Előadás_#09.

1. Bemeneti- és kimenetei rendszer (I/O)

   Egy számítógépnek a tiszta aritmetikai műveletek végzése mellett, képesek kell lennie a külvilággal való kommunikációra, adatcserére. A számítógép operációs rendszere, az azon futó programok, és a programok által igényelt adatok is valahogy be kell hogy jussanak a működési területre. A programok jellemzően futásuk közben a feladatuk elvégzéséhez szükséges bemenő adatokat igényelhetnek, futásuk végeztével pedig kimenő adatokat produkálnak.

   Periféria alatt azokat a hardver elemeket értjük, melyek nem beágyazott részei a számítógépnek, hanem a számítógéphez lettek csatlakoztatva. A számítógép tehát képes a perifériák nélkül is működni, de a perifériák, a számítógép képességeit és lehetőségeit bővítve, kényelmesebb és hatékonyabb működést eredményeznek.

   A perifériákat három nagy csoportra bonthatjuk feladatköreik alapján:

   - ember – gép kapcsolattartás (HID / Human Interface Device)
     billentyűzet, egér, touchpad, monitor, touchscreen, hangszóró, mikrofon, kamera, nyomtató, plotter, szkenner, ujjlenyomat olvasó, joystick, stb.
   - gép – gép kapcsolattartás
     LAN, WLAN, WWAN, Infra, Bluetooth, NFC, stb.
   - adattárolás
     mágneses, optikai és flash alapú tárolók

   A perifériák működését, viselkedését és használhatóságát meghatározó jellemzők:

   - csatlakozási szabvány
     soros, párhuzamos, USB, FW, eSATA, SCSI, iSCSI, FC, UTP/FTP (CAT5e, CAT6), DSUB, DVI, DisplayPort, HDMI, Memóriakártyák (SD, CF, MS, xD, MM), stb.
   - vezérlőegység
     A perifériák általában beépített vezérlőegységet tartalmaznak, de előfordulnak kivételek is. Például: GDI nyomtatás, xD kártya.
     A GDI ([Windows] Graphical Device Interface) nyomtatók közbeiktatott lapleíró nyelv nélkül képesek a (Windows) alkalmazásokból nyomtatni.
   - működési sebesség
     Gyakorlatilag mindig lassabb, mint a CPU sebessége, különös tekintettel a HID eszközökre.

- szinkronizáció

  A szinkronizáció azért kiemelten fontos, mert ha a számítógép nem képes a periféria által elvárt mennyiségű, ütemű és sebességű adatátvitelre, akkor az elvárt adatátvitel egy jellemzően lassabb (kisebb adategységű) ismétléssel lesz csak sikeres. A perifériákat a számítógép mellett egymáshoz is szinkronizálni kell, hiszen a perifériák a számítógéppel illetve egymással is képesek a párhuzamos működésre.

A fentiek ismeretében látszik, hogy a különböző perifériákat egyedi, a saját paramétereikhez legjobban megfelelő algoritmusokkal, módszerekkel (IRQ, DMA) kell használni.

Az operációs rendszer közvetlenül a következő feladatokat kell hogy ellássa az I/O eszközök kezelése során:

- egységes alkalmazói felület kialakítása (API / Application Interface)
  - műveletek egységesítése

    Fájl alapú periféria elérés esetén csak az utolsó szinten konkretizálódik, hogy mi lesz az adat sorsa. A periféria szempontjából megkülönböztetünk adatfájlokat és vezérlőfájlokat.
  - logikai periféria kezelés

    Perifériafüggetlen programozással, azaz az egyes perifériák sajátságainak ismerete nélkül, általános periféria meghajtók (Device Driver) segítségével kisebb CPU terheléssel lehet kezelni a perifériákat.
  - műveletek átirányíthatósága
- egységes csatlakozófelület kialakítása (DI / Device Interface)
- eszközmeghajtók illesztése, hatékony periféria működtetés

  A logikai perifériakezelésnél ismertetett általános periféria meghajtókkal szemben speciális, egyedi periféria meghajtó (Device Driver) használatával egyedi és hatékony perifériakezelés valósítható meg. A párhuzamos perifériakezelés szintén a hatékonyságot növeli. Multiprogramozás használatával lehetővé válik, hogy folyamatok I/O-kra várakozzanak, miközben más folyamatok végrehajtás alatt állnak.

Az operációs rendszer szempontjából az I/O műveletek tulajdonképpen adatblokkok mozgatása a memória vagy a háttértár illetve a perifériák között. Az adatblokkok paramétereit az IOCB tartalmazza. Ezen adatblokkokkal alacsony szintű I/O kezelés esetén a következő három művelet végezhető el:

- indítási művelet (Start)
  A logikai periférianév ismeretében kiválasztásra kerül a fizikai periféria, amely ha már aktív, akkor kész az adatcserére, ha nem aktív, akkor a kezelőprogram egy Start_Device utasítással indítja el az eszközt. Az operációs rendszer csak az indítást végzi, nem várja meg az I/O művelet teljesülést illetve végét, hiszen eközben a CPU-t más feladatra is tudja használni.
- várakozás/tesztelés (Wait)
  Szinkronizációs várakozás, illetve műveleti időkorlátok használatának lehetősége. Az IOCB állapotjelzője ez alatt az átvitel pillanatnyi állapotának megfelelően vehet fel értékeket (elkezdődött, várakozik, fut, befejeződött). Az I/O művelet befejezése után visszatér az I/O műveletet elindító folyamathoz a megfelelő állapotjelzéssel.
- kényszerített leállítás, hiba esetére (Stop)
  Az adott várakozási sorból az IOCB-t csak akkor lehet eltávolítani, ha nincs aktív átvitel. A már elindított, de (például az időkorlátig) be nem fejeződött, vagy programhiba miatt megállt I/O műveletek kényszerített megállítása a Stop_Device utasítás használatával lehetséges.

Az alacsony szintű, egyszerű I/O műveletek lehetővé teszik a szinkron (blokkolt, várakozó) illetve az aszinkron (nem blokkolt, továbbfutó) programhívást is.

Az adatmozgatás három legfontosabb paramétere:

- forrás
- cél
- adatmennyiség

A forrás és a cél egyaránt lehet fizikai vagy logikai memóriacím, illetve fizikai vagy logikai periféria név. Az adatmennyiség az adatcsere során küldött, illetve fogadott bájtok számát jelenti.

2. Elosztott rendszerek

Elosztott rendszer alatt jellemzően az úgynevezett lazán csatolt rendszert értjük, nem feledkezve meg arról, hogy az együttműködésnek nem csak ez az egy módja ismert. Lazán csatolt rendszer esetén a feladatok végrehajtása több, egymástól független, önálló CPU-val, lokális memóriával és saját órajellel rendelkező számítógép között oszlik meg. Szorosan kapcsolt rendszer esetén feladatot végrehajtó számítógépek közös órajellel és osztott memóriával rendelkeznek, jellemzően közös operációs rendszer használata mellett. Az erőforrások viszont mindkét esetben közösek, illetve a műveletvégzés is osztottan történik mindkét esetben.

Az elosztott rendszer nem keverendő össze egy általános célú, számítógépekből kialakított hálózattal. Ugyan mindkét megoldás autonóm számítógépekből áll, de a gépek közti viszony azonban eltérő. Egy számítógép hálózat minden egyes gépe egyértelműen azonosítható, címezhető.

Az elosztott rendszer is hálózattal összekötött autonóm számítógépek összessége, de ezen felül rendelkezik valamilyen elosztott szoftverrel is. Az elosztott szoftver (ami akár elosztott operációs rendszer is lehet) két legfontosabb funkciója a feladatok szétosztása és az erőforrások megosztása. Egy elosztott rendszer akkor teljesíti a vele szemben támasztott elvárásokat, ha a felhasználó azt egyetlen, integrált erőforrásnak látja, azaz az elosztottság a felhasználó számára rejtve marad.

Az elosztott rendszerek tehát:

- önálló, autonóm számítógépekből állnak,
- melyeket valamilyen kommunikációs csatorna köt össze,
- és az egyes hardver komponensek összehangoltan működnek.

Elosztott rendszerek esetében a következők a megvalósítandó célok:

- hatékony erőforrás gazdálkodás
- több felhasználó egyidejű kiszolgálása
- a részfeladatok gyors, párhuzamos megoldása
- megfelelő szintű biztonság elérése, akár redundancia igénybevételével
- a stabilitás biztosítása egyes gépek kiesése esetén is

Az elosztott rendszerek által használt hálózati megoldások, topológiák részletes ismertetésére a számítógép hálózatok tárgy keretein belül lesz lehetőség.

Az elosztott rendszerek tehát lehetővé teszik olyan informatikai környezetek kialakítását, amelyeknek jellemzői túlmutatnak az egyéni, azaz egygépes rendszer nyújtotta szolgáltatásokon. Az elosztott rendszerek fő jellemzői:

- erőforrás megosztás (Resource Sharing)
- nyílt rendszer (Open System)
- konkurens működés (Concurrency)
- méretezhetőség (Scalability)
- hibatűrés, megbízhatóság (Fault Tolerance, Reliability)
- átlátszóság (Transparency)

Ezen tulajdonságok együttes megléte az elosztott struktúrának köszönhető. Ez azonban nem jelenti azt, hogy minden elosztott rendszer jellegéből fakadóan teljesíti is ezeket a jellemzőket. A rendszer hardveres és szoftveres komponenseinek ismeretében, a topológia figyelembe vételével az elosztott rendszereket testre kell szabni, oly módon, hogy a fenti jellemzők mindegyike a megfelelő mértékben valósuljon meg. Az egyedi igényeknek megfelelően, az egyes rendszerekben ezek a jellemzők különböző prioritással vannak jelen.

Erőforrás megosztás

Az elosztott rendszerek esetében is mind a szoftveres, mind a hardveres erőforrások megosztása a cél. A hardveres erőforrások megosztása költséghatékony működést eredményez (például csökkenthető a nyomtatók darabszáma). A szoftveres erőforrások megosztása pedig a feladat végrehajtás hatékonyságát növelik, jellemzően több felhasználó esetében.

Azt sem szabad elfelejteni, hogy egyes erőforrások, például a memória abszolút elosztott kezelésének már operációs rendszer szinten is vannak korlátai, hiszen a memória használata részben dedikált módon történik.

Az osztott erőforrásokhoz jellemzően a rendszer bármelyik pontjából hozzáférhetünk, miközben maga az erőforrás fizikailag bármely másik ponthoz kötődhet. Az erőforrások megosztásáért az egyes számítógépeken futó erőforráskezelő (Resource Manager) folyamatok felelnek.

Az erőforráskezelő legfontosabb feladatai:

- az erőforrás megnevezésének és tulajdonságainak publikálása
- kommunikációs interface biztosítása az erőforrás eléréséhez
- az erőforrás párhuzamos elérésének szabályozása

Például egy nyomtató esetében ismerni kell, hogy a nyomtató melyik számítógép melyik portjára van csatlakoztatva, milyen lapleíró nyelv segítségével küldhetők nyomtatási feladatok a nyomtatóra. Kezelni kell a nyomtató nyomtatási sorát az egymás után, vagy egyszerre beérkező feladatok átmeneti tárolásával.
Az elosztott rendszerek az erőforrások kezelésére leggyakrabban a kliens-szerver modellt alkalmazzák. A kliensek egy előre meghatározott protokoll segítségével érik el a szervereket kéréseikkel. A szerverek a szolgáltatás megvalósulásáról, annak aktuális státuszáról hasonló módon tájékoztatják a klienseket. A kliens-szerver modell mindig az adott feladatra (illetve az erőforrás fizikai helyére) vonatkozik, nem magára valamelyik számítógépre. Például az egyes gépeken található fájlok megosztása kapcsán mindegyik gép viselkedhet szerverként és kliensként is, az adatkérés irányának megfelelően.

Nyílt rendszer

A nyílt rendszer elnevezésben a nyíltság azt jelenti, hogy a rendszer szoftveres illetve hardveres bővítése egyszerűen megoldható, azaz nyitott a változtatásokra. Hardveres bővítés alatt további perifériák hozzáadását, a rendszer hardver elemeinek bővítését (például memória, háttértár) értjük. A szoftveres bővítés az adott gép szoftverének újabb funkciókkal történő kiegészítése (például egy addig csak egy időben egy felhasználó által elérhető erőforrás párhuzamos elérését biztosító szoftver telepítése).
A nyílt elosztott rendszerek jellemzői:

- A folyamatok közötti kommunikációs megoldások egységesek, a folyamatok elhelyezkedésétől függetlenek.
- Az osztott erőforrások publikus interface-eken keresztül érhetők el.
- Heterogén rendszerek kialakításának támogatása, különböző gyártók szabványos eszközeiből.
- Szabványos kommunikáció használata az egyes hardver elemek között, előre definiált interface-ek segítségével.

A UNIX operációs rendszer volt az első olyan rendszer, amelyik támogatta a nyílt rendszerarchitektúrát. További rugalmasságot biztosít az, hogy a UNIX egy közkedvelt magas szintű nyelven (C nyelv) íródott, így egyszerűen lehetett új rendszerhívásokkal bővíteni, melyek lehetőséget adtak új perifériák, vagy új szolgáltatások implementálására.

Konkurens működés

A konkurens működés a folyamatok párhuzamos futtatásának az alapja az elosztott rendszer erőforrásainak esetében is. Ez esetben valós párhuzamosságról beszélünk, hiszen minimum a rendszerben lévő számítógépek számának megfelelő számú CPU mag áll egy időben rendelkezésre. (Több CPU-s gépek, illetve többmagos CPU-k esetében a rendelkezésre álló magok száma nyilván csak több lehet.)

Amennyiben az elosztott rendszert kezelő szoftvernek (vagy operációs rendszernek), lehetősége nyílik egy feladatot párhuzamosan végrehajtható részekre bontania, akkor a feladat végrehajtása jelentősen fel tud gyorsulni. Természetesen a konkurens végrehajtás esetén szükség van a folyamatok végrehajtásának időbeli szabályozására, azaz a folyamatok szinkronizációjára, különös tekintettel a közösen használt erőforrások elérésével kapcsolatos műveletekre.

A konkurens végrehajtás szerver és kliens folyamatokra egyaránt vonatkozhat, azaz egy időben több párhuzamosan futó szerver illetve kliens folyamat lehet a rendszerben, melyek hasonló szolgáltatást nyújtanak, illetve vesznek igénybe.

Méretezhetőség

A rendszer méretezhetősége azt jelenti, hogy a rendszer mérete – és ezzel együtt a kapacitása is – lényeges strukturális változtatások nélkül is megváltoztatható, megnövelhető. Egy jól kialakított rendszer képes követni a működése közben felmerült felhasználói igényeket, a rendszer így szükség szerint rugalmasan bővíthető.

Érdekes példa erre a telefonszámok hatjegyűről hétjegyűre történt változtatása, mely az előzetes tervezés fontosságára hívja fel a figyelmet. A rendszer tervezésekor meg kell találni az egyensúlyt, azaz a műszaki megvalósítás és a költségek helyes egymáshoz való viszonyát, valamint kalkulálni kell a bővítés lehetőségével is.

Egy másik jellemző példa maga a fájl megosztás, hiszen ennek kapcsán lehetőségünk van meghatározni az egyidejűleg létesíthető kapcsolatok számát. Ez esetben az indokolatlanul alacsony szám a rendszer lassulásához vezet, az indokolatlanul magas szám pedig feleslegesen köt le olyan erőforrásokat, melyeket csak a valós igények megjelenésekor célszerű igénybe venni.

Hibatűrés, megbízhatóság

Hiba alatt olyan működési állapotot értünk, mely kezelői beavatkozás nélkül nem szüntethető meg. Meg kell különböztetni a zavartól, amely mint fogalom a kezelő beavatkozása nélkül megszüntethető problémák összefoglaló elnevezése.

Fontos belátni és megérteni azt, hogy egyes hibák bekövetkezése objektíve elkerülhetetlen (például egy HDD fizikai meghibásodása). Amennyiben valamilyen redundáns tárolást használunk (például RAID1), akkor a bekövetkező probléma az elromlott HDD szempontjából valóban egy HIBA, viszont az egész rendszer szempontjából már csak egy ZAVAR. Azaz a rendszerünk továbbra is képes feladatainak ellátására. A meghibásodott HDD-t természetesen mielőbb cserélni kell, hogy a rendszer hibatűrése visszaállhasson az elvárt szintre.

A hibatűrés megvalósításához elengedhetetlen bizonyos redundanciák használata, mind a szoftver, mind a hardver tekintetében.

Egy informatikai rendszert akkor tekinthetünk hibatűrőnek, ha képes a működés során jelentkező hibák felismerésére, kezelésére oly módon, hogy közben a rendszer alapvető működése jelentős mértékben nem változik meg.

Az elosztott rendszerek egyik előnyös tulajdonsága az, hogy akár hardver hibák jelenlétében is nagyfokú rendelkezésre állást biztosítanak. Egy többfelhasználós, de nem elosztott rendszerben egyetlen hiba hatására jellemzően a rendszer elérhetetlenné válik az összes felhasználója számára. Ezzel szemben, amikor egy elosztott rendszerben meghibásodik valamelyik komponens, a hiba csak a meghibásodott komponensen folyó munkát érinti.

Homogén rendszerek, azaz azonos felépítésű és funkciójú gépek esetében egyes gépeket eleve tartaléknak állíthatunk be. Azaz, ha a rendszerben egy gép meghibásodik, a tartalék gép azonnal a helyére léphet.

Léteznek olyan abszolút kritikus rendszerek is, ahol a redundancia egy még magasabb szintje az úgynevezett szavazó rendszer kiépítése indokolt. Ebben az esetben három (minimum 3, és mindenképpen páratlan számú) számítógép párhuzamosan ugyanazt a feladatot hajtja végre. A rendszer végső kimenete, azaz a futtatás eredménye többségi elven keletkezik. Ez azt jelenti, hogy vagy mind a három gép ugyanazt az eredményt produkálja, vagy legalább kettő azonos eredményt produkál. Három különböző eredmény esetén meg kell ismételni a számítást, és/vagy el kell kezdeni a hibás gép(ek) kiszűrését. Ezt az elvet Neumann János dolgozta ki 1956-ban. Ne feledkezzünk meg az akkori és a mostani hibaarányokról sem.

A hibatűrés szoftveres támogatásának jellemző megvalósítása a javító blokkok (Recovery Block) módszere. Ez esetben az érintett szoftver modulokra kell bontani. Minden egyes modul több példányban kerül lefuttatásra, az eredményeket pedig egy önálló programmodul, egy elfogadási teszt ellenőrzi. Célszerű olyan szoftver komponenseket tervezni, amelyek segítségével az adatok jelentős része a meghibásodás észlelésekor visszaállítható, a számítások pedig visszagörgethetők (Recovery, Roll-Back).

Átlátszóság

Az átlátszóság ez esetben azt jelenti, hogy egy elosztott rendszerben a komponensek elosztott természete, az egyes szolgáltatások és erőforrások fizikai helye a felhasználó számára (közvetlenül) nem érzékelhető. A felhasználó a rendszert egy egységes egészként látja. A szolgáltatások igénybe vétele független a szolgáltatások fizikai helyétől, illetve a szolgáltatást biztosító hardvertől és szoftvertől. Ennek eléréséhez hálózati, kommunikációs, menedzsment és integrációs technikákra van szükség.

Az elosztott rendszerek az alábbi a kilenc féle átlátszósággal rendelkeznek:

- Hozzáférés átlátszóság (Access Transparency)
  A felhasználó (és tulajdonképpen a futó programok) előtt rejtett módon teszi lehetővé azt, hogy azonos módon történjen a helyi és a távoli erőforrások azonos műveleteket használó kezelése.
- Hely átlátszóság (Location Transparency)
  A felhasználó (és tulajdonképpen a futó programok) előtt rejtett módon teszi lehetővé az információs objektumok elérését, az objektumok helyének ismerete nélkül.
- Hálózati átlátszóság (Network Transparency)
  Ez egy összefoglaló terminológia, a hozzáférés átlátszóság és a hely átlátszóság együttes értelmezése.
- Konkurencia átlátszóság (Concurrency Transparency)
  A felhasználó (és tulajdonképpen a futó programok) előtt rejtett módon teszi lehetővé a folyamatok konkurens együttműködését, osztott információs objektumok használatának segítségével.
- Másolat átlátszóság (Replication Transparency)
  A felhasználó (és tulajdonképpen a futó programok) előtt rejtett módon biztosít több másolatot az információs objektumokból, így növelve a rendszer teljesítményét és megbízhatóságát.

- Hiba átlátszóság (Failure Transparency)
  A felhasználó (és tulajdonképpen a futó programok) előtt rejtett módon biztosítja a meghibásodások kezelését, lehetővé téve a rendszer használatát hardver- illetve szoftver meghibásodások esetén is.
- Vándorlási átlátszóság (Migration Transparency)
  A felhasználó (és tulajdonképpen a futó programok) előtt rejtett módon, az érintett programok működését nem befolyásolva biztosítja az információs objektumok szabad mozgatását a rendszerben.
- Teljesítmény átlátszóság (Performance Transparency)
  A felhasználó (és tulajdonképpen a futó programok) előtt rejtett módon teszi lehetővé azt, hogy a terhelések változására a rendszer folyamatos ön-átkonfigurálással legyen képes reagálni, így tartva fenn (esetleg növelve) a rendszer teljesítményét.
- Skálázási átlátszóság (Scaling Transparency)
  A felhasználó (és tulajdonképpen a futó programok) előtt rejtett módon biztosítja a rendszer bővíthetőségét, oly módon, hogy közben a rendszer struktúrája és algoritmusai változatlanok maradhatnak.

A gyakorlatban a fenti átlátszóságok közül a hozzáférés- és a hely átlátszóság (azaz a hálózati átlátszóság) megléte a legkritikusabb az osztott erőforrások használhatóságának szempontjából, illetve a többi átlátszóság alapját is jellemzően ezek képezik. Nyilván a többi átlátszóság is fontos, de az átlátszóságok egy része eseti jellegű. (Például a skálázási átlátszóság megléte indifferens, ha nem bővítjük a rendszerünket.)

3. Az elosztott rendszerek operációs rendszerei

A valamilyen módon összekapcsolt rendszereken futó operációs rendszereket két fő csoportra oszthatjuk:

- hálózati operációs rendszerek
  - nem biztosítanak átlátszóságot
  - ez egyes csomópontok egyenként elérhetőek
  - programok futtathatók közvetlenül távoli csomópontokon
  - adatok közvetlenül mozgathatóak az egyes csomópontok között
- elosztott operációs rendszerek
  - átlátszóság biztosítása
  - az átlátszóság miatt
    - az egyes csomópontok egyenként nem címezhetőek
    - a programok futtatásának, futásának helye közvetlenül nem határozható meg
    - adatok közvetlen mozgatására a csomópontok között nincs lehetőség
    - számításvándorlás és folyamat vándorlás megvalósítható a terhelésnek megfelelően
    - a felhasználó nem feltétlenül ismeri fel, hogy elosztott rendszert használ

Elosztott fájl rendszerek

Elosztott fájl rendszernek jellemzően azt az operációs rendszerrel szorosan együttműködő szoftver réteget értjük, amely képes biztosítani a távoli fájlrendszerek tartalmának elérését és használatát a helyi fájlrendszer tartalmának elérésével megegyező módon. Azaz ez esetben röviden a fájlok elérésének átlátszóságáról van szó.

Az elosztott fájl rendszerek használatakor a következő tulajdonságokat várjuk el:

- Hozzáférési átlátszóság
  A helyi és a távoli fájlok azonos eljárással kezelhetők.
- Elhelyezkedési átlátszóság
  A fájlnevek nem hivatkoznak, nem mutatnak a fájlok fizikai elhelyezkedésére.
- Vándorlási átlátszóság
  A fájlok a rendszerben az elnevezésük megváltozatása nélkül mozgathatóak.

- Méretezési átlátszóság
  A fájlrendszer maga is méretezhető, azaz amennyiben ezt a terhelés indokolja új komponensekkel a mérete illetve a teljesítménye is növelhető.
- Hibatűrő fájlrendszer
  A fájlrendszer egyes komponenseinek hibái esetén a fájlrendszer legalapvetőbb funkcióinak továbbra is elérhetőnek kell maradniuk, ami esetenként csak kompromisszumok árán (csökkent funkcionalitás mellett) valósítható meg.
- A felhasználók mobilitása
  Egy adott jogosultságú felhasználó az összes olyan fájlt, melyet jogosult elérni, a rendszer bármely tetszőleges pontjáról valóban el is érheti.
- A fájlok mobilitása
  A fájlok a rendszerben futás közben is áthelyezhetők.

A fájlok elnevezése, azaz a fájlok állapottere elosztott fájlrendszerek esetében kétféle megvalósítású lehet:

- Egységes (uniform)
  Ebben az esetben az azonos fájlok azonos névvel érhetők el a különböző hosztokon.
- Nem egységes (nem uniform)
  Ebben az esetben az azonos fájlok különböző névvel érhetők el a különböző hosztokon.

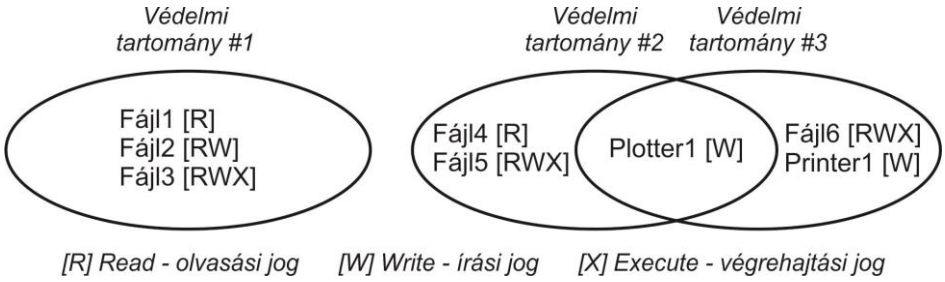
4. Védelem és biztonság
   Az erőforrás megosztás és a biztonság kérdései
   A SETI program valóban „magasztos” céloktól vezérelt elképzelése volt az erőforrás megosztás önkéntes módja. De, ez lett az alapja a ma egyik jelentős támadási módszerének, az erőforrás lopásnak. (trójai → majd az előadás vége felé jön…)

   SW illetve HW védelem kapcsán egy kis kitekintés klasszikus irányba:
   Decimus Iunius Iuvenalis (Aquinum, kb. i.u. 50 – ?, kb. i.u. 130) Számos szókapcsolata máig szállóigeként él, mint például a „Quis custodiet ipsos custodes?” - „Ki őrzi az őrzőket?” vagy a „Mens sana in corpore sano” - „Ép testben ép lélek”, illetve a „Panem et circenses” - „Kenyeret és cirkuszt”.
   A „Ki őrzi az őrzőket?” más megközelítésben: Például egy szultán szempontjából az, hogy a háremhölgyek vigyáztak egymásra az egy szoftveres védelem, de a háremhölgyet őrző eunuch már hardveres védelmet jelent ☺.

A védelem nem jelent mást, mint az erőforrások elérésének egy előre megállapított szabályrendszer szerinti elérési módját. A biztonság, mint fogalom, a rendszerben tárolt adatok sérthetetlenségét fejezi ki. Minkét esetben szintekről beszélünk.
Védelem illetve a biztonság mindig több tényezős:

- szoftveres tényező
- hardveres tényező
- emberi tényező

Ha van védelem, nyilván kell lennie fenyegetésnek is, mely lehet belső, ha a rendszerben futó (esetleg kéretlen) folyamatokból fakad, illetve lehet külső, ha rendszertől független tényezők a forrásai.
Bármely védelem alapja tehát egy szabályrendszer, a jogosultságok hierarchikus rendszere. Ideális esetben minden folyamatnak illetve erőforrásnak csak az éppen szükséges jogosultságokkal kell rendelkeznie. Minden elemnek egyéni azonosítója van, és előre definiált műveletekkel érhető el. A valóságban az elosztott rendszereken belül védelmi tartományokat (Domain) használunk a szabályzásra, statikus vagy dinamikus elérési mátrix-szal. A statikus mátrix a folyamat életciklusa alatt nem változik, a dinamikus képes bővülni, szűkülni. Védelmi tartomány megadja, hogy az adott folyamat az adott állapotában mely erőforrásokhoz férhet hozzá és azokon milyen műveleteket végezhet.



Dinamikus védelmi tartományos esetén szabályozni kell a tartomány váltás vagy megváltoztatás módját.

- Védelmi tartomány váltás (Switch)
  Azt kell definiálni, hogy mely tartományokból melyik másik tartományba lehet átlépni azaz védelmi tartományt váltani.
- Elérési jogosítványok másolása (Copy, Transfer, Limited Copy)
  Ebben az esetben egy folyamatnak lehetőséget adunk arra, hogy adott védelmi tartományban lévő védelmi jogosítványát egy másik védelmi tartományba átmásolhassa. Ez innentől további lehetőségeket is felvet: a jogot átadjuk (Transfer) vagy átmásoljuk (Copy), illetve, hogy a másolt jog továbbmásolható lesz-e.

- Objektumok tulajdonlása (Owner)

  Az erőforrás tulajdonosa más tartományokban is adhat vagy törölhet az erőforráshoz tartozó védelmi jogokat, műveleteket. A mátrixban az oszlopok mentén történő módosítások tartoznak ide.
- Vezérlő (Control)

  A mátrixban a sorok mentén történő módosítások tartoznak ide. Ha egy tartomány ilyen módon egy másik tartományt is vezérel, akkor annak sorában adhat vagy törölhet jogosultságokat.

Az elérési mátrix

A mátrix sorait tartományok, oszlopait statikus védelmi tartományok esetén (első táblázat) csak objektumok, illetve dinamikus védelmi tartományok esetén (második táblázat) tartományok is alkotják, az egyes cellákban pedig a kapcsolódó jogok helyezkednek el.

Tartományok	Objektumok							
	Fájl1	Fájl2	Fájl3	Fájl4	Fájl5	Fájl6	Printer1	Plotter1
1	olvasás	írás, olvasás	írás, olvasás, végrehajtás					
2				írás	írás, olvasás, végrehajtás			írás (nyomtatás)
3						írás, olvasás, végrehajtás	írás (nyomtatás)	írás (nyomtatás)

Tartományok		Objektumok								Tartományok		
		Fájl1	Fájl2	Fájl3	Fájl4	Fájl5	Fájl6	Printer1	Plotter1	1	2	3
	1	olvasás	írás, olvasás	írás, olvasás, végrehajtás							váltás	
	2				írás	írás, olvasás, végrehajtás			írás (nyomtatás)	váltás		váltás
	3						írás, olvasás, végrehajtás	írás (nyomtatás)	írás (nyomtatás)	váltás		

Jellemző a nagyszámú védelmi tartomány és a nagyszámú objektum, melyekhez relatíve kisszámú jogosultság tartozik. Ezek alapján a mátrix mérete szélsőségesen nagy is lehet, miközben kitöltése jellemzően ritka, ezért célszerű a várható méretéhez és a várható tartalmához optimalizálni a mátrix tárolási módját.

- Globális tábla

  A legegyszerűbb tárolási mód, ezért mérete nagy, feldolgozása hosszú. Egy listába gyűjtve tartalmazza a következő hármas adatcsoportokat: „tartomány; objektum; művelet végzési jog”.

- Objektum elérési lista
  A mátrixot az oszlopai mentén fel kell darabolni, majd objektumonként a következő kettes adatcsoportot kell eltárolni: „tartomány; művelet végzési jog”. Ez a globális táblánál gyorsabb feldolgozást tesz lehetővé.
- Tartományok jogosítványainak listája
  A mátrixot a sorai mentén fel kell darabolni, majd tartományonként a következő kettes adatcsoportot kell eltárolni: „objektum; művelet végzési jog”. Ez a globális táblánál gyorsabb feldolgozást tesz lehetővé, különösen a tartományok szerinti összehasonlításban.
- Zár-kulcs módszer
  Az előző két módszer elvi ötvözete. Minden objektumhoz egyéni bitminták tartoznak, ezekre mint „zár” hivatkozunk. A védelmi tartományokhoz pedig egy vagy akár több bitminta is tartozhat, melyekre mint „kulcs” hivatkozunk. Amennyiben egy „kulcs” illik az adott „zár”-ba, akkor az adott védelmi tartományban lévő folyamat képes végrehajtani a vonatkozó műveletet.

Védelem a kártékony programok ellen

A kártékony programok egy része észrevétlenül jut a rendszerbe („A” eset), másik részét a felhasználó maga engedi be („B” eset).

A védelemnek átfogónak kell lennie. Ez egy egyszerű példán keresztül könnyen megérthető. Vegyünk egy embert, akinek kancsal a szeme, fáj a foga és el van törve a lába. Mire van ennek az embernek szüksége? Szemüvegre, fogfájás csillapítóra, és arra, hogy begipszeljék a lábát, de nyilván a szemüvegtől nem múlik el a fogfájása… Egyik eszköz/folyamat sem helyettesíti a másikat.

Ennek analógiájára, a tűzfal képes portot blokkolni, de a vírust nem ismeri fel. Az adathalászat elleni program megvédi pl. a bankkártya számunkat, de nem véd meg a címhamisítás ellen, stb.

Ezért a védelemnek mindenképpen több lábon kell állnia.

A kártevőket a tevékenységük, a szaporodásuk illetve terjedésük módja szerint is csoportosíthatjuk.

„A” eset

- Vírus
  Ez a klasszikus támadás egy számítógép ellen. Az első vírus címért hivatalos források szerint dúl a harc. 1982-ben, a 15 éves iskolás Rich Skrenta egy számítógépes partin (ez még LAN nélküli party volt) szerette volna megtréfálni barátait. A gépeken Apple II operációs rendszer futott, és az

Elk Cloner nevet kapott vírus floppy lemezen terjedt. Működése alapján ezt a vírust tekinthetjük a boot vírus elődjének. Amikor a számítógép a fertőzött floppy lemezről indult, a vírus bekerült a memóriába, s amikor egy fertőzés mentes lemez került a meghajtóba, akkor oda kimásolódott, így terjedt gépről-gépre. (már akkor is jelen volt a programok illegális másolása, ezért tudott terjedni) A megfertőzött gép minden 50-dik indításnál egy poénos szöveget írt a képernyőre. Az első igazán komoly vírust (állítólag) palesztin informatikusok írták egy izraeli hadi gyár megzavarására. De kapott vírus már iráni uránium centrifuga is…

A vírusok általában file (lehet akár adat file, vagy futtatható állomány, stb.) vagy boot szektor szinten fertőznek, de előfordul, hogy olyan HDD helyre íródnak be, amihez a felhasználó nem is fér hozzá. A vírus egyik legfontosabb tulajdonsága a szaporodás. A vírus aktiválódhat azonnal, de aktivizálódása akár dátumhoz, vagy bekapcsolások számához, stb. is köthető.

- Féreg (Worm)

  A féreg a vírus közvetlen leszármazottja. Szinte minden alapvető tulajdonságban egyezik a vírussal, ám van egy nagy különbség, ami a férgeket rendkívül sikeressé teszi: a férgek – a vírusokkal ellentétben – képesek „önálló" akciókra, a felhasználó közreműködése nélkül is tudnak terjedni. Nem fertőznek meg a gépen állományokat, a céljuk a gyors terjedés, ezért különösen szeretik a levelezőprogramokat, és benne a személyes leveleződlistákat. (mostanában a közösségi oldalakat is kezdik felfedezni maguknak) Jellemző példa, hogy amikor egy féreg megérkezik – általában egy e-mail csatolmányba rejtve – és mi azt megnyitjuk, azaz aktiváljuk a férget, a féreg automatikusan továbbküldi magát a listánkban található összes címzettnek.

- Adathalászat (Phishing)

  Személyes adatok, bankkártya információk ellen irányul. Jellemzően web-es alkalmazások, vagy weboldalak hamisításával csapják be a felhasználót.

„B” eset

- Trójai és Erőforrás halászat

  Trójai faló → Homérosz, Iliász, Odüsszeusz, Trója, Laokoon ismert története alapján. Az óvatlan felhasználó maga juttatja a gépébe a kártékony programot, leggyakrabban nyilvános vagy demo programokhoz kapcsolódóan. Létezik olyan változat is, ami önálló programnak látszik, de funkcióit a felhasználó felé csak nagyon kis részben (főleg csak az álcázás

miatt teljesíti). Pl. aktuális, hogy az NSA-nak létezik olyan megfigyelő programja, ami Firefox böngészőnek látszik, és szinte ugyanúgy is viselkedik! De létezik szinte minden népszerű programnak az álcázott verziója…

Az erőforrás halászat a trójai programok speciális működése. A már tárgyalt erőforrás megosztás kéretlen és agresszív esete. A felhasználó annyit érzékel, hogy lassabb lett a gépe, az okot nem ismeri. Valójában a gépe erőforrásain már nem egyedül rendelkezik, sőt, neki csak a maradék jut.

- Megtévesztő programok (Misleading Application)
  Igaziból nem létező hibára, fertőzésre hívja fel a felhasználó figyelmét, segítő szándéknak tűnő módon. Sőt, megoldást is kínál, miszerint az általa ajánlott alkalmazás „megtisztítja” a gépet…
- Címhamisítás (Address Spoofing)
  Ebben az estben például olyan email érkezik hozzánk melynek a feladója látszólag egy ismerősünk (aki a címlistánkban is szerepel) vagy egy ismert közintézmény, de valójában az igazi feladó, aki nem tiszta szándékkal küldte a levelet, amiben esetleg segítséget, pénzt, egy adott oldalra való belépést stb. kér, vagy belépési kódot ajánl fel, ismeretlen.
- Pszichológiai manipuláció (Social Engineering)
  Összefoglaló néven így hivatkozunk a felhasználó bizalma ellen intézett támadásokra. A következő (nem is feltétlenül csak számítógép segítségével végrehajtott) tevékenységeket soroljuk ide:
  - Zsarolóvírus (Ransomware)
  - Üzenetküldésen keresztüli adathalászat (Messaging, Smishing)
  - Személyre szabott csaló támadás (Personalized Scams)
  - Vállalatvezető nevében elkövetett csalás (CEO Fraud)
  - Telefonhívásos támadás, csalás (Phone Call Attack, Scams)
  - Közösségi oldalon keresztül elkövetett csalás (Scamming You Trough Social Media)
  - Információ ellesése (Shoulder Surfing)
  - Kattintásvadász oldalak (Clickbait)