Operációs rendszerek 1. kidolgozott tételsor

Verzió: 1.0 (Build: 1.0.2011.05.19.)

Készült:

- Operációs rendszerek I. elméleti (Dr. Fazekas Gábor), gyakorlati (Dr. Adamkó Attila) jegyzet
- Számítógép architektúrák elméleti (Dr. Fazekas Gábor) jegyzet
- Számítógép architektúrák kidolgozott tételsor
- Operációs rendszerek 1 fóliák
- Ellenörző kérdések 2011.05.19. állapota alapján

Készítette: Boruzs Tibor

Figyelem! Bukásért felelősséget nem vállalok!

2010/11-es tanév, 2. félév

1. Mit jelent a beállítási idő (setup time)? Miért volt a korai rendszerekben nagy a beállítási idő? Hogyan lehet csökkenteni?

A beállítási idő az az idő mely egy program befejezése és a következő program indítása közti idő. Lyukszalagos időben ez az idő túl sok volt, mivel nem volt semmilyen automatizmus az egyik program befejezése és a másik indítása között, mivel minden program külön lyukszalagon volt, és a felhasználónak kellett a lyukszalagot behelyeznie az olvasóba. Tehát folyamatos felhasználói figyelmet igényelt. A megoldást a kötegelt feldolgozás jelentette, a lyukkártya, és a végrehajtó monitor. A lyukszalagot leváltotta a lyukkártya, mellyel megszűnt a lyukszalag behelyezésének és felcsévélésének ideje, kevesebb felhasználói felügyeletet igényelt. A kötegelt feldolgozással egy időben megjelentek a merevlemezek, melyeknek köszönhetően először a lyukkártyáról a merevlemezre töltődnek a programok, és a merevlemezről töltődnek be végrehajtásra (Spooling).

2. Mit jelent a „Moore törvény”?

Az alkatrészek integrációja egy szilíciumlapkára 18 havonta megduplázódik a processzor teljesítménnyel együtt. A probléma: Az adatátvitel (I/O műveletek) sokkal lassabban nő. (Pl.: CPU VS. Winchester) Megoldás a processzor és az I/O sebesség egyre nagyobb távolságának leküzdésére: multiprogramozás.

3. Milyen következtetést kell levonni az operációs rendszerek tervezőinek a Moore törvényből?

A processzor és az I/O sebesség egyre nagyobb távolságával keletkezett hézagot át kell hidalni. Az I/O problémákat függetleníteni kell a processzortól, külön segédprocesszor kell (csatorna). Ezek a párhuzamos I/O eszközök, az I/O adapterek.

4. Mi a csatorna (channel) szerepe a számítógép átbocsátóképességének növelésében?

A csatorna párhuzamosítja a processzor és a I/O műveleteket. A multiprogramozásban használják.

5. Mit jelent a kötegelt feldolgozás? Milyen előnyei és hátrányai vannak?

Olyan megoldás, melynek köszönhetően az egymás után végrehajtható folyamatok indításai automatizálhatók. Ennek következtében csökken a beállítási idő (Setup Time). A hátránya, hogy nincs interrupció.

6. Milyen funkcionális részei vannak a kötegelt feldolgozás vezérlő monitorának?

Vezérlő kártya interpreter: felelős a vezérlőkártyák beolvasásáért és értelmezéséért.
Betöltő (loader): háttértárból betölti az egyes rendszer és felhasználói programokat az operatív memóriába.
Készülék meghajtó programok (device drivers): ismerik a rendszer az egyes I/O berendezéseinek tulajdonságait és működtetésük logikáját.

7. Mi a multiprogramozás? Hogyan értékelhető a Moore törvény tükrében?

Egyszerre több program van a memóriában, a processzort felváltva kapják meg. A multiprogramozás párhuzamosítást szimulál, ezért van az, hogy például egyszerre tudunk a számítógépen szöveget szerkeszteni és zenét hallgatni. A multiprogramozás segít a processzor és az I/O műveletek sebessége közti különbséget kiküszöbölni.

8. Melyek a multiprogramozás által az operációs rendszerekkel szemben támasztott legfontosabb követelmények?

- Adatvédelem: I/O teljesen az operációs rendszerre van utalva, csak az operációs rendszerrel lehet használni.
- Memóriavédelem: biztosítani kell, hogy ne férjen hozzá egyik folyamat kódja a másik folyamat kódjához a memórián belül
- Processz ütemezés: kell az operációs rendszer közreműködése, hogy a processzort melyik folyamat kapja meg.
- Készülék hozzárendelés: Eldönti melyik folyamatnak melyik hardvert adja oda

9. Mit jelent a SPOOL-ing?

A kártyaolvasó teljesítményének növelését segítette. Amíg futott egy folyamat, a kártyaolvasó beolvasta a következő utasítást, a kódot a merevlemezre mentette, és mikor el kellett indítani a következő folyamatot, betöltötte a merevlemezről.

10. Jellemezze az időosztásos rendszereket!

Multiprogramozott rendszer, CPU-kiosztás specifikus. A rendszerben meg van határozva egy bizonyos intervallum. Minden felhasználó aktuális processze erre az időszeletre megkapta a CPU-t, és ha nem futott le az időszeleten belül, elveszi az erőforrást, és vár, amíg újra rá nem kerül a sor. Lassú, ha egy hálózaton sokan várnak a CPU-ra. Pótolja a batch programozás hiányát, a gép interaktivitását.

11. Jellemezze a személyi számítógépes rendszereket!

Új operációs rendszer paradigma létrejöttét eredményezte, a rendszerszoftverét. Az elgondolás az volt, hogy több ezer gépen és különböző konfigurációkon is működőképes legyen. Létre kell hozni a kapcsolatot ember és gép között (ergonómia), valamint a számítógép használata ne legyen egészségtelen (Pl. régi képernyők sugárzása).

12. Jellemezze a párhuzamos rendszereket!

Multiprocesszoros rendszerek több mint egy – szoros kommunikációs kapcsolatban levő – CPU-val. Szorosan kapcsolt/csatolt rendszerek – a processzorok közösen használják a memóriát és a rendszer órát. A kommunikáció a közös memória segítségével történik. Párhuzamos rendszerek elônyei:

- Megnövelt átbocsátó képesség,
- Gazdaságosság,
- Növekvô megbízhatóság,
- Redundancia,
- Graceful degradation,

- Fail-soft rendszerek.

Párhuzamos rendszerek

- Szimmetrikus multiprocesszálás

Minden egyes processzor az operációs rendszer azonos változatát(másolatát) futtatja. Ezek egymással szükség szerint kommunikálhatnak. Sok processz futhat egyszerre teljesítménycsökkenés nélkül. I/O problémák, ütemezés

- Aszimmetrikus multiprocesszálás (master-slave modell)

Minden egyes processzor a hozzárendelt specifikus feladatot (task)oldja meg. A feladatot a mester határozza meg! Ezek a taskok egymással szükség szerint kommunikálhatnak.

  - Nagyon nagy rendszerekben elterjedtebb megoldás.
  - RJE (remote job entry), front-end processzorok.

13.Jellemezze a valós idejű rendszereket!

Olyan kapcsolat gép és környezete között, hogy igény beadása után a számítógép belátható időn belül megkezdi a válaszadást. Gyakran úgy jelenik meg, mint valamilyen dedikált alkalmazás (pl. tudományos kísérlet támogatása, orvosi képfeldolgozás, ipari kontroll, kijelző rendszerek) irányító-felügyelő rendszere. A „kiszolgálás" azonnal megkezdődik! Jól definiált, rögzített idejű korlátozások.

- „Hard" ("merev" valós idejű) rendszerek.

A másodlagos tár korlátozott, vagy teljesen hiányzik; az adatokat az operatív memóriában (RAM), vagy akár ROM-ban tárolják. Konfliktus az időosztásos rendszerekkel.

- „Szoft" ("puha" valós idejû) rendszerek.

Korlátozott szolgáltató programok az ipari kontroll, a robotika területén. A fejlett operációs rendszer szolgáltatásokat igénylő alkalmazásoknál(Multimédia, VR) igen hasznosak.

14.Mit jelent a duál módú működés és mi a szerepe az operációs rendszer szempontjából?

A duál módú működés lényege az, hogy a processzor két (vagy a 2 fő módon belül több köztes mód) módban tud futni. Ez a két mód a User és a Kernel mód. A kernelmód arra szolgál, hogy futtassa az operációs rendszert és lehetővé tegye az utasítások végrehajtását. A felhasználói (user) mód feladata az alkalmazások futtatása, de nem engedi bizonyos érzékeny utasítások végrehajtását (mint például a közvetlen beavatkozást a cache-be). Minden program a processzor user módjában fut. Ha a CPU user módban fut, bizonyos gépi utasításokat nem hajlandó végrehajtani -> (privilegizált utasítások) Azokat az utasításokat nevezzük privilegizált utasításoknak, amiket csak rendszer módban szabad végrehajtani. Privilegizált utasításo többek között a CLI, IN, OUT utasítások és a védett memóriacímek írása, olvasása. Ha egy alkalmazás privilegizált utasítást akar végrehajtani, akkor védelmi hiba lép fel, és meghívódik az operációs rendszer hibakezelő eljárása.

15.Mi a privilegizált gépi utasítás és mi a szerepe az operációs rendszerek szempontjából?

Ha a CPU user módban fut, bizonyos gépi utasításokat nem hajlandó végrehajtani -> (privilegizált utasítások) Azokat az utasításokat nevezzük privilegizált utasításoknak, amiket

csak rendszer módban szabad végrehajtani. Privilegizált utasítások többek között az összes I/O utasítás és a védett memóriacímek írása, olvasása. Ha egy alkalmazás privilegizált utasítást akar végrehajtani, akkor védelmi hiba lép fel, és meghívódik az operációs rendszer hibakezelő eljárása.

16.Mi a megszakítás és hogyan történik a feldolgozása?

INT jel alapján történik, forrás sok minden lehet.

- 1: megszakítási kérelem fogadása
- 2: elbírálás
- 3: végrehajtás alatt álló program befejezése, környezetének mentése, IP/PC tartalmának kimentése a verembe
- 4: megszakítást kérő rutinnak megfelelő prioritás beállítása
- 5: megszakítás végrehajtása
- 6. Kernel módba váltás
- 7. A megszakítást kiváltó folyamat elvégzése
- 8: az eredetileg futó folyamathoz tartozó prioritási szint visszaállítása
- 9: az eredetileg futó folyamat környezetének visszatöltése

17.Mit jelent az I/O megszakítás?

Az I/O adapter küld az operációs rendszer felé, jelezvén, hogy az I/O a működését. Pl.: a nyomtató jelez a rendszernek, hogy a nyomtatás elkészült.

18.Mit jelent a szoftveres megszakítás (INT) és mi a jelentősége az operációs rendszerek szempontjából?

Létezik az úgynevezett **szoftveres megszakítás** is, mely egy utasítás hatására váltja ki a folyamatot (pl. INT 134), csak értelemszerűen ehhez nem kell a megszakítás vezérlő. A szoftveres megszakítás folyamatát a fentebb leírt ***(16.kérdés)*** 3.-9. lépések végrehajtása jelenti. Szoftveres megszakításokkal tipikusan a BIOS vagy a DOS szolgáltatásait érhetjük el. A szoftveres megszakításhoz a processzornak saját parancsa van az utasításkészletben. Úgy működik, mint egy tényleges megszakítás. Olyan mint a Call (rendszerhívás), csak annyival több, hogy Kernel módba vált a processzor (szubrutin hívás).

19.Mit jelent a rendszer hívás?

Ha a futó (felhasználói) program valamilyen rendszerszolgáltatást igényel, ezt rendszerhívás formájában teheti meg. Általában assembly szintű utasításként jelen van az architektúrában. (INT, Tcc, SC, stb.). Magas szintű, rendszerprogramok írására szánt programozási nyelvekbe is beépítették. (C). Folyamata:

- INT gépi utasítás,
- paraméter átadás, funkció kódok,
- paraméter ellenôrzés ,
- végrehajtás,
- vezérlés visszaadása.

20.Melyek egy (idealizált) operációs rendszer főbb komponensei?

Folyamat kezelés (Process management): egy végrehajtás alatt levő program. A folyamatnak bizonyos erőforrásokra (így pl. CPU idő, memória, állományok, I/O berendezések) van szüksége,hogy a feladatát megoldhassa.

Az operációs rendszer az alábbi tevékenységekért felel a folyamatok felügyeletével kapcsolatban:

- Folyamat létrehozása és törlése.
- Folyamat felfüggesztése és újraindítása.
- Eszközök biztosítása a folyamatok szinkronizációjához és kommunikációjához.

Memória kezelés (gazdálkodás): Memória (főtár) kezelés (gazdálkodás). Az operatív memóriát bájtokból (szavakból) álló tömbnek fogjuk tekinteni, amelyet a CPU és az I/O vezérlő megosztva (közösen) használ. Tatalma törlődik a rendszerkikapcsolásakor és rendszerhibáknál. Az operációs rendszer a következő dolgokért felelős a memóriakezelést illetően:

- Nyilvántartani, hogy az operatív memória melyik részét ki (mi) használja.
- Eldönteni, melyik folyamatot kell betölteni, ha memória felszabadul.
- Szükség szerint allokálni és felszabadítani memória területeket a szükségleteknek megfelelően

Másodlagos tár kezelés: Mivel az operatív tár (elsődleges tár) törlődik (és egyébként sem alkalmas arra, hogy minden programot tároljon), a másodlagos tárra szükség van. Legelterjedtebb a merevlemez. Az operációs rendszer a következő dolgokért felelős a másodlagos tárkezelést illetően:

- Szabad hely kezelés
- Tár-hozzárendelés
- Lemez elosztás (scheduling)

I/O rendszer kezelés: Az I/O rendszer az alábbi részekből áll:

- Puffer (Buffer/Cache) rendszer
- Általános készülék-meghajtó (device driver) interface
- Speciális készülékek meghajtó programjai

Fájl kezelés: Egy fájl kapcsolódó információ (adatok) együttese, amelyet a létrehozója definiál. Általában program- (különböző formák), vagy adatfájlokkal dolgozunk. Az operációs rendszer a következő dolgokért felelős a fájl kezelést illetően:

- Fájl létrehozása és törlése
- Könyvtár létrehozása és törlése
- Fájlokkal és könyvtárakal történô alap-manipulációhoz nyújtott támogatás.
- Fájlok „leképezése” a másodlagos tárba.
- Fájlok mentése valamilyen nemtörlôdô, stabil adathordozóra.

Védelmi rendszer: Védelem általában valamilyen mechanizmusra utal, amelynek révén mind a rendszer-, mind a felhasználói erőforrásoknak a programok, folyamatok, vagy felhasználók által történő elérése felügyelhető, irányítható. A védelmi mechanimusnak tudnia kell:

- különbséget tennie authorizált (jogos) és jogtalan használat között,
- specifikálni az alkalmazandó kontrolt,
- szolgáltatni a korlátozó eszközöket.

Hálózat-elérés támogatása (elosztott rendszerek): Egy elosztott rendszer processzorok adat és vezérlő vonallal összekapcsolt együttese, ahol a processzoroknak nincs közös memóriájuk

és órájuk. (lokális memória, óra). Az adat- és vezérlő vonalak egy kommunikációs hálózat részei. Az elosztott rendszer a felhasználóknak különböző osztott erőforrások elérését teszi lehetővé. Az erőforrások osztott elérése lehetővé teszi:

- a számítások felgyorsítását,
- a jobb adatelérhetőséget,
- a nagyobb megbízhatóságot.

Parancs interpreter (al)rendszer: Az operációs rendszernek sok parancsot ún. vezérlő utasítás formájában lehet megadni. Ezek a vezérlő utasítások az alábbi területekhez tartozhatnak, mint folyamat létrehozás és kezelés, I/O kezelés, másodlagos tár kezelés, operatív tár kezelés, fájl rendszer elérés, védelem, hálózat kezelés, stb. Az operációs rendszernek azt a programját, amelyik a vezérlő utasításokat (be)olvassa és interpretálja a rendszertől függően más és más módon nevezhetik:

- vezérlő kártya interpreter (control-card Job Control, JC)
- parancssor interpreter (command-line)
- héj (burok, shell) (UNIX)

21.Melyek egy (idealizált) operációs rendszer főbb szolgáltatásai?

- Program végrehajtás (programbetöltés és futtatás)
- I/O műveletek (fizikai szint: blokkolás, pufferezés)
- Fájl-rendszer manipuláció (r,w, c, d)
- Kommunikáció – a folyamatok közötti információ csere (ugyanazon, vagy különbözô gépeken) Shared memory – Message passing.
- Hiba detektálás (CPU, memória, I/O készülékek, felhasználói programok, ...)

Nem közvetlenül a felhasználó támogatását, hanem a hatékonyabb rendszermûködést segítik:

- Erőforrás kiosztás – multiprogramozás, többfelhasználós működés
- Accounting – rendszer és felhasználói statisztikák.
- Védelem – minden erôforrás csak az operációs rendszer felügyelete mellett érhető el.

22.A rendszerprogramok szerepe az operációs rendszer használatánál!

A rendszerprogramok kényelmes környezetet teremtenek a programfejlesztéshez és program végrehajtáshoz. Egy lehetséges osztályozásuk:

- Fájl manipuláció
- Státus információ
- Fájl módosítás
- Programozási nyelv támogatás
- Program betöltés és végrehajtás
- Kommunikáció
- Alkalmazói programok ...

Felhasználói szemszögbôl nézve (egyes nézetek szerint) az operációs rendszer sokszor a rendszerprogramok együttesével azonosítható.

23.Mit jelent a mechanizmus és a politika az operációs rendszer tervezése és implementációja során?

A mechanizmusok meghatározzák, hogy hogyan kell valamit csinálni, a politikák pedig, hogy mit kell csinálni. (példa: timer megszakítás.) A mechanizmus és a politika különválasztása nagyon fontos; maximális rugalmasságot teremt, ha a politika késôbb megváltozik. Rendszer implementálás: Régen assembly nyelven írták az operációs rendszereket, ma már ez nem igaz; magas szintű nyelven is megírhatók. (pl.: MCP/Borroughs Þ Algol; MULTICS ÞPL/1; UNIX, OS/2, WinNT Þ C) A magas szintű nyelven írott kód gyorsabban elkászíthetô, kompaktabb, könnyebben áttekinthető és nyomon követhető (debug). A magas szintű nyelven írt rendszer portábilis, más hardverre könnyen átvihető. (pl.: Unix, Linux). A szűkkeresztmetszeteket meghatározva a gépi kód korrigálható (patch, service pack).

24.Mi a rendszergenerálás?

Az operációs rendszereket úgy tervezik, hogy azok működőképesek legyenek egy géposztály minden gépén. Ehhez azonban minden egyes számítógép-környezetre az operációs rendszert konfigurálni kell. Ezt a műveletet rendszergenerálásnak nevezzük. Program segíti (SYSGEN, setup, install). Egy SYSGEN program informálódik a hardver rendszer specifikus konfigurációjáról. (pl. Milyen processzor, aritmetika vehető alapul; processzorok száma, típusa, stb.; rendelkezésre álló memória mérete; perifériális berendezések típusai; megszakítási rendszer tulajdonságai). Tisztázni kell, hogy a rendszer mely szolgáltatásai tényleg szükségesek: multiprogramozási stratégia, hálózat elérés, stb. Booting, bootstrap loader – rendszer betöltés.

25.Mit jelent a folyamat (processzus)? Milyen jellemzői vannak?

Végrehajtás alatt álló program. Jellemzői:

- aktív
- memóriában van/volt
- regiszterek aktuális értéke
- program counter aktuális értéke
- verem tartalma (kimentett értéken kívül pl. lokális, globális változók értékei)

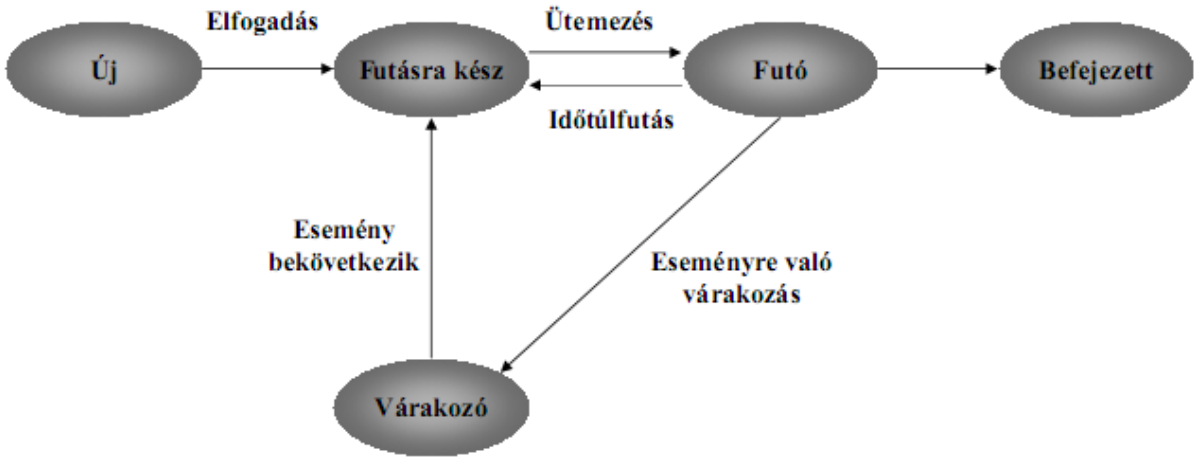
26.Mit jelent a processzus vezérlő blokk (PCB)?

Speciális adatszerkezet. Minden processzhez tartozik egy item, mely a processz jellemzőit tárolja.

- Processzus állapot (process state) : new, ready, running, waiting, halted, sleeping, …
- Program címszámláló (PC) értéke
- CPU regiszterek tartalma
- memória foglalási adatok
- account/user adatok
- I/O státusz információ (a folyamathoz rendelt I/O erôforrások, állományok listája)

Linuxnál a fájlrendszeren tárolódik, a /PROC-ban.

27.A folyamat állapotai. Hogyan változnak (mely állapotok között lehet átmenet)?



Running állapotba Ready-ből lehet kerülni.

28.Processzus állapot-információk egyes konkrét rendszerekben (UNIX, WIN_NT).

UNIX: PROC, PS

PID (Process ID): Folyamat azonosító. Ha befejeződik egy folyamat, azt a PID-et nem osztja ki még egyszer.

Thread (szál): hány szálon fut a processz, hányszor indult újra a folyamat, hányszorosan használjuk.

Nice: a felhasználónak lehetősége van a folyamatának prioritásának a csökkentésére (szinte senki se használja)

Size: Címtartomány mérete

Res: rezidens méret, aktuális kódrészlet memóriamérete

Sleep: waiting speciális változata. A programok memóriában vannak, kérésre várnak. pl.: daemonok: kérésre szolgáltatást indít.

Command: az a parancs, mely alapján a szóban forgó folyamat létrejött

Time: meddig használta a folyamat a CPU-t

TT: van-e hozzá programablak

S: state – állapot: S=sleep, R=Ready

%CPU: Cpu kihasználtság aránya

%MEM: Memória kihasználtság aránya

PPID: annak a folyamatnak az azonosítója, amely kérte a szóban forgó folyamatot az elinduláshoz.

Processzust létrehozni CSAK az operációs rendszer tud.

Winnt: taskmgr (Feladatkezelő), vagy Sys Internals hasonló programjai

29.Mit jelent a kaszkád termináció?

A szülő gyermek processzus párnál van jelen. Ha a szülő folyamat befejeződik, akkor a rendszer a gyermek folyamatokat is leállítja (le kell állniuk). Kaszkád termináció kivétel: nohup.

30.Milyen sorokat kezel az operációs rendszer a processzusütemezésnél?

- job queue (munka sor)
- ready queue (készenléti sor)
- device queue (berendezésre váró sor)

31.Mi a rövid távú processzus ütemezés célja?

Az hogy mindig legyen egy olyan futó folyamat, mely a processzort kihasználja. Melyik folyamat kapja meg következő alkalommal a CPU-t, gyorsaság lényeges

32.Milyen alapfeladatokat old meg az operációs rendszer a processzusokkal kapcsolatosan?

- Processz indítása (valami/ki kérésére)
- processz megállítása: jelzi a folyamat, hogy befejeződött, visszaadja az erőforrásokat, amelyeket a rendszer adott. 2 fajtája van:
  - EXIT
  - a szülő folyamat kéri a bezárást.
- Az oprendszer támogatást nyújt a folyamatok közti kommunikációhoz és kooperációhoz (együttműködéshez).

33.Mit jelent a szál? Milyen előnyei vannak a szálak alkalmazásának?

A szál egy újraindított programkód. A kódból csak 1 van a memóriában, csak a PC/IP-t és a regisztereket kell megjegyezni és a kód újraindul. újra felhasználjuk ugyan azt a kódot, memóriát spórolunk. Logikai háttere, hogy 1 ágon a végrehajtás várakozik, 1 másikkal tudok működtetni (Pl.: fájl szerver, vagy böngésző)

34.Milyen szolgáltatásokat nyújt az operációs rendszer a folyamatok kommunikációjához?

Processzusok közötti kommunikáció (IPC)
Az IPC olyan mechanizmust jelent, amely lehetővé teszi, hogy folyamatok egymással kommunikáljanak, és akcióikat összehangolják, szinkronizálják.
Üzenő rendszer: a folyamatok úgy kommunikálnak, hogy nem rendelkeznek közösen használható memóriával. IPC két műveletet nyújt:
send (message) az üzenetek lehetnek fix, vagy változó hosszúak
receive (message) példák (UNIX rendszerhívások: send, sendmsg, socket, recv, recvfrom, …)
Ha a P és Q folyamatok kommunikálni szeretnének, akkor szükségük van egy kommunikációs vonalra (communication link) A kommunikációs vonal implementációs kérdései

- Fizikai implementáció: megosztott memória, hardver busz, hálózat, ...
- Logikai implementáció:
- Hogyan épül fel a link?
- Tartozhat-e egynél több folyamathoz?
- Két folyamat között hány élő link lehet? ... üzenet mérete, mozgás iránya, stb

35.Mit jelent a direkt és indirekt kommunikáció?

Direkt kommunikáció
send (P, message) küldj egy üzenetet P-nek (utasítás Q-ban)
receive (Q, message) fogadj egy üzenetet Q-tól (utasítás P-ben)
A kommunikációs vonal ebben az esetben automatikusan épül fel a két folyamat között (PID ismerete szükséges!) A vonal pontosan két folyamat között létezik. Minden egyes folyamatpár között pontosan egy link létezik Egy, vagy többirányú is lehet. Termelő-fogyasztó példa Szimmetrikus/asszimetrikus címzés
Indirekt kommunikáció

- send(A, message) küldj egy üzenetet az A Mail-boxba (utasítás Q-ban)
- receive(A, message) olvass ki egy üzenetet az A Mail-boxból (utasítás P-ben)
- A kommunikációs vonal abban az esetben épül fel a két folyamat között, ha közösen használhatják az A Mail-boxot (PID ismerete nem szükséges!)
- A vonal több folyamat között létezik (mindenki, aki A-hoz hozzáférhet!). Minden egyes folyamatpár között több link is létezik, létezhet (Mail-box-függô).
- Egy, vagy többirányú is lehet.
- „Ki kapja az üzenetet?" - probléma.
    - Pufferelés (a link által egyidőben befogadott üzenetek száma)
    - Zéró kapacitás (0 üzenetet tárol a link; szinkronizáció kell, „rendezvous")
    - Korlátozott kapacitás (n üzenet lehet a linken)
    - Korlátlan kapacitás
    - Más: delay, reply, RPC (remote procedure call)
    - Kivétel-kezelés, kivételes helyzetek
    - processzus felfüggesztés
    - Elveszett üzenetek
    - hibás (scrambled) üzenetek

36.Mit jelent a végtelen kapacitású puffer?

A végtelen kapacitású puffernél a rendszer nem ellenőrzi, hogy a pufferbe betöltendő adat bele fér-e.

37.Mit jelent a preemptív processzus ütemezés?

Beavatkozó ütemezés, az operációs rendszer ha úgy dönt, elveszi a processztől a CPU-t.

38.Értelmezze a processzusütemezéshez kapcsolódó alábbi fogalmakat: végrehajtási idő, várakozási idő, válaszidő!

Végrehajtási (turnarund, fordulási) idő: várakozás a memóriába kerülésre + készenléti sorban töltött idő + CPU-n töltött idő + I/O-val töltött idő.
Várakozási idô (waiting time): készenléti sorban (ready queue) töltött idő
Válaszidô (response time): a „kérés benyújtásától" az elsô válasz megjelenéséig (nem output!) Eltelt idő. (Interaktív rendszerekben a turnaround nem jó jellemző.)

39.Melyek az FCFS processzus ütemezés előnyei és hátrányai?

Igény bejelentési sorrend szerinti kiszolgálás (first-come, first-served =FCFS), mindenki sorra kerül. Hátrány: az átlagos várakozási idő nagy szórása, konvoj effektus (Pl.: 4-es úton a szénásszekér)

40.Van-e optimális processzus ütemezési stratégia?

Van, a legrövidebb először. Azt a processzt engedjük a CPU-hoz, melynek a következő helyfoglalási ideje a legrövidebb. Melyik fogja a legrövidebb ideig lefoglalni a CPU-t azt előre nem tudjuk.

41.Mit jelent az exponenciális átlagolás? Mondjon példát az alkalmazására!

Megpróbáljuk megjósolni a következő helyfoglalási időt. Az eddigi viselkedések alapján kiszámoljuk a következő futás várható idejét. A mai rendszerek ezt használják.
Definíciója:

Jelölje t_n az n-edik CPU foglaltsági ciklus hosszát, τ_{n+1} a következônek jósolt foglaltsági periódus hosszát.

Definiáljuk a következô foglaltság várható hosszát: Legyen $1 \geq \alpha \geq 0$ és

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\, \tau_n$$

Alkalmazási példa: A rövidebb igény először (shortest job first, shortest job next, SJF, SJN) kiszolgálás elve.

42.Mit jelent a prioritásos processzus ütemezési stratégia?

Számokat rendelünk a processzekhez, minél kisebb ez a szám és a foglalási ciklus, annál nagyobb a prioritás. Az SJF, ill. FCFS is prioritásos kiszolgálásnak tekinthető! Belső és külső prioritás. Problémák: végtelen indefinit blokkolódás = éhezés, éhhalál. Példa: MIT 1973 IBM 7094: 1967 óta várt egy processzus a CPU-ra!! Megoldás: aging (unix példa)

43.Mit jelent a körleosztásos (Round Robin) processzus ütemezési stratégia?

Minden processzus sorban q ideig (q=10-100 millisec.) használhatja a CPU-t. n processzus esetén a maximális várakozási idő (n-1)/q. Ha q kicsi: ->FCFS, ha q nagy:-> kontextus váltási költség relatíve megnő! Egy bizonyos idő múlva az ütemező elveszi a feladattól a processzort /preemptív/ és az addig futó folyamatot a sor végére küldi és a következő folyamatot indítja egy bizonyos időre. Hátránya hogy sok az adminisztrációja.

44.Mit jelent a többsoros processzus ütemezés?

A célja a szálak egyenletes elosztása,a prioritás korrekt figyelembevétele. Többprocesszoros ütemezéskor használt paraméterek: Processzor megfelelés, ideális processzor, következő processzor

45.Mit jelent a processzus ütemezés esetén a visszacsatolásos sor?

Round Robinoknál előfordulhat, hogy pár folyamat abba a sorban ahol van „nem érzi jól magát” és akkor másik sorba kerül.

46.Mit jelent az „éhezés” (starvation)?

Prioritásos ütemezésnél a kiszolgálóhoz nem jut hozzá a sor.

47.Mi a kritikus szakasz probléma?

Az a kódrészlet, mikor egy vele konkruens kód ugyan azt az adatot kezeli. Az utasítások ilyenkor összefésülődnek az adatok sérülésének megelőzése érdekében.

48.Milyen követelményeket kell kielégítenie a kritikus szakasz kezelésének?

A kölcsönös kizárást (nem lehet egynél több processzus a kritikus szakaszában), a progressziót (a kiválasztás (futásra) nem késleltethető határozatlan ideig), és a várakozás elvét (minden igény kiszolgálása megkezdődik korlátozott számú kritikus szakasz végrehajtása után).

49.Mit jelent a processzusok szinkronizációja kapcsán a kölcsönös kizárás elve?

Azt, hogy egyszerre csak egy processzus hajthassa végre a kritikus szakaszát.

50.Mi az „entry” és „extern” kódszekciók funkciója a kritikus szakasz kezelésében?

Entry szekció: eldönti, hogy a processz beléphet-e a kritikus szakaszába
Exit szekció: közölje a konkruensekkel, hogy ő már befejezte a műveleteket az adatokon

51.A szemaforok mint szinkronizációs primitívek (eszközök).

Szemaforok (Dijkstra, 1965)

- Aktív várakozás (busy waiting) problémája
- Szemafor (S)
  - integer változó
  - csupán az alábbi két elemi (atomikus) mûvelet hajtható rajta végre
    wait(S): S:=S-1; if S<0 then block(S)
    signal(S): S:=S+1; if S£ 0 then wakeup(S)
- block(S): felfüggeszti a hívó processzus végrehajtását
  - a processzust hozzáadja az S szemaforon alvó processzusok sorához
- wakeup(S): reaktivál pontosan egy, korábban block(S) hívást kezdeményezô folyamatot
  - az S szemaforon alvó processzusok sorából egyet felébreszt

52.A monitor, mint szinkronizációs primitív (eszköz).

magas szintű szinkronizációs eszközök (szikronizációs primitív), amelyek lehetővé teszik egy absztrakt adattípus biztonságos megosztását konkurens processzusok között. Formálisan: a monitor eljárások, változók és adatszerkezetek együttese, amelyek egy speciális csomagba vannak integrálva. A processzusok hívhatják a monitorban levő eljárásokat, de nem érhetik el közvetlenül annak belsô adatszerkezetét.
Speciális típus: condition
Speciális mûveletek: x.wait, x.signal

A monitor azt kívánja biztosítani, hogy egy idôben csak egy processzus legyen aktív (rajta). Megvalósítás: pl. szemaforokkal.

53.A kritikus régió, mint szinkronizációs primitív (eszköz).

Magas szintű szinkronizációs eszköz. Egy megosztott változó v, amelynek típusa T, deklarációja: var v: shared T;
A v változó csak az alábbi alakú utasításokon keresztül érhetô el: region v when B do S; ahol B egy logikai kifejezés. Amíg az S utasítás végrehajtás alatt áll, másik processzus nem érheti el a v változót. Ha a processzus megpróbálja végrehajtani a region utasítást, a B logikai kifejezés kiértékelődik. Ha B igaz, az S utasítás végrehajtódik. Ha hamis, akkor a processzus végrehajtása késleltetődik addig, amíg a kifejezés igaz nem lesz és egyetlen más processzus sincsen a v-hez kapcsolt régióban.
Példa: korlátos puffer problémája. Implementáció: pl. szemaforokkal

54.Ismertesse az írók és olvasók problémáját!

Egy adatot, állományt több processzus megosztva, párhuzamosan használ, egyesek csak olvassák, mások csak írják. Hogyan biztosítható az adatok konzisztenciája?
Stratégiák:
1. Minden olvasó azonnal hozzáférhet az adatokhoz, hacsak egy író nem kapott már engedélyt az írásra.
2. Egy író azonnal írhat, ha erre kész és más írók éppen nem írnak.
Mindkét stratégia éhezéshez (starvation) vezethet, de van megoldás.

55.Ismertesse az „vacsorázó filozófusok” problémáját. Miért/hogyan vezethet holtponthoz?

Egy kör alakú asztal mellett öt filozófus ül, mindegyik előtt van egy tányér rizs és a szomszédos tányérok között egy-egy evő-pálcika. Evéshez a filozófus a saját tányérja melletti két evőeszközt használ-hatja úgy, hogy ezeket egymás után kézbe veszi. Ha befejezte az étkezést, visszateszi az eszközöket, és gondolkodni kezd. Majd újra megéhezik, stb. Példa számos konkurencia kontroll problémára. Mivel csak a szomszédos tányérok között 1 evőpálcika van, előfordulhat, hogy valamelyik filozófus nem jut pálcikához, és éhen marad.

56.Ismertesse a Bakery –algoritmust!

Bakery algoritmus (kritikus szakasz n processzusra)
Minden processzus kap egy sorszámot, mielőtt belép a kritikus szekciójába (ticket). A legkisebb sorszámú processzus hajthatja végre a kritikus szekcióját. Ha P_i és P_j sorszáma megegyezik, akkor a kisebb indexű jogosult a kritikus szekciójába belépni.
A sorszámozó rendszer mindig monoton növekvő sorszámokat generál. Pl.: 1,2,2,3,3,3,4,4, ...
Jelölés: „<” jelentse a (ticket #, process ID) párra definiált lexikografikus rendezést.