БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий   
и робототехники

Кафедра «Программное обеспечение вычислительной   
техники и автоматизированных систем»

О Т Ч Е Т

по лабораторной работе № 8

Дисциплина

«Системное программирование»

Тема

# «Управление потоками в Windows»

Проверил: Разорёнов Н.А.

Минск 2020

# Управление потоками в Windows

**Цель работы:** изучить основы создания и управления потоками в ОС Windows.

**Изучаемые вопросы**

1. Виды потоков, состояния потока.
2. Структура CONTEXT.
3. Создание потока.
4. Относительный приоритет потока.
5. Потоковая функция.
6. Функции WinAPI для управления потоками.
7. Окончание потока.
8. Время выполнения потока.

**Постановка задачи**

Разработать многопоточное Win32-приложение, которое использует диалоговое окно для управления потоками процессов. Дизайн диалогового окна задается вариантом, преподавателем или самостоятельно. Для визуализации работы потоков использовать соответствующие элементы управления диалога, графику. Приложение должно содержать три потока. Предусмотреть вывод системной информации о потоках (например полей CONTEXT , временны параметры и т.д.). В отчете привести диаграмму состояния потоков, копии окон.

**Ход выполнения работы**

1. Виды потоков, состояния потока.

Поток – последовательность команд, обрабатываемых CPU. В рамках одного процесса может находиться один или несколько потоков. Процесс предоставляет ресурсы, поток – команды и данные для обработки. Процесс содержащий один поток называется однопоточным, в противном случае – многопоточным.

Многопоточная модель охватывает 2 категории потоков и их комбинацию:

* потоки на уровне пользователя ULT (User Level Thread);
* потоки на уровне ядра KLT (Kernel Level Thread);
* комбинированная модель UKLT.

ULT управляются самим приложением. KLT управляется самим ядром через интерфейс прикладного программирования средств ядра ОС.

1. **Структура CONTEXT.**

В структуре CONTEXT хранятся данные о состоянии регистров с учетом специфики конкретного процессора. Она используется системой для выполнения различных внутренних операций.

Получить сведения о текущем состоянии регистров процессора. можно с помощью функции:

BOOL GetThreadContext( HANDLE hThread, PCONTEXT pContext);

Например в программе:

CONTEXT ct;

if (SendMessage(GetDlgItem(hwnd, IDC\_RADIO1), BM\_GETCHECK, 0, 0))

{

GetThreadContext(hThread[0], &ct);

Offset += sprintf(String + Offset, "%s\r\n", "КОНТЕКСТ 1-ГО ПОТОКА");

}

else if (SendMessage(GetDlgItem(hwnd, IDC\_RADIO2), BM\_GETCHECK, 0, 0))

{

GetThreadContext(hThread[1], &ct);

Offset += sprintf(String + Offset, "%s\r\n", "КОНТЕКСТ 2-ГО ПОТОКА");

}

else

{

GetThreadContext(hThread[2], &ct);

Offset += sprintf(String + Offset, "%s\r\n", "КОНТЕКСТ 3-ГО ПОТОКА");

}

1. **Создание потока.**

Для создания потока используется функция CreateThread().

Ниже приведен код из программы, демонстрирующий создание 3 потоков в приостановленном виде:

static HANDLE hThread[3];

for (int i = 0; i < 3; i++)

{

hThread[i] = CreateThread(NULL, 0,

(i == 0) ? TrackBarThreadOne :

(i == 1) ? TrackBarThreadTwo :

TrackBarThreadThree,

NULL, CREATE\_SUSPENDED, &dTs[i]

);

}

1. **Относительный приоритет потока.**

При создании потока функцией ***CreateThread***() существует возможность задать относительный приоритет потока. Если не указывать данный флаг, то по умолчанию потоку присвоится нормальный приоритет. Возможные приоритеты перечислены в таблице 1.1. Также задать приоритет потока можно с помощью функции ***SetThreadPriority***.

Таблица 1.1

Относительные приоритеты потоков

|  |  |
| --- | --- |
| Приоритет | Флаговый идентификатор |
| Below normal (ниже обычного) | THREAD\_PRIORITY\_BELOW\_NORMAL |
| Normal (обычный) | THREAD\_PRIORITY\_NORMAL |
| Above normal (выше обычного) | THREAD\_PRIORITY\_ABOVE\_NORMAL |
| Highest (высокий) | THREAD\_PRIORITY\_HIGHEST |
| Realtime (реального времени) | THREAD\_PRIORITY\_TIME\_CRITICAL |
| Lowest (низкий) | THREAD\_PRIORITY\_LOWEST |
| Idle (простаивающий) | THREAD\_PRIORITY\_IDLE |

comboPosition = SendMessage(GetDlgItem(hwnd, IDC\_COMBO1), CB\_GETCURSEL, 0, 0);

if (SendMessage(GetDlgItem(hwnd, IDC\_RADIO1), BM\_GETCHECK, 0, 0))

setPriority(hThread[0], comboPosition);

else if (SendMessage(GetDlgItem(hwnd, IDC\_RADIO2), BM\_GETCHECK, 0, 0))

setPriority(hThread[1], comboPosition);

else

setPriority(hThread[2], comboPosition);

1. **Потоковая функция.**

Функция потока принимает указатель на структуру с параметрами типа LPVOIDи возвращает значение типа DWORD .

Функция потока может выполнять любые задачи. Рано или поздно она закончит свою работу и вернет управление. В этот момент поток остановится, память, отведенная под его стек, будет освобождена, а счетчик пользователей его объекта ядра "поток" уменьшится на 1. Когда счетчик обнулится, этот объект ядра будет разрушен. Но, как и объект ядра "процесс", он может жить гораздо дольше, чем сопоставленный с ним поток.

Пример потоковой функции приведен ниже:

DWORD WINAPI TrackBarThreadOne(PVOID pvParam)

{

int count = 0;

for ( ; ; )

{

for (int i = 0; i < 10000000; i++)

{

}

count++;

if (count == 100) count = 0;

char buffer[20];

SendMessage(GetDlgItem(hwndMain, IDC\_PROGRESS1), TBM\_SETPOS, (WPARAM)TRUE, (LPARAM)count);

sprintf\_s(buffer, 20, "%d", count);

SetWindowText(GetDlgItem(hwndMain, IDC\_EDIT1), buffer);

}

return 0;

}

1. **Функции WinAPI для управления потоками.**

Создание потока:

hThread3 = CreateThread( NULL, 0,TrackBarThread3, NULL,CREATE\_SUSPENDED,NULL);

Назначение потоку относительного приоритета:

SetThreadPriority(hThread1,THREAD\_PRIORITY\_BELOW\_NORMAL);

Возобновление потока:

ResumeThread(hThread1);

Приостановка потока:

SuspendThread(hThread1);

Получение времени создания, окончания потока и др.:

GetThreadTimes(hThread1,&pftCreationTime,&pftExitTime,&pftKernelTime,&pftUserTime);

Ожидание завершения множества потоков:

WaitForMultipleObjects(3, hThread, TRUE, INFINITE);

Получение контекста потока:

GetThreadContext(hThread1, &ct);

Завершение потока:

ExitThread(0);

Завершение потока извне:

TerminateThread(hThread1,0);

1. **Окончание потока.**

Поток можно завершить способами:

* функция потока возвращает управление (рекомендуемый способ);
* поток самоуничтожается вызовом функции ***ExitThread*** (нежелательный способ);
* один из потоков данного или стороннего процесса вызывает функцию ***TerminateThread*** (нежелательный способ);
* завершается процесс, содержащий данный поток (тоже нежелательно);
* какой-либо поток процесса вызывает функцию [***ExitProcess***](file:///D:\STUDY\6%20сем\СПО\Other\Лабы\Reference_API\Functions\Fn_E\fn_exitprocess.htm)();
* какой-либо поток вызывает функцию [***TerminateThread***](file:///D:\STUDY\6%20сем\СПО\Other\Лабы\Reference_API\Functions\Fn_T\fn_terminatethread.htm)() с дескриптором потока;
* какой-либо поток вызывает функцию [***TerminateProcess***](file:///D:\STUDY\6%20сем\СПО\Other\Лабы\Reference_API\Functions\Fn_T\fn_terminateprocess.htm)() с дескриптором процесса.

В программе поток завершается с помощью функции:

TerminateThread(hThread[i], 0);

1. **Время выполнения потока.**

Временные показатели работы потока определяются функцией ***GetThreadTimes***, которая возвращает четыре временных параметра: Время создания (creation time), Время завершения (exit time), Время выполнения ядра (kernel time), Время выполнения User (User time).

В программе для определения времени выпонения потоков используется код:

FileTimeToSystemTime(&pftCreationTime, &creattime);

SystemTimeToTzSpecificLocalTime(NULL, &creattime, &stLocal);

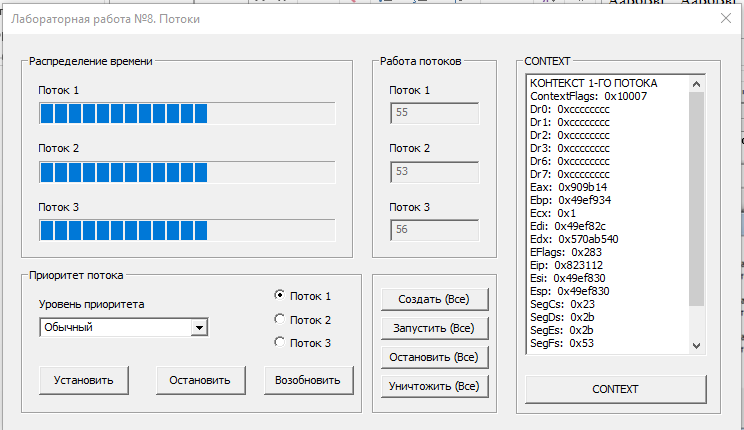
j += wsprintf(buf + j, "Время создания %d-го потока: %02d:%02d:%02d\r\n", i + 1, stLocal.wHour, stLocal.wMinute, stLocal.wSecond);

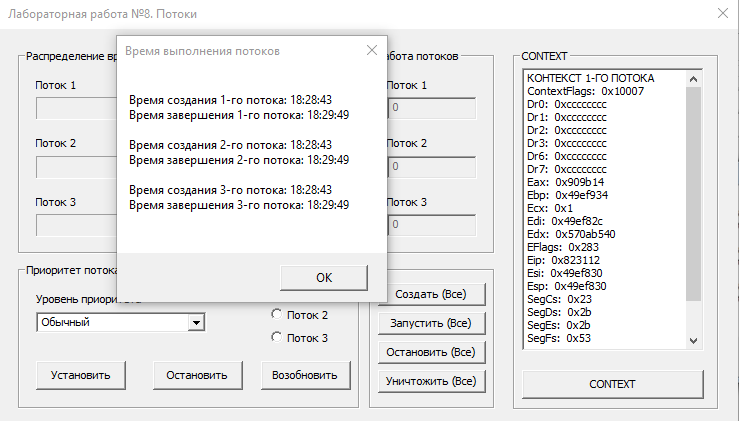
FileTimeToSystemTime(&pftExitTime, &exittime);

SystemTimeToTzSpecificLocalTime(NULL, &exittime, &stLocal1);

j += wsprintf(buf + j, "Время завершения %d-го потока: %02d:%02d:%02d\r\n\n", i + 1, stLocal1.wHour, stLocal1.wMinute, stLocal1.wSecond);

Результаты работы программного обеспечения приведены на рисунках 1 и 2:

Рисунок 1 - Запуск процессов и отображение системной информации

Рисунок 2 - Завершение выполнения потоков и вывод времени выпонения

**Выводы**

1. Функция потока может выполнять любые задачи. Рано или поздно она закончит свою работу и вернет управление. В этот момент поток остановится, память, отведенная под его стек, будет освобождена, а счетчик пользователей его объекта ядра "поток" уменьшится на 1. Когда счетчик обнулится, этот объект ядра будет разрушен. Но, как и объект ядра "процесс", он может жить гораздо дольше, чем сопоставленный с ним поток.
2. Поток – последовательность команд, обрабатываемых CPU. В рамках одного процесса может находиться один или несколько потоков.
3. В структуре CONTEXT хранятся данные о состоянии регистров с учетом специфики конкретного процессора. Она используется системой для выполнения различных внутренних операций.