Московский авиационный институт
Национальный исследовательский университет

Операционные системы
Кафедра 806

Курсовой проект

Студент: Минибаев Айдар
Группа: М8О-301Б-18
Преподаватель: Миронов Е.С.
Дата: .
Подпись: .

Москва 2020

1. Постановка задачи

Необходимо написать 3-и программы. Далее будем обозначать эти программы A, B, C.

Программа A принимает из стандартного потока ввода строки, а далее их отправляет программе C. Отправка строк должна производится построчно. Программа C печатает в стандартный вывод, полученную строку от программы A. После получения программа C отправляет программе A сообщение о том, что строка получена. До тех пор пока программа A не примет «сообщение о получение строки» от программы C, она не может отправлять следующую строку программе C.

Программа B пишет в стандартный вывод количество отправленных символов программой A и количество принятых символов программой C. Данную информацию программа B получает от программ A и C соответственно.

Способ организация межпроцессорного взаимодействия выбирает студент.

2. Метод решения

1. Программа A.out считывает из стандартного потока ввода строку, пишет ее в файл и меняет значения семафоров так, чтобы программа A.out находилась в активном ожидании получения ответа от программы Cout, а с программы C.out снимается блокировка.
2. Программа C.out считывает строку из файла, выводит ее в стандартный поток вывода, а также размер отправленной строки
3. Программа B.out пишет в стандартный вывод количество отправленных символов.

3. Программная реализация

Программа A.out

```
#include "sem_lib.h"

#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <iostream>
#include <string>
#include <fstream>

using namespace std;

#define FILE_NAME "buffer.txt"

void exit_signal(Semaphore_manager& sm_C) {
   ofstream i(FILE_NAME);
   i << "exit" << "\n" << -1 << endl;
   sm_C.post();
   i.close();
```

```
}

int main() {
   ofstream output(FILE_NAME);
   output.close(); // удаление данных из файла

   Semaphore_manager sm_A("A"), sm_C("C");

   string line;
   cout << ">> ";
   while(getline(cin, line)) {
       ofstream output(FILE_NAME); // открытие файла на перезапись
       cout << "sending..." << endl;
       output << line << "\n" << line.size() << endl; // запись строки и ее
размера в файл
       sm_C.post();
       sm_A.wait();
       cout << "done" << endl;
       output.close();
       cout << ">> ";
   }

   exit_signal(sm_C);
   return 0;
}
```

Программа B.out

```
#include "sem_lib.h"

#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <iostream>
#include <string>
#include <fstream>
#include <algorithm>

using namespace std;

#define FILE_NAME "buf_size.txt"

// структура Count лучше, чем pair<size_t, sizt_t>,
// т.к. во втором случае поля first и second не так выразительны
как post и received
struct Count {
   size_t post; // отправленное количество символов
   size_t received; // полученное количество символов

   bool equal() const {
       return post == received;
```

```
    }
};

size_t parse_size(ifstream& input) {
    string line;
    getline(input, line); // чтение строки
    return atoi(line.c_str()); // преобразование к числу
}

void print_result(const Count& count) {
    cout << "count.post(" << count.post << ") ";
    if(count.equal()) {
        cout << "==";
    } else {
        cout << "!=";
    }
    cout << " count.received(" << count.received << ")" << endl;
}


int main() {
    Semaphore_manager sm_B("B"), sm_C("C");
    Count count;
    string line;
    while(true) {
        sm_B.wait(); // ожидание введения строки из программы A.out
        ifstream input(FILE_NAME); // открытие файла

        count.post = parse_size(input);
        count.received = parse_size(input);
        if(count.post == 0 || count.received == 0) {
            break;
        }
        print_result(count);

        input.close(); // закрытие файла
    }

    return 0;
}
```

Программа C.out

```
#include "sem_lib.h"

#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <iostream>
#include <string>
```

```cpp
#include <fstream>
#include <algorithm>

using namespace std;

#define FILE_NAME "buffer.txt"

// структура Count лучше, чем pair<size_t, sizt_t>,
// т.к. во втором случае поля first и second не так выразительны как post и received
struct Count {
    size_t post; // отправленное количество символов
    size_t received; // полученное количество символов

    bool equal() const {
        return post == received;
    }
};

int64_t parse_size(ifstream& input) {
    string line;
    getline(input, line); // чтение числа отправленных символов
    return atoi(line.c_str()); // преобразование к числу
}

void sender_to_prog_B(const Count& count, Semaphore_manager& sm) {
    ofstream input("buf_size.txt");
    input << count.post << "\n" << count.received << endl;
    input.close();
    sm.post();
}

void print_result(const Count& count) {
    if(count.equal()) { // вывод результата в поток ошибок
        cerr << "success" << endl;
    } else {
        cerr << "fail (post != received)" << endl;
    }
}


int main() {
    Semaphore_manager sm_A("A"), sm_B("B"), sm_C("C");
    Count count;
    string line;
    while(true) {
        sm_C.wait(); // ожидание введения строки из программы A.out
        ifstream input(FILE_NAME); // открытие файла

        getline(input, line); // получение строки
        if(line == "exit" && parse_size(input) == -1) { // если переданная строка пустая
            input.close(); // закрытие файла
            sender_to_prog_B({0, 0}, sm_B);
            break;
        }
```

```
        cout << "received line: \"" << line << "\"" << endl; // вывод в стандартный
 поток вывода

        count.received = line.size(); // число прочитанных символов
        count.post = parse_size(input);

        sender_to_prog_B(count, sm_B);
        print_result(count);

        input.close(); // закрытие файла
        sm_A.post(); // особождение мьютекса
    }

    return 0;
}
```

4. Тестирование

Параллельно запустим 3 программы:

```
$ ./A.out
>> ABC
sending...
done
>> QQQQQQQQ
sending...
done
>> 123456789
sending...
done
>> q
sending...
done
>> ^C

$ ./B.out
count.post(3) == count.received(3)
count.post(8) == count.received(8)
count.post(9) == count.received(9)
count.post(1) == count.received(1)
^C

$ ./C.out
received line: "ABC"
success
received line: "QQQQQQQQ"
```

```
success
received line: "123456789"
success
received line: "q"
success
^C
```

5. Выводы

Был написан проект, программы которого взаимодействовали посредством технологии семафора, а также отображения данных в файл.