Operációs rendszerek pót pót ZH (VIMIA219) 2010. 05. 20.
Mérnök informatikus szak Teljes rendelkezésre álló idő: 60 perc

Név:	Neptun kód:

I. Teszt kérdések **Σ** **/ 30 pont**

Útmutató:

Jelölje egyértelműen a helyes választ! Karikázza be az I-t, ha az állítás igaz. Karikázza be a H-t, ha az állítás hamis. Karikázza be a ?-et, ha nem tudja a választ. Ha javítani akar a válaszon, akkor húzza át mind a három mezőt, és írja a sor végére a választ (Igaz/Hamis/Nem tudom).

Pontozás: Helyes válasz 1 pont, rossz válasz -1 pont. Kérdéscsoportonként a minimum pont 0 pont.

1. kérdéscsoport. A feladatok egyszerűsített állapot diagramja.

1.	Nem-preemptív (kooperatív) operációs rendszerekben egy FUTÓ feladat nem kerülhet közvetlenül FUTÁSRA KÉSZ állapotba.	I	H	?	Yield() rendszerhívással bármikor.
2.	ESEMÉNYRE VÁRAKOZIK állapotban a feladat pl. várhat egy erőforrás felszabadulására, amit egy másik feladat lefoglalt.	I	H	?	
3.	Feladatok mindig FUTÁSRA KÉSZ állapotban jönnek létre.	I	H	?	
4.	A feladatok állapotátmenetei mindig megszakításra történnek, a megszakítások lehetnek külsők (hardver) vagy belsők (szoftver).	I	H	?	Kivétel is lehet.

2. kérdéscsoport. Egyszerű ütemező algoritmusok.

5.	Az SRTF ütemező a SJF ütemező preemptív változata, ami periodikus hardver megszakításra ütemezi újra a feladatokat a CPU löket alapján	I	H	?	Az aktuálisan megmaradt CPU löket alapján ütemez újra egy új folyamat futásra késszé válása esetén.
6.	Az SRTF és a SJF ütemezők prioritásos ütemezők.	I	H	?	
7.	Az RR ütemező kiküszöböli a FIFO ütemezőben tapasztalható konvoj hatást, úgy, hogy eközben a CPU kihasználtság nem változik.	I	H	?	Csökken a CPU kihasználtság az RR adminisztratív overhead-je miatt.
8.	Az ismertetett egyszerű ütemező algoritmusok közül egyetlen egyet sem használunk a mai operációs rendszerekben, azoknak csak elméleti jelentősége van.	I	H	?	RR és FIFO széles körben alkalmazott, SJF és SRTF valós-idejű OS-ekben elérhető (konfigurálható) lehet (ott esetleg tudhatjuk a CPU löketet).

3. kérdéscsoport. Folyamatok és szálak.

9.	Folyamatok befejezésekor a folyamat gyermekeit is befejezzük, azokat ter-	I	H	?	OS függő, többnyire valamilyen alapértelmezett

	minálja az operációs rendszer.				folyamat gyermekévé válnak.
10.	A szálnak saját halom (heap) és virtuális processzor áll rendelkezésre, a verem (stack), adat és kód memórián és az erőforrásokon osztozik azokkal a szálakkal, amelyek azonos folyamat kontextusában futnak.	I	H	?	A verem saját, és a halom közös.
11.	A JAVA virtuális gép a hoszt operációs rendszer számára egy folyamat, és a JAVA szálaknak megfelelnek a hoszt operációs rendszer szálai.	I	H	?	JAVA VM függő a JAVA szálak leképzése, a one-to-one leképzés többnyire erőforrásokban bővelkedő rendszerekben kerül alkalmazásra (dekstop és szerver)
12.	Felhasználó módú szálak (green threads) alkalmazása esetén többprocesszoros rendszerben csak egy végrehajtó egységet tudunk kihasználni.	I	H	?	Az OS nem látja a felhasználói módú szálakat, még ha van is natív szál támogatása, azokat nem tudja ütemezni.

4. kérdéscsoport. H.

13.	A közös erőforrásokat egy időben egy feladat tudja helyesen használni (írás és olvasás).	I	H	?	vagy maximum megadott számú
14.	A rendszertervezőnek és programozónak a legfontosabb feladata, hogy felismerje a közös erőforrásokat, és biztosítsa azok helyes használatát.	I	H	?	
15.	Kritikus szakaszt (critical section) a hozzá tartozó erőforrásra atomi műveletként (nem megszakítható módon) kell végrehajtanunk minden egyes az erőforrást használó feladatban.	I	H	?	
16.	A megszakítás tiltásával majd engedélyezésével csak egyprocesszoros rendszerben tehető egy utasítássorozat atomivá, SMP és NUMA architektúrájú rendszerekben más megoldást kell alkalmazni.	I	H	?	

5. kérdéscsoport. A Windows operációs rendszerekkel kapcsolatos kérdések.

17.	A Windowsban csak a HAL-ban található hardver specifikus kód.	I	H	?	Található kernelben és driverekben is.
18.	A Windows képes a POSIX API rendszerhívásait használó programot futtatni.	I	H	?	Igen, Subsystem for UNIX-based Applications alrendszer segítségével
19.	Windowsban a kernel és az Executive réteg komponensei egy binárisban találhatóak.	I	H	?	Igen, mindkettő az ntoskrnl.exe-ben
20.	A Windowsban az ablakozást és grafikát megvalósító komponens azért került kernel módba, hogy megbízhatóbb legyen a rendszer.	I	H	?	Pont hogy így kevésbé megbízható, elsősorban teljesítmény okai voltak a döntésnek.

6. kérdéscsoport. A UNIX operációs rendszerekkel kapcsolatos kérdések.

21.	A kernel módba lépett felhasználói folyamatok kernel kontextusba váltanak.	I	H	?	Hamis, mivel folyamat kontextusa nem változik.
22.	A folyamatokat a felhasználó kivonhatja a rövid távú ütemezés alól.	I	H	?	Igaz (felfüggesztett módok).
23.	Minden UNIX folyamatnak van szülő folyamata kivéve az init-et.	I	H	?	Igaz
24.	Van olyan UNIX folyamatállapot, amelyben a folyamat semmilyen erőforrást (memóriát, processzort) nem foglal, de még nem szűnt meg.	I	H	?	Igaz, a zombi állapot

7. kérdéscsoport. Virtualizációval kapcsolatos kérdések.

25.	Egy hosted típusú platform virtualizációs megoldás esetén a VMM kezeli a hardver erőforrásokat.	I	H	?	A host OS kezeli.
26.	CPU virtualizáció esetén tiszta emuláció használatával jó teljesítményt lehet elérni.	I	H	?	Ez éppen a hátránya, a tiszta emuláció erőforrás igényes.
27.	Bináris átírást alkalmazó VMM esetén szükség van a vendég operációs rendszer forráskódjának módosítására.	I	H	?	Ezért bináris átírás.

8. kérdéscsoport. RAID-del kapcsolatos kérdések.

28.	RAID 1 alkalmazása esetén a diszkek tároló kapacitása összeadódik, és egy diszk meghibásodása esetén az adat elveszik.	I	H	?	RAID 0 esetén történnek ezek.
29.	A RAID 5 azért kerülendő, mert csendes hibák (silent error) és egy diszk meghibásodása esetén a RAID tömb visszaállítása nem lehetséges.	I	H	?	
30.	RAID hamis biztonságérzetet kelt, alkalmazása esetén a biztonsági másolatok készítése elkerülhetetlen az adatbiztonság biztosítására.	I	H	?	

II/1. Nagykérdés **Σ / 10 pont**

Egy UNIX rendszerben öt egyforma ütemezési adattal (p_pri = 60, p_cpu=0, nice=5) rendelkező folyamat van futásra kész állapotban t=0 időpontban. A lenti táblázatban írja le a folyamatok ütemezését! Eredményét az ütemezési algoritmus tömör leírásával, képletekkel és számítási részletekkel is igazolja! Vegye figyelembe a tanult korrekciós faktort, valamint mindhárom UNIX ütemezési tevékenységet!

	A folyamat		B folyamat		C folyamat		D folyamat		E folyamat		Áz óraütés…		
óraütés	**p_pri**	**p_cpu**	**p_pri**	**p_cpu**	**p_pri**	**p_cpu**	**p_pri**	**p_cpu**	**p_pri**	**p_cpu**	**alatt fut**	**után fut**	**óraütés**
kiindulás	60	0	60	0	60	0	60	0	60	0			kiindulás
1		1									**A**		1
2													2
3													3
9													9
10													10
20													20
21													21
90													90
91													91
99													99
100													100
101													101

Algoritmus (3 pont), képletek (4 pont), számítás (3 pont):

Megoldás:
Ütemezési algoritmus: minden lépésben prioritás ellenőrzés és a magasabb prioritású fog futni, minden 10. lépésben round-robin ütemezés az azonos prioritás(osztályú) folyamatok között, minden 100. lépésben prioritás számítás és átütemezés
Képletek: LA = FK folyamatok KF = (2 * LA) / (2 * LA + 1) p_cpu = p_cpu * KF p_pri = 50 + p_cpu / 4 + 2 * p_nice
Számítás: LA = 4 KF = 2 * 4 / (2 * 4 + 1) = 8/9 p_cpu = 20 * 8 / 9 = 18 (17 is elfogadható) p_pri = 50 + 18 / 4 + 2 * 5 = 65 (ill. 64)

Eredmények: az öt folyamat 100 lépésben ugyannyi cpu-t fog kapni, így minden adatuk megegyezik a 100. lépésre.

	A folyamat		**B folyamat**		**C folyamat**		**D folyamat**		**E folyamat**		**Az óraütés…**		
óraütés	**p_pri**	**p_cpu**	**p_pri**	**p_cpu**	**p_pri**	**p_cpu**	**p_pri**	**p_cpu**	**p_pri**	**p_cpu**	**alatt fut**	**után fut**	**óraütés**
kiindulás	60	0	60	0	60	0	60	0	60	0			kiindulás
1	60	1	60	0	60	0	60	0	60	0	**A**	**A**	1
2	60	2	60	0	60	0	60	0	60	0	**A**	**A**	2
3	60	3	60	0	60	0	60	0	60	0	**A**	**A**	3
9	60	9	60	0	60	0	60	0	60	0	**A**	**A**	9
10	**60**	10	**60**	0	**60**	0	**60**	0	**60**	0	**A**	**B**	10
20	60	10	60	10	60	0	60	0	60	0	**B**	**C**	20
21	60	10	60	10	60	1	60	0	60	0	**C**	**C**	21
90	60	20	60	20	60	20	60	20	60	10	**D**	**E**	90
91	60	20	60	20	60	20	60	20	60	11	**E**	**E**	91
99	60	20	60	20	60	20	60	20	60	19	**E**	**E**	99
100	**65**	18	**65**	18	**65**	18	**65**	18	**65**	18	**E**	**A**	100
101	65	19	65	18	65	18	65	18	65	18	**A**	**A**	101

Pontozás: minden helyes képlet 1 pont (az LA megnevezése 1 pont), KF, p_cpu és p_pri számítása 1-1 pont, az algoritmus helyes leírása a táblázatban jól megadva 3 pont.

II/2. Nagykérdés **Σ** **/ 10 pont**

Egy igény szerinti lapozást használó rendszerben 3 vagy 4 fizikai memórialap áll egy folyamat rendelkezésére. A futás folyamán sorban a következő virtuális lapokra történik hivatkozás:

0, 1, 3, 2, 3, 0, 1, 4, 3, 2, 1, 2, 3, 4

Hány laphiba következik be a rendszerben a következő algoritmusok esetén, ha kezdetben a 3 vagy 4 lap üres?

- legrégebbi lap (FIFO) algoritmus alkalmazásánál 3 vagy 4 fizikai memória lap esetén, (1.5-1.5 pont)
- legrégebben nem használt (Least recently used, LRU).algoritmus alkalmazásánál 3 és 4 fizikai memória lap esetén (2.5-2.5 pont).

Röviden magyarázza meg az eredményeket! (2 pont)
A megoldásban mutassa be, hogyan jutott az eredményre, csak a laphibák számának megadását nem fogadjuk el!

FIFO 3 fizikai memória kerettel:

Lapok	0	1	3	2	3	0	1	4	3	2	1	2	3	4
FIFO0	0	1	3	2	2	0	1	4	3	2	1	1	1	4
FIFO1		0	1	3	3	2	0	1	4	3	2	2	2	1
FIFO2			0	1	1	3	2	0	1	4	3	3	3	2
Laphiba	I	I	I	I		I	I	I	I	I	I			I

Laphibák száma: 11

FIFO 4 fizikai memória kerettel:

Lapok	0	1	3	2	3	0	1	4	3	2	1	2	3	4
FIFO0	0	1	3	2	2	2	2	4	4	4	4	4	4	4
FIFO1		0	1	3	3	3	3	2	2	2	2	2	2	2
FIFO2			0	1	1	1	1	3	3	3	3	3	3	3
FIFO3				0	0	0	0	1	1	1	1	1	1	1
Laphiba	I	I	I	I				I						

Laphibák száma: 5

LRU 3 fizikai memória kerettel (alul a lap legutolsó használata óta eltelt lépések száma):

Lapok	0	1	3	2	3	0	1	4	3	2	1	2	3	4
LRU0	0 1	0 2	0 3	2 1	2 2	2 3	1 1	1 2	1 3	2 1	2 2	2 1	2 2	2 3
LRU1		1 1	1 2	1 3	1 4	0 1	0 2	0 3	3 1	3 2	3 3	3 4	3 1	3 2
LRU2			3 1	3 2	3 1	3 2	3 3	4 1	4 2	4 3	1 1	1 2	1 3	4 1
Laphiba	I	I	I	I		I	I	I	I	I	I			I

Laphibák száma: 11

LRU 4 fizikai memória kerettel (alul a lap legutolsó használata óta eltelt lépések száma):

Lapok	0	1	3	2	3	0	1	4	3	2	1	2	3	4
LRU0	0 1	0 2	0 3	0 4	0 5	0 1	0 2	0 3	0 4	2 1	2 2	2 1	2 2	2 3
LRU1		1 1	1 2	1 3	1 4	1 5	1 1	1 2	1 3	1 4	1 1	1 2	1 3	1 4
LRU2			3 1	3 2	3 1	3 2	3 3	3 4	3 1	3 2	3 3	3 4	3 1	3 2
LRU3				2 1	2 2	2 3	2 4	4 1	4 2	4 3	4 4	4 5	4 6	4 1
Laphiba	I	I	I	I				I		I				

Laphibák száma: 6

A FIFO esetén nem jelentkezik a Béládi anomália, a nagyobb számú fizikai memória keret jobb eredményt hoz.

A lapsorozat első része a 0,1,2,3, míg a második része az 1,2,3,4 memória lapokra lokális, ennek megfelelően a 4 fizikai lappal dolgozó megoldások szignifikánsabban jobban működnek, mint a 3 fizikai lapot használó megoldások.

Az eredmények szerint a FIFO jobb, de statisztikailag az LRU jobban működik.