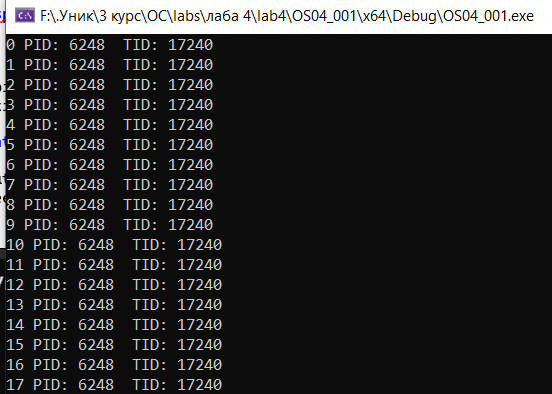
Лабораторная работа 04

OC

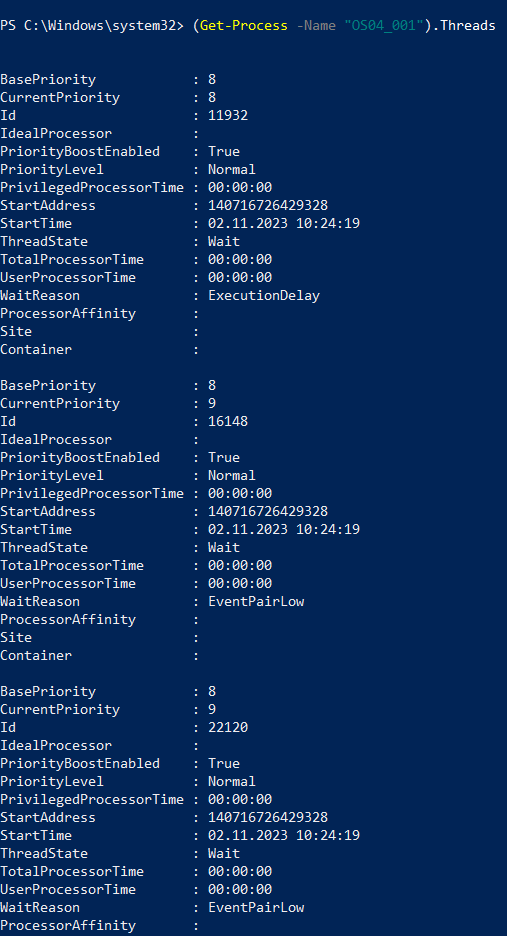
**Задание 01**

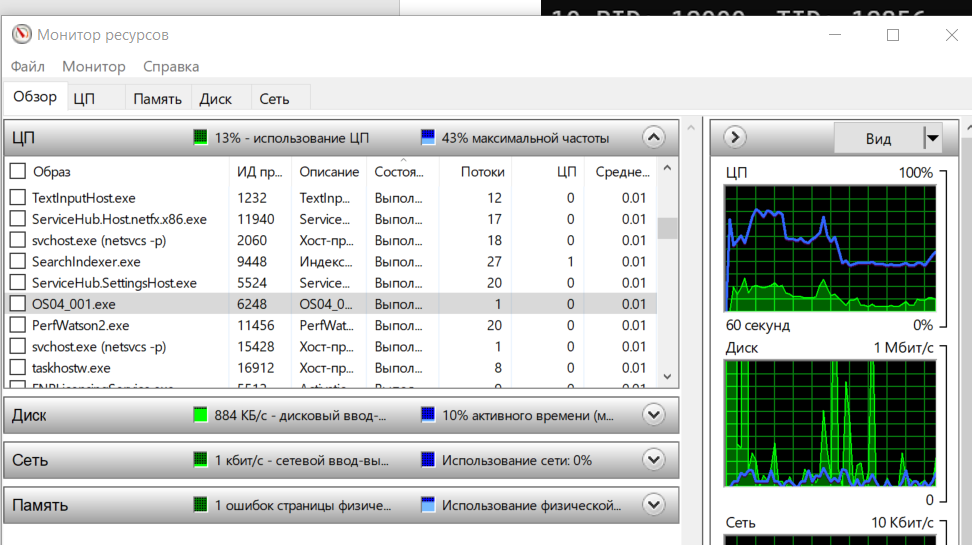
1. Разработайте консольное Windows-приложение **OS04\_01** на языке С++, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификаторов текущего процесса и текущего потока.

|  |
| --- |
| 1. #include <Windows.h> 2. #include <iostream> 3. using namespace std; 4. int main() 5. { 6. DWORD pid = GetCurrentProcessId(); 7. DWORD tid = GetCurrentThreadId(); 8. for (int i = 0; i < 1000; i++) 9. { 10. cout << i << " " << "PID: " << pid << " " << " TID: " << tid << " " << endl; 11. Sleep(1000); 12. } 13. } |



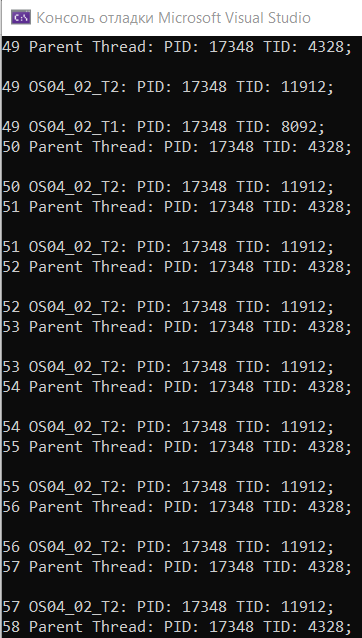
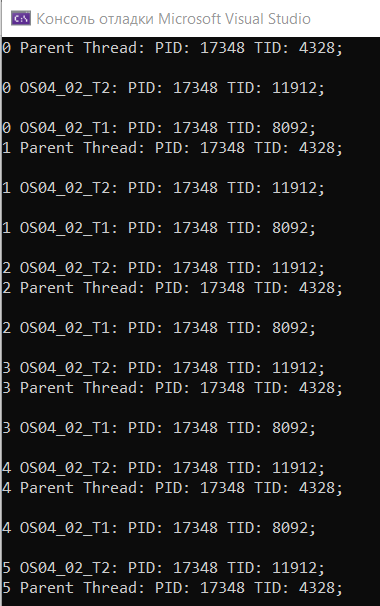
1. Продемонстрируйте информацию об потоках процесса **OS04\_01** с помощью утилит  **PowerShell ISE** и **Performance Monitor**.





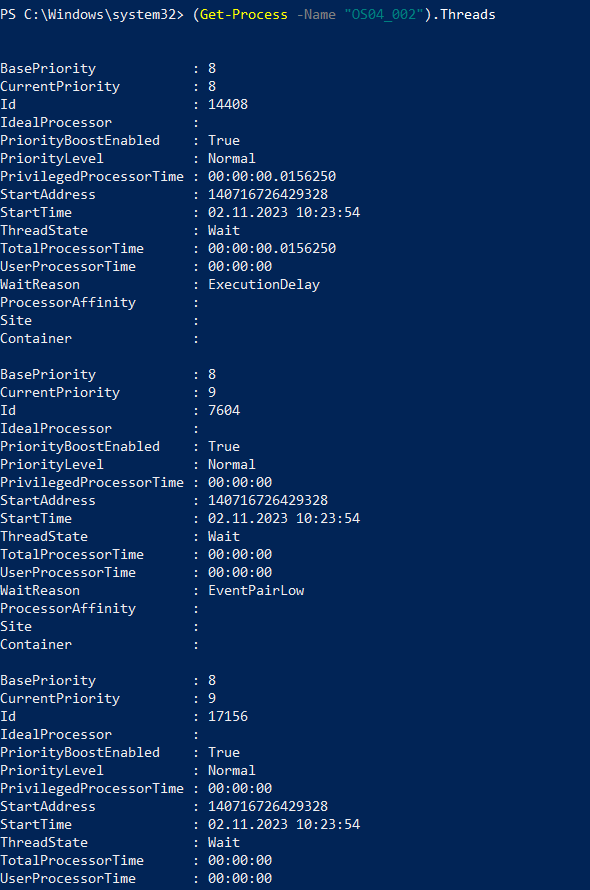
**Задание 02**

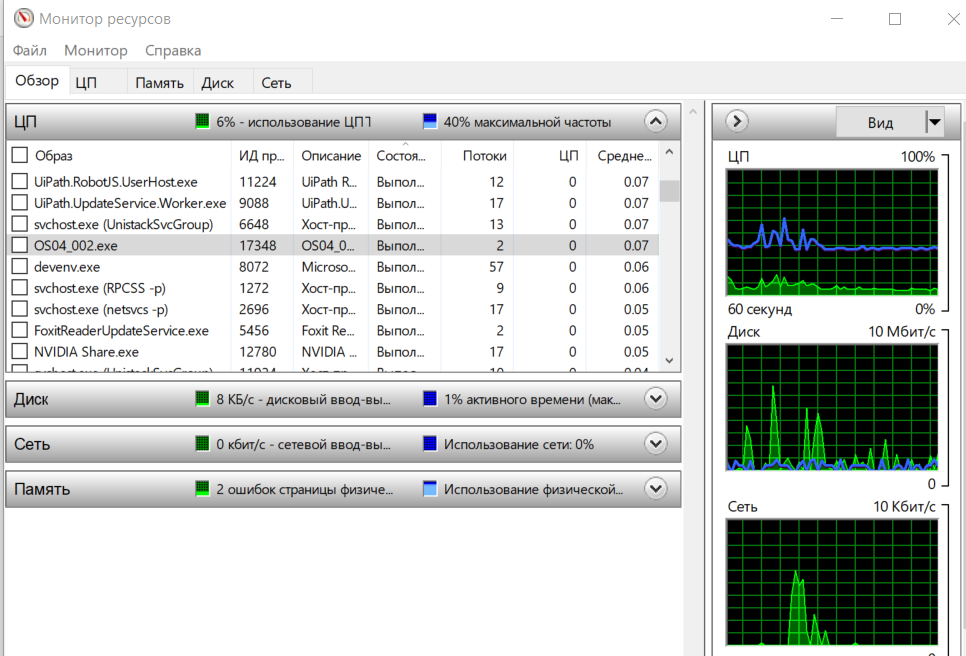
1. Разработайте на языке консольное Windows-приложение **OS04\_02** на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.



|  |
| --- |
| #include <Windows.h>  #include <iostream>  #include "os04\_02.h"  DWORD pid = NULL;  using namespace std;  DWORD WINAPI ChildThread() {  DWORD tid = GetCurrentThreadId();  for (int i = 0; i < 50; i++)  {  cout << i << " " << "OS04\_02\_T1: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  }  return 0;  }  DWORD WINAPI ChildSecondThread() {  DWORD tid = GetCurrentThreadId();  for (int i = 0; i < 125; i++)  {  cout << i << " " << "OS04\_02\_T2: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  }  return 0;  }  int main()  {  pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  DWORD childId = NULL;  DWORD childSecondId = NULL;  // SECURITY\_ATTRIBUTES  HANDLE hChild = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildThread, NULL, 0, &childId);  HANDLE hChildSecond = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildSecondThread, NULL, 0, &childSecondId);  for (int i = 0; i < 100; i++)  {  cout << i << " " << "Parent Thread: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  }  WaitForSingleObject(hChild, INFINITE);  CloseHandle(hChild);  } |

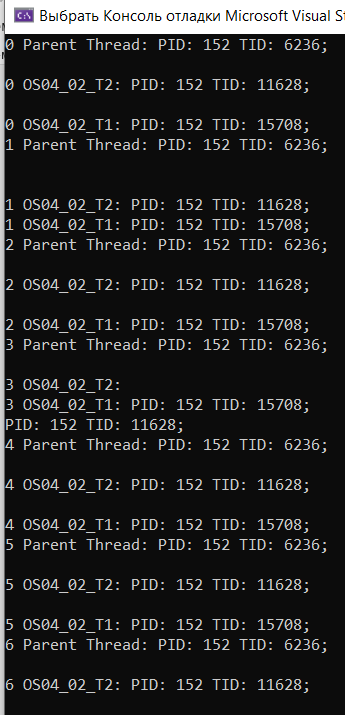
1. Процесс **OS04\_02** должен создать два потока: потоковые функции **OS04\_02\_T1, OS04\_02\_T2.**
2. Поток **OS04\_02\_T1** - выполняет цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.
3. Поток **OS04\_02\_T2** - выполняет цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.
4. Продемонстрируйте информацию об потоках процесса **OS04\_02** с помощью утилит  **PowerShell ISE** и **Performance Monitor**.





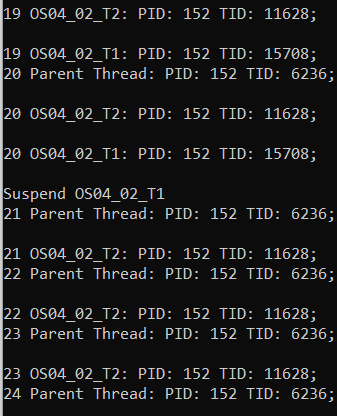
**Задание 03.**

1. Разработайте на языке консольное Windows-приложение **OS04\_03** на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

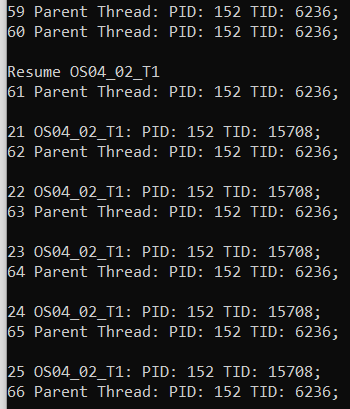
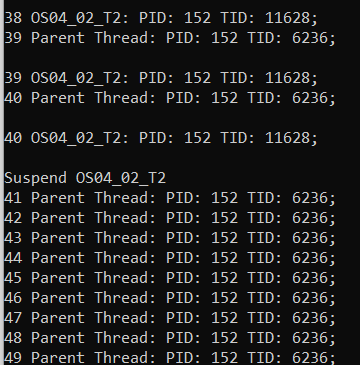


|  |
| --- |
| #include <Windows.h>  #include <iostream>  DWORD pid = NULL;  bool flag = false;  using namespace std;  DWORD WINAPI ChildThread() {  DWORD tid = GetCurrentThreadId();  for (int i = 0; i < 50; i++)  {  cout << i << " " << "OS04\_02\_T1: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  }  return 0;  }  DWORD WINAPI ChildSecondThread() {  DWORD tid = GetCurrentThreadId();  for (int i = 0; i < 125; i++)  {  cout << i << " " << "OS04\_02\_T2: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  if (i == 124) {  flag = true;  cout << "Wakeup Parent Thread \n";  exit(0);  }  }  return 0;  }  int main()  {  pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  DWORD childId = NULL;  DWORD childSecondId = NULL;  HANDLE hChild = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildThread, NULL, 0, &childId);  HANDLE hChildSecond = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildSecondThread, NULL, 0, &childSecondId);  for (int i = 0; i < 100; i++)  {  cout << i << " " << "Parent Thread: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  if (i == 20) {  SuspendThread(hChild);  cout << "\nSuspend OS04\_02\_T1 \n";  }  else if (i == 60) {  ResumeThread(hChild);  cout << "\nResume OS04\_02\_T1 \n";  }  if (i == 40) {  SuspendThread(hChildSecond);  cout << "\nSuspend OS04\_02\_T2 \n";  }  }  ResumeThread(hChildSecond);  cout << "\nResume OS04\_02\_T2 \n";  cout << "\nSuspend Parent Thread \n";  Sleep(10000000);  WaitForSingleObject(hChild, INFINITE);  CloseHandle(hChild);  } |

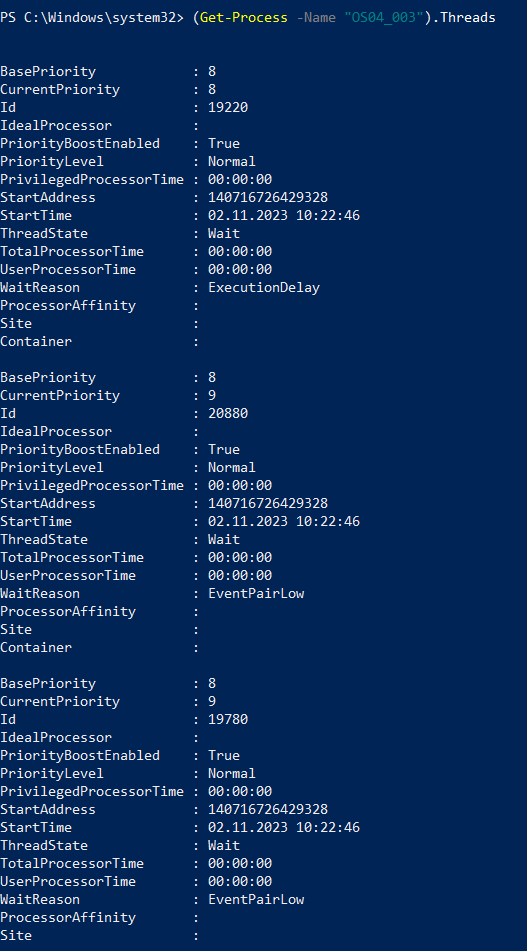
1. Процесс **OS04\_03** должен создать два потока: потоковые функции **OS04\_03\_T1, OS04\_03\_T2.**
2. Поток **OS04\_03\_T1** - выполняет цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.
3. Поток **OS04\_03\_T2** - выполняет цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.
4. Поток **main** приостанавливает работу потока **OS04\_03\_T1** на20й своей итерации и возобновляют на 60й своей итерации.

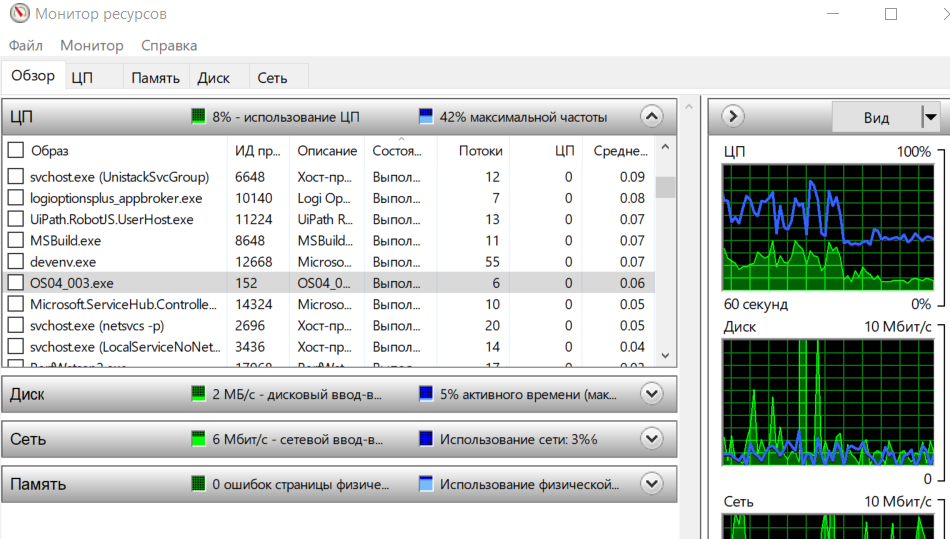


1. Поток **main** приостанавливает работу потока **OS04\_03\_T2** на40й своей итерации и возобновляют на 60 после окончания собственных итераций (по окончании собственного цикла).



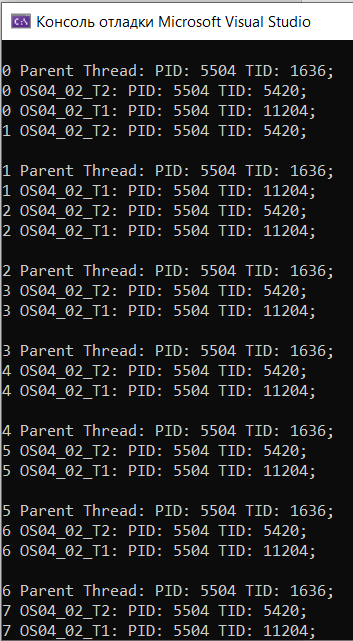
1. Продемонстрируйте информацию об потоках процесса **OS04\_03** с помощью утилит  **PowerShell ISE** и **Performance Monitor**.





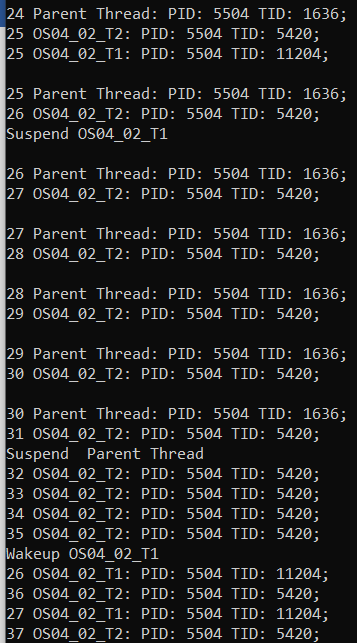
**Задание 04**

1. Разработайте на языке консольное Windows-приложение **OS04\_04** на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

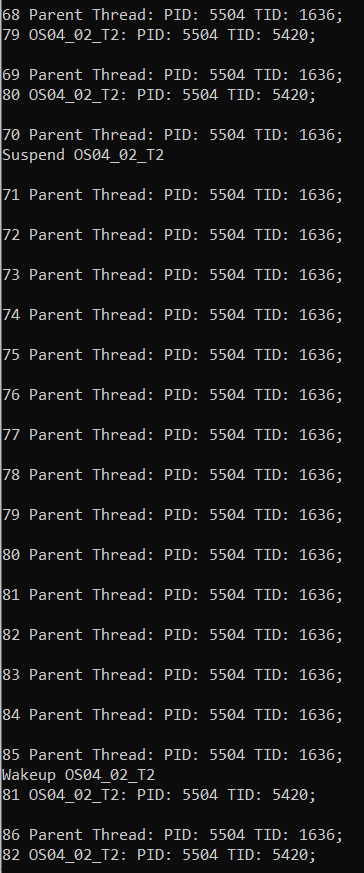


|  |
| --- |
| #include <Windows.h>  #include <iostream>  DWORD pid = NULL;  bool flag = false;  using namespace std;  DWORD WINAPI ChildThread() {  DWORD tid = GetCurrentThreadId();  for (int i = 0; i < 50; i++)  {  cout << i << " " << "OS04\_02\_T1: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  if (i == 25) {  cout << "Suspend OS04\_02\_T1 \n";  Sleep(10000);  cout << "Wakeup OS04\_02\_T1 \n";  }  }  return 0;  }  DWORD WINAPI ChildSecondThread() {  DWORD tid = GetCurrentThreadId();  for (int i = 0; i < 125; i++)  {  cout << i << " " << "OS04\_02\_T2: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  if (i == 80) {  cout << "Suspend OS04\_02\_T2 \n";  Sleep(15000);  cout << "Wakeup OS04\_02\_T2 \n";  }  }  return 0;  }  int main()  {  pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  DWORD childId = NULL;  DWORD childSecondId = NULL;  HANDLE hChild = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildThread, NULL, 0, &childId);  HANDLE hChildSecond = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildSecondThread, NULL, 0, &childSecondId);  for (int i = 0; i < 100; i++)  {  cout << i << " " << "Parent Thread: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  if (i == 30) {  cout << "Suspend Parent Thread\n";  Sleep(10000);  cout << "Wakeup Parent Thread\n";  }  }  WaitForSingleObject(hChild, INFINITE);  CloseHandle(hChild);  } |

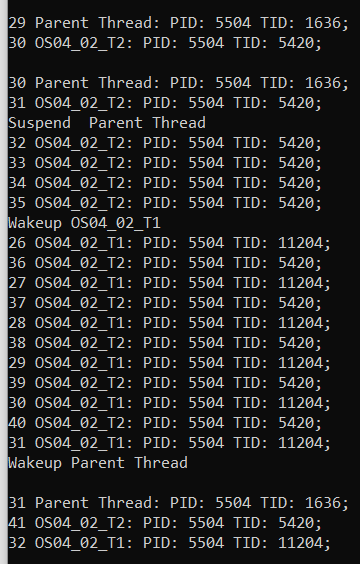
1. Процесс **OS04\_04** должен создать два потока: потоковые функции **OS04\_04\_T1, OS04\_04\_T2.**
2. Поток **OS04\_04\_T1** - выполняет цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока. Поток засыпает на 25й собственный итерации на 10сек.



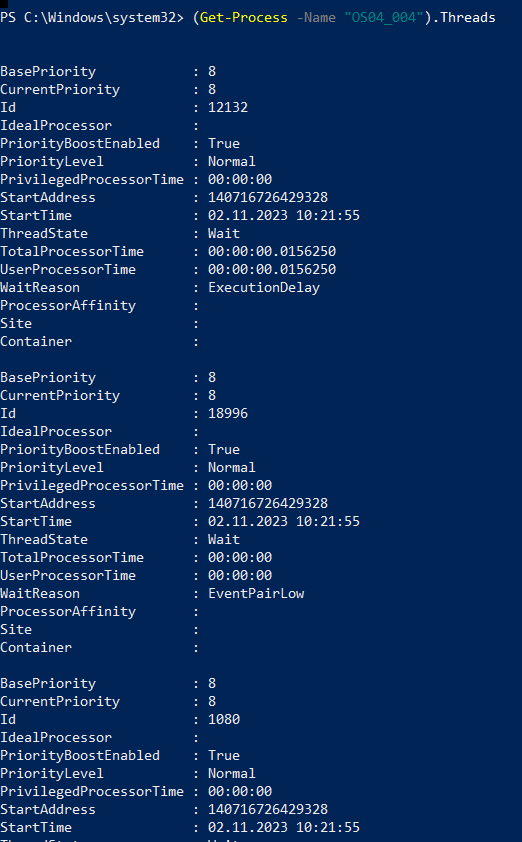
1. Поток **OS04\_04\_T2** - выполняет цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока. Поток засыпает на 80й собственной итерации на 15сек.

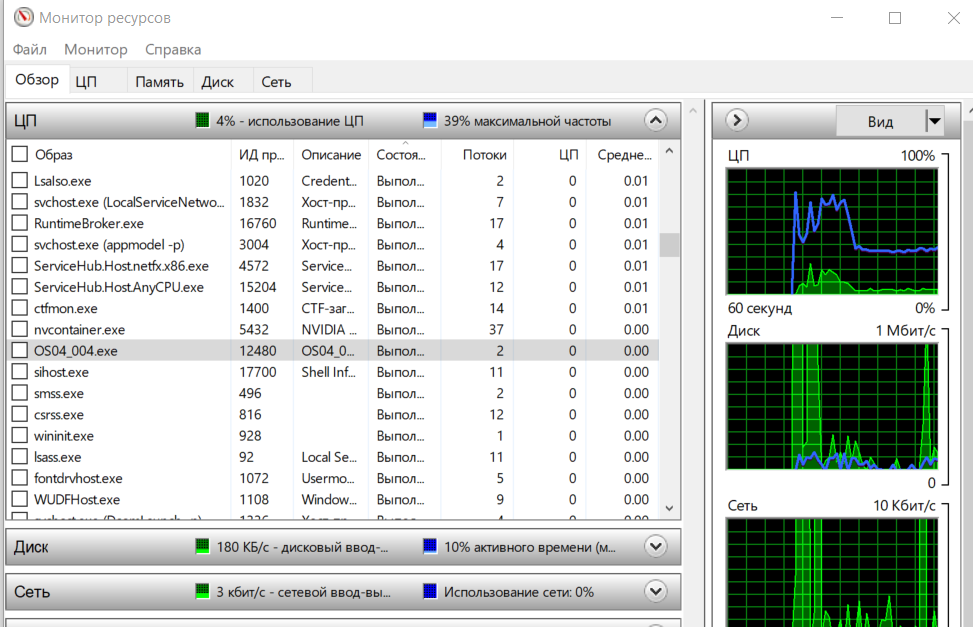


1. Поток **main** засыпает на 30й итерации на 10 сек.



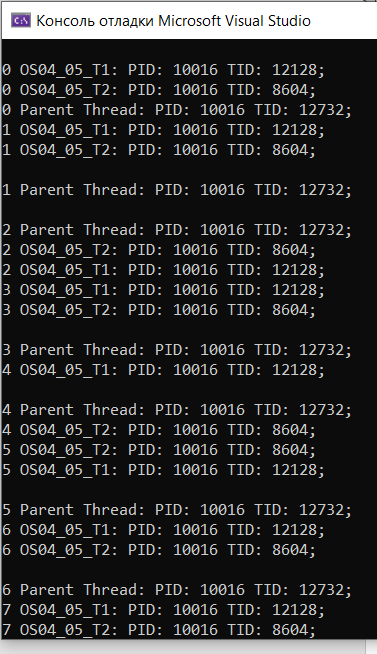
1. Продемонстрируйте информацию об потоках процесса **OS04\_04** с помощью утилит  **PowerShell ISE** и **Performance Monitor**.





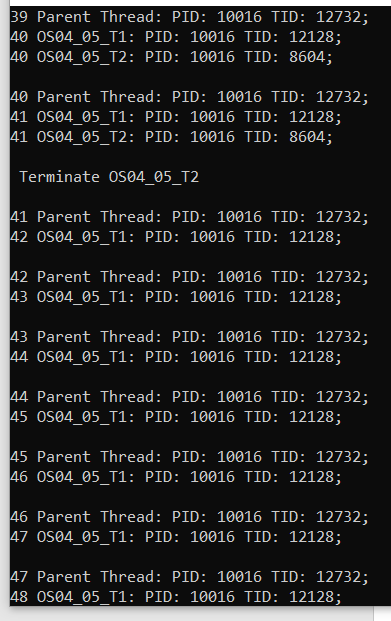
**Задание 05**

1. Разработайте на языке консольное Windows-приложение **OS04\_05** на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

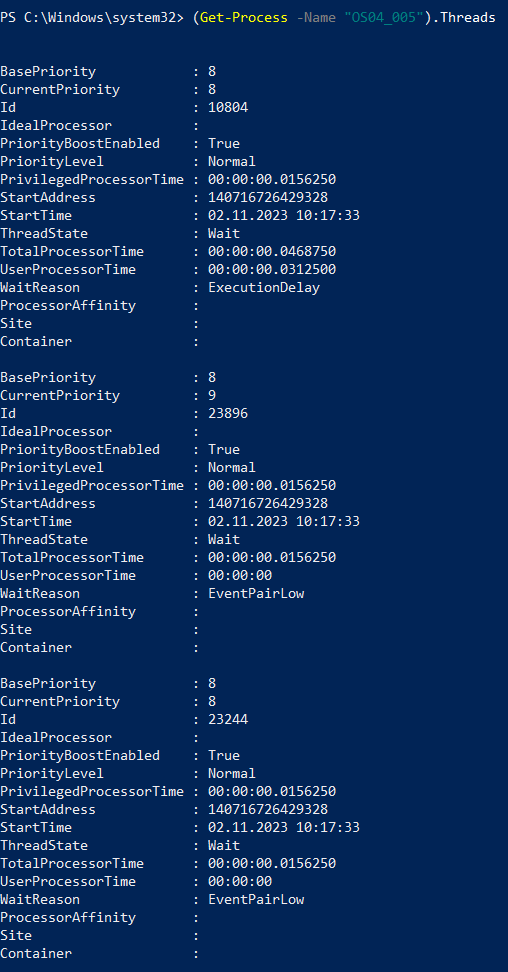


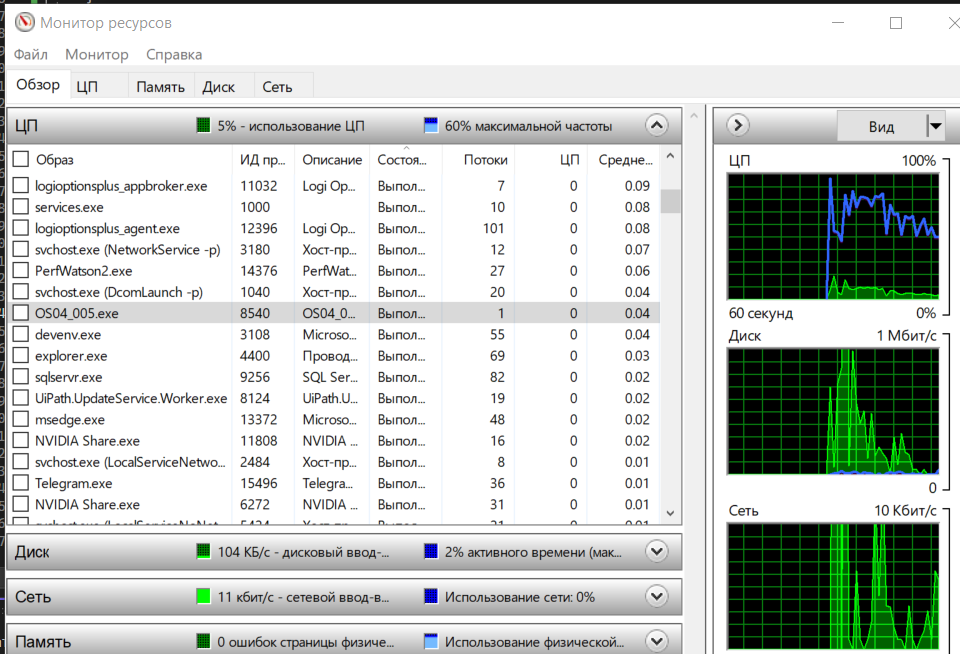
|  |
| --- |
| #include <Windows.h>  #include <iostream>  DWORD pid = NULL;  using namespace std;  DWORD WINAPI ChildThread() {  DWORD tid = GetCurrentThreadId();  for (int i = 0; i < 50; i++)  {  cout << i << " " << "OS04\_05\_T1: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  }  return 0;  }  DWORD WINAPI ChildSecondThread() {  DWORD tid = GetCurrentThreadId();  for (int i = 0; i < 125; i++)  {  cout << i << " " << "OS04\_05\_T2: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  }  return 0;  }  int main()  {  pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  DWORD childId = NULL;  DWORD childSecondId = NULL;  HANDLE hChild = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildThread, NULL, 0, &childId);  HANDLE hChildSecond = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildSecondThread, NULL, 0, &childSecondId);  for (int i = 0; i < 100; i++)  {  cout << i << " " << "Parent Thread: " << "PID: " << pid << " " << "TID: " << tid << ";" << endl;  Sleep(1000);  if (i == 40) {  cout << "\n Terminate OS04\_05\_T2 \n";  TerminateThread(hChildSecond, -1);  }  }  WaitForSingleObject(hChild, INFINITE);  CloseHandle(hChild);  } |

1. Процесс **OS04\_05** должен создать два потока: потоковые функции **OS04\_05\_T1, OS04\_05\_T2.**
2. Поток **OS04\_05\_T1** - выполняет цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.
3. Поток **OS04\_05\_T2** - выполняет цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.
4. Поток **main** завершаетпоток **OS04\_05\_T2** на 40й собственной итерации.



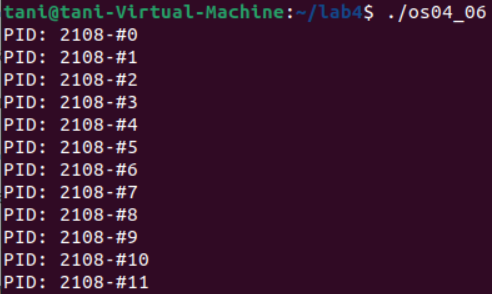
1. Продемонстрируйте информацию об потоках процесса **OS04\_05** с помощью утилит  **PowerShell ISE** и **Performance Monitor**.

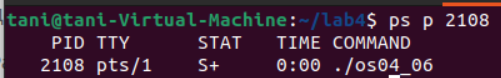




**Задание 06**

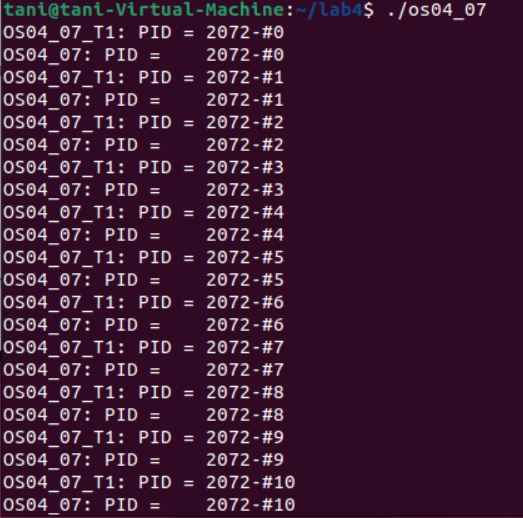
1. Разработайте на языке консольное Linux-приложение **OS04\_06** на языке С, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификатора процесса.
2. Продемонстрируйте информацию о потоках процесса **OS04\_06** с помощью утилиты **ps**.

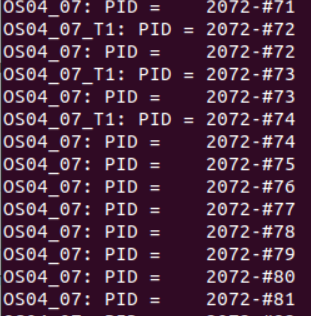


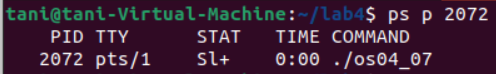


**Задание 07**

1. Разработайте на языке консольное Linux-приложение **OS04\_07** на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
2. Процесс **OS04\_07** должен создать поток: потоковая функция **OS04\_07\_T1.**
3. Поток **OS04\_07\_T1** - выполняет цикл 75 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса.
4. Продемонстрируйте информацию о потоках процесса **OS04\_07** с помощью утилиты **ps**.

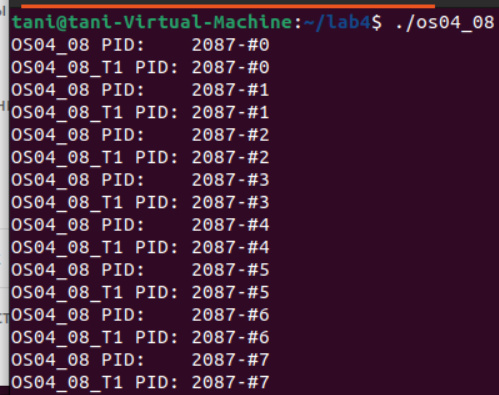


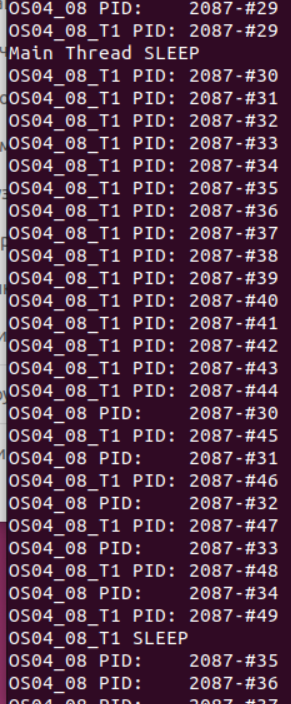


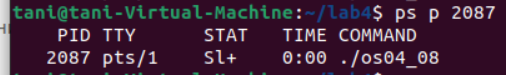


**Задание 08**

1. Разработайте на языке консольное Linux-приложение **OS04\_08** на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
2. Процесс **OS04\_08** должен создать поток: потоковая функция **OS04\_08\_T1.**
3. Поток **OS04\_08\_T1** - выполняет цикл 75 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса. Поток должен засыпать на 10сек. на 50й собственной итерации.
4. Поток **main** должен засыпать на 15 сек. на 30й собственной итерации.
5. Продемонстрируйте информацию о потоках процесса **OS04\_08** с помощью утилиты **ps**.







**Задание 09.ответьте на следующие вопросы**

1. Что такое поток управления OS?
2. С помощью каких системных вызовов создаются потоки в Windows и Linux?
3. Что такое системные и пользовательские потоки?
4. Что такое многопоточность?
5. Чем отличаются приоритетная многопоточность от и кооперативной многопоточности?
6. Что такое диспетчеризация потоков управления OS?
7. Что такое контекст потока и для чего он нужен?
8. Перечислите состояния в которых может быть поток и поясните их назначение.
9. Что такое LWP?
10. Что такое потокобезопасность программного кода?
11. Что такое реентерабельность кода?
12. Что такое Fiber?
13. Дайте развернутое определение потока OS.

**1.Что такое поток управления OS?**

поток (управления) OS – объект ядра операционной системы, которому OS выделяет процессорное время. Наименьшая единица работы ядра OS. LWP ( light – weight process)

последовательность инструкций, выполняемых процессором в выделенные OS интервалы времени

**2. С помощью каких системных вызовов создаются потоки в Windows и Linux?**

CreateThread(Windows)

Pthread\_create(linux)

**3. Что такое системные и пользовательские потоки?**

Системный поток: выполняют различные сервисы операционной системы, запускаются ядром операционной системы

Пользовательские потоки: служат для решения пользовательских задач, запускаются приложением

**4. Что такое многопоточность?**

многопоточность – модель (парадигма) программирования (OS, системы программирования, программы) поддерживать потоки управления. Приоритетная и кооперативная многопоточность.

**5. Чем отличаются приоритетная многопоточность от кооперативной многопоточности?**

КООПЕРАТИВНАЯ

Тип многозадачности, при котором следующая задача выполняется только после того, как текущая задача явно объявит себя готовой отдать процессорное время другим задачам.

Преимущества:

- отсутствие необходимости защищать все разделяемые структуры данных объектами типа критических секций и мьютексов, что упрощает программирование, особенно перенос кода из однозадачных сред в многозадачные

Недостатки:

- неспособность всех приложений работать в случае ошибки в одном из них, приводящей к отсутствию вызова операции «отдать процессорное время».

- затрудненная возможность реализации многозадачной архитектуры ввода-вывода в ядре ОС, позволяющей процессору исполнять одну задачу в то время, как другая задача инициировала операцию ввода-вывода и ждет её завершения.

ПРИОРИТЕТНАЯ:

Вид многозадачности, в котором операционная система сама передает управление от одной выполняемой программы другой в случае завершения операций ввода-вывода, возникновения событий в аппаратуре компьютера, истечения таймеров и квантов времени, или же поступлений тех или иных сигналов от одной программы к другой. Процессор может быть переключен с исполнения одной программы на исполнение другой без всякого пожелания первой программы и буквально между любыми двумя инструкциями в её коде.

Преимущества:

- возможность полной реализации многозадачного ввода-вывода в ядре ОС, когда ожидание завершения ввода-вывода одной программой позволяет процессору тем временем исполнять другую программу;

- сильное повышение надежности системы в целом, в сочетании с использованием защиты памяти

- возможность полного использования многопроцессорных и многоядерных систем.

Недостатки:

необходимость особой дисциплины при написании кода, особые требования к его реентерабельности, к защите всех разделяемых и глобальных данных объектами типа критических секций и мьютексов.

**6. Что такое диспетчеризация потоков управления OS?**

Диспетчеризация заключается в реализации найденного в результате планирования (динамического или статистического) решения, то есть в переключении процессора с одного потока на другой

Диспетчеризация сводится к следующему:

-сохранение контекста текущего потока, который требуется сменить;

-загрузка контекста нового потока, выбранного в результате планирования;

-запуск нового потока на выполнение

**7. Что такое контекст потока и для чего он нужен?**

контекст потока: данные, которые сохраняются при переключении процессов и предназначенные для продолжения работы

- программный код;

- набор регистров;

- стек памяти;

- стек ядра операционной системы;

- маркер доступа.

**8. Перечислите состояния в которых может быть поток и поясните их назначение.**

-модель пяти состояний: interrupt – прерывание по окончанию кванта, block – заблокировать до наступления события, unblock – ожидаемое событие наступило.

-модель семи состояний, Suspend – приостановить поток, Resume – возобновить поток.

-модель 12 состояний, sleep – остановить поток на заданное время, wakeup – возобновить работу.

**9. Что такое LWP?**

Легковесный процесс — является средством достижения многозадачности в компьютерной операционной системе, поддерживающий работу потока пространства пользователя. Каждый поток пространства пользователя неразрывно связан с легковесным процессом. Процедура создания легковесного процесса отличается от процедуры создания обычного процесса;

**10. Что такое потокобезопасность программного кода?**

свойство программного кода (программы) корректно работать в нескольких потоках одновременно.

**11. Что такое реентерабельность кода?**

-свойство одной копии программного кода работать в нескольких потоках одновременно.

-всегда потокобезопасен.

-не использует статическую память и не изменяет сам себя, все данные сохраняются в динамической памяти.

**12. Что такое Fiber?**

Fibers (Файберы, Фибра, волокна): Windows, Linux. Ручное планирование исполнение кода, механизм для ручного планирования выполнения кода в рамках потока

**13. Дайте развернутое определение потока OS.**

- поток – это объект OS

- поток – средство диспетчеризации доступа к процессорному времени (квант примерно 20мс);

- поток – последовательность команд процессора;

- поток – наименьшая единица работы ядра OS;

- процесс – контейнер для потоков;

- процесс имеет как минимум один поток (main);

- создание потока осуществляется с помощью системного вызова;

- потоки в рамках одного процесса не изолированы, все ресурсы кроме процессорного времени – общие;

- для работы с потоками в OS есть специальный API;

- каждый поток имеет свой идентификатор;

- состояния потока: исполняется, готов к исполнению, блокирован, спит; приостановлен;

- код потока – потоковая функция (специфицирована в OS);

- диспетчеризация потоков осуществляется OS или самим потоком;

- контекст потока – данные необходимые для возобновления работы потока при его приостановке (диспетчеризация, синхронизация): программный код, набор регистров, стек памяти, оперативная память, стек ядра, маркер доступа);

- поток может создавать дочерние потоки и их завершать;

- потоки могут создавать дерево потоков;

- завершение родительского потока приводит к завершению всех его дочерних (требуется ожидание дочернего завершение потока);

- многопоточность – парадигма программирования, поддерживается OS, приоритетная и корпоративная многопоточность;

- API в разных OS отличается; Linux IEEE POSIX, NPTL (Native POSIX Thread Library),

- потокобезопасность кода (программы) – свойство программного кода (программы) корректно работать в нескольких потоках одновременно;

- реентерабельность кода (программы) – свойство одной копии программного кода работать в нескольких потоках одновременно;

- фибра – механизм для ручного планирования выполнения кода в рамках потока.