Практична робота № 8

Потоки в ОС Windows

Мета: Розробити програму мовою Asm для Win32, яка керує потоками за допомогою функції CreateThread.

Теоретичні відомості.

Потік – це ланцюг інструкцій. Завжди є можливість також створювати додаткові потоки у вашій програмі. Вважається, що мультипотоковість – це багатозадачність всередині однієї програми.

Потік – це функція, яка виконується паралельно з основною програмою. Можливо запускати декілька екземплярів однієї і тієї ж функції, або можливо запускати декілька функцій одночасно в залежності від вимог програміста. Мультипотоковість властива для Win 32. Потоки виконуються у тому ж процесі, тому вони мають доступ до всіх ресурсів процесу, але кожний потік має свій власний стек, тому локальні змінні в кожному потоці приватні. Кожний потік має свій власний набір регістрів, тому коли Windows перемикається на інший потік, попередній, запам'ятовує свій стан і може відновити його коли основа отримує контроль. Це забезпечується внутрішніми засобами Windows.

Потоки поділяються на дві категорії:

потік інтерфейсу користувача: потік такого типу створює своє власне вікно, тому він отримує двійкові сповіщення. Він може відповідати користувачу за допомогою свого вікна.

робочі потоки: цей тип потоків не створює вікна тому він не може приймати будь-які Windows-сповіщення та існує тільки для того, щоб виконувати призначену йому роботу на задньому фоні відповідно до його призначення.

Найкраща стратегія при використанні мультипотокових властивостей Win 32- це дозволити основному потоку робити все, що пов'язане з користувацьким інтерфейсом, а іншим виконувати роботу у фоновому режимі. В цьому випадку основний потік – головний, інші потоки – його помічники.

Ми можемо створити потік за допомогою виклику функції *CreateThread*, яка має наступний синтаксис:

```
    CreateThread proto LpThreadAttributes:DWORD,\
    dwStackSize: DWORD,\ LpStartAddress:
DWORD,\LpParameter: DWORD,\dwCreationFlags: DWORD,\
LpThreadid: DWORD
```

Функція *CreateThread* схожа на *CreateProcess*.

lpThreadAttributes - можливо використовувати NULL, якщо необхідно, щоб у потоку були установки безпеки за замовчуванням.

dwStackSize – вказати розмір стека потоку. Якщо необхідно, щоб потік мав такий же розмір стека, як і у основного, використовують NULL в якості параметру.

lpStartAddress – адреса функції потоку. Ця функція буде виконувати призначену для потоку роботу. Ця функція має отримувати один і тільки один 32-бітний параметр і повертати 32-бітне значення.

lpParametr – параметр, який необхідно передати функції потоку.

dwCreationFlags – 0 означає, що потік починає виконуватися відразу після створення. Для зворотнього можна використовувати прапор *CREATE_SUSPEND*.

lpThreadId – CreateThread – помістити сюди *ID* створеного потоку.

Якщо виклик *CreateThread* пройшов успішно, вона повертає хэндл створеного потоку, в зворотньому випадку вона повертає *NULL*.

Функція потоку запускається так швидко, як тільки закінчується виклик *CreateThread*, якщо тільки не позначено прапор *CREATE_SUSPEND*. В цьому випадку потік буде заморожений до виклику функції *ResumThread*.

Коли функція потоку повертається (за допомогою інструкції *ret*) Windows опосередковано викликає *ExitThreat* для функції потоку.

Можливо отримати код виходу потоку за допомогою функції *GetExitCodeThread*.

Існує три методи комунікації між потоками:

Використання глобальних змінних.

Windows повідомлення.

Події.

Потоки розглядають ресурси процесу, включаючи глобальні змінні, тому потоки можуть використовувати їх для того, щоб взаємодіяти один з одним. Але все таки, цей метод має використовуватись обережно. Синхронізацію потрібно уважно спланувати.

Також можливо використовувати Windows повідомлення, щоб виконувати взаємодію між потоками.

Якщо всі потоки мають користувацький інтерфейс, то немає проблем: цей метод може використовуватися для двохсторонньої комунікації. Все що потрібно зробити – це визначити один або більше додаткових Windows повідомлень, які будуть використовуватися з потоками. Визначаємо повідомлення використовуючи значення

WM_USER як базове, наприклад так:

WM_MYCUSTOMMSG equ WM_USER+100h

Якщо один із потоків має інтерфейс користувача, а інший є робочим ми не можемо використовувати даний метод для двостороннього користування, так як робочого потоку немає свого вікна, а відповідно і черги повідомлень.

Можливо використовувати наступні схеми:

Потік з інтерфейсом користувача → глобальна змінна → робочий потік.

Робочий потік → Windows –повідомлення → Потік з користувацьким інтерфейсом.

Фактично ми будемо використовувати цей метод в нашому прикладі.

Хід виконання роботи

Створюємо в блокноті файл potok.asm.

```
386
.model flat,stdcall
option casemap:none

WinMain proto :DWORD,:DWORD,:DWORD,:DWORD

include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib

.const
IDM_CREATE_THREAD equ 1
IDM_EXIT equ 2
WM_FINISH equ WM_USER+100h
```

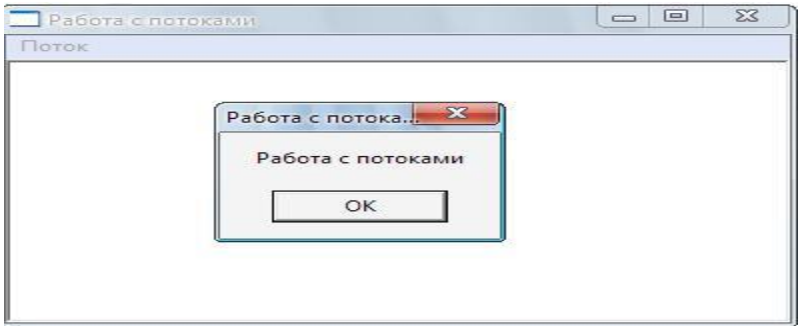
```
.data
……
……
……


ThreadProc PROC USES ecx
        Param:DWORD mov ecx,600000000
Loop1:
        add eax,eax
        dec ecx
        jz Get_out
        jmp Loop1
Get_out:
        invoke
        SendMessage,hwnd,WM_FINISH,NULL,NULL ret
ThreadProc ENDP
end start
```

Створюємо в блокноті файл potok.rc.

```
#define IDM_CREATE_THREAD 1
#define IDM_EXIT 2
FirstMenu MENU
{
POPUP "&Поток"
        {
        MENUITEM "&Створити поток",IDM_CREATE_THREAD
        MENUITEM SEPARATOR
       MENUITEM "&Вихід",IDM_EXIT
        }
}
```

Результат компіляції і запуску програми.



Контрольні питання

Що таке потік?

Назвіть елементи, що використовуються потоками одночасно?

Назвіть елементи, що використовуються потоками нарізно?

Які існують способи реалізації потоків?

Які переваги реалізації потоків в просторі користувача?

В яких ОС можлива реалізація паралельного виконання команд різних потоків?