HARASZIN PÉTER

Operációs rendszerek – Kidolgozott beugrókérdések (ABC-sorrendben)

Köszönet mindazoknak, akik a BME-s Wiki-n hozzájárultak segítségükkel a beugrók kidolgozásához!

2011.06.18.

Operációs rendszerek tárgy beugróinak kidolgozása. TODO-val jelöltem a még átnézendő, rendbe szedendő vagy hiányos válaszokat. Ezek kiegészítésében segítsetek, ha tudtok! Ebben az esetben és hibák esetén is kérlek javítsátok a vonatkozó Wikis lapot, illetve ezt a dokumentumot is! Köszönjük!

32 bites kliens Windows operációs rendszer maximum mennyi fizikai memóriát kezelhet, és miért?
PAE (Physical Address Extension) támogatás, ezzel lehet 32 bites címbuszú CPU-val is 64 GB memóriát kezelni a maximális 4 GB helyett)

32 bites szerver Windows képes-e 4 GB-nál több fizikai memória kezelésére? Válaszát indokolja!
Igen, PAE (Physical Address Extension) támogatás segítségével.

32 bites Windows használata esetén egy felhasználói folyamat maximum mekkora virtuális címteret használhat?
Alapból 2GB felhasználói módú címterületet használhat, ez a /3GB kapcsolóval 3 GB felhasználói címterületre bővíthető.

32 bites Windows szerver operációs rendszerek képesek-e 4 GB-nál több fizikai memória kezelésére? Indokolja válaszát!
Igen, PAE (Physical Address Extension) támogatás segítségével.

32 bites x86-os (Windows) esetén mekkora a felhasználói és a rendszer mód címtartomány mérete?
Alapból 2GB felhasználói módú és 2GB kernel címterület van, ezt a /3GB kapcsolóval 3GB felhasználói és 1GB kernelre lehet módosítani.

A uC/OS II-ben hány taszk tartózkodhat egy prioritási szinten és miért?
1, mert így gyorsan eldönthető, hogy melyik a legmagasabb futásra kész taszk.

A Windows OS grafikus komponensének mik az előnyei, hátrányai?
A grafikus komponens kernel módban fut. Emiatt a hibái az egész rendszert magával ránthatják, viszont gyorsabb, mert kevesebb kontextusváltás kell a rajzoláshoz.

Adja meg a holtpont definícióját!
Egy rendszer folyamatainak egy H halmaza holtponton van, ha a H halmazba tartozó valamennyi folyamat olyan eseményre vár, amelyet csak egy másik, H halmazba tartozó folyamat tudna előidézni.

Adja meg a hosszútávú ütemezés célját!
→A hosszútávú ütemező feladata az elindított feladatok rendszerbe, illetve a "futásra kész" várakozási sorba való beengedését szabályozni. Igyekszik a CPU-t és a perifériákat terhelő folyamatokat egyensúlyban tartani. Batch rendszerekre jellemző; a PC-k oprendszere általában azonnal indítja a folyamatokat, mikor azt a felhasználó kéri.
→Sokkal több feladatunk van, mint amennyit párhuzamosan eltudunk látni hatékonyan. Az ütemezés ezért percenként vagy gyakrabban fut és ismernie kell a feladat által okozott terhelést. Többnyire maximum időzíteni lehet a feladatokat.

Adja meg a kemény valós idejű (hard real-time) rendszer definícióját!
A valószínűség = 1. Ha nem válaszol időben, a válasz rossz. A határidők nem teljesítésének katasztrofális következményei vannak. A rendszer NEM késhet!

Adja meg a középtávú ütemezés célját!
Swapping, azaz a program a fizikai memória és a háttértár közti mozgatása.

Adja meg a kritikus szakasz (critical section) fogalom definícióját!
→A magában szekvenciális feladatok azon kódrészletei, amely során a kölcsönös kizárást egy bizonyos közös erőforrásra biztosítjuk.
→A kritikus szakasz a kérdéses közös erőforráshoz tartozik.
→A kritikus szakaszt a hozzá tartozó erőforrásra atomi műveletként (nem megszakítható módon) kell végrehajtanunk.

Adja meg a System V üzenetsorok főbb jellemzőit (tömör felsorolást kérünk)!
→diszkrét, tipizált üzenetek
→nincs címzés, üzenetszórás

Adja meg a tradicionális UNIX ütemező három legjellemzőbb tulajdonságát!
→nem preemptív kernel módban (a kernel módot végrehajtó folyamatot (pl. rendszerhívás, megszakítás-kezelés) nem lehet kényszeríteni, hogy a CPU használatáról lemondjon egy nagyobb prioritású folyamat javára)
→ újraütemezés csak akkor következik be, ha egy folyamat önként lemond a CPU-ról és sleep rendszerhívást hajt végre, vagy a folyamat kernel módból visszatér user módba
→Nem méretezhető megfelelően. Az algoritmus nem képes rugalmasan alkalmazkodni a folyamatok számának növekedése esetén. A korrekciós faktor nem elég hatékony eszköz.
→A CPU-t adott esetben nem lehet „kiosztani" adott folyamat számára.
→Nem garantálható fix válaszidő. Nagy rendszerterhelés esetén a válaszidő megnőhet. A UNIX ütemezés épp ezért nem alkalmazható real-time rendszerekben.
→A kernel nem preemptív, ezért az egész rendszert feltarthatja.

→A felhasználó nem tudja megfelelő módon befolyásolni folyamatai prioritását – a nice szám nem megfelelő eszköz erre a célra.

Adja meg a vergődés (trashing) definícióját!

A gyakori laphibák okozta teljesítménycsökkenést vergődésnek (thrashing) nevezzük. Az ellene való védekezés a munkahalmaz méretének jó megválasztása. Célszerű egy folyamatnak annyi lapot adni, amennyi szükséges az egyensúlyhoz, azaz ahány lapra hivatkozik a laphiba kiszolgálás ideje alatt (ugyanakkor nem sokkal többet, mert ekkor leromlik a multiprogramozás foka).

Adja meg az átbocsátó képesség definícióját és mértékegységét!

→ (throughput) Mértékegység: munka/s, vagy 1/s

→Adott időegység alatt elvégzett feladatok száma. $\frac{\sum \text{elvégzett munkák}}{\text{idő}}$

→A rendszerfeladatokat nem számoljuk.

Adja meg az M (M pozitív egész szám) diszket tartalmazó RAID5 tömb tulajdonságait! (hibatűrés és sebesség)

→RAID 5 (block interleaved distributed paritity).
→Több diszk redundáns és párhuzamos használata
→Adat és paritás elosztása *N+1* diszkre.

- ●A sebesség tekintetében közel áll az *N* diszket használó RAID 0-hoz (HW támogatás esetén).
- ●1 diszk meghibásodása esetén az adat elérhető.
- ●2 vagy több diszk meghibásodása esetén az adat elveszik.
- ●Az adat nem feltétlenül állítható helyre. (Csendes/néma hibák (silent error). A 2. meghibásodás észlelése a tömb újraépítése során)

Adja meg rövid ütemezés célját!

A futásra kész sorból választ egy futó állapotba átmenő feladatot.

Az éppen futó taszkot megszakítja egy IT. Preemptív OS esetén mindig a megszakított taszk fogja-e visszakapni a futási jogot? Miért?

→Nem feltétlenül; például a preemptálás maga is úgy működik, hogy egy időzítő a szál quantumjának lejártakor megszakítást generál; ilyenkor értelemszerűen az ütemező általában nem ugyanazt a folyamatot választja ki futásra.
→Másik: Nem, mert akitől elvették a futás jogát az futásra kész állapotba fog kerülni és az ütemező dönti el, hogy melyik folyamat fogja megint megkapni a futást.

Az operációs rendszer milyen általános eljárásokat használhat a holtpont kezelésére?

→strucc algoritmus (nem vesz róla tudomást)
→holtpont feloldása - melyik holtpontban érintett folyamatot számoljuk fel?
→menthető állapotú erőforrások elvétele,
→minél kevesebb folyamat felszámolása,
→folyamatok prioritása,
→már elvégzett munka,
→folyamatok visszaállíthatóságának biztosítása
→holtpont megelőzése

Definiálja a "folyamat" (process) fogalmát!

→végrehajtás alatt álló program (program maga a végrehajtható kód), amely folyamat virtuális címterébe van leképezve
→folyamat egy szála az, ami éppen fut egy CPU-n, és nem maga a folyamat
→minden folyamathoz tartozik legalább egy szál, ami elinduláskor elkezdi futtatni a program main metódusát
→privát virtuális címtér (virtuális memóriacímek készlete, amiket a folyamat használhat)
→tartozik hozzá egy egyedi folyamatazonosító (process ID)
→saját kód,adat, halom, verem
→rendszererőforrások listája, melyekhez a folyamat összes szála hozzáfér
→a folyamat virtuálisan összefüggő memóriát lát (virtuális memória) (valójában az összefüggő memóriaterület ritka)
→háttértárolóra is kiírható (swapping)
→A folyamat által látott logikai címtartomány, és a ténylegesen használt fizikai címtartományok teljesen elkülönülnek
→A folyamatok nem férnek hozzá egymás lapjaihoz
→ Folyamatok megoszthatnak memóriaterületeket olvasás- vagy akár írás- és olvasás-hozzáféréssel (Az ilyen memória területek több folyamat virtuális címtartományába vannak belapozva)

Definiálja a "szál" (thread) fogalmát!

→párhuzamos végrehajtású, közös memóriát használó programrészek a folyamaton belül (egy program végrehajtása több szálon futhat). A szálaknak saját logikai processzoruk van, azonban memóriáik nincsenek elkülönítve, közös logikai memóriát

használnak, azaz a kódon és a változókon osztoznak, vagyis egymás adatait olvashatják és írhatják. Emiatt az operációs rendszer lényegesen gyorsabban tud végrehajtani egy átkapcsolást a szálak között, mint a folyamatok között.
→A folyamat egy szála az, ami éppen fut egy CPU-n (ami ütemezésre kerül), és nem maga a folyamat.
→Minden folyamathoz tartozik legalább egy szál, ami elinduláskor elkezdi futtatni a program main metódusát (szál nélkül a folyamat programja nem futhat).
→szálak még véletlenül sem hivatkozhatnak más folyamatok címterére, hacsak a másik folyamat nem teszi elérhetővé privát virtuális címterének egy részét megosztott memóriaszakaszként (file mapping object a Windows API-ban), vagy – Windows-nál – hacsak egyik folyamatnak nincs joga megnyitni más folyamatot, hogy olyan folyamatok közti memóriafüggvényeket használjon, mint a ReadProcessMemory vagy WriteProcessMemory

Definiálja a holtpont (deadlock) fogalmát!

Egy rendszer feladatainak egy *H* részhalmaza holtponton van, ha a *H* halmazba tartozó valamennyi feladat olyan eseményre vár, amelyet csak egy másik, *H* halmazbeli feladat tudna előállítani.

Definiálja a hosszútávú ütemezés fogalmát!

A hosszútávú ütemező feladata az elindított feladatok rendszerbe, illetve a "futásra kész" várakozási sorba való beengedését szabályozni. Igyekszik a CPU-t és a perifériákat terhelő folyamatokat egyensúlyban tartani. Batch rendszerekre jellemző; a PC-k oprendszere általában azonnal indítja a folyamatokat, mikor azt a felhasználó kéri.
→feladata: A háttértáron várakozó feladatok közül kiválasztja azt, amelyiket el kell indítani.

Egy futó taszkra IT érkezik. Preemptív OS esetén az interrupt után mindenképpen 'ide' térünk vissza?

Nem feltétlenül; például a preemptálás maga is úgy működik, hogy egy időzítő a szál quantumjának lejártakor megszakítást generál; ilyenkor értelemszerűen az ütemező általában nem ugyanazt a folyamatot választja ki futásra.

Egy prioritási szinten hány szál futtatását teszi lehetővé a uCOS/II, miért?

minden szál egy fix prioritási szinthez van kötve, egyszerűségi és gyorsasági okokból ???
→TODO

Elosztott rendszerekben milyen konzisztencia kérdésekkel kell foglalkozni?

→frissítés konzisztencia
→másolat konzisztencia
→cache konzisztencia
→hiba konzisztencia
→óra konzisztencia
→felhasználói interfész konzisztencia

Engedélyezési rendszerekben mit tartalmaz egy művelet kontextusa? (felhasználó- és jogosultságkezelés)

A műveletek kontextusa tartalmazza a szereplő azonosítóját, a célobjektumot és az elvégzendő művelet fajtáját.

Executive (Windows)

Ez a réteg tartalmazza az NTDLL.DLL által definiált függvények hívásainak megvalósítását, valamint a rendszer külső objektumai közti kommunikáció. Legfontosabb szolgáltató funkciója a lokális eljárás hívás - LPC (Local Procedura Call) megvalósítása.

Hasonlítsa össze a közös memórián illetve az üzenetváltáson alapuló folyamatok közti együttműködést!

Közös memórián keresztül történő adatcsere esetén az együttműködő folyamatok mindegyike saját címtartományában lát egy közös memóriát. A közös memória elérését valamilyen adatátviteli rendszer teszi lehetővé. Üzenetváltásos adatcsere esetén a folyamatoknak nincs közös memóriája. Az adatátviteli rendszer most a logikai processzorokat kapcsolja össze. Rajta keresztül a folyamatok üzeneteket tudnak küldeni, illetve fogadni. Az üzenetküldésre a folyamatok logikai processzorainak utasításkészletében megfelelő utasítások állnak rendelkezésre. Ezek a Küld (Send) és a Fogad (Receive) műveletek.

Hasonlítsa össze az általános célú (asztali) és a beágyazott operációs rendszereket az indulás szempontjából!

A beágyazottnál először indul az alkalmazás, és az indítja az operációs rendszert, az asztalinál az operációs rendszer indítja az alkalmazásokat.

Hasonlítsa össze két azonos diszkből álló RAID0 és RAID1 tömb tulajdonságait! Hogyan alakul a hozzáférési idő, az adatátviteli sebesség, és a megbízhatóság egyetlen diszkhez képest a két esetben?

RAID 0-1 szabványok általában SW implementációval és kevés (2db) diszkkel
→RAID 0 (striped disks):
- ●Több diszk párhuzamos használata;
- ●file részei N diszkre kerülnek;
- ●Az egyes részek egymástól függetlenül elérhetők

●A diszkek tárolókapacitása összeadódik
●N azonos diszk esetén a RAID 0 virtuális diszk olvasásai és írási adatátviteli sebessége maximum N-szeres közelébe nő.
●A hozzáférési idő közel eléri egy diszk hozzáférési idejét.
●Bármelyik diszk meghibásodása esetén az adat elveszik
→RAID 1 (mirroring):
●Több diszk redundáns használata.
●A file minden része minden (N) diszkre kikerül.
●Azonos diszkeket feltételezve a tárolóterület egy diszk tárolóterületével azonos.
●Az adatátviteli sebesség lassabb, mint egy diszk sebessége.
●A hozzáférési idő nő.
●Speciális esetben az olvasási sebesség N-szeresre nőhet, feltételezve a diszk meghibásodásának más módon történő észlelését (nem kell az azonosságot ellenőrizni többségi szavazással).
●Egy működőképes diszk esetén az adat elérhető.

Hogyan lehet Test_and_Set utasítással kritikus szakaszba lépést (entry) és kilépés (exit) megvalósítani?
→Belépésnél csökkentjük a value értékét ezzel jelezve hogy használni akarjuk a kritikus szakaszt
→Kilépésnél növeljük a value értékét
Egy változót kijelölünk "lock object"-nek; ha ennek a tartalma 0, nincs senki a kritikus szakaszban. A kritikus szakasz elején egy ciklusban test-and-set-et hajtunk végre rá (az utasítást a ciklus feltételébe téve); ha valaki van a szakaszban már, a ciklusban fogunk keringeni, amíg ki nem lép belőle a másik. Amikor kilépett, a test-and-set következő végrehajtása beállítja a változót, és továbbengedi az egyik várakozó ciklust. A szakaszból kilépéskor pedig simán (nem test-and-set-tel) 0-ba állítjuk.

Hogyan működik a test_and_set?
Visszaadja egy bit értékét, és ha 0 volt, 1-re állítja. Mindezt oszthatatlanul, vagyis ha 0 volt ott, és többen egyszerre hívtak rá test-and-set-et, akkor az egyiké teljesen lefut, 1-be állítja és nullát ad vissza, mielőtt a többi elkezdene futni (így ők mind 1-et fognak visszaadni)

Hogyan oldották meg, hogy az alkalmazások többféle API-n (Win32, POSIX) keresztül is meg tudják hívni a Windows operációs rendszer funkcióit?
→Megoldás: környezeti alrendszer (environment subsystem)
→Alkalmazás 1>>>Windows API (Windows alrendszer)>>>NT API (NT Kernel)<<<Posix API (Posix alrendszer)<<<Alkalmazás 2
→alkalmazások viszont nem keverhetik az alrendszereket, mindegyik csak egyet használhat; ezt linkeléskor kell eldönteni

Hogyan számítható ki egy kernel módban futó UNIX folyamat prioritása?
A kernel módú prioritás meghatározása (9. diasor/6)

– A prioritást a folyamat elalvásának az oka határozza meg.

– Ez az ún. alvási prioritás (sleep priority).

– A folyamat felébredése után ez fogja meghatározni a prioritását.

– Az alvási prioritás az alvás okához kötött, pl.:

- 20 diszk I/O-ra vár
- 28 inputra vár a karakteres terminálról

Hogyan történnek a címfordítások ha az OS szegmens és lapszervezést is használ a memóriánál?
CPU --> Segmentation unit --> Paging unit --> Physical memory

Hogyan vált egy UNIX folyamat felhasználói/user módból kernel módba?
Rendszerhívásokon keresztül.

Holtpont megelőzése (prevention) esetén milyen módszerrel lehet a foglalva várakozás előfordulását kizárni?
→Az erőforrást birtokló feladat kér újabb erőforrást.
→Minden szükséges erőforrást egyben kell lefoglalni, egyetlen rendszerhívással.
→Alkalmazástól függ a használhatósága.
→Erőforrás-kihasználás romlik.
→ A foglalva várakozás elkerülhető, ha minden folyamat betartja azt a szabályt, hogy az egyidejűleg szükséges valamennyi erőforrását egyetlen rendszerhívással kéri el. A szabály betartásával megelőzhető a holtpont, de ára az erőforrás-kihasználás jelentős romlása.
→ Ha a folyamatokat kötelezzük arra, hogy minden erőforrásukat egyszerre kérjék el. Ha meg akarjuk engedni a rákérést, akkor menthető állapotú erőforrások esetén megtehetjük, hogy a várakozó folyamatoktól elvesszük az erőforrásaikat.

Írja le a holtpont kialakulásának feltételeit!

→Kölcsönös kizárás
→Foglalva várakozás (Erőforrás lefoglalása és másik erőforrásra való várakozás)
→Nincs erőszakos erőforrás elvétel a rendszerben
→Körkörös várakozás

Mely utasításokkal és miért történik a memóriafoglalás két lépésben Windows alatt?

→A két lépés: Reserve és Commit. Az első csak címtartományt foglal, amögött nem lesz ténylegesen használható memóriaterület; a másik a már lefoglalt címtartományhoz rendel fizikai memóriát.
→(MZ): "a már lefoglalt címtartományhoz rendel fizikai memóriát" --> virtuális memóriát. A lefoglalt lap lehet a fizikai memóriában vagy a lapozófájlban.
→A folyamatok címtartományának töredezettsége csökkenthető azzal, ha a címtartományt már akkor előre foglalja, mikor a memóriára még nincs szüksége, és ez nem jár olyan memóriapocsékolással, mintha fizikai memóriát is foglalna ugyanakkor.

Melyek a Windows hardverfüggő részei?

→A kernel egyes részei és a HAL.
→Megjegyzés: én ide a drivereket is beírtam, nem vontak le érte pontot, de azt mondták, azokat nem szokás a rendszer részének tekinteni.

Melyik az az alrendszere a Windowsnak, ami nélkül nem tud futni?

A Windows alrendszer, avagy Client/Server Runtime SubSystem (csrss.exe). Ennek kilövése kékhalált eredményez.

Mi a belső biztonság?

Belső biztonság = védelem. Védelemnek nevezzük az eljárásoknak és módszereknek azon rendszerét, amely lehetőséget teremt a számítógép erőforrásainak programok, folyamatok illetve felhasználók által történő elérésének szabályozására.

Mi a fájl az operációs rendszer szempontjából? (háttértár-kezelés)

→Absztrakt adattípus (objektum, fájl mutató).
→Adat, név (name – elnev. konvenciók), típus (type – kezelés módja) tulajdonságok (attributes). Tulajdonosok, jogosultságok. Hozzáférési időpontok
→Kölcsönös kizárás (file locking)
→TODO

Mi a fő oka, hogy a Windows NT-ben a képernyőkezelő és grafikus funkciókat megvalósító függvények kernel módba kerültek? Elméleti megfontolások alapján hol lenne a helyük?

Windows NT 4.0-ban került le kernel szintre ez a komponens, hogy kevesebb folyamat- és módváltás legyen (Ne kelljen mindig visszaváltani a csrss.exe-be, majd onnan átváltani kernel módba, utasítani a hardvert, visszaváltani felhasználói módba, majd visszaváltani a felhasználói folyamatba, aki kezdeményezte a változtatást.) (A felhasználói módú folyamatban (csrss.exe) csak a konzol kezelés maradt.) Elméletileg felhasználói szinten kéne lennie.

Mi a HAL (Windows)?

"HAL" - Hardware Abstraction Layer Az aktuális HW-hez (minden processzorhoz saját HAL) virtuális gépet valósít meg. lényegében processzor független szolgáltatások nyújtása - azonban ez architektúra függő.

Mi a jogosultság fogalma, mi a kapcsolata az engedélyezési sémák többi alapfogalmával?

→A jogosultság egy reláció a szereplők és védett objektumok között.
→engedélyezés ált. sémáinál: szereplő>>>szereplőt leíró adatszerkezet>>>biztonsági szabályzat (policy), JOGOSULTSÁG>>>védett objektumok
→Jogosultságkezelés alapjai: A rendszer működése során
- ●A szereplők műveleteket kezdeményeznek
- ●A műveletek kontextusa tartalmazza a szereplő azonosítóját, a célobjektumot és az elvégzendő művelet fajtáját
- ●A jogosultsági döntő komponens kiértékeli kontextust és engedélyezi vagy megtiltja a műveletet
- ●A jogosultsági végrehajtó komponens biztosítja, hogy a döntő által hozott döntés érvényre jusson

→NT: SMR (Secure Reference Monitor) – objektumok elérési jogosultságainak ellenőrzése
→NT: Az LSA a SAM segítségével azonosítja a felhasználót és jogosultságait. Ha a felhasználó jogosult bejelentkezni, a logon elindítja a számára kijelölt shellt
→UNIX: hozzáférési jogosultságok (owner, group, others, read, write, execute)
→Engedélyezés általános sémái: ●Szerep alapú hozzáférés-vezérlés ●Hozzáférési jogosultság listák
→TODO

Mi a Kernel (Windows)?

A rendszer álllandóan memóriában lévő, védett módban futó része. Az NT egyetlen HW függő része, szerepe a HW elfedése a felette található eszközök elől, ezáltal a felette lévő részek már teljesen HW függetlenek. Megvalósítja a szálütemezést, multiprocesszor ütemezést és a TRAP kezelést.

Mi a konvoj hatás, és a tanult ütemező algoritmusok közül melyekben jelentkezhet?

→igen nagy lehet az átlagos várakozási idő, mivel egy-egy hosszú CPU-löketű folyamat feltartja a mögötte várakozókat
→FCFS-nél (First-come, first-served) tapasztalható (pl. SJF (Shortest Job First) és RR algoritmus küszöböli ki)

Mi a középtávú ütemező feladata?

Swapping, azaz a program a fizikai memória és a háttértár közti mozgatása.

Mi a különbség a hierarchikus és a globális erőforrásgazdálkodás között?

hierarchikus: a gyermek folyamatok csak a szülő erőforrásaiból részesülhetnek, és nem léthezetnek önállóan, csak amíg a szülőjük is létezik. globális: a rendszer valamennyi folyamata létrejötte után egyenrangú, önálló szereplő, és versenyezhet a teljes erőforráskészletből való részesedésért.

Mi a különbség a holtpont megelőzése (prevention) és holtpont elkerülése (avoidance) között?

megelőzése: olyan rendszert tervezünk, ahol nem teljesülnek a holtpont feltételei, így elvileg sem lehet holtpont.
→Kölcsönös kizárás minimálisra csökkentése: lehetőleg többpéldányos erőforrásokat alkalmazunk, ahol ez nem lehetséges, ott a hozzáférést megpróbáljuk oszthatatlan műveletté tenni.
→Foglalva várakoztatás megszüntetése: Ha minden folyamat betartja a szabályt, miszerint az egyidejűleg szükséges valamennyi erőforrását egyetlen rendszerhívással kéri el, akkor elkerülhető a foglalva várakoztatás. Ennek ára van: az erőforrás-kihasználtság romlása.
→Nincs erőszakos erőforrás-elvétel kiküszöbölése: Ha menthető állapotú erőforrásaink vannak, akkor megtehetjük, hogy elvesszük egy adott folyamat erőforrását és egy másiknak adjuk, majd annak lefutása után visszaadjuk a régi állapotában az erőforrást az első folyamatnak.
→Körkörös várakozás megakadályozása: A folyamatok megegyeznek az erőforrások sorszámozásában, minden folyamat csak nagyobb sorszámú erőforrást igényelhet azoknál az erőforrásoknál melyeket birtokol. Ekkor biztosan nem alakulhat ki kör. elkerülése (pl. bankár algoritmus): A rendszer minden erőforrásigény kielégítése előtt mérlegeli, hogy nem vezet-e holtpontveszélyre a kérés teljesítése, más szóval fennmarad-e a biztonságos állapot.

Mi a különbség a Hosted és a Bare-metal virtualizáció között?

A Hosted egy Host (teljes értékű) OS-en futó virtualizáció, míg a Bare-metal esetén a virtualizáció közvetlenül a hardware felett van

Mi a különbség a külső és belső tördelődés között? (Memória foglalás)

→Belső tördelődés: A program számára lefoglalt memória területen belül, a kihasználatlan terület
→Külső tördelődés: A programok számára kiosztott memória területek közötti üres (holt) terület.
→A tördelődött memóriaterületet külső tördelődés esetén az operációs rendszer szabadon hagyja, míg belső tördelődés esetén pedig odaadja egy olyan folyamatnak, aminek nincs igazából rá szüksége. tördelődött memóriaterület: Olyan területet, amelyet a operációs rendszer nem tud kiosztani egy folyamatnak sem ha csak összefüggő fizikai címtartományokat oszt ki.
→ tördelődött memóriaterület: Olyan terület, amelyet a operációs rendszer nem tud kiosztani egy folyamatnak sem ha csak összefüggő fizikai címtartományokat oszt ki.

Mi a különbség a logikai és a fizikai memória között?

A logikai memória a fizikai tár leképezve, ráadásul a leképezés a végrehajtás során változhat is.

Mi a különbség a statikus és a dinamikus védelmi tartományok között?

Statikus védelmi tartományok esetén az egy folyamathoz tartozó védelmi tartomány a folyamat végrehajtása során nem változik, míg dinamikus védelmi tartományok esetén igen.

Mi a logikai és mi a fizikai tárolás egysége a permanens táron?

→logikai egység: fájl (file)
→fizikai egység: adatblokkok (cilinder, sáv és szektor együtt azonosítja az írható/olvasható adatblokkot; OS képzi le a logikaiakat fizikaiakra)

Mi a modified/dirty bit és a referenced/used bit szerepe? (Virtuális memóriakezelés)

→laptáblában:
→Módosítás nyilvántartása (modified/dirty bit): minden memórialaphoz tartozik egy HW által kezelt bit (pl. a laptáblában) - betöltéskor törlik, módosításkor beállítják.
→Hivatkozások nyilvántartása (referenced/used bit): OS adott időnként és/vagy adott eseményekre törli - használat esetén beállítják.

Mi a monitor alkalmazásának lényege? (Kölcsönös kizárás)

A lockolás nem szétszórva történik a programban, hanem egyetlen, a közös erőforráshoz szorosan tartozó programrészletben.

Mi a processzor Memory Management Unit (MMU) komponensének a feladata?

→Speciális HW a CPU-ban

→Memória állapotának nyilvántartása
- ●Tulajdonos folyamat azonosítója
- ●Hozzáférési jogosultságok (ACL)
- ●CACHE-elhetőség, ha van CACHE (pl. DMA)

→Virtuális memória leképzése fizikai memóriára
- ●Pl. Translation lookaside buffer (TLB)
- ●Kontextus váltásnál ezt is kezelni kell (ha van)
- ●Pagefile vagy SWAP (HDD)

→Memória védelem
- ●Tiltott memória hozzáférés megakadályozása vagy legalább jelzése (ACL alapján)
- ●General Protection Fault (GPF) a Windows-ban

Mi a referenced/used bit szerepe? (Virtuális memóriakezelés)

→Bizonyos algoritmusok igénylik a lapra történő hivatkozások figyelését is, ami ugyancsak hardvertámogatással hatékony. A laptáblában erre a célra is fenntarthatunk egy bitet. Ezt a hivatkozott bitet (referenced bit, used bit, R bit) a címképző hardver állítja be minden esetben, amikor az adott lapon belüli címre történik hivatkozás. A bitet az operációs rendszer törli adott időnként, vagy eseményhez (például laphiba) kötötten.
→Hivatkozások nyilvántartása (referenced/used bit): OS adott időnként és/vagy adott eseményekre törli - használat esetén beállítják.

Mi a rövidtávú ütemezés, mikor jár környezetváltással?

→Ha a futó folyamatnak lejár az időszelete (csak preemptívnél), önként lemond a processzorról (együttműködő folyamatok), blokkoló rendszerhívást hajt végre (pl. I/O művelet), egy másik szál futásra kész állapotba kerül (bekövetkezik, amire várt, vagy újonnan elindítanak egy szálat), egy szál prioritása megváltozik, esetleg egy szál processzor-affinitása megváltozik.
→Környezetváltással akkor jár, ha másik szál választódik ki futásra, mint ami eddig futott. Pl. Windows NT alatt, ha a legmagasabb prioritási szinten pontosan egy folyamat van, akkor megtörténhet, hogy ugyanaz a szál fut tovább, és nem történik környezetváltás.

Mi a Solaris DTrace megoldás célja?

dinamikus hibakereső rendszer, nyomkövető eszköz, amivel a rendszer és a programok működését futási időben lehet megfigyelni.

Mi a szerepe a quantumnak a Windows ütemezőjében?

A szálak adott ideig futnak (quantum) Quantum hossza: Időegység, amíg egy szál fut Óra megszakításban mérik (clock interval, clock tick) 1 clock tick = ~ 10-15 ms (HALtól függ)

Mi a UNIX inode?

→A fizikai állományokhoz tartozó leíró, azonosító
→minden file-hoz tartozik egy inode állomány amiben a file minden tulajdonsága megtalálható (azonosító,leíró)

Mi a UNIX vnode/vfs?

→Implementáció-független fájlrendszer absztrakció
→vnode: virtuális csomópont, vfs: virtuális állományrendszer
→inode --> vnode
→fs --> vfs
→Új absztrakció: annak felismerése, hogy több állományrendszernek számos előnye van, szükségessé vette a virtuális csomópont (vnode) és a virtuális állományrendszer (vfs) leíró adatszerkezetek bevezetését. Követelmények, elvárások az állományrendszerrel kapcsolatban:
- ●egyszerre támogasson több – UNIX, nem UNIX – állományrendszert
- ●különböző diszk partíciók különböző állományrendszereket is tartalmazhatnak, de mountolás esetén egységet képet kell, hogy mutassanak
- ●támogassa a hálózati állományok osztott használatát
- ●modulárisan bővíthető legyen.

Mi a virtuális gép koncepció lényege?

→A programok elől az operációs rendszer elfedi a hardver implementációs részleteit, és kibővíti azt plusz funkciókkal.
→Az op.rendszer egy olyan réteget képez a hardver fölött, mely elrejti annak körülményességét és bonyolultságát a programozó elõl és kibõvíti a hardver szolgáltatását. A felhasználó így egy sokkal kellemesebb virtuális gépet (virtual machine, extended machine) lát.
→Az operációs rendszer egy kényelmesen kezelhető virtuális gépet jelenít meg a felhasználói és a programozói felületen.

Mi a zombi állapot szerepe egy UNIX rendszerben?

A folyamat már felszabadította a foglalt memóriát, lezárta az állományokat, minden erőforrását visszaadta a rendszernek, csak a proc struktúráját tartja fogva, amiben visszatérési és statisztikai információt tárol a szülő számára. A folyamat szülő wait hívása után szűnik meg.

Mi alapján az azonosítja a Windows a usereket és a csoportokat?

SID - Security Identifier

Mi az a körülfordulási idő?

→TAT (Turnaround Time) -> Egy feladatra vonatkozóan a rendszerbe helyezéstől a teljesítésig eltelt idő.
→Mértékegység: s,
→$t_{CPU,végrehajtási\ idő} + t_{várakozás}$ (Magában foglalja a ténylegesen munkával töltött időt és a várakozást is.)
→felhasználó minél előbb szeretné látni a végeredményt

Mi az a szorosan csatolt rendszer?

Ahol több CPU közös óra és közös memória segítségével működik együtt. Általában egyetlen operációs rendszer van, de az bonyolult. (Megjegyzés: az architektúrákból megtanult "közös erőforrást használnak" definícióra csak fél pontot adtak.)

Mi az a Translation Lookaside Buffer, fizikai címcsatolásnál mi a szerepe?

A virtuális címet fizikai címre a laptábla segítségével lehet fordítani; de ez lassú, plusz egy memória-hozzáférést jelent. Ezért a lapkezdőcímek egy részét egy asszociatív cache-ben eltárolják, ez a TLB. Címfordításkor párhuzamosan indul a keresés a laptáblában és a TLB-ben, ha az egyikben megtalálta, akkor kész.

Mi az a vergődés, és hogyan védekezzünk ellene?

→Ha több memóriára lenne szüksége a folyamatoknak, mint amennyi rendelkezésre áll, ezért túl gyakran keletkezik laphiba, és a processzor idejének nagy része haszontalan lapcserékkel telik.
→Védekezni ellene például azzal lehet, ha a laphiba-gyakoriság függvényében az ütemező változtatja a multiprogramozás fokát: ha kevés a memória, folyamatokat függeszt fel, és swappel ki; ha van elég, akkor épp ellenkezőleg.

Mi az az éhezés?

A folyamatnak megvan mindene, ami a futásához kellene (ezért nem holtpont), de az erőforrásokat, amiket használni akar, más folyamatok kapják meg (ezért nem tud futni).

Mi az eltérés a folyamatok illetve a szálak között, és milyen előnnyel jár a szálak alkalmazása?

A szálak lényegében párhuzamos végrehajtású, közös memóriát használó programrészek a folyamaton belül (egy program végrehajtása több szálon futhat). A szálaknak saját logikai processzoruk van, azonban memóriáik nincsenek elkülönítve, közös logikai memóriát használnak, azaz a kódon és a változókon osztoznak. Emiatt az operációs rendszer lényegesen gyorsabban tud végrehajtani egy átkapcsolást a szálak között, mint a folyamatok között.

Mi az NTDLL.DLL fő funkciója?

Összeköti a User és Kernel módot. Az Executive függvényeknek megfelelő függvénycsonkok vannak benne.

Mi az rpcgen program feladata?

→Az RPC nyelv alkalmas a szerver interfészének formális leírására. A formális leírásból az rpcgen program képes a szerver és a kliens programok megfelelő részeit, valamint a szükséges XDR konverziós függvényeket elkészíteni C nyelven. Az így kapott C forráskódú modulokat a kliens és szerver alkalmazással kibővítve kapjuk a teljes kommunikáló rendszert.
→XDR (Extended Data Representation, kiterjesztett adatreprezentáció): Többféle egyszerű adattípust definiál, illetve szabályokat határoz meg bonyolultabb adatstruktúrák létrehozására. Az adatstruktúrák meghatározásán kívül az XDR egy formális nyelvet is bevezet az adatok leírására. Az RPC rendszer is ezen nyelv kiterjesztését használja a távoli eljáráshívás formális leírására.
→RPC (remote procedure call, távoli eljáráshívás): Az RPC-rendszer egy protokoll-leírást és egy programozói interfészt tartalmaz. Az XDR által definiált formális nyelv kiterjesztését használja a távoli eljáráshívás formális leírására.

Mi határozza meg a UNIX folyamatok kernel módú prioritását a tradicionális UNIX ütemezésben?

A kernel módban futó folyamat prioritása statikus, nem függ attól, hogy a folyamat mennyit használta a CPUt, vagyis mennyi ideig futott. A prioritás attól függ, hogy a folyamat milyen ok miatt hajtott végre sleep rendszerhívást, vagyis, hogy milyen eseményre várakozik. Emiatt a kernel prioritást szokták alvási prioritásnak is nevezni.

Mi történik a PRAM modellben írás-írás ütközés esetén?

Az írás-írás ütközésekor valamelyik művelet hatása érvényesül, a két beírni zándékozott érték valamelyike írja felül a rekesz tartalmát (versenyhelyzet), harmadik érték nem alakulhat ki.

Mi volt a fő oka annak, hogy a Windows NT,ben a képernyőkezelő és grafikus funkciókat megvalósító komponens kernel módba került?

Mert ezek a folyamatok intenzíven használják a hardware-t, és futásuk gyorsaságára az egész rendszer teljesítménye érzékeny. A user módban történő megvalósítás a rendszert nagyon lelassítaná a gyakori környezetváltás miatt.

Miben különbözik egy hosted egy bare-metal típusú virtualizációs megoldástól?

→hosted: jellemzően desktop megoldások (pl. VMware Server, Player)
→bare-metal: jellemzően szerver megoldások (pl. VMware ESX Server)

→hosted-rendszerek: szükségük van egy létező operációs rendszerre (pl. Windows, Linux)
→Az ESX egy saját kernelre épül, amelynek emiatt nincs szüksége egy operációs rendszerre, közvetlenül hardveren fut

Miért előnyös, és miért hátrányos RAID5 használata?

→+ *N* azonos diszk esetén az olvasási és írási adatátviteli sebessége maximum N-szeres közelébe nő.
→+ 1 diszk meghibásodása esetén az adat elérhető.
→- 2 vagy több diszk meghibásodása esetén az adat elveszik.
→- Az adat nem feltétlenül állítható helyre. (Csendes/néma hibák (silent error)).
→-Bonyolultabb, mint a Raid 0/1, ezért hardveresen valósítják meg, ami viszont drága

Miért tud az s5fs gyorsabban írni, mint olvasni (az előadás példája alapján)?

TODO

Miért van a Windowsban külön szabad, és nullázott (freed, és zeroed) memórialap-lista?

Az üres (freed) lapot tilos másik felhasználói programnak adni, ez esetben nullázni (zeroed) kell, nullázás nélkül felhasználhatja pl az OS.

Mik a jogosultságok (privilege) szerepe a Windowsban?

→operációs rendszer szintű jog
→meghatározzák azokat a rendszerműveleteket, amelyeket egy felhasználói azonosító elvégezhet. Egy rendszergazda jogosultságokat felhasználóknak és csoportazonosítóknak oszt (http://msdn.microsoft.com/en-us/library/bb530716(VS.85).aspx)
→pl. számítógép leállítása, eszközmeghajtó betöltése
→név: SeShutDownPrivilege, SeLoadDriverPrivilege

Mikor fut a rövidtávú ütemező és mikor jár környezetváltással?

→Ütemezés következhet be, ha
- a futó folyamat befejeződik,
- egy folyamat felébred, futásra késszé válik,
- a futó folyamat várakozni kényszerül (valamilyen esemény bekövetkezésére), illetve,
- a futó folyamat önként lemond a futás jogáról vagy pedig elveszik tőle.

→Az első és a harmadik esetben az ütemezés mindig környezetváltással jár, hiszen a következő futó folyamat egészen biztosan nem a korábban futott lesz. A másik két esetben előfordulhat, hogy az ütemezőnek nem kell másik folyamatot kiválasztania.
→Ha a futó folyamatnak lejár az időszelete (csak preemptívnél), önként lemond a processzorról (együttműködő folyamatok), blokkoló rendszerhívást hajt végre (pl. I/O művelet), egy másik szál futásra kész állapotba kerül (bekövetkezik, amire várt, vagy újonnan elindítanak egy szálat), egy szál prioritása megváltozik, esetleg egy szál processzor-affinitása megváltozik.
→Környezetváltással akkor jár, ha másik szál választódik ki futásra, mint ami eddig futott. Pl. Windows NT alatt, ha a legmagasabb prioritási szinten pontosan egy folyamat van, akkor megtörténhet, hogy ugyanaz a szál fut tovább, és nem történik környezetváltás.

Mikor lehet két tevékenységet (utasítássorozatot) párhuzamosan végrehajtani (Bernstein)?

Bernstein feltétele:
→Legyen P_i és P_j két darabja egy programnak.
→A P_i összes bemeneti változója I_i , és az összes kimeneti változója O_i , ugyanez P_j -re I_j és O_j .
→A két program párhuzamosan végrehajtható (vagyis független), ha: $I_j \cap O_i = \mathbf{0}$, $I_i \cap O_j = \mathbf{0}$ és $O_i \cap O_j = \mathbf{0}$

Mikor nevezünk egy ütemezőt preemptívnek?

Ha az OS elveheti a futásjogot (a CPU-t) egy folyamattól/futó feladattól (interrupt).

Mikor nevezünk statikusnak, illetve dinamikusnak egy operációs rendszert?

statikus: azok a rendszerek, amelyeknek muködése során - a felépülés és inicializálás kezdeti szakaszától eltekintve - nem jönnek létre és nem szűnnek meg folyamatok. dinamikus: működés közben bármikor születhetnek illetve megszünhetnek folyamatok.

Milyen címtranszformációk történnek együttes szegmens- és lapszervezésű memória használata során?

→https://wiki.sch.bme.hu/pub/Infoalap/OpRe/oprewiki1.pdf, 48. oldaltól.
→Változó méretű szegmensek fix méretű lapokat tartalmaznak. Kicsi mind a belső, mind a külső tördelődés. A cím felépítése: (szegmens szám, lapszám, lapon belüli eltolás)
→Hasonlít a szegmensszervezéshez és a kétszintű lapszervezéshez: A memóriában szegmensek vannak ugyan, de ezek lapokból épülnek föl. Van szegmenstábla, és minden bejegyzéséhez tartozik egy laptábla is. Külső töredeződés nincs, belső töredeződés minimális (szegmensenként átlag fél lap); ez a kombinált módszer egyesíti a két módszer előnyeit
→TODO

Milyen előnnyel jár a rendszerhívások valamilyen magas szintű programnyelvvel történő megadása?
Abból a szempontból előnyös, hogy az alkalmazási felület így processzorfüggetlenné válik.

Milyen futási módban és kontextusban zajlik a UNIX rendszerhívások kiszolgálása?
Kernel módban fut a kód, és a rendszert hívó folyamat kontextusában.

Milyen interfészen keresztül érhetők el a UNIX kernel szolgáltatásai?
→System Call Interface
→Az alkalmazások a rendszerkönyvtárakat hívják meg, amelyek szükség szerint meghívják az operációs rendszer szolgáltatásait ??

Milyen kontextusban hajtódnak végre a UNIX rendszerhívások?
folyamat kontextusban

Milyen módban és kontextusban zajlik a rendszerhívások kiszolgálása a UNIX operációs rendszerben?
kernel mód, folyamat kontextus ((az ehhez tartozó ábra jobb felső része))

Milyen módban hajtódnak végre a UNIX rendszerhívások?
kernel módban

Milyen módokon képezheti le a JAVA virtuális gép a JAVA natív szálakat a hoszt operációs rendszer folyamataira/szálaira?
A JAVA virtuális gép egy folyamat a hoszt operációs rendszeren belül. A JAVA szálak feleltethetők meg a hoszt operációs rendszer szálainak, ez többnyire one-to-one (JAVA szál egyben OS szál is) napjainkban.
(https://wiki.sch.bme.hu/pub/Infoalap/OpRe/20100507_ZH_megoldas.pdf)

Milyen rendszereket nevezünk "szorosan csatolt" rendszereknek?
Ahol több CPU közös óra és közös memória segítségével működik együtt. Általában egyetlen operációs rendszer van, de az bonyolult. (Megjegyzés: az architektúrákból megtanult "közös erőforrást használnak" definícióra csak fél pontot adtak.)

Milyen részekből áll az RPC technológia?
→RPC: Remote Procedure Call, távoli eljáráshívás. Magas szintű folyamatok közti kommunikációt tesz lehetővé. Részei:
→ a hívható eljárások és típusaik (interfész) leírása
→programgenerátor - rpcgen: a leírásból C programkódot generáló program
→kommunikációs infrastruktúra - portmapper: a programazonosítók és a hálózati portok összerendelése
→azonosítók: a leírásban megadott egyedi számok (program, eljárás)

Milyen részekből áll az RPC technológia?
→Unix folyamatok kommunikációja diasor, 12. dia

Milyen részidőkből áll össze a háttértáron levő lapokhoz való tényleges hozzáférési idő? Kis vagy nagy lapok használata esetén kapunk "jobb" byte hozzáférést?
→adatátviteli sebesség + fejmozgás sebessége + lemezek forgási sebessége
→nagy lapok esetén (mert így közvetlenül egymás után helyezkednek el az összetartőző adatok így nem kell a fejnek "ugrálnia")
→Először a laptáblából kell kikeresni a lap bejegyzését, és konstatálni, hogy nincs hozzá fizikai lap rendelve. Majd, ki kell választani egy szabad fizikai lapot (ha nincs, ki kell vinni egyet háttértárra), a szabad helyre beolvasni a lapot, majd újraindítani a laphibát okozó utasítást. Ezek közül a háttértárról olvasás nagyságrendekkel lassabb a többinél, ezért lényegében ez határozza meg a teljes hozzáférési időt.
→Ha csak a háttértáron lévő lapokat nézzük, akkor, mivel kisebb lapot gyorsabban lehet beolvasni, ezért kisebb lapoknál gyorsabb a hozzáférés. Ha egy folyamat teljes munkahalmazát nézzük, akkor viszont a kisebb lapok több adminisztrációs költséggel járnak (gyakrabban kell háttértárhoz fordulni), és átlagban a nagyobb lapok adnak jobb eredményt.

Milyen speciális joga van egy védett objektum tulajdonosának az adott objektumra a Windowsban?
Megváltoztathatja az objektum engedélyeit, akkor is, ha nincs explicit joga. (https://wiki.sch.bme.hu/pub/Infoalap/OpRe/03-opre-windows-biztonsag.pptx)

Milyen szinkronizációs kényszereket jelent, ha egy lazán csatolt rendszer kommunikációja során véges kapacitású csatornát alkalmazunk?
ha a küldő folyamat túl gyorsan küldözget, akkor a csatorna megtelik, úgyhogy túlcsordulás lesz, ami miatt a küldőnek várnia kell mielőtt újra küld.

Mire szolgál a standby memória lap lista a Windowsban (miért nem szabad lapként vannak ezek nyilvántartva)?
→Page belonged to a working set but was removed; not modified
→ - s: memórialapok állapota

●felső lista: fizikai memórialapok állapota. A Zeroed, Free és Standby lapokat lehet odaadni valakinek, ha memóriát igényel
●alsó lista: Vista kernel óta a Standby lista 8 prioritásos listára van osztva. Ha új igény van, akkor az alacsonyabb prioritású listákról vesz lapokat, így a fontos folyamatok memórialapjai még ottmaradnak a Standby listában, ha mégis kell nekik.
→Superfetch (Vista) : 8 Prioritás a memórialapokhoz - Standby listából 8 darab ennek megfelelően. Lapok használatának követése. Memória felhasználása esetén lassan visszahoz lapokat a standby listára, amik kellhetnek még
→TODO (https://wiki.sch.bme.hu/pub/Infoalap/OpRe/02-opre-windows-memoria.pptx)

Mire szolgál a Translation Lookaside Buffer és mi a szerepe a fizikai cím kiszámításánál (virtuális címképzés)?

A virtuális címet fizikai címre a laptábla segítségével lehet fordítani; de ez lassú, plusz egy memória-hozzáférést jelent. Ezért a lapkezdőcímek egy részét egy asszociatív cache-ben eltárolják, ez a TLB. Címfordításkor párhuzamosan indul a keresés a laptáblában és a TLB-ben, ha az egyikben megtalálta, akkor kész.

Mire szolgál a UNIX exec() rendszerhívás?

exec(): új programkód betöltése egy folyamat címterébe
Azaz a fork() paranccsal létrehozunk egy új folyamatot, exec() paranccsal pedig beöltöjük a folyamatba a kódot.

Mit állítunk be, ha egy szálnak beállítjuk a processzor affinitását, és miért lehet arra szükség?

→processzoraffinitás: minden szál rendelkezik egy maszkkal, amely kijelöli, hogy a szál mely processzorokon képes futni
→szerepe: ez alapján dől el, hogy a szál mely processzoron fog futni
→ütemezésnél: multiprocesszoros esetben a processzor kiválasztása a processzor-affinitás alapján történik
→A feladat más processzorra, vagy processzormagra kerülése csökkenti a végrehajtás sebességét (pl. cache-elésnél) >> Cél: A feladatot ugyanazon a végrehajtó egységen tartani - Laza vagy kemény processzor affinitás (soft or hard processor affinity).
●Laza: Nincs garancia, de törekszik rá az OS (többnyire alapeset)
●Kemény: Biztosan ugyanazon a CPU-n marad (rendszerhívással)

Mit jelent a "graceful degradation" fogalma?

Fokozatos leromlás/összeomlás: Ha a rendszer terhelése eléri az ún. könyökkapacitást, akkor utána viselkedése megváltozik, a tovább növekvő terhelésre már egyre rosszabb működéssel reagál (overhead). Elvárható, hogy ezt fokozatosan tegye (ne omoljon össze).

Mit jelent a kritikus szakasz?

A magában szekvenciális feladatok azon kódrészletei, amely során a kölcsönös kizárást egy bizonyos közös erőforrásra biztosítjuk. A kritikus szakasz a kérdéses közös erőforráshoz tartozik. A kritikus szakaszt a hozzá tartozó erőforrásra atomi műveletként (nem megszakítható módon) kell végrehajtanunk.

Mit jelent a Windowsban az egy folyamathoz tartozó munkakészlet fogalma?

A folyamat azon lapjainak halmaza, amelyekre egy időintervallumban (munkahalmaz-ablak) a folyamat hivatkozik. (WSS - Working-Set)

Mit jelent az inkrementális mentés?

Csak a változtatásokat mentjük az előző mentéshez képest -> kisebb helyet foglal, hamarabb végez a mentés.

Mit jelent az újrahívhatóság (reentrancy) fogalma?

A közös erőforrás problémájának egyfajta kiterjesztett esete egy függvényen/objektumon belül is felléphet, amennyiben ezt a függvényt (metódust) egyszerre többen is meghívhatják. Előfordulhat ha, ugyanazt a függvényt hívjuk egy taszkból is és egy megszakítás rutinból is, vagy az ütemezés preemptív, és ugyanazt a függvényt hívjuk két taszkból is.

Mit jelent az, ha egy x86-os processzor hardveres virtualizáció támogatással rendelkezik?

Speciális utasításokkal látják el a processzort, amit szoftveresen akár több 100 utasításon keresztül lehetne csak megoldani.

Mit jelent az, hogy a Windowsban a rendszerhívások újrahívhatóak?

a rendszerhívásokat több alkalmazás is meghívhatja egyszerre, nem blokkódnak, ha már valakit éppen kiszolgál az adott rendszerhívás

Mit jelent az, hogy egy virtualizációs megoldás paravirtualizációt használ a CPU virtualizálásához?

→ a hardverszimuláció helyett – vagy azt kiegészítendő – egyfajta hozzáférési, programozási réteget (API) ad a hardver és a futtatott rendszerek közé. Ezen megoldások esetén a futó rendszereknek idomulniuk kell a környezethez, amennyiben például az operációs rendszer magjában (kernel) megfelelő támogatásra van szükség a fent említett API-n keresztül való működéshez.Az API - úgynevezett hypervisor – biztosítja a közvetett hardverelérést.
→a guest speciális API-n (hypervisor) keresztül éri el a hardvert
→fizikai hardver elérés a hipervizoron keresztül (hypervisor: szuper-privilegizált módban futó kernel, amin a virtuális gépek futnak)
→jó teljesítményt tesz lehetővé, hardvertámogatottság növekvő, de guest OS-t módositani kell (a úgy, hogy a problémás utasítások helyett speciális rendszerhívásokat használjon)

→ a memória laptáblák kezelésére is speciális rendszerhívásokat használ a közvetlen MMU-elérés helyett
→a vendég operációs rendszer CPU ütemezése kooperálhat a virtualizációs keretrendszer ütemezőjével

Mit jelent és miért van szükség arra, hogy a virtuális tárkezelésnél egyes lapokat ideiglenesen a tárba lehessen "fagyasztani" (page locking)?

→Azt jelenti, hogy a page replacement algoritmus nem lapozhatja ki a háttértárra az adott lapot. Ok: periféria-művelet van az adott lappal kapcsolatban.
→Azt jelenti, hogy bizonyos lapokat a memóriában tartunk, mert I/O műveletek hivatkozhatnak rá, és ilyenkor a memóriában kell lenniük, mert az I/O műveletek fizikai memóriacímeket használnak.

Mit jelentenek a számok és szavak a következő verzióleírásban: "Microsoft (R) Windows (R) 5.01.2006 Service Pack 2 Uniprocessor Free"?

→(MZ) 5.01.2006 a verziószám, major.minor.build formában, 5.1 a Windows XP verzója, 2006 az SP2-es verzió build száma. Uniprocessor = egy processzoros kernel verzió, Free = debug szimbólumok nélküli verzió.
→Most computers run a "uniprocessor free" version of Windows, which is a version that runs on a single CPU and does not contain extra errorchecking.

Mit nevezünk Béládi-anomáliának?

FIFO algoritmusnál egyes esetekben, ha a munkahalmaz méretét növeljük, a várakozásokkal ellentétben a laphibák száma is nő.

Mit nevezünk vergődésnek és hogy lehet védekezni ellene?

→A gyakori laphibák által okozott rendszer teljesítmény csökkenést vergődésnek (trashing) nevezzük. védekezés: Lokális lapcsere stratégia:
→A folyamatok nem tudják egymástól elvenni a fizikai memória kereteket.
→Nem terjedhet át a hatás más feladatokra.
→Csökkenti a problémát, de nem oldja meg.
→Ha több memóriára lenne szüksége a folyamatoknak, mint amennyi rendelkezésre áll, ezért túl gyakran keletkezik laphiba, és a processzor idejének nagy része haszontalan lapcserékkel telik.
→Védekezni ellene például azzal lehet, ha a laphiba-gyakoriság függvényében az ütemező változtatja a multiprogramozás fokát: ha kevés a memória, folyamatokat függeszt fel, és swappel ki; ha van elég, akkor épp ellenkezőleg.

Mit takar a külső biztonság fogalma?

Annak mértéke, hogy mennyire lehetünk biztosak a számítógépes rendszer, illetve a rendszerben tárolt adatok sérthetetlenségében.

Mivel azonosítja a Windows a felhasználókat és csoportokat a fájl hozzáférési listákban?

→objektum (SecurableObject) → (SID, engedélyek) ; engedély: adatok írása, attribútumok olvasása…
→SecurityDescriptor (biztonsági leíró, összefogja a többi elemet) >> Owner (Tulajdonos, megváltoztathatja az objektum engedélyeit, akkor is ha nincs explicit joga), Discretionary Access Control List (DACL, belátás szerinti, erőforrás szintű, hozzáférési lista – hozzáférés szabályozása), SACL (biztonsági naplózás szabályozása – kinek milyen művelete esetén kell naplózni az adott műveletet)
→AccessControlEntry:
- ●Típus: megengedő, tiltó, audit
- ●Flag: Pl. öröklődés
- ●SID: kire vonatkozik
- ●Maszk: végrehajtás | törlés | tulajdonos írása…

→elérési lista (ACL), melyben megadható, hogy mely folyamatok jogosultak az adott section object elérésére ???
→ minden objektumhoz tároljuk a hozzá tartozó <tartomány, műveletvégzési jog> párokat ????
→TODO

Mivel azonosítja a Windows a felhasználókat és csoportokat a hozzáférések ellenőrzése során?

→<Gép SID>-<RID> (SID security identifier - gépspecifikus, RID: relative identifier)
→Jól ismert SID-ek: Everyone: S-1-1-0, Administrator: S-1-5-domain-500
→Vista: szolgáltatások is kapnak SID-et
→objektum → (SID, engedélyek) ; engedély: adatok írása, attribútumok olvasása…
→TODO

Mondjon legalább egy, UNIX VFS-alapú "fájlrendszert", amelynek a célja nem fájlok tárolása.

/dev, /proc, stb.

N db azonos diszkből álló RAID6 tömb esetén a tömb tárolókapacitása és sebessége (nagy fájlok írása/olvasása során elérhető adatátviteli sebesség) hogyan viszonyul az egyetlen diszk azonos adataihoz?

→RAID 6 (block interleaved dual distributed parity)

→Több diszk redundáns és párhuzamos használata.
→Adat és paritás elosztása N+2 diszkre.
- ●A sebesség tekintetében közel áll az N diszket használó RAID 0-hoz (HW támogatás esetén).
- ●2 diszk meghibásodása esetén az adat elérhető.
- ●3 vagy több diszk meghibásodása esetén az adat elveszik.
- ●Az adat nagyobb valószínűséggel állítható helyre a RAID 5-höz képest

→A minimum of four disks is required to create RAID 6. The capacity of the array is (N-2) times the size of the smallest member disk for the array of N disks.
→Read speed is (N-2) times faster than in case of a single disk - two disks in the row hold a parity which is useless to read. Such read speed values are roughly the same as in RAID 5.

Nevezzen meg egy kliens-szerver-modell alapján működő komponenst az NT-ben!

→A szolgáltatások
→TODO

Sorolja fel a fontosabb UNIX fájl attribútumokat!

→Típus
→Linkek
→Eszköz, inode, méret…
→Időbélyegek
→Azonosítási és hozzáférés-szabályozási adatok

Sorolja fel a holtpont kialakulásának szükséges feltételeit!

→Kölcsönös kizárás: Vannak olyan erőforrások a rendszerben, melyeket a folyamatok csak kizárólagosan használhatnak.
→Foglalva várakozás: legyen olyan folyamat mely lefoglalva tart erőforrásokat, miközben más erőforrásokra várakozik.
→Nincs erőszakos erőforrás-elvétel: a folyamatok addig birtokolják az erőforrást, míg saját jószántukból fel nem szabadítják azokat.
→Körkörös várakozás: Létezik a rendszerben egy olyan folyamatsorozat, melyben minden folyamat az utána következő folyamat által foglalt erőforrásra vár, a sorozat utolsó tagja pedig a sorozat első tagjára.

Sorolja fel a RAID technika leglényegesebb elemeit!

→Használjunk több merevlemezt egyszerre.
→Több redundáns alkalmazása növeli a megbízhatóságot.
→Több párhuzamos használata növeli a sebességet.
→Hozzunk létre egy virtuális diszket a fizikai diszkekből.
→Redundant Array of Inexpensive Disks: több lemez összekapcsolása.
→A RAID-0 esetében két lemezre vannak szétosztva az adatok, így egyetlen fájlt kétszer akkora sebességgel lehet írni (a két felét parhuzamosan).
→A RAID-1 esetében ugyanazt az adatot tároljuk le a két lemezen, így gyorsabb nem lesz, de az egyik lemez hibája esetén visszanyerhetőek az adatok.
→Megjegyzés: ez csak példa, több lemezzel is lehet csinálni, a sebesség/tárhely/hibatűrés között különböző kompromisszumokat elérve.

Sorolja fel a terhelés végrehajtó egységek közötti megosztásának megoldásait! (Többprocesszoros rendszerek)

→Master and slaves (egy CPU osztja ki a feladatokat)
→Self-scheduling / peering (minden CPU ütemez)
→Globális futásra kész sor
→Processzoronkénti futásra kész sor
- ● Push alapú: OS kernel folyamat mozgatja a sorok között a feladatokat.
- ● Pull alapú: Az idle állapotban (idle feladatot végrehajtó) CPU próbál a többi sorából feladatot kapni.

● Kettő kombinációja
→Összefüggő, párhuzamosan futtatható feladatok optimalizálása (pl. Gang scheduler)

Sorolja fel a UNIX fájlrendszeri bejegyzések alapvető tulajdonságait (legalább hármat, ls -l oszlopok)!

→pl.: drwxr-xr-x 2 root root 4096 dec 22 12.27 txt
→sorrendben: I. UNIX-fájltípusok (pl. közönséges fájl (-), katalógus (d), szimbolikus link (l), stb.), II. hozzáférési jogosultságok (3*3-as bontásban – 1. hármas csoport a tulajdonos, a 2. a csoport, a 3. a többiek jogosultságait; 'r' az olvasás (read), a 'w' az írás (write), az 'x' pedig a végrehajtás (execute) jele), III. jogosultságok után egy szám áll (ez könyvtárak esetén azt mondja meg, hogy az adott könyvtár hány elemet tartalmaz, fájlok esetén azt tudhatjuk meg, hogy az adott fájlra hány hardlink mutat), IV. tulajdonos, V. méret (bájtokban), VI. utolsó módosítás dátuma, VII. fájl neve

Sorolja fel a UNIX folyamatok legalább 4 alapvető adminisztratív adatát!

→PID (Process ID): egyedi, a folyamatot azonosító szám (PPID: szülő folyamat azonosítója)
→A folyamat állapota (fut, alszik, stb.; ütemezési információk (prioritás, CPU használat, nice érték))
→Hitelesítők (UID, GID: a kapcsolódó felhasználó adatai)
→Memória-kezelési adatok (címleképezési térkép)
→Kommunikációs adatok (fájlleírók, jelzés információk)
→Statisztikák (erőforrás használat (számlázáshoz))

Sorolja fel a UNIX operációs rendszer főbb belső szerkezeti elemeit!

→betöltő
→virtuálismemória-kezelő
→állományrendszer
→blokkos berendezés-meghajtó kapcsoló (+ a hozzá kapcsolódó eszközmeghajtók, pl.: lemezegység, szalagos meghajtó)
→karakteres berendezés-meghajtó kapcsoló (+ a hozzá kapcsolódó eszközmeghajtók, pl.: hálózat, nyomtató)

Sorolja fel a UNIX System V IPC elemek közös alapjának részeit!

Minden IPC erőforrás rendelkezik a következő azonosítókkal: kulcs (key), létrehozó (creator), tulajdonos (owner), hozzáférési jogok (permissions)

Sorolja fel az indexelt tárolás (indexed allocation) előnyeit és hátrányait! (Fájlrendszer leképzés)

→Szekvenciális és indexelt elérésre is alkalmas.
→Sérülékeny (az index blokkok sérülése a fájlt elérhetetlenné teszi).
→Az index blokkokat viszont könnyű többszörözni (replikálni).
→Sok fejmozgást okoz (seek), a blokkok el vannak szórva a diszken.
→Itt is lehet a láncolt listás töredezettség mentesítéshez hasonló algoritmusokat használni a fejmozgás minimalizálására.

Sorolja fel az NT hardware-függő rétegeit!

HAL (Hardware Abstraction Level) Kernel

Sorolja fel milyen tényezők határozzák meg egy UNIX folyamat felhasználói módú prioritását (tradicionális UNIX ütemező esetén)!

→Korábbi CPU-használat
→Futásra kész folyamatok száma (p_cpu „öregítésével”)
→nice érték (nice, és renice parancsok)

Soroljon fel fő UNIX fajtákat!

Linux, Solaris, BSD,System V,HP/UX,…

Soroljon fel legalább 4 UNIX folyamatok között kommunikációs megoldást!

→System V IPC: szemafor, osztott memória, üzenetsor
→Csővezeték és nevesített csővezeték
→Jelzések
→RPC
→Folyamat-nyomkövetés
→Szemaforok
→Üzenetsorok
→Osztott memória
→Hálózati, socketeken keresztüli kommunikáció

Soroljon fel UNIX folyamatok közötti adatátviteli eszközöket (legalább hármat)!

→System V eszközök: – szemaforok, – üzenetsorok, – osztott memória
→Jelzések: – aszinkron események keltése és kezelése
→Csővezetékek: – FIFO kommunikáció a „rokonságban”
→Szemaforok: – a korábban megismert szinkronizációs megoldások
→Üzenetsorok: – diszkrét, típusos üzenetek folyamatok között
→Osztott memória: – azonos fizikai memóriaterület használata több folyamatban
→„hálózati” (socket) kommunikáció: – címzéssel és protokollokkal támogatott kommunikáció
→TODO

Soroljon fel UNIX szabványokat!

→POSIX.1 (teljes nevén: POSIX1003.1): C nyelvű szabványos rendszerhívás-interfész
→System V Interface Definition
→X/Open Portability Guide

Soroljon fel UNIX típusokat (a családfa jellemző ágait)!

→System V (AT&T változat; Solaris, SCO),
→BSD (Berkeley változat; SunOS, OpenBSD)

Soroljon fel UNIX VFS-alapú fájlrendszereket (legalább négyet)!

xfs, zfs, brtfs, nfs

Soroljon fel UNIX-höz köthető szabványokat (legalább kettőt)!

→POSIX.1 (teljes nevén: POSIX1003.1): C nyelvű szabványos rendszerhívás-interfész
→System V Interface Definition
→X/Open Portability Guide
→– AT&T SVID (pl. SVR4), – IEEE POSIX, – Open Group X/Open, Unix95, Unix98, … ????

Definiálja a szál (thread) fogalmát!

A szálak egy folyamaton belül párhuzamosan futó programrészek. Minden szálnak saját logikai processzora van, a folyamatokhoz hasonlóan versenyezhetnek a processzorért. A szálak logikai memóriája ezzel szemben közös, tehát az egy folyamaton belüli összes szálnak egyetlen logikai memóriája van

Trap & emulate virtualizációs módszer használata esetén mi történik a vendég gép által kiadott nem privilegizált utasítással?

→nem privilegizált utasítások közvetlenül a valós CPU-n hajtódnak végre (no VMM intervention)
→TODO

UNIX alatt milyen rendszerhívásokra van szükség ha a user elindít egy programot (folyamat létrehozása és programkód betöltése)?

→Folyamatot létrehozni a fork() hívással, majd betölteni a programkódot pedig az exec() hívással lehet

UNIX alvási prioritásának ütemezését mi végzi?

Az alvási prioritást is az ütemező határozza meg, az alapján, hogy mire várakozik a folyamat, vagyis miért hajtott végre sleep() rendszerhívást. Kernel módban az ütemező nem veheti el a futási jogot, ezért amíg nem hajt végre sleep() hívást, addig nincs is szükség a prioritásának meghatározására.

UNIX OS esetén mi történik folyamat kontextus esetén kernel módban?

Kivételek, rendszerhívások kezelése.

Windowsban miért került le az ablakkezelő kernel módba?

Hogy kevesebb folyamat és módváltás legyen, mivel a Windows szerves része az ablakkezelés, ezért rengeteg user-kernel mód váltás lenne ha a csrss.exe-en keresztül használnánk. Tehát teljesítménybeli okokból.