Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский университет "Высшая школа экономики"

Факультет компьютерных наук

Департамент программной инженерии

Отчет по домашнему заданию

По дисциплине «Архитектура вычислительных систем»

Выполнил:

Студент БПИ195

Тимохин Никита

Условие задачи

5. Задача о каннибалах. Племя из n дикарей ест вместе из большого горшка, который вмещает m кусков тушеного миссионера. Когда дикарь хочет обедать, он ест из горшка один кусок, если только горшок не пуст, иначе дикарь будит повара и ждет, пока тот не наполнит горшок. Повар, сварив обед, засыпает. Создать многопоточное приложение, моделирующее обед дикарей. При решении задачи пользоваться семафорами.

Описание Решения

В этой задаче я решил воспользоваться моделью производитель – потребитель. В качестве производителя выступает повар. Он производит пищу и передает ее в горшок. Потребителем выступают дикари. Они берут еду из горшка и съедают. Если еда в горшке отсутствует, то потребитель должен оповестить производителя и ждать пока еда не появится. Повар оповестит дикарей, когда еда будет готова. Чтобы программа завершалась я решил, что повар будет уставать после 3х готовок. Я решил, что после того, как повар устанет, дикари поедят только 1 раз. Иначе это может привести к ошибке (т. к. дикари не организованный народ очереди у них нет и если они видят еду, то сразу рвутся ее взять. Тогда может получиться так, что 2 дикаря увидят один кусок и оба захотят его. Один из них его съест, а второй будет бесконечно ждать. Если бы программа работала бесконечно, то такого бы не возникало).

Для организации готовки (положить элемент в буфер) и поедания (взять элемент из буфера) используются mutex'ы.

```
pthread_mutex_lock(&mutexF);
// Уменьшаем значение семафора и ждем пока все съедят.
sem_wait(&empty);
printf("The cook prepared the food. Now there are %d servings.\n", m);
buf = m;  // Заполняем горшок(put item)
numOfCooking++;
for (int i = 0; i < m; i++) {
    // Увеличиваем значение семафора и разрешаем дикарям есть.
    sem_post(&full);
}
if (numOfCooking >= 3){
    isCookTired = true;
}
pthread_mutex_unlock(&mutexF);
```

```
pthread_mutex_lock(&mutexS);
// Уменьшаем значение семафора. Если горшочек пуст - ждум пока повар приготовит.
sem_wait(&full);
printf("Barbarian number %d ate. %d servings left.\n", cNum + 1, buf - 1);
// Уменьшаем число порций в горшке.
buf--;
// Если еда закончилась
if (buf == 0) {
    // Увеличивает значение семафора и говорит повору о том, что пора работать.
    sem_post(&empty);
}
pthread_mutex_unlock(&mutexS);
```

Для взаимодействия потоков используются семафоры. Семафор full показывает, насколько буфер полон. Семафор empty показывает, пустой ли буфер (начинать ли повару работать).

```
for (int i = 0; i < m; i++) {
    // Увеличиваем значение семафора и разрешаем дикарям есть.
    sem_post(&full);
}
```

```
// Увеличивает значение семафора и говорит повору о том, что пора работать.
sem_post(&empty);
```

Ввод производится с клавиатуры во время работы программы. Ниже представлена обработка некорректных данных.

```
cout << "enter the capacity of the pot\n";
cin >> m;
while (m < 1){
    cout << "wrong input. m >= 1. Try again.\n";
    cin >> m;
}
cout << "enter number of barbarians\n";
cin >> n;
while (n < 1){
    cout << "wrong input. n >= 1. Try again.\n";
    cin >> n;
}
```

```
enter the capacity of the pot
-1
wrong input. m >= 1. Try again.
```

Тестирование программы

Обработка некорректного ввода показана выше. Далее несколько примеров работы при корректном вводе.

Тест 1.

```
enter the capacity of the pot
5
enter number of barbarians
5
The cook prepared the food. Now there are 5 servings.
Barbarian number 2 ate. 4 servings left.
Barbarian number 3 ate. 3 servings left.
Barbarian number 1 ate. 2 servings left.
Barbarian number 4 ate. 1 servings left.
Barbarian number 5 ate. 0 servings left.
The cook prepared the food. Now there are 5 servings.
Barbarian number 2 ate. 4 servings left.
Barbarian number 1 ate. 3 servings left.
Barbarian number 5 ate. 2 servings left.
Barbarian number 3 ate. 1 servings left.
Barbarian number 4 ate. 0 servings left.
The cook prepared the food. Now there are 5 servings.
Barbarian number 2 ate. 4 servings left.
Barbarian number 1 ate. 3 servings left.
Barbarian number 3 ate. 2 servings left.
Barbarian number 5 ate. 1 servings left.
Barbarian number 5 ate. 1 servings left.
```

Тест 2.

```
enter the capacity of the pot
3
enter number of barbarians
5
The cook prepared the food. Now there are 3 servings.
Barbarian number 2 ate. 2 servings left.
Barbarian number 5 ate. 1 servings left.
Barbarian number 4 ate. 0 servings left.
The cook prepared the food. Now there are 3 servings.
Barbarian number 1 ate. 2 servings left.
Barbarian number 3 ate. 1 servings left.
Barbarian number 2 ate. 0 servings left.
The cook prepared the food. Now there are 3 servings.
Barbarian number 4 ate. 2 servings left.
Barbarian number 5 ate. 1 servings left.
Barbarian number 1 ate. 0 servings left.
```

Источники

Большую часть информации, нужной для разработке этого приложения я получил из семинара «Многопоточное программирование. Синхронизация» http://www.softcraft.ru/edu/comparch/practice/thread/02-sync/

В частности, из этого кода. http://www.softcraft.ru/edu/comparch/practice/thread/02-sync/readwriters01/main.cpp

Также освежал в памяти информацию отсюда http://www.softcraft.ru/edu/comparch/lect/07-parthread/multitreading.pdf