Operációs rendszerek

A halálos ölelés

1.1

Pere László (pipas@linux.pte.hu)

PÉCSI TUDOMÁNYEGYETEM TERMÉSZETTUDOMÁNYI KAR

INFORMATIKA ÉS ÁLTALÁNOS TECHNIKA TANSZÉK

Erőforrások használata

Az erőforrások használata három szakaszra bontható:

1. Az erőforrás kérése.
2. Az erőforrás tényleges használata.
3. Az erőforrás felszabadítása.

A folyamatközi kapcsolattartás során a közös erőforrásokat e három szakasznak megfelelően használjuk a kölcsönös kizárás elvének megfelelően.

Erőforrások csoportosítása

A deadlock szempontjából az erőforrásokat két csoportba sorolhatjuk:

- Preemtable resource: az erőforrás elvehető, megvonható a folyamattól.
  A memóriát például ki lehet írni a csereterületre így elvehetõ a folyamattól.
- Nonpreemtable resource: az erőforrás károkozás nélkül nem megvonható a folyamattól.
  Ilyen például a CD író, amelyet ha elveszünk a lemezt eldobhatjuk.

A halálos ölelés

A halálos ölelelés vagy holtpont (*deadlock*) a közös erőforrás(ok) használata közben beálló statikus állapot.

A folyamatok egy halmaza halálos ölelésben van, ha a halmaz minden eleme olyan eseményre vár, amelyet csak a halmaz valamely eleme válthat ki.

Deadlock kialakulásának feltételei

1. Kölcsönös kizárás feltétele: az erőforrások szabadok vagy egy folyamathoz rendeltek.
2. Hold and wait feltétel: a folyamat, amely erőforrást kapott, újabb erőforrásokat kérhet.
3. No preemption feltétel: az erőforrás nem vonható meg a folyamattól.
4. Körkörösség: a várakozás – tartás hatásvázlatban körkörös résznek kell lennie.

A feltételek bármelyikének megszüntetésével a deadlock kialakulása megelőzhető.

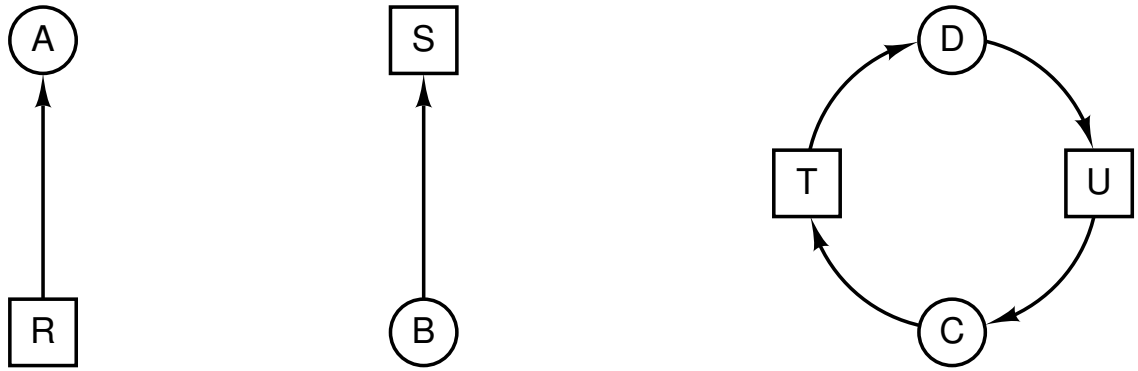
Modellezés

A deadlock modellezhető gráfokkal (Holt, 1972), ahol:

- a folyamatok körök,
- az erőforrások négyszögek,
- erőforrástól folyamatig rajzolt nyíl jelzi, ha az erőforrást a folyamat kérte, megkapta és jelenleg tartja,
- folyamattól erőforrásig rajzolt nyíl jelzi, hogy a folyamat az erőforrást igényelte és most várakozik rá (blokkolt).

Modellezés eredménye

Ha feltesszük, hogy minden erőforrásból egyetlen példány létezik, a rendszer halálos ölelésben van, ha a gráfban hurok található.



1. ábra. Halálos ölelés gráfja

Alapvető stratégiák

A halálos öleléssel kapcsolatban 4 alapvető stratégia alakítható ki:

- Hagyjuk figyelmen kívül, reméljük nem következik be.
- Detektáljuk és lépjünk közbe. Ha deadlock lép fel, vegyük észre és tegyünk valamit, hogy feloldjuk.
- Kerüljük el az erőforrások körültekintő kiosztásával (sorrendre ügyeljünk).
- Előzzük meg a 4 feltétel valamelyikének kizárásával. Itt eleve lehetetlenné tesszük a deadlock kialakulását.

1. Ostrich algoritmus

Struccpolitika, műszaki szempontból nem biztos, hogy érdemes foglalkozni a kérdéssel.

Ha például minden hónapban egyszer fordul elő deadlock, a rendszer viszont naponta lefagy, nem kell sokat foglalkoznunk a kérdéssel.

A legtöbb általánosan használt operációs rendszer nem kezeli az alkalmazásprogramozó által kialakított halálos ölelést, de az operációs rendszert ezek a holtpontok nem akadályozzák.

2. Detektálás és közbelépés

Ha minden erőforrásból csak egy áll rendelkezésre, a deadlock detektálható gráfkereső algoritmusokkal.

Ha egy vagy több erőforrásból több példány is rendelkezésre áll, a deadlock mátrixalgebrát használó módszerekkel felismerhető.

Amikor a rendszer felismeri, hogy deadlock áll fenn, a következő oldalakon olvasható módon járhat el.

2.1 Erőforrás megvonása

Bizonyos esetekben az erőforrás megvonható, visszavonható.

Ilyen esetekben a folyamatot blokkoljuk, az erőforrást visszavonjuk, megvárjuk amíg a másik folyamat végez, az erőforrást visszaadjuk majd a folyamat blokkolását feloldjuk.

2.2 Visszagörgetés

A folyamat teljes állapotát időről-időre elmentjük és szükség esetén visszatöltjük.

Ez a módszer a számítógépes játékokban megszokott mentés/visszatöltés módszerhez nagyon hasonlít.

A visszagörgetés módszerét elterjedten használjuk adatbáziskezelő rendszerekben, amelyek a deadlock detektálására és a visszagörgetésre sokszor egyszerű módszereket használnak.

2.3 Megszakítás

A deadlock halmazban található folyamatok némelyikének megszakításával a deadlock általában feloldható.

Ügyelnünk kell, hogy olyan folyamatot szakítsunk meg, amely újraindítható és az eredmény pótolható, ezért a megszakítás sokszor operátori beavatkozást igényel.

3. A deadlock elkerülése

Mivel az operációs rendszer osztja ki az erőforrásokat, képes lehet arra, hogy az erőforrások odaítélését elhalassza, ha úgy találja, hogy az deadlock veszélyes helyzet kialakulásához vezet.

Ez a módszert összetett, bonyolult algoritmusokat igényel, amelyek a halálos ölelés veszélyére időben felhívják a figyelmet (sakk és más táblajátékok).

4. Deadlock megelőzése

A következő oldalakon olyan módszerekről, eljárásokról olvashatunk, amelyek egyszer s mindenkorra lehetetlenné teszik a deadlock kialakulását.

4.1 Kölcsönös kizárás

Bizonyos esetekben a kölcsönös kizárás feltétele eliminálható, ha az erőforrást egyidőben több folyamat által használhatóvá tesszük, azaz megszüntetjük a kölcsönös kizárás szükségességét.

Példa: a nyomtatót csak egy folyamat használhatja egyszerre, de ha spoolert használunk több folyamat nyomtathat egyszerre.

4.2 Hold and wait

A hold and wait állapot lehetetlenné tehető, ha kötelezővé tesszük a folyamat számára, hogy előbb az összes erőforrást lefoglalja, csak utána végezze el a munkát.

Ez nem szerencsés, mert feleslegesen fogja az erőforrásokat, ezért néhány rendszer lehetővé teszi, hogy az erőforrásokról lemondva újra kérje a folyamat a következő szakaszban szükséges erőforrásokat.

4.3 Megszakíthatatlanság

A deadlock feltételek közül az erőforrás visszavonhatatlansága a legkevésbé megelőzhető, hiszen az operációs rendszer egyik feladata éppen az erőforrások kisztása.

Ha az operációs rendszer nem garantálja, hogy a lefoglalt erőforrást a folyamat használhatja, akkor mit végzi el a feladatát!

Körkörösség

A körkörösség feltétele több módszer segítségével is kizárható a rendszerből:

1. Minden folyamat csak egy erőforrást tarthat magánál. Ez sokszor nem tartható be.
2. A folyamatok csak egy bizonyos sorrendben foglalhatják le az erőforrásokat (nem minden erőforrás számozható meg).
3. A folyamat csak a lefoglalt erőforrások közül a legnagyobb sorszámú erőforrásnál magasabb sorszámút kérhet.