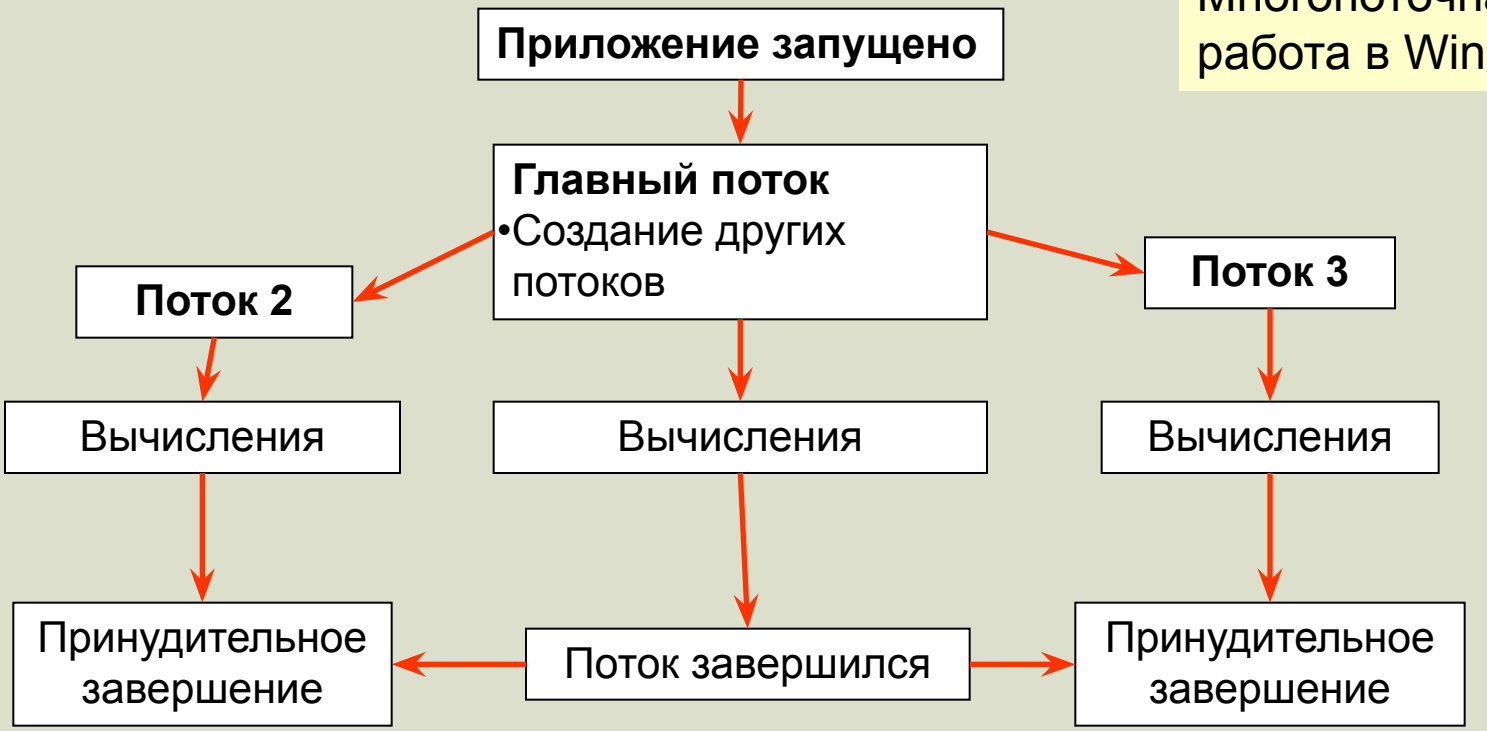
Системное ПО

Потоки

Системное ПО

Многопоточная работа в Windows



- При запуске приложения создается главный поток
- Любой поток может создавать другие потоки
- Потоки могут работать одновременно
- При завершении главного потока все остальные потоки принудительно завершаются

Системное ПО

Многопоточная работа в Windows

Функция создает новый поток

```
HANDLE WINAPI CreateThread(
  _In_opt_   LPSECURITY_ATTRIBUTES lpThreadAttributes,
  _In_       SIZE_T dwStackSize,
  _In_       LPTHREAD_START_ROUTINE lpStartAddress,
  _In_opt_   LPVOID lpParameter,
  _In_       DWORD dwCreationFlags,
  _Out_opt_  LPDWORD lpThreadId
);
```

lpStartAddress – функция, реализующая новый поток

lpParameter – адрес, который передается потоку

lpCreationFlags – флаги:

CREATE_SUSPENDED – поток создается остановленным

Системное ПО

Многопоточная работа в Windows

Функция, реализующая поток

```
DWORD WINAPI ThreadProc(
  _In_  LPVOID lpParameter
);
```

lpParameter – адрес, который был передан функции CreateThread

С его помощью можно передать потоку данные для обработки. Например можно создать для потока массив, структуру, экземпляр класса, и передать потоку соответствующий адрес.

Системное ПО

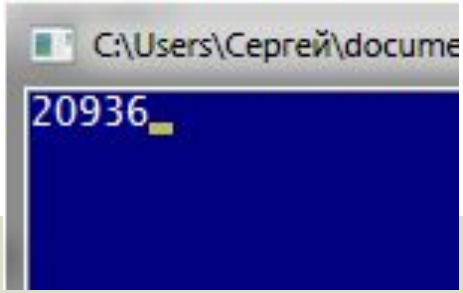
```
DWORD WINAPI MyThread(LPVOID Param)
{
    int i,j,a=0;
    for(i=0;i<100000;i++)
        for(j=0;j<100000;j++) a++;
    return 0;
}

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE hThread;
    DWORD Tick;
    Tick=GetTickCount();
    hThread=CreateThread(NULL,0,MyThread,NULL,0,NULL);
    WaitForSingleObject(hThread,INFINITE);
    Tick=GetTickCount()-Tick;
    printf("%d",Tick);
    _getch();
    return 0;
}
```

Пример

Создаем новый поток, реализуемый функцией **MyThread**

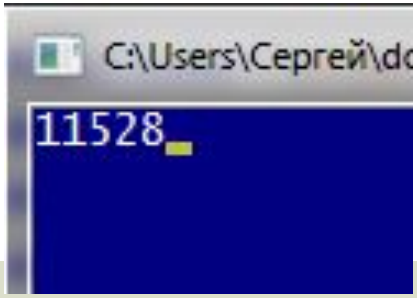
С помощью **GetTickCount** засекаем время работы потока

WaitForSingleObject ждет, пока поток завершится



Системное ПО

```
DWORD WINAPI MyThread(LPVOID Param)
{
    int i,j,a=0;
    for(i=0;i<50000;i++)
        for(j=0;j<100000;j++) a++;
    return 0;
}

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE hThread[2];
    DWORD Tick;
    Tick=GetTickCount();
    hThread[0]=CreateThread(NULL,0,MyThread,NULL,0,NULL);
    hThread[1]=CreateThread(NULL,0,MyThread,NULL,0,NULL);
    WaitForMultipleObjects(2,hThread,TRUE,INFINITE);
    Tick=GetTickCount()-Tick;
    printf("%d",Tick);
    _getch();
    return 0;
}
```



Пример

Запуск двух потоков

Одна и та же функция может реализовывать несколько потоков

Вычисления такой же сложности на двух ядрах выполняются быстрее

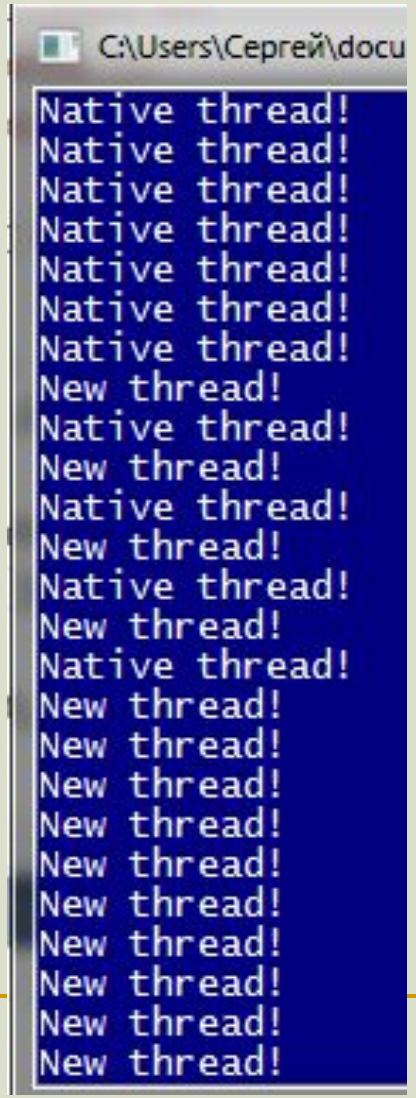
Системное ПО

Управление потоками

```
DWORD WINAPI MyThread(LPVOID lpParam)
{
    int i,j;
    for(i=1;i<15;i++)
    {
        printf("New thread!\n");
        for(j=0;j<100000;j++);
    }
    return 0;
}

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE hMyThread;
    hMyThread=CreateThread(NULL,0,MyThread,NULL,0,NULL);
    int i,j;
    for(i=0;i<100000;i++);
    for(i=1;i<15;i++)
    {
        printf("Native thread!\n");
        for(j=0;j<100000;j++);
    }
    _getch();
    return 0;
}
```

Выполняем одновременно два потока

Никакой упорядоченности при выполнении не гарантируется



Системное ПО

Управление потоками

```
DWORD WINAPI ResumeThread(
  _In_  HANDLE hThread
);
```

Уменьшает значение счетчика

```
DWORD WINAPI SuspendThread(
  _In_  HANDLE hThread
);
```

Увеличивает значение счетчика

Поток имеет счетчик (suspend count), определяющий состояние потока. Если значение счетчика >0, поток остановлен

Системное ПО

```
DWORD WINAPI MyThread(LPVOID lpParam)
{
    int i,j;
    for(i=0;i<10;i++)
    {
        printf("New thread!\n");
        for(j=0;j<100000;j++);
    }
    return 0;
}

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE hMyThread;
    hMyThread=CreateThread(NULL,0,MyThread,NULL,CREATE_SUSPENDED,NULL);
    int i,j;
    for(i=0;i<5;i++)
    {
        printf("Native thread!\n");
        for(j=0;j<100000;j++);
    }
    ResumeThread(hMyThread);
    system("pause");
    return 0;
}
```

Управление потоками

Поток создается приостановленным

«Размораживаем» поток с помощью **ResumeThread**