Operációs rendszerek

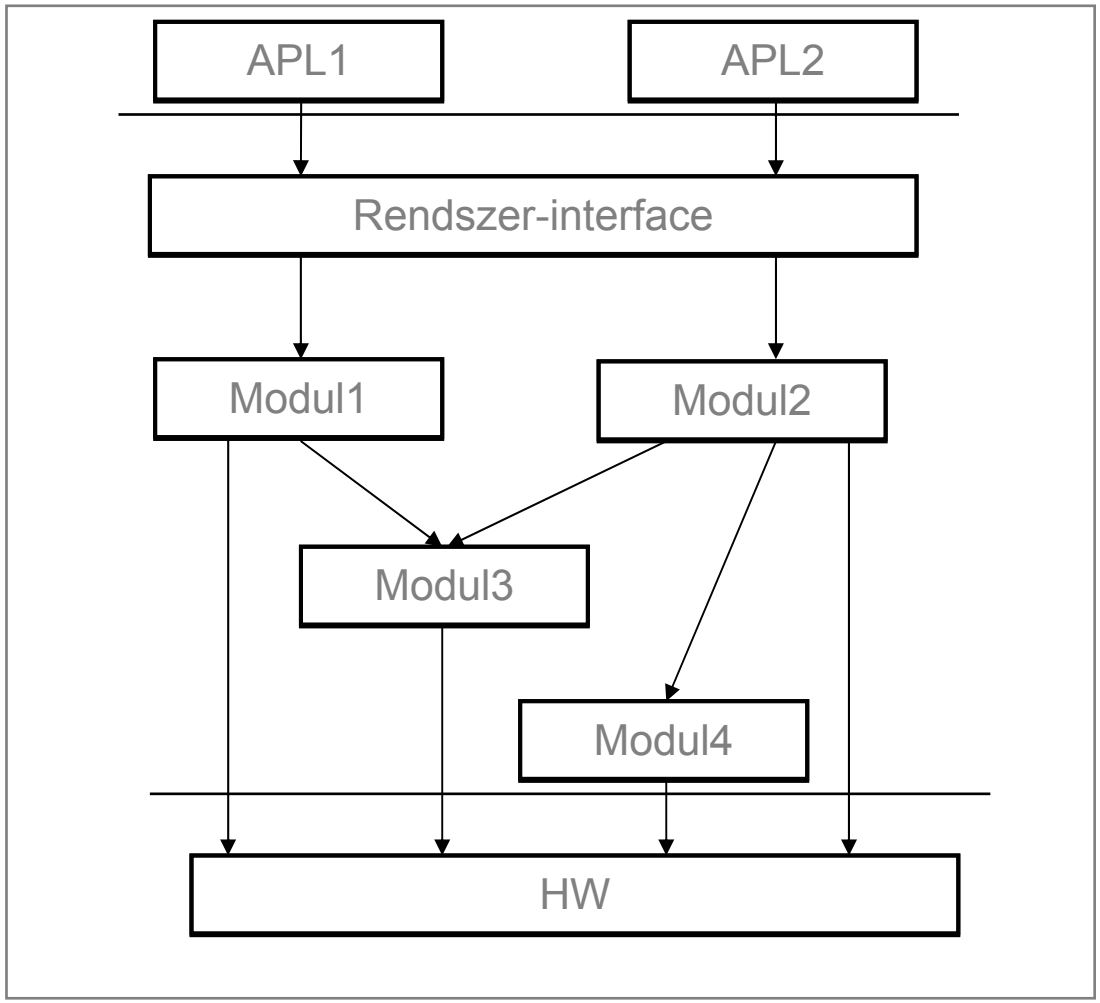
A Windows NT felépítése

A Windows NT

- 1996: NT 4.0 .
- Felépítésében is új operációs rendszer: *New Technology* (*NT*).
- 32-bites Windows-os rendszerek felváltása.
- Windows 2000: *NT* alapú.

Operációs rendszerek felépítése

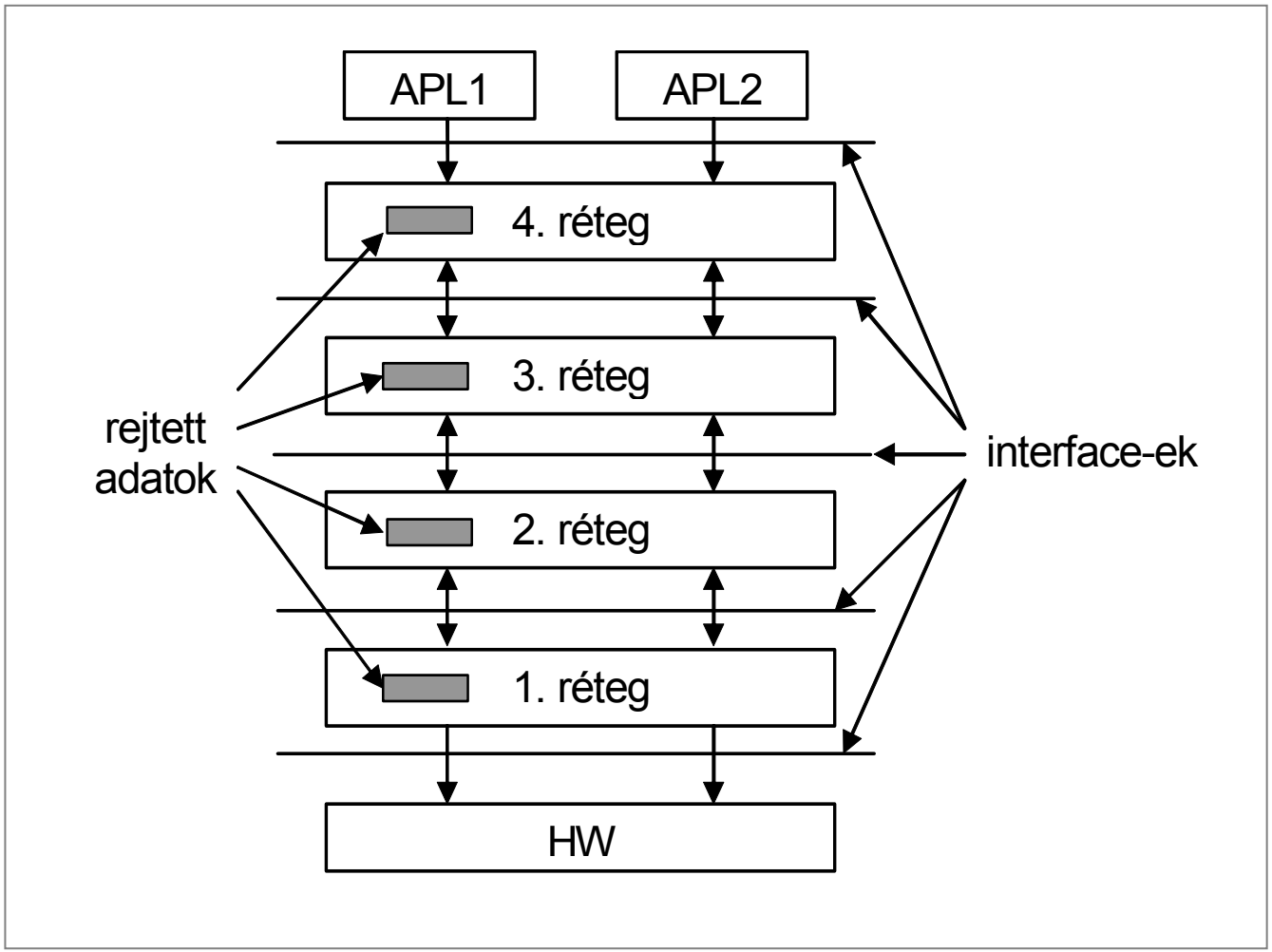
- Monolitikus kernel



Monolitikus kernel

- Közvetlen kommunikáció modulok között.
- ”Közösen" használt adatok a közös memórián.
- Nincs szabványos érintkezési felülete (interface) a moduloknak.
- Nem megbízható:
  egy hibás modul hibája továbbterjedhet a többi modul felé.
- Nehezen karbantartható, fejleszthető.

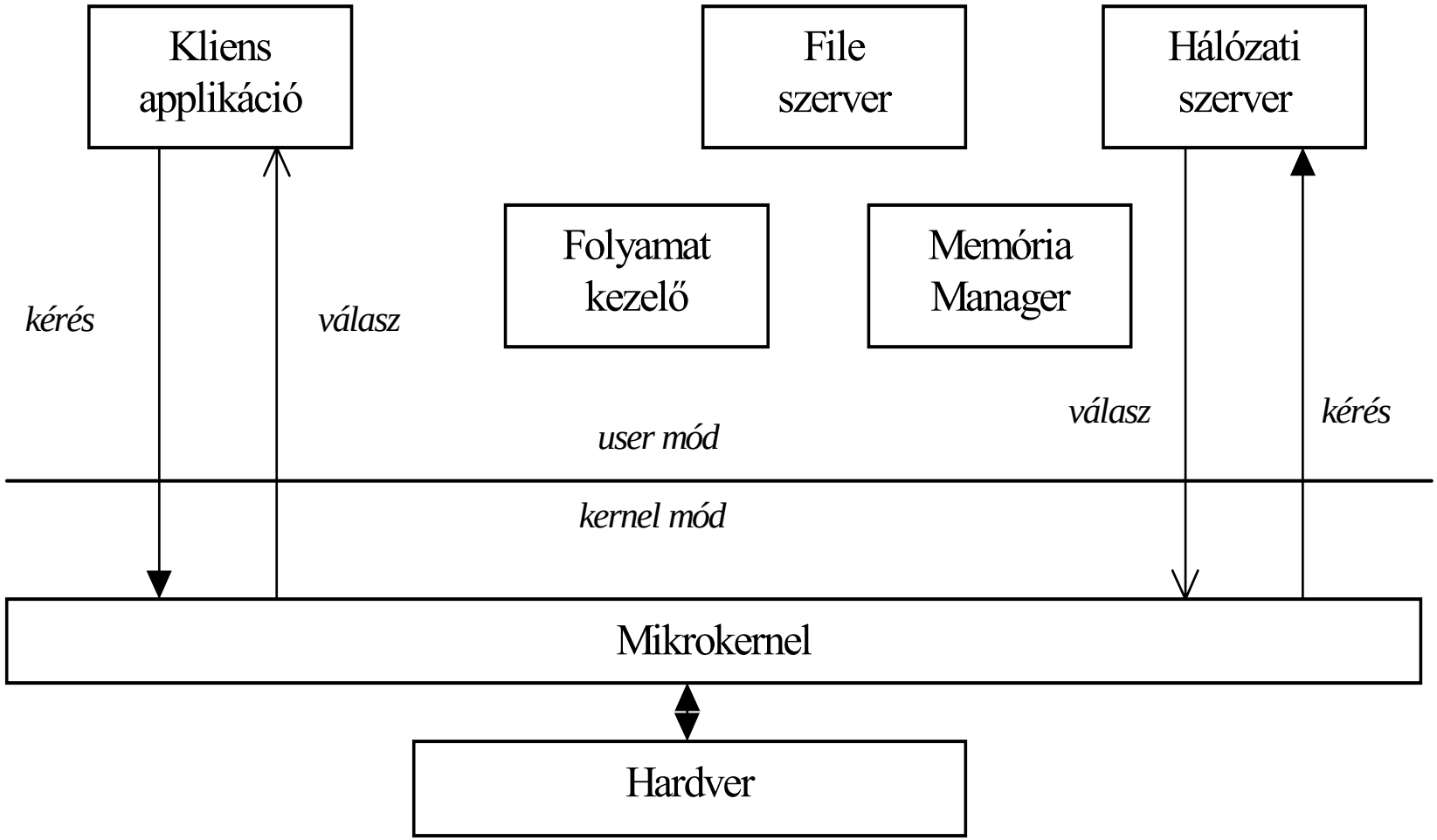
Réteg szerkezetű operációs rendszer I.



Réteg szerkezetű operációs rendszer II.

- A modulok rétegszerűen épülnek egymásra (objektum-orientált megközelítés).
- A kommunikáció jól definiált interface-eken keresztül bonyolódik.
- A rétegek csak a felettük illetve alattuk levő réteget érik el közvetlenül.
- Az interface-ek használata:
  - biztonságos: ellenőrizhető a kérés/hozzáférés jogossága,
  - megbízható: nem terjed a hiba,
  - karbantartható: fejlesztés, módosítás egyszerű.

Kliens-szerver modellre épülő operációs rendszer I.



Kliens-szerver modellre épülő operációs rendszer II.

- Önálló, jól definiált funkciókat ellátó komponensek.
- Egymásra épülő szolgáltatások nyújtása, ill. azok használata.
- Mag az ún. *mikrokernel* (mikrokernel-es operációs rendszerek).
- Funkciója:
  - Szolgáltatás kéréseket üzenetek formájában továbbítja az operációs rendszer moduljai számára.
  - Biztosítja a folyamatok számára a hardver elérését. (Hardver elérés csak mikrokernel-en keresztül.)

Kliens-szerver modellre épülő operációs rendszer III.

- A modulok önálló szerver folyamatok.
- Mikrokernel: privilegizált (kernel) mód.
- Folyamatok: felhasználói mód.
- Előny: rugalmas, biztonságos működés.
- Hátrány: nem hatékony hardver kihasználás (környezetváltások!).

  Pl.: *Mach* operációs rendszer.
  UNIX daemon folyamatok

Az NT felépítésének fő jellemzői

- Réteg szerkezet.
- Kliens-szerver működés
  (pl.: alrendszerek, szolgáltatások).
- Szigorú interface használat.
- Hatékony hardver kihasználás:
  a user módban megvalósítható egyes
  funkciók kernel módban futnak (pl.: WIN32-es
  grafikus szolgáltatások).

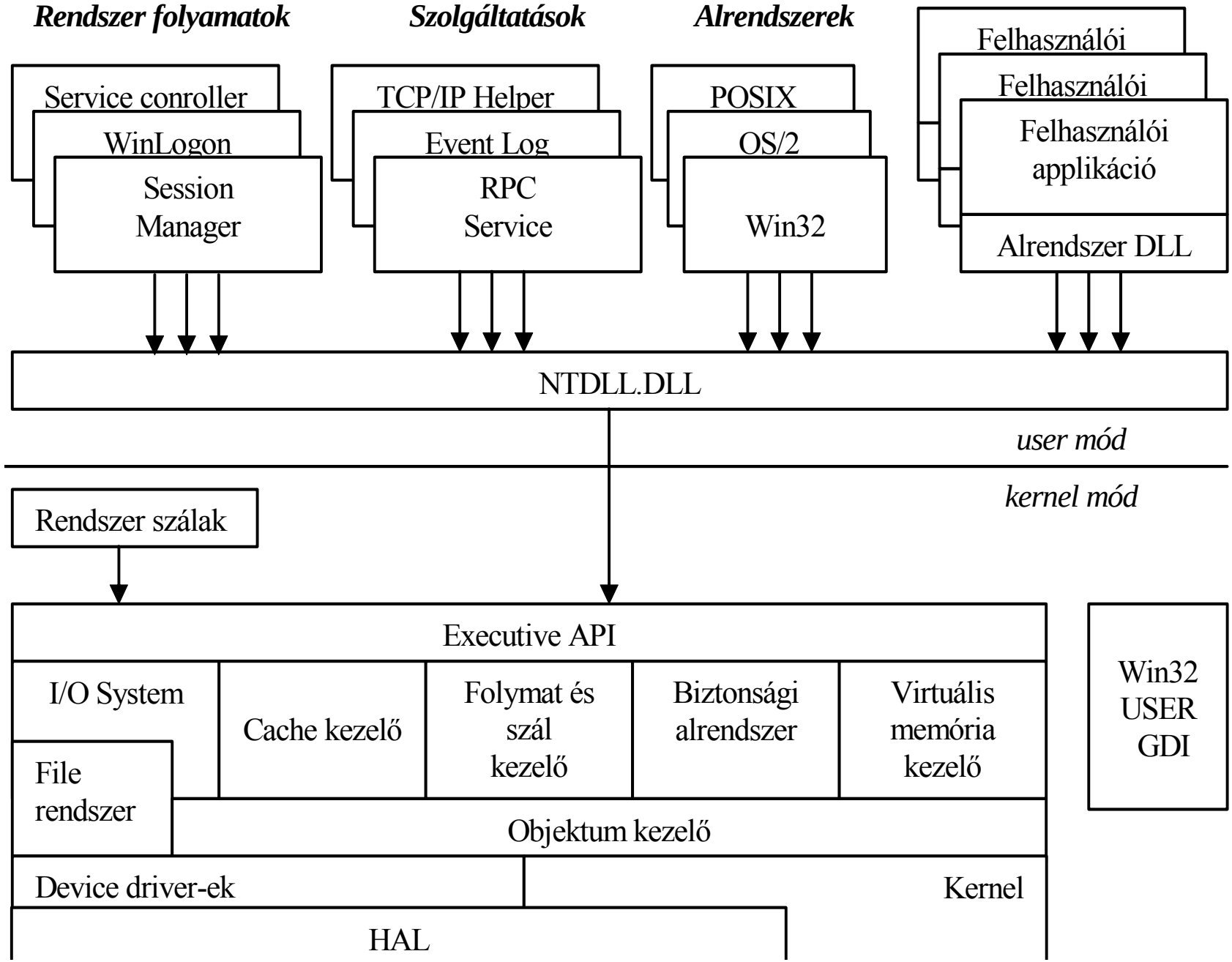
Az NT objektum-orientált szemlélete I.

- *Adatrejtés*:
  az operációs rendszer objektumai csak saját adataikat érhetik el.
- *Interface-használat*:
  az objektumok formális interface-t használnak egymás elérésére.
- *Hierarchikus objektum szerkezet*:
  kernel objektumok, executive objektumok.

Az NT objektum-orientált szemlélete II.

- A Windows NT ***nem*** valósítja meg a következő objektum-orientált tulajdonságokat:
  - *polimorfizmus*:
    azonos néven különböző objektumok elérése.
  - *öröklődés*:
    az objektumoknak hierarchikus származási rendszere.
  - *dinamikus adattípus-kötés*:
    definiálhatóak adattípussal paraméterezett objektumok.

A Windows NT felépítés


Rendszer folyamatok
Szolgáltatások
Alrendszerek
Service conroller
WinLogon
Session Manager
TCP/IP Helper
Event Log
RPC Service
POSIX
OS/2
Win32
Felhasználói
Felhasználói
Felhasználói applikáció
Alrendszer DLL
NTDLL.DLL
user mód
kernel mód
Rendszer szálak
Executive API
I/O System
Cache kezelő
Folymat és szál kezelő
Biztonsági alrendszer
Virtuális memória kezelő
File rendszer
Objektum kezelő
Device driver-ek
Kernel
HAL
Win32 USER GDI

HAL (Hardware Abstraction Layer)

- A hardvert teljesen elfedő legalacsonyabb szoftver réteg.
- Az aktuális hardverre támaszkodva egy *virtuális gépet* valósít meg.
- Hardver elérés csak ezen keresztül.
- Processzortól független szolgáltatásokat ad a többi rétegnek.

HAL II.

- Mindegyik processzorhoz külön HAL-réteg.
- A rendszer a telepítésekor választ HAL-t. (NT CD \i386–os alkönyvtára).
- Azonos architektúrájú processzorok esetén (pl. az x86 típusú processzorokon futó NT-k között) csak a HAL-ban van különbség, szolgáltatásokban nincs.

  (Arthur C. Clarke: 2001. űrodisszeia, 1971

  HAL 9000 heurisztikus programozású algoritmikus számítógép)

Kernel

- Állandóan memóriában levő kernel (védett) módban futó komponens.
- A kernel a HAL-on kívül az egyetlen hardver-függő rész:
  - a HAL a processzor típusától függ (típus funkciók megvalósítása),
  - a kernel a processzor architektúrájától (architektúra funkciók megvalósítása).
- A kernel által a nyújtott interface hardver független:
  - így a többi komponens is az.

NT hordozhatósága

- Hordozható operációs rendszer megvalósítása:
  - Hardver független interface-szel rendelkező szoftver rétegek:
    - a HAL, illetve a kernel feletti rétegek.
  - Hordozható programnyelv használata:
    - C: a kód túlnyomó része,
    - C++: grafikus alrendszer, user interface kisebb része,
    - assembly: a HW közvetlen kezelését végző részek.

A kernel funkciója

- Jól definiált működésű alapmodulokat (ún. primitíveket) biztosít az NT többi tagja számára.
- A primitívek a következő funkciókat valósítják meg:
  - *thread (szál) ütemezés, környezetváltás,*
  - *kivételkezelés, trap-kezelés, IT-kezelés,*
  - *multiprocesszor-ütemezés,*
  - *kernel objektumok kezelése.*

Kernel objektumok

- A kernel objektumok egyszerűek.
- Cél:
  - gyors kezelés.
- Megvalósítás:
  - Nem végez ellenőrzést az egyes objektumok elérésekor, megbízik abban, hogy a rendszer tagjai helyesen használják az objektumokat.

Device driver-ek (készülék meghajtók)

- Az I/O alrendszer és a HW közötti kapcsolat.
- HW elérés: HAL-rétegen keresztül. ("Hordozható" device driver-ek.)
- A device driver-ek négy típusát különböztetjük meg:
  - Hardver driver-ek.
  - File-rendszer driver-ek.
  - Szűrő típusú device driver-ek.
  - Hálózat elérését biztosító device driver-ek.

Device driver-ek típusai I.

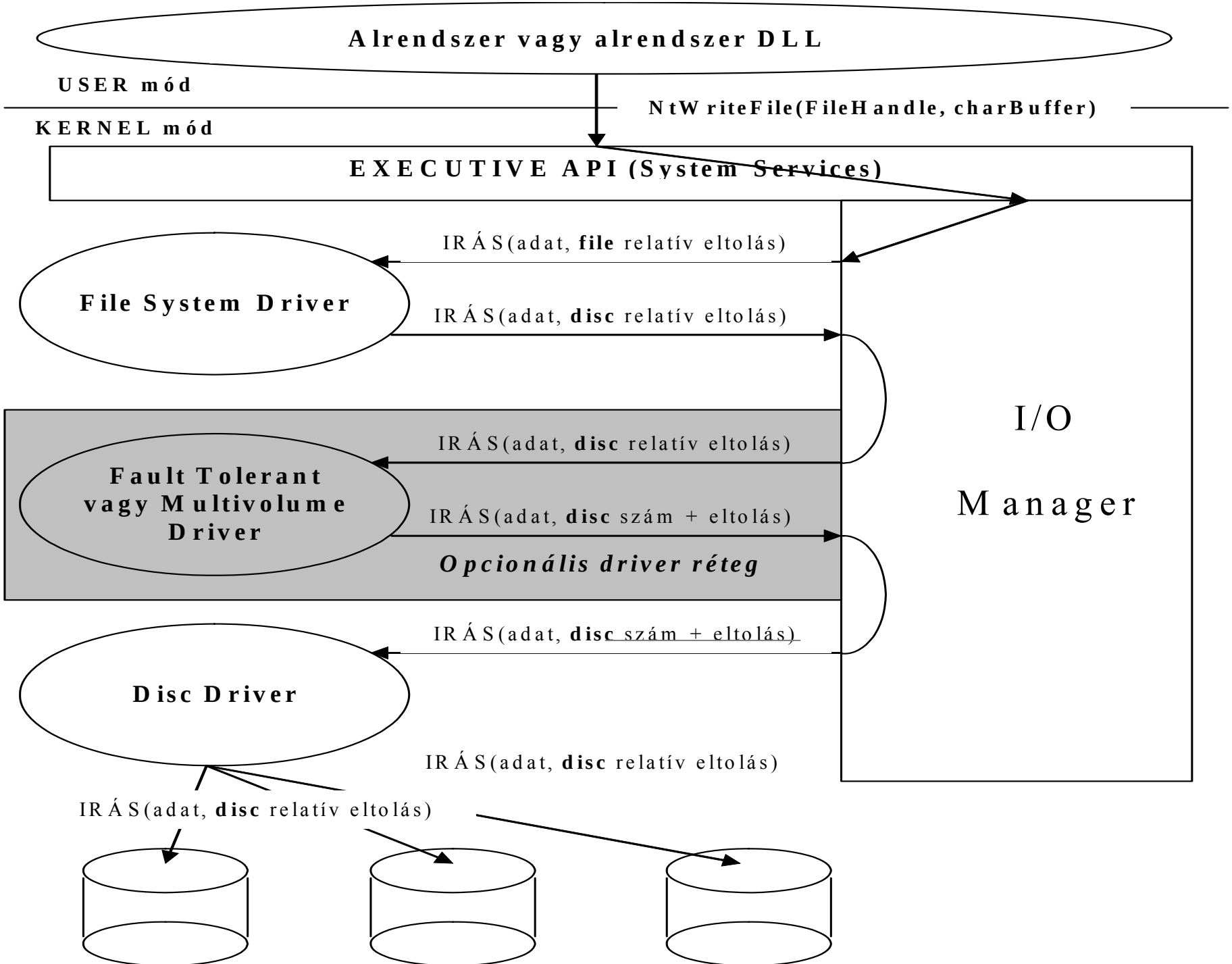
- Hardver driver-ek (alacsony szintű):
  - hardver egységek elérése, eszközök közvetlen be- és kimeneteit kezelik.
- File-rendszer driver-ek: (magas szintű)
  - file-rendszerek elérését biztosító kéréseket fogadnak el és I/O kérésekké transzformálják azokat.

Device driver-ek típusai II.

- Szűrő típusú device driver-ek:
  - többletfunkciók megvalósítása (pl.: RAID technikák) a réteg szerkezetű device driver struktúra segítségével.
    A magasszintű driver-ek és alacsony szintű driver-ek közé ékelődnek.
- Hálózat elérését biztosító device driver-ek:
  - hálózati kéréseket szolgálják ki, illetve továbbítják azokat.

Réteg szerkezetű device driver struktúra

- Különböző funkcionalitású driver-ek rugalmas használata.
- Igényeknek megfelelő rendszer építése.
- Különböző driver szintek:
  - A szintek funkcionalitása, illetve az adott funkcionalitáshoz tartozó interface jól definiált.
- Példa: Lemez írása.


Alrendszer vagy alrendszer DLL
USER mód
NtWriteFile(FileHandle, charBuffer)
KERNEL mód
EXECUTIVE API (System Services)
IRÁS(adat, file relatív eltolás)
File System Driver
IRÁS(adat, disc relatív eltolás)
I/O
Manager
IRÁS(adat, disc relatív eltolás)
Fault Tolerant
vagy Multivolume
Driver
IRÁS(adat, disc szám + eltolás)
Opcionális driver réteg
IRÁS(adat, disc szám + eltolás)
Disc Driver
IRÁS(adat, disc relatív eltolás)
IRÁS(adat, disc relatív eltolás)

Executive

Az OPR szolgáltatásait nyújtó alrendszereket megvalósító réteg.

Megvalósított függvények:

- elérhető alrendszerek hívások,
- nem dokumentált alrendszerek hívások,
- dokumentált driver hívások,
- nem dokumentált belső (kernel, executive) hívások,
- belső hívások.

Executive önálló részei

- Folyamat és szál kezelő
  - létrehozás, kezelés, leállítás.
- Virtuális memória kezelő
  - memória gazdálkodás, cache kezelés támogatása.
- Biztonsági alrendszer (monitor)
  - lokális gép objektum védelme.
- Cache kezelő
  - file kezelő I/O gyorsítása.
- I/O rendszer kezelő
  - eszköz független I/O.

LPC funkció I.

- Csak az **NT belső objektumai** közötti kommunikáció biztosítására szolgál.
  - LPC (*Local Procedure Call*, lokális eljárás hívás):
    - az NT IPC (*Inter Process Communication*) eszköze,
    - egy user objektum egy másik user objektum függvényét hívhatja meg,
    - így érhetik el pl. az egyes alkalmazások a hozzájuk tartozó alrendszer szolgáltatásait.

LPC funkciói II.

- Az NTDLL.DLL által definiált függvények hívásainak megvalósítása.
- A *run-time library* (rendszerhívások) függvények megvalósítása.
- Rendszer processzek és a szolgáltatások számára támogató funkciókat megvalósító függvények.

Rendszer processzek (rendszer folyamatok)

Operációs rendszer-funkciókat megvalósító önálló folyamatok:

- ugyan felhasználói módban futnak, de
- nélkülözhetetlen részei a rendszernek.

Fontosabb rendszer folyamatok I.

- *SMSS* (*Session Manager*):

  A rendszer indításakor létrehozott folyamat, amely az alkalmazások elindításáért felelős.

  Feladatai:

  - egyes alrendszerek elindítása, ha futásukra szükség van, és még nem futnak,
  - kapcsolat biztosítása a debugger és az általa futatott applikáció között,
  - biztosítja a környezeti változók definiálásának és elérésének lehetőségét.

Fontosabb rendszer folyamatok II.

- *Logon*:
  - A felhasználók ki és beléptetését intéző folyamat.
  - Minden ún. *SAS* (*Secure Attention Sequence*) billentyű kombináció (alapesetben: CTRL-ALT-DEL) aktivizálja.
  - Beléptetéskor a username és password kombinációt az önálló folyamatként futó *Local Security Authentication Server*-hez (*LSASS*) továbbítva ellenőrzi.
  - Ha az azonosítás sikeres, elindítja a *USERINIT.EXE* programot, ami beállítja a felhasználó által definiált környezetet, és elindítja a user által kért shell-t. (alapesetben: *EXPLORER.EXE*)

Fontosabb rendszer folyamatok III.

- *Service Controller*:
  - A szolgáltatások indításáért és leállításáért felelős folyamat.
  - User interface:
    - szolgáltatás állapota,
    - szolgáltatás beállításai,
    - szolgáltatás indítása/leállítása.

Szolgáltatások

- Szolgáltató folyamatok:
  - kliens-szerver működésű szerver folyamat,
  - OPR-re támaszkodva többletszolgáltatást nyújt,
  - szolgáltató folyamatok pl.:
    - RPC (Remote Procedure Call) hálózati szolgáltató,
    - NT specifikus eseménynaplózó (event logger) szolgáltató.

Szolgáltatások tulajdonságai

- Felhasználói módban futó folyamatok.
- A szolgáltatásokat megvalósító folyamatok futása nélkül is képes működni az NT.
- A szolgáltatások a Service Manager segítségével elindíthatók és leállíthatók a rendszer működése közben.
- Egyszerű Win32-es alkalmazások, de együttműködnek a *Service Conroller* (*SERVICES.EXE*) folyamattal (regisztrál, indít leállít).
  - Elérés: *Vezérlőpult → Felügyeleti Eszközök → Szolgáltatások* ikonja

A szolgáltatások három elnevezése

Ahogy a szolgáltatás a *Vezérlőpult → Felügyeleti Eszközök → Szolgáltatások* alpontján keresztül elérhető.

- Ahogy Registry-ben (rendszerleíró adatbázisban [paraméterek tárolása]) szerepel.
- A szolgáltatást megvalósító futtatható program neve.

(Pl.: Hálózati bejelentkezés: lsass.exe.)

NTDLL.DLL

- Dinamikusan link-elhető könyvtár (*Dinamically Linked Library*, -*DLL*).
  - Interface a felhasználói módú folyamatok számára.
  - Minden felhasználói objektum az NTDLL.DLL-en keresztül éri el a környezetét (pl. LPC).
- Működés:
  - A hívási paraméterek ellenőrzése.
  - A user–kernel módváltás.
  - Az NT kért függvényének a meghívása.

Alrendszerek I.

- Applikációk futtatása az alrendszerek segítségével történik:
  - Win32 (szükséges),
  - POSIX (opcionális),
  - OS/2 (opcionális).
- Alrendszerek feladata:
  - az alkalmazások futásához szükséges NT szolgáltatások nyújtása,
  - alkalmazások kontrollálása.

Alrendszerek II.

- Minden alrendszerhez tartozik egy DLL.
- Az adott alrendszerhez tartozó alkalmazás ezen keresztül éri el az NT-t.
- Tehát az alkalmazások és az NT interface-e az: **alrendszer DLL** (programozói interface (*API*, *Application Programming Interface*).

Alrendszerek programozói interface-ei

- Jól definiált publikus interface, míg a többi interface (pl. NTDLL.DLL) nem publikus, nem dokumentált.
- Az egyes alrendszerekhez tartozó API-k lényegesen különböznek egymástól. (A legszélesebb a Win32 API.)
- API funkciók: pl. ablakozás, szálak kezelése.
- Az applikációk csak egy alrendszerhez tartozhatnak, nem keveredhetnek egymással egy applikáción belül különböző alrendszerekhez tartozó hívások.

POSIX alrendszer I.

- Szigorúan a POSIX 1003.1-es szabványban rögzített tulajdonságokat valósítja meg.
- Lehetőség van:
  - új folyamatok létrehozására, fork-olásra,
  - hard link-ek definiálására,
  - interprocess (folyamatok közötti) kommunikációra,
  - folyamatok kezelésére,
  - karakteres I/O kezelésre.

POSIX alrendszer II.

- **Nincs** lehetőség:
  - szálak (thread-ek) létrehozására,
  - ablakkezelésre,
  - RPC-elérésére (távoli eljáráshívás),
  - socket-ek (logikai csatlakozók) használatára, a hálózati kapcsolatok kialakításakor.

Win32 alrendszer I.

- A Win32 alrendszer:
  - 32-bites alkalmazások,
  - 16-bites és
  - DOS-os alkalmazások futtatása.

- Nélküle nem futhat az NT!

Win32 alrendszer II.

Az alrendszer egy része kernel módban fut. (*WIN32K.SYS*):

- Grafikus képernyő-kezelési funkciók. (A *USER32.DLL*, *GDI.DLL*, *KERNEL32.DLL*, *ADVAPI.DLL* könyvtárakban definiált funkciók.)
- Rendszer gyorsítása miatt kerültek ide, módváltás nélkül lehetséges az executive, illetve a kernel-szolgáltatások (függvények) elérése.

Win32 alrendszer III.

Grafikus eszközözök és a printer kezelése szabványos felületen történik:

- A *GDI* (*Graphical Device Interface*), illetve a hozzá tartozó *GDI32.DLL* (WIN32 API része) teszi lehetővé. Ezek is a WIN32 alrendszer kernel módú részében (*WIN32K.SYS*) vannak megvalósítva.
- Ezek a rutinok hívják meg az eszközökhöz tartozó device driver-eket.

Win32 API-hívások megvalósítása I.

- Megvalósításuk helye szerint:
  - Az alrendszer DLL-ben:
    egyszerű funkcionalitású függvények, a végrehajtásukhoz nincs szükség a rendszer más részeinek elérésére.

Win32 API-hívások megvalósítása II.

- Az alrendszerben:
  a hívást az alrendszer DLL továbbítja az *NTDLL.DLL* felé, ami az Executive réteg LPC szolgáltatását igénybe véve, eljut a Win32 alrendszerhez, ami a kérést teljesíti.
- Más (kernel módban futó) rétegben:
  a hívás az executive réteghez kerül, ami továbbítja a megfelelő kernel komponens felé.