FELADATKIÍRÁS

A piaci termékek előállításánál gyakran a költségek kétharmadát a szoftverfejlesztés teszi ki. Ezért a beágyazott alkalmazásoknál egyre inkább megszokottá válik beágyazott operációs rendszerek használata. Az összetett hardverek alkalmazása, a kód újrahasznosíthatósága, a csoportban történő fejlesztés és a multitaszk igénye szintén szükségessé teszi valamilyen operációs rendszer alkalmazását.

A feladat célja különböző operációs rendszerek megismerése, előnyeik és hátrányaik kitapasztalása és egy ipari alkalmazáson keresztül való összehasonlítása. A hallgató feladatának a következőkre kell kiterjednie:

- Adjon áttekintést az elterjedt operációs rendszerek
  - felépítéséről,
  - működéséről,
  - előnyeiről,
  - hátrányairól!
- Különböző fejlesztőkártyák segítségével hozzon létre beágyazott operációs rendszeres alkalmazást!
- A feladat megoldása során használjon különböző komplexitású és erőforrás-igényű operációs rendszereket!
- Hasonlítsa össze az operációs rendszereket különböző szempontok alapján!
- A hallgató végezzen irodalomkutatást a teljesítménymutatók mérésének témakörében!
- Tegyen javaslatot az összehasonlítás alapját adó metrikákra és végezze el az összehasonlító teszteket!
- Adjon javaslatot az elemzett operációs rendszerek felhasználásának területeire!

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Operációs rendszerek összehasonlítása mikrovezérlős rendszerekben

Diplomaterv

Készítette
Bálint Ádám

Egyetemi konzulens
Szabó Zoltán

Céges konzulens
Mikó Gyula

2016. december 3.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Bálint Ádám*, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2016. december 3.

Bálint Ádám
hallgató

Kivonat

Jelen dokumentum egy diplomaterv sablon, amely formai keretet ad a BME Villamosmérnöki és Informatikai Karán végző hallgatók által elkészítendő szakdolgozatnak és diplomatervnek. A sablon használata opcionális. Ez a sablon LaTeX alapú, a *TeXLive* TeX-implementációval és a PDF-LaTeX fordítóval működőképes.

Abstract

This document is a LaTeX-based skeleton for BSc/MSc theses of students at the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics. The usage of this skeleton is optional. It has been tested with the *TeXLive* TeX implementation, and it requires the PDF-LaTeX compiler.

Bevezető

A bevezető tartalmazza a diplomaterv-kiírás elemzését, történelmi előzményeit, a feladat indokoltságát (a motiváció leírását), az eddigi megoldásokat, és ennek tükrében a hallgató megoldásának összefoglalását.

A bevezető szokás szerint a diplomaterv felépítésével záródik, azaz annak rövid leírásával, hogy melyik fejezet mivel foglalkozik.

1. fejezet

Elméleti áttekintés

1.1. Általánosan használt licencek

A mindennapi életben naponta szembesülünk a különböző gyártók vagy fejlesztők által elérhetővé tett programokkal, melyek mindegyike tartalmaz valamilyen megkötést a program használatával kapcsolatban. Ezek között találhatóak egyedileg alkalmazott feltételek, de vannak elterjedt licencek, melyek például nyílt forráskódú szoftverek esetén gyakran alkalmaznak.

1.1.1. Zárt forráskódú szoftver

Zárt forráskódú szoftver esetén a forráskód kizárólag a gyártó/fejlesztő kezében marad. A szoftver használatáért általában valamekkora összeget kell fizetni, amely gyakran magában foglalja a fellépő problémák megoldásában való segítségnyújtást is. Természetesen a zárt forráskód nem vonja kötelezően maga után, hogy a felhasználónak fizetnie kell a szoftver használatáért. Sok gyártó a termékét ingyen elérhetővé teszi (freeware). Ilyen esetben a jövedelemszerzési felhasználásból és/vagy a segítségnyújtásból származik a fejlesztést fedező bevétel. A gyártó a felhasználás feltételeit saját maga szabhatja meg, így ebben az esetben nem lehet általános elemzést végezni a licencekkel kapcsolatban.

1.1.2. Nyílt forráskódú szoftver

A szoftverek másik nagy csoportját alkotják a nyílt forráskódú szoftverek. Ekkor a forráskód publikusan elérhető. A nyílt forráskódú szoftverek esetén is lehetőség van egyéni licenc használatára, azonban a szoftverek többségénél elterjedt, általánosan használt licencekkel találkozunk.

MIT License

Az MIT licenc a legegyzerűbb licencek egyike. A licenc alá eső forráskód szabadon másolható, módosítható, terjeszthető, akár más licenc alá helyezhető.

Apache License 2.0

[ToDo]

BSD licenc

A BSD licenc eredetileg egy egyszerű és szabad felfogású licenc számítógép szoftverek számára, amit először 1980-ban használtak a BSD UNIX operációs rendszerhez. Az idő előrehaladtával több módosítást kellett végrehajtani a licenc szövegezésén.

4-clause BSD (eredeti)

Az eredeti, másnéven 4-clause BSD licenc négy pontban foglalta össze a felhasználás feltételeit. A licenc tartalmazza a szerzői jog keletkezésének évét, a fejlesztő szervezet nevét és a szerzői jog tulajdonosának nevét. A licenc megengedi a szoftver használatát mind forrás, mind bináris formában, akár módosítással, akár anélkül, amennyiben a felhasználás feltételei teljesülnek. A szöveg azonban tartalmaz egy megkötést, amely a használat során gyakran kellemetlenségként jött elő a felhasználók körében: amennyiben a szoftver bármilyen részére vagy a szoftver használatára hivatkozás történik egy reklámanyagban, úgy a szoftver fejlesztőjét fel kellett tüntetni a szövegben. Amennyiben több, eredeti BSD licenc alá eső szoftvert tartalmazó rendszer került hivatkozásra, úgy az említett lista hamar nagy méretűvé válhatott. Ezen kellemetlenség miatt került először módosításra a BSD licenc.

A négy pont az alábbi feltételeket szabja a feljhasználó számára:

1. A forráskódnak tartalmaznia kell az copyright szövegét, a felhasználás feltételeit felsorakoztató listát, illetve a nyilatkozatot.

2. Bináris formában történő terjesztés esetén a a copyright szövegét, a felhasználás feltételeit felsorakoztató listát, illetve a nyilatkoztatot a dokumentációban és/vagy valamely másik, a terjesztett szoftverhez adott anyagban fel kell tüntetni.

3. Bármiféle hirdetési anyagban, ami a szoftver jellemzőire vagy használatára hivatkozik, fel kell tüntetnie az alábbi elismerést: *Ez a termék a <szervezet> által fejlesztett szoftvert tartalmaz.*

4. Sem a <szervezet> neve, sem a közreműködő partneinek neve nem használható fel a szoftverből származó termék népszerűsítésére vagy ajánlására erre vonatkozó írásbeli engedély nélkül.

A nyilatkozatban tisztázásra kerül, hogy a szerzői jog tulajdonosa nem vonható felelősségre a szoftver használatából, felhasználásából eredő üzemkimaradás, adatvesztés, profitvesztés vagy egyéb veszteség bekövetkezése esetén.

Ez a licencelési forma ma már nem elterjedt, inkább a módosított (3-clause BSD) vagy az egyszerűsített (2-clause BSD) licenc használata ajánlott.

3-clause BSD (módosított)

Az előző, eredeti verzióhoz képest a különbség, hogy eltávolították a hirdetésre vonatkozó elismerési feltételt, valamint a nyilatkozat szövegében a közreműködőket is mentesíti minden garanciális felelősség alól.

2-clause BSD (egyszerűsített)

Az egyszerűsített (vagy gyakran FreeBSD licencnek is nevezett) BSD licenc két pontban foglalja össze a felhasználás feltételeit, amit a módosított licencből a népszerűsítésre vonatkozó pont eltávolításával kaptak.

GNU GPL

A GNU General Public Licence azzal a céllal jött létre, hogy garantálja egy program szabad módosítását és terjesztését, illetve hogy az szabad szoftver maradjon a felhasználói számára. Az licenc az angol free kifejezést nem a termék árára, hanem annak szabad felhasználására használja.

A licenc több módosításon keresztül ment. A két legutolsó változatot ismertetem a továbbiakban.

GNU GPLv2

A GNU GPLv2 licenc megengedi a program szabad terjesztését és módosítását, amennyiben a terjesztett/módosított örökli az eredeti szerzői jogi megjegyzéseket. A garanciális kötelezettségek elutasításra kerülnek, de nem tiltja meg (akár ellenszolgáltatás fejében) a vállalásukat.

Ha módosítás történt a programon, akkor fel kell tüntetni a módosító nevét és a módosítás dátumát.

A programtól elkülöníthető munkára nem kötelező kiterjeszteni a licenc hatályát, amennyiben az külön kerül terjesztésre. Ha az eredeti (szerzői jogok alá eső) munkával együtt kerül terjesztésre, akkor automatikusan kiterjesztésre kerülnek a jogok.

A program terjeszthető futtatható, bináris formában is, amennyiben a forráskód publikusan elérhető marad.

A felhasználó jogait tovább korlátozni nem lehet.

Ha valamilyen okból kifolyólag (például bírói végzés folytán) a terjesztés csak bináris formában lehetséges, akkor a program nem terjeszthető.

Ha valamely országban a terjesztés nem lehetséges, úgy a szerzői jogok eredeti tulajdonosa földrajzi megkötést adhat a terjesztésre vonatkozóan, ami a licenc teljes értékű részét képezi.

A programot más, szerzői jogi szabályozásában különböző szoftverbe csak a szerzőtől kapott engedély birtokában lehet.

A GPLv2 licenc nem engedi meg, hogy a program része legyen szellemi tulajdont képező szoftvernek. Ebben az esetben az LGPL licenc használata javasolt.

GNU GPLv3

A GPL 2007-es módosítása során a tartalmi változtatásokon kívül formai változtatás is történt, ami a megérthetőséget hivatott szolgálni.

A tartalmi változtatások megcélozzák a más licencekkel való használat megkönnyítését, a licenccel védett szoftverhez járó felhasználó termék módosított programmal való használatának biztosítását (tivoizáció[lábjegyzet] elkerülése), jobban specifikálja a jogok elvesztését, illetve azok visszaszerzésének lehetőségeit, illetve megjelenik a *megkülönböztető* licenc definíciója, ami nem tartalmazza, vagy megtiltja valamely, az eredeti licencben foglalt jog gyakorlását.

1.2. Operációs rendszer feladatai

Az operációs rendszer elsődleges feladata a használt processzor perifériáinak kezelése, azokhoz meghatározott interfész biztosítása. További feladata még a létrehozott taszkok ütemezése, a taszkok közötti kommunikáció és szinkronizáció megvalósítása.

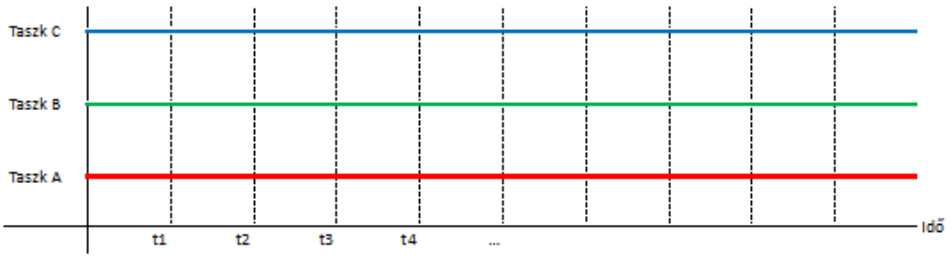
1.2.1. Operációs rendszer definíciója

Az operációs rendszer egy szoftver, ami kezeli a számítógép alap funkcióit és szolgáltatásokat biztosít más programok (vagy alkalmazások) számára. Az alkalmazások valósítják meg azt a funkcionalitást, amire a számítógép felhasználójának szüksége van, vagy a számítógép felhasználója akar. Az operációs rendszer által nyújtott szolgálatások az alkalmazások feljesztését egyszerűsítik, ezáltal felgyorsítják a fejlesztést és karbantarthatóbbá teszik a szoftvert.

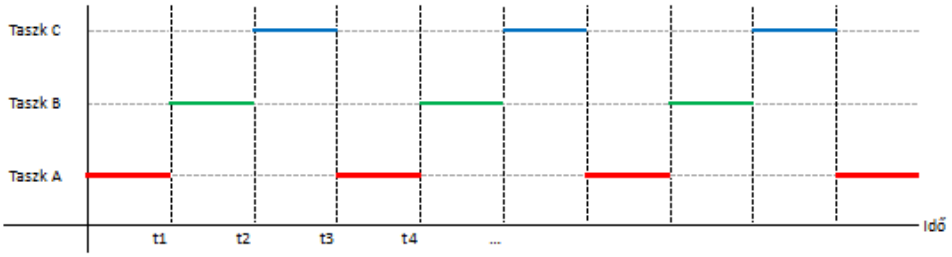
Többféle operációs rendszert különböztetünk meg a futtatható folyamatok, az egyidejűleg kezelt felhasználók számának és az ütemező működésének függvényében. A továbbiakban csak a multitaszkot támogató operációs rendszerekkel foglalkozom.

Valós idejű operációs rendszer

A legtöbb operációs rendszer látszólag lehetővé teszi több program egyidejű futtatását (multitasking). A valóságban minden processzor mag csak egy szálat tud egy időpillanatban futtatni. Az operációs rendszer ütemezője a felelős azért, hogy eldöntse, melyik program mikor fusson, és a programok közti gyors váltással éri el az egyidejű futás látszatát.



1.1. ábra. *Látszólag a folyamatok párhuzamosan futnak.*



1.2. ábra. *A valóságban minden folyamat egy kis időszeletet kap a processzortól.*

Az operációs rendszer típusát az ütemező döntési mechanizmusa határozza meg. Egy real time operációs rendszer ütemezője úgy van megtervezve, hogy a végrehajtási minta determinisztikus legyen. Ez beágyazott rendszerek esetén érdekes, mert a beágyazott rendszereknél gyakran követelmény a valósidejűség, vagyis hogy a rendszernek egy szigorúan meghatározott időn belül reagálnia kell egy adott eseményre. A. valósidejű követelményeknek való megfelelés csak úgy lehetséges, ha az operációs rendszer ütemezője előre megjósolható döntéseket hoz.

Valósidejű operációs rendszerek közül megkülönböztetjük a soft real-time és a hard real-time rendszereket.

Soft real-time rendszer esetén nem probléma, ha nem érkezik válasz a megadott határidőn belül, az csak a rendszer minősítését rontja.

Hard real-time rendszer esetén viszont a rendszer alkalmatlanná válik a feladatra, ha a határidőt nem tudja betartani. Például egy személygépjármű légzsákjának késedelmes nyitása akár halálos következményekkel is járhat.

1.2.2. Ütemezés

Az operációs rendszer egyik meghatározó része a használt ütemező működésének elve, mely meghatározza az eszköz alkalmasságát egy adott feladatra. Az ütemező dönti el, hogy a futásra kész taszkok közül melyik futhat a következő ütemezési szakaszban. További feladata tasztváltáskor az éppen futó taszk állapotának elmentése és a futtatandó taszk állapotának betöltése. Az ütemező három okból futhat le:

- Egy taszk lemond a futás jogáról,
- Megszakítás érkezik (tick esemény, külső megszakítás),
- Egy tasz létrejött vagy befejeződött.

A taszkváltási mód szerint kétféle működést különböztetünk meg:

- Preemptív ütemezés: ekkor az ütemező megszakíthatja az éppen futó taszkot, amennyiben magasabb prioritású taszk futásra kész állapotban várakozik,
- Nem-preemptív vagy kooperatív ütemezés: ebben az esetben az ütemező csak akkor futtat új taszkot, ha az éppen futó taszk befejeződött vagy explicit lemond a futásról (blokkolódik vagy átadja a futás jogát).

A továbbiakban ismertetem néhány elterjedt ütemezési mechanizmus alapelvét.

First-come first-served (Kiszolgálás beérkezési sorrendben)

A taszkok futtatása az érkezési sorrendben történik. A beérkező taszkok egy sorba kerülnek, ahonnan sorrendben kapják meg a CPU használat jogát.

Az átlagos várakozási idő nagy lehet, ha egy időigényes folyamatra több gyors lefutású folyamat várakozik.

Nem preemptív.

Shortest Job First (Legrövidebb feladat először)

A várhatóan leggyorsabban lefutó taszk kerül futási állapotba ütemezés bekövetkezésekor. Helyes működés esetén az átalgos várakozási idő optimális lesz.

A gyakorlatban nehéz előre megjósolni egy taszk futási idejét.

Lehet preemptív és nem-preemptív is. Preemptív esetben Shortest Remaining Time First (Legrövidebb hátralevő idejű először).

Prioritásos ütemezés

A következő taszk kiválasztása a prioritása alapján történik. Növelni lehet a hatákonyságát más metódusok keverésével (például a prioritást a várható futási idejéből határozzuk meg; a prioritást növeljük az idő elteltével; a prioritást csökkentjük az eddigi futó állapotban töltött idő arányában).

Rossz tervezés esetén az alacsony prioritású feladatok nem jutnak processzoridőhöz (kiéheztetés - starvation).

Lehet preemptív és nem-preemptív is.

Round Robin

Időosztáson alapuló mechanizmus. A várakozási sor egy cirkuláris buffer. Minden taszk adott időszeletet kap, majd az időszelet lejártával a sorban következő taszk kapja meg a futás jogát.

Lehet preemptív és nem-preemptív is.

Hibrid ütemezés

A felsorolt ütemezési elveket akár keverve is lehet alkalmazni. Ilyen ütemezési mechanizmus például a többszintű sorok használata, ahol minden sor saját ütemezési algoritmussal rendelkezik, és egy külön algoritmus felel az egyes sorok arbitrációjáért.

1.2.3. Operációs rendszer által nyújtott szolgáltatások

Az operációs rendszer felelős az egyes taszkoknak szükséges memóriaterületek kezeléséért és a taszkok közötti kommunikáció megvalósításáért.

1.3. ábra. *Multilevel queue ütemezés.*

Memóriakezelés

Egy taszk létrehozásakor az operációs rendszer ossza ki a taszk számára a használható memóriaterület helyét és méretét, és ezt az információt a rendszernek tárolnia kell. Ha az adott taszk befejezi a futását (megszűnik), akkor az operációs rendszer feladata a taszkhoz tartozó memóriaterület felszabadítása is.

Amennyiben a taszk dinamikusan allokál memóriát futás közben, úgy ezen memória kezelése szintén az operációs rendszer feladatkörébe tartozik, viszont az egyszerűbb operációs rendszerek esetében a futás közben lefoglalt memória felszabadítása a taszk felelősége.

Atomi művelet

Bizonyos műveletek során szükség van a műveletsor szigorúan egymás utáni futtatására. Ilyen eset például a megosztott adatterületre való írás, amikor ha taszkváltás következik be az adatterület írása közben, akkor a tartalmazott adatt érvénytelen értéket vehet fel. Az ilyen, *oszthatatlan* műveletek nevezzük atomi műveleteknek.

A konzisztencia biztosítása céljából az atomi műveleteket *kritikus szakaszokba* kell ágyazni, ezzel jelezve az operációs rendszernek, hogy a műveletek végrehajtása alatt nem következhet be taszkváltás, bizonyos esetekben megszakítás sem.

Az operációs rendszerek a kritikus szakaszokat több szinten megvalósíthatják. Legszigorúbb esetben a megszakítások letiltásra kerülnek, és az ütemező a kritikus szakasz befejezéséig felfüggesztett állapotban van. A kritikus szakasz megvalósításának egy kevésbé drasztikus módja az ütemező letiltása. Ekkor a kódrészlet védett a más taszkok általi preemtálástól, viszont a megszakítások nem kerülnek letiltásra.

A kritikus szakaszt a lehető leggyorsabban el kell hagyni, mert különben a beérkező megszakítások és magasabb prioritású taszkok késleltetést szenvednek, ami rontja az alkalmazás hatékonyságát.

Kommunikációs objektumok

Az alkalmazások egymástól független taszkok konstrukciójából áll. Viszont ezeknek a taszkoknak ahhoz, hogy a feladatukat el tudják látni, gyakran kommunikálniuk kell egymással.

A felsorolt kommunikációs objektumok listája nem teljes. Bizonyos operációs rendszerek ezektől eltéró struktúrákat is használhatnak, és az itt felsorolt struktúrák implementációja is eltérhet a leírtaktól (természetesen az sem biztos, hogy implementálva van az adott objektum).

Szemafor

Két szemafor típust különböztetünk meg:

- Bináris szemafor, amikor a szemafor két értéket vehet fel,
- Számláló szemafor, mikor a szemafor több állapotot is felvehet.

A szemaforon két művelet értelmezett:

- A szemafor jelzése (elterjedt elnevezések: give, signal, post, V() művelet[lábjegyzet-> Dijkstra holland, verhogen]), amikor a szemafor értéke növelésre kerül.
- A szemafor elvétele (elterjedt elnevezések: take, wait, pend, P() művelet [lábjegyzet-> proberen]), amikor a szemafor először tesztelésre kerül, hogy tartalmaz-e elemet, ha igen, akkor az értékét csökkentjük, ha nem, akkor várakozunk addig, amíg valamelyik másik taszk elérhetővé nem teszi azt.

A szemafor látszólag egyszerűen helyettesíthető egy egyszerű változó (boolean vagy előjel nélküli egész) használatával, viszont a beépített szemafor struktúra atomi műveletként kerül kezelésre (így nem következik be ütemezés a szemafor tesztelése és állítása közben), illetve a rendszer automatikusan tudja kezelni a várakozó folyamatok állapotok közti mozgatását.

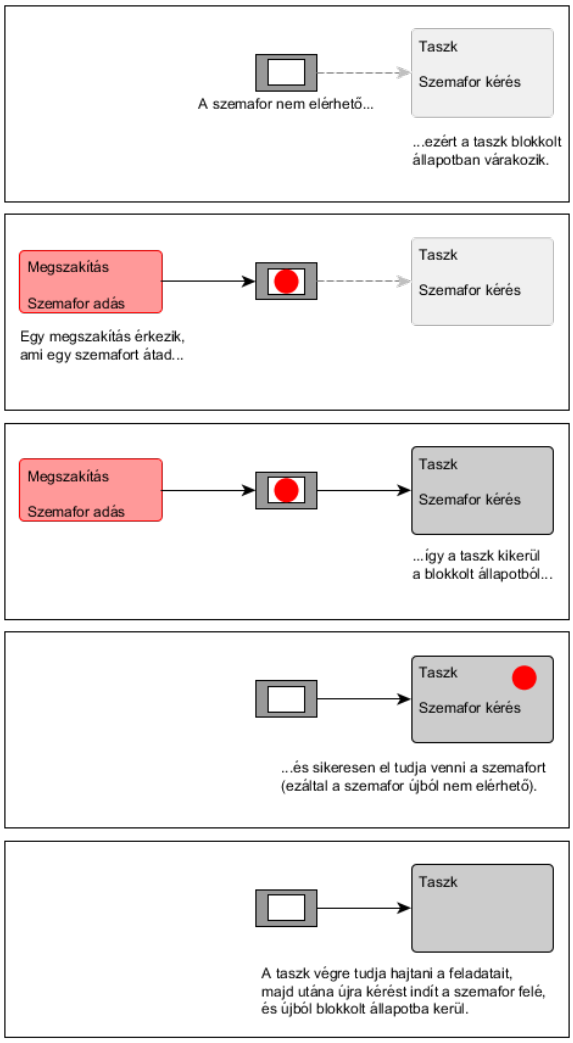
Szemaforokat leggyakrabban szinkronizációs célból, vagy erőforrások védelmére használnak.

Bináris szemafor

Bináris szemafor esetén a szemafor két értéket vehet fel. Felfogható úgy is, mint egy egy adat tárolására (egy elem hosszú) sor, melynek nem vizsgáljuk a tartalmazott értékét, csak azt, hogy éppen tartalmaz-e adatot vagy sem.

Leggyakoribb felhasználása a taszkok szinkronizálása. Ekkor az egyik taszk a futásának egy adott pontján várakozik egy másik taszk jelzésére. Ezzel a módszerrel megvalósítható a megszakítások taszkokban történő kezelése, ezzel is minimalizálva a megszakítási rutin hosszát.

Bináris szemafor használatakor különös figyelmet kell fordítani arra, hogy ha a szemafor egy adott taszkban gyakrabban kerül jelzésre, mint ahogy feldolgozzuk, akkor jelzések veszhetnek el (amíg az egyik jelzés várakozik, addig az utána következő eseményeknek nincs lehetőségük várakozó állapotba kerülni).



1.4. ábra. *Szinkronizáció bináris szemafor segítségével.*

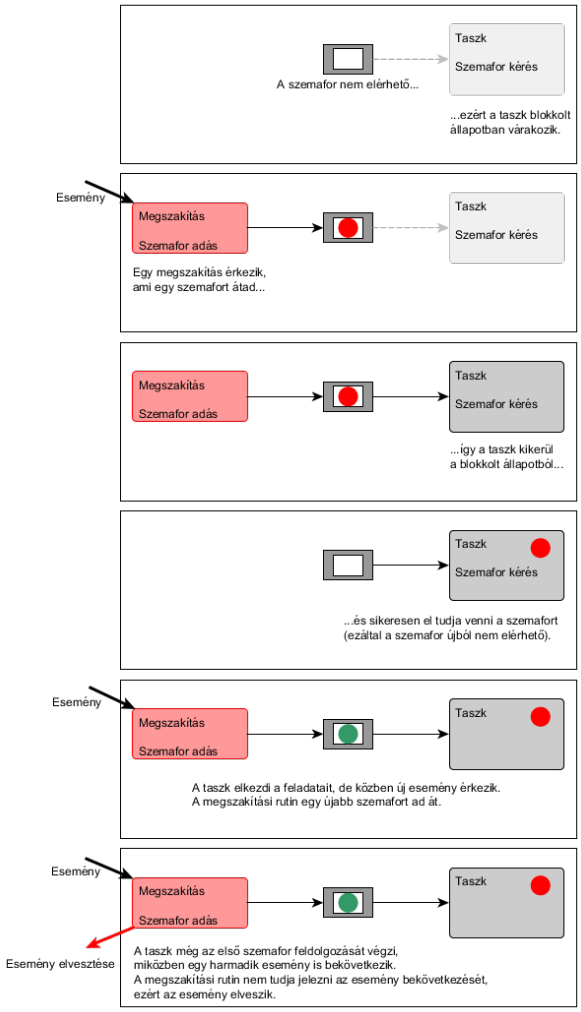
Számláló szemafor

A számláló típusú szemafor minden jelzéskor növeli az értékét. Ekkor (amíg el nem éri a maximális értékét) nem kerül Blokkolt állapotba a jelző taszk. A számláló szemafor felfogható úgy, mint egy egynél több adat tárolására képes sor, melynek nem vizsgáljuk az értékét, csak azt, hogy éppen tartalmaz-e még adatot vagy sem.

Két felhasználása elterjedt a számláló szemaforoknak:

- Események számlálása: ekkor minden esemény hatására növeljük a szemafor értékét (új elemet helyezünk a sorba). A szemafor aktuális értéke a beérkezett és a feldolgozott események különbsége. A számlálásra használt szemafor inicializálási értéke nulla.

- Erőforrás menedzsment: ekkor a szemafor értéke a rendelkezésre álló erőforrások számát mutatja. Mikor az operációs rendszertől az erőforrást igényeljük, akkor a szemafor értékét csökkentjük, mikor felszabadítjuk a birtolkolt erőforrást, akkor a szemafor értékét növeljük. Ha a szemafor értéke nulla, akkor nincs rendelkezésre álló erőfor-



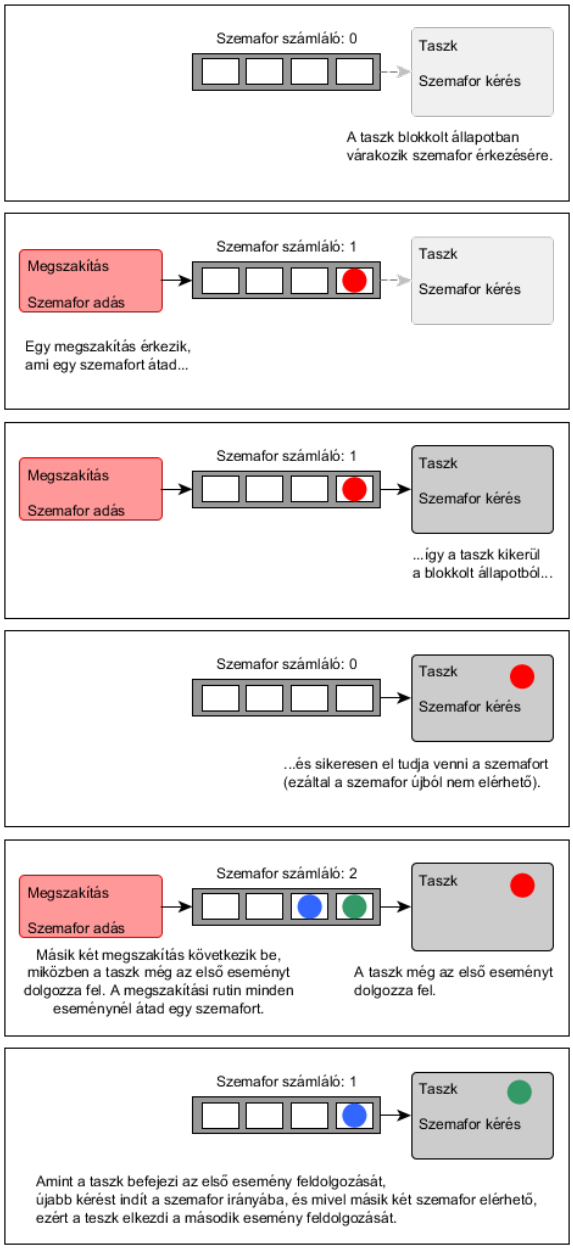
1.5. ábra. *Esemény bekövetkezésének elvesztése bináris szemafor használata során.*

rás. Erőforrások kezelésére használt számláló szemafor esetén az inicializálási érték az elérhető erőforrások száma.

Mutex

Taszkok vagy taszkok és megszakítási rutinok között megosztott erőforrás kezelésekor a Mutex (kölcsönös kizárás) használata indokolt. Mikor egy taszk vagy megszakítás hozzáférést indít egy erőforráshoz, akkor a hozzá tartozó mutex-et elkéri. Ha az erőforrás szabad, akkor az igénylő taszk megkapja a kezelés jogát, és mindaddig megtartja, amíg be nem fejezi az erőforrással való munkát. A mutex-et a lehető legkorábban (az erőforrással való munka befejeztével) fel kell szabadítani, ezzel is csökkentve az esetleges holtpont kialakulásának veszélyét.

Látható, hogy a mutex nagyon hasonlít a bináris szemaforhoz. A különbség abból adódik, hogy mivel a bináris szemafort leggyakrabban szinkronizációra használjuk, ezért azt nem kell felszabadítani: a jelző taszk vagy megszakítás jelzést ad a szemforon keresztül a feldolgozó taszknak. A feldolgozó taszk elveszi a szemafort, de a feldolgozás befejeztével a



1.6. ábra. *Számláló szemafor működésének szemléltetése.*

szemafort nem adja vissza.

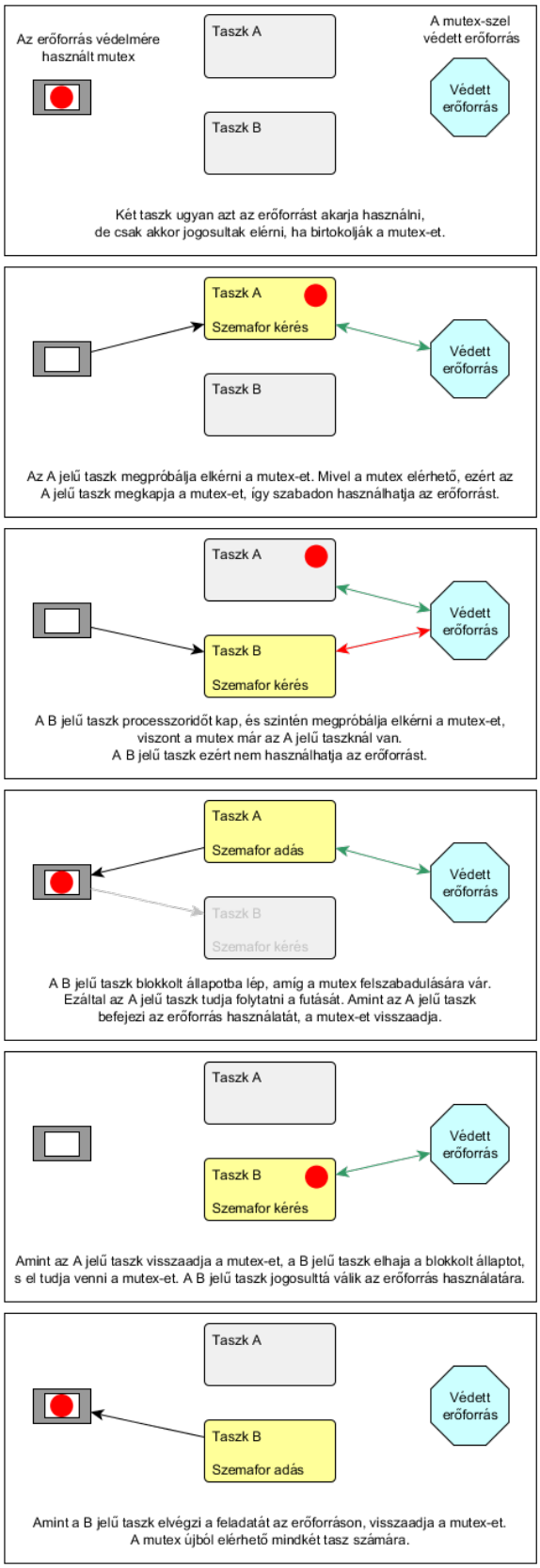
Sor (Queue)

A sorok fix méretű adatból tudnak véges számú üzenetet tárolni. Ezek a jellemzők a sor létrehozásakor kerülnek meghatározásra. Alapértelmezetten FIFO-ként működnek.

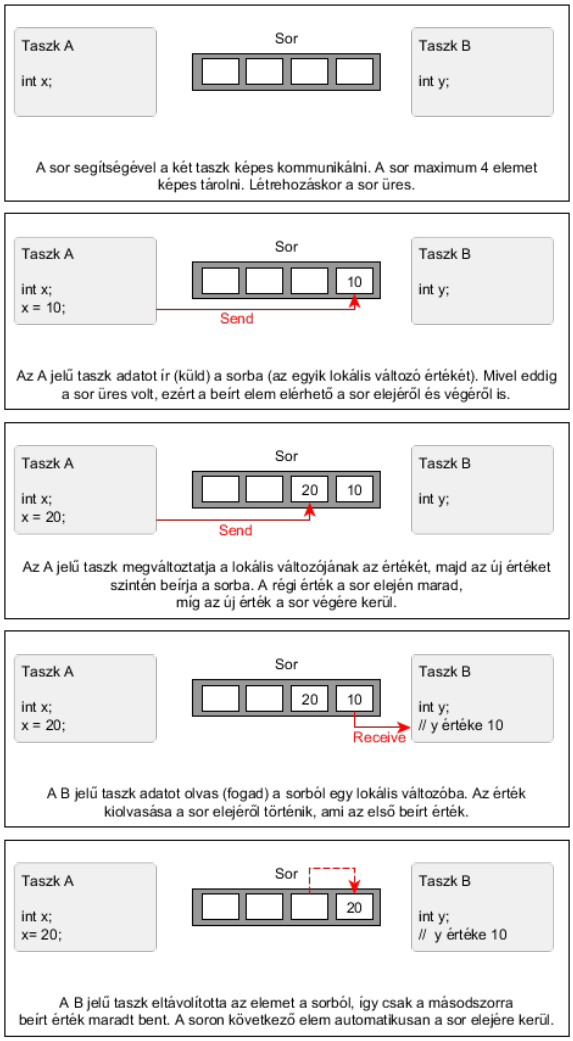
A sorba való írás során másolat készül az eredeti változóról, és ez a másolat kerül tárolásra a sorban.

Üzenet (Message)

Az üzenet három dolgot tartalmaz:



1.7. ábra. *Mutex működésének szemléltetése.*



1.8. ábra. *Sor működésének szemléltetése.*

- Az adatra mutató pointert,
- A mutatott adat méretét,
- Egy időbélyegzőt, ami megmutatja, mikor került az pointer a tárolóba.

A pointer mutathat egy egész adatterületre vagy akár függvényre is. A küldő és fogadó félnek tudnia kell, hogy az üzenet tartalma mire mutat.

Mivel a kommunikáció során csak az adat referenciája kerül átvitelre, ezért különös figyelmet kell fordítani az adat konzisztenciájára.

Események (Event flag)

Az események taszkok közötti szinkronizációra szolgálnak. Megvalósítható diszjunktív szinkronizáció, amikor több esemény küzül bármelyik bekövetkezése esetén a taszk futásra kész állapotba kerül (logikai VAGY), és megvalósítható konjuktív szinkronizáció, amikor az események mindegyikének bekövetkezése feltétele a taszk futásának (logikai ÉS).

1.2.4. Operációs rendszer használata esetén felmerülő problémák

Operációs rendszer használatával bizonyos problémák is felmerülhetnek egy alkalmazás fejlesztése során. Ezek közül látunk példákat a továbbiakban.

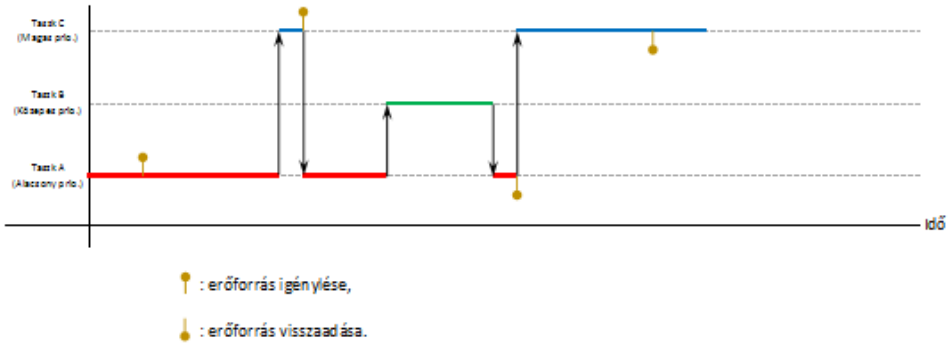
Kiéheztetés (Starving)

Prioritásos ütemezés esetén, ha egy magas prioritású taszk folyamatosan futásra kész állapotban van, akkor az alacsony prioritású taszkok sosem kapnak processzoridőt. Ezt a jelenséget nevezzük kiéheztetésnek (starving), amit átgondolt tervezéssel könnyedén elkerülhetünk.

Prioritás inverzió (Priority inversion)

Vegyünk egy esetet, mikor legalább három különböző prioritási szinten futó taszkot hozunk létre. A legalacsonyabb prioritásútól a magasabb felé haladva nevezzük őket *TaskA*-nak, *TaskB*-nek és *TaskC*-nek.

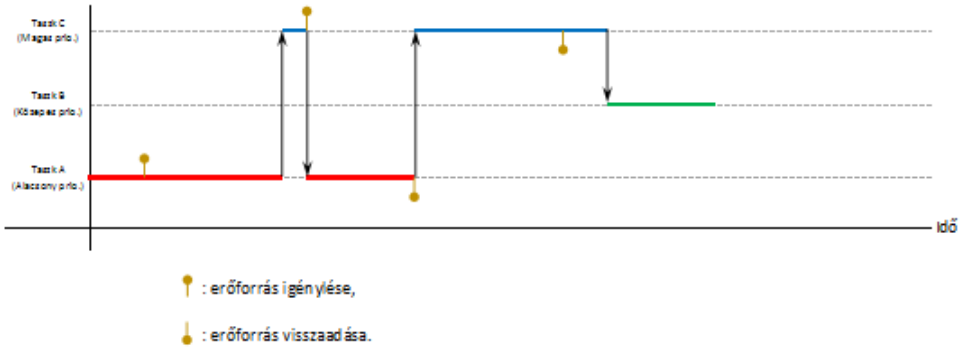
Vegyünk egy esetet, mikor legalább három, különböző prioritási szinten futó taszkot hozunk létre. Kezdetben csak a *TaskA* képes futni, ami egy mutex segítségével megkapja egy erőforrás használati jogát. Közben a *TaskC* futásra kész éllapotba kerül, ezért preemptálja a *TaskA*-t. A *TaskC* is használná az erőforrást, de mivel azt már a *TaskA* birtokolja, ezért várakozó állapotba kerül. Közben a *TaskB* is futásra kész állapotba került, és mivel magasabb a prioritása, mint a *TaskA*-nak, ezért megkapja a futás jogát. A *TaskA* csak a *TaskB* befejeződése (vagy blokkolódása) esetén kerül újra futó állapotba. Miután a *TaskA* befejezte az erőforrással a feladatait és felszabadítja azt, a *TaskC* újból futásra kész állapotba kerül, és preemptálja a *TaskA*-t.



1.9. ábra. *Prioritás inverzió jelensége.*

A vizsgált példa során a *TaskB* késleltette a *TaskC* futását azzal, hogy nem engedte a *TaskA*-nak az erőforrás felszabadítását. Így látszólag a *TaskB* magasabb prioritással rendelkezett, mint *TaskC*. Erre mondjuk, hogy prioritás inverzió lépett fel.

A prioritás inverzió problémájának egy megoldása a prioritás öröklés. Ekkor a magad prioritású taszk a saját prioritási szintjére emeli azt az alacsony prioritású taszkot, mely blokkolja a további futását. Amint a szükséges erőforrás felszabadul, az eredeti prioritási értékek kerülnek visszaállításra.



1.10. ábra. *Prioritás öröklés, mint a prioritás inverzió egyik megoldása.*

Holtpont (Deadlock)

Szemaforok és mutexek használata során alakulhat ki holtponti helyzet. Nézzük azt az esetet, hogy van két, azonos prioritású taszk (a taszkok prioritása itt nem lényeges), melyek működésük során ugyan azt a két erőforrást használják. A két taszkot nevezzük *TaskA*-nak és *TaskB*-nek, a két erőforrást pedig *ResA*-nak és *ResB*-nek.

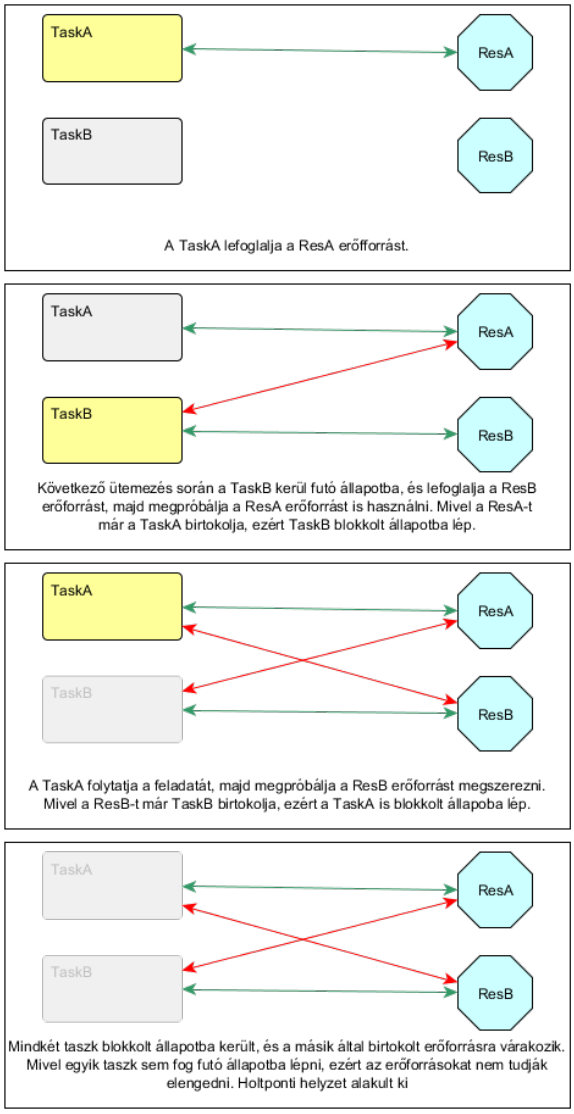
Induláskor a *TaskA* kapja meg a futás jogát, és lefoglalja a *ResA*-t. Közben lejár a *TaskA*-nak kiosztott időszelet, és *TaskB* kerül futó állapotba. A *TaskB* lefoglalja a *ResB* erőforrást, majd megpróbálja lefoglalni a *ResA* erőforrást is. Mivel a *ResA*-t már a *TaskA* használja, ezért a *TaskB* várakozó állapotba kerül. A *TaskA* újból megkapja a processzort, és hozzáférést kezdeményez a *ResB* erőforráshoz. Mivel a *ResB* erőforrást a *TaskB* folyamat birtokolja, ezért a *TaskA* is várakozó állapotba lép. Egyik folyamat sem tudja folytatni a feladatát, emiatt az erőforrásokat sem tudják felszabadítani.

A holtponti helyzetek elkerülésére és feloldására több szabály létezik, de beágyazott rendszereknél átgondolt tervezéssel, illetve időkorlát megadásával általában elkerülhető a kialakulásuk.

Újrahívható függvények (Reentrant functions)

Egy függvény reentráns (újrahívható), ha biztonságosan meghívható több különböző taszkból vagy megszakításból.

Minden taszk rendelkezik saját stack-kel és saját regiszterekkel. Ha egy függvény minden adatot a stack-jén vagy a regisztereiben tárol (vagyis nem használ globális és statikus változókat), akkor a függvény reentráns.



1.11. ábra. *Holtponti helyzet kialakulásának egy egyszerű példája.*

2. fejezet

Operációs rendszerek bemutatása

2.1. Használt fejlesztőkártyák

2.1.1. STM32F4 Discovery

Manapság egyre inkább teret nyernek maguknak az ARM alapú mikrokontrollerek, melyek nem csak nagy számítási kapacitásukkal, de egyre alacsonyabb árukkal szorítják ki versenytársaikat. Az egyik legelterjedtebb gyártó, az STMicroelectronics (továbbiakban STM) több fejlesztőkártyát is piacra bocsátott az elmúlt években, melyeken a különböző mikrokontrollerek képességeit ismerheti meg a fejlesztő. A kapható fejlesztőkártyák amellett, hogy megkíméli a fejlesztőt a saját hardver tervezésétől - így a tervezési hibából adódó problémák keresésétől is -, a legtöbb esetben a gyártó széles körű támogatást is nyújt a termékhez (mintaprogramok, fórumok, stb.).

Az STM által gyártott fejlesztőkártyák közül ár-érték arányának köszönhetően talán a leggyakrabban használt az STM32F407 Discovery kártya. A rajta található mikrokontroller rendelkezik a legtöbb alkalmazásban előforduló perifériák mindegyikével. A teljesség igénye nélkül:

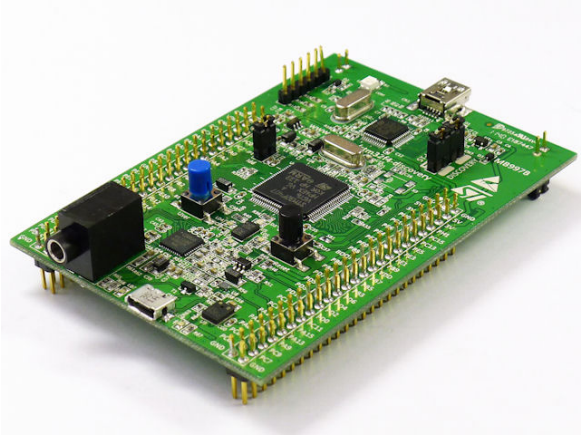
- GPIO-k,
- Soros kommunikációs portok (szinkron és aszinkron egyaránt),
- Ethernet port,
- Analog-Digital Converter,
- Digital-Analog Converter,
- Időzítők,
- High-Speed USB (OTG támogatással),
- SPI,
- I2C,
- I2S,

- SDIO (SD illetve MMC kártya kezeléséhez),
- CAN,
- Szoftveres és hardveres magszakítások.

A nagyszámú periféria mellett a mikrokontroller számítási kapacitása is kimagaslik a hasonló árkategóriájú eszközök köréből, köszönhetően az akár 168 MHz-es órajelének, a beépített CRC és lebegő pontos aritmetikai egységének, illetve a több perifériát is kezelni képes DMA-knak.

Az eszköz 1MByte Flash memóriával és 192kbyte RAM-mal rendelkezik.

A kártyán megtalálható több periféria, mely a különböző interfészek kipróbálását teszi lehetővé (mint például gyorsulásszenzor, mikrofon).



2.1. ábra. *STM32F4 Discovery fejlesztőkártya.*

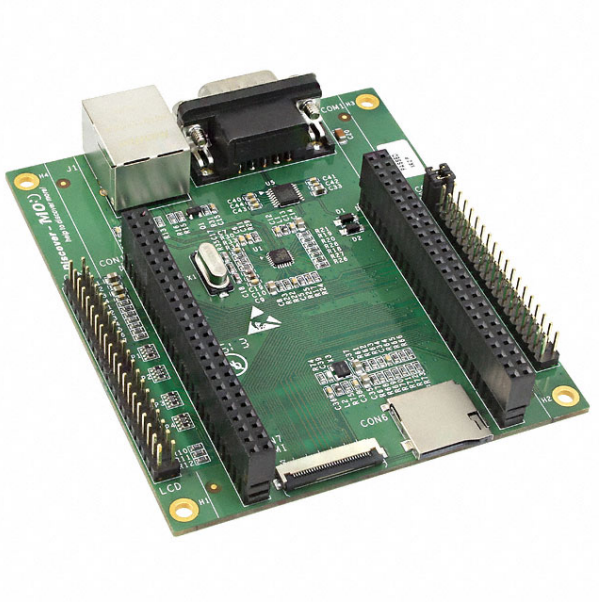
STM32F4 Discovery - Base Board

Az STM32F4 Discovery fejlesztőeszközhöz több kiegészítő kártya is kapható, melyek célja a kipróbálható perifériák számának növelése. Egyik ilyen bővítőkártya az STM32F4DIS-BB, ami tartalmaz microSD-kártya foglalatot, az Ethernet interfész fizikai rétegét megvalósító IC-t, illetve a csatlakoztatáshoz szükséges RJ45-ös csatlakozót. Ezen kívül kivezetésre került egy DB9-es csatlakozó - mely az egyik soros kommunikációs portot teszi elérhetővé -, egy FPC csatlakozó - mely kamera csatlakoztatását teszi lehetővé -, illetve az egyik oldali csatlakozósorra ráköthető 3,5̈-os TFT kijelző.

2.1.2. Raspberry Pi 3

A Raspberry Pi Foundation 2008-ban alapították azzal a céllal, hogy a informatikai tudományok területén segítse az oktatást.

Az első nagyteljesítményű, bankkártya méretű számítógépüket 2012 februárjában bocsájtották piacra, melynek ára töredéke volt az asztali számítógépekének. Azóta több verziója is kijött az eszköznek, melyet folyamatosan fejlesztettek mind teljesítményben, mind



2.2. ábra. *STM32F4 Discovery BaseBoard kiegészítő kártya.*

az integrált funkciók számában. 2015 novemberében a világ első 5 $-os számítógépével jelentek meg a piacon, melynek a Raspberry Pi Zero nevet adták.

A legújabb verziójú kártya, a Raspberry Pi 3 model B az előző verzióhoz képest erősebb processzort kapott, illetve tartalmaz beépített Bluetooth illetve WiFi modult.

2.3. ábra. *Raspberry Pi 3 bankkártya méretű PC.*

A hardver főbb jellemzői:

- 1,2GHz 64-bit quad-core ARMv8 CPU,
- 802.11n Wireless LAN,
- Bluetooth 4.1,

- Bluetooth Low Energy,
- 1GB RAM,
- 4 USB port,
- 40 GPIO pin,
- HDMI port,
- Ethernet port,
- Kombinált 3,5mm-es jack aljzat (audio és kompozit videó),
- MicroSD kártyafoglalat,
- VideoCore IV GPU.

Interfészek:

- SPI,
- UART,
- DPI (Display Parallel Interface),
- SDIO,
- PCM (Pulse-code Modulation),
- 1-WIRE,
- JTAG,
- GPCLK (General Purpose Clock),
- PWM.

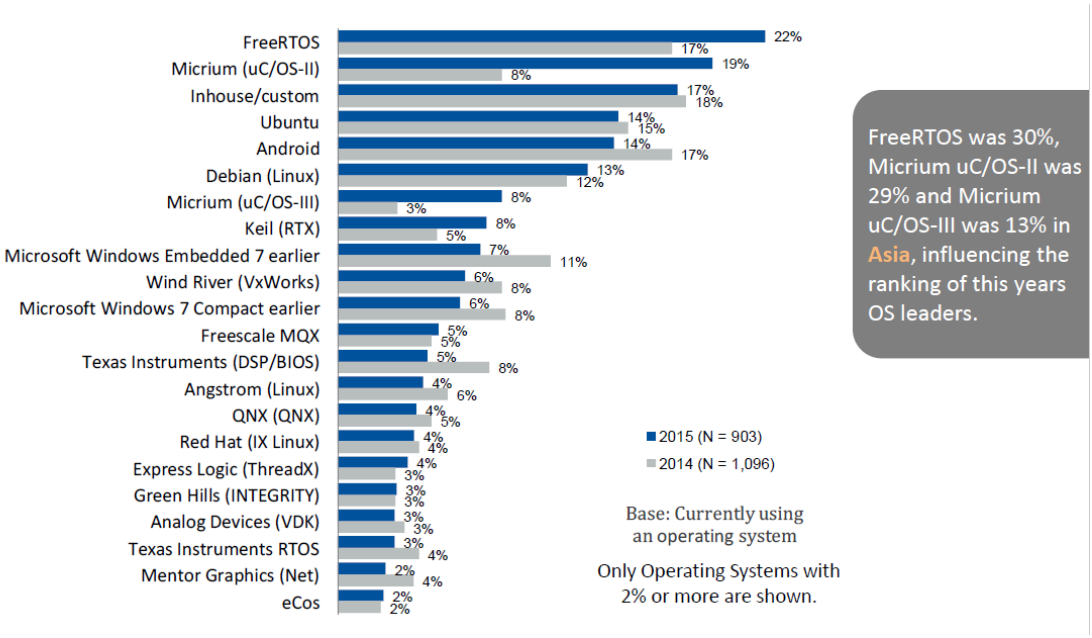
Már az első verzió megjelenésekor rendelkezésre álltak különböző linux disztribúciók portjai, melyekkel az eszköz asztali számítógépként használható volt. Népszerűségének köszönhetően a második verziótól kezdve már a Windows 10 IoT Core is támogatja a platformot.

2.2. A választás szempontjai

Az operációs rendszerek kiválasztásánál elsődleges szempont a hardverek támogatottsága volt, de ezen kívül még az oktatási célra való elérhetőséget is szem előtt tartottam.

2.2.1. STM32F4 Discovery

Az UBM Tech minden évben készít egy kutatást a beágyazott rendszereket piacán, melyben többek között a használt beágyazott operációs rendszerekkel kapcsolatban is publikál adatokat. A statisztika alapján a két leggyakrabban használt operációs rendszer a FreeRTOS és a uC/OS-II. A uC/OS új verziója, a uC/OS-III a listán hátrébb kapott helyet, de még így is befért a tíz vezető rendszer közé. Mindekettő operációs rendszer népszerűsége nőtt a 2014-es évhez képest.



2.4. ábra. *Az UBM Tech által 2015-ben publikált beágyazott operációs rendszer használati statisztika.*

Az STM32F4 Discovery kártyához elérhető szoftvercsomag tartalmazza a FreeRTOS rendszert, ami jelzi a rendszer támogatottságának mértékét. A FreeRTOS hivatalos oldalán megvásárolhatóak a rendszer használatát bemutató könyvek, illetve online elérhető leírások, amik szintén segítik a rendszer megismerését. Ezáltal az első megvizsgált rendszernek a FreeRTOS-t választottam.

A Micrium uC/OS rendszerei oktatási célra ingyenesen elérhetőek, beleértve a rendszer dokumentációit is. A választott másik rendszer a uC/OS-III.

2.2.2. Raspberry Pi 3

A Raspberry Pi megjelenése óta támogatja különböző linux disztribúciók futtatását az eszközön, és a fejlesztőkártya második verziója óta a Windows 10 IoT Core is telepíthető rá. Az asztali alkalmazásfejlesztők körében mind a linux, mind a Windows elterjedten használt operációs rendszer, ezért a Raspberry Pi 3-on ezt a két rendszert vizsgálom meg.

2.3. FreeRTOS

2.3.1. Bevezetés

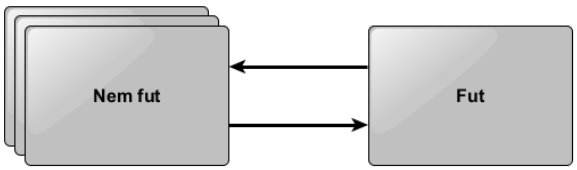
A FreeRTOS a Real Time Engineers Ltd. által fejlesztett valós idejű operációs rendszer. A fejlesztők céljai között volt a rendszer erőforrásigényének minimalizálása, hogy a legkisebb beágyazott rendszereken is futtatható legyen. Ebből adódóan csak az alap funkciók vannak megvalósítva, mint ütemezés, taszkok közötti kommunikáció lehetősége, memóriamenedzsment, de nincs beépített támogatás hálózati kommunikációra vagy bármiféle külső hardver használatára (ezeket vagy nekünk meg megírnunk, vagy harmadik féltől származó kódot kell használnunk).

A rendszer módosított GPLv2 licencet használ. A licencmódosítás lehetővé teszi GPL-től eltérő licenccel ellátott modulok használatát, amennyiben azok a FreeRTOS-sal kizárólag a FreeRTOS API-n keresztül kommunikálnak.

2.3.2. Taszkok

A FreeRTOS nem korlátozza a létrehozható taszkok és prioritások számát, amíg a rendelkezésre álló memória lehetővé teszi azok futtatását. A rendszer lehetőséget biztosít ciklikus és nem ciklikus taszkok futtatására egyaránt.

A beágyazott rendszerek döntő része egymagos processzorokat használ, amiből az következik, hogy egyszerre csak egy taszk futhat. A taszkok eszerint két nagy csoportba oszthatóak: éppen futó taszk (**Fut** állapot), illetve az összes többi (**Nem fut** állapot).



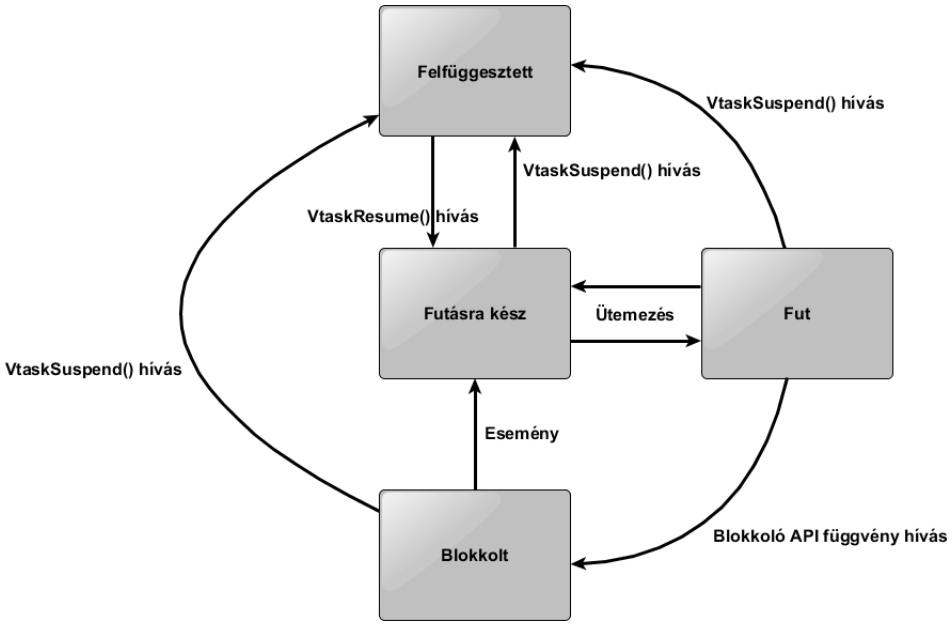
2.5. ábra. *Taszk lehetséges állapotai a FreeRTOS rendszerben (egyszerűsített).*

Annak, hogy egy taszk éppen miért nem fut, több oka lehet. Ez alapján a **Nem fut** állapot több állapotra feloszthető. Ha egy taszk képes lenne futni, de például egy nagyobb prioritású taszk birtokolja a processzort, akkor a taszk állapota **Futásra kész**. Ha a taszk valamilyen eseményre vár (időzítés, másik taszk szinkronizáló jele), akkor a taszk **Blokkolt** állapotban van. Az operációs rendszer lehetőséget ad arra, hogy a taszkokat függvényhívással **Felfüggesztett** állapotba kényszerítsük. Ekkor egy másik függvényhívással tudjuk visszahozni az ütemezendő feladatok sorába a taszkot.

A taszkok önként lemondhatnak a futásról (időzítés, szinkronizáció, taskYIELD() függvény hívása), viszont futó állapotba csak az ütemező helyezheti.

Idle taszk

A processzor folyamatosan utasításokat hajt végre működése közben (eltekintve a különböző energiatakarékos üzemmódoktól), ezért legalább egy taszknak mindig Futásra kész



2.6. ábra. *Taszk lehetséges állapotai a FreeRTOS rendszerben.*

állapotban kell lennie. Hogy ez biztosítva legyen, az ütemező indulásakor automatikusan létrejön egy ciklikus **Idle taszk**.

Az Idle taszk a legkisebb prioritással rendelkezik, így biztosítva, hogy elhagyja a Fut állapotot, amint egy magasabb prioritású taszk Futásra kész állapotba kerül.

Taszk törlése esetén az Idle taszk végzi el a különböző erőforrások felszabadítását. Mivel a FreeRTOS nem biztosít védelmet egy taszk *kiéheztetésével* szemben ezért fontos, hogy az alkalmazás tervezésekor biztosítsunk olyan időszeletet, amikor másik, nagyobb prioritású taszk nem fut.

Idle hook függvény

Előfordulhat, hogy az alkalmazásunkban olyan funkciót szeretnénk megvalósítani, amelyet az Idle taszk minden egyes iterációjára le kell futtatni (például teljesítménymérés érdekében). Ezt a célt szolgálja az **Idle hook** függvény, ami az Idle taszk minden lefutásakor meghívódik.

Az Idle hook általános felhasználása:

- Alacsony prioritású háttérfolyamat, vagy folyamatos feldolgozás,
- A szabad processzoridő mérése (teljesítménymérés),
- Processzor alacsony fogyasztású üzemmódba váltása.

Korlátozások az Idle hook függvénnyel kapcsolatban

1. Az Idle hook függvény nem hívhat blokkoló vagy felfüggesztést indító függvényeket[lábjegyzet->A hook függvény blokkolása esetén felléphet az az eset, hogy nincs Futásra kész állapotban levő taszk],

2. Ha az alkalmazás valahol töröl egy taszkot, akkor az Idle hook függvénynek elfogadható időn belül vissza kell térnie. [lábjegyzet->Az Idle taszk felelős a kernel erőforrások felszabadításáért, ha egy taszk törlésre kerül. Ha az Idle taszk bentragad az Idle hook-ban, akkor ez a tisztítás nem tud bekövetkezni.]

2.3.3. Ütemező

A FreeRTOS által használt ütemezési mechanizmust Fix Prioritásos Preemptív Ütemezésnek hívjuk[lábjegyzet->A FreeRTOS kooperatív ütemezést is támogat, viszont a valós idejű futás eléréséhez a preemptív ütemezés szükséges, ezért a továbbiakban csak a preemptív ütemezéssel foglalkozunk.]. *Fix prioritásos*, mivel a rendszer magától nem változtatja a prioritásokat, *preemptív*, mert egy taszk Futásra kész állapotba lépésekor preemtálja az éppen futó taszkot, ha a futó taszk prioritása alacsonyabb.

[ToDo -> Itt átfogalmazni kicsit a dolgokat. Elég kusza a gondolatmenet.]

A taszkok lehetnek Blokkolt állapotban, ahonnan egy esemény bekövetkezését követően automatikusan Futásra kész állapotba kerülnek.

Időbeli események azok, amik egy bizonyos időpillanatban következnek be (például egy késleltetési idő letelik). Az időbeli eseményeket kihasználva lehetőség nyílik periodikus futtatásra, vagy időtúllépés detektálására.

Szinkronizáló események azok, amikor egy taszk vagy megszakításkezelő rutin jelzést küld valamilyen kommunikációs struktúrán keresztül egy másik taszknak. Tipikusan aszinkron jelzésre használjuk, mint például adat érkezését egy periférián keresztül.

Taszkok prioritásának meghatározása

[ToDo -> Itt átfogalmazni kicsit a dolgokat. Elég kusza a gondolatmenet.]

Ökölszabályként alkalmazható, hogy a hard real-time funkciókat magasabb prioritással látjuk el, mint a soft real-time funkciókat.

Rate Monotonic Scheduling (Gyakoriság Monoton Ütemezés) az a gyakran alkalmazott prioritás-hozzárendelés, amikor minden taszkhoz egyedi prioritásszintet rendelünk a végrehajtási gyakoriság függvényében. A gyakrabban lefutó folyamatokhoz magasabb, a ritkábban lefutó folyamatokhoz alacsonyabb prioritást állítunk be.

2.3.4. Kommunikációs objektumok

[ToDo -> Átfogalmazni. Ne legyen nagyon redundáns az átlalános résszel.]

Az alkalmazások egymástól független taszkok konstrukciójából áll. Viszont ezeknek a taszkoknak ahhoz, hogy a feladatukat el tudják látni, gyakran kommunikálniuk kell egymással. A FreeRTOS három alap kommunikációs struktúrát kínál a felhasználónak:

- sor (*queue*),
- szemafor (*semaphore*),
  - bináris (*binary semaphore*),

  - számláló (*counting semaphore*),

- mutex (*mutex* [**MUT**ual **EX**cluison]).

Sorok (queues)

A sorok fix méretű adatból tudnak véges számú üzenetet tárolni. Ezek a jellemzők a sor létrehozásakor kerülnek meghatározásra. Alapértelmezetten FIFO-ként működnek (First In First Out), bár a FreeRTOS lehetővé teszi a sor elejére való írást is. A sorba való írás során másolat készül az eredeti változóról, és ez a másolat kerül tárolásra a sorban. Olvasáskor a paraméterként átadott memóriacímre kerül átmásolásra, ezért figyelni kell arra, hogy az átadott változó által elfoglalt memória legalább akkora legyen, mint a sor által tartalmazott adattípus mérete. Amennyiben a továbbítandó adattípus már nagynak tekinthető, akkor érdemes a memóriára mutató pointereket elhelyezni a sorban, ezzel csökkentve a RAM kihasználtságát (ekkor viszont különösen figyelni kell arra, hogy a kijelölt memóriaterület tolajdonosa egyértelmű legyen, vagyis ne történjen különböző taszkokból módosítás egyidőben, illetve biztosítani kell a memóriaterüet érvényességét). A FreeRTOS API kétféle függvényt használ olvasásra:

- Az egyik automatikusan eltávolítja a kiolvasott elemet a sorból (**xQueueReceive()**),

- A másik kiolvassa a soronkövetkező elemet, de azt nem távolítja el a sorból(**xQueuePeek()**).

A FreeRTOS-ban minden kommunikációs struktúra a sor valamilyen speciális megvalósítása.

A sorok egy taszkhoz sem tartoznak, így egy sorba akár több taszk is írhat, illetve olvashat egy alkalmazáson belül.

Olvasás sorból

Sorból való olvasás során az olvasó függvény paramétereként megadhatunk egy várakozási időtartamot. A taszk maximálisan ennyi ideig várakozik Blokkolt állapotban új adat érkezésére, amennyiben a sor üres. Ha ezen időtartam alatt nem érkezik adat, akkor a függvény visszatér, és a visszatérési értékével jelzi az olvasás sikertelenségét. A Blokkolt állapotból a taszk Futásra kész állapotba kerül, ha:

- Adat érkezik a sorba a megadott időtartamon belül,

- Nem érkezik adat a sorba, de a megadott várakozási idő lejárt.

Egy sorból egyszerre több taszk is kezdeményezhet olvasást, ezért előfordulhat az az eset, hogy több taszk is Blokkolt állapotban várja az adat érkezését. Ebben az esetben csak egy taszk kerülhet Futásra kész állapotba az érkezésének hatására. A rendszer a legnagyobb prioritású taszkot választja, vagy ha a várakozó taszkok azonos prioritásúak, akkor a legrégebben várakozót helyezi a Futásra kész állapotba.

Írás sorba

Az olvasáshoz hasonlóan a sorba való írás során is megadható egy várakozási idő. Ha a sorban nincs üres hely az adat írásához, akkor a taszk Blokkolt állapotba kerül, amelyből Futásra kész állapotba lép, ha:

- Megüresedik egy tároló a sorban,
- Letelik a maximális várakozási idő.

Egy sorba több taszk is kezdeményezhet írást egyidőben. Ekkor ha több taszk is Blokkolt állapotba kerül, akkor hely felszabadulásakor a legnagyobb prioritású taszkoz választja a rendszer, azonos prioritású taszkok esetén a legrégebben várakozót helyezi Futásra kész állapotba.

Szemaforok

Két szemafor típust különböztetünk meg:

- Bináris szemafor, amikor a szemafor két értéket vehet fel,
- Számláló szemafor, mikor a szemafor több állapotot is felvehet.

A szemaforon két művelet értelmezett:

- A szemafor jelzése (elterjedt elnevezések: give, signal, post, V() művelet[lábjegyzet-> Dijkstra holland, verhogen]), amikor a szemafor értéke növelésre kerül.
- A szemafor elvétele (elterjedt elnevezések: take, wait, pend, P() művelet [lábjegyzet-> proberen]), amikor a szemafor először tesztelésre kerül, hogy tartalmaz-e elemet, ha igen, akkor az értékét csökkentjük, ha nem, akkor várakozunk addig, amíg valamelyik másik taszk elérhetővé nem teszi azt.

A FreeRTOS rendszerben implementált szemaforok paraméterei között megadható a maximális tick szám, ameddig várakozhat a szemaforra az adott taszk. Ezen maximális időtartam megadása az esetek nagy részében megoldást jelent a holtponti helyzetekre.

A szemafor látszólag egyszerűen helyettesíthető egy egyszerű változó (boolean vagy előjel nélküli egész) használatával, viszont a beépített szemafor struktúra atomi műveletként kerül kezelésre (így nem következik be ütemezés a szemafor tesztelése és állítása közben), illetve a rendszer automatikusan tudja kezelni a várakozó folyamatok állapotok közti mozgatását.

Bináris szemafor

Bináris szemafor esetén a szemafor két értéket vehet fel. Felfogható úgy is, mint egy egy adat tárolására (egy elem hosszú) sor, melynek nem vizsgáljuk a tartalmazott értékét, csak azt, hogy éppen tartalmaz-e adatot vagy sem.

Leggyakoribb felhasználása a taszkok szinkronizálása. Ekkor az egyik taszk a futásának egy adott pontján várakozik egy másik taszk jelzésére. Ezzel a módszerrel megvalósítható

a megszakítások taszkokban történő kezelése, ezzel is minimalizálva a megszakítási rutin hosszát.

Bináris szemafor használatakor különös figyelmet kell fordítani arra, hogy ha a szemafor egy adott taszkban gyakrabban kerül jelzésre, mint ahogy feldolgozzuk, akkor jelzések veszhetnek el (amíg az egyik jelzés várakozik, addig az utána következő eseményeknek nincs lehetőségük várakozó állapotba kerülni).

Számláló szemafor

A számláló típusú szemafor minden jelzéskor növeli az értékét. Ekkor (amíg el nem éri a maximális értékét) nem kerül Blokkolt állapotba a jelző taszk. A számláló szemafor felfogható úgy, mint egy egynél több adat tárolására képes sor, melynek nem vizsgáljuk az értékét, csak azt, hogy éppen tartalmaz-e még adatot vagy sem.

Két felhasználása elterjedt a számláló szemaforoknak:

- Események számlálása: ekkor minden esemény hatására növeljük a szemafor értékét (új elemet helyezünk a sorba). A szemafor aktuális értéke a beérkezett és a feldolgozott események különbsége. A számlálásra használt szemafor inicializálási értéke nulla.

- Erőforrás menedzsment: ekkor a szemafor értéke a rendelkezésre álló erőforrások számát mutatja. Mikor az operációs rendszertől az erőforrást igényeljük, akkor a szemafor értékét csökkentjük, mikor felszabadítjuk a birtolkolt erőforrást, akkor a szemafor értékét növeljük. Ha a szemafor értéke nulla, akkor nincs rendelkezésre álló erőforrás. Erőforrások kezelésére használt számláló szemafor esetén az inicializálási érték az elérhető erőforrások száma.

Mutexek

Taszkok vagy taszkok és megszakítási rutinok között megosztott erőforrás kezelésekor a Mutex (kölcsönös kizárás) használata indokolt. Mikor egy taszk vagy megszakítás hozzáférést indít egy erőforráshoz, akkor a hozzá tartozó mutex-et elkéri. Ha az erőforrás szabad, akkor az igénylő taszk megkapja a kezelés jogát, és mindaddig megtartja, amíg be nem fejezi az erőforrással való munkát. A mutex-et a lehető legkorábban (az erőforrással való munka befejeztével) fel kell szabadítani, ezzel is csökkentve az esetleges holtpont kialakulásának veszélyét.

Látható, hogy a mutex nagyon hasonlít a bináris szemaforhoz. A különbség abból adódik, hogy mivel a bináris szemafort leggyakrabban szinkronizációra használjuk, ezért azt nem kell felszabadítani: a jelző taszk vagy megszakítás jelzést ad a szemforon keresztül a feldolgozó taszknak. A feldolgozó taszk elveszi a szemafort, de a feldolgozás befejeztével a szemafort nem adja vissza.

A felhasználásból adódó különbségek miatt a mutex védett a prioritás inverzió problémájával szemben, míg a bináris szemafor implementációjából hiányzik.[lábjegyzet-> a FreeRTOS prioritás öröklési mechanizmusa csak egyszerű implementációt tartalmaz, és feltételezi, hogy csak egy taszk csak egy mutex-et birtokol egy adott pillanatban.]

2.3.5. Megszakítás-kezelés

Beágyazott rendszereknél gyakran kell a környezettől származó eseményekre reagálni (például adat érkezése valamely kommunikációs interfészen). Az ilyen események kezelésekor a megszakítások alkalmazása gyakran elengedhetetlen.

Megszakítás használata esetén figyelni kell arra, hogy a megszakítási rutinokban csak *FromISR*-re végződő API függvényeket hívhatunk. Ellenkező esetben nem várt működés következhet be (blokkoljuk a megszakítási rutint, ami az alkalmazás fagyásához vezethet; kontextus-váltást okozunk, amiből nem térünk vissza, így a megszakítási rutinból sosem lépünk ki, stb.).

A FreeRTOS ütemezője a (STM32-re épülő rendszerekben) a SysTick interruptot használja az ütemező periodikus futtatásához. A megszakítási rutin futása közben emiatt nem történik ütemezés. Amennyiben valamely magasabb prioritású taszkunk a megszakítás hatására Futásra kész állapotba kerül, akkor vagy a következő ütemezéskor kapja meg a processzort, vagy explicit függvényhívással kell kérni az operációs rendszer az ütemező futtatására.

Az alacsonyabb prioritású megszakítások szintén nem tudnak érvényre jutni, így azok bekövetkezéséről nem kapunk értesítést (az első beérkező, alacsonyabb prioritású megszakítás jelző bitje bebillen az esemény hatására, de amennyiben több is érkezik a magasabb prioritású megszakítási rutin futása alatt, úgy azok elvesznek). Az említett problémák végett a megszakítási rutint a lehető legrövidebb idő alatt be kell fejezni.

Késleltetett megszakítás-kezelés

A megszakítási rutint a lehető legrövidebb idő alatt el kell hagyni, emiatt célszerű a kevésbé fontos műveleteket egy kezelő taszkban megvalósítani. A FreeRTOS a szemaforokon keresztül biztosít lehetőséget a megszakítás és taszk szinkronizációjára.

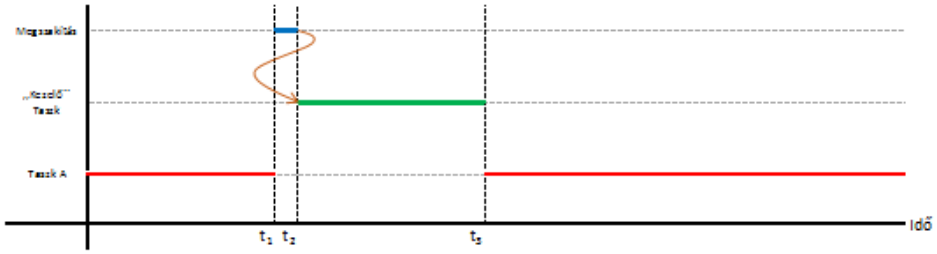
A megszakítás hatására a megszakítási rutinban csak szükséges lépéseket végezzük el (például eseményjelző bitek törlése), majd egy szemaforon keresztül jelzi a feldolgozó taszknak az esemény bekövetkeztét. Ha a feldolgozás időkritikus, akkor a feldolgozó szálhoz rendelt magas prioritással biztosítható, hogy az éppen futó taszkot preemtálja. Ekkor a megszakítási rutin végén az ütemező meghívásával a visszatérés utána azonnal feldolgozásra kerül az esemény.

Az eseményt kezelő taszk blokkoló *take* utasítással várakozik Blokkolt állapoban a szemafor érkezésére. A jelzés hatására Futásra kész állapotba kerül, ahonnan az ütemező (az éppen futó taszkot preemtálva) Futó állapotba mozgatja. Az esemény feldolgozását követően újra meghívja a blokkoló *take* utasítást, így újból Blokkolt állapotba kerül az újabb esemény bekövetkezéséig.

Todo: Példa

Megszakítások egymásba ágyazása

Az STM32 mikrokontrollerek lehetővé teszik prioritások hozzárendelését a megszakításokhoz. Ezáltal létrejöhet olyan állapot, amikor egy magasabb prioritású megszakítás érkezik



2.7. ábra. *Megszakítások késleletett feldolgozásának szemléltetése.*

egy alacsonyabb szintű megszakítási rutin futása közben. Ekkor a magasabb prioritású megszakítás késleltetés nélkül érvényre jut, és a magasabb prioritású megszakítási rutin befejeztével az alacsony prioritású folytatódik.

A FreeRTOS rendszer konfigurációs fájlában (*FreeRTOSConfig.h*) beállítható azon legmagasabb prioritás, amihez a FreeRTOS hozzáférhet. Ezáltal a megszkaításoknak két csoportja adódik:

- A FreeRTOS által kezelt megszakítások,
- A FreeRTOS-tól teljesen független megszakítások.

A FreeRTOS által kezelt prioritások kritikus szakaszba lépéskor letiltásra kerülnek, így azok nem szakíthatják meg az atomi utasításként futtatandó kódrészletet. Viszont a megszakítások ezen csoportja használhatja a *FromISR*-re végződő FreeRTOS API függvényeket.

A FreeRTOS-tól független megszakítások kezelése teljes mértékben a fejlesztő feladata. Az operációs rendszer nem tudja megakadályozni a futásukat, így a kritikus szakaszban is képesek futni. A kiemelkedően időkritikus kódrészleteket célszerű ezekben a megszakítási turinokban megvalósítani (például motorvezérlés). A FreeRTOS-tól független megszakítási rutinokban nem szabad API függvényeket használni!

2.3.6. Erőforrás-kezelés

Multitaszk rendszerek esetén fennáll a lehetősége, hogy egy taszk kikerül a Futó állapotból még mielőtt befejezné egy erőforrással a műveleteket. Ha az erőforrást egy másik taszk is használni akarja, akkor inkonzisztencia léphet fel. Tipikus megjelenései a problémának:

- Periféria elérése,
- Egy közös adat olvasása, módosítása, majd visszaírása,[lábjegyzet-> magas szintű nyelv esetén (pl C) ez látszólag lehet egy utasítás, viszont a fordító által előállított gépi kód több utasításból áll]
- Változó nem atomi elérése (például több tagú struktúra értékeinek megváltoztatása),
- Nem reentráns függvények, [lábjegyzet-> reentráns függvény]

Az adat inkonzisztencia elkerüléséhez hasnzálhatunk mutex-et. Amikor egy taszk megkapja egy erőforrás kezelésének jogát, akkor más taszk nem férhet hozzá, egészen addig,

amig a birtokló taszk be nem fejezte az erőforrással a feladatát, és az erőforrás felszabadítását mutex-en keresztül nem jelezte.

Kritikus szakasz

[ToDo -> Itt átfogalmazni kicsit a dolgokat. Elég kusza a gondolatmenet.]

A közös erőforrások használatakor gyakran szükség van egy adott műveletsor atomivá tételére, azaz arra, hogy a kijelölt műveletek futását semmi ne szakíthassa meg, látszólag egy utasításként fussanak le.

A FreeRTOS támogatja a kritikus szakaszok használatát a *taskENTER_CRITICAL()* és *taskEXIT_CRITICAL()* makrók használatával.

A kritikus szakaszokat minden probléma nélkül egymásba lehet ágyazni, mivel a rendszerkernel nyílvántartja, hogy milyen mélyen van az alkalmazás a kritikus szakaszokban. A rendszer csak akkor hagyja el a kritikus szakaszt, ha a számláló nullára csökken, vagyis ha minden *taskENTER_CRITICAL()* híváshoz tartozik egy *taskEXIT_CRITICAL()* is.

A kritikus szakaszt a lehető leggyorsabban el kell hagyni, különben a beérkező megszakítások válaszideje nagy mértékben megnőhet.

Ütemező felfüggesztése

A kritikus szakasz megvalósításának egy kevésbé drasztikus módja az ütemező letiltása. Ekkor a kódrészlet védett a más taszkok általi preemtálástól, viszont a megszakítások nem kerülnek letiltásra. Hátránya, hogy az ütemező elindítása hosszabb időt vehet igénybe.

Gatekeeper taszk

A gatekeeper taszk alkalmazása a kölcsönös kizárás egy olyan megvalósítása, mely működésénél fogva védett a prioritás inverzió és a holtpont kialakulásával szemben.

A gatekeeper taszk egyedüli birtokosa egy erőforrásnak, így csak a taszk tudja közvetlenül elérni az erőforrást, a többi taszk közvetetten, a gatekeeper taszk szolgáltatásain keresztül tudja használni az erőforrást.

Amikor egy taszk használni akarja az erőforrást, akkor üzenetet küld a gatekeeper taszknak (általában sor használatával). Mivel egyedül a gatekeeper taszk jogosult elérni az erőforrást, és nincs szükség explicit mutex használatára.

A gatekeeper taszk Blokkolt állapotban vár, amíg nem érkezik üzenet a sorba. Az üzenet beérkezése után elvégzi a megfelelő műveleteket az erőforráson, majd ha kiürült a sor, akkor ismét Blokkolt állapotba kerül.

A megszakítások probléma nélkül tudják használni a gatekeeper taszkok szolgáltatásait, mivel a sorba való írás támogatott megszakítási rutinból is.

2.3.7. Memória-kezelés

Beágyazott alkalmazások fejlesztése során is szükség van dinamikus memóriafoglalásra. Az asztali alkalmazásoknál megszokott *malloc()* és *calloc()* függvények több szempontból sem felelnek meg mikrokontrolleres alkalmazásokban:

- Kisebb rendszerekben nem biztos, hogy elérhető,
- Az implementációjuk sok helyet foglalhat,
- Nem determinisztikus a lefutásuk; a végrehajtási idő különbözhet különböző híváskor,
- Memóriatöredezettség léphet fel.

Minden taszkhoz tartozik egy TCB (*Task Control Block*) és egy stack. A TCB struktúra a következő elemeket tartalmazza (a teljesség igénye nélkül):

2.1. táblázat. *A FreeRTOS TCB-jének főbb változói.*

Változó	Jelentés
pxTopOfStack	Az stack utolsó elemére mutató pointer
xGenericListItem	A FreeRTOS a TCB ezen elemét helyezi az adott állapothoz tartozó listába (nem magát a TCB-t)
xEventListItem	A FreeRTOS a TCB ezen elemét helyezi az adott eseményhez tartozó listába (nem magát a TCB-t)
uxPriority	A taszk prioritása
pxStack	A stack kezdetére mutató pointer
pcTaskName	A taszk neve. Kizárólag debug célokra
pxEndOfStack	A stack végére mutató pointer a stack túlcsordulásának detektálására
uxBasePriority	Az utojára taszkhoz rendelt prioritás. Mutex használata esetén a prioritás öröklés során megnövelt prioritás visszaállítására
ulRunTimeCounter	A taszk **Fut** állapotban töltött idejét tárolja futási statisztika készítéséhez

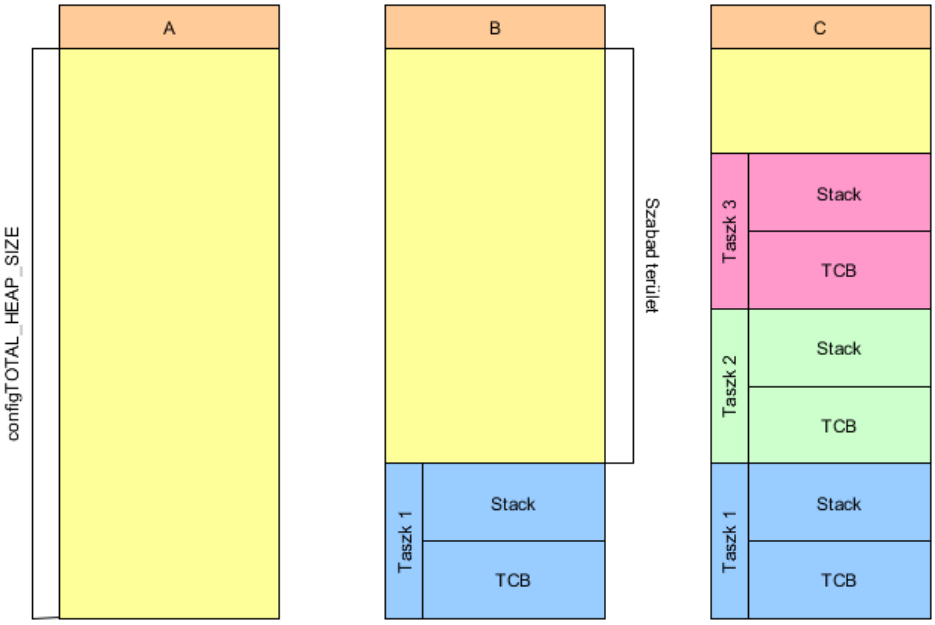
Az egyes alkalmazások különböznek memória-allokációs igényükben és az erőírt időzítési korlátokban, ezért nincs olyan memória-allokációs séma, amely minden alkalmazásban megállná a helyét. A FreeRTOS több allokációs algoritmust is a fejlesztő rendelekezésére bocsát, amiből az alkalmazásnak megfelelő kiválasztásával lehet elérni a megfelelő működést.

Heap_1.c

Kisebb beágyazott alkalmazásoknál gyakran még az ütemező indulása előtt létrehozunk minden taszkot és kommunikációs objektumot. Ilyenkor elég a memóriát lefoglalni az alkalmazás indulása előtt, és a futás alatt minden memória lefoglalva marad. Ez azt is jelenti, hogy nem kell komplex algoritmusokat megvalósítani a determinisztikusság biztosítására és a Memóriatöredezettség ellkerülésére, hanem elég a kódméretet és az egyszerűséget szem előtt tartani.

Ezt az implementációt tartalmazza a *heap_1.h*. A fájl a *pvPortMalloc()* egyszerű megvalósítását tartalmazza, azonban a *pvPortFree()* nincs implementálva. A *heap_1.c* nem fenyegeti a rendszer determinisztikusságát.

A *pvPortMalloc()* függvény a FreeRTOS heap-jét ossza fel kisebb területekre, majd ezeket rendeli hozzá az egyes taszkokhoz. A heap teljes méretét a *configTOTAL_HEAP_SIZE* konfigurációs érték határozza meg a *FreeRTOSConfig.h* fájlban. Nagy méretű tömböt definiálva már a memórifoglalás előtt látszólag sok memóriát fog felhasználni az alkalmazás, mivel a FreeRTOS ezt induláskor lefoglalja.



2.8. ábra. *A heap_1.c implementációjának működése.*

Heap_2.c

A *heap_2.c* szintén a *configTOTAL_HEAP_SIZE* konfigurációs értéket használja, viszont a *pvPortMalloc()* mellett már implementálva van a *pvPortFree()* is. A memóriafoglalás során a legjobban illeszkedő területből oszt ki a taszk számára memóriát.

A legjobban illeszkedő (best fit) algoritmus biztosítja, hogy a memóriakérés a hozzá méretben legközelebb eső, elgendő nagyságú blokkból legyen kiszolgálva.

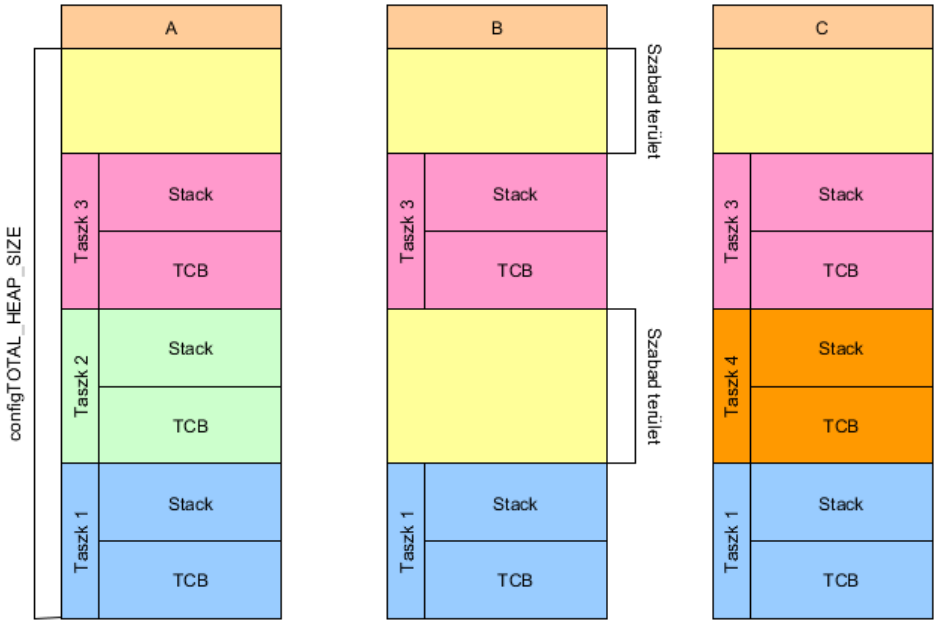
A megvalósítás nem egyesíti a szomszédos szabad területeket egy nagyobb egységes blokkba, így töredezettség léphet fel. Ez nem okoz gondot, ha a lefoglalt és felszabadított memória mérete nem változik.

A *heap_2.c* fájl használata javasolt, ha az alkalmazás ismételve létrehoz és töröl taszkokat, és a taszkokhoz tartozó stack mérete nem változik.

A *heap_2.c* működése nem determinisztikus, de hatékonyabb, mint a *standard library* implementációi.

Heap_3.c

A *heap_3.c* a *standard library* függvényeit használja, de a függvények alatt felfüggeszti az ütemező működését, ezzel elérve, hogy a memória-kezelés thread-safe legyen.



2.9. ábra. *A heap_2.c implementációjának működése.*

A heap méretét nem befolyásolja a *configTOTAL_HEAP_SIZE* érték, ehelyett a linker beállításai határozza meg.

2.4. µC/OS-III

2.5. Linux

2.6. Windows 10

3. fejezet

Teljesítménymérő metrikák

Asztali alkalmazás fejlesztésekor a rendelkezésre álló teljesítmény és tárhely ma már nem számít akadálynak. Ha valamelyik erőforrás szűk keresztmetszetté válik, akkor alkatrészcserével általában megszüntethető a probléma.

Nem ez a helyzet beágyazott rendszerek esetén. A rendszer központi egységének számítási kapacitása általában nem haladja meg nagy mértékben az elégséges szintet, így különösen figyelni kell a fejlesztés során, hogy az implementált kód hatékony legyen. A rendelkezésre álló memória sem tekinthető korlátlannak, és gyakran a bővítés is nehézkes, esetleg nem megoldható. Az eszköz fogyasztása is fontos szempont, amit a szoftver szintén nagy mértékben befolyásolhat.

Az operációs rendszer választása összetett folyamattá is bonyolódhat, mert mérlegelni kell az alkalmazásunk igényeit, az operációs rendszer támogatottságát (támogatott mikrokontrollerek, fórumok, gyártói támogatás), a becsülhető fejlesztési időt és az ezzel járó költségeket, illetve fizetős operációs rendszer esetén a rendszer árát.

A választást az sem segíti előre, hogy nincs egyértelmű módszer az operációs rendszerek értékelésére. [lábjegyzet-> Bár a Német Szabványügyi Intézet (Deutsche Institut für Normung - DIN) az 1980-as évek végén hozott létre szabványt a folyamatirányító számítógépes rendszerek teljesítménymutatóinak mérésére (DIN 19242 szabvány-sorozat), a valós idejű operációs rendszerek értékelésére ez nem jelent megoldást.]

Az alkalmazott mérési folyamatnak több szempontnak is eleget kell tennie, hogy az eredmény használható legyen. Egy mérés során több forrásból is eredhet hiba, melyek mértékét szeretnénk a lehető legkisebb szintre csökkenteni. A kulcsfontosságú szempontok az alábbiak:

- Megismételhetőség: egy mérést meg kell tudnunk ismételni. Ehhez szükséges a pontos mérési összeállítás, a mérés körülményei, a használt eszközök és szoftverek.

- Véletlenszerűség: a mérés során nem független események következhetik egymást, amik a mérés eredményét befolyásolják. Ezeket csak ritkán lehet teljes mértékben kiküszöbölni, ezért törekedni kell a mérési folyamatok véletlenszerű futtatására (például méréssorozat esetén az egyes folyamatok ne mindig ugyan abban a sorrendben kövessék egymást).

- Vezérlés: a mérés során a vezérelhető paramétereket (melyek a mérést befolyásolhatják) lehetőségeinkhez mérten kézben kell tartani.
- Szemléletesség: a mérés eredményének reprezentatívnak kell lennie. Számértékek esetén két mérés eredményét össze kell tudnunk hasonlítani és tudnunk kell relációt vonni a két érték közé.

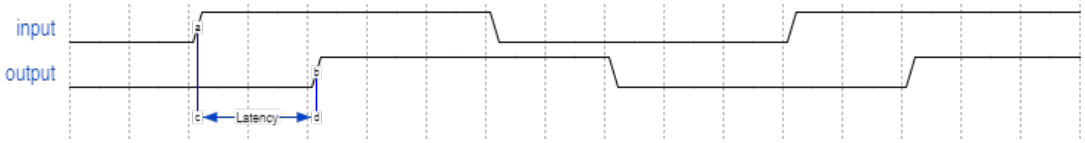
A továbbiakban különböző forrásokból vett szempontokat vizsgálok meg, majd azok alapján állítom fel a dolgozat során megfigyelt tulajdonságok listáját.

3.0.1. Memóriaigény

A mikrokontrollerek területén a memória mérete korlátozott (ROM és RAM egyaránt), ezért fontos, hogy a használt rendszer minél kisebb lenyomattal rendelkezzen.

3.0.2. Késleltetés

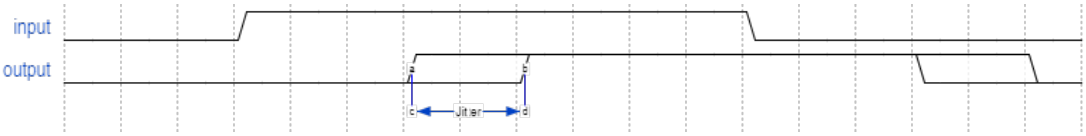
A rendszer késleltetése az az idő, ami egy esemény beérkezésétől a rendszer válaszáig eltelik. Ezt okozhatja a mikrovezérlő megszakítási mechanizmusához szükséges műveletek sora, az operációs rendszer ütemezőjének overhead-je, de a közben végrehajtandó feladat is nagy mértékben befolyásolja a nagyságát.



3.1. ábra. *Az operációs rendszer késleltetésének szemléltetése.*

3.0.3. Jitter

A jitter egy folyamat vizsgálata során a többszöri bekövetkezés után mért késleltetésekből határozható meg.



3.2. ábra. *A késleltetés jitterének szemléltetése.*

3.0.4. Rhealstone

1989-ben a Dr. Dobbs Journal cikkeként jelent meg egy javaslat, ami a valós idejű rendszerek objektív értékelését célozta meg. Rabindra P. Kar, az Intel Systems Group senior mérnöke ismertette a módszer előnyeit és szempontjait, melynek a Rhealstone nevet adta.[lábjegyzet-> név eredete]

A cikk megjelenésekor már léteztek teljesítménymérő metódusok (Whetstone, Dhrystone), de ezek a fordító által generált kódot, illetve a hardware-t minősítették. A Rhealstone metrika célja, hogy a fejlesztőket segítse az alkalmazásukhoz leginkább megfelelő operációs rendszer kiválasztásában.

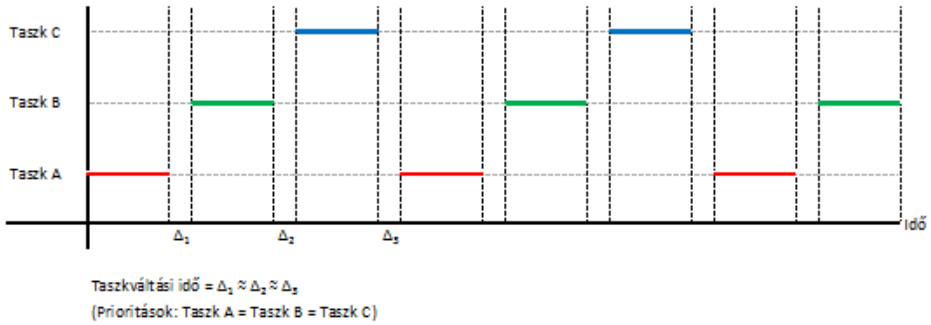
A Rhealstone hat kategóriában vizsgálja meg az operációs rendszer képességeit:

- Taszkváltási idő (Task switching time),
- Preemptálási idő (Preemption time),
- Megszakítás-késleltetési idő (Interrupt latency time),
- Szemafor-váltási idő (Semaphore shuffling time),
- Deadlock-feloldási idő (Deadlock breaking time) [lábjegyzet-> Nincs kialakult holtpont, igazából a prioritásinverzió jelenségét vizsgálja],
- Datagram-átviteli idő (Datagram throughput time) [lábjegyzet-> Bár időnek van feltüntetve, kB/sec-ben értendő. Minden kategóriának a reciprokát kell venni a mréés végén, így Rhealstone/sec értéket kapva. Ha az 1kB-hoz tartozó időt mérjük, akkor minden úgy történik, mint a többi kategóriánál.]

1990-ben megjelent egy második cikk is, amely amellett, hogy az egyes kategóriák példaprogramjait tartalmazta (iRMX operációs rendszerhez), pár kategória meghatározását megváltoztatták. Ezen változtatásokat az adott kategória részletezésekor ismertetem.

Taszkváltási idő

A taszváltási idő a két független, futásra kész, azonos prioritású taszkok váltásához szükséges átlagos időtartam.



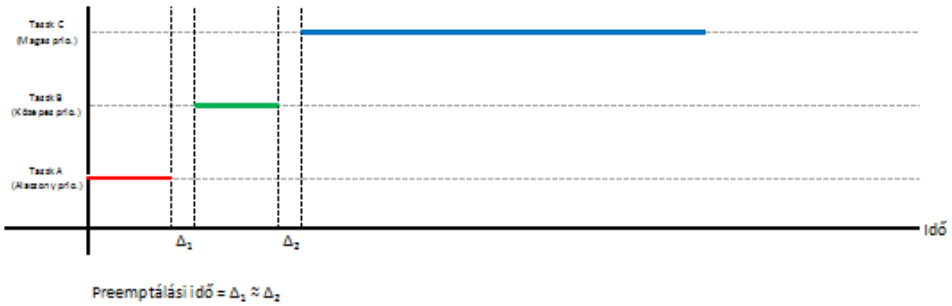
3.3. ábra. *A taszváltási idő szemléltetése.*

A taszkváltási idő alapvető jellemzője egy multitaszk rendszernek. A mérés a taszkokat nyilvántartó struktúrák hatékonyságáról ad képet. A taszkváltási időt a használt processzor architektúrája, utasításkészlete is befolyásolja.

A rendszerek a futtatható taszkokat általában valamilyen listában tárolják, így különböző számú taszkkal elvégezve a mérést más eredményt kaphatunk.

Preemptálási idő

A preemptálási idő egy magasabb prioritású taszk érvényre jutásához szükséges átlagos időtartam.

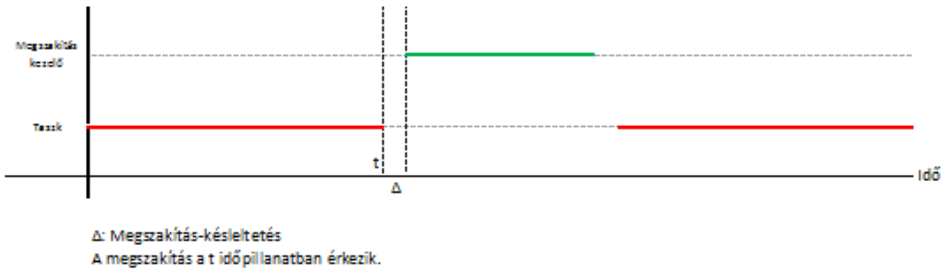


3.4. ábra. *A preemptálási idő szemléltetése.*

A preemptálási idő nagyban hasonlít a taszkváltási időhöz, azonban a járulékos utasítások miatt általában hosszabb időt jelent.

Megszakítás-késleltetési idő

A megszakítás-késleltetési idő egy esemény beérkezése és a megszakítás kezelő rutin első utasítása között eltelt átlagos időtartam.



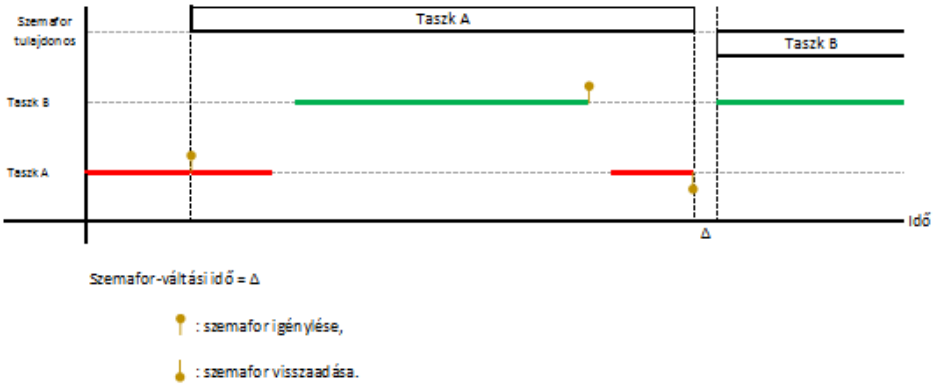
3.5. ábra. *A megszakítás-késleltetési idő szemléltetése.*

Szemafor-váltási idő

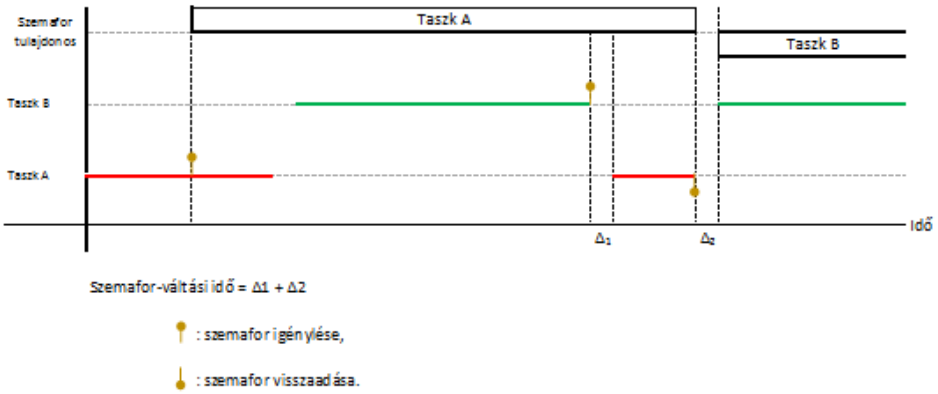
Az 1989-es cikk szerint szemafor-váltási idő az az átlagos időtartam, ami egy szemafor elengedése és egy, a szemaforra várakozó taszk elindulása között eltelik.

Ezt a meghatározást 1990-ben annyival módosították, hogy a szemafor-váltási idő egy már birtokolt szemafor kérése és a kérés teljesítése között eltelt időtartam, a birtokló taszk futási idejétől eltekintve.

A mérés célja az overhead meghatározása, mikor egy szemafor kölcsönös kizárást (mutex) valósít meg.



3.6. ábra. *A szemafor-váltási idő szemléltetése (1989-es meghatározás alapján).*



3.7. ábra. *A szemafor-váltási idő szemléltetése (1990-es meghatározás alapján).*

Deadlock-feloldási idő

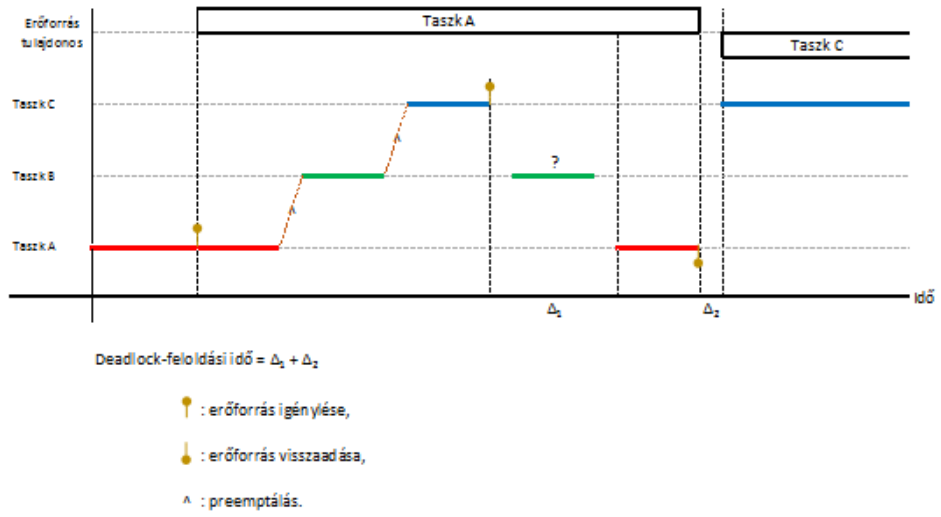
A deadlock-feloldási idő az az átlagos időtartam, ami egy olyan erőforrás eléréséhez szükséges, amit egy alacsonyabb prioritású taszk már birtokol.

Vagyis a deadlock-feloldási idő a birtoklási probléma feloldásához szükséges összesített idő egy alacsony és egy magas prioritású taszk között.

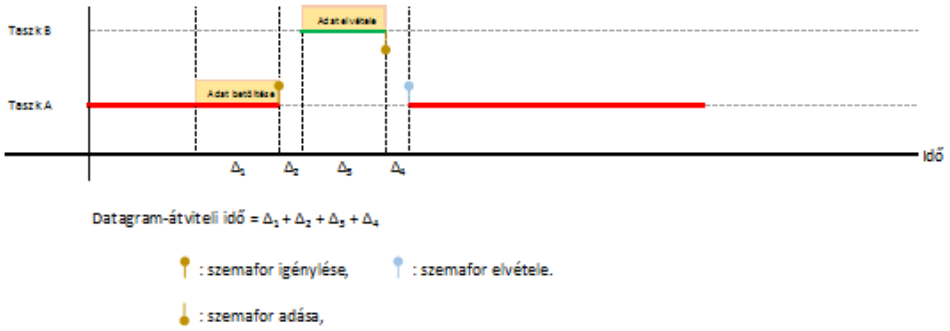
Datagram-átviteli idő

A datagram-átviteli idő a taszkok között elérhető adatsebesség az operációs rendszer objektumait kihasználva (vagyis nem megosztott memórián vagy pointeren keresztül). Az adatküldő taszknak kapnia kell értesítést az adat átvételéről.

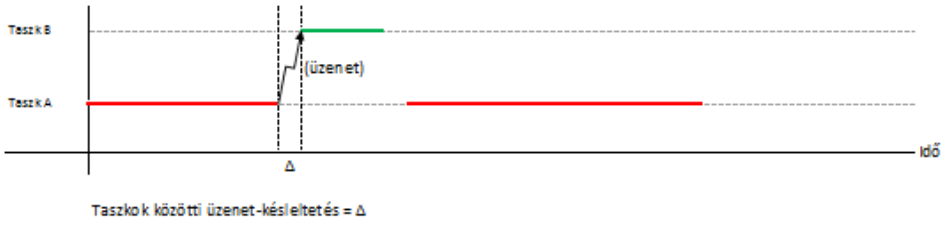
Az egy évvel később megjelent cikkben ezt a kategóriát is módosították kis mértékben. Egyrészt a megnevezést taszk közötti üzenet-késleltetésre változtatták, másrészt nem a maximális adatsebesség meghatározása a mérés célja, hanem az adattovábbítást végző objektum kezelésének és az operációs rendszer járulékos műveleteinek hatékonyságának megmérése.



3.8. ábra. *A deadlock-feloldási idő szemléltetése.*



3.9. ábra. *A datagram-átviteli idő szemléltetése.*



3.10. ábra. *A taszk közötti üzenet-késleltetési idő szemléltetése.*

Rhealstone jellemzők összegzése

Az elvégzett mérések várhatóan mikroszekundum és milliszekumdum nagyságrendű értékeket adnak vissza. Minden értéket másodpercre váltva, majd a reciprokát véve az értékek összegezhetőek egy reprezentatív értékké. Az átváltásnak köszönhetően a nagyobb érték jobb teljesítményt jelent, ami a teljesítménymutatók világában elvárt.

Objektív Rhealstone érték

Objektív értékelés esetén minden jellemző azonos súllyal szerepel a számolás során.
[Képlet]

Súlyozott Rhealstone érték

Az esetek döntő többségében a vizsgált feladatok nem azonos gyakorisággal szerepelnek egy alkalmazásban, sőt, előfordulhat, hogy valamely funkciót nem is használ az alkalmazás. Ekkor informatívabb eredményt kapunk, ha az egyes jellemzőkre kapott számértékeket különböző súllyal vesszük bele a végeredmény meghatározásába.

[Képlet]

3.0.5. Legrosszabb válaszidő

2001-ben a Nemzetközi Automatizálási Társaság (International Society of Automation - ISA) egy jelentésben fejtette ki azt az álláspontját, miszerint a késleltetés nem jellemzi a valós idejű rendszert, mert lehet, hogy a legtöbb esetben az előírt időn belül válaszol, de ritkán előfordulhatnak késleltetések vagy kihagyott események, amiket a mérés során nem lehet detektálni.

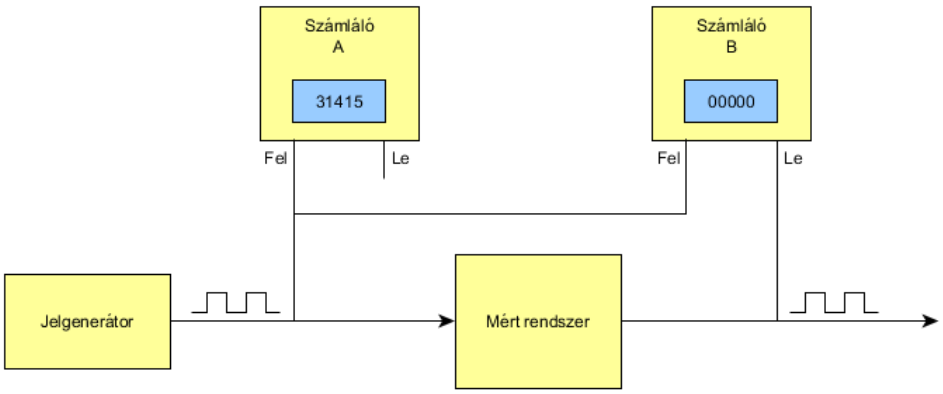
A Társaság egy olyan mérési összeállítást javasol a fejlesztőknek, ami egyszerűsége ellenére képes meghatározni a rendszer legrosszabb válaszidejét.

A méréshez szükséges eszközök:

- Mérendő rendszer (legalább egy be- és kimenettel),
- Jelgenerátor.
- Két darab digitális számláló.

Mérési elrendezés

A jelgenerátor kimenetét a mérendő rendszer bemenetére, illetve mindkét számláló *count up* bemenetére kötjük, míg a mérendő rendszer kimenetét a kimeneti számláló *count down* bemenetére kötjük.



3.11. ábra. *A legrosszabb válaszidő mérési összeállítása.*

Mérési elv

A mérés során azt a legkisebb frekvenciát keressük, amit a rendszer már nem tud követni, vagyis impulzusokat veszít. Ezáltal a kimenetén megjelenő impulzusok száma különbözni fog a bemenetére adott impulzusok számától. A kapott frekvencia a a legrosszabb válaszidő reciproka.

Mérés menete

1. A rendszeren futó program a bemenetére érkező jelet a kimenetére másolja. A mérés során digitális és analóg I/O láb is használható.
2. Mérési eszközök csatlakoztatása.
3. Alacsony frekvenciáról indulva növeljük a bemeneti jel frekvenciáját. Az A számláló folyamatosan számol felfele. Amíg a rendszer képes követni a bemenetet, addig a B számláló 1 és 0 között váltakozik. Amikor a rendszer már nem képes követni a bemenetet, akkor a B számláló elkezdt felfele számolni.
4. Csökkentjük a bemeneti frekvencia értékét egészen addig, amíg a rendszer újból képessé nem válik a bemenet követésére. A kapott frekvencia a legrosszabb válaszidő reciproka.

A mérést célszerű elvégezni különböző terhelés mellett. Ha valamelyik funkció használata közben (adattároló vezérlése, hálózati kommunikáció, stb.) a B számláló elindul, akkor az adott frekvencián a rendszer nem determinisztikus.

3.0.6. Vizsgált operációs rendszer jellemzők

A feladat megoldása során elsődlegesen az operációs rendszerek jellemzőit vizsgálom, ezért nem kerülnek külön tesztelésre az egyes hardverek előnyei. Az egyes rendszer-jellemzőket terheletlenül és terhelés alatt is megmérem.

Egy ipari alkalmazás szimulációját is megvalósítom, mely egy másik összehasonlítási alapot nyújt a dolgozathoz. Az alkalmazást felhasználom a terhelés alatti mérés megvalósításához is.

A dolgozat további részeiben a [referencia] fejezetben felsorolt összes jellemző meghatározására képes rendszert állítok össze, mellyel végrehajtom a méréseket. A meghatározandó jellemzők:

- Memóriaigény,
- Késleltetés,
- Jitter,
- Rhealstone értékek,
- Legrosszabb válaszidő.

4. fejezet

Rendszerterv

5. fejezet

Eredmények kiértékelése

6. fejezet

Konklúzió

Köszönetnyilvánítás

Ez nem kötelező, akár törölhető is. Ha a szerző szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelező, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

Ábrák jegyzéke

Táblázatok jegyzéke

Irodalomjegyzék

[1] James C. Candy. Decimation for sigma delta modulation. *IEEE Trans. on Communications*, 34(1):72–76, January 1986.

[2] Peter Kiss. *Adaptive Digital Compensation of Analog Circuit Imperfections for Cascaded Delta-Sigma Analog-to-Digital Converters.* PhD thesis, Technical University of Timişoara, Romania, April 2000.

[3] Wai L. Lee and Charles G. Sodini. A topology for higher order interpolative coders. In *Proc. of the IEEE International Symposium on Circuits and Systems*, pages 459–462, Philadelphia, PA, USA, May 4–7 1987.

[4] Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Kar. Diplomaterv portál (2011 február 26.). `http://diplomaterv.vik.bme.hu/`.

[5] Richard Schreier. *The Delta-Sigma Toolbox v5.2.* Oregon State University, January 2000. URL: http://www.mathworks.com/matlabcentral/fileexchange/.

[6] Ferenc Wettl, Gyula Mayer, and Péter Szabó. *LaTeX kézikönyv.* Panem Könyvkiadó, 2004.

Függelékek

A. Függelék

Licencek eredeti szövegei

A.1. MIT License

The MIT License (MIT)
Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

A.2. Apache License 2.0

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the

Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work

by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason

of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

A.3. BSD

A.3.1. 4-clause BSD (eredeti)

Copyright (c) <year> <copyright holder> . All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright

notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement:
This product includes software developed by the <organization>.

4. Neither the name of <copyright holder> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY COPYRIGHT HOLDER "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL <COPYRIGHT HOLDER> BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.3.2. 3-clause BSD (módosított)

Copyright (c) <YEAR>, <OWNER>
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.3.3. 2-clause BSD (egyszerűsített)

Copyright (c) <YEAR>, <OWNER>
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.4. GNU GPL

A.4.1. GNU GPLv2

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the

terms of this License.
c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)
These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)
The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to

these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the program's name and an idea of what it does.
Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301,
USA.
Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type 'show w'. This is free software, and you are welcome
to redistribute it under certain conditions; type 'show c'
for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright
interest in the program ‘Gnomovision’
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

A.4.2. GNU GPLv3

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs,

or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This

requirement modifies the requirement in section 4 to "keep intact all notices".
c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.
A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.
A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information

must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it.

(Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.
All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the

form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make,

use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies

made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public

License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above

cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year> <name of author>

    This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

    <program> Copyright (C) <year> <name of author>

```
    This program comes with ABSOLUTELY NO WARRANTY; for details type
    ‘show w’.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, your program’s commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

B. Függelék

Második dolog

B.1. Még egy kis melléklet