Energiakezelés linuxos rendszerekben

Az energiakezelés megvalósítása minden rendszerben nehéz feladat. Az alábbiakban áttekintjük, hogyan tudjuk kezelni gépünk normál üzemi és energiatakarékos állapot közötti átmeneteit.

Az energiakezelő programok az akkumulátoros gépeknél – hordozható számítógépek, tenyérgépek – szinte létfontosságúak, segítségükkel ugyanis takarékoskodhatunk az energiával, ha az illető eszközt nem használjuk folyamatosan. Egyszerű példaként a kijelző megadott, használat nélkül eltelt idő utáni kikapcsolásának lehetőségét említeném. Ilyen módszerekkel élve meghosszabbíthatjuk az akkumulátoros üzemidőt, vagyis hosszabb ideig dolgozhatunk a géppel, mielőtt újra kellene töltenünk az akkumulátort.

Az eszközoldal támogatása az energiakezelés működéséhez elengedhetetlen; ha ez megvan, akkor a programoldal feladata a lehetőségek okos kihasználása. Az energiakezelés támogatásának szintje minden eszköznél más és más. Akadnak olyan készülékek, amelyek csak kétféle állapotot ismernek, a be- és a kikapcsoltat. Más eszközök, például az SA1110 processzor összetettebb energia-megtakarítási lehetőségeket is kínálhatnak, mint például az órajel változtatása.

Az energiakezelés megvalósítása minden rendszernél bonyolult művelet, ugyanis számos, egymással kapcsolatban nem álló alrendszer működését kell ugyanolyan irányvonalak szerint összhangba hozni. Írásomban az energiakezelés Linux (2.4.x) alatti működését és a fejlett energiakezelést támogató, akkumulátoros működésű készülékeken történő megvalósítását tekintem át, az illesztőprogramok és az alkalmazások szintjére egyaránt kitérve.

A két energiakezelési szabvány

A számítógéprendszerek energiakezelése az évek során egyre kifinomultabb lett, eme folyamat során több szabvány is született. A két legnépszerűbb ezek közül az APM (Advanced Power Management, azaz fejlett energiakezelés) és az ACPI (Advanced Configuration and Power Interface, vagyis fejlett beállítás- és energiafelület). Az APM-et a Microsoft és az Intel vezette be, az energiakezelés támogatását egy vagy több programréteg segítségével valósítja meg.

A rétegek közötti adatcserét szabványosították. Az APM modellben a BIOS kulcsszerepet játszik. A két említett megoldás közül az ACPI az újabb, fejlesztésében a Toshiba, az Intel és a Microsoft vett részt. Az ACPI intelligensebb energiakezelést tesz lehetővé, amelyet inkább az operációs rendszer végez, nem annyira a BIOS. Ugyan ez a két szabvány elsősorban x86 alapú gépekben terjedt el, más géptípusokon is gond nélkül megvalósíthatók.

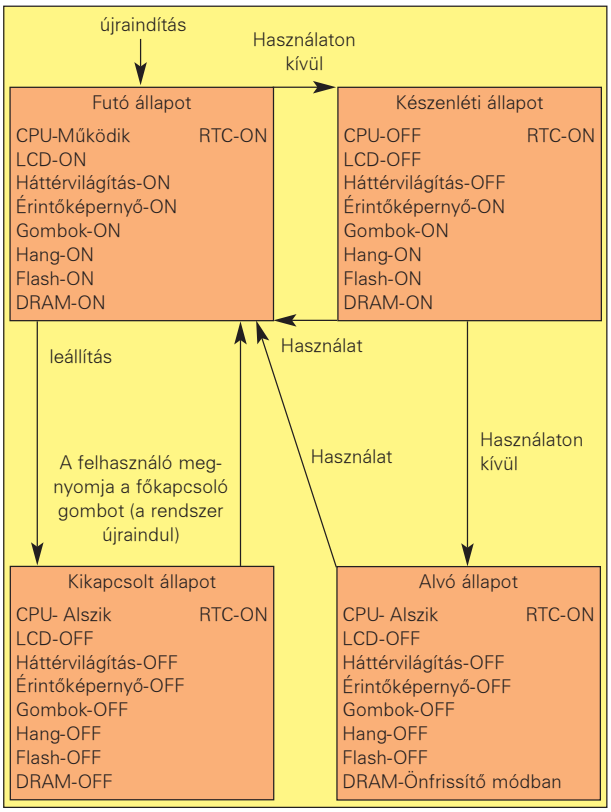
Az energiakezelés megvalósítása

Az energiakezelő eljárások megvalósítása előtt tudnunk kell, hogy az eszközoldal egyáltalán milyen szolgáltatások támogatására képes. Az energiakezelő program egyik legfontosabb feladata az, hogy minden eszközt olyan állapotban tartson, amely a lehető legkisebb energiafelhasználással jár.

Az energiakezelés kérdéskörének egyik lehetséges megközelítésénél a megvalósítást egy energiahasználati–állapot-átmeneti folyamatábra felrajzolásával kezdjük. Ebben többféle energiahasználati rendszerállapotot határozunk meg, illetve az állapotok közötti átmeneteket kiváltó szabályokat és eseményeket is megadjuk.

Példaként vegyünk egy tenyérgépet, amelyben a következő eszközök találhatók: Intel SA1110 processzor, valós idejű óra, DRAM, flash, LCD, előlap megvilágítása, UART, hangkodek, érintőképernyő, gombok és főkapcsoló. Az Intel SA1110 processzor több energiamegtakarító szolgáltatást is támogat, ilyen például az órajel programból megvalósított változtatásának lehetősége. Az órajel csökkentésével a processzor energia-felvétele is csökken – igaz, a teljesítménye is. A processzor többféle működési módot is ismer:

- Futó mód: az SA1110 normál állapota, amelyben valamilyen programkódot futtat. Minden energiaforrás engedélyezett, minden órajel él, a lapka minden erőforrása rendelkezésre áll.
- Tétlen mód: lehetővé teszi, hogy a processzort programból leállítsuk, ha éppen nincsen rá szükség. Ilyenkor a processzor órajele megszűnik, ezzel valamennyi energia-megtakarítás elérhető. A lapka összes többi erőforrása aktív marad. Ha megszakítás érkezik, a processzor újra bekapcsol.
- Alvó mód: a legnagyobb mértékű energia-megtakarítási lehetőséget nyújtja, amely egyben a legtöbb szolgáltatás kikapcsolásával is jár. Ebben a módban a processzor túlnyomó részének tápellátása megszűnik. Alvó módból a processzor előre beprogramozott események hatására lép ki, ilyen például a főkapcsoló gomb megnyomása.



1. ábra Energiakezelési állapotátmeneti folyamatábra

Mint látható, tétlen vagy alvó módba a programoldal kapcsolhatja a processzort.
Az ilyen tenyérgépekben a DRAM-cellákat a processzorban található memóriavezérlő egység rendszeresen frissíti. Alvó állapotban azonban a processzor nagy része kikapcsol, vagyis a DRAM-cellák frissítése megszűnik, ez tartalmuk elvesztését okozza. Az adatvesztés elkerülése érdekében a legtöbb DRAM támogatja az úgynevezett önfrissítő módot, amelyben a DRAM önmaga gondoskodik saját celláinak frissítéséről. Ha ez a helyzet, akkor a DRAM – ugyancsak programból megvalósítottan, néhány vezérlőregiszter tartalmának megváltoztatásával – a processzor alvó módba kapcsolása előtt önfrissítő módba állítható, és ezzel megőrizhető a tartalma.
Tenyérgépünk fő energiafogyasztó egységei a processzor, a memória és a kijelző háttérvilágítása lesznek. Ezeket kell tehát a lehető leginkább energiatakarékos állapotban tartani.
Az *1. ábrán* a tenyérgéphez látható egy lehetséges energiahasználati–állapotátmeneti folyamatábra. Röviden tekintsük át az energiaállapotokat:

- Futó állapot: a rendszer indítás után ebbe az alapértelmezett állapotba jut. Az energiafogyasztás ebben az állapotban a legnagyobb, hiszen minden eszköz aktív vagy bekapcsolt.
- Készenléti állapot: a rendszer akkor lép ebbe az állapotba, ha megadott ideig nem használják. Az LCD és a háttérvilágítás kikapcsol, a processzor órajele csökken, ezzel bizonyos mértékű energia-megtakarítás érhető el.
- Alvó állapot: a rendszer akkor lép tovább ebbe az állapotba, ha hosszabb időn keresztül nem használják. Az energia-megtakarítás érdekében határozottabb lépések történnek, a processzort ugyanis alvó módba állítjuk, ekkor viszont az egyéb eszközök túlnyomó része is lekapcsol. A DRAM önfrissítő módba lép, ezzel biztosítja a gép állapotának, vagyis a memóriába írt adatok megőrzését. Alvó állapotból a gép előre beprogramozott események hatására léphet ki. Ekkor futó állapotba kapcsol, és helyreállítja saját állapotát.
- Kikapcsolt állapot: a gép a `shutdown` parancs hatására jut ebbe az állapotba. A kikapcsolt állapotból történő kilépéskor sor kerül a rendszer újratöltésére. Ez azt jelenti, hogy nincs szükség a gép állapotának, azaz a DRAM tartalmának megőrzésére, vagyis a memória kikapcsolható. Kikapcsolt állapotban a legkisebb az energiafogyasztás.

A valós idejű óra a rendszeridő pontos értéken tartása érdekében minden rendszerállapotban működik.
A folyamatábra alapján bátran kijelenthetjük, hogy a használaton kívüliség felismerése és az eszközök alacsony energiafogyasztású állapotba kapcsolása jelenti az energiakezelő program szívét.

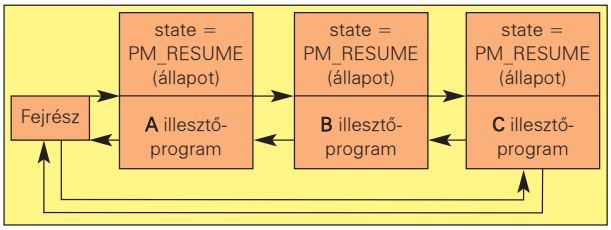
A Linux és az energiakezelés

Az energiakezelő program az állapotátmeneteket az illesztőprogramokkal és az alkalmazásokkal együttműködve vezérli. Minden energiakezelési eseményről tud, ide értve az állapotátmeneteket, az alvó módba kapcsolást és az akkumulátor lemerülését is. Ennek köszönhetően a programoldalnak módja van az állapotátmenetek megakadályozására, ha valamiért veszélyesnek ítéli meg végrehajtásukat.
Az illesztőprogramok általában azért is felelősek, hogy alacsonyabb energiafogyasztású állapotukba kapcsolásuk előtt mentsék az eszközök állapotát, a rendszer újraaktiválásakor pedig helyre is állítsák.
Az alkalmazások általában nem szólnak bele az energiakezelési állapotátmenetekbe. Előfordulhat azonban, hogy olyan, az eszközöket közvetlenül használó programot futtatunk, amely valami miatt mégis részt szeretne venni az átmenetekről szóló döntések meghozatalában. Ebben a részben áttekintjük, mire van szüksége egy illesztőprogramnak ahhoz, hogy az energiakezelésbe beleszólhasson:

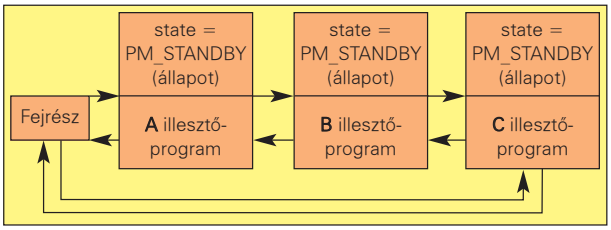
- `pm_dev` adatszerkezet: a Linux-rendszermag energiakezelő alrendszere a bejegyzett illesztőprogramok számára bizonyos adatokat a `pm_dev` nevű adatszerkezetbe ír be, így tudja értesíteni őket az energiakezelési eseményekről.
- `pm_register`: az illesztőprogramoknak először be kell jegyezniük magukat az energiakezelő alrendszernél, az energiakezelésben csak ezt követően vehetnek részt. Ezt a `pm_register` függvény meghívásával tehetik meg:

```
struct pm_dev *pm_register(pm_dev_t type, unsigned
        ➥long id, pm_callback cbackfn);
```

A `type` az illesztőprogram által kezelt eszköz típusát adja meg, az `id` az eszköz azonosítóját, a `cbackfn` pedig egy mutató az illesztőprogram visszahívó függvényére.



2. ábra A rendszer futó állapota



3. ábra A rendszer készenléti állapota

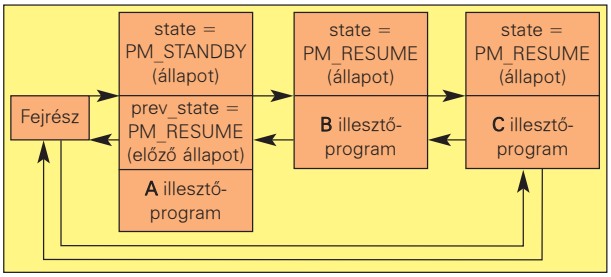
Az illesztőprogramokban használható típusokat és azonosítókat a ***linux/pm.h*** fájlban lehet megtalálni. Sikeres futás esetén a `pm_register` visszatérési értéke egy `pm_dev` adatszerkezetre hivatkozó mutató. Az illesztőprogram visszahívó függvényét az energiakezelő alrendszer hívja meg energiakezelési esemény esetén. A függvénynek átadott értékek a következők:

- `dev`: mutató az eszközt képviselő `pm_dev` adatszerkezetre, megegyezik a `pm_register` által visszaadott mutatóval.
- `event`: az energiakezelési esemény típusát adja meg. A lehetséges események a következők:
  - `PM_STANDBY` – a rendszer készenléti állapotba fog váltani;
  - `PM_SUSPEND` – a rendszer felfüggesztett, alvó állapotba kapcsol;
  - `PM_RESUME` – a rendszer visszatér üzemi állapotba, akár készenléti, akár alvó állapotból. Megvalósítástól függően akár több esemény is megadható.
- `data`: a kéréshez tartozó adat, ha van ilyen.

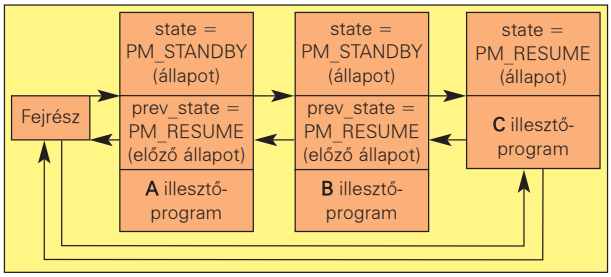
Minden illesztőprogramtól azt várjuk, hogy az energiakezelési esemény típusától függően elvégezzen valamilyen műveletet. A `PM_SUSPEND` esemény hatására, például az LCD illesztőprogramjának mentenie kell az eszközállapotot, majd ki kell kapcsolnia az LCD-t. A `PM_RESUME` eseménynél az LCD illesztőprogram feladata az LCD bekapcsolása és a mentett állapot visszaállítása.
A visszahívó függvénynek egy egész (integer) értékkel kell visszatérnie. A nulla visszatérési érték azt jelzi, hogy az illesztőprogram elfogadta az energiakezelési eseményt. Ha a visszatérési érték nem nulla, akkor az illesztőprogram elutasította az energiakezelési eseményt. Ilyenkor előfordulhat, hogy az állapotátmenet meghiúsul. Ha például az LCD illesztőprogram energiakezelő visszahívó függvénye egy `PM_SUSPEND` eseményről kap jelzést, visszatérési értéke pedig 1, akkor a felfüggesztett állapotba váltás el fog maradni. Az illesztőprogramok visszahívó függvényeinek meghívása előre meghatározott sorrendben történik. A sorrend felállítása egyszerű, „később jött, előbb szolgáljuk ki” elv alapján történik, ami viszont az eszközök egymástól való függése esetén okozhat gondot. Vegyünk példának egy Bluetooth eszközt, amelynek csatolófelületét egy USB állomásvezérlőn keresztül érjük el. A Bluetooth illesztőprogramnak szüksége van erre a felületre, nélküle nem tud kapcsolatba lépni a Bluetooth eszközzel. A függés miatt az USB állomásvezérlő illesztőprogramja a Bluetooth illesztőprogram előtt kerül betöltésre. Vagyis az USB állomásvezérlő előbb jegyzi be magát az energiakezelési eseményekre, mint a Bluetooth illesztőprogram.
Ha a rendszer alvó állapotba szeretne lépni, egy `PM_SUSPEND` kérést küld először a Bluetooth illesztőprogramnak, majd az USB illesztőprogramnak. Az USB állomásvezérlő a `PM_SUSPEND` feldolgozásának részeként gond nélkül lekapcsolhatja a Bluetooth kaput. Amikor viszont a rendszer normál állapotba vált vissza, a `PM_RESUME` esemény először a Bluetooth illesztőprogramhoz jut el, az USB állomásvezérlő illesztőprogramjának értesítése csak ezt követően történik meg. Amikor a Bluetooth illesztőprogram feldolgozza, pontosabban feldolgozná az eseményt, akkor az eszköz felé vezető csatolófelület még nem áll rendelkezésre, vagyis a Bluetooth eszköz felébresztése sikertelen lesz. Az ilyen helyzeteket a legegyszerűbben oly módon lehet elkerülni, hogy a `PM_RESUME` eseményekről „elsőként érkezett, elsőként szolgáljuk ki” sorrend szerint küldjük ki az értesítéseket.
A `pm_unregister` függvény meghívásával a megadott illesztőprogramot kivehetjük az energiakezelésben résztvevők listájából:

```
pm_unregister(pm_callback cbackfn);
```

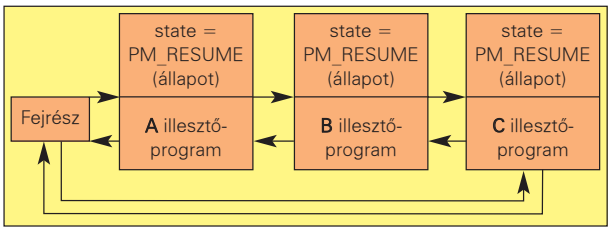
A bejegyzés törléséhez ugyanazt a mutatót kell átadott értékként használni, mint amit a bejegyzésnél is megadtunk. Ha egy illesztőprogram törölte saját bejegyzését, az energiakezelő alrendszer többé nem értesíti az energiakezelési eseményekről.
A Linux az illesztőprogramok számára két felületet is megad, ezek a `pm_access` és a `pm_dev_idle`. A `pm_access` meghívását az eszköz elérése előtt kell megejteni, a `pm_dev_idle` pedig akkor jut szerephez, ha egy eszközt nem használunk. Ugyanakkor vegyük figyelembe, hogy ezeket a felületeket nem minden géptípuson lehet megvalósítani.
Példaként lássunk egy átlagos állapotátmenetet, amelyben csak illesztőprogramok jutnak szerephez. Az energiakezelő alrendszer a nála bejegyzett illesztőprogramokat egy kétszeresen láncolt, vagyis körkörösen bejárható listában tartja nyilván. A 2. *ábrán* a lista három illesztőprogram bejegyzését követő állapotát próbáltam szemléltetni. Feltételezzük, hogy elsőként a C, majd a B és végül az A illesztőprogram jegyezte be magát.
Tegyük fel, hogy most a rendszer futó állapotból készenléti állapotba vált. Az energiakezelő alrendszer `PM_STANDBY` kérést küld mindhárom illesztőprogramnak. Kétféle eredményre juthatunk. Az egyik eset az, hogy mindegyik illesztőprogram elfogadja a kérést, és a rendszer készenléti állapotba lép. A második lehetőség az, hogy valamelyik



4. ábra Az A illesztőprogram elfogadta a PM_STANDBY kérést



5. ábra Az A és a B illesztőprogram elfogadta a PM_STANDBY kérést



6. ábra A C illesztőprogram elutasította a PM_STANDBY kérést, ezért a rendszer újra futó állapotba váltott

illesztőprogram, esetleg több is, elutasítja a kérést. Ilyenkor az állapotváltás sikertelen lesz, a rendszer futó állapotban marad.

A *3. ábrán* látható állapot akkor alakul ki, ha mindegyik illesztőprogram elfogadja a `PM_STANDBY` kérést. Vegyük észre, hogy a `pm_dev` adatszerkezetek állapot- (`state`) mezőjének értéke a kérés elfogadását követően megváltozott.

Most vegyük azt az esetet, amikor az A és a B illesztőprogram elfogadja a `PM_STANDBY` kérést, a C viszont nem.

A *4. ábrán* az az állapot látható, amikor az A illesztőprogram már teljesítette a kérést. Miután az A illesztőprogram jelezte beleegyezését, a B illesztőprogram is megkapja a `PM_STANDBY` kérést.

Az *5. ábrán* az azt követő állapot szerepel, hogy a B illesztőprogram is jelezte egyetértését. Ekkor az A és a B eszköz már készenléti állapotban van, a C viszont még futóban.

A következő művelet a `PM_STANDBY` kérés elküldése a C illesztőprogramnak, amely viszont elutasítja őt. Ekkor az állapotváltást vissza kell vonni. Mivel az A és a B eszköz már készenléti állapotban áll, az energiakezelő alrendszernek esetükben vissza kell vonnia a műveletet, vagyis küld nekik – először a B majd az A illesztőprogramnak – egy `PM_RESUME` kérést. A visszavonás elvégzése után újra futó módban lesz az összes eszköz, s ez az állapot a *6. ábrán* látható.

APM

A *7. ábrán* az APM modell látható. Fontosabb összetevői a következők:

- APM BIOS: programfelület az alaplaphoz és energiakezelésre képes eszközeihez és összetevőihez. Ez a rendszer energiakezelő programrendszerének legalacsonyabb rétege.
- APM illesztőprogram: az APM megadott operációs rendszeren történő megvalósításáért felelős.
- APM támogatással rendelkező illesztőprogramok és alkalmazások: az APM illesztőprogram minden energiakezelési eseménynél velük lép kapcsolatba.

Az APM BIOS észleli és jelenti a különféle energiakezelési eseményeket, mint például az akkumulátor lemerülését, az áramforrás megváltozását, a rendszer készenléti állapotba való be- vagy onnan kilépését stb. Az APM illesztőprogram rendszeresen lekérdezi az APM BIOS-tól az energiakezelési eseményekre vonatkozó adatokat, majd az APM-képes illesztőprogramokkal és alkalmazásokkal együttműködve dolgozza fel őket.

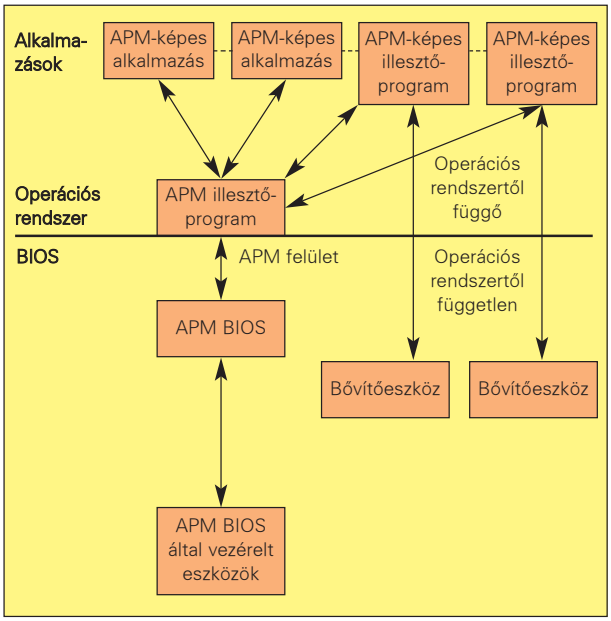
A Linux APM illesztőprogramja két felületet tesz elérhetővé az alkalmazások számára. Az első a */proc/apm*, amely a rendszer áramforrásáról – hálózati forrás vagy akkumulátor – tájékoztat. Ha a gép akkumulátorról üzemel, akkor az akkumulátor töltöttségi szintjéről és a teljes lemerüléséig hátralévő időről is tájékoztat. A második felület a */dev/apm-bios*, amelynek segítségével az alkalmazások értesülhetnek az energiakezelési eseményekről, illetve részt is vehetnek bennük. Az alkalmazások a megfelelő `ioctl` hívások révén maguk is kezdeményezhetnek állapotátmeneteket. A fájlra vonatkozó olvasási hívások mindaddig blokkolva maradnak, amíg valamilyen energiakezelési esemény nem történik. Az olvasási hívás visszatérésekor a következőként bekövetkező energiakezelési eseményről kapunk tájékoztatást.

Előfordulhat, hogy rendszerünkön a */dev/apm_bios* fájlt megnyitó alkalmazások egy része rendszergazdai jogosultsággal fut. Ezeket az alkalmazásokat az APM illesztőprogram kiemelten kezeli. Az események egy részéről, mint az alvó vagy készenléti állapotba váltás, az APM illesztőprogram minden olyan alkalmazást értesít, amely megnyitotta a */dev/apm_bios* fájlt. A rendszergazdai jogosultságokkal futó alkalmazásoktól visszaigazolást is vár, a tényleges állapotátmenetre csak a beleegyező üzenetek beérkezése után kerül sor. Egyetértésüket az alkalmazások az e célra szolgáló `ioctl` hívásokkal fejezhetik ki.

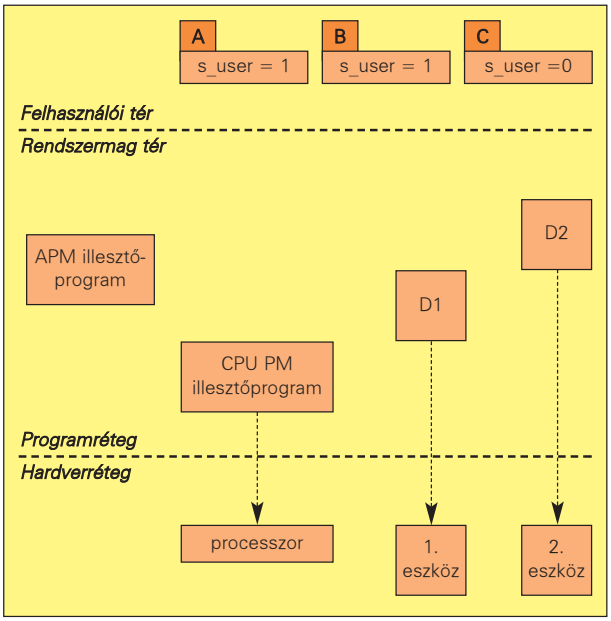
Normál esetben a következő `ioctl` hívások támogatása valósul meg:

- `APM_IOC_STANDBY`: készenléti állapotba lépteti a rendszert.
- `APM_IOC_SUSPEND`: felfüggesztett állapotba lépteti a rendszert.

Az `APM` két, felhasználói térben futó segédprogramot is rendelkezésünkre bocsát. Az `apm` parancs a rendszermag `APM` alrendszerével tart kapcsolatot. A neki átadott értékektől függően képes a rendszer energiaállapotának meg-



7. ábra Az APM modell



8. ábra Példa az állapotátmenet végrehajtására

jelenítésére, de arra is használható, hogy a felhasználó készenléti vagy felfüggesztett állapotba léptesse a gépet. Az `apmd` démon a különféle energiakezelési eseményeket jelenti és dolgozza fel, valamint minden energiakezeléssel kapcsolatos eseményt naplóz a */var/log/messages* fájlba. A naplózás mellett az `apmd` a különféle energiakezelési események hatására bizonyos műveleteket is képes végrehajtani, ezek megadására egy parancsfájl, általában az *apmd_proxy* fájl szolgál. A parancsfájlt az `apmd` démon hívja meg, és egy vagy két átadott érték segítségével közli vele, hogy milyen esemény fog bekövetkezni. Nézzünk egy példaparancsfájlt:

```
case 1:2 in

"standby":*)
     #A rendszer használaton kívül van, ezért
     #készenléti
     #állapotba vált. Csökkentjük a processzor
     #sebességét.
     echo 162200 > /proc/sys/cpu/0/speed
     ;;

"resume":"standby")
     #A rendszert használni akarjuk, ezért
     #készenléti állapotból futó
     #állapotba hozzuk. Növeljük a processzor
     #sebességét.
     echo 206400 > /proc/sys/cpu/0/speed
     ;;

"suspend":*)
     #A rendszer felfüggesztett állapotba lép.
     #Leállítjuk a hálózati csatolót.
     ifconfig eth0 down
     ;;

"resume":"suspend")
     #A rendszer kilép a felfüggesztett állapotból.
     #bekapcsoljuk a hálózati csatolót
     #és növeljük a processzor sebességét.
     ifconfig eth0 up
     echo 206400 > /proc/sys/cpu/0/speed
     ;;
```

Példa az energia-állapotátmenetre

Az állapotátmenetekkel kapcsolatosan homályosnak tűnő részek azonnal megvilágosodnak, ha veszünk egy illesztőprogramokat és alkalmazásokat egyaránt tartalmazó példát. Tegyük fel, hogy a rendszerben két illesztőprogram található, D1 és D2, mindkettő energiakezelésre bejegyezve, valamint a */dev/apm_bios* megnyitásával három alkalmazás (A, B és C) is résztvesz a folyamatokban. Az A és a B alkalmazás rendszergazdai (superuser, s_user) jogosultsággal fut, a C pedig normál felhasználóként.
A *8. ábrán* ez a felállás látható.
Vegyük most azt az esetet, amikor a rendszer futó állapotból alvó állapotba akar váltani. A szükséges lépések sora az alkalmazásoknak (A, B és C) a végrehajtandó átmenetről való értesítésével kezdődik. Így az alkalmazások elvégezhetik az általuk az átmenethez szükségesnek vélt műveleteket. Mivel az A és a B alkalmazás rendszergazdai jogosultsággal fut, a további lépések megtétele előtt meg kell várni, hogy beleegyeznek-e az átmenet végrehajtásába. Amikor az A és a B alkalmazás végzett mindazzal, amit az alvó állapotba való átlépés előtt végre akart hajtani, engedélyező jelzést küld az APM illesztőprogramnak.
Az APM illesztőprogram most készen áll a rendszer alvó állapotba kapcsolására. `PM_SUSPEND` üzenetet küld D1-nek és D2-nek. D1 és D2 az általa kezelt eszközt alvó állapotba kapcsolja, majd nyugtázó jelzést küld az APM-nek. Miután D1 és D2 végzett a szükséges műveletekkel, az APM jelzi a processzor energiakezelő illesztőprogramjának, hogy a pro-

cesszort alvó állapotba lehet kapcsolni. Amikor ez is megtörtént, akkor a rendszer alvó állapotba kapcsolásának folyamata lezárult.

Összegzés

Ugyan az APM alkalmazásának vannak bizonyos hátrányai is, egyszerűségéből fakadóan gyakorlatilag bármilyen készülékben megvalósítható. Más szabványok, például az ACPI, magasabb fokú energiakezelést tesznek lehetővé, ám ennek a lehetőségnek a növekvő bonyolultság az ára. Fontos az is, hogy minden illesztőprogram és alkalmazás helyesen végezze az energiakezelési műveleteket, ellenkező esetben akár egyetlen illesztőprogram is megakadályozza például azt, hogy a gép felfüggesztett állapotba lépjen. Ugyanakkor hibátlan megvalósítással az energiakezelés jelentős mértékben hozzájárulhat a gép hatékonyabb, energiatakarékosabb működéséhez.

Linux Journal 2004. március, 119. szám

Srivatsa Vaddagiri (vsrivatsa@in.ibm.com)
1996 óta dolgozik az IBM India alkalmazásában. Számos különböző tervezetben vett részt, elsősorban Unix alapú rendszerekkel foglalkozik. Jelenleg a Linux alapú tenyérgépek energiakezelés-támogatásán dolgozó beágyazott Linuxot fejlesztő csoport kiemelkedő tagja.

Anand K. Santhanam (asanthan@in.ibm.com) 1999-ben került az IBM Global Services (Software Labs) indiai csapatába. Az IBM Linux Group tagja, ahol leginkább illesztőprogramokkal, ARMLinuxszal és a beágyazott rendszerek energiakezelésével foglalkozik.

Vijay Sukthankar (vksuktha@in.ibm.com) 1994-ben került az IBM-hez. Jelenleg a Linux Competency Center vezetője, emellett egyéb Linux alapú, nyílt forrású fejlesztéseket is irányít. Az IBM különféle munkacsoportjaiban számos beágyazott Linux alapú szolgáltatás fejlesztésébe kapcsolódott be.

Murali Iyer (mniyer@us.ibm.com) 1995 óta dolgozik az IBM-nél, a vállalatnak több laboratóriumában is megfordult világszerte. 2000. óta Linux alapú beágyazott rendszerek tervezésével foglalkozik. Többek közt felső kategóriájú tenyérgépek fejlesztésében és szívritmus-szabályzó készülékek programozásában vett részt.

FORRÁSOK

APM szabvány
A rendszermag forrásának *Documentation/pm.txt* állománya
Intel SA1110 Advanced Developers kézikönyv