

2. Úloha (2016/2017)

Popis úlohy

Implementujte v jazyce C modifikovaný synchronizační problém *Child Care* (můžete se inspirovat knihou [The Little Book of Semaphores](#)). Existuje centrum, které se stará o děti, a dva typy procesů: dospělý člověk (*adult*) a dítě (*child*). V centru se dospělí lidé starají o děti, přičemž jeden dospělý se může starat nejvýše o tři děti. Např. pokud je v centru 5 dětí, musí být přítomni alespoň 2 dospělí, pokud jsou pouze 3 děti, stačí jeden dospělý. Do centra přicházejí a z centra odcházejí dospělí lidé a děti, ovšem tak, aby nebyla porušena výše uvedená výjimka. Pokud by byla odchodem dospělého porušena podmínka, musí s odchodem vyčkat, než odejde dostatečný počet dětí. Podobně, pokud chce vstoupit dítě a v centru není dostatek dospělých, musí vyčkat, až přijde další dospělý.

Detailní specifikace úlohy

Spuštění

```
$ ./proj2 A C AGT CGT AWT CWT
```

kde

- *A* je počet procesů *adult*; $A > 0$.
- *C* je počet procesů *child*; $C > 0$.
- *AGT* je maximální hodnota doby (v milisekundách), po které je generován nový proces *adult*; $AGT \geq 0 \ \&\& \ AGT < 5001$.
- *CGT* je maximální hodnota doby (v milisekundách), po které je generován nový proces *child*; $CGT \geq 0 \ \&\& \ CGT < 5001$.
- *AWT* je maximální hodnota doby (v milisekundách), po kterou proces *adult* simuluje činnost v centru; $AWT \geq 0 \ \&\& \ AWT < 5001$.
- *CWT* je maximální hodnota doby (v milisekundách), po kterou proces *child* simuluje činnost v centru; $CWT \geq 0 \ \&\& \ CWT < 5001$.
- Všechny parametry jsou celá čísla.

Implementační detaily

- Pracujte s procesy, ne s vlákny.
- Každému dospělému odpovídá jeden proces *adult*.
- Každému dítěti odpovídá jeden proces *child*.
- Hlavní proces vytváří ihned po spuštění dva pomocné procesy pro generování procesů *adult* a *child*. Poté čeká na ukončení všech procesů, které aplikace vytváří. Jakmile jsou tyto procesy ukončeny, ukončí se i hlavní proces s kódem (*exit code*) 0.
- Generování procesů
 - *adult*: pomocný proces generuje procesy pro dospělé osoby; každý nový proces je generován po uplynutí náhodné doby z intervalu $<0, \ AGT>$; celkem vygeneruje *A* procesů.
 - *child*: pomocný proces generuje procesy pro děti; každý nový proces je generován po uplynutí náhodné doby z intervalu $<0, \ CGT>$; celkem vygeneruje *C* procesů.
 - Každý proces *adult* i *child* bude interně identifikován celým číslem I , začínajícím od 1. Číselná řada je pro každou kategorii procesů zvlášť.
 - postupně tedy vznikne hlavní proces, dva pomocné procesy, *A* procesů dospělých osob a *C* procesů dětí
- Každý proces *adult* i *child* vykonává své akce a současně zapisuje informace o akcích do souboru s názvem `proj2.out`.

- Přístup k výstupnímu zařízení (zápis informací) musí být výlučný.
- Součástí výstupních informací o akci je pořadové číslo *A* prováděné akce (viz popis výstupů). Akce se číslují od jedničky.
- Použijte sdílenou paměť pro implementaci čítače akcí a sdílených proměnných nutných pro synchronizaci.
- Použijte *semafory* pro synchronizaci procesů.
- *Nepoužívejte aktivní čekání* (včetně cyklického časového uspání procesu) pro účely synchronizace.
- Procesy, které již dokončily všechny akce, čekají na všechny ostatní procesy; všechny procesy *adult* i *child* se ukončí současně.
- Budete-li potřebovat generovat unikátní klíč, je vhodné použít funkci [ftok](#).
- Další funkce a systémová volání: [fork](#), [wait](#), [shmat](#), [semctl](#), [semget](#), [shmget](#), [sem_open](#), [usleep](#), ...

Chybové stavy

- Pokud některý ze vstupů nebude odpovídat očekávanému formátu nebo bude mimo povolený rozsah, program vytiskne chybové hlášení na standardní chybový výstup, uvolní všechny dosud alokované zdroje a ukončí se s kódem (*exit code*) 1.
- Pokud selže systémové volání, program vytiskne chybové hlášení na standardní chybový výstup, uvolní všechny alokované zdroje a ukončí se s kódem (*exit code*) 2.

Popis procesů a jejich výstupů

Poznámka k výstupům:

- *A* je pořadové číslo prováděné akce,
- *NAME* je zkratka kategorie příslušného procesu, tj. *A* pro *adult* a *C* pro *child*,
- *I* je interní identifikátor procesu v rámci příslušné kategorie,
- *CA* je počet procesů *adult* aktuálně přítomných v centru včetně těch, kteří chtějí odejít, ale zatím musí zůstat,
- *CC* je počet procesů *child* aktuálně přítomných v centru.
- Při vyhodnocování výstupu budou ignorovány mezery a tabelátory.

Proces adult

- Po spuštění tiskne *A: NAME I: started*.
- Proces vstupuje do centra. Po vstupu tiskne *A: NAME I: enter*.
- Proces simuluje aktivitu v centru tak, že se uspí na náhodnou dobu z intervalu $<0, AWT>$.
- Po probuzení tiskne proces *A: NAME I: trying to leave* a pokusí se opustit centrum:
 - pokud by po jeho opuštění byla porušena podmínka, tj. $\text{počet_dětí} > \text{počet_dospělých} * 3$
 - proces nemůže opustit centrum a musí počkat, až centrum opustí příslušný počet dětí. Proces tiskne *A: NAME I: waiting: CA: CC*
 - jakmile centrum opustí dostatečný počet dětí, proces opustí centrum a poté tiskne *A: NAME I: leave*
 - pokud by po jeho opuštění nebyla porušena podmínka, proces opustí centrum a poté tiskne *A: NAME I: leave*
- Těsně před ukončením proces tiskne *A: NAME I: finished*.

Proces child

- Po spuštění tiskne *A: NAME I: started*.
- Proces se pokusí vstoupit do centra:
 - pokud by po jeho vstupu byla porušena podmínka, tj. $\text{počet_dětí} > \text{počet_dospělých} * 3$
 - proces nemůže vstoupit a musí počkat, až do centra vstoupí další dospělý. Proces tiskne *A: NAME I: waiting: CA: CC*

2. jakmile do centra vstoupí dospělý a vstupem dítěte by nebyla porušena podmínka, vstoupí proces do centra a poté tiskne A: NAME I: enter
2. pokud by po jeho vstupu nebyla porušena podmínka, proces vstoupí do centra a poté tiskne A: NAME I: enter
3. Proces simuluje aktivitu v centru tak, že se uspí na náhodnou dobu z intervalu $<0, CWT>$.
4. Po probuzení tiskne proces A: NAME I: trying to leave a poté opouští centrum.
5. Po opuštění centra tiskne A: NAME I: leave.
6. Těsně před ukončením proces tiskne A: NAME I: finished.

Společné podmínky

1. Všechny procesy *adult* a *child* se ukončí současně, tj. čekají, až všichni dokončí operace a opustí centrum. Teprve poté tisknou informaci ... finished.
2. Pokud má AGT nebo CGT hodnotu 0, znamená to, že se všechny příslušné procesy vygenerují ihned.
3. Pokud má AWT nebo CWT hodnotu 0, znamená to, že na příslušném místě nedojde k uspaní procesu.
4. Při vcházení do centra a odcházení z centra nezáleží na pořadí procesů - pokud např. proces *child* čeká na vstup jako první, nemusí nutně vstoupit jako první.
5. Mezi výstupem trying to leave a leave (příp. waiting) **stejněho procesu** se nesmí objevit výstup enter žádného procesu.
6. **Problém zablokování běhu aplikace:** Může nastat situace, kdy v centru nejsou žádné procesy *adult*, další se již negenerují (jejich počet byl vyčerpán), přesto vznikají procesy *child*, které chtějí vstoupit. V takovém případě by došlo k zablokování. Implementujte řešení tak, aby v této situaci nebyla aplikována podmínka centra, tj. procesy *child* mohly vstupovat a vystupovat bez omezení.

Ukázka výstupů

Ukázka č. 1

Spuštění: \$./proj2 1 2 0 0 0 0

Výstup (proj2.out):

```
1 : A 1 : started
2 : A 1 : enter
3 : C 2 : started
4 : C 1 : started
5 : C 2 : enter
6 : A 1 : trying to leave
7 : A 1 : waiting : 1 : 1
8 : C 1 : enter
9 : C 2 : trying to leave
10 : C 2 : leave
11 : C 1 : trying to leave
12 : C 1 : leave
13 : A 1 : leave
14 : A 1 : finished
15 : C 2 : finished
16 : C 1 : finished
```

Ukázka č. 2

Spuštění: \$./proj2 1 4 1 10 5 5

Výstup (proj2.out):

```
1 : A 1 : started
2 : A 1 : enter
3 : A 1 : trying to leave
```

```
4 : A 1 : leave
5 : C 1 : started
6 : C 1 : enter
7 : C 1 : trying to leave
8 : C 1 : leave
9 : C 2 : started
10 : C 2 : enter
11 : C 2 : trying to leave
12 : C 2 : leave
13 : C 3 : started
14 : C 3 : enter
15 : C 3 : trying to leave
16 : C 3 : leave
17 : C 4 : started
18 : C 4 : enter
19 : C 4 : trying to leave
20 : C 4 : leave
21 : C 2 : finished
22 : A 1 : finished
23 : C 1 : finished
24 : C 4 : finished
25 : C 3 : finished
```

Podmínky vypracování projektu

Obecné informace

- Projekt implementujte v jazyce C.
- Komentujte zdrojové kódy, programujte přehledně. Součástí hodnocení bude i kvalita zdrojového kódu.
- Kontrolujte, zda se všechny procesy ukončují korektně a zda při ukončování správně uvolňujete všechny alokované zdroje (např. příkazem `ipcs` můžete zjistit, jaké zdroje System V jsou v systému alokovány).
- Dodržujte syntax zadaných jmen, formát souborů a formát výstupních dat! Čtěte pozorně zadání a poznámky k vypracování u jednotlivých zadání.
- Projekt musí být přeložitelný a spustitelný na počítači `merlin.fit.vutbr.cz`.
- Dotazy k zadání: Veškeré nejasnosti a dotazy řešte pouze prostřednictvím diskuzního fóra k projektu 2.

Překlad

- Pro překlad používejte nástroj `make`. Součástí odevzdání bude soubor `Makefile`.
- Překlad se provede příkazem `make` v adresáři, kde je umístěn soubor `Makefile`.
- Po překladu vznikne spustitelný soubor se jménem `proj2`, který bude umístěn ve stejném adresáři jako soubor `Makefile`.
- Zdrojové kódy překládejte s přepínači `-std=gnu99 -Wall -Wextra -Werror -pedantic`.

Odevzdání

- Součástí odevzdání budou pouze soubory se zdrojovými kódy (`*.c`, `*.h`) a soubor `Makefile`. Tyto soubory zabalte pomocí nástroje `zip` do archivu s názvem `proj2.zip`.
- Archiv vytvořte tak, aby po rozbalení byl soubor `Makefile` umístěn ve stejném adresáři, jako je archiv.
- Archiv `proj2.zip` odevzdejte prostřednictvím informačního systému, termín *Projekt 2*.
- Pokud nebude dodržena forma odevzdání nebo projekt nepůjde přeložit, bude projekt hodnocen 0 body.
- Archiv odevzdejte pomocí informačního systému v dostatečném předstihu (odevzdaný soubor

můžete před vypršením termínu snadno nahradit jeho novější verzí, kdykoliv budete potřebovat).

Ověřovací skript

Základní skript pro ověření korektnosti výstupního formátu a základní posloupnosti. Informace o skriptu jsou uvedeny v komentáři skriptu. Skript je dodán tak, jak je. Pokud naleznete v testovacím skriptu chybu, napište tuto informaci na fórum.