1. Processzusok kommunikációja

A processzusoknak gyakran szükségük van kommunikációra. Előnyös ezt nem megszakításokkal kezelni, hanem jól struktúrált módon. A processzusok ne keresztezzék egymás útját, mert akkor versenyhelyzet alakulhat ki.

2. Versenyhelyzet:

Megosztott adatok esetén a végeredmény attól függ, hogy ki mikor fut. Általános probléma a háttérnyomtatás.

Két szerepkör van:

-kliens: aki küldi a nyomtatandó fájlokat

-kiszolgáló(nyomtató): aki periodikusan ellenőrzi a közös részt és nyomtat, ha kell

Ez a memóriaterület kvázi egy verem, amelyiknek mindig van két megosztott változója:

-in (A háttérkatalógus következő szabad helyére mutat)

-out (a következő nyomtatandó állományra mutat)

Egy mód a versenyhelyzetek elkerülésére

-kooperatív

3. Konkurens és kooperatív processzusok

Konkurens:

Ezek a folyamatok gyakran versenyeznek az erőforrásokért ezért nevezzük őket konkurensnek.

A konkurens processzusokra jellemző:

-versenyhelyzet kialakulsa

-Szinkronizáció: tehát a processzusoknak alkalmazkodni kell az erőforrásokhoz való hozzáféréshez

-nem determinisztikus: a folyamatok interakciója nem látható előre

Kooperatív:

Ezek a folyamatok együttműködnek a végrehajtásuk során, ez azt jelenti, hogy egy processzus tudatosan megáll annak érdekében, hogy a többi processzust ne akadályozza.

A kooperatív processzusokra jellemző:

-önkéntes váltás

-megjósolhatóság

-blokkolás kockázata

4. Kritikus szekció

A programnak azt a részét, amelyben a megosztott memóriát használják a processzusok, kritikus területnek vagy kritikus szekciónak nevezzük. Ha úgy tudjuk rendezni a dolgokat, hogy soha ne legyen két processzus egyszerre kritikus szekcióban, akkor elkerülhetnénk a versenyhelyzetet.

A jó megoldáshoz négy feltételt kell betartani:

-Ne legyen két processzus a saját kritikus szekciójában

-Semmilyen előfeltétel ne legyen a sebességekről vagy a CPU-k számáról

-Egyetlen a kritikus szekcióján kívül futó processzus sem blokkolhat más processzusokat

-Egyetlen processzusnak se kelljen örökké várni, hogy belépjen a kritikus szekciójában.

5. Kölcsönös kizárás tevékeny várakozással

Módszerek a kölcsönös kizárás eléréséhez:

-megszakítások tiltása

-zárolásváltozók

-szigorú válogatás

-Peterson megoldása

-TSL utasítás

Tevékenységek várakozásának nevezzük azt az állapotot, amikor egy változót folyamatosan tesztelünk bizonyos érték bekövetkeztéig.

Megszakítások tiltása

Az operációs rendszereknél megengedett technika a megszakítások tiltása. Minden processzus letiltja az összes megszakítást a kritikus szekcióba lépés után és újra engedélyezi, mielőtt elhagyja azt.

Változók zárolása

Egyetlen osztott változót használunk, tehát egy közös változó ellenőriz és módosít minden processzust.

Szigorú váltogatás

A zárolásváltozókhoz hasonló megvalósítás a Szigorú válogatás, mely esetén egy while ciklus folyamatosan teszteli a változó értékét, ezáltal pazarolja a CPU-t. Továbbás sérti az „Egyetlen, a kritikus szekcióján kívül futó processzus sem blokkolhat más processzusokat”.

Peterson megoldása:

Két változót használ, ezek a flag és turn. Egy processzus saját flag értékének igazra állításával jelzi, hogy ő belépne a kritikus szekcióba, a turn pedig azt jelöli melyik a következő processzus.

Az első processzus addig van várakozó állapotban, amíg a második processzus kritikus szekcióban van és a turn változó nem magára mutat. Ha a második processzus kilépett a kritikus szekciójából, akkor a flag értékét falsera állítja, ezzel megengedve az első processzusnak, hogy belépjen a kritikus szekciójába.

TSL utasítás(Test Set and Lock)

Ez egy alacsony szintű szinkronizációs művelet, amely egyetlen, összetett lépésben képes ellenőrizni (test) egy zárolási változó értékét, majd, ha az szabad (általában 0), a zárat beállítja (set and lock) egy foglalt állapotra (általában 1). Ezzel biztosítja, hogy egyszerre csak egy szál vagy folyamat módosíthassa az adott erőforrást vagy kritikus szakaszt.

6. Altatás és ébresztés

termelő-fogyasztó probléma

Két processzus osztozik egy közös tárolón. Az egyikük gyártó adatokat helyez a tárolóba a másik pedig fogyasztó adatokat vesz ki a tárolóból. A probléma akkor jelentkezik amikor nincsen hely már a gyártónak adatot behelyezni a tárolóba, mert az már tele van. A megoldás erre, hogyha a gyártó elalszik és felébresztik amikor a fogyasztó egy vagy több adatot kivett a tárolóból. Hasonló esetben ha a fogyasztó szeretne elemet kivenni a tárolóból és a tároló üres, a fogyasztó elalszik és a gyártó fogja felébreszteni, ha betett a tárolóba egy vagy több elemet, amit a fogyasztó ki tud venni.

szemaforok

A szemafor egy változótípus amelyben számolhatjuk a felébresztéseket későbbi felhasználás céljából.

Ha a szemafor értéke:

-0 akkor nincs elmentett ébresztés

-pozitív érték: egy vagy több ébresztés van függőben

Két alapvető művelet van a down(sleep) és az up(wakeup).

-down: megvizsgálja a szemafor értékét és ha nagyobb mint 0 akkor csökkenti egyel. Ha a számláló 0, akkor a folyamatot "blokkolja" vagy várakozó állapotba helyezi, amíg az erőforrás szabad nem lesz.

-up: a szemafor értékét növeli egyel. Ha vannak várakozó folyamatok akkor egyet felébreszt, hogy megpróbálhassák a down műveletet. Ezáltal a szemafor még mindig 0 lesz viszont egyel kevesebb processzus fog rajta aludni.

mutex-ek:

A szemafor egyszerűsített változata, hiszen a mutexek is megakadályozzák, hogy több szál vagy folyamat egyszerre férjen hozzá ugyanahhoz az erőforráshoz

Kétféle állapotban lehetnek:

-zárolt

-nem zárolt

Egyetlen bit elengedő a reprezentáláshoz.

Amikor egy processzus hozzá szeretne férni a kritikus szekciójához, akkor meghívja a mutex_lock eljárást, amennyiben a mutex nem zárolt állapotban van. Ha a mutex zárolt állapotban van akkor a hívás blokkolódik addig amíg a kritikus szekcióban lévő processzus nem végez és meg nem hívja a mutex_unlock metódust.

monitorok

Egy speciális modulba összegyűjtött eljárások, változók és adatszerkezetek együttese. Minden időállapotban csak egy processzus használhatja a monitor elemeit.

-belépés: egy szál kezdeményezi a belépést, ha foglalt akkor várakozik amíg elérhetővé nem válik.

-kilépés: amikor egy szál befejezte a műveletet a monitorban, kilép, ezzel felszabadítva a monitort más processzusok számára.

Üzenetek (adás-vétel)

Egy olyan módszer a számítógépes tudományban, amely lehetővé teszi a folyamatok közötti kommunikációt és szinkronizációt több szálas rendszerekben. A folyamatok üzeneteket küldenek és fogadnak egymástól, hogy megosszák az információt az állapotukról és tevékenységükről.

Írók és olvasók problémája:

Egy klasszikus szinkronizációs probléma, ahol a folyamatok két csoportra oszlanak:

-olvasók: akik csak olvasni szeretnék az adatokat

-írók: akik módosítani szeretnék az adatokat

Tehát a probléma, hogy az olvasók csak akkor tudjanak olvasni, ha az adat nincs írásban és az írók pedig csak akkor tudjanak írni, ha az adat nincs olvasásban.

Példa:

-Egy író küld egy üzenetet, hogy szeretne írni

-A monitor vagy sorompó gondoskodik arról, hogy amíg az írás tart addig az olvasók ne férjenek hozzá az adatokhoz.

-Amikor az író befejezte az írást, küld egy üzenetet, hogy az adatbázis most már szabad.

-Az olvasók ezeket az üzeneteket felhasználva tudják, hogy mikor férhetnek hozzá biztonságosan az adatokhoz.

Sorompók:

Fázisokra osztja az alkalmazást, a lényege, hogy egyetlen processzus sem mehet tovább addig amíg az összes processzus nem végzett. Mindegyik fázis végére helyezünk egy sorompót. Amikor egy processzus a sorompóhoz ér akkor addig blokkolódik amíg a többi processzus nem végez, azaz el nem éri a sorompót.