1. Processzusok

Definíció: Egy végrehajtás alatt lévő program. Tartozik hozzá saját címtartomány azaz memória szelete, amelyen belül olvashat és írhat. A címtartomány tartalmazza a végrehajtadnó programot, annak adatait és a vermét. Minden processzushoz tartozik egy regiszterkészlet, beleértve az utasításszámlálót, veremmutató és egyéb információkat ami szükséges a program futtatásához.

2. Processzusok állapotai

Állapotok:

-futó: éppen használja a CPU-t

-futáskész: kész a futásra, ideiglenesen le lett állítva, hogy egy másik processzus futhasson

-blokkolt: bizonyos külső esemény bekövetkeztéig nem képes futni

-segédállapot: iniciális(megjelenéskor), terminális(befejezéskor), felfüggesztett (pl.: ha kevés a memória)

Az operációs rendszer legalsó szintje az ütemező felette helyezkednek el a processzusok.

3. Processzus leírása

PCB(Process Control Block): A processzus nyilvántartására, tulajdonságainak leírására szolgáló memóriaterület.

Tartalma:

-azonosító

-processzus állapota

-kontextus csere

-utasításszámláló

-veremmutató

-jogosultságok(prioritás)

-birtokolt erőforrások(lefoglalt memória, megnyitott fájlok állapota)

-kért, de nem kapott erőforrások

-ütemezési információk

-elszámolás

-ébresztő jelek

-egyéb információk

A processzusok állapotait a processzustáblázatban tárolja a rendszer, minden processzushoz egy PCB tartozik. A táblázatban talált feljegyzett rekordok a következő célt szolgálják:

-processzus azonosítás

-processzor állapot visszaíállitás

-processzus állapot visszaállítás

4. Processzus létrehozása

A processzusokat létrehozó események:

-rendszer inicializálása

-gyerek processzus kezdeményezése

-felhasználó által kezdeményezett

-kötegelt feladat kezdeményezése

Az OS indulásakor sok processzus keletkezik:

-felhasználókkal tartják a kapcsolatot

-Démonok -> nincsenek felhasználóhoz rendelve (saját feladatuk van)

-futó processzusok is létrehozhatnak processzusokat (kooperatív folyamatok)

Létrehozás lépései:

-memóriaterület foglalása a PCB számára

-PCB kitöltése iniciális adatokkal (állapot, jogosultságok stb)

-memóriafoglalás

-PCB processzusok láncra fűzése

Processus befejezése

Előbb vagy utóbb a processzus befejeződik:

-szabályos kilépés

-kilépés hiba miatt

-kilépés végzetes hiba miatt

-másik processzus kezdeményezi a befejezést

Lépései:

-a gyermek processzus megszüntetése

-a PCB processzusok láncáról való levétele

-a megszűnésekor a processzus birtokában lévő erőforrások felszabadítása

-a memóriatérképének megfelelő memóriaterületek felszabadítása

Processzus hierarchiák*

-amikor egy processzus másik processzust hoz létre szülő-gyermek

-billentyűzetről küldött szignál

5. Szálak

A processzus modell két külön fogalmon alapszik: erőforrás csoportosítás és végrehajtás.

Egy szál rendelkezik:

-utasításszámlálóval

-regiszterekkel

-veremmel

-állapottal

A szálak megvalósíthatóak a felhasználói területen és a kernlelben. Processzusnál blokkolódáskor az egész processzus blokkolódik, viszont szál blokkolódásakor az OS választja ki a következő szálat. Egy globális száltáblázatban tartja nyílván a szálhoz tartozó információkat.

Felbukkanó szálak (Popup szálak)

Ha új üzenet érkezik be akkor azt egy új gyorsan létrehozható szál fogja kezelni, amelynek nincsenek regiszterei.

6. Ütemezési stratégiák kötegelt rendszereknél

Mivel nincs végtelen mennyiségű CPU erőforrás, ezért szükség van ütemezésre, ami alatt a processzorhasználatának irányítást értjük.

Ütemezésre akkor van szükség amikor:

-egy processzus befejeződik (CPU felszabadítás)

-egy processzus blokkolódik

-új processzus jön létre(CPU foglalás)

Ütemezési algoritmusok csoportosítás:

-kötegelt: maximalizálni a feladatok számát és minimalizálni a válaszidőt

-interaktív rendszerek: nem együttműködő processzusok, minimalizálni a válaszidőt

-valós idejű: előrejelezhetőség, határidők betartása, adatvesztés elkerülése

Az össze ütemező rendszerre jellemző:

-páratlanság

-ütemezési elvek betartása

-egységes terhelés elosztás

Sorrendi ütemezés:

A sorrendi ütemezés egy olyan nem megszakítható ütemezés, amely FIFO elv szerint osztja ki a processzusok között a CPU-t. Tehát addig fut egy processzus amíg nem blokkolódik, ha mégis akkor a következő processzus kapja a CPU-t és a sor végére kerül a blokkolódott processzus.

Legrövidebb feladat először:

Ez az ütemezés egy olyan nem megszakítható ütemezés, mely a CPU-t egyformán fontos feladatok közül annak adja amely előreláthatólag hamarabb végez.

Legrövidebb maradék idejű következzen:

Ez egy olyan megszakítható ütemezés, mely a CPU-t a processzus hátralévő idejétől függően osztja ki, mindig az kapja amelyik előreláthatólag a leghamarabb végez.

Háromszintes ütemezés:

Az újonnan érkező feladatok először egy belépési várakozó sorba kerülnek. A bebocsátó ütemező eldönti, hogy mely feladatok léphetnek a rendszerbe. A rövid feladatokat hamar beengedik a hosszabbaknak várniuk kell. Az a komponens (második szint) amely ezt eldönti memória ütemezőnek nevezzük. Az ütemezés harmadik szintje választja ki, hogy melyik processzus fusson következőnek.

7.Stratégiák interaktív rendszerekben

Round robin:

A legegyszerűbb és legtöbbet használt algoritmus a Round robin ütemezés. Az ütemezőnek csak egy listát kell karbantartania a futtatandó processzusokról. A round robin nem figyel a processzusok prioritására.

Prioritásos ütemezés:

Az alapötlet, hogy a processzusokhoz rendelünk prioritást is és a legmagasabb prioritású futásra kész processzust engedjük, hogy fusson. Annak elkerülésére, hogy a magas prioritású processzus a végtelenségig fusson az ütemező minden óraütemben tudja a prioritást csökkenteni.

Többszörös sorok:

A CPU igényes processzusoknak nagyobb időszeletet adunk, mivel ekkor kevesebbet kell a lemezre írni. Viszont ha mindegyik processzusnak nagy időszeletet adunk akkor gyengül a válaszidő.

Legrövidebb processzus következzen:

Ez az algoritmus becsléseket végez múltbeli viselkedés alapján és a processzust a legkisebb becsült futási idő alapján futtatjuk.

Garantált ütemezés:

Ígéreteket tesz a felhasználónak a teljesítménnyel kapcsolatban. Tfh ha n darab felhasználó van bejelentkezve akkor a CPU teljesítményke 1/n-ed részét fogod megkapni.

Sorsjáték-ütemezés:

Minden processzusnak sorsjegyet adunk különböző erőforrásokhoz. Ha ütemezési döntés lép fel akkor kiválasztunk egy sorsjegyet és a sorsjeggyel rendelkező processzus fogja megkapni az erőforrásokat.

Arányos ütemezés:

Figyelembe veszi, hogy ki a processzus tulajdonosa. Minden felhasználó kap valamekkora hányadot a CPU időből attól függetlenül, hogy az adott felhasználó hány processzus szálat futtat.

8. Stratégiák valós idejű rendszerekben

Két csoportba soroljuk őket:

-szigorú valós idejű rendszerek (határidőket kötelező betartani)

-toleráns valós idejű rendszerek (tolerál egy-egy határidőre elmulasztott processzust)

Az események, amelyekre a valós idejű rendszereknek választolniuk kell tovább csoportosíthatók:

-periodikusak (rendszeres intervallumként fordul elő)

-aperiodikusak (megjósolhatatlan az előfordulása)

X. Kontextus-csere

A CPU átvált a P1 processzusról a P2 processzusra. P1 állapotát menteni kell az erre a célra fenttartott memóriaterületre. P2 korábban memóriába mentett állapotát helyre kell állítani a CPU regiszteriben.