

Отчет
Итоговое домашнее задание №2
Выполнила: Захарова Наталья Владимировна
БПИ223

**Синхронизация параллельных процессов в ОС Linux с
использованием семафоров**

Вариант № 31

Условие:

Задача о Пути Кулака. На седых склонах Гималаев стоит древний буддистский монастырь: Гуань-Инь-Янь. Каждый год в день сошествия на землю боддисатвы монахи монастыря собираются на совместное празднество и показывают свое совершенствование на Пути Кулака. Всех соревнующихся монахов первоначально разбивают на пары. Бои продолжаются до выявления победителя. Монах который победил в финальном бою, забирает себе на хранение статую боддисатвы.

Реализовать многопроцессное приложение, определяющего победителя.

В качестве входных данных используется массив, в котором хранится количество энергии Ци каждого монаха. При победе монах забирает энергию Ци своего противника. Новые пары образуются среди победителей других пар в порядке завершения поединков. То есть, возможна ситуация, когда бойцы, участвующие в поединке могут быстро победить и начать биться с другими, в то время как поединки начавшиеся ранее, могут продолжаться. Причем длительное время. Каждый поединок протекает некоторое случайное время, которое пропорционально отношению энергии Ци побеж- денного к энергии Ци победителя, умноженному на поправочный коэффициент, позволяющий отслеживать протекание поединка на экране дисплея (например, путем умножения этого отношения на 1000 миллисекунд или другое более удобное значение).

Ожидаемая оценка: 9 баллов

Соответствие проделанной работы и выполненным критериям

Описание общих алгоритмов работы

Для модели бойцов используется структура Fighter, которая содержит в полях всю необходимую информацию об определенном бойце.

```
struct Fighter {  
    int strength;  
    bool defeated;  
  
    explicit Fighter(int str) : strength(str), defeated(false) {}  
};
```

Функция fight моделирует бой между двумя бойцами. Она засыпает на время, равное доле силы бойца 2 относительно силы бойца 1. После этого определяет победителя, опираясь на силы бойцов и обновляет его силу. Семафор semaphore используется для синхронизации доступа к общим данным.

```
void fight(Fighter& fighter1, Fighter& fighter2, sem_t& semaphore) {  
    std::this_thread::sleep_for(std::chrono::milliseconds (fighter2.strength / fighter1.strength *  
1000));  
  
    sem_wait(&semaphore);  
  
    sem_post(&semaphore);  
}  
...
```

Функция `finalFight` определяет победителей первых трех боев и устраивает между ними финальные поединки. Она также использует семафор для синхронизации доступа к общим данным.

```
void finalFight(std::vector<Fighter>& fighters, sem_t& semaphore) {  
}
```

На 4-5 баллов:

- Разработана программа, которая использует множество процессов взаимодействующих с использованием именованных POSIX семафоров. Обмен данными ведется через разделяемую память в стандарте POSIX.

Описание входных данных

В терминале пользователю предлагается ввести энергию Ций каждого из бойцов, после чего начинается соревнование

Описание алгоритма работы программы

- Создаются три потока, каждый из которых управляет парным боем между двумя бойцами из массива. В каждом потоке запускается функция `fight`, где бойцы сражаются, а доступ к их силе синхронизируется через семафор.
- В функции `fight` происходит имитация боя с временной задержкой, которая отражает продолжительность боя. После этого происходит состязание на силу между двумя бойцами, где победитель забирает силу проигравшего. Семафор применяется для того, чтобы обеспечить безопасный доступ к общим ресурсам.
- В функции `finalFight` выбираются победители первых трех боев, которые затем вступают в финальный бой: победитель первого боя сражается с победителем второго, а их победитель — с победителем третьего боя.
- В `main` основной поток дожидается завершения всех трех парных боев.
- После окончания всех боев определяется сила финального победителя и выводится результат.
- В завершение программы все семафоры уничтожаются для того, чтобы освободить использованные ресурсы.

Описание кода программы

Ввод необходимых данных.

```
std::vector<Fighter> fighters;
for (int i = 0; i < numFighters; ++i) {
    int strength;
    printf("Введите энергию Ций бойца %d: ", i + 1);
    std::cin >> strength;
    fighters.emplace_back(strength);
}
printf("\nИнформация об энергиях Ций бойцов:\n");
for (int i = 0; i < numFighters; ++i) {
    printf("Боец %d: %d\n", i + 1, fighters[i].strength);
}
printf("\n");
```

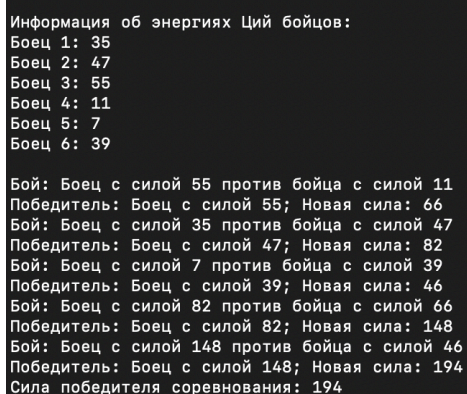
Для начала соревнования создаются три потока для трех парных боев между бойцами. Каждый поток запускает функцию `fight`, передавая каждого из бойцов и семафор в качестве входных данных.

```
std::thread fights[3];
for (int i = 0; i < 3; ++i) {
    fights[i] = std::thread(fight, std::ref(fighters[i * 2]), std::ref(fighters[i * 2 + 1]),
std::ref(semaphore));
}
```

Определение победителя соревнований и вывод итоговой силы победившего бойца

```
int winner_strength = -1;
for (const auto& fighter : fighters) {
    if (!fighter.defeated) {
        winner_strength = fighter.strength;
        break;
    }
}
if (winner_strength != -1) {
    std::cout << "Сила победителя соревнования: " << winner_strength << std::endl;
} else {
    std::cout << "Все бойцы были побеждены." << std::endl;
}
```

Пример работы программы



```
Информация об энергиях Ций бойцов:
Боец 1: 35
Боец 2: 47
Боец 3: 55
Боец 4: 11
Боец 5: 7
Боец 6: 39

Бой: Боец с силой 55 против бойца с силой 11
Победитель: Боец с силой 55; Новая сила: 66
Бой: Боец с силой 35 против бойца с силой 47
Победитель: Боец с силой 47; Новая сила: 82
Бой: Боец с силой 7 против бойца с силой 39
Победитель: Боец с силой 39; Новая сила: 46
Бой: Боец с силой 82 против бойца с силой 66
Победитель: Боец с силой 82; Новая сила: 148
Бой: Боец с силой 148 против бойца с силой 46
Победитель: Боец с силой 148; Новая сила: 194
Сила победителя соревнования: 194
```

На 6-7 баллов:

- Разработана программа, которая использует множество процессов взаимодействующих с использованием неименованных POSIX семафоров

расположенных в разделяемой памяти. Обмен данными также ведется через разделяемую память в стандарте POSIX.

Описание входных данных

В терминале пользователю предлагается ввести энергию Ций каждого из бойцов, после чего начинается соревнование

Описание алгоритма работы программы

- В функции `main()` инициализация именованного семафора происходит через вызов `sem_open()` с именем `/fighter_semaphore`. Семафор устанавливается с начальным значением 0, указывая на отсутствие изначального доступа к общим ресурсам.
- Этот семафор применяется для координации потоков во время имитации боев.
- Каждый боевой поток использует `sem_wait()` для ожидания уведомления от остальных потоков о завершении их боев.
- По окончании своего боя каждый поток активирует `sem_post()`, сигнализируя остальным потокам о завершении боя.
- Функция `finalFight()` также задействует семафор для координации ожидания завершения поединка между победителями первых трех боев.

Описание кода программы

Каждый поток боя вызывает `sem_wait()`, чтобы ожидать сигнала от других потоков о завершении боя. После завершения боя каждый поток вызывает `sem_post()`, чтобы сигнализировать о том, что бой завершен.

Сигнал о завершении боя
`sem_post(semaphore);`

После завершения всех операций с семафором он закрывается и удаляется с помощью `sem_close()` и `sem_unlink()`.

```
// Закрытие семафора  
sem_close(semaphore);  
// Удаление семафора  
sem_unlink(SEMAPHORE_NAME);
```

Определение победителя соревнований и вывод итоговой силы победившего бойца

```
int winner_strength = -1;  
for (const auto& fighter : fighters) {
```

```

    if (!fighter.defeated) {
        winner_strength = fighter.strength;
        break;
    }
}
if (winner_strength != -1) {
    std::cout << "Сила победителя соревнования: " << winner_strength << std::endl;
} else {
    std::cout << "Все бойцы были побеждены." << std::endl;
}

```

```

Last login: Mon Apr 22 13:53:16 on ttys002
natalazaharova@MacBook-Air-5 point7 % g++ -o main main.cpp -pthread -std=c++11
natalazaharova@MacBook-Air-5 point7 % ./main
Введите энергию Ций бойца 1: 13
Введите энергию Ций бойца 2: 20
Введите энергию Ций бойца 3: 100
Введите энергию Ций бойца 4: 38
Введите энергию Ций бойца 5: 30
Введите энергию Ций бойца 6: 60

Информация об энергиях Ций бойцов:
Боец 1: 13
Боец 2: 20
Боец 3: 100
Боец 4: 38
Боец 5: 30
Боец 6: 60

Бой: Боец с силой 100 против бойца с силой 38
Победитель: Боец с силой 100; Новая сила: 138
Бой: Боец с силой 13 против бойца с силой 20
Победитель: Боец с силой 20; Новая сила: 33
Бой: Боец с силой 30 против бойца с силой 60
Победитель: Боец с силой 60; Новая сила: 90
Бой: Боец с силой 33 против бойца с силой 138
Победитель: Боец с силой 138; Новая сила: 171
Финальный бой: Боец 2 против Бойца 3
Сила победителя соревнования: 171
natalazaharova@MacBook-Air-5 point7 %

```

На 8 баллов:

- Разработана программа, которая использует множество независимых процессов взаимодействующих с использованием семафоров в стандарте UNIX SYSTEM V. Обмен данными ведется через разделяемую память в стандарте UNIX SYSTEM V.

Описание входных данных

В терминале пользователю предлагается ввести энергию Ций каждого из бойцов, после чего начинается соревнование

Описание алгоритма работы программы

- Изначально создается именованный канал `FIGHT_CHANNEL`, который предназначен для общения между разными программами.
- Одновременно стартуют бои для первых трех пар бойцов, где каждый из боев ведется через отдельный канал.
- После окончания трех боев осуществляется проверка на наличие победителя в каждой паре, и данные о победителях отправляются в `FIGHT_CHANNEL`. В то же время, основной поток ждет завершения всех боевых потоков и получает информацию о победителях из `FIGHT_CHANNEL`.
- Затем проходит финальный бой между победителями предыдущих трех боев.

Описание кода программы

Сначала создается именованный канал `FIGHT_CHANNEL`, который будет использоваться в программе далее.

```
const char* FIGHT_CHANNEL = "/tmp/fight_channel";  
mkfifo(FIGHT_CHANNEL, 0666);
```

Запускается 3 потока `fights`, каждый из которых отвечает за проведение одного боя.

```
sem_t* semaphore;  
semaphore = sem_open(SEMAPHORE_NAME, O_CREAT, 0666, 0);  
  
std::thread fights[3];  
for (int i = 0; i < 3; ++i) {  
    fights[i] = std::thread(fight, std::ref(fighters[i * 2]), std::ref(fighters[i * 2 + 1]),  
        semaphore);  
}
```

Вторая программа открывает именованный канал FIGHT_CHANNEL для чтения результатов боев.

```
int fd = open(FIGHT_CHANNEL, O_RDONLY);
```

Далее считываются индексы победителей из именованного канала и выводит информацию о боях на экран.

```
int fighterIndex;  
while (read(fd, &fighterIndex, sizeof(int)) > 0) {  
    std::cout << "Победитель боя: Боец " << fighterIndex + 1 << std::endl;  
}
```

```

natalazaharova@MacBook-Air-5 point8 % ./main
Введите энергию Ций бойца 1: 13
Введите энергию Ций бойца 2: 15
Введите энергию Ций бойца 3: 20
Введите энергию Ций бойца 4: 60
Введите энергию Ций бойца 5: 10
Введите энергию Ций бойца 6: 40

Информация об энергиях Ций бойцов:
Боец 1: 13
Боец 2: 15
Боец 3: 20
Боец 4: 60
Боец 5: 10
Боец 6: 40

Бой: Боец с силой 13 против бойца с силой 15
Победитель: Боец с силой 15; Новая сила: 28
Бой: Боец с силой 20 против бойца с силой 60
Победитель: Боец с силой 60; Новая сила: 80
Бой: Боец с силой 10 против бойца с силой 40
Победитель: Боец с силой 40; Новая сила: 50
Бой: Боец с силой 28 против бойца с силой 80
Победитель: Боец с силой 80; Новая сила: 108
Финальный бой: Боец 2 vs Бойца 4
Сила победителя соревнования: 108
natalazaharova@MacBook-Air-5 point8 %

```

На 9 баллов:

- Разработана программа, которая использует множество независимых процессов взаимодействующих с использованием именованных POSIX семафоров. Обмен данными ведется через каналы в стандарте POSIX.

Описание входных данных

В терминале пользователю предлагается ввести энергию Ций каждого из бойцов, после чего начинается соревнование

Описание алгоритма работы программы

- Инициализируется именованный канал `FIGHT_CHANNEL`, предназначенный для обмена данными между различными программами.
- Первые три пары бойцов начинают свои бои одновременно, используя отдельные каналы для каждого боя.
- По завершении трех парных боев в каждом бою определяется победитель, который затем записывается в канал `FIGHT_CHANNEL`. В это время основной поток ожидает окончания всех боевых потоков и извлекает информацию о победителях из канала `FIGHT_CHANNEL`.
- Затем проводится финальный бой между победителями предыдущих трех боев.

Описание кода программы

Сначала создается именованный канал `FIGHT_CHANNEL`, который будет использоваться в программе далее.

```
const char* FIGHT_CHANNEL = "/tmp/fight_channel";  
mkfifo(FIGHT_CHANNEL, 0666);
```

Запускается три потока `fights`, каждый из которых отвечает за проведение одного боя.

```
sem_t* semaphore;  
semaphore = sem_open(SEMAPHORE_NAME, O_CREAT, 0666, 0);
```

```
std::thread fights[3];  
for (int i = 0; i < 3; ++i) {  
    fights[i] = std::thread(fight, std::ref(fighters[i * 2]), std::ref(fighters[i * 2 + 1]),  
        semaphore);  
}
```

Вторая программа открывает именованный канал `FIGHT_CHANNEL` для чтения результатов боев.

```
int fd = open(FIGHT_CHANNEL, O_RDONLY);
```

Далее считываются индексы победителей из именованного канала и выводит информацию о боях на экран.

```
int fighterIndex;  
while (read(fd, &fighterIndex, sizeof(int)) > 0) {  
    std::cout << "Победитель боя: Боец " << fighterIndex + 1 << std::endl;  
}
```