Технологично училище "Електронни системи"
Технически университет – София

Операционни системи

Задача 3: Реализация на синхронизация на нишки с `mutex`

elsys.os.2013@gmail.com

6 март 2014 г.

1 Условие на задачата

1.1 Основна функционалност (25 точки)

Целта на задачата е да се реализира синхронизация на нишки посредством `mutex`. Подобно на компютърната игра Warcraft, разполагате с работници, всеки от които може да копае злато от мина. Всеки работник влиза в мина, копае известно време, като намалява количеството злато в мината и увеличава количеството злато, което има, след което излиза от мината. Програмата приключва, когато свърши златото в мините.
Изисквания към работниците и мините:

- Всеки работник се изпълнява в отделна нишка.
- Само по един работник може да копае злато от мината по едно и също време.
- Всяка мина има определено количество злато в себе си, което се подава на входа на програмата и е еднакво за всички мини.
- Всеки работник трябва да пази какво количество злато е събрал.
- При всяко влизане в мината, работникът взима по 10 злато.
- Взимането на злато отнема известно време, което трябва да симулирате чрез извикването на `sleep`.
- Синхронизацията на отделните работници трябва да става посредством `mutex`.

При всяко влизане в мина, на стандартния изход трябва да се изпише "Worker N entered mine M а при излизане от мината "Worker N exited mine M". Програмата трябва да изведе общия брой на първоначалното злато в мините и сумата на златото, събрано от всички работници. При правилно изпълнение, тези две стойности ВИНАГИ ще са равни.

На входа се подава количеството злато в мина. Броя на работниците по подразбиране е 2, а броя на мините - 1.

Примерен изход при 50 злато в мина:

```
Worker 1 entered mine 1
Worker 1 exited mine 1
Worker 2 entered mine 1
Worker 2 exited mine 1
Worker 2 entered mine 1
Worker 2 exited mine 1
Worker 1 entered mine 1
Worker 1 exited mine 1
Worker 2 entered mine 1
Worker 2 exited mine 1
All gold 50, gold collected 50
```

1.2 Поддръжка на N на брой работници (25 точки)

На входа освен количеството злато във всяка мина се подава и броя работници. Примерен изход при 50 злато в мина и петима работници:

```
Worker 5 entered mine 1
Worker 5 exited mine 1
Worker 3 entered mine 1
Worker 3 exited mine 1
Worker 1 entered mine 1
Worker 1 exited mine 1
Worker 2 entered mine 1
```

```
Worker 2 exited mine 1
Worker 3 entered mine 1
Worker 3 exited mine 1
All gold 50, gold collected 50
```

1.3 Поддръжка на M на брой мини (25 точки)

На входа освен количеството злато във всяка мина и броя работници на работници се подава и броя на мините. Примерен изход при 20 злато в мина, трима работници и две мини:

```
Worker 1 entered mine 1
Worker 2 entered mine 2
Worker 1 exited mine 1
Worker 3 entered mine 1
Worker 2 exited mine 2
Worker 2 entered mine 2
Worker 3 exited mine 1
Worker 3 entered mine 2
Worker 3 exited mine 2
All gold 40, gold collected 40
```

1.4 Обработка на грешки (25 точки)

Ако при извикване на pthread_create, pthread_join, pthread_wait или pthread_mutex_init възникне грешка, трябва да се изведе съобщение.

2 Изисквания към решението и оценяване

1. Програмата трябва да бъде написана на езика `C` съгласно ISO/IEC 9899:1999.
2. Правилата за оценяване са следните. Приемаме, че напълно коректна и написана спрямо изискванията програма получава максималния брой точки — 100% или 100 точки. Ако в решението има пропуски, максималният брой точки ще бъде намален съгласно правилата описани по-долу.
3. За работа с нишки трябва да се използва семейството от функции `pthread_create()`, `pthread_join()`, `pthread_wait()` и т.н.
4. Задължително към файловете с решението трябва да е приложен и `Makefile`. Изпълнимият файл, който се създава по време на компилация на решението, трябва да се казва `wc3`.
5. При проверка на решението програмата ви ще бъде компилирани и тествана по следния начин:

   ```
   make
   ./wc3 50 5 2
   ```

   Предходната процедура ще бъде изпълнена няколко пъти с различни входни данни за да се провери дали вашата програма работи коректно.
6. Реализацията на програмата трябва да спазва точно изискванията описани по-горе. Всяко отклонение от изискванията ще доведе до получаване на 0 точки за съответната част от условието.
7. Работи, които са предадени по-късно от обявеното (или не са предадени), ще бъдат оценени с 0 точки.
8. Програмата ви трябва да съдържа достатъчно коментари. Оценката на решения без коментари или с недостатъчно и/или мъгляви коментари ще бъде намалена с 30%.

9. Всеки файл от решението трябва да започва със следният коментар:

```
//------------------------------------------
// NAME: Ivan Ivanov
// CLASS: Xia
// NUMBER: 13
// PROBLEM: #3
// FILE NAME: xxxxxx.yyy.zzz (unix file name)
// FILE PURPOSE:
//  няколко реда, които описват накратко
//  предназначението на файла
//  ...
//------------------------------------------
```

   Всяка функция във вашата програма трябва да включва кратко описание в следния формат:

```
//------------------------------------------
// FUNCTION: xxyyzz (име на функцията)
//  предназначение на функцията
// PARAMETERS:
//  списък с параметрите на функцията
//  и тяхното значение
//------------------------------------------
```

10. Лош стил на програмиране и липсващи заглавни коментари ще ви костват 30%.

11. Програми, които не се компилират получават 0 точки. Под „не се компилират“ се има предвид произволна причина, която може да причини неуспешна компилация, включително липсващи файлове, неправилни имена на файлове, синтактични грешки, неправилен или липсващ `Makefile`, и т.н. Обърнете внимание, че в `UNIX` имената на файловете са case sensitive.

12. Програми, които се компилират, но не работят, не могат да получат повече от 50%. Под „компилира се, но не работи“ се има предвид, че вие сте се опитали да решите проблема до известна степен, но не сте успели да направите пълно решение. Често срещан проблем, който спада към този случай, е че вашият Makefile генерира изпълним файл, но той е именуван с име, различно от очакваното (т.е. `head` в разглеждания случай).

13. Безсмислени или мъгляви програми ще бъдат оценявани с 0 точки, независимо че се компилират.

14. Програми, които дават неправилни или непълни резултати, или програми, в които изходът и/или форматирането се различава от изискванията ще получат не повече от 70%.

15. Всички наказателни точки се сумират. Например, ако вашата програма няма задължителните коментари в началото на файлове и функциите се отнемат 30%, ако няма достатъчно коментари се отнемат още 30%, компилира се, но не работи правилно — още 30%, то тогава резултатът ще бъде: $50 * (100 - 30 - 30 - 30)\% = 50 * 10\% = 5$ точки

16. Работете самостоятелно. Групи от работи, които имат твърде много прилики една с друга, ще бъдат оценявани с 0 точки.