IMSc labor Operációs rendszerek tárgyból

A kölcsönös kizárás a konkurens programozás és az elosztott rendszerek fejlesztésének klasszikus problémája. A lényege a következő [1]: Processzek egy csoportjának egy megosztott erőforrást kell elérnie, amelyet egyidejűleg legfeljebb egy processz használhat. A megoldásnak a következő tulajdonságokkal kell rendelkeznie.

- *Kölcsönös kizárás*: Minden processznek végre kell hajtania egy kritikus szakasznak nevezett kódszegmenst úgy, hogy bármely adott időpillanatban legfeljebb egy processz hajthatja azt végre (azaz egy processz van a kritikus szakaszban).
- *Holtpontmentesség*: Ha egy vagy több processz megkísérel belépni a kritikus szakaszba, valamikor végül egy sikerrel jár, feltéve, hogy egy processz sem tartózkodik a kritikus szakaszban örökké.
- *Kiéheztetésmentesség*: A kritikus szakaszba belépni kívánó processz valamikor végül sikerrel jár, feltéve, hogy egy processz sem tartózkodik a kritikus szakaszban örökké.

Ennek a problémának az eredeti megoldásai olyan speciális szinkronizációs eszközökre támaszkodnak, mint a szemafor vagy a monitor. A feladat keretein belül olyan elosztott algoritmusokat vizsgálunk, amelyek csak közönséges megosztott változókat használnak, tehát a kölcsönös kizárást a megosztott változókon végrehajtott írás illetve olvasás műveletekkel kell megvalósítani.

Feltesszük, hogy egy processz programja az alábbi szakaszokra bomlik:

- Belépő (próbálkozó): a kritikus szakaszba való belépés előkészítése.
- Kritikus: az egyidejű végrehajtástól megvédendő szakasz.
- Kilépő: a kritikus szakasz elhagyásakor végrehajtott kód.
- Fennmaradó: a kód többi része.

Egy-egy processz rendre a Belépő – Kritikus szakasz – Kilépő – Fennmaradó kódrészleteket hajtja végre ciklikusan. Feltesszük, hogy nincs olyan processz, ami állandóan a kritikus szakaszban tartózkodik.

A következőkben az a célunk, hogy a megadott algoritmusokat

- modellezzük (ha lehetséges) választott Petri-háló alapú eszközök segítségével,
- majd próbáljuk meg a fenti elvárt tulajdonságok (mint követelmények) teljesülését igazolni,
- ha egy követelmény nem teljesül, akkor pedig elemezzük, hogy miért.

1. Hyman algoritmusa

Harris Hyman 1966-ban javasolta a következő algoritmust [2]. Legyen két processz P1 és P2, a használt megosztott változók pedig a következők:

- blocked0: Az első processz (P1) be akar lépni;
- blocked1: Második processz (P2) be akar lépni;
- turn: Ki következik belépni (0 esetén P1, 1 esetén P2).

Az algoritmusok a két processz esetén a következők:

P1 processz	**P2 processz**

while (true) {
blocked0 = true;
while (turn!=0) {
while (blocked1==true) {
skip;
}
turn=0;
}
// Critical section here
blocked0 = false;
// Other things
}
	while (true) {
	blocked1 = true;
	while (turn!=1) {
	while (blocked0==true) {
	skip;
	}
	turn=1;
	}
	// Critical section here
	blocked1 = false;
	// Other things
	}

Vizsgáljuk meg, hogy az algoritmus teljesíti-e az alapvető követelményeket (holtpontmenteség, kölcsönös kizárás, kiéheztetésmentesség). Ha valamelyik követelmény nem teljesül, mutassuk meg, hogy miért (milyen példa adható a követelmény megsértésére).

2. Peterson algoritmusa

Peterson 1981-ben adott egy módosított algoritmust a következők szerint [3]:

P1 processz	**P2 processz**

while (true) {
blocked0 = true;
turn=1;
while (blocked1==true &&
turn!=0) {
skip;
}
 	
// Critical section here
blocked0 = false;
// Other things
}
	while (true) {
	blocked1 = true;
	turn=0;
	while (blocked0==true &&
	turn!=1) {
	skip;
	}
	
	// Critical section here
	blocked1 = false;
	// Other things
	}

A modellezés során ügyeljünk arra, hogy a belső while ciklus feltételvizsgálata nem atomi jellegű!

Vizsgáljuk meg az algoritmus tulajdonságait a következők szerint:

- Ellenőrizzük, hogy ez a módosított algoritmus teljesíti-e az alapvető követelményeket (holtpontmenteség, kölcsönös kizárás, kiéheztetésmentesség). Ha valamelyik követelmény nem teljesül, mutassuk meg, hogy miért (milyen példa adható a követelmény megsértésére).
- Ellenőrizzük, hogy az algoritmus teljesíti-e a kölcsönös kizárás követelményét, ha a belső while ciklus feltételvizsgálat részében felcseréljük az && operátorral összekötött két vizsgálat sorrendjét!
- Az algoritmus wikipedia lapján[1] a következő olvasható (a változók neveit értelemszerűen átírva): *If P1 is in its critical section, then blocked0 is true and either blocked1 is false or turn is 0.* Próbáljuk meg bebizonyítani ezt az állítást!
- Peterson algoritmusának jellemzője, hogy egy processz esetén a kritikus szakaszba való belépéshez szükséges várakozási idő korlátos, amennyiben feltesszük, hogy minden processz korlátos időt tölt el a kritikus szakaszban. Be tudjuk-e ezt bizonyítani a modell kiterjesztésével?

Hivatkozások

1. Iványi Antal (szerkesztő): Informatikai algoritmusok II. ELTE Eötvös Kiadó, 2005.
2. H. Hyman: Comments on a problem in concurrent programming control. Communications of the ACM, 9(1):45, 1966.
3. G. L. Peterson: Myths about the mutual exclusion problem. Information Processing Letters, 12(3):115-116, 1981.

[1] http://en.wikipedia.org/wiki/Peterson's_algorithm