Előadás_#01.

1. Bevezetés

   Minden magára kicsit is adó tudomány vizsgálatát, vagy legalább a tudomány látszatát kelteni szándékozó okoskodást azzal szokás kezdeni, hogy „...már az ókori görögök tapasztalatai szerint is...” vagy „...már az arab matematika hajnalán is...” vagy „...már az egyiptomi fáraók idejében is...”. Az operációs rendszerek esetében ez a sablon megbukik, mert az általunk vizsgálandó témakör nem tekint ilyen dicső múltra vissza. Az ókorból talán csak a praktikus és logikus megoldások keresésének vágya maradt ránk.

   A technika gyors fejlődésének következtében a 30-40 vagy akár csak az 5-10 évvel ezelőtt történtek is roppant távolinak, és (lássuk be) korszerűtlennek tűnnek. Persze az előző állítás sem abszolút igazság, hiszen a legalapvetőbb belső mechanizmusok, ütemezések, algoritmusok tekintetében sok – informatikai szempontból matuzsálemi kort megért, és azóta is vígan működő – régi, jól bejáratott megoldással találkozunk.

   Az operációs rendszer – mint a működés/működtetés alapvető része – minden számítógépes rendszerben megtalálható. Operációs rendszer nélkül a hardver egy átlag felhasználó számára kb. annyit ér, mint az utas számára egy helikopter pilóta nélkül.

   Az operációs rendszer az a program, amely közvetítőként működik a felhasználó, valamint a felhasználó által futtatott szoftverek és a számítógép hardvere között. Elsődleges cél az, hogy a felhasználó egyszerűen, zökkenőmentesen és a megfelelő biztonsági szinten legyen képes futtatni a saját programját. A másodlagos cél a számítógép hardverének minél hatékonyabb használata.

   A számítógépek hardver felépítéséről, azon belül is a processzor, az alaplap, a különféle tárolóeszközök és perifériák működéséről a „Számítógépek Felépítése” című tantárgy keretein belül már foglalkoztunk. A lényeg, hogy ne felejtsük el: egy informatikai értelemben hardvernek nevezett eszköz (komplex berendezés) működtetéséhez mindenképpen valamilyen szoftverre van szükségünk.

   Az operációs rendszer feladata tehát az, hogy a rendelkezésre álló szoftveres és hardveres erőforrásokat (legyen az fizikai, vagy logikai erőforrás), a felhasználó (vagy felhasználók) által futtatott programok részére optimálisan és hatékonyan rendelkezésére bocsájtsa.

Az operációs rendszerek feladataikat és funkcióikat jellemzően réteges szerkezetben valósítják meg. Az egyes rétegek csak a közvetlenül alatta vagy felette elhelyezkedő réteggel kommunikálnak. (Hasonló logikával felépített modellek fognak visszaköszönni majd a „Számítógéphálózatok” tantárgyban is.)

<table>
<tr><td rowspan="5">SZOFTVER</td><td colspan="2">Alkalmazások
(Szövegszerkesztő, Adatbázis kezelő, Webböngésző, stb.)</td></tr>
<tr><td colspan="2">Magas szintű nyelvek
(C++, Pascal, Basic)</td></tr>
<tr><td colspan="2">Alkalmazási Interfész (API)
Alacsony szintű nyelvek (Assembly, Gépi kód)</td></tr>
<tr><td rowspan="2">OPERÁCIÓS RENDSZER</td><td>user mód</td></tr>
<tr><td>kernel mód</td></tr>
<tr><td rowspan="3">HARDVER</td><td colspan="2">Buszrendszer, Memória, Háttértár, Perifériák</td></tr>
<tr><td colspan="2">Processzor (ALU, regiszterek)</td></tr>
<tr><td colspan="2">Logikai kapuáramkörök</td></tr>
</table>

A táblázatban a BIOS illetve az UEFI nem szerepel, de nem szabad arról megfeledkezni, hogy a BIOS illetve az UEFI is az operációs rendszer és a hardver között teremt kapcsolatot. A BIOS-szal illetve az UEFI-val a „Számítógépek Felépítése” című tantárgy keretein belül már foglalkoztunk.

2. Az operációs rendszerek fejlődése

Az operációs rendszerek evolúciója nem kezelhető a hardver komponensek fejlődésétől, változásától függetlenül. A hardverek és szoftverek fejlődése általában jótékony hatással vannak egymásra, de persze akadnak kivételek is. Ilyen például a Windows Vista, amely operációs rendszer fejlesztői abból indultak ki, hogy felesleges a kódot egy szinten túl optimalizálni (azaz nem baj, ha lassú a rendszer), hiszen a hardver elemek gyorsan fejlődnek, valószínűleg az áruk is csökken, és várhatóan annyit gyorsulnak, hogy kompenzálni fogják a szoftver lassúságát.

Ok és okozati viszonyban, másképpen fogalmazva „tyúk és a tojás problémája” szintjén könnyen belátható, hogy csak egy hardver megjelenése után lehet a hozzá szánt szoftvereket elkezdeni fejleszteni, optimalizálni.

A hardverek fejlődése során az egyik legjelentősebb lépés a buszrendszerek megjelenés után a specializálódás volt. Eleinte minden aritmetikát a CPU számolt, majd megjelentek az egyedi vezérlők (Controller), melyek csak egy adott periféria működéséért voltak felelősek. Megjelentek továbbá a megszakítások (IRQ, NMI), és megjelent a perifériák közvetlen memória elérése is (DMA).

3. Az operációs rendszerek típusai

Az első számítógépek elnevezésükön túl nem sok hasonlóságot mutattak a mai számítógépekkel. Nem létezett sem a billentyűzet, sem az egér, sem az alfanumerikus vagy grafikus kijelző. A futtatni kívánt kódot egy manuális telefonközpont kapcsolótáblájához hasonló dugaszolós táblán lehetett közölni a géppel. A kód lefutási idejét legfeljebb tapasztalati állandók segítségével lehetett megsaccolni. Az eredmény – ami jellemzően egy számérték volt – helyiérték lámpák segítségével volt leolvasható. Komoly előrelépés volt a lyukkártya megjelenése, amely egy részt adatbevitelre, másrészt az eredmény megjelenítésére is használatos volt a nyomtatás megjelenéséig.

A kötegelt (batch) feldolgozás a programozó mellett még egy operátor jelenlétét is igényelte. Az operátor (mivel programoznia nem kellett) folyamatosan képes volt a számítógépet újabb és újabb lyukkártyákkal ellátni, ezzel csökkentve a holtidőt, viszont a programozó így már csak a kód írásával, azaz a program megalkotásával kellett hogy foglalkozzon. A háttértárak fejlődésével, a mágnesszalag megjelenésével lehetőség nyílott a programok lyukkártyáról szalagra másolására, tovább növelve ezzel a hatékonyságot. A „teljes kihasználtságot” az egyszerű monitor (Resident Monitor) kifejlesztése jelentette, amikor egy kód lefutása és az eredmények tárolása után a következő kód automatikusan képes volt elindulni. A „resident” szó arra utal, hogy ennek a programnak az aktuális programokkal egy időben kellett a memóriában lennie, azaz a memóriát két részre kellett osztani.

A perifériák sebessége ma is lényegesen lassabb a processzor sebességénél, így már az informatika hajnalán elkezdődött a megoldás keresése arra problémára, hogy a processzor lehetőleg ne a lassú perifériákra való várakozással töltse az idejét.

A spooling technika (Simultaneous Peripherial Operation On-Line / Spool → orsó) a mágneses tárat, mint nagyméretű puffert használja, így egyszerre nemcsak egy, hanem több kódot (azaz programot) is lemezre tölt és tárol. Ez a megoldás lényegében az off-line perifériás műveletek egy gépen való megvalósítása. Így lehetővé válik a perifériás és processzor műveletek szétválasztása oly módon, hogy több munka is átlapolódhat. Például az egyik munka végrehajtásával egy időben egy másik munka beolvasása is folyhat, valamint egy harmadik munka eredményeinek rögzítése is történhet. Ezáltal a perifériák és a processzor kihasználtsága egyaránt jelentősen javult.

Gyakorlatilag ezzel megnyílt az út (azaz modell már létezett) a folyamatok illetve a szálak megjelenéséhez.

4. Multiprogramozott rendszerek, Időosztásos rendszerek

A multiprogramozás megjelenése, azaz több programkód együttes futása megnövelte a processzor kihasználtságát. Így amíg az egyik program például egy lassú perifériával cserél adatot, addig a processzor egy másik programot képes futtatni. További előnyt jelentett a RAM (Random Access Memory) szervezésű tárak illetve memóriák megjelenése. A „random” szó jelentése: tetszőleges. Ez azt jelenti, hogy a korábbi kizárólag szekvenciális (azaz kötött sorrendű) elérést a tetszőleges sorrendű elérés váltotta fel. A multiprogramozás foka ebben a megközelítésben jellemezhető az adott pillanatban a memóriában jelenlévő és tetszőlegesen elérhető folyamatok számával.

A felhasználó szempontjából a hatékonyság így már képes volt megközelíteni a processzor teljesítőképességének körülbelül 90%-át. A felhasználó szempontjából a 100%-os kihasználtság elméletileg sem érhető el, hiszen a háttérfolyamatok, az egyes programok közötti váltás, a környezetváltás, a használatban lévő adatok mentése illetve visszaállítása is igényel erőforrást a processzortól.

Az időosztásos rendszerek első generációi arra voltak képesek, hogy több felhasználó igényeit (kódját) gyakorlatilag közel egyszerre befogadják, és mindenegyes felhasználó úgy érezhette, hogy a számítógép csak vele, vagyis csak az ő programjával foglalkozik. Megszületettek tehát a folyamatok és az ütemezés. Az ütemezés lehet idő alapú, vagy folyamat alapú. Első esetben egy időszelet lejártáig fut a folyamat, utóbbi esetben a folyamat befejeződéséig.

A multiprogramozás lépései:

- A rendszer nyilvántartja és tárolja a futtatandó munkákat.
- A kiválasztott munka addig fut, amíg várakozni nem kényszerül, vagy esetleg be nem fejeződik.
- Az operációs rendszer feljegyzi a várakozás okát, majd kiválaszt egy másik futni képes munkát és azt elindítja.
- Ha a félbehagyott munka várakoztatása alatt a munka folytatásának feltételei teljesülnek, akkor amint lehetőség nyílik rá, a munka újra futtatásra kerül.

Az időosztásos (Time Sharing) rendszerek:

- Az időosztásos rendszerek közvetlen interaktív kommunikációt biztosítanak a felhasználó és a programja, valamint az operációs rendszer között.
- Adatok közvetlen elérésű (On-line) állományrendszerben tárolódnak.
- A felhasználó egy jellemzően lassú „periféria”, így például adatbevitel közben az operációs rendszer más tevékenységet végre tud hajtani.

- Gyors a reakció a parancsokra, a válaszidő (Response Time) kicsi, de cserébe a processzornak gyakran kell a programok között váltani.
- Több felhasználó egymástól függetlenül használja a gépet úgy, mintha mindenki egy saját gépen dolgozna.

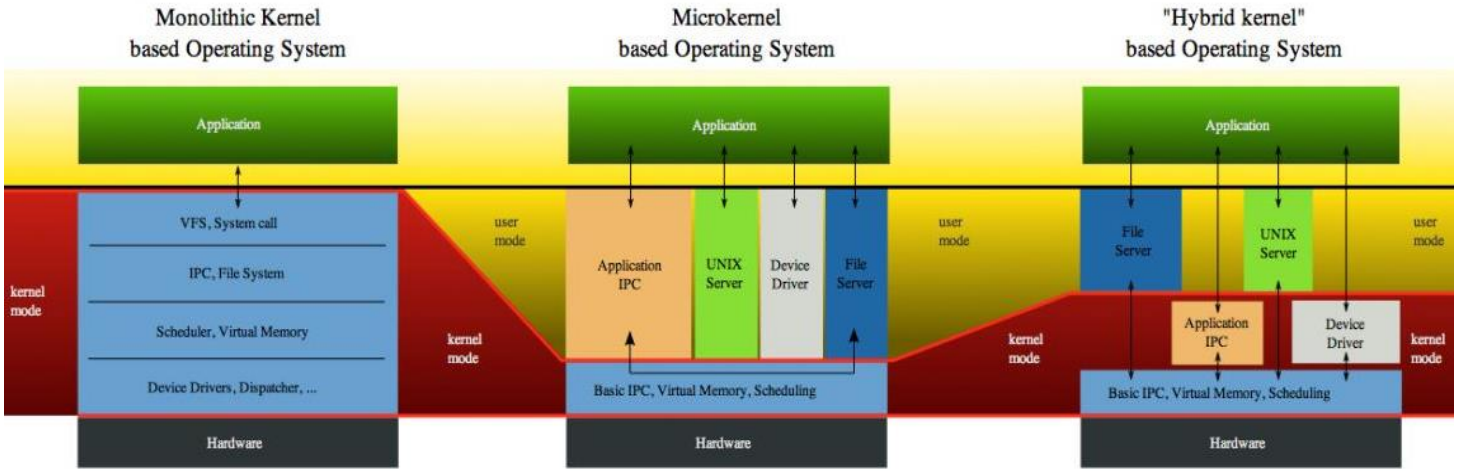
5. Operációs rendszerek tipikus komponensei és jellemző felépítése
   Az összetett funkcionalitás megköveteli a komplex feladatok részekre bontását. A legjellemzőbb rendszer komponensek:
   - Folyamatkezelő
   - Központi tárkezelő (operatív tár)
   - Állománykezelő (elsődleges háttértár)
   - I/O rendszer kezelő
   - Állománykezelő (másodlagos háttértár)
   - Védelmi rendszer
   - Hálózat kezelő
   - Felhasználói felület kezelő

   Az operációs rendszer alapja (vagy magja) a kernel. Maga a kernel a felhasználó számára láthatatlan program, hiszen a háttérben fut és a legalapvetőbb feladatok ellátása a feladata. Az operációs rendszerek különböző módon választják szét a kernel módban illetve a user módban futó komponenseket.

   A kerneleknek négy fő kategóriáját különböztethetjük meg (illetve léteznek olyan programkörnyezetek is, melyek kernel nélkül futnak):
   - A monolitikus kernelek gazdag és hatékony absztrakciókat biztosítanak az alattuk található hardver elemekhez. Jellemző az egyetlen nagy programból álló rendszermag, azaz nem a különálló, egymással különböző interfészeken keresztül kommunikáló programok összessége. (UNIX, Linux)
   - A mikrokernelek egy kisméretű alapkészletet biztosítanak a hardver kezeléséhez, és számos alkalmazással – amiket „szervereknek” nevezünk – biztosítják a további, részletesebb funkcionalitást. A mikrokernelek azáltal, hogy az általuk nyújtott funkciók nagy részét felhasználói szintre helyezték egy plusz absztrakciós szintet biztosítanak. A mikrokernel az a minimális forma, amely csupán néhány olyan alapvető funkciót – rendszerhívást – valósít meg, ami nélkülözhetetlen operációs rendszer szolgáltatásainak realizálásához. Ilyen például a címtartomány menedzsment, szál ütemezés, folyamatok közti kommunikáció, virtuális memória kezelés. Ennek előnye,

hogy a felhasználói szinten futó programrészek hibáinak vagy működési zavarainak esetén azok nem veszélyeztetik magának a rendszermagnak a működését, azaz a rendszer stabilitása nagymértékben nő. Hátránya viszont az, hogy minden új absztrakciós szint bevezetésével csökken a rendszer teljesítménye. Ezért előfordulhat, hogy egyes kritikus feladatokat (például nagyon gyors és pontos elérést igénylő hardver elemek kezelését) a rendszer nem képes kellő hatékonysággal megoldani. (AIX, Mac OS X)

- A hibrid kernel (vagy vegyes kernel) megoldásai hasonlóak a színtiszta mikrokernelekhez, de több, részletesebb kódot tartalmaznak a kernelmagban, hogy nagyobb sebességet érjenek el. A „hibrid” elnevezés arra utal, hogy ezen kernelnek mind a monolitikus, mind a mikrokernelek elveit és mechanizmusait alkalmazzák. Így például az üzenetcserét (message passing) és a „nem létfontosságú” kódok felhasználói szintre való áthelyezését is amellett, hogy néhány ilyen kód teljesítményi okokból mégis a kernelmagba kerül. (Windows NT, Linux, BeOS)
- Az exokernelek (vagy rendszer rutinkönyvtárak) nem biztosítanak absztrakciókat vagy állandó rendszermagot, hanem egy – a futtatandó programok részére használható – rutinkönyvtárból állnak, ami kizárólag a hardver közvetlen vagy közvetett elérését biztosítja. Az exokernel elve körülbelül 20 éves múltra tekint vissza, de napi használatú gyakorlati alkalmazása még várat magára.



Összességében megállapíthatjuk, hogy a legelterjedtebb kiszolgálási mód kernel szinten is a „kliens – szerver” működés. Egy szerver jellemzően több kliens szinte egyidejű kiszolgálására is képes.