICATOR 0x232e /* 9006 */
+#define IBM_SYS_IDENTIFICATION 0x232f /* 9007 */

 /* Status return values */
 #define SENSOR_CRITICAL_HIGH   13
@@ -176,6 +178,17 @@
 static ssize_t ppc_rtas_errinjct_read(struct file *file, char *buf,
                                       size_t count, loff_t *ppos);

+static ssize_t ppc_rtas_sysfault_read(struct file * file, char * buf,
+               size_t count, loff_t *ppos);
+static ssize_t ppc_rtas_sysfault_write(struct file * file, const char * buf,
+               size_t count, loff_t *ppos);
+
+static ssize_t ppc_rtas_sysident_read(struct file * file, char * buf,
+               size_t count, loff_t *ppos);
+static ssize_t ppc_rtas_sysident_write(struct file * file, const char * buf,
+               size_t count, loff_t *ppos);
+
+
 struct file_operations ppc_rtas_poweron_operations = {
        .read =         ppc_rtas_poweron_read,
        .write =        ppc_rtas_poweron_write
@@ -206,6 +219,16 @@
     .release =                 ppc_rtas_errinjct_release
 };

+struct file_operations ppc_rtas_sysfault_operations = {
+       .read =         ppc_rtas_sysfault_read,
+       .write =        ppc_rtas_sysfault_write
+};
+
+struct file_operations ppc_rtas_sysident_operations = {
+       .read =         ppc_rtas_sysident_read,
+       .write =        ppc_rtas_sysident_write
+};
+
 int ppc_rtas_find_all_sensors (void);
 int ppc_rtas_process_sensor(struct individual_sensor s, int state,
                int error, char * buf);
@@ -288,6 +311,18 @@
                if (entry) entry->proc_fops = &ppc_rtas_progress_operations;
        }

+       if ((get_sensor_state != RTAS_UNKNOWN_SERVICE) ||
+           (set_indicator != RTAS_UNKNOWN_SERVICE)) {
+               entry=create_proc_entry("sysfault",S_IRUGO|
S_IWUSR,rtas_proc_dir);
+               if (entry) entry->proc_fops = &ppc_rtas_sysfault_operations;
```

```
+       }
+
+       if ((get_sensor_state != RTAS_UNKNOWN_SERVICE) ||
+           (set_indicator != RTAS_UNKNOWN_SERVICE)) {
+               entry=create_proc_entry("sysident",S_IRUGO|
S_IWUSR,rtas_proc_dir);
+               if (entry) entry->proc_fops = &ppc_rtas_sysident_operations;
+       }
+
 #ifdef CONFIG_RTAS_ERRINJCT
        errinjct_token = rtas_token("ibm,errinjct");
        if (errinjct_token != RTAS_UNKNOWN_SERVICE) {
@@ -490,6 +525,114 @@
 }

 /* ****************************************************************** */
+/* SYSFAULT                                                           */
+/* ****************************************************************** */
+static long ppc_rtas_sysfault_write(struct file * file, const char * buf,
+               size_t count, loff_t *ppos)
+{
+       char state = 0;
+       int error;
+
+       if (copy_from_user(&state, buf, 2))
+               return -EFAULT;
+       if (state == '1')
+               state = 1;
+       else
+               state = 0;
+
+       error = rtas_call(set_indicator, 3, 1, NULL,
+                       IBM_SYSFAULT_INDICATOR, 0, state);
+       if(error != 0)
+       {
+               printk(KERN_WARNING "rtas-proc.o: sysfault write failed\n");
+               return -EFAULT;
+       }
+       return count;
+}
+/* ****************************************************************** */
+static ssize_t ppc_rtas_sysfault_read(struct file * file, char * buf,
+               size_t count, loff_t *ppos)
+{
+       int n, sn;
+       unsigned long ret;
+       int state, error;
+       char tmpbuf[4];
+
+       error = rtas_call(get_sensor_state, 2, 2, &ret,
+                       IBM_SYSFAULT_INDICATOR, 0);
+       state = (int) ret;
+       n = snprintf(tmpbuf, 4, "%d\n", state);
+       sn = n + 1;
+       if (*ppos >= n)
+               return 0;
+
+       if (n > count)
+               n = count;
+       if (n > sn - *ppos)
+               n = sn - *ppos;
+       if (copy_to_user(buf, tmpbuf + (*ppos), n))
+               return -EFAULT;
```

```
+
+       *ppos += n;
+       return n;
+}
+
+
+/* ********************************************************************** */
+/* SYSIDENT                                                               */
+/* ********************************************************************** */
+static ssize_t ppc_rtas_sysident_write(struct file * file, const char * buf,
+               size_t count, loff_t *ppos)
+{
+       char state = 0;
+       int error;
+
+       if (copy_from_user(&state, buf, 2))
+               return -EFAULT;
+       if (state == '1')
+               state = 1;
+       else
+               state = 0;
+       error = rtas_call(set_indicator, 3, 1, NULL,
+                         IBM_SYS_IDENTIFICATION, 0, state);
+       if(error != 0)
+       {
+               printk(KERN_WARNING "rtas-proc.o: sysident write failed\n");
+               return -EFAULT;
+       }
+
+       return count;
+}
+/* ********************************************************************** */
+static ssize_t ppc_rtas_sysident_read(struct file * file, char * buf,
+               size_t count, loff_t *ppos)
+{
+       int n, sn;
+       unsigned long ret;
+       int state, error;
+       char tmpbuf[4];
+
+       error = rtas_call(get_sensor_state, 2, 2, &ret,
+                         IBM_SYS_IDENTIFICATION, 0);
+       state = (int) ret;
+       n = snprintf(tmpbuf, 4, "%d\n", state);
+       sn = n + 1;
+       if (*ppos >= n)
+               return 0;
+
+       if (n > count)
+               n = count;
+       if (n > sn - *ppos)
+       n = sn - *ppos;
+       if (copy_to_user(buf, tmpbuf + (*ppos), n))
+               return -EFAULT;
+
+       *ppos += n;
+       return n;
+}
+
+
+/* ********************************************************************** */
 /* SENSOR STUFF                                                           */
 /* ********************************************************************** */
```

```
 static int ppc_rtas_sensor_read(char * buf, char ** start, off_t off,
@@ -632,6 +775,8 @@
        const char * battery_charging[]    = { "Charging", "Discharching", "No
current flow" };
        const char * ibm_drconnector[]     = { "Empty", "Present" };
        const char * ibm_intqueue[]        = { "Disabled", "Enabled" };
+       const char * sysfault_indicator[]    = { "Normal", "Fault" };
+       const char * sysident_indicator[]    = { "Normal", "On" };

        int temperature = 0;
        int unknown = 0;
@@ -715,6 +860,16 @@
                        value_arrsize = sizeof(ibm_intqueue)/sizeof(char *);
                        value_arr = ibm_intqueue;
                        break;
+               case IBM_SYSFAULT_INDICATOR:
+                       label_string = "System Fault indicator:";
+                       value_arrsize = sizeof(sysfault_indicator)/sizeof(char
*);
+                       value_arr = sysfault_indicator;
+                       break;
+                case IBM_SYS_IDENTIFICATION:
+                       label_string = "Identification light:";
+                       value_arrsize = sizeof(sysident_indicator)/sizeof(char
*);
+                       value_arr = sysident_indicator;
+                       break;
                default:
                        n += sprintf(buf+n,  "Unkown sensor (type %d), ignoring
it\n",
                                             s.token);
```

Függelék G. - OpenFirmware device-tree - 6F1

```
00c654b0: /cpus
00c66738:   /PowerPC,RS64-III@0
00c66c68:     /l2-cache
00c67790:   /PowerPC,RS64-III@2
00c67cc0:     /l2-cache
00c68128: /chosen
00c68290: /memory@0
00c69428:   /IBM,memory-module@0
00c696d8:   /IBM,memory-module@1
00c69988:   /IBM,memory-module@2
00c69c38:   /IBM,memory-module@3
00c69ee8:   /IBM,memory-module@4
00c6a198:   /IBM,memory-module@5
00c6a448:   /IBM,memory-module@6
00c6a6f8:   /IBM,memory-module@7
00c6a9a8: /memory-controller@fffe000000
00c6ab48: /nvram@febd800000
00c6b088: /options
00c6bc88: /aliases
00c6e720: /packages
00c6e7c8:   /disassembler
00c73e78:   /assembler
00c85490:   /dev-tree
00c85be0:   /deblocker
00c86a08:   /disk-label
00c8a2e8:   /tape-label
00c8a568:   /ip
00c90618:   /obp-tftp
00c9c540:   /prep-boot
00c9cb70:   /fat-files
00c9ea58:   /iso-13346-files
00ca7ac0:   /utilities
00cb2b10:   /net
00cb4760:   /iso-9660-files
00cb58e0:   /boot-mgr
00ccb610:   /chrp-loader
00ccb7a0:   /pe-loader
00ccbfc0:   /elf-loader
00cce1f0:   /terminal-emulator
00cce288:   /dynamic-reconfig
00d5d650:   /gui
00de3df0:   /post
00cce6e0: /reserved@f4000000
00ccfa30: /openprom
00ccfec0: /event-sources
00ccff28:   /epow-events
00ccffb8:   /internal-errors
00cd0048: /interrupt-controller@ffff100000
00cd0ac8: /interrupt-controller@fff7f00140
00cd0cd0: /interrupt-controller@fff7f09400
00cd0f08: /interrupt-controller@fff7f0a400
00cd1140: /rtas
00ce3470: /pci@fff7f09000
00ce6ff0:   /isa@10
00ce90c8:     /reserved@i78
00ce91d0:     /rtc@i70
00ce9dc8:     /parallel@i378
00cea4a8:     /serial@i3f8
00ceb7a8:     /serial@i2f8
```

```
00cecaa8:     /serial@i898
00cedda8:     /serial@i890
00cef0a8:     /8042@i60
00cf09b8:       /keyboard@0
00cf4a18:       /mouse@1
00cf5528:     /fdc@i3f0
00cf9840:       /disk@0
00cfa038:     /timer@i40
00cfa690:     /interrupt-controller@i20
00cfa928:     /dma-controller@i0
00cfaf30:   /pci@b
00d3cdc0:     /scsi@1
00d43350:       /sd
00d444b8:       /st
00d45a00:     /scsi@1,1
00d4bf90:       /sd
00d4d0f8:       /st
00cff0d8:   /pci@b,2
00d03678:   /pci@b,4
00d07c18:   /pci@b,6
00d33290:   /IBM,sp@11
00d0c658: /pci@fff7f0a000
00d0f798:   /pci@b
00d4e668:     /ethernet@1
00d13cc0:   /pci@b,2
00d18260:   /pci@b,4
00d1c800:   /pci@b,6
00d58fc8:     /display@1
00d20da0:   /pci@c
00d25350:   /pci@c,2
00d29900:   /pci@c,4
00d2deb0:   /pci@c,6
```