Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра «МОП ЭВМ»

ЛАБОРАТОРНАЯ РАБОТА 4

по дисциплине «Операционные системы и оболочки»

## Процессы, потоки и нити в операционной системе Windows

Студент группы 6ИСб-1 О. Хусейнов

Преподаватель В.А Тихомиров

2018

**Цель работы**: Изучить теоретические вопросы управления потоками в ОС Windows и освоить приемы практической реализации этого управления с использованием системных функций.

**Задание:**

Программа должна запускать другую программу, путь которой указан в командной строке и анализировать код ее завершения.

Выполнение:

При создании процесса в системе появляется новый программный поток, при­надлежащий этому процессу. Вначале любой только что созданный процесс об­ладает лишь одним потоком. Этот поток может создавать новые потоки, а эти новые потоки, в свою очередь, могут создавать другие новые потоки. Процесс продолжает свое существование до тех пор, пока в его владении находится по крайней мере один программный поток (или до тех пор, пока не произойдет что-либо, в результате чего весь процесс или все задание не прекратит работу, напри­мер, обращение к функции **TerminateProcess**).

Зачем процессу несколько потоков? Потоки могут выполнять какие-либо действия в фоновом режиме относительно вашей основной программы. Например, вы можете создать новый программный поток, который будет в фоновом режи­ме осуществлять вывод информации на принтер. Потоки удобно использовать также в случае, если блокирование или подвисание какой-либо процедуры не должно стать причиной нарушений функционирования основной программы. Например, в то время как основная программа выполняет сложные математичес­кие вычисления, отдельный программный поток может осуществлять обмен дан­ными через асинхронный последовательный канал связи (например, через модем). В случае замедления передачи данных через канал или в случае подвисания мо­дема функционирование основной программы не будет нарушено.

Базовый системный вызов, предназначенный для создания потока, — это **CreateThread**. Однако на практике программисты фактически никогда его не использу­ют. Почему? Потому что этот вызов создает абсолютно пустой поток, который в изначальном виде не может использоваться вашей библиотекой (С или MFC).

Программа запускается из командной строки с параметром, содержащим путь к файлу, который необходимо запустить.

Программа запускает файл и останавливает поток до тех пор, пока объект идентификатор не окажется в "сигнальном" состоянии. Код приведен в листинге 1.

Листинг 1 – Текст файла prog1\_cpp

|  |
| --- |
| #include <windows.h>  #include <iostream>  #include <string>  int main(int argc, char \*argv[])  {  setlocale(LC\_ALL, "rus");  std::string path = " ";  std::cout << "Введите путь к файлу: ";  std::cin >> path;  SHELLEXECUTEINFO ShExecInfo = { 0 };  ShExecInfo.cbSize = sizeof(SHELLEXECUTEINFO);  ShExecInfo.fMask = SEE\_MASK\_NOCLOSEPROCESS;  ShExecInfo.hwnd = NULL;  ShExecInfo.lpVerb = NULL;  ShExecInfo.lpFile = path.c\_str();  ShExecInfo.lpParameters = "";  ShExecInfo.lpDirectory = NULL;  ShExecInfo.nShow = SW\_SHOW;  ShExecInfo.hInstApp = NULL;  ShellExecuteEx(&ShExecInfo);  std::cout << "Ожидание завершения процесса...\n";  WaitForSingleObject(ShExecInfo.hProcess, INFINITE);  std::cout << "Процесс завершен\n";  system("pause");  return 0;  } |

Результат работы программы показан на рисунках 1,2.

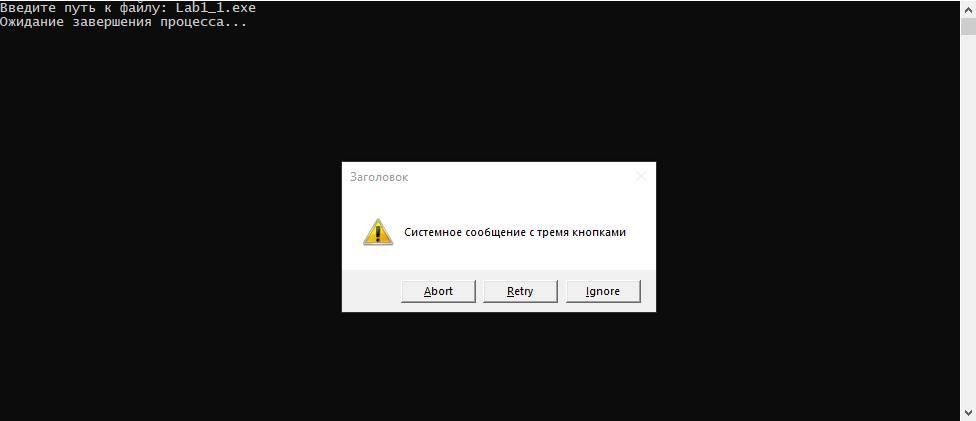


Рисунок 1 – Работа программы

Программа завершится после выполнения того, Что мы передали в качестве параметра (рис 2).

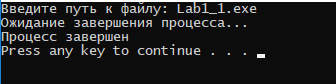


Рисунок 2 – Завершение программы

**Список использованных источников**

1. Саймон, Р. Windows 2000 API Энцеклопедия программиста / Р. Саймон. - М: DiaSoft, 2002