<https://docs.google.com/document/d/1dRxf20PMHfC4J8IiLnUVwjzqDZMWen4U5KbK3vpF-Xs/edit>

FPoznámka: Opravný termín sa nachádza pod riadnym termínom.

Riadny termin 2013/2014

Hola!  
 Cecko

-

**Otázky**

1. Napíš rozdiely medzi dávkovým OS s multiprogramovaním, OS so zdieľaním času.

odpoved:

*Multiprogramovanie* je založené na myšlienke, že pokiaľ jeden proces čaká na I/O operáciu (a teda práve nevyužíva CPU), iný proces môže využívať CPU.

*Dávkový*: programy (úlohy) sa zadávajú ako dávky, pocas výpoctu nie je interakcia medzi používatel’om a jeho úlohou.

*OS so zdieľaním času* sú založené na princípe, že prostriedky (CPU) sú zdieľané medzi viacerými používateľmi v tom istom čase. Každému používateľovi je prostriedok pridelený len istý veľmi krátky časový úsek. Používatelia majú takto pocit, že každý z nich- môže využívať prostriedky v tom istom čase. (gt)

1. Napísať 3 veci podľa čoho sa meria efektivita systému.

odpoved:

*využitie zariadenia (utilization), priepustnosť (throughput), doba odozvy (response time))*

// nemali ste niekto aj otazku na vykon systemu?

VYKONNOST podla Tanenbauma : Troughput, Turnaround time, CPU utilization

1. Rozdiel medzi funkciami semafora a podmienkovými premennými v monitore (skor ich spolocne a rozdielne znaky).

odpoved:

Premenne semafora maju ,,hodnotu a rad pre procesy, premenne monitora iba rad.

Hodnota premennej v semafore sa testuje v jeho funkcionalite a v monitore sa testuje v klientskom kode

*sem\_wait sa zablokuje len ked je semafor “vycerpany”*

*cond\_wait sa zablokuje vzdy*

*sem\_signal zvysi hodnotu semafora ak je cakajuci rad prazdny*

*cond\_signal sa strati ak je cakajuci rad prazdny*

Sú tri procesy. Každý z nich sťahuje 10MB súbory z internetu. Keď proces stiahne 50 súborov, spracuje ich a vymaže. Môže nastať uviaznutie? Napíšte 4 podmienky pre uviaznutie a diskutuje, či môžu v tomto prípade nastať.

odpoved:Môže nastať uviaznutie.

Ak by bol HDD MENŠÍ ako 1480MB (3\*49\*10MB + MENEJ ako 10MB) tak by sa mohlo stať, že každý proces by mal stiahnutých 49 súborov a 50. súbor by sa ani jednému z procesov už nepodarilo na disk uložiť. (je dostupných menej ako 10MB).

//preco nie ked je HDD mensi ako 1500? lebo ak bude stiahnute 3\*49 + 99,999… percent z kazdeho 50teho suboru, tak to podla mna uviazne //podla mna si este pred stiahnutim jedneho suboru alokuje celych 10MB na disku, a teda staci 1480MB-ovy disk a jeden proces si tam moze dat celych 500MB, spracuje ich a vymaze a potom uz ostatne dva procesy sa zmestia bezproblemov a mozu sa tiez dokoncit.

//tak ako to teda je? Kto ma teda pravdu?

//podla mna to netreba presne rátať, máme diskutovať teoreticky, nie udať presné číslo že ako velky ma byť ten disk aby nastalo uviaznutie, to presne aj tak nevieme lebo nevieme ako veľký priestor pridelujeme naraz, za to že má subor 10MB ešte nie je hned jasne že naraz procesu pridelíme 10MB kludne to može byť zopar kB, čiže ja by som tam napísal len podmienky a k nim obkeci že čo to v tomto prípade znamená a kde može prísť problem

Podmienky uviaznutia (deadlock moze nastat ak su v systeme simultanne splnene vsetky z tychto podmienok): //EEE…. coze? cize myslite naraz? Podla mna, musi system splnat aspon JEDNU z tych podmienok. //Určite to je všetky naraz. Vezmi si napr. druhú - ak systém môže prostriedok hocikedy urvať a dať ho niekomu inému, tak asi ľahko zabezpečí, aby sa to nezaseklo

* Vzájomné vylučovanie (na disk môže v jednom čase pristupovať len jeden proces)
* Nepreemtívne prideľovanie (diskový priestor sa nemôže procesu násilne odobrať)
* Čiastočné pridelenie (Proces ukladá dáta na disk po blokoch)
* Čakanie na prostriedky (proces čaká až kým nemá na disku dostatok miesta)

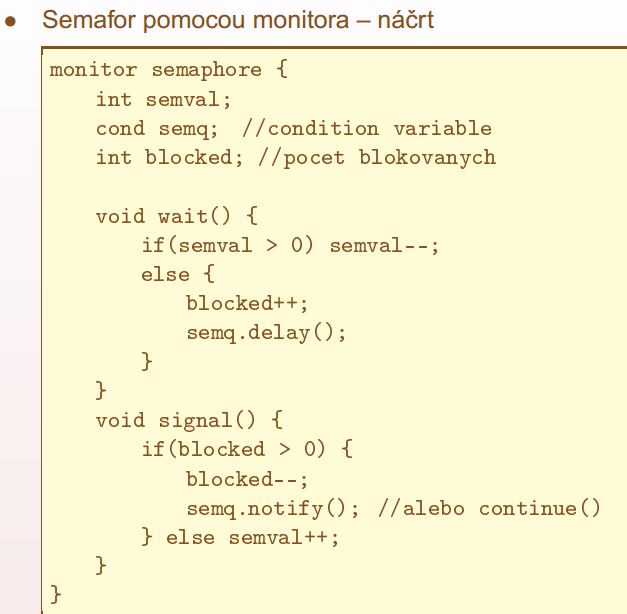
1. Popísať ako by ste implementovali semafory pomocou monitoru.

odpoved:

Pre kazdy semafor sa pouzije privatna premenna a funkcie wait() a signal() , ktore volaju funkcie cwait() a csignal() privatnej premennej.

*//takto by som to riesil v pripade, ze sa jedna o semafory typu MUTEX. V opacnom pripade tam treba asi este pridat pocitadlo.*

//predpokladám že treba zbežne opísať nasledujúci kód



op

1. Napísať dve udalosti v systéme, keď sa môže stav procesu zmeniť z blokovaného na pripravený.

odpoved:

* Nastane udalosť, na ktorej vykonanie proces čakal
* Skončí sa I/O operácia, ktorú proces vykonával

(gt)

//nebola ta otazka takto? //+1

Napísať dve udalosti v systéme, keď sa môže stav procesu zmeniť na pripravený.

//potom by bola odpoved podla mna takato:

Nastane udalosť, na ktorej vykonanie proces čakal

Skončí sa I/O operácia, ktorú proces vykonával

//Ak by to bolo že sa môže zmeniť na pripravený tak na pripravený sa mení aj pri bežnom plánovaní…, že ho os v rámci plánovania vypne…+1

Zo stavu beziaci po uplynuti casoveho kvanta

1. Klasifikácia prostriedkov v systéme. Plus určiť typ pri nejakých príkladoch. *(jednorazovo použiteľné, opakovane použiteľné, viacnásobne použiteľné)*

odpoved:

*- myslím že stránkový rám - OP* ***// ZLE, dnes (4.2.2014) mi na konzultacii Solcany osobne povedal, ze ak by bol strankovy ram OP tak by nedokazal fungovat, takze to bude zrejme VP //+2***

*- nejaký blok na disku - OP*

*- nespomeniem si čo ale viem, že tam šlo o stavy takže to bol isto - VP*

*//vidlicky pri filozofoch? +1 // seriously ? :D*

*//hej hej asi hej :D // no mne sa zda ze vidlicka je opakovane pouzitelny, ked drzis vidlicku tak sa jej bud vzdas alebo ju drzis kym nedojes, nesmie sa stat ze uprostred toho ti ju*

*iny filozof vytrhne z ruky //dava zmysel, vidlicky budu v tom pripade asi* ***OP****// +2*

*-správa, papier v tlačiarni - JP (jednorazovo použiteľný prostriedok) //správa sa pošle, prijme a tým je spotrebovaná; na papier sa niečo vytlačí a tým je spotrebovaný*

*co je OP/VP ? // opakovane pouzitelny prostriedok, viacnasobne pouzitelny prostriedok*

1. Na čo sa *v*yužíva asociatívna pamäť pri stránkovaní a z čoho sa skladá. *(číslo stránky, číslo str. rámu)*

odpoved:

*Tabuľka obsahujúca dvojicu (klúč,hodnota) - dal som konkrétne tú s adresami (page*

*frame,page)*

*- na urýchlenie prepočtu adries`*

*Na zrýchlenie prekladu lineárnej adresy stránky na fyzickú adresu stránkového rámu sa takmer u všetkých procesorov používa asociatívna pamäť*

*Asociatívna pamät obsahuje položky v tvare (stránka, stránkový rám), pričom*

*kľúčom pre vyhladávanie je stránka. Adresovací hardvér hľadá adresu strány*

*vo všetkých položkách paralelne, čo je veľmi rýchle.*

*Iba ak nenájde správnu položku, musí zisťovať adresu stránkového rámu*

*pomocou tabuľky stránok, čo trvá oveľa dlhšie.*

**PríkladyRT**

**Realne poradie prikladov na RT Bolo:**

**P2 Synchro pomocou semaforov, P3 bankar, P4 R-R vypoctove casy, P5 Second chance, P6 File system**

**1. Fragment programu. 4 stránkové rámy, používaš second chance. Vypísať fyzickú adresu pre pole A[64], B[218] resp. A[100] B[200],**

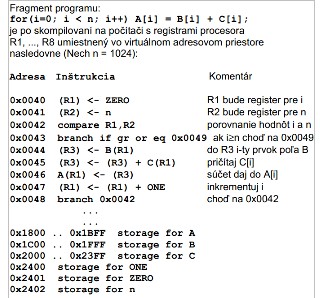
**//to bolo za 10b**

**//n=2 a stranka mala 512B, reset R bitu každé 4inštrukcie,inak bol kod identicky //okrem adries s tymto, tak robte, kto ma cas :)**

**// niekto čo to mal za full a mohol by to celé prerátať? :D**

**// btw zadanie bolo pri jednej sk//+2**

**upiny 4ramy 3instr restart, pri druhom 3ramy 4inst restart**

**+-** **t**

**Viem ze A[100] bola 0x1E64 a B[200] nebolo v pamati //+5**

**// je to 0x1E64 → rám = 0x0F = 15 → 15\*512(page size) + 100(offset) = 7780 = 0x1E**

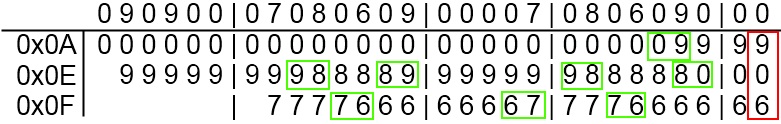
**64 -toto je spravny vypocet, 0x1C64 je zle//pre A[100] mi vyslo 0x1C64 //+2 //-2 :D // dajte mu niekto facku za mňa :D**

**//pocet vypadkov neratal nikto? mne vyslo 13// tie su tu nepodstatne // mne vyslo 11/ / tiez mi vyslo 11// ked sa pozriete na fotku uvidite tiez 11 :)**

**40TU JE MOJE RIESENIE SNAD TO BUDE DOST VIDIET aspoň   
odskúšam foták na novom mobile hah :D**

**pocet ramov = 3 (0x0A,0x0E,0x0F), restart po 4 inštrukciach, n = 1, takže prejdú 2 iterácie(malo byt n=2 ale vsimol som si neskor, takže je to pre n=1), FA adresy pre A[100] a B[200] - Second chance**

**//Pravdepodobne to nie je úplne správne, pretože potom ako spravíš second chance algoritmus, ktorý ti určí, že ktoré stránky vyberieš, tak ich ešte musíš správne usporiadať v stránkových rámoch, čo sa robí nasledovne… keďže 6 ti vyšlo na správnom mieste, tak celkový výsledok by mal byť správny, ale to bola len náhoda**



**// Ked pri Second Chance vnesiem stranku, nastavim R hned na 1?**

**//takže potom ako som hodinu pátral v dococh čo vlastne po nás tí … na skúš**

**ke chcúsom došiel k tomu, že sa na konci vážne dorába iná tabuľka…//nechcem to tu kazit. len na zamyslenie, ako moze byt cislo stranky pri B 13, ked 1C00 pri velkosti stranky 512 dava 14? to nedava zmysel A,B,C vsetky zaberaju 2stranky.**

**Kedže som jej sám nechápal, tak tu v skratke napíšem ako to funguje… Predtým by som rád však poďakoval anonymovi čo ma upozornil na chybu :)**

**Takže potom ako si spravíte hlavnú tabuľku page faultov klasickým postupom…**

***1) pozriete sa na klasickú tabuľku, dobre je si značiť pekne kde je PF a čo letí von***

***2)začnete robiť novú tabuľku s tým,že stránky zostávajú vo framoch kde boli prvý krát pridané, menia sa až keď vyletia b***

***3)pozerám postupne si zakladnú tabuľku a keď dôjde k obetovaniu, pozriem sa, ktorú som hore obetoval a namiesto nej napíšem do daného stránkového rámu novú stránku!!!***

***4)takto postupujeme a nakoniec nám vznikne konečné umiestnenie stránok v ráme z ktorého môžeme vyrátať FA .)***

**PS:dúfam, že to z tohto každý pochopil, a ešte raz dik za upozornenie a sorry ostatným, že som nedal 100% riešenie :) //v pohodicke, lepsie sa takto postupne dopracovat k spravnemu rieseniu, ako len kukat a spoliehat sa na to ze niekto iny to vyriesi ;) // diki moc :)**

**// Ako sme zistili offset? //offset v tomto pripade vyjadruje to A[100], tak tomu aspon rozumiem // rozumies, offset je 100**

**\_**

**2. Filozofovia pomocou bankárovho algoritmu.**

**N = 5 ; Vieme, že 0. a 3. filozof nemali žiadne vidličky, 1. mal ľavú aj pravú , 2. mal pravú a 4. ľavú**

**- napísať prečo sa dajú filozofovia riešiť pomocou BA**

**- načrtnúť ten počiatočný stav (spravil som maticu Filozofovia x Vidličky)**

**- nakoniec načrtnúť postupovú cestu BA a ukázať že je bezpečný stav**

**Bankar - prednaska 10, slajd 33**

[**http://osa.fiit.stuba.sk/os/11/os\_11\_2.ppt**](http://osa.fiit.stuba.sk/os/11/os_11_2.ppt)

**// Môže sem niekto napísať riešenie za plný počet ? lebo ten kokot čo mi to opravoval mi za toto dal 4/6b**

**//Možno by bolo dobré urobiť dve tabuľky jedna need a druhá alocated tak ako sa to robí v zbierka mal som to tak za 6**

**// a zato mi strhli ako 2b ??? jebnute hlavy, skus tam dopisat alebo opravit ako si to mal ked si mal za full**

**a) BA -> riešenie deadlocku pomocou vyhýbania -> hladáme postupovú cestu**

**procesov, aby nedošlo k deadlocku, to isté je filozofovia, hladáme postupnú postupovú cestu ako pridelovať vidličky(prostriedky) procesom(filozofom) aby nedošlo k deadlocku**

**Takze na konzultaciach vravel, ze jeho nezaujima, co robi bankar a akym sposobom to teda mozno pomocou bankara vyriesit, ale ide o to, PRECO je mozne bankara uplatnit prave na tuto situaciu. Cize by**

**bolo vhodne si nastudovat, v akych situaciach je mozne uplatnit bankara, na aky typ prostriedku ( OP, VP … ) a za akych podmienok a potom povedat ze prave toto problem filozofov splna... Pokusim sa to zistit, ale ak niekto nieco najdete tiez tak sem napiste**

**// tak ja som mal 4/6b podľa mňa mi chýbala podstatná vec v tej prvej časti a to to, že si treba uvedomiť podstatnú vec a to takú, že nie je fork ako fork, nie sú všetky hodené na stôle, nie každú fork môže každý filozof zobrať, proste je dôležité si uvedomiť že je tu viac typov prostriedkov. // ano to budes mat asi pravdu, bavili sme sa so Solcanym chvilu aj o pocte roznych prostriedkov, a vysvitlo, ze prostriedok nie je jeden, ani dva ale kazdy filozof ma dva prostriedky, pricom su zdielane so susedmi. A ak sa mozem spytat, co si napisal do bodu A? to co je vyssie napisane cervenym? mohol by si to prosim pekne zrekapitulovat? po tom, co nas vsetkych Solcany totalne rusil na konzultaciach mam pocit ze uz neviem nic :D…**

**//ja som sem dával toto riešenie a napísal som to čo je hore + aloc tabulku + postup bankara(nemyslim ze need table je potrebna a ak ano nie ta co je nizsie ale taka co mu v dakom okamihu treba nie co mu treba dokopy, pretoze uz nieco v ruke ma vid alloc ), no a mal som 4/6 a myslím že kvôli tomu a) kde chýbalo to že je viac typov prostriedkov lebo to je podľa mňa v tomto velice dôležiá info // ja som rozmyslal, ci by sa tam nedalo napisat nieco v zmysle, ze na to, aby mohol byt bankarov algoritmus aplikovany na dany problem a aby spravne fungoval, je potrebne, aby boli k dispozicii tieto informacie: kolko dany proces (filozof) potrebuje prostriedkov (a akych), kolko a akych prostriedkov v danom momente proces ma a kolko prostriedkov ma bankar k dizpozicii ( z kazdeho typu prostriedku). Kedze problem vecerajucich filozofov nam tieto info poskytuje ( resp. ich vieme vycitat), je mozne bankarov algoritmus na problem vecerajucich filozofov pouzit. Tym, ze bankar tieto informacie bude mat dokaze pracovat korektne, a teda dokazeme riesit filozofov pomocou bankara. skuste sa mi na to niekto pls vyjadrit :)**

**//už je to jedno stejne budeme čil filozofov na 100% implementovať. :)**

**Alocated Need**

**b) V0 V1 V2 V3 V4 V0 V1 V2 V3 V4**

**F0 0 0 0 0 0 F0 1 0 0 0 1**

**F1 1 1 0 0 0 F1 0 0 0 0 0**

**F2 0 0 1 0 0 F2 0 1 0 0 0**

**F3 0 0 0 0 0 F3 0 0 1 1 0**

**F4 0 0 0 1 0 F4 0 0 0 0 1**

**//opravil som cislovanie a NEED tabulku, teraz je to spravne, predtym ta need table bola MAX table a nie NEED**

**c)**

**takže bankár má V4, ale kedže F má V0 a V1 naje sa a hned mu ich dá atd… myslím, že bankára pozná každý… stav je bezpečný**

**//a to stačí takto slovne popísať? lebo to bolo jediné čo som urobil a body za to nemám**

**// nie ja som to aj pekne pokreslil ako pri bankarovi, len sa mi to nechelo tu rozpisovat, kazdopadne ja som mal 4/6 takze by bolo fajn keby to dakto doplnil, tu need tabulku som ja nemal, ale nezda sami zeby to bolo -2b**

**//mal si napisane preco to mozme riesit banka rom?**

**//to riešenie čo tu je som písal ja, áno mal som to čo tu je okrem need tabulky**

**//napadlo ma mozno doplnit kedy by bol nebezpecny stav-->keby ma kazdy lavu vidlicku ale to nevyzadovali tak nevim co im chybalo**

**//** **Solcanyho som sa pytal a vravel, ze jeho nezaujimaju popisy a slohy ( podla jeho v zadani je jasne napisane zistit na zaklade bankarovho algoritmu, a teda tak, ako to robi bankar a nie popisat) takze treba minimalizovat text a opisy, strucne jasne pomocou tabuliek presne tak, ako to robi bankar**

**3. Pomocou semaforov spraviť posielanie správ medzi dvomi procesmi. Bolo potrebné implementovať funkcie send(msg\_t m) a receive(msg\_t \*m).**

**h**

**semaphore sem1 = 0, sem2 = 0;**

**msgt buff;**

**send (msgt m) {**

**buff = m;**

**sem1.signal();**

**sem2.wait();**

**}**

**recieve(msgt \*m) {**

**sem1.wait();**

**\*m = buff;**

**sem2.signal();**

**}**

**// tak je to dobre, mám to tak za 6b /+3,25 (0,25 je Miroslav Krotky)**

**//podobne //nemôže sa stať že sa odošle ďalšia správa skôr ako stihne prijať tu predchádzajúcu? a preto by mal ist jeden zo semaforov este pred upravu toho buffera, aby zabranil odoslaniu dalsej spravy skor ako bola ta posledna prijata, tak ako to napisal Gabo… ako kedze ste za to dostali plny pocet bodov, tak sa nejdem hadat, len mi to pride tak logickejsie // neviem ako to presne myslis, ale ked sa odosle jedna sprava, dalsia sa neposle, dokial nebola ta prva prijata… tym sem1.signal() povie receiverovi, ze: “pozri si spravu, uz som ju poslal”, vtedy si receiver pozre spravu, ulozi ju a az potom da signal senderovi, nech posle dalsiu. Ak by sa pri prijati spravy aj preplanoval proces na sendera, tak on je stale na sem2.wait(), pretoze receiver mu este neposlal signal, ze moze posielat dalej**

**// alternatíva FAIL //+3**

semafor sem\_send=1;

semafor sem\_recv=0;

send(msg\_t m) {

sem\_send.wait();

odosli(m);

sem\_recv.signal();

}

receive(msg\_t \*m) {

sem\_recv.wait();

m = prijmi();

sem\_send.signal();

}

//bolo zadane ze to ma byt “Rendezvous (randevú)” (tzv. synchrónna komunikácia), to znamena ze funkcia send nesmie skoncit skor ako funkcia receive prijme spravu

//A TOTO SA TU PRAVE STANE CI ? //vyzera ze ano

**4. Bola 32MB pamäť resp. 16MB. Sedem alokovaní a jeden free za použitia Buddy algoritmu. zapisane vo formate alloc(velkosť,poradie)**

**alloc(1,1) 1 1 2 4 8**

**alloc(3.2) 1 1 2 4 8**

**alloc(3,3) 1 1 2 4 4 4**

**free(1) 4 4 4 4**

**alloc(3,4) 4 4 4 4**

**alloc(1,5) 4 4 4 1 1 2**

**alloc(1,6) 4 4 4 1 1 2**

**// neplati nahodou ze najmensi mozny usek je 2 ?**

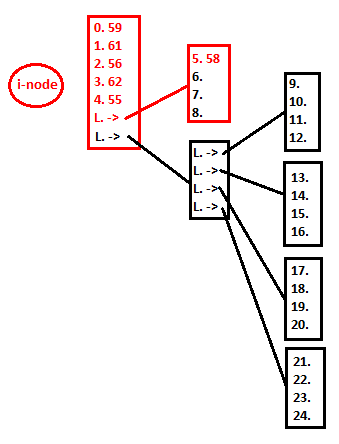
**// vsetky mocniny 2**

**// 2^0=1 //+1**

**5. Súbor zapísaný vo FAT tabuľke. Napísať ho pomocou i-node. Štruktúra i-nodu: 5 priamych, 1 nepriamy, 1 dvojito nepriamy.**

[**https://www.youtube.com/watch?v=ymYZPtrvgec**](https://www.youtube.com/watch?v=ymYZPtrvgec)

**Z tabulky vyslo 8 datovych blokov + inode + 1 nepriamy = 10 blokov**



**// išlo tu ale o to vytiahnuť z tej tabuľky proces A a ten napchať podľa čísiel do i-node**

**// co tu bolo vlastne zadane ?? z coho sa to pocita ked nevieme zigadne velkosti ?**

**// toto je podla mna uplne zle, zaprve kde si zobral ze tabulka ma 4bloky a za druhe tie bloky nemali ist od 0 atd dalej (bloky procesu A, mali ist na dane cislo bloku)… ma sa to zapisovat tak ako vo fatke prve mas cislo bloku druhe polozku… // mam pocit, ze tento obrazok je len ilustracny, neriesi tento konkretny priklad**

**//tak to sprav, viaceri sa to pytali na skuske aj sme to rieslili v skupine a nikto nevedel cislovanie, tak ked vies, nakresli..**

**//nechapem, co si hned agresivny, aky mudrc, ja sa tu nesnazim hrat na mudrkreslenie cil nemam cas len viem ze mi to vyslo tak ze to bolo az v 2-nepriamom a posledny blok mal cislo 77**

**// to urcite nie, ty ich mas radit odvrchu hned aby isiel cely subor**

**// tak ako pani, vedeli by ste to niekto prosim nejak prehladnejsie napisat? momentalne mam z tejto konverzacie velky chaos. Zacina v FS2 ( pomocou i-node) subor A hned na 0. bloku alebo nie? pocet blokov, na ktore bude nepriamy ukazovat?Ak nemame zadane kolko slov zabera ukazovatel, ako mozeme urcit**

**// ja som tu trepal blbosti, bolo to tak ako je obrázok podľa všetkého, nech to niekto potvrdí**

**//Mal som to takto za 3.5 boda nwm aky bol plny počet za ulohu//5 bolo**

**// dakto co mal full?**

**//podla mna ten postup na obrazku od 0 je teda fajn, ale je blbost naznacit ze sektor ma 4bloky prave, lebo nikde neni velkost adresy a sektoru takze nevim kolko tam mozem napchat...len by som napisal 5tku a … v dalsich nepriamych len naznacil sektory ale nie kolko je v nich blokovuvedena**

**//podľa mňa veľkosť bloku treba vydedukovať zo zadania, pretože tam je, že i-node obsahuje 5 priamych, 1 nepriamy, 1 dvojito nepriamy, z toho je podľa mňa počet blokov**

**// to ti však neudáva veľkosť jedného bloku a veľkosť záznamu v tabuľke**

**7.**

**//stym nic velkost bloku nema,tu o velkost nejde bolo povedane od solcanyho //+1**

**//napisal som to tiez tak ze som si myslel ze to bude 7 a mam za to 0 takze zle**

**// urcite sa cisluje od 0, treba vyriesit pocet blokov, na ktore ukazuje jeden nepriamy ukazovatel..**

**// ako ale vieš koľko máš dať zázamov do tabuľky na ktorú ukazuje tá prvá ?**

Opravný termin 2013/2014

**Teória:**Bola rovnaka ako na riadnom. Jedna otazka bola nova a sice, ake su dva rozdiely medzi synchronizaciou pomocou semaforov a pomocou specialnych premennych(alebo nieco v tom zmysle). //skor specialnych instrukcii //+3 //100% semafory a specialne instrukcie.

**Príklady:**

**PRIKLAD 1 : Bol zadana obsadenost disku cez zretazeny zoznam. Riesit pomocou best fit alokacie/dealokacie.**

**Tvar prvku zretazeneho zoznamu ( typ, adresa zaciatku, dlzka)**

**typ bol bud P - process**

**H - Hole**

**Napr. (P,0,120) -> (H,120,20) -> (P,140,250) ....**

**Taktiez zapisat stav zoznamu po kompakcii/kondenzacii pamati.//+**

**//vie to niekto riešiť ??**

**PRIKLAD 2 : Bankar na filozofov ( asi 4 b) a podotazka k tomu napisat nejaky synchronizacny kod pre filozofov cez mailboxy (WTF) …**

**Kod bol nejak takto:**

**void filozof(int i) {**

**think();**

**take\_fork(i);**

**take\_fork((i+1)%N);**

**eat();**

**put\_fork(i);**

**put\_fork((i+1)%N)**

**}**

**//ten kod bol take\_fork() a put\_fork() pomocou mailboxov (nepriame adresovanie, neblokujuci send, blokujuci receive). riesil som to nejak takto (*nevravim, ze spravne*):**

mbox\_t sem[5];

take\_fork(int i){

msg\_t m;

receive(sem[i], &m);

}

put\_fork(int i){

send(sem[i], true);

}

**//netreba este nejaky init(), aby v mailboxoch bola na zaciatku jedna sprava? //urcite ano, ale na to som sa nejak vykaslal. treba vybrat, ktory pojde prvy, ale to nebolo predmetom riesenia ulohy. mal si implementovat take\_fork a put\_fork… nie inicializaciu, tak hadam to je takto kompletne.**

**//ale asi to nie je dobre, lebo keby nasledkom preplanovania sa vykonala u kazdeho filozofa prva funkcia take\_fork(i), tak by boli pouzite vsetky vidlicky a uz nikto by sa dalej nepohol (doslo by k starvaci).//nie je tam potrebne pridat este nejaku prazdnu spravu na synchronizaciu? ci ked mame nepriame adresovanie tak nie?**

**PRIKLAD 3 : Planovanie procesov, ako na RT.**

**PRIKLAD 4 : Synchronizacny kod pre filozofov ( napisat wait\_for\_turn/ eating\_done ??? funkcie ), ked zacina druhy filozof a po nom mal ist (i+1)%N. Pocet semaforov nebolo nutne minimalizovat. Naraz moze papat iba jeden filozof.**

**void filozof(int i) { //nie som si ale isty, ci bol kod zadany takto na 100%//bol takto :)**

**think();**

**wait\_for\_turn(i);**

**eat();**

**eating\_done(i);**

**}**

semaphore sem[N] = {0};

sem[2] = 1; **//zacat mal druhy filozof, cize jeho index je 1, nie? //ale tusim bolo napisane ze mal zacat filozof z cislom 2 //vpodstate je to jedno, za to hadam body strhavat nebudu :)**

void wait\_for\_turn(int i) {

sem[i].wait();

}

void eating\_done(int i) {

sem[(i+1)%N].signal();

}

**//imho tam este trebalo spravit mutex pre vstup do KO, lebo naraz mohol papat iba jeden filozof**

semaphore sem[N] = {0, 1, 0, 0, 0};

void wait\_for\_turn(int i) {

sem[i].wait();

mutex.wait();

}

void eating\_done(int i) {  
 mutex.signal();

sem[(i+1)%N].signal();

}

**//nie je to rovnake? ved cez wait\_for\_turn() prejde vzdy iba jeden filozof a v eating\_done() da signal dalsiemu v poradi. //hmm, teraz ked tak nad tym premyslam, tak asi aj hej, to len na skuske som mal nejaku silnu teoriu, ze by nasledkom preplanovania mohli jest dvaja naraz**

**PRIKLAD 5 : I-nody.**

**Bola zadana velkost adresy*(ukazovatela?)* ( 32 b=4B), bloky mali 512 B. Bola zadana celkova velkost blokov na disku cca. 7 500 000 a volnych cca 5 900 000. 1tabuľka i-uzla sa zmestí do jedného bloku**

**a.) Ako sa zmeni pocet blokov po ulozeni suboru ktory mal cca. 4785kB.**

**b.) To iste ale ked ulozim dva maximalne subory.**

**//i-node mal myslim 8 priamych, 1 jednoducho nepriamy a 1 dvojito nepriamy ukazovatel**

Riešenie:

Rekapitulácia zadania:

* 1tabuľka i-uzla sa zmestí do jedného bloku
* tabuľka i-uzla má tieto ukazovatele/polozky: 8 priamych, 1 nepriamy a 1 dvojito-nepriamy blok

**A)**

**Krok 1:** Koľko blokov potrebujeme na tento súbor?

4785kB (veľkosť) / 0,5kB (blok) = 9570 blokov.

Koľko odkazov môžeme mať najviac v nepriamych blokoch?

512B (blok) / 4B (adresa) = 128 odkazov.

**Krok 2:** Priradíme bloky na priame adresovanie: máme 8 priamych, teda využijeme 8 blokov (zostáva nám 9570-8=9562 blokov).

**Krok 3:** Jednoduché nepriame (single indirect): máme iba jeden, ktorý vie ukázať na 128 blokov. 1\*128=128 blokov. (zostáva nam 9562-128=9434 blokov)

**Krok 4:** Dvojito nepriame (double indirect): tento mám tiež len jeden. V ňom môžeme mať maximálne 128 odkazov na bloky, ktoré sa odkazujú na iné odkazovacie bloky. Avšak nevyužijeme všetky odkazy z prvého bloku (lebo 128 (prvý blok) \* 128 (každý druhý blok) = veľa, nepotrebujeme toľko na 9434 blokov).

Otázka teda je: koľko blokov odkazov po 128 potrebujeme na pokrytie 9434 blokov? Je to 9434 / 128 = 73,703125. Zaokrúhľujeme hore, lebo potrebujeme aj ten 47. blok i keď ho celý nevyužijeme.

Teda pre dvojito nepriamu máme 1\*74\*128=9472 (áno, toto číslo je vacsie preto, lebo v poslednom sedemdesiatom stvrtom bloku nevyžívame všetkých 128 odkazov, ale iba 90 (=128\*0,703125)

**Zhrnutie úlohy A:** Takže dostávame sa k výsledku. Pre i-uzol potrebujeme 1 blok. Pre samotný súbor potrebujeme 9570 blokov. Pre nepriame bloky potrebujeme: 1 (nepriamy) + 1\*74 (dvojito nepriame) = 75 blokov.

**Odpoveď:** Po pridaní súboru veľkosti 4785 kB **sa zníži počet voľných blokov pre uloženie tabuliek i-uzlov o 1** a počet ostatných blokov o 9646 (9570 súbor + 1 nepriamy + 75 dvojito nepriame).

//trochu prehľadnejšie: súbor zaberie 9647blokov (1-Inode + 8 dátových blokov +

1 nepriamy ukazovateľ + 128 dátových blokov + 1 dvojito nepriamy ukazovateľ + 74 nepriamych ukazovateľov + 9434 dátových blokov)

//v tomto zelenom riešení nemalo by byť namiesto 9434 číslo 9472 ? Ajkeď toľko blokov nevyužijeme tak sme pre

súbor toľko alokovali

**B)**

**Krok 1:** Najskôr si potrebujeme vypočítať veľkosť najväčšieho súboru. Keďže jeden súbor má práve jeden i-uzol tak najväčší súbor je taký, ktorý má naplnené všetky odkazy v tabuľke i-uzla aj v nepriamych. Z výpočtov z úlohy A dostávame: 8 (priame) + 1\*128 (nepriame) + 1\*128\*128 (dvojito nepriame) = 16520 blokov (čo je 8260kB).

**Odpoveď:** Na dva najväčšie súbory potrebujeme dve tabuľky i-uzlov a spolu 33300 blokov ( 16520 pre samotné dáta + 1 nepriamy + 129 dvojit nepriamich = 16650 a keďže máme dva súbory tak \*2=33300 blokov). //nie este nahodov +2 ?? lebo i-nody este ?, jop +2 to ma byt

**PRIKLAD 6 :**

**Fragment kodu. Velkost stranky 512B. Kazda premmenna aj prvok pola mala velkost 1 slabiku. ‘n’ v cykle bolo 1024**

**Storage pre A,B,C,N,ZERO .. nemali intervaly po tych 512. Myslim ze A zacinal na 41 stranke.**

**Chcel iba zostavit reference string a druha podotazka bolo napisat cislo stranky a offset pre dajme tomu A[67] B[8alebo2] .... // A[660] B[770]**

**//chytacik bol v tom, ze cyklus s 1024 iteraciami sa v reference stringu rozdelil na dva cykly po 512 iteracii (R1 < 512 a R1 >= 512). jedno pole bolo ulozene v dvoch strankach (velkost pola 1024B, velkost stranky 512B) .. mne vysli cisla stranok nasledovne   
kod -> 0  
A[ ] -> 15, 16  
B[ ] -> 17, 18  
C[ ] -> 19, 20is  
konšt. -> 29**

**ak som to riesil spravne, tak vysledok mal byt  
A[660] -> stranka 16, offset 148 *(660 je viac ako 512, cize pouzijeme druhu stranku pre pole, a offset je 660-512=148)*B[770] -> stranka 18, offset 258 //vie to tu niekto napísať cele ??**

**next exam:**

**alost na ktoru cakal**

[**https://docs.google.com/document/d/19lp4OT9axdeillbZ8FLKLFGwGRzTOUS2AOhIn9nZqhQ/pub**](https://docs.google.com/document/d/19lp4OT9axdeillbZ8FLKLFGwGRzTOUS2AOhIn9nZqhQ/pub)

**That's all, good luck :)**