2022. tavasz “Operációs rendszerek B” vizsga kérdések

A BSc képzés tesztjei linken nagyon hasonló kérdések megtalálhatóak, csak ott részletesebb az anyag, ezáltal több kérdés is van. Biztos vagyok benne, hogy ez nem az összes kérdés, csak ennyit sikerült összegyűjteni.
Frissítve: 2022-06-06.

Igaz-Hamis:

A biztonsági mentés visszaállításához előbb telepítenünk kell az operációs rendszert.
Hamis

A blokkgyorsítótár a fizikai memóriát használja a diszkműveletek gyorsítására.
Igaz

A cserehely a fizikai memória egy elszeparált része, ahol a nem futó taszkok adatait tároljuk.
Hamis

A folyamatokon belül csak egy verem lehet.
Hamis

A fork() Unix rendszerhívás betölt és elindít egy új programot.
Hamis

A FreeBSD egy Linux disztribúció.
Hamis

A futási szint (runlevel) meghatározza a Unix rendszerekben futó taszkok prioritását.
Hamis

A hálózati kommunikáció csak fizikai hálózati kapcsolat megléte esetén alkalmazható. (Nem biztos a válasz)
Hamis

A karakteres felületen működő parancsértelmezők jellemzően programozható eszközök.
Igaz

A keret- és a laptáblák száma megegyezik.
Hamis

A kernel adatstruktúrák egy része a fájlrendszeri interfészen keresztül hozzáférhető Linux alatt.
Igaz

A kernel az első program, amit a háttértárról betöltve a processzor futtatni kezd.
Hamis

A kiéheztetés statikus prioritást alkalmazó prioritásos ütemezőkben nem kerülhető el.
Igaz

A konvoj-hatás például a legrövidebb hátralevő löketidejű (SRTF) algoritmussal megszüntethető.
Igaz

A kooperatív ütemezők nem veszik el a futó taszktól a processzort.
Igaz

A körforgó (RR) ütemező használata optimális átlagos várakozási időt eredményez.
Hamis

A körforgó (RR) ütemező kooperatív és elkerüli a kiéheztetést.
Hamis

A körgorgó (RR) ütemezés esetén a konvoj-hatás nem jelentkezik, mivel az ütemező a taszkokat csak adott ideig futtatja, utána átütemezéssel megszakítja a futásukat.
Igaz

A lapcsere során egy lap behozásakor jellemzően nem kell keretet felszabadítani, mivel szabad keretek létezéséről egy speciális folyamat gondoskodik.
Igaz

A laphibák száma nagyon sok párhuzamosan működő taszk esetén lineárisan függ a multiprogramozás fokától.
Hamis

A laplopó (page daemon) valamilyen lapcsere-algoritmust futtat.
Igaz

A laplopó (page daemon) valamilyen lapozási (lapbehozási) stratégiát alkalmaz.
Hamis

A lapok tárba fagyasztásának alapvető célja a firssen behozott lapokhoz tartozó keretek felszabadításának megakadályozása.
Igaz

A leggyorsabb 1TiB méretű adattároló rendszer a merevlemezes meghajtó (HDD).
Hamis

A legrégebben várakozó (FCFS) ütemezőnél akkor is jelentkezhet a konvoj-hatás, ha csak I/O intenzív taszkok vannak a rendszerben.
Hamis

A legrövidebb hátralevő löketidejű (SRTF) ütemező preemptív.
Igaz

A Linux felhasználói módban többszintű ütemezőt használ.
Igaz

A Linux kernel átütemezési pontjainak alkalmazása javítja a válaszidőt, mivel így a kernel módban futó taszkok bármikor megszakíthatók.
Hamis

A mai Windows operációs rendszerekben a FAT32 a legelterjedtebb fájlrendszer.
Hamis

A memória-intenzív taszkok nagy memóriafoglalás esetén CPU-intenzívvé válnak.
Hamis

A memóriaalapú permanens tárolók (SSD-k) élettartama jellemzően 5 év körüli.
Hamis

A Microsoft RDP egy elterjedt kijelzőszerver protokoll.
Igaz

A modern mikrokernelek (pl. L4) nagyon lassú üzenetalapú kommunikációval működnek.
Igaz

A multiprogramozott operációs rendszer abban különbözik más rendszerektől, hogy többféle programozási nyelvet támogat.
Hamis

A OS kernelek minden része (eljárása) védett módban működik.
Hamis

A PRAM modell nem engedi meg a közös memória konkurens írását két (vagy több) taszk által, ezért ilyen esetekben is garantálja a programok helyes működését
Hamis

A PRAM modell írás-olvasás ütközésnél mindig először az írás műveletet hajtja végre, hogy az olvasás már az új értékkel térhessen vissza.
Hamis

A RAID0 (stripe) két diszk esetén két példányban tárol minden adatot.
Hamis

A RAID0 általában gyorsabb a RAID1-nél. de a RAID1 megbízhatóbb.
Igaz

A RAID5 egy kijelölt paritástárolót (partíciót, diszket) használ a redundáns tárolás céljára
Hamis

A rendszerhívások jellemzően megszakítással járnak együtt.
Igaz

A rendszerkönyvtárak olyan eljárásokat tartalmaznak, amelyek sokféle feladatban előfordulnak, így nem kell minden programban külön-külön megvalósítanunk azokat, hanem támaszkodhatunk egy közös implementációra.

Igaz

A rendszerprogramok védett módban futnak.

Hamis

A statikus többszintű ütemezőkben nem jelentkezhet konvoj-hatás, hiszen a globális ütemező preemptív

Hamis

A szál a taszk egy olyan megvalósítása, amely önálló memóriaterülettel rendelkezik.

Hamis

A szál egy szekvenciális működésű taszk, amely egy folyamaton belül más szálakkal közös halmot (heap-et) használ.

Igaz

A számítógépeken futó taszkok többsége I/O-intenzív

Hamis

A taszkok löketidejét a gyakorlatban működő ütemezők előre ismerik.

Hamis

A taszkok már elinudláskor lefoglalják a számukra szükséges teljes memóriatartományt.

Hamis

A távoli eljáráshívás (RPC) számítógépek között is működik, nem csak egy gépen belül.

Igaz

A többszintű visszacsatolt sorok (MFQ) ütemező bemeneti szintválasztó algoritumsa a taszkok prioritása alapján határozza meg a kezdeti szintjüket.

Hamis

A többszintű visszacsatolt sorok (MFQ) ütemező lefelé lépteti a taszkokat a szintek között, ha azok az adott szinten kihasználják a rendelkezésükre álló CPU-időt (pl. a RR időszeletet).

Igaz

A többszintű visszacsatolt sorok (MFQ) ütemező preemptív.

Igaz

A Unix cron egy középtávú ütemező.

Hamis

A Unix operációs rendszer első változata az AT&T Bell Lab kommerciális termékeként jelent meg, amelyet számos cég és egyetem vásárolt meg.

Hamis

A valósidejű működés alapvető célja az, hogy a felhasználók valós időben végezhessék a rendszeren a feladataikat.

Hamis

A virtuális és fizikai memóriacímek futásidejű transzformációja alapvetően szoftveres úton történik.

Hamis

A válaszidő mindig kisebb, mint a körülfordulási idő.

Igaz

A várakozási idő a taszk futásra kész állapotban eltöltött ideje.

Hamis

A Windows egyes beágyazott (kevés erőforrással rendelkező) rendszerben is működik

Igaz

A Windows nem rendelkezik programozható, karakteres parancsértelmezővel.

Hamis

A Winlogon előbb fut, mint az SMSS (munkamenet-kezelő) a Windows-on.

Hamis

Az alkalmazások által használt összes memória mérete sosem haladhatja meg a fizikai memória méretét

Hamis

Az Apache webszerver szálalapú változata nagyobb teljesítményre (kérés / mp) képes, mint a folyamatalapú.

Igaz

Az időosztásos operációs rendszerek egyben multiprogramozott rendszerek is.

Igaz

Az IEEE POSIX egy szabvány, amely előírja a kernel belső felépítését.

Hamis

Az iSCSI (internet SCSI) fájlokon elvégzett műveletek hálózati átviteli protokollja.

Hamis

Az operációs rendszer kernele felügyeli a felhasználói módban futó taszkok működését.

Igaz

Az SJF ütemező képes kezelni a konvoj-hatást, de nem a kiéheztetést.

Igaz

Az SRTF ütemező olyan prioritásos ütemezőnek is tekinthető, ahol a prioritás egyenlő a taszk hátralévő löketidejével, és a legkisebb értékkel rendelkező taszk fog futni.

Igaz

Az óra és az újabb esély lapcsere ugyanazon múltbéli adatokra támaszkodva működik.;???
Az ütemező a várakozó állapotú taszkok közül választja ki a következő futó taszkot.
Hamis

Az ütemező átbocsájtó képessége az egységnyi időszelet alatt átütemezett taszkok száma.
Hamis

Az üzenetsor egy indirekt kommunikációs megoldás
Igaz

Az „óra” lapcsere algoritmus valójában az „újabb esély” algoritmus hatékonyabb megvalósítása.
Igaz

Egy mai általános célú operációs rendszer kernelének forráskódja több százezer programsor.
Hamis

Egy operációs rendszer forráskódja lehet néhány tízezer programsor, de akár sok millió is.
Igaz

Egy operációs rendszer nem lehet egyszerre monolitikus és moduláris felépítésű.
Hamis

Egy operációs rendszerben nem lehet több működő taszk, mint ahány végrehajtó egység van.
Hamis

Egy párhuzamos végrehajtást (több konkurens taszk együttműködését) igénylő feladat egyetlen folyamaton belül is megvalósítható.
Igaz

Egy statikus többszintű ütemező kimeneti szintválasztó algoritmusa lehet RR ütemező, ha azt az adott alkalmazási környezet megkívánja
Igaz

Egy taszk futásra kész állapotból várakozó állapotba is átkerülhet.
Hamis

Egy taszk várakozó állapotból futó állapotba is átkerülhet.
Hamis

Egy általános célú operációs rendszerben jellemzőek 1-2 kontextusváltás történik másodpercenként
Hamis

Ha egy felhasználói program kernel módba vált (pl. rendszerhívással), a CPU-utasításkészlete akkor is korlátozott marad, hogy ne okozzon gondot a kernelben.
Hamis

Ha egy memória-intenzív taszkokat futtató rendszerben alacsony a CPU-kihasználtság, akkor nincs elegendő memória a taszkok számára.
Igaz

Ha letörlök egy szimbolikus link által hivatkozott fájlrendszeri bejegyzést, akkor az adat elvész. Ha magát a szimbolikus linket törlöm le, akkor nem.

Igaz

Ha növeljük egy rendszerben a fizikai memória méretét, akkor mindig csökkenni fog a laphibák száma, hiszen egyszerre több lapot tarthatunk bent a memóriában.

Hamis

Klinens platformon a Windows részesedése 90% feletti.

Hamis

Két, egy folyamaton belüli szál azonos virtuális címen jellemzően ugyanazt látja, de van a virtuális címtartományuknak olyan része, amely biztosan különböző adatokat tartalmaz.

Igaz

Különböző folyamatokban található szálak is kommunikálhatnak egymással PRAM modell szerint az OS szolgáltatásait felhasználva.

Igaz

Lehetséges várakozásmentes I/O műveletek alkalmazása a programjainkban.

Igaz

Minden rendszerhívás védett módban hajtódik végre.

Hamis

Van olyan operációs rendszer, amely részben hangutasításokkal is kezelhető.

Igaz

Feleletválasztós:

- A **CPU-löket** a taszk processzoron végrehajtott utasításainak sorozata.
- Mire szolgál a System V üzenetsorok üzeneteiben a típusjelzés? **Az üzenetek szűrése**
- Mondjon egy szabványos (!) eszközt (teljes megnevezés), amely a PRAM modell szerinti kommunikációt teszi lehetővé! **POSIX Shared Memory (?)**
- Egy fájl megnyitásakor a kernel egy **???** ad vissza az alkalmazás számára, amely a további műveletek során a felhasználható a megnyitott fájl azonosítására.
- Mekkora az 5 darab 1TiB méretű diszkből összeállított RAID6 tárolórendszer teljes hasznos kapacitása? **3 TiB**
- A kiéheztetés jelensége jelentkezhet **SJF** ütemező esetén és ott **sehogyan sem** kezelhető.
- A taszk egy **végrehajtás** alatt álló program.

- A taszkok állapotváltozásai alapvetően **megszakítások** miatt következnek be.
- A több feladatot egymás után megoldó **kötegelt (batch)** rendszerek után a feladatokat párhuzamosan futtató **multiprogramozott** OS-típusok jelentek meg.
- A FAT32 maximális fájlmérete **4 GiB**.
- A **I/O-löket** a taszk működésének azon fázisa, amikor I/O művelet végrehajtására vár.
- A folyamatnak saját **memóriatartománya**, a szálnak pedig saját **verme** van.
- A(z) **átbocsájtó-képesség** meghatározza az időegység alatt elvégzett feladatok számát.
- A konvoj-hatás nem jelentkezhet **legrövidebb hátralevő löketidejű (SRTF)** és **körforgó (RR)** ütemező esetén.
- A(z) **öregítés** a futásra kész taszk prioritásának növelése az ebben az állapotában eltöltött idejével arányosan.
- A(z) **kiéheztetés** az a jelenség, amikor prioritásos ütemezés esetén egy taszkot folyamatosan megelőznek nála magasabb prioritásúak, ezért nem jut processzorhoz.
- A(z) **FCFS** ütemező saját döntése alapján nem helyez át futásra kész állapotba taszkokat. (Ne egy konkrét ütemezőt nevezzen meg, hanem egy kategóriát!)
- A Béládi-féle anomália azt jelenti, hogy a fizikai memóriakeretek számának **növelése** időnként a laphibák gyakoriságának **növekedését** vonja maga után.
- A UNIX mmap(), illetve a Windows CreateFileMapping() rendszerhívások alapvető célja, hogy a fájlok tartalma fájlműveletek helyett **memória** műveletekkel legyen elérhető.
- A Unix flock() egy **ajánlott** zárolási forma.
- Mekkora az 5 darab 1TiB méretű diszkből összeállított RAID5 tárolórendszer teljes hasznos tárolókapacitása? **4 TiB**
- A logikai kötetkezelés (LVM) egyik alapvető célja, hogy növelje a maximális **tárolórendszer-méretet**.
- A legrövidebb hátralevő löketidejű ütemező (SRTF) végrehajthat "**Fut → Futásra kész"** állapotátmenetet egy taszkon, míg például a legrégebben várakozó (FCFS) nem. "**Fut → Várakozik"** állapotátmenetet egyetlen ütemező sem hajt végre taszkokon.
- Állítsa párba az alábbi feladattípusokat és a rájuk leginkább jellemző feladatjelleget!
  Önvezető autó irányítása: **valósidejű**
  Műveletek nagy adatbázisokon: **memóriaintenzív**
  Kisvállalati fájlszerver: **I/O intenzív**
  Számítógépes játékok: **cpu intenzív**

- Rendezze sorrendbsra kész taszk prioritásának növelése az ebben az állapotában eltöltött idejével arányosan.e a feladatkezeléssel kapcsolatban megfogalmazott alábbi időjellemzőket hosszuk szerint!
  A legrövidebb: **válaszidő**
  Rövidebb: **várakozási idő**
  Leghosszabb: **körülfordulási idő**

- Állítsa párba az alábbi OS-típusokat és jellemzőiket!
  Leglényegesebb tulajdonsága, hogy több feladatot futtat egyszerre: **Multiprogramozott**
  Törekszik azonos CPU-időt biztosítani a feladatok végrehajtása számára. **Időosztásos**
  Több feladatot adhatunk a rendszernek, de egyszerre csak egyet fog guttatni. **Kötegelt**

- Milyen ütemezőket rendelne egy többszintű ütemezőben az alábbi feladattípusokhoz a lenti készletből?
  Valósidejű: **???**
  I/O intenzív: **???**
  CPU-intenzív: **???**

- Hol tároljuk az alábbi adatokat a virtuális memóriakezelésben?
  Taszkok memóriatartományainak leirói: **laptábla**
  A cserehely adatblokkjainak leírói: **diszk blokk (?)**
  A fizikai memóriatartományok leírói: **???**

- A virtuális tárkezelés során milyen hiba generálódik, amikor…
  a taszk olyan címet használ, amely számára nem engedélyezett: **védelmi hiba**
  a taszk egy igény szerint kitöltendő lapot először használ: **fill-on-demand hiba (?)**
  a taszk olyan lapra hivatkozik, amelyik cserehelyen van: **laphiba**

- Melyik memóriakezelési tevékenység tud jobban jósolni (ha lehet ilyen különbséget tenni köztük)?
  Kevesebb információt tud a lapokról, ezért kevésbé jósol jól: **Lapozási stratégia**
  Több információ áll a lapokról a rendelkezésére, ezért jobban jósol: **Lapcsere algoritmus**

Ütemezők (levezetni):

- FCFS
- RR
- SJF
- SRTF
- MFQ

Lapcsere algoritmusok (tudni):

- FIFO
- Újabb esély
- Óra lapcsere
- LRU
- LFU
- NRU

Programkódok:

- Forkolás:

```
if ((res = fork()) == 0) {   // elágazás a visszatérési értékkel
                             // gyerek ága
  exec(...);                 // betöltünk egy másik programkódot
                             // ha visszatér, exec hiba történt
} else if ( res < 0 ) {      // szülő ága, hibaellenőrzés
                             // ide jön a fork() hiba kezelése
}
// res = CHILD_PID (>0), szülő kódja fut tovább
```

- Hálózati szerver:

```
sfd1 = socket()
bind(sfd1)
listen(sfd1)
while
  sfd2 = accept(sfd1)
    fork()
  szülő:
    vissza a ciklusba
  gyerek:
    recv(sfd2)
    send(sfd2)
    close(sfd2)
    exit()
```