25.11.22-02.12.22

C++: Модель делегирования

Работаем в аудитории. Результат размещаем на edufpmi в двух версиях:

- в конце пары;
- до 16:00 02.12.2022.

[Из прошлой жизни]
Лабораторная работа 3
Задача. Умножение матрицы на вектор.

Задание 2. Параллельная реализация.
Реализовать в двух вариантах: Модель делегирования 1 и Модель делегирования 2.

Задача

Вычислить количество повторений слова в каждом файле.
Вход: список имен файлов, искомое слово.
Выход: список пар (имя файла, количество повторений).

Задание 1

Ниже представлена Модель делегирования 1 на базе std::thread.
Реализовать на базе std::async.
Провести эксперименты. Сравнить реализации.

```
void count(std::string file_path, std::string word,int& counter) {
	std::ifstream stream(file_path);
	std::vector<std::string> words;
	counter = 0;
	if (stream.is_open()) {
	while (!stream.eof()) {
		std::string temp_;
		stream >> temp_;
		if (temp_ == word) {
			counter++;
		}
	}
	}

}

int main()
{
	std::ifstream fin("files.txt");
	std::vector<std::string> files;

	while (!fin.eof()) {
		std::stringstream buffer;
		std::string file_path;
		fin >> file_path;
		files.push_back(file_path);
	}
	std::vector<int> counters(files.size());
	for (int i = 0; i < files.size(); i++)
	{
```

```
            std::cout << files.at(i) << "\n";
      }
      std::string word = "std::string";
      std::vector<std::thread> threads(files.size());
      for (int i = 0; i < files.size(); i++)
      {
            threads[i]=std::thread(count, files.at(i), word,
std::ref(counters[i]));
      }
      for (int i = 0; i < files.size(); i++)
      {
            threads[i].join();
      }
      for (size_t i = 0; i < counters.size(); i++)
      {
            std::cout << "counter " << i <<" "<< counters[i] << "\n";
      }
}
```

Задание 2

Ниже представлена Модель делегирования 2 на базе WinAPI. Реализовать на C++, объектно-оринтированная реализация. Провести эксперименты. Сравнить реализации.

```
string word_for_find = "quo";
CRITICAL_SECTION criticalSection;
int n = 20;
int p = 4;
queue<string> fileNames;


DWORD ThreadFunction(LPVOID pvParam) {

      while (true)
      {
            EnterCriticalSection(&criticalSection);
            if (fileNames.empty()) {
                  LeaveCriticalSection(&criticalSection);
                  break;
            }
            string fileName = fileNames.front();
            fileNames.pop();
            LeaveCriticalSection(&criticalSection);
            ifstream fin;
            fin.open("files/" + fileName);
            int counter = 0;
            while (!fin.eof())
            {
                  string word;
                  fin >> word;
                  if (word == word_for_find) {
                        counter++;
                  }
            }
            fin.close();
            EnterCriticalSection(&criticalSection);
            cout << fileName << ": " << counter << endl;
```

```
            LeaveCriticalSection(&criticalSection);
      }

      return 0;
}

int main() {
      InitializeCriticalSection(&criticalSection);
      HANDLE* hThreads = new HANDLE[p];
      ifstream fin("files.txt");
      for (int i = 0; i < n; i++) {
            string word;
            fin >> word;
            fileNames.push(word);
      }
      fin.close();
      DWORD dwStartTime = GetTickCount();
      for (int k = 0; k < p; k++) {
            hThreads[k] = (HANDLE)_beginthreadex(NULL,
                  0,
                  (_beginthreadex_proc_type)ThreadFunction,
                  NULL,
                  0,
                  NULL);
            if (hThreads[k] == NULL) {
                  printf("Create Thred %d Error=%d\n", k, GetLastError());
                  return -1;
            }
      }
      WaitForMultipleObjects(p, hThreads, TRUE, INFINITE);
      for (int k = 0; k < p; k++) {
            CloseHandle(hThreads[k]);
      }

      DWORD dwElapsedTime = GetTickCount() - dwStartTime;
      printf("Time = %d\n", dwElapsedTime);

      DeleteCriticalSection(&criticalSection);
      return 0;
}
```

Отчет

В одном текстовом файле: Тексты программ, Результаты экспериментов, Сравнение результатов.