Operációs rendszerek 1

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

A beállítási idő azon időtartam, amely ahhoz szükséges, hogy a működéshez egy eszközt vagy egy készüléket előkészítsünk. Egy számítógép működési szakaszai közötti idő, mely pl. lemez vagy szalag cseréjéhez szükséges. A beállási idő az az idő, ami ahhoz szükséges, hogy egy jel az egyik állapotából a másik állapotába kerüljön.

A korai rendszereket nagyméretű gépek jellemezték melyet konzolról irányítottak és a programozó egyben az operátor is. A rendszer egyszerre egy felhasználót tudott kezelni és lyukszalaggal vagy lyukkártyával megvalósított adatbevitel és kivitel létezett. A processzor nincs kihasználva, és minden felhasználó, aki gép elé jutott ezt a folyamatot végigjátszotta. Tehát míg a felhasználó végigjátssza a fordító betöltését az a beállítási idő. Ezt úgy tudták csökkenteni, hogy a gép mellé egy operátort rendeltek, aki egyszer töltötte be a fordítót és az összegyűjtött algoritmusokat egymás után fordította le és adta ki szalagra vagy kártyára. Azt hogy az operátor szépen egymás után adja az utasításokat (kártyánként) hívjuk kötegelt feldolgozásnak. Tehát a beállítási idő kötegelt feldolgozással gyorsítható.

2. Mit jelent a „Moore törvény”?

Moore-törvénynek nevezzük azt a tapasztalati megfigyelést a technológiai fejlődésben, mely szerint az integrált áramkörökben lévő tranzisztorok száma – a legalacsonyabb árú, ilyen komponenst figyelembe véve, ami használható a számítási teljesítmény durva mérésére – körülbelül 18 hónaponként megduplázódik. A háttértárak sebessége azonban messze nem nő ilyen ütemben, erre megoldás a multiprogramozás kivitelezése. Ez nem jelenti azt, hogy a számítógép teljesítménye 18 havonta megduplázódik, ugyanis a teljesítmény nem csak a tranzisztorok számától függ. Ennek a növekedésnek egyszer (egyes becslések szerint 15-25 év múlva) azonban vége szakad. Az egy négyzetcentiméteren összesűrített elemek száma olyan nagy lesz, hogy az elemek mérete a nanométer-tartományba esik, azaz összemérhető lesz az atomok nagyságával.

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

Multiprogramozás kivitelezése:

- A processzornak egynél több programot kell végrehajtania
- A programok végrehajtásának sorrendje függ azok relatív prioritásától illetve attól, hogy várnak-e valamilyen I/O műveletre
- A megszakítás-kezelő rutin befejeztével a vezérlés nem feltétlenül kerül vissza ahhoz a programhoz, amelyik futása közben a megszakításkérés történt

Az egy négyzetcentiméteren összesűrített elemek száma olyan nagy lesz, hogy az elemek mérete a nanométer-tartományba esik. Ebben a nagyságrendben új fizikai törvények érvényesek, a kvantummechanika törvényei. E kérdések, s ezzel együtt egy új számítástechnikai rend megalapozásával foglalkozik a nanotechnológia rejtélyes, új tudománya, amelynek jelentőségét jelzi az, hogy az USA sok milliárd dolláros évi kerettel s több tízezer kutató kiképzésével készíti elő az új eredményeket, amelyek egyik célja éppen újfajta, rendkívül nagyteljesítményű számítógépek előállítása.

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

Ez köti össze a bus rendszert az I/O processzorral. Ha nagyobb a csatorna átviteli sebessége, csökken a kommunikáció idő a központi egység és az I/O berendezések között. Ezzel növekszik a számítógép átbocsátóképessége, gyorsabb lesz az információáramlás.

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

A kötegelt feldolgozást a korai rendszerek gyorsítására találták ki. Arra szolgál, hogy a felhasználók ne egyesével kezdjék az algoritmusait futtatni és ezzel a drága gépidőt pazarolni, hanem azt egy hozzáértő operátor kezébe adta, aki rendszerezte az azonos feladatokat és egy betöltési idővel lefutatta őket. Így a gép kihasználtsága növekedett, viszont a „programozók” kiszorultak a gépteremből.

Egy idő múlva az is elérkezett, hogy az operátor volt lassú, és néha hiba is becsúszott ezért a General Motors írt egy operációs rendszerecskét, ami Monitor névre hallgatott és állandóan a tárban futott. Ez azt jelentette, hogy az operátor már csak a perifériákat kezelte.

Majd megjelentek a segédszámítógépek, amelyek terminálként funkcionáltak, vagyis az oda begépelt programok mágnesszalagra rögzültek és az ment a nagy gépnek. A segédszámítógépek buták voltak és ezért nem futathatott rajta az ember algoritmust. Tehát az operátor vitte a szalagot a nagy gépnek és ott futatta. Így az adatátvitel a kezdeti időköz képest a töredéke lett. A szalag legnagyobb előnye az volt, hogy az olvasó fejet oda lehetett állítani ahová kellett anélkül, hogy az operátor beavatkozott volna. Az adatok itt is sorosan voltak tárolva, mint a kártyánál.

Hátránya: honnan tudja a Monitor hogy mi az adat és a program, hogy mi a „jobhatár”, illetve milyen természetű jobot kell futatnia (Fortran, Assembler). Ezt úgy küszöbölték ki, hogy egy új nyelvcsalád a parancsnyelv jött létre. Így a feladatokba beépített kódokat írtak: $JOB, $FTN, $RUN, $DATA, $END. Ezzel monitor látta, hogy job vagy adat, amit futtat és azt is látta, hogy hol a job határa. A jobok természet is különleges karakterek használatával adták meg.

Előny: csökken a beállítási idő.

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

- vezérlő kártya interpreter – felelős a vezérlőkártyák beolvasásáért és az értelmezésért
- betöltő (loader) – háttértárból a betöltő 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?

- A multiprogramozás célja: a CPU foglaltság (kihasználás) hatásfokának növelése
- A multiprogramozás lényege: egy időben több folyamat is az operatív tárban helyezkedik el, készen állva a CPU kiszolgálására. Ha egy folyamatnak várnia kell (Pl. I/O-ra), a CPU-t egy másik folyamat kapja meg.

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

- Az I/O-nak az operációs rendszer részéről történő teljes körű felügyelete (adatvédelem)
  - Az I/O-t az operációs rendszer nem egyszerűen támogatja, hanem a végrehajtásához elkerülhetetlen
  - Hardver feltételek (kernel/supervisor mode, privileged operations)
- Memóriagazdálkodás – a rendszernek fel kell osztani a memóriát a futó jobok között
  - Hardver feltételek (+segmentation)
- CPU ütemezés – a rendszernek választani kell tudni a futásra kész jobok között
- Készülék hozzárendelés (nem jut minden jobnak printer, lemez stb.)
  - Eszközkezelés – a felhasználói programok elől el kell fedni a perifériák különbözőségét, egységes kezelői felületet kell biztosítani
  - Megszakításkezelés – alkalmasnak kell lennie a perifériák felől érkező kiszolgálási igények fogadására, megfelelő ellátására
  - Rendszerhívás, válasz – a rendszer magjának ki kell szolgálni a felhasználói programok periféria igényeit úgy hogy, lehetőleg ne vegyék észre hogy nem közvetlenül használhatják a perifériákat. Erre szolgálnak a programok által küldött rendszerhívások, amelyekre a mag válaszokat küldhet.
  - Erőforrás kezelés – az egyes eszközök közös használatából adódó konfliktusokat meg tudja előzni, vagy ha bekövetkeztek, akkor azt tudja kezelni, feloldani
  - Processzorütemezés – az operációs rendszerek ütemező funkciójának a várakozó munkák között valamilyen rend szerint el kell osztani a processzor idejét, illetve vezérelni kell a munkák közötti átkapcsolási folyamatot
  - Memóriakezelés – úgy kell felosztania a memóriát a feladatok között, hogy azok egymást ne zavarják és az operációs rendszerben se tegyen kárt
  - Állomány és lemezkezelés – rendet kell tudni tartania a hosszabb távra megtartandó állományok között
  - Felhasználói felület – a parancsnyelvek feldolgozására alkalmas monitor fejlettebb utódja, ami közölni tudja a felhasználó kívánságait és a rendszer állapotáról információt adjon.

9. Mit jelent a SPOOL-ing?

- absztrakt periféria fogalom (Simultaneous Peripheral Operation On-Line)
- mialatt egy job végrehajtódik, az operációs rendszer beolvassa a következő jobot a kártyaolvasóról a lemezre (job queue), egy előző job által nyomtatni szánt adatokat a lemezről a printerre továbbítja
- megjelenik a job pool – olyan adatszerkezet, melyek segítségével az operációs rendszer kiválaszthatja a következő job-t, növeli a CPU kihasználtságot
- kevesebb a CPU veszteségi idő
  CPU tovább gyorsulásával és a mechanikus perifériák nem gyorsulásával egy új rendszer került, ami a SPOOL (**S**imultaneous **P**eripheral **O**peratios **O**n **L**ine) nevet kapta. Ez azt jelenti hogy kell egy olyan gyors tároló eszköz ami a beolvasandó adatokból a lehető legtöbbet tárolja és a kimenő adatok gyors rögzítésére és tárolására is alkalmas addig amíg a mechanikus rendszer nem képes feldolgozni azt. Ezt a funkciót a mágneses lemez tölti ki. Egyszerre több job végezhető és válogathat a CPU köztük (fontosság, kihasználtság), adatok újbóli feldolgozására ill. további feldolgozására képes, így azt névvel látja el.

10. Jellemezze az időosztásos rendszereket!

- (Time-sharing rendszerek), legfőbb jellemzője az interaktivitás
- A CPU váltakozva áll olyan jobok rendelkezésére, amelyek a memóriában, vagy a lemezen találhatók (CPU-t csak olyan job kaphatja meg, amely éppen a memóriában van)
- egy job a lemezről a memóriába, vagy a memóriából a lemezre betölthető/kimenthető az ütemezési stratégiának megfelelően
- a rendszer a felhasználók között online kommunikációt biztosít, ha az operációs rendszer befejezi egy parancs végrehajtását, a következő vezérlő-utasítást, nem a kártyaolvasóról, hanem a felhasználó klaviatúrájáról várja

- egy – adatokat és utasításokat tároló – online fájlrendszer kell, hogy a felhasználó rendelkezésére álljon.
  - a rendszer a program fejlesztését támogatja
  - a számítógépek információ kezelés eszközeivé váltak

11. Jellemezze a személyi számítógépes rendszereket!

- a teljes számítógép rendszer egy egyszerű felhasználó kizárólagos rendelkezésére áll
- tipikus konfigurációjú I/O berendezések (klaviatúra, egér, képernyő kijelző, kisteljesítményű nyomtató)
- előtérben a felhasználó kényelme és felelőssége
- sokszor adaptál – eredetileg nagygépes operációs rendszerekre kidolgozott információ technológiai megoldásokat (migráció)
- a felhasználó sokszor a számítógép kizárólagos tulajdonosa, felhasználója, így nincs szüksége fejlett CPU kiszolgáló és adatvédő szolgáltatásokra
- What you see is what you get
- IBM (DOS, Basic, Windows)

12. Jellemezze a párhuzamos rendszereket!

- Multiprocesszoros rendszerek több mint egy – szoros kommunikációs kapcsolatban lévő – CPU-val
- Szorosan kapcsolt/csatolt rendszerek – a processzorok közösen használják a memóriát és a rendszer órát. A kommunikáció a közös memória segítségével történik.
- Előnyei: megnövelt átbocsátó képesség, gazdaságosság, növekvő megbízhatóság (redundancia, graceful degradation, fail-soft 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 multiprocesszitás(maste-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)
- Ha több fukcionális egység ugyanazt a feladatot látja el és egy kiesik, akkor a rendszer tovább működik teljesítmény csökkenés árán.

13. Jellemezze a valós idejű rendszereket!

- Real-time rendszerek
- 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 tárban (RAM) vagy akár ROM-ban tárolják
  - Konfliktus az időosztásos rendszerekkel
- Szoft (puha valós idejű) rendszerek
  - Korlátozott szolgáltató programok az ipari kontroll, a robotika területén
  - A fejlett operációs rendszerszolgáltatásokat igénylő alkalmazásoknál a (Multimedia, VR) igen hasznosak

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

- Felhasználó mód – rendszer (supervisor, monitor mód):
  Létezik a pancsoknak egy korlátozottabb felhasználói üzemmódja (user mód), amelyet a felhasználói programok használnak.
  Másrészt létezik egy teljes utasításkészletet támogató üzemmód, a rendszer üzemmód (kernel mode, privilegized mode, system mode, supervisor mode), melyet csak az operációs rendszer használhat.
  Ha egy felhasználói folyamat olyan utasítást ad ki, melyet a processzor user módjában nem hajthat végre, akkor az utasítás csapdába esik (trap) és a vezérlés az operációs rendszerhez kerül.
- Módus bit:
- Privilegizált műveletek: Az I/O műveletek privilegizáltak. Hogyan hajthat végre egy felhasználói program I/O műveletet?

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

Privilegizált utasítások használata védve van a felhasználói programoktól. A kiváltságos utasítások rendszer funkciókat befolyásolnak (úgymint a rendszer regiszterek írása). Ezen utasításokat csak 0-s CPL-ű (legprivilegizáltabb) privilégium szinten

hajthatjuk végre. Ha egyet is az alábbi utasítások közül nem 0 CPL-en hajtunk végre, akkor általános védelmi hiba (#GP) generálódik.

16. Mi a megszakítás és hogyan történik a feldolgozása?

- A megszakítás (interrupt) átadja a vezérlést a megszakítás feldolgozó rutinnak. Ez általában a megszakítás vektor segítségével történik, amelynek megfelelő elemi tartalmazzák a megszakítási osztályokhoz tartozó feldolgozó rutin első végrehajtandó utasításának címét.
- A megszakítási rendszernek tárolnia kell a megszakított utasítás címét
- Egy megszakítás feldolgozása idején jelentkező további megszakítások letilthatók (mask out, disable), hogy el ne vesszenek (lost interrupt)
- A megszakítási jel forrását tekintve lehet külső (I/O,Timer,Hardver), vagy belső(szoftver megszakítás, trap)
- A megszakítást feldolgozó rutin (operációs rendszer része) közvetlen feladatai:
  - A CPU állapotának megőrzése,
  - A megszakítás okának, körülményeinek részletesebb elemzése,
  - Maszkolás,
  - Megszakított programhoz történő visszatérés megszervezése

17. Mit jelent az I/O megszakítás?

A berendezés vezérlő egységek, I/O processzorok egy-egy megszakítás generálásával jelezhetik a CPU-nak az I/O művelet befejezését. (DMA)

Külső megszakítás, ami az I/O egység felől jön, ami után átadja a vezérlést a megszakítás feldolgozó rutinnak. Pl.: kifogy a lap a nyomtatóból.

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

- A megszakítási jel forrását tekintve egy megszakítás lehet külsô (pl. I/O, Timer, Hardver, …), vagy belső (szoftver megszakítás, angolul: trap).
- Az operációs rendszer (megszakítás feldolgozó rutin) közvetlen feladatai:
  - A CPU állapotának megőrzése (a hardver ehhez minimális kezdeti támogatást nyújt);
  - A megszakítás okának, körülményeinek részletesebb elemzése;
  - A "maszkolás";
  - A megszakított programhoz történő "visszatérés" megszervezése.

19. Mit jelent a rendszer hívás?

- Szoftver által direkt módon kiváltott megszakítás. Speciális megszakítás az operációs rendszer szolgáltatásainak igénybevételére.
  - INT gépi utasítás,
  - paraméter átadás, funkció kódok,
  - paraméter ellenőrzés ,
  - végrehajtás,
  - vezérlés visszaadása.
- Ha a futó (felhasználói) program valamilyen rendszerszolgáltatást igényel, ezt rendszerhívás formájában teheti meg
  - Általában assembly szintű utasításként jelen van az architektúrában (INT,Tcc,SC)
  - Magas szintű, rendszerprogramok írására szánt programozási nyelvekbe is beépítették (C)
- Paraméterátadás módjai a rendszernek:
  - regiszterben
  - paraméter táblázatban
  - veremben
  - módszerek kombinálása, statusword-ok

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

- Folyamatkezelés (process management)
- Memóriakezelés (gazdálkodás)
- Másodlagos tár kezelés
- I/O rendszerkezelés
- Fájlkezelés
- Védelmi rendszer
- Hálózat-elérés támogatása
- Parancs interpreter rendszer

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

- Program végrehajtása (program betöltése és futtatása)
- I/O műveletek (fizikai szint: blokkolás, pufferezés)
- Fájl-rendszer manipuláció (r, w, c, d)
- Kommunikáció – a folyamatok közötti információ csere (ugyanazon, vagy különböző gépeken) Shared memory – message passing
- Hibadetektálás (CPU, memória, I/O készülékek, felhasználói programok)
- Nem közvetlenül a felhasználó támogatását, hanem a hatékonyabb rendszerműködést segítik
- Erőforrás kiosztás – multiprogramozás, több felhasználós működés
- Accounting – a rendszer és felhasználói statisztikák
- Védelem – minden erőforrás csak az operációs rendszer felügyelete mellett érhető el

22. A rendszerprogramok szerepe az operációs rendszer használatánál!

- Rendszerprogramok kényelmes környezetet teremtenek a programok fejlesztéséhez és a program végrehajtásához.
- Osztályozásuk:
  - Fájl manipuláció
  - státus manipuláció
  - fájlmódosítás
  - programozási nyelv támogatás
  - program betöltés és végrehajtás
  - kommunikáció
  - alkalmazói programok
- Felhasználói szemszögből nézve 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 (pl.: timer megszakítás)
- A mechanizmus és a politika különválasztása nagyon fontos, maximális rugalmasságot teremt, ha a politika később megváltozik

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. Ez a folyamat a rendszergenerálás. Program segíti (SYSGEN, setup, install)
- Egy SYSGEN program informálódik a hardver rendszer specifikus konfigurációjáról (pl. milyen a processzor, aritmetika vehető alapul, processzorok száma, típusa, rendelkezésre álló memória mérete, perifériális berendezések típusai, megszakítási rendszer tulajdonságai)
- Tisztázni kell, hogy a rendszer mely szolgáltatásai tényleg szükségesek: multiprogramozási stratégia, hálózat elérés
- Booting, bootstrap loader – rendszer betöltés

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

- Folyamat (processzus): végrehajtás alatt álló program
- Processzus jellemzői:
  - aktív
  - a memóriában volt/van
  - program címszámláló értéke
  - regiszterek értéke
  - lokális/globális változók értéke
  - verem állapota
  - core dump
- Betölthető program jellemzői:
  - passzív
  - lemezen tárolt
  - betöltési cím
  - belépési pont

26. Mit jelent a processzus vezérlő blokk (PCB)?
Más néven process control block, folyamatok vezérlése szolgál tárolja a:

- processzus állapotát (process state):
  - new (új)
  - ready (futásra kész)
  - running (futó)
  - waiting (várakozó)
  - halted
  - sleeping
  - terminated (befejezett)
- program címszámláló (PC) értékét
- CPU regiszterek tartalmát
- memóriafoglalási adatokat
- account/user adatokat
- I/O státus információkat (a folyamathoz rendelt I/O erőforrások, állományok listáját)

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

- új (new)
- futásra kész (ready)
- várakozó (waiting)
- futó (running)
- befejezett (terminated)
- halted, sleeping
- új -> felvéve -> futásra kész
- futó -> interrupt -> futásra kész
- futásra kész -> ütemező intézkedése -> futó
- várakozó -> I/O vagy eseménye befejeződése -> futásra kész
- várakozó -> I/O eredménye vagy esemény kell -> futó

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

- UNIX:
  - ps: a felhasználó aktuális terminálján futó process-eket jelzi ki
  - Top: teljes rendszerinformáció összegzést ad és az éppen futó folyamatok listáját
- NT: task manager

29. Mit jelent a kaszkád termináció?
Processzus megszűnése illetve megszüntetése (termination) során léphet fel.
A szülő folyamat felfüggesztheti a gyermek folyamatok végrehajtását (abort). Ennek egyik lehetséges oka, hogy a szülő befejezi működését. Az operációs rendszer legtöbb esetben nem engedi, hogy a gyermek tovább éljen, mint a szülő. Azt jelenti, hogy a szülővel együtt elhal az összes gyermeke is.

30. Milyen sorokat kezel az operációs rendszer a processzus ütemezésnél?

- Muka sor (Job queue)
- Készenléti sor (Ready queue)
- Berendezésre váró sor (Device queue)

31. Mi a rövid távú processzus ütemezés célja?

- Folyamatütemezés célja:
- Mindig legyen legalább egy processzus, amelyik képes és kész a processzort lefoglalni.
- Rövidtávú ütemező (Short term scheduler)
  - Meghatározza, hogy melyik folyamat kapja meg a következő alkalommal a CPU-t.
  - A gyorsaság a lényeges.

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

- Processzus létrehozása
  - Processzust csak processzus hozhat létre (a kezdeti betöltést kivéve).
  - Szülő létrehozhat gyermek folyamatokat, majd a gyermekek további gyermekeket (fastruktúra).
- Erőforrás megosztás
  - Szülő és gyermek közösen használ minden erőforrást.

  - A gyermek a szülőerőforrásainak egy részét használja.
  - Nincs közös erőforrás használat.
- Végrehajtás
  - Szülő és gyermek konkurens módon fut.
  - Szülő a gyermekre vár.
- Címtér (adress space)
  - A gyermek a szülő duplikáltja.
  - A gyermek betölt egy programot önmaga helyett.
- Processzus megszüntetése (termination)
  - A folyamat végrehajtja az utolsó utasítást, majd megkéri az operációs rendszer, hogy törölje.

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

- A CPU kiszolgálás egy alapegysége, jellemzői:
  - Program címszámláló
  - Regiszter készlet (tartalma)
  - Verem tartalma
- Egy szál megosztva használja a vele egyenrangú (társ-) szálakkal a kód szekcióját, adat szekcióját, operációs rendszer által biztosított erőforrásait, ezek együttesét task-nak nevezzük.
- Olyan mechanizmust szolgálnak, amely lehetővé teszi a szekvenciális processzusoknak a rendszer hívások blokkolását, s közben a párhuzamosság elérését.
- Előnye, hogy csökken a context-switch végrehajtására fordított idő!
  Egy több szálat tartalmazó task esetén, amíg az egyik szál várakozik (blokkolt) a másik futhat.

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

- Üzenőrendszer:
  A folyamatok úgy kommunikálnak, hogy nem rendelkeznek közösen használható memóriával.
- Az IPC olyan mechanizmust jelent, amely lehetővé teszi, hogy folyamatok egymással kommunikáljanak, és akcióikat összehangolják, szinkronizálják.
- IPC két műveletet nyújt:
  - Send (küldés) [Az üzenetek lehetnek fix, vagy változó hosszúak.]
  - Receive (fogadás)
- UNIX rendszerhívások: send, sendmsg, socket, recv, recvfrom, stb.
- Ha két különböző folyamat kommunikálni szeretne, akkor szükségük van egy kommunikációs vonalra.

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.
  - 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 akkor 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 függő)
  - Egy, vagy többirányú is lehet.
  - „Ki kapja az üzenetet?" - probléma.

36. Mit jelent a végtelen kapacitású puffer?

A puffer tényleges betelése utáni információ egyszerűen elvész. Tehát nem azt jelenti, hogy végtelen sok adat fér el a pufferben, hanem azt, hogy nem foglalkozik a puffer túlcsordulással. Ilyen információ-vesztés például az új adat figyelmen kívül hagyása vagy a régi adat felülírása a pufferben.

37. Mit jelent a preemptív processzus ütemezés?
Preemptív processzus ütemezésről beszélünk, ha az operációs rendszer elveheti a futás jogát az éppen futó folyamattól, azt készenléti állapotúvá teheti, és a CPU-t egy másik folyamatnak adhatja, azaz egy másik folyamatot indíthat el.

38. A processzus ütemezéshez kapcsolódó fogalmak: végrehajtási idő, várakozási idő, válaszidő.
- Végrehajtási idő (turnaround time): várakozás a memóriába kerülésre + készenléti sorban töltött idő + CPU-n töltött idő + I/O-val töltött idő.
- Várakozási idő (waiting time): A készenléti sorban töltött idő.
- Válaszidő (response time): A kérés benyújtásától az első válasz megjelenéséig (nem output) eltelt idő.

39. Melyek az FCFS (First Come First Served) processzus ütemezés előnyei és hátrányai?
- Igény-bejelentési sorrend szerinti kiszolgálás.
- Előnye:
  - Egyszerű biztos és mindegyik folyamat sorra kerül.
- Hátránya:
  - Az átlagos várakozási idő túl nagy.
  - Folyamatok egész sorát tartja fel (konvoj effektus).
  - Az erőforrások kihasználatlanok.

40. Van-e optimális processzus ütemezési stratégia?
- SJF (Shortest Job First)
  Olyan processzus ütemezési stratégia, amely a várakozó processzek közül a legrövidebb végrehajtási időt igénybevevőt választja ki, mint végrehajtandó processzt. Ez a megoldás minimalizálja a folyamatok közötti átlagos várakozási időt.
- SSTF (Shortest Seek Time First)
  Olyan stratégia, amely minimalizálja a lemezolvasó mozgását. Lényege, hogy azon processz követi a másikat, amelynek kódja a legközelebb áll az olvasófejhez, így fizikailag csökken a végrehajtási idő.

41. Mit jelent az exponenciális átlagolás? Mondjon példát az alkalmazására!
- Jelölje t_n az n-edik CPU foglaltsági ciklus hosszát.
- Jelölje t_{n+1} a következőnek jósolt foglaltsági periódus hosszát.
- Definiáljuk a következő foglaltság várható hosszát: Legyen 1 □□□□□□□0.
- Ekkor t_{n+1} = □□□□□t_n + (1–□) t_n.

42. Mit jelent a prioritásos processzus ütemezési stratégia?
- Prioritási sorrend szerinti kiszolgálás.
  - Prioritás: a processzushoz rendelt egész érték. A kisebb érték nagyobb prioritást jelent.
  - Prioritási függvény.
  - Belső és külső prioritás.
  - Preemptív, non-preemptív prioritási ütemezők.
- Problémák: végtelen indefinit blokkolódás = éhezés, éhhalál.
  - Példa: MIT 1973 IBM 7094: 1967 óta várt egy processzus a CPU-ra!
- Megoldás: aging (unix példa)
  - A folyamatokat fontosság szerint rangsorolja, így a magasabb rendű folyamatot a várakozási sor elejére teszi.
  - Az RR algoritmusnál nagyobb időszeletet biztosít, a kevesebb joggal rendelkező folyamatok háttérbe szorulnak. De előbb utóbb a kisebb prioritású folyamatok is lefutnak.

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 $\Rightarrow$ FCFS.
- Ha q nagy $\Rightarrow$ kontextus váltási költség 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. Nagy hátránya hogy sok az adminisztrációja.

44. Mit jelent a többsoros processzus ütemezés?
- Cél a szálak egyenletes elosztása
  - Prioritás korrekt figyelembevétele többprocesszoros ütemezéskor.
- Használt paraméterek:
  - Processzor affinitás

- Ideális processzor
- Következő processzor

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

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

- Végtelen indefinit blokkolódás.
- Túl sokáig vár egy processzus a CPU-ra.
- Egy folyamat az erőforrás kezelő stratégiája miatt beláthatatlan ideig nem jut erőforráshoz.

47. Mi a kritikus szakasz probléma?

- n folyamat verseng egy bizonyos (közös) adat használatáért.
- Mindegyik folyamatnak van egy kódszegmense, ahol ezt a bizonyos adatot eléri (olvassa / írja): ez az úgynevezett kritikus szakasz.
- Probléma: Hogyan biztosítható, hogy amíg egy processzus a kritikus szakaszát hajtja végre, addig egyetlen más processzus se léphessen be a saját kritikus szakaszába?

48. Milyen követelményeket kell kielégítenie a kritikus szakasz kezelésének?

- Kölcsönös kizárás (nem lehet egynél több processzus a kritikus szakaszában).
- Progresszió (a kiválasztás (futásra) nem késleltethető határozatlan ideig).
- Korlátozott várakozás (minden igény kiszolgálása korlátozott számú kritikus szakasz végrehajtása után).

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

Nem lehet egynél több processzus a kritikus szakaszában.

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

Funkciójuk, hogy biztosítva legyen az, hogy amíg egy processzus a kritikus szakaszát hajtja végre addig egyetlen más processzus se léphessen be a saját kritikus szakaszába.

51. A szemaforok mint szinkronizációs primitívek (eszközök).

- Tegyük fel, hogy van két folyamat ami egy közös memória területet használ és azon keresztül kommunikál egymással.
- Az egyik folyamat a termelő ami adatokat ír a memóriába a másik a fogyasztó aki onnan adatot olvas ki.
- Nyilván a memória kölcsönös kizárást igénylő erőforrás hiszen amíg termelő a memóriába ír addig a fogyasztó onnan nem olvashat ki illetve fordítva.
- Rendeljünk a folyamatokhoz egy szemafortot. Ez egy változó a memória területén amit mind a két folyamat elér és olvasni tud.
- Legyen a szemafort érteke 0 ha a folyamat használja a területet és 1 ha nem használja a területet valamelyik folyamat. Mivel kétféle dolgot jelez a szemafort ezért azt bináris szemafortnak nevezzük.

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 közvetlenül nem érhetik el belső adatszerkezetét.
- Speciális típus: condition.
- Speciális műveletek: x.wait, x.signal.
- A monitor azt kívánja biztosítani, hogy egy időben csak egy processzus legyen aktív (rajta).
- Megvalósítás: pl. szemaforokkal.

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

- Magas szintű szinkronizációs eszköz.
- Legyen v egy megosztott változó, amelynek típusa T. Deklarációja: **var v: shared T;**
- A v változó csak az alábbi alakú utasításokon keresztül érhető 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.
- A filozófus a saját tányérja melletti két evőeszközt használhatja úgy, hogy mind a kettőt kézbe veszi.
- Ha befejezte az étkezést, visszateszi az eszközöket, és gondolkodni kezd.
- Majd újra megéhezik, stb.
- Példa számos konkurencia kontroll problémára:
  5 kínai bölcs ül egy kerek asztal körül. Mindegyik előtt ott egy tányér rizs és a szomszédos tányérok között egy-egy pálcika. Az evéshez két pálcika kell, ezért egy bölcsnek mind a jobb-, mind a baloldali pálcikát meg kell szereznie, hogy elkezdhessen enni. Ha egy bölcs eszik, akkor egyik szomszédja sem tud, hiszen legalább az egyik pálcikája hiányzik hozzá. Ha mindegyik bölcs egyszerre jut arra a döntésre, hogy először a bal-, majd a jobboldali pálcikát veszi kézbe, mindegyiknek csak egy pálcika jut senki sem tud enni, HOLTPONT-ra jutnak. Ebben az esetben feltehetően születik megoldás, hiszen bölcsekről van szó. Ez azonban az egymásról mit sem tudó folyamatok esetén nem várható. Egy felsőbb erőnek, az operációs rendszernek kell rendet teremtenie, például úgy, hogy egyiktől elveszi a pálcikát és a szomszédjának adja.

56. Ismertesse a Bakery – algoritmust (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.

57. Mit jelent a logikai és fizikai címtér?

- Logikai cím A CPU által generált cím(virtuális cím).
- Fizikai cím A Memory Management Unit (MMU) által generált cím (reális cím).
- Fordítási és betöltési időben csak a logikai címtér (címhozzárendelés) érhető el.
- Logikai címet az MMU (Memory Management Unit) képezi le fizikai címmé.

58. Mit jelent a tárallokálásnál a „worst fit" (legkevésbé megfelelő) stratégia? (link FazekasG)

- Foglaljuk le a leghosszabb lyukat (ha az elég hosszú).

59. Mit jelent a tárallokálásnál a „firs fit" (legelső megfelelő) stratégia? (link FazekasG)

- Foglaljuk le az első lyukat, amely elég hosszú.

60. Mit jelent a tárallokálásnál a „best fit" (leginkább megfelelő) stratégia? (link FazekasG)

- Foglaljuk le az elég hosszú lyukak közül azt, amelynek hossza legközelebb esik a szükséges hosszhoz.

61. Mi a külső fragmentáció?

- Azt jelenti, hogy az összességében rendelkezésre álló hely elegendő, de ez apró részekben jelenik meg, ahová viszont nem fér be egyetlen folyamat sem.

62. Mi a belső fragmentáció?

- Az az eset, amikor tárkezelési eljárások egy csoportjánál a rendszer nagyobb tárterületet foglal le, mint amennyi szükséges, ezért a lefoglalt terület egy része kihasználatlan marad.

63. Hogyan épül fel a laptábla?

- Minden logikai laphoz tartalmaz egy bejegyzést a logikai lapot tartalmazó keret fizikai címéről + egyéb információk.

64. Mit jelent az, hogy egy lap megosztott?

- Közös lap megosztva használható több processzus által (pl: szövegszerkesztők, compilerek).

65. Mit jelent az invertált laptábla?

- Minden egyes fizikai laphoz (frame) tartalmaz egy bejegyzést (entry). Egy bejegyzés az adott frame-ben tárolt logikai lap virtuális címét és annak a processzusnak az azonosítóját tartalmazza, amelyikhez a frame tartozik.
- Csökken a laptáblák tárolásához szükséges memória mérete, de nő a tábla átnézéséhez keresési idő a lapreferencia felmerülése esetén,
- Hash - megoldásokkal a keresési idő csökkenthető.

66. Mit jelent az „igény szerinti" (kényszer) lapozás?

- Csak akkor töltsünk be egy lapot a memóriába, ha szükséges.

, ezáltal:

  - Kevesebb I/O művelet szükséges
  - Kevesebb memória szükséges
  - Kisebb a válaszidő
  - Több felhasználó jut lehetőséghez
- Szükséges egy lap, ha egy aktuális (logikai) címhivatkozás rá utal.
  - Érvénytelen címhivatkozás abortálás
  - A lap nincs a memóriában be kell tölteni a memóriába
- Validációs (valid/invalid) bit (v) szerepe a laptáblában: [v=0] => [page fault]
- Page Fault (laphiba)
  - Az első hivatkozás a lapra biztosan laphibát okoz.
  - Az operációs rendszer eldönti, hogy a hivatkozás
  - maga is hibás-e (abortálás), vagy
  - a lap nincs a memóriába betöltve.
  - Üres keret (frame) keresés.
  - Lap betöltése a keretbe.
  - A megfelelő táblaelemek beállítása (v=1, stb...)
  - A hivatkozott (gépi) utasítás végrehajtásának folytatása (restart).
- Mi történik, ha egy hiányzó lap betöltéséhez nincs szabad (üres) frame a memóriában?
- Algoritmusok az „áldozat" lap kiválasztására és hatásuk az átbocsátó képességre (cél: a page faultok számának minimalizálása)
- A kényszer-lapozás teljesítő képessége
  - laphiba (page fault) arány: $0 \leq p \leq 1$ (p=0: nincs laphiba)
  - A „módosított" (dirty) bit szerepe a lapcserére fordított idő redukálásában
- Effektív elérési idő (EAT: Effective Access Time)
  - EAT= (1-p) * memória elérési idő
  - + p(a laphibával kapcsolatos póttevékenység * [az áldozat-lap kimentési ideje]
  - + a hivatkozott lap betöltési ideje
  - + újraindítási idő)

67. Mi a FIFO laphelyettesítési algoritmus?

- A FIFO-algoritmus úgy próbálja közelíteni az optimálist, hogy *hátranéz* és a *behozatal idejét* figyeli. Az algoritmus azt a lapot cseréli le, amelyik legrégebb óta a tárban van. Megvalósítása egy egyszerű FIFO-listával történhet. Hibája azonban, hogy azokat a lapokat is lecseréli, amelyeket a folyamatok gyakran használnak.
- Előny: csekély hardver támogatás szükséges.

68. Mi a Bélády – anomália?

FIFO algoritmusnál léphet fel egy érdekes jelentéség, miszerint – várakozásainkkal ellentétben – bizonyos esetekben, ha növeljük a folyamathoz tartozó fizikai memóriakeretek számát, akkor nem csökken, hanem éppen növekszik a laphiba gyakoriság.

69. Ismertessen egy optimális laphelyettesítési algoritmust!

Az algoritmus *előrenéz* és a lapok *következő használatának idejét* veszi figyelembe. Ennél az algoritmusnál a legkevesebb a laphibák száma. Sajnos azonban a megvalósítása nehézségekbe ütközik, mivel jövőbeni laphivatkozásokra vonatkozó

információt igényel. Az egzakt végrehajtás gyakorlatilag megoldhatatlan, a kódok valamiféle előreolvasását és szimulált végrehajtását igényelné az adatfüggő elágazások figyelembevételével, ami csaknem olyan bonyolult lenne, mint a program valódi végrehajtása. Így csak közelítő megoldásokkal találkozunk. Az optimális algoritmus szerepe pedig az, hogy összehasonlítási alapként szolgáljon egy-egy eset utólagos értékelésekor, amiből következtetéseket vonhatunk le arra nézve, hogy az alkalmazott algoritmus mennyire közelítette meg az optimumot.

70. Ismertesse az LFU / LRU / MFU laphelyettesítési algoritmust?

- Legrégebben nem használt (LRU: Least Recently Used) algoritmus
  - Az LRU-algoritmus azt a lapot választja áldozatul, amelyre a folyamatok leghosszabb ideje nem hivatkoztak. Ez az algoritmus közelíti legjobban az optimálist, mivel ugyan *hátrafelé* néz, viszont a *használat idejét* veszi figyelembe (azaz a közelmúltból következtet a közeljövőre, ami a lokalitás miatt jó becslést adhat).
  - Ennél a jó teljesítménye miatt gyakran használni kívánt algoritmusnál gondot okoz, hogy „drága". A használat ideje alapján történő sorbarendezés külön hardvertámogatást igényel, és az algoritmus futási ideje is magasabb az egyszerű algoritmusokénál. A megvalósításra több változatot is kidolgoztak.
- Legkevésbé használt (LFU: Least Frequently Used vagy NFU: Not Frequently Used) algoritmus
  - A bonyolult megvalósítás miatt az LRU-algoritmus helyett sokszor annak – hardvertámogatást nem, vagy alig igénylő – közelítését szokták használni (LFU)
  - Ennél az algoritmusnál abból indulhatunk ki, hogy a közelmúltban gyakran használt lapokat a folyamatok a közeljövőben is használni fogják még, és ugyanígy, a közelmúltban ritkán, vagy nem használt lapokra a közeljövőben nem lesz szükség. Ilyenkor az operációs rendszer rendszeres időközönként végignézi a memóriában levő lapokat, és a hozzájuk rendelt számlálóhoz hozzáadja az *R* bit (0 vagy 1) értékét, és egyben törli az *R* biteket. Az algoritmus a legkisebb számláló értékkel rendelkező – vagyis a legritkábban használt – lapot választja ki kivitelre.
  - Hátrányt jelent, hogy az algoritmus „nem felejt", vagyis az egykor gyakran használt lapok még sokáig a memóriában maradnak akkor is, ha már biztosan nem lesz rájuk hivatkozás (például többmenetes fordításnál az egyes menetekhez tarozó lapok). A problémán *öregítéssel* segíthetünk, például úgy, hogy az *R* bitet a legnagyobb helyiértékű bit helyére másoljuk, de előtte a számlálót jobbra léptetve csökkentjük a régebbi hivatkozások súlyát.
  - A módszer másik problémája, hogy a frissen betöltött (és így biztosan kis számláló értékű) lapokat is könnyen kiteheti újra a háttértárra. Ezért általában a frissen behozott lapokat az első használatig **befagyasztjuk (page locking)** a tárba.
- (NRU: Not Recently Used) algoritmus
  - az *R* (hivatkozott) és *M* (módosított) bitek használatán alapszik. A hivatkozottság egy idő elteltével elveszti a jelentőségét, ezért az operációs rendszer rendszeres időközönként törli az *R* bitet. Ugyanakkor az *M* bit értékét őrizni kell, hiszen törlése információ veszteséghez vezetne.
  - A két bit értéke alapján az algoritmus a lapokat négy csoportba osztja, és lapkivitelnél *hátranézve* és a *használat idejét és módját* is figyelembe véve, a lehető legkisebb prioritású csoportból választ véletlenszerűen

71. Ismertesse a „második esély" laphelyettesítési algoritmust!

A második esély algoritmus a FIFO olyan változatát valósítja meg, amely a sor elején levő lapot csak akkor cseréli le, ha nem hivatkoztak rá. Vagyis *hátranéz* és a *használat tényét* figyeli. Ha hivatkoztak a lapra, akkor a *hivatkozott* bitet törli és a lapot visszateszi a FIFO-lista végére, vagyis a lap kap egy újabb esélyt. Ezzel kiküszöböli a FIFO-algoritmus hibáját és a gyakran használt lapokat nem cseréli le. A *hivatkozott* bit tehát ebben a megoldásban azt jelzi, hogy a lapra történt-e hivatkozás azóta, mióta az operációs rendszer legutóbb megvizsgálta, mint lehetséges áldozatot.

72. Mit jelent a szegmentálás, mint memóriakezelési technika?

- A szegmentálás egy memóriakezelési módszer. Célja a memória több címtérre bontása. A memóriát logikailag részekre un. szegmensekre osztjuk, minden résznek megvan a saját, 0-tól kezdődő címtartománya. Egy memóriacím így két részből áll, egy szegmenscímből és egy eltolási- (offset) címből, azaz a memória kétdimenziós. Két szinten valósul meg, hardver és operációs rendszer szinten. A lapozással ellentétben ez nem marad rejtve a felhasználó (programozó) előtt.
- Szegmentálás nélkül egyetlen egydimenziós címterünk lenne, pl.:0-100 címig. Tegyük fel, hogy a programunk két folyamatosan növekvő méretű memóriaterületet használ. Előfordulhat, hogy az első a 0-49. címig tart, a második az 50-től 80-ig. Az első memóriaterületet nem tudjuk tovább növelni, pedig még lenne szabad memória a 81-től kezdődően. A megoldás a szegmentálás. Létrehozunk egy-egy szegmenst a két memóriaterület számára, mindkettő a 0. címtől kezdődik, így mindkét memóriaterületet addig tudjuk növelni amíg a memória el nem fogy.
- A szegmensekhez elérési jogok tartoznak:írható, olvasható, futtatható. Így például a programunk nem írhatja felül saját magát, mert a programkódot egy olyan szegmensben tároljuk, amely csak futtatható.
- A szegmentálás hasznos osztott programkönyvtárak használata esetén is. Ha a függvények külön-külön szegmensben helyezkednek el, akkor a programkönyvtár újabb verziójával a függvények kezdőcíme nem fog megváltozni, még akkor sem, ha a méretük megváltozik.

73. Ismertesse egy felhasználói program feldolgozásának lépéseit!
forrás program ->
compiler/assembler(fordítási idő)->
tárgymodul->
kapcsolat szerkesztő+más tárgymodulok(betöltési idő)->
betölthető modul(betöltési idő)->
betöltő+rendszer könyvtár(betöltési idő)->
végrehajtható program a memóriában+dinamikus rendszerkönyvtár(futási idő)

74. Mi a dinamikus betöltés?

- Egy szubrutin nem töltődik be, amíg meg nem hívódik. Minden szubrutin a lemezen található áthelyezhető bináris formában.
- A hívó rutin először tisztázza, hogy a hívott benn van-e a memóriában. Ha nincs, akkor aktivizálódik az áthelyező betöltő, és a betöltés után a program címhivatkozása (címtáblázat,címkonstansok)
- A nem szükséges rutinok soha nem töltődnek be.
- Nincs szükség speciális operációs rendszer támogatásra (a futtató rendszer - run time system - saját hatáskörén belül megoldja).

75. Mi a dinamikus szerkesztés?

- Statikus szerkesztés: a nyelvi könyvtárak úgy kezelődnek, mint bármely más (felhasználói) object modul. Probléma: a gyakran használt rutinok sok-sok végrehajtható program kódjával együtt letárolódnak. (lemez-pazarlás!)
- Dinamikus szerkesztésnél nemcsak az (áthelyező) betöltés, hanem a (szimbolikus) kapcsolat szerkesztés is kitolódik a futási időre. Egy kisméretű kód (stub=csonk) helyettesíti a szükséges rutint a végrehajtható program kódjában, amely segítségével a szükséges rutin a memóriában (ha az memória rezidens), vagy a lemezes könyvtárban lokalizálható. A lokalizálás után (következő alkalommal) a rutin már direkt módon hajtódik végre (nincs újra töltés!).
- További előnyök: könyvtár módosítások, verziók, bug fixes, patches, service pack, shared library.
- Operációs rendszer támogatás: védett területre betöltött rutinok elérése

76. Mi az overlay?

- felhasználói program logikájába (szubrutin struktúrájába) beépített dinamika
- alapötlet: a teljes programnak (kód és adat) csak az a része legyen benn az operatív memóriában, amelyre ténylegesen szükség van
- pl: többmenetes fordítók, assemblerek (lsd. link)

77. Mi a kapcsolat szerkesztő (linkage editor)?

- más néven linker
- különböző időben fordított tárgymodulokból össze tud rakni egy programot
- bővebben: szerkesztésre akkor van szükség, ha egy program több egymástól függetlenül lefordított modulból áll, amelyek hivatkoznak más modulban definiált logikai címre. A kapcsolatszerkesztő (linker) program feladata a modulok közötti kereszthivatkozások feloldása, és a modulok egymás mögé helyezése. Ennek eredményeképpen a tárgykódban a logikai címek helyére fizikai címek helyettesítődnek, azonban az esetek többségében csak a modulok logikai címtartományainak egyesítése történik meg, és áthelyezhető kódot (ROF - Relokálható Objektum Fájl) kapunk.

78. Mit jelent a POP(3)?

- A Post Office Protocol version 3 (POP3) egy alkalmazás szintű protokoll, melynek segítségével az e-mail kliensek egy meglévő TCP/IP kapcsolaton keresztül letölthetik az elektronikus leveleket a kiszolgálóról. Napjainkban ez a legelterjedtebb protokoll az elektronikus levelek lekéréséhez.
- A jelenleg használatos harmadik változat (version 3) elődjei a POP, illetve POP2 változatok.
- … Wikipédián

79. Mi a futtató rendszer (Run Time System)?

- szoftvergyűjtemény, mely a különböző programozási nyelveken megírt programok futtatását teszi lehetővé. Erőforrásokat biztosít, pl
  - szubrutinok és függvénykönyvtárak általános műveletekhez
  - programozási nyelvek utasításainak implementációja
  - típusellenőrzés, debuggolás, kód optimalizálás

80. Ismertesse és röviden jellemezze a integrált programfejlesztői környezet (IDE) szolgáltatásait!

- Integrated Development Environment azaz integrált fejlesztői környezet.
- Grafikus felületű program, mely általában a fejlesztéshez szükséges részalkalmazásokat (pl. editor, fordító, debugger, esetleg hardverközeli emulációs interfész) integráltan tartalmazza.

81.Az operációs rendszer grafikus felhasználói felülete, GUI.

- A GUI az UI (User Interface - Felhasználói felület) egyik fajtája.
- A CLI-től (Command Line Interface - Parancssoros felület) eltérően a GUI a felhasználóval grafikus elemeken át tart kapcsolatot, ilyen grafikus elemek például az ablak, az ikon, az egérkurzor, stb. Elterjedése nagyban köszönhető a Windows-nak, habár meg kell jegyezni, hogy már a Commodore 64-en is létezett grafikus felület, pl. GEOS néven. Windows-zal valo elterjedése főként annak köszönhető, hogy az nagyjából egybeesett azon időszakkal, amikor a PC-k teljesítménye elérte a grafikus felület megvalósításához szükséges szintet. Manapság szinte minden Operációs rendszernek van grafikus felülete. Szabad szoftveres körökben például az X, GNOME, KDE rövidítéseket hozza be ilyen tárgykörben a pavlovi reflex.
- Szűkebb értelemben lehet egy programnak GUI-ja, azaz programozási szempontból a szoftver azon része ami a grafikus kezelői felület kialakításáért felelős.

82. Mi az SMTP / MIME / POP(3) / IMAP/?

- Simple Mail Transfer Protocol (RFC-822)
  - Interneten a levelezésnél használt protokoll, két MTA (Mail Transfer Agent) között, vagy a MUA(Mail User Agent) és az MTA között levélküldés esetén
- Post Office Protocol 3
  - Levélletöltésre használt protokoll, általában a MUA kezdményezi a POP3 kapcsolatot a szerver felé, ahová a felhasználó levelei megérkeznek, és tárolódnak letöltésükig
- (Interactive Mail Access Protocoll = (interaktív levél hozzáférési protokoll)
  - A POP3-hez hasonló levelezési protokoll, mely lehetővé teszi, hogy leveleinket a szerveren fenntartott postafiókunkban letöltés előtt megnézhessük és távolról kezelhessük.
  - Ellentétben a POP3-mal, mely minden levelet teljes terjedelmében átjátszik gépünkre, az IMAP4 először csak azok fejléceit küldi el; lehetővé teszi hogy a címeiket elolvassuk, a leveleket ott csoportosítsuk és hogy az egyes levelek letöltéséről külön dönthessünk
- MIME (Multipurpose Internet Mail Extensions)
  - Sokcélú Internetes Levélbővítések, más néven Metamailaz Internet eredeti, SMTP levelező protokolljának 1992-ben bevezetett kiterjesztése, amellyel 8 bites karaktereket tartalmazó (ékezetes) szöveges vagy bináris (multimédia) állományok is továbbíthatók elektronikus levélben; a file-ok tartalma 7 bites ASCII kódokra konvertálva kerül átvitelre; a levelezésen kívül más internetes alkalmazások is használják már a MIME kódolást

83. Mi a különbség a TELNET és az FTP szolgáltatásai között?

- Telnet:
  - az egyik legősibb hálózati protokoll.
  - A Telnet protokoll célja egy általánosan elérhető, kétirányú, nyolcbites byte-alapú **kommunikációs rendszer biztosítása**. Egyaránt használható két terminál közötti (linking), illetve processzek közötti kommunikációra. TCP alapon működik.
  - Ma már nincs elterjedve, hiszen nem biztosít semmilyen titkosítást.
- FTP
  - A File Transfer Protocol, vagy rövid nevén FTP TCP/IP hálózatokon – mint amilyen az internet is – történő **állományátvitelre szolgáló szabvány**
  - Kezdi elveszíteni jelentőségét a peer-to-peer protokollokkal szemben.