Harkály Gergő

Programtervező informatikus BSc.

Korszerű webtechnológiák szakirány

Felsőoktatási tankönyv informatikus hallgatók részére.

Verzió: 2019. január 30.

Tartalomjegyzék

I. rész

Preambulum

Jelen kiadvány kizárólagos használatára jogosult: ....

Az alábbi könyv a Miskolci Egyetem Programtervező informatikus BSc. 2012-2015 szak Webtechnológiák szakirány során hallgatott előadások alapján készült, azonban hasznos lehet minden, a matematika és az informatika világa iránt érdeklődő számára, kortól és nemtől függetlenül.

A szerző nem titkolt célja, hogy e könyv segítségével az ifjúság képviselői minél nagyobb számban, kimagasló eredményeket elérve végezzék el sikeresen a Programtervező informatikus BSc szakot, melyet követően a nemzetközi, az IT teljes területén jelen lévő vállalkozás keretein belül kamatoztathassák megszerzett tudásukat.

Bár a könyv készítői törekedtek a tökéletességre, ezen igényük még nem jelenthet garanciát annak megtörténtére. Így amennyiben bármilyen észrevétele, javaslata lenne a Kedves Olvasónak, örömmel fogadjuk visszajelzéseit a `https://www.harkalygergo.hu` weboldalon található elérhetőségek valamelyikén.

Tekintettel arra, hogy a világ, így benne a matematika és az informatika témaköre is folyamatosan változik, fejlődik, érdemes mindig a legújabb kiadványt kezébe vennie. A kiadások fejlődését sorszámozástól eltérően mi a borítólapon feltüntetett dátummal jelezzük, mely a könyv legutolsó módosítása során automatikusan változik, s mely frissített könyv szintúgy elérhető a fent közölt honlapon.

Amennyiben úgy érzi, így utólag szívesen támogatná jelen könyv megszületését, vagy jövőbeli továbbfejlesztését, tetszőleges összeggel adhatja le támogatását PayPalon keresztül az alábbi link segítségével: `https://paypal.me/harkalygergo`

II. rész

Használati útmutató

Jelen dokumentum kizárólag a vizsgára való felkészülés során nyújt segítséget, de nem helyettesíti az előadások látogatását, és a tananyag önálló elsajátítását!

Az egyes témakörök a következőképpen épülnek fel:

- definíciók
- témakör részletes bemutatása
- példák
- vizsgafeladatok

A könyvben az egyes információk az alábbi jelölésekkel jelennek meg:

- definíció: dőlt betűtípus
- megjegyzés: * : csillaggal jelölt, az adott szöveg végén, 10-es betűméret
- hivatkozás: _ : aláhúzással jelölt

III. rész

Adatbázis rendszerek I.

1. fejezet

Vizsgák

1.1. 2008.01.04.

1. Adja meg a többértékű tulajdonság ábrázolási módját az ER, IFO modellben és a megvalósítását a hálós és relációs modellben.

2. Relációs algebra join műveletei (jelölés, értelmezés). Adja meg a szelekciós join relációs kalkulusbeli alakját.

   - alap join: két reláció rekordjainak párosai, jele: $r_1 \triangleright\triangleleft r_2$
   - szelekciós join: a két reláció rekordpárosaiból a feltételnek eleget tévő párosokat adja eredményként, jele: $r_1 \triangleright\triangleleft_{feltetel} r_2$
   - outer join: olyan szelekciós join, melyben az illeszkedő pár nélküli rekodok is bekerülnek az eredmény halmazba (üres étékekkel kiegészítve), típusai:
     - left, jele: $r_1 + \triangleright\triangleleft_{feltetel}\ r_2$
     - right, jele: $r_1 \triangleright\triangleleft_{feltetel} + r_2$
     - full, jele: $r_1 + \triangleright\triangleleft_{feltetel} + r_2$

3. Redundancia oka, veszteségmentes felbontás fogalma és az ide vonatkozó tételek.

4. Adott az alábbi séma: csapat [kod, nev, pont] és jatekos [kod, nev, csapat, kod, poszt, kor]. SQL parancsok:

   - A CSAPAT tábla kiegészítése VAROS mezővel
     ```
     ALTER TABLE csapat ADD varos VARCHAR(64);
     ```

- Az X nevű csapatban lévő játékosok rekordjainak törlése
  `DELETE FROM jatekos LEFT JOIN csapat ON jatekos.csapat=csapat.kod WHERE csapat.nev='X';`
- Az X poszton játszók neve és csapatuk neve a játékos neve szerinti sorrendben
  `SELECT nev, csapat FROM jatekos LEFT JOIN csapat ON jatekos.csapat = csapat.kod WHERE jatekos.poszt='X' ORDER BY jatekos.nev ASC`
- Mennyi X poszton játszó van az egyes csapatokban (csapatkód és létszám)
  `SELECT COUNT(*), csapat FROM jatekos WHERE poszt='X' GROUP BY csapat`
- Mely csapatokban van 3-nál kevesebb olyan játékos, akik fiatalabbak az átlagéletkornál.
  `SELECT IF(COUNT(*)<3,csapat,'') FROM 'jatekos'WHERE kor<(SELECT AVG(kor) FROM jatekos) GROUP BY csapat`

1.2. 2008.01.07.

1. A táblák különböző megvalósulási típusai, működési módjuk, létrehozásuk SQL parancsai.
2. Redundancia oka. A független felbontás fogalma és tétele. Normalizálja az alábbi sémát: R(A,B,C,D,E), ahol B->D, EA->C, AB->AD, E->A
3. A relációs algebra csoportképzés és osztás műveletei. Jelölés, típus és jelentés. Hogyan adható meg a szelekciós join algebrai, relációs kalkulusbeli és SQL alakját?
4. Adott DOLGOZO[kod, nev, beosztas, fizetes, projekt], PROJEKT [nev, varos] séma. Adja meg az alábbi műveletek SQL megfelelőjét:
   - A PROJEKT tábla bővítése létszám mezővel.
   - Azon dolgozóknak, akiknek a nevében szerepel a KO szó, új beosztásuk IRNOK
   - Az IRNOK-ok átlagánál többet keresők darabszáma
   - A dolgozó neve a hozzá tartozó projekt nevével együtt a dolgozó neve szerint rendezve
   - Mely projektekben dolgoznak 5-nél többen;

- Hogyan lehet automatikusan növekedő értéket adni a KOD mezőnek egy új DOLGOZO rekord felvitelekor

IV. rész

Operációs rendszerek GEIAL302B

Negyedik félévre ajánlott tárgy. Előadó: Dr. Vincze Dávid.

Jelen jegyzet többségében Dr. Vincze Dávid előadásai és diái, valamint Dr. Vadász Dénes elektronikusan kiadványa alapján készült.

2. fejezet

Dolgozat

2.1. Aláíráspótlás - 2014.05.27.

A sikeres teljesítéshez 50 százalék feletti pontszám elérése szükséges.

1. Mik és hogyan működnek az osztott memória kezelés rendszerhívásai? Írja le az osztott memória használatának lépéseit! (8 pont)
   A process egy közös memória területen keresztül kommunikálnak, melynek előnye, hogy gyors és nagy információ mennyiség vihető át. További tulajdonságai: indirekt, szimmetrikus és zéró pufferelt. Megvalósítása az alábbi rendszerhívásokkal történik:

   - shmget() // készítés, beazonosítás
   - shmat() // a processz címtartományára csatolás
   - shmdt() // lecsatolás
   - shmctl() // kontroll, jellemzők lekérdezése

   Készíthetünk (asszociálhatunk) adott méretű osztott memória szegmens ist. Ekkor hozzáférések állíthatóak be. Ezt leképezzük a process címtartományára (megadjuk, milyen címeken és milyen típusként lássa a process a szegmens rekeszeit). Leképzés nélkül használhatatlan a szegmens, hisz nem éri el a process! Használjuk az adott címekre való hivatkozásokkal a szegmens rekeszeit (pointer) . Végül lekapcsoljuk a címtartományt, esetleg töröljük a szegmenst.

2. Mi az a CoW (Copy-on-Write), hogyan működik, hol lehet szerepe OS szemszögből? (4 pont)
   Nem készül tényleges másolat, csak hivatkozás, de amint írás történik, készül egy másolat, és oda történik az írás, például a fork() parancsnál.

3. Mit jelent az, hogy könnyűsúlyú processz? Hol van ennek szerepe? (4pont)
   Párhuzamos programozási környezetben a végrehajtási menet neve a "könnyűsúlyú" processz, vagy fonál, vagy szál (thread).

4. Milyen célt szolgálnak a "/dev/zero" és "/dev/null" speciális file-ok? (4 pont)
   /dev/null: Olyan, mint egy "fekete lyuk". Bármi, amit beleírunk, az eltűnik, olvasni pedig nem lehet belőle, mivel rögtön EOF-ot (End Of File) ad. Például fájl tisztítása: cat /dev/null > /var/log/messages /dev/zero : Bármi, amit beleírunk az eltűnik, olvasni pedig nullát lehet belőle.

5. Mitől virtuális a virtuális memória (2 dolog)? (4 pont)
   A virtuális memória a processz számára azt a képzetet kelti, hogy igen nagy címtartományt és memóriát kezelhet. Minden processznek igen nagy virtuális címtartománya lehet, ami egy vagy több résztartományból állhat. A virtuális címek virtuális memória cellákat címeznek. A virtuális cellákat a memóriamenedzselés biztosítja: a cellákat vagy a fizikai memória cellái, vagy a másodlagos memória (diszkek) cellái adják.
   A taszképítés során virtuális címeket generálnak, a processzek kontextusában virtuális címeket találunk. A processz futása során dinamikus címleképzés van: a virtuális címeket a buszra kiadható fizikai címekké kell leképezni. A processznek nem kell törődnie azzal, hogy a hivatkozott vírtuális címhez tartozó cella jelenleg fizikai memória cella-e vagy a másodlagos tároló cellája-e. Utóbbi esetben a memóriamenedzser ki-be lapozást is végez, gondoskodik arról, hogy a virtuális memória másodlagos táron található része bekerüljön a fizikai memóriába.
   A virtuális memória jóval nagyobb mint a fizikai.

3. fejezet

Vizsgák

3.1. 2014.06.26.

A sikeres teljesítéshez négyből három kérdésre helyes választ kell adni.

1. Mi az az SVID (nem csak az, hogy minek a rövidítése)?
   Az SVID (System V Interface Definition) az AT&T által szabványosnak tekintett System V rendszerhívásainak és szubrutinjainak egységes felhasználói felületét és definícióit határozza meg.

2. Egy átlagos számítógépen melyik hardware-ből jön a legtöbb interrupt?

3. Milyen két futási szint/mód van általában?
   Felhasználói (user) és kernel mód. A kernel-hívás kívülről egyszerű függvény- vagy eljáráshívásnak tűnik, hiszen megadjuk a szolgáltató rutin nevét és aktuális paramétereit. Valójában ez nemcsak egyszerű függvény- vagy eljáráshívás paraméter átadással, hanem egyben úgynevezett trap: a processzor felhasználói módból kernel módra vált.

4. Mi az a hard link?
   Az úgynevezett hard link esetén egy új directory bejegyzés készül a megfelelő jegyzékben. Az új jegyzék-bejegyzésben az i index egy már meglévő fájlhoz tartozó i-node-ra mutat. Ugyanakkor az i-node-ban növekszik a linkszámláló, amely azt jelzi, hogy ezen i-node-dal azonosított fájlra több jegyzék bejegyzés is hivatkozik. Minden más attribútum változatlan. Marad a védelmi maszk,

a tulajdonosi és csoporttulajdonosi bejegyzés, ezért az elérési jogok korlátozhatnak! (Még egy másik korlát is lehet: csakis ugyanazon a fájlrendszeren lévő fájlok kapcsolhatóak össze hard linkkel!) Fájl törlés esetén csak a linkek száma csökken, és egy directory bejegyzés tűnik el, ha a linkszám még nem 0, az i-node továbbra is foglalt.

5. Hogyan keletkezhet kivétel (exception)?
   A kivétel (exception) váratlanul, de szinkron módon keletkezik, hiszen egy adott instrukcióhoz tartozik.

1. Gigabit Gézának 8 magos CPU-ja van. Olyan letöltőprogramot használ Windows 8 alatt, ami csak egy szálon képes futni. Mennyi ilyen letöltőprogramot tud elindítani egymás mellett, illetve mennyi tud egyszerre egy időben futni ezek közül?

2. Bill Gates Linuxot installált egy 2GB fizikai memóriával rendelkező számítógépre. Elindít rajta egy Mozilla Firefoxot. Mitől függ, hogy összesen mennyi memóriát használhat a Firefox processze?

3. Letöltő János pendrive-ján FAT filerendszert használ, sajnos meghibásodik rajta az indextábla, mert nem választotta le rendesen, mielőtt kihúzta. Helyre tudja állítani róla az adatokat? Hogyan?

4. Torrent Gyuszi venni szeretne egy USB-s locsolókannát. Már a pénztárnál áll sorba a kannával a kosarában, azonban kifogy a papírtekercs a pénztárgépből, így megáll a sor. Ez holtpont helyzet? Ha igen, milyen?

. . . . . .

3.2. 2014.05.27.

3.2.1. Beugró

1. processz kontextus

2. laphiba

3. kivétel megszakítás különbség

4. mi a rendszerhívás?
5. IPC

Vizsgakérdések

1. Kernelekről: monolitikus, réteges struktúra, mikrokernel
2. Lapozásos virtuális memória : allokálás/címképzés
3. i-node
4. Osztott memória: alapelve, rendszerhívásai

3.3. 2011. június 23.

3.3.1. Beugró kérdések (rövid fogalom meghatározások, 5-ből 3):

1. spooling
   A SPOOLING (simultaneous peripherial operation on-line) egy mozaikszó, melynek jelentése: szimuáltán perifériás műveletek online. A nagykapacitású, gyors és véletlen hozzáférésű mágnesdobokat és -lemezeket a SPOOLING lényegében hatalmas méretű pufferként használja oly formán, hogy egyszerre több munkát is a lemezre tölt.
2. IPC
   Az IPC egy olyan mechanizmus, mely lehetővé teszi, hogy processzek egymással kommunikáljanak, műveleteket összehangoljanak, illetve szinkronizáljanak. Az IPC két művelete a küldés és fogadás.
3. TSL instrukció
4. swapping
   Egy blokkolt processz memória területét teljes egészében kiírja a háttértárra, hogy a felszabaduló helyre el lehessen helyezni egy másik processzt, majd visszatölti mikor a processz sorra kerül

5. FCB
   Az FCB a File Control Block szavakból alkotott mozaikszó, mely a fájlrendszer gyorsításának egy módja, lényegében az ismételt keresések elkerülése érdekében egy file megnyitásakor jön létre

3.3.2. Kifejtendő kérdések:

1. szemafor
   1965 körül Dijkstra javaslatára alkották meg a kölcsönös kizárás megoldására. A klasszikus szemafor egy pozitív egész számot tartalmazó változó és egy hozzá tartozó várakozási sor (melyen processzek blokkolódhatnak). A szemaforon - kivéve az inicializációját - két, atomi (ezzel biztosított, hogy ha az egyik folyamatban, másik nem léphet életbe) operáció hajtható végre. A két művelet:

   - DOWN (P: Passeren): Ha a változó nagyobb, mint nulla a változó értékét csökkenti eggyel, ha nulla akkor a processz a várakozó sorba kerül (blokkolódik).
   - UP (V: Vrijgeven [vrejhéfen]): A változó értéke növekszik eggyel és ha nulla volt akkor jelez egy blokkolódott processznek, hogy felébredhet.

2. I-node-s file allokáció részletes ismertetése.
   Az i-node az "information node" rövidített változata. Az i-node lényegében egy bejegyzés az információs táblában (i-list), mely a fájlok jellemzőit, többek között az elhelyezkedésükre vonatkozói információkat tartalmazza. UNIX rendszerben így minden fájl egyedi i-node-dal rendelkezik. Mivel a fájlok egyes blokkjai a lemezen szétosztva, több helyen találhatóak meg, így egy táblázat foglalja össze a fájl blokkjait és az azokat tartalmazó fizikai helyeket. Az i-list bár a logikai lemezen található, a kernel beolvassa a teljes táblát memóriába és azon manipulál.

   - Előnyök:
     - Nincs fregmentáció
     - Egy blokk megtalálásához kevesebbet kell olvasni mint bármelyik megelőző módszernél.
     - A file elhelyezési információk egy helyről elérhetők (nem pedig jegyzék + indextábla)
     - Kevésbé sérülékeny

- Hátránya:
  - A szabad blokkok külön nyilvántartás igényelnek (index tábla egyben szabad blokk nyilvántartás is)

3. Ütemező döntési stratégiái. Összefüggés az állapotokkal. Ütemező feladatai. Az ütemező döntési stratégiája, hogy mely futásrakész processz kapja meg a CPU-t. Ezekből többféle is létezik:

   - nem beavatkozó (non-preemptive)
     - run-to-completion jellegű: a processz, ha megkapta a CPU-t, addig használja, míg a (rész) feladatát el nem végzi
     - együttműködő (kooperative): a processz, ha megkapta a CPU-t, lemondhat róla.
   - beavatkozó (preemptive): akkor is elvehetik a CPU-t egy processztől, ha az nem akar lemondani róla.
   - szelektív beavatkozó: bizonyos processzek (ált. rendszer processzeknél) futásába nem lehet beavatkozni, más processzektől elveszik a CPU-t, még ha nem is mondana le róla.
   - Beavatkozó: folyamatok nem mondanának le a CPU használatáról, beavatkozva elveszik tőlük bizonyos körülmények között.

   Ütemezési döntési helyzetek bekövetkezhetnek:

   - 1. futó -> blokkoltba (wait/sleep/request álapotátmenet): I/O kérés, vagy gyermekprocesszre várakozás miatt
   - 2. futó -> futásra kész (preemption átmenet): megszakítás bekövetkezése miatt
   - 3. Blokkolt -> futásra kész (signal/respond állapotátmenet): I/O befejeződése miatt
   - 4. Terminálódás

   Ha ütemezési döntési helyzet csak az 1.c és a 4. esetben lép fel, akkor az ütemező nem beavatkozó. A 2. és 3. esetben is döntési helyzet van, akkor már beavatkozó az ütemező. Ütemező feladatai:

   - Döntés a beavatkozásról
   - Döntés a kiosztásról
   - Context Switch elvégzése

4. Szegmentálásos virtuális memória kezelés. Elhelyezés, nyilvántartás, címképzés, szegmenshiba lekezelése, előnyei, hátrányai.

5. Kilapozási algoritmusok számpélda: LRU kilapozási algoritmus szerint töltse ki!

V. rész

Webes alkalmazások