Table of Contents

[Deadlockok jellemzése 1](#_Toc166869015)

[Erőforrás-allokációs grafikon 2](#_Toc166869016)

[Erőforrás-allokációs grafikon holtponttal 2](#_Toc166869017)

[Deadlock megelőzése 3](#_Toc166869018)

[Deadlock elkerülő algoritmusok: 4](#_Toc166869019)

[Deadlock észlelése 4](#_Toc166869020)

[Helyreállás a deadlockból 5](#_Toc166869021)

**7. Deadlockok**

# Deadlockok jellemzése

A deadlock helyzetet az jellemzi, hogy több folyamat végtelenül várakozik olyan eseményekre, amelyeket csak az egyik várakozó folyamat tud előidézni. A deadlock négy alapvető feltételének egyidejűleg kell fennállnia:

Közös kizárás: Legalább egy erőforrás nem osztható meg; vagyis egyszerre csak egy folyamat használhatja.

Tartás és várakozás: A folyamatok legalább egy erőforrást tartanak, miközben további, már más folyamatok által birtokolt erőforrásokra várnak.

Nem megelőzhetőség: Az erőforrásokat nem lehet megszakítani; azaz egy erőforrást csak az azt birtokló folyamat engedhet el, miután befejezte feladatát.

Körkörös várakozás: Léteznie kell egy olyan folyamatokból álló körnek, ahol minden folyamat a következő folyamat által birtokolt erőforrásra vár.

Ezek a feltételek együttesen hozzák létre a deadlock helyzetet, és ha bármelyikük nem teljesül, a deadlock elkerülhető. A számítástechnikai rendszerekben a deadlock elkerülésére több stratégia is létezik, beleértve az erőforrás-kiosztási grafikonokat, amelyek segítenek azonosítani a potenciális deadlockokat, és elősegítik azok megelőzését vagy feloldását​​.

# Erőforrás-allokációs grafikon

Az erőforrás-allokációs grafikon egy irányított gráf, amely pontosan írja le a rendszerben lévő erőforrás-allokációs viszonyokat. A grafikon csúcsai két csoportra oszlanak: a folyamatokra (P1, P2, ... Pn) és az erőforrástípusokra (R1, R2, ... Rm). A folyamatokat körök, míg az erőforrástípusokat téglalapok jelölik. Minden folyamat és erőforrástípus között irányított élek húzódnak, amelyek kétféle kapcsolatot jelölhetnek:

Kérés él: Egy irányított él a folyamatból egy erőforrástípus felé (Pi → Rj) azt jelzi, hogy a folyamat az adott erőforrástípust igényli.

Hozzárendelés él: Egy irányított él egy erőforrástípusból egy folyamat felé (Rj → Pi) azt jelenti, hogy az erőforrástípus egy példánya már hozzá lett rendelve a folyamathoz.

A grafikon dinamikusan változik az erőforrás-igények és -kiadások folyamán. Amikor egy folyamat erőforrást igényel, egy kérés élt helyezünk be. Amikor ez az igény kielégül, a kérés él átalakul hozzárendelés élévé. Amikor a folyamat már nem szükséges az erőforrást, az él eltávolításra kerül. A grafikon segítségével az erőforrás-allokációs állapotok és a potenciális deadlockok könnyen vizsgálhatók. Ha a grafikon tartalmaz kört, az jelenthet deadlockot, ha minden érintett erőforrástípusnak csak egy példánya van​​.

# Erőforrás-allokációs grafikon holtponttal

Az erőforrás-allokációs grafikonok alkalmazása holtpontos helyzetekben különösen jelentős, mivel ezek a grafikonok segítenek vizualizálni a rendszer erőforrás-függőségeit. A grafikonban, ha létezik kör, az jelenthet holtpontot. Például, ha minden erőforrástípusnak csupán egy példánya van, és egy körben minden folyamat kizárólag egy következő folyamat által birtokolt erőforrást vár, akkor ez a kör egyértelműen holtpontot jelez.

Amikor egy új kérésél él bekerül a grafikonba, az megbontja a korábbi állapotot és potenciálisan kör kialakulásához vezethet. Ha a grafikonban kör jön létre, ez az új állapot biztonságtalan állapotot jelez, amely holtpont kialakulásához vezethet. A körök felismerése és kezelése kritikus a holtpont elkerülése és a rendszer stabilitásának fenntartása érdekében.

A könyvben található példában a folyamatok és az erőforrások közötti kérés és hozzárendelés élének átalakítása egyértelműen le van írva. A kérés élek átalakulnak hozzárendelés élekké, amikor az erőforrásokat kiosztják, és visszaalakulnak kérés élekké, amikor a folyamatok felszabadítják az erőforrásokat​​.

# Deadlock megelőzése

A holtpont megelőzése stratégiák sorozatán alapszik, amelyek azért szükségesek, hogy legalább egyik alapvető holtpontot előidéző feltétel ne teljesülhessen. Ezek a megelőzési stratégiák a következők:

**Közös kizárás megtagadása**: Bizonyos esetekben a rendszer próbálja minimalizálni a kizárólagosan használt erőforrásokat. Amennyiben egy erőforrás megosztható, azzal a közös kizárás feltétele megszüntethető.

**Tartás és várakozás megtagadása**: Ez a stratégia megköveteli, hogy a folyamatok minden szükséges erőforrást egy időben igényeljenek, és azokat csak akkor tarthassák meg, ha minden igényelt erőforrás megkapható. Egy alternatív megközelítésben egy folyamat csak akkor kérhet erőforrást, ha jelenleg egyet sem birtokol.

**Nem megelőzhetőség** megtagadása: Egyes esetekben a rendszerek lehetővé teszik, hogy bizonyos erőforrásokat "elővegyenek" a folyamatoktól. Ezzel a folyamatok elkerülhetik a holtpontot, ám ez gondot jelenthet, ha az erőforrás kritikus állapotban van.

**Körkörös várakozás megtagadása:** Erőforrás-típusok egy meghatározott sorrendjének kialakításával a folyamatok csak növekvő sorrendben igényelhetnek erőforrásokat. Ezzel a stratégiával a körkörös várakozás kiküszöbölhető, mivel a folyamatok nem várhatnak olyan erőforrásokra, amelyek már egy korábban megszerzett erőforrástípushoz tartoznak.

Ezek a stratégiák együttesen működnek a holtpontok elkerülésére, de gyakran korlátozzák a rendszer erőforrás-kihasználtságát vagy általános teljesítményét, mivel ezek a megközelítések gyakran korlátozásokat vezetnek be a folyamatok működésére​​.

# Deadlock elkerülő algoritmusok:

Az operációs rendszerek két fő holtpontelkerülő algoritmusa a Biztonságos Állapot és a Bankár Algoritmus.

Biztonságos Állapot (Safe State)

Egy rendszer biztonságos állapotban van, ha lehetséges az erőforrások olyan sorrendben történő kiosztása a folyamatoknak, amely megelőzi a holtpontot. Egy biztonságos sorozat megléte garantálja, hogy minden folyamat végre tudja hajtani feladatait anélkül, hogy holtpontba kerülne. Ha egy ilyen sorozat létezik, akkor a rendszer biztonságos állapotban van. Ha a folyamatok erőforrásigénye azonnal nem teljesíthető, akkor várakoztatásra kerülnek, amíg a korábbi folyamatok befejeződnek és felszabadítják az erőforrásaikat .

Bankár Algoritmus (Banker’s Algorithm)

Ez az algoritmus egy többpéldányos erőforrás-allokációs rendszerekre alkalmazható holtpontelkerülő megoldás. A folyamatoknak előre deklarálniuk kell az egyes erőforrástípusokból legfeljebb igénybe vehető példányok maximális számát. Az algoritmus ezt az információt felhasználva dönti el, hogy egy adott erőforrás-igény kielégíthető-e anélkül, hogy a rendszert veszélyeztetné holtpont kialakulása szempontjából. Ha az erőforrás kiosztása nem vezet veszélyes állapotba, akkor az igény kielégítésre kerül; ellenkező esetben a folyamatnak várnia kell. A Bankár Algoritmus fő célja, hogy mindig biztonságos állapotban tartsa a rendszert, ezzel elkerülve a holtpontok kialakulását .

# Deadlock észlelése

A holtpontok észlelése fontos lépés az operációs rendszerekben, amely lehetővé teszi, hogy a rendszer felismerje és kezelje a folyamatok közötti erőforrás-konfliktusokat. A holtpontok észlelésére több módszer is létezik, amelyek közül a leggyakrabban használt a várakozási gráf algoritmus. Ez az algoritmus a következő lépéseket tartalmazza:

Várakozási gráf létrehozása: A rendszer fenntart egy várakozási gráfot, amely az egyes folyamatok és az általuk igényelt erőforrások közötti kapcsolatokat ábrázolja. A gráf csomópontjai a folyamatok, az élek pedig az erőforrások irányába mutatnak, amelyeket a folyamatok igényelnek.

Cikluskeresés a gráfban: A rendszer rendszeresen ellenőrzi a gráfot ciklusok jelenlétére. Egy ciklus jelenléte jelzi, hogy holtpont áll fenn. Egy holtpont akkor fordul elő, ha egy folyamatokból álló kör minden eleme vár egy másik elem által birtokolt erőforrásra.

Észlelés és kezelés: Amennyiben a rendszer holtpontot észlel, többféle kezelési stratégia létezik. Ezek közé tartozik a folyamatok leállítása, erőforrások előzetes megszakítása, vagy az operátor értesítése a problémáról.

Ez az észlelési módszer különösen hasznos többpéldányos erőforrások esetén is alkalmazható, ahol a folyamatok több különböző típusú erőforrást igényelhetnek egyszerre. Az algoritmus hatékonysága és a rendszer általános teljesítménye közötti egyensúly fenntartása érdekében fontos, hogy az észlelési algoritmust megfelelő gyakorisággal hívják meg .

# Helyreállás a deadlockból

A holtpontból való helyreállás több alternatívát kínál az operációs rendszerek számára, amikor a detektálási algoritmus holtpontot észlel. A lehetséges megoldások között szerepel a folyamatok manuális leállítása, illetve az automatikus helyreállás. A folyamatok manuális leállításakor az operátor informálódik a holtpont bekövetkezéséről, és kézi beavatkozással oldja fel a helyzetet. Az automatikus helyreállás során két fő lehetőség áll rendelkezésre: egy vagy több folyamat megszakítása a ciklikus várakozás megszüntetése érdekében, vagy erőforrások elvonása a holtpontban lévő folyamatoktól.

Folyamatok megszakítása

A holtpont feloldásának egyik módja, hogy megszakítjuk az egyik vagy több folyamatot. Ehhez két módszer létezik:

Az összes holtpontban lévő folyamat megszakítása, ami garantáltan feloldja a holtpontot, de nagy költséggel jár, mivel a részlegesen elvégzett számításokat el kell vetni.

Egy folyamat megszakítása egyszerre, majd a holtpont-észlelési algoritmus újbóli futtatása, hogy megállapítsuk, maradt-e még holtpontban lévő folyamat.

Erőforrások elvonása

A holtpont feloldásának másik módja az erőforrások elvonása. Ebben az esetben az alábbi lépéseket kell megtenni:

Áldozat kiválasztása: Meg kell határozni, melyik folyamatoktól és melyik erőforrásoktól vonjuk el az erőforrásokat. A költségek minimalizálása érdekében rangsoroljuk az áldozatokat.

Visszagördítés: Ha egy folyamatból erőforrást vontunk el, a folyamatot biztonságos állapotba kell visszagördíteni, ahonnan újraindítható.

Éhezés megelőzése: Biztosítani kell, hogy az erőforrásokat ne mindig ugyanattól a folyamattól vonjuk el, így elkerülve, hogy a folyamat soha ne tudja befejezni a feladatát.

A helyreállási mechanizmusok megvalósítása komplex feladat, mely magában foglalja a rendszer állapotának nyomon követését, a holtpont-észlelési algoritmusok időszakos futtatását, valamint a helyreállítási stratégiák hatékony végrehajtását​​.