Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа №4

По дисциплине «Операционные системы»

На тему «Управление потоками»

Выполнила:

Студентка 3 курса 7 группы

Гибадулина Алина Фархадовна

Преподаватель: асс. Бернацкий П.В.

2025, Минск

# 1 **Введение**

**Цель работы:** получить практические навыки в работе с потоками в операционных системах Windows и Linux, включая их создание, управление (приостановку, возобновление, завершение), синхронизацию и мониторинг.

**Используемые инструменты:**

* PowerShell
* Process Explore

# 2 Основная часть

## 2.1 Выполнение задания для Windows

### 2.1.1 Функция Lab-04х:

Функция выполняет N-ое количество итераций в цикле. На каждой итерации поток выводит следующее:

• Идентификатор процесса;

• Идентификатор потока;

• Номер итерации;

• Одну из букв вашего пользователя (см. в конце описания функции).

Количество итераций задаётся через аргументы функции.

Между итерациями должна быть задержка в 300-400 мс.

Предусмотреть обработку ошибок и корректную очистку всех ресурсов.

Вывод данной функции должен быть организован через семейство функций printf. Сам вывод должен удовлетворять шаблону:

PID – TID - № итерации – Буква из имени вашего пользователя

Причём буквы из вашего имени выводятся циклически. (Прим. Для User-d3hc25h на 1-й итерации будет U, 2-й – s, 3-й – e, и так далее)

|  |
| --- |
| #include <windows.h>  #include <iostream>  #include <string>  #include <cctype>  using namespace std;  const char\* username = "User-db4d49dd";  bool isABS(char c) {  return (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z');  }  string getOnlyLetters(const char\* username) {  string letters = "";  for (int i = 0; i < strlen(username); i++) {  if (isABS(username[i])) {  letters += username[i];  }  }  return letters;  }  DWORD WINAPI Lab\_04x(LPVOID lpParam) {  int iterations = \*(int\*)lpParam;  string letters = getOnlyLetters(username);  if (letters.empty()) {  cout << "no letters" << endl;  return 1;  }  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  for (int i = 0; i < iterations; i++) {  char letter = letters[i % letters.length()];  printf("%d - %d - %d - %c\n", pid, tid, i + 1, letter);  Sleep(300 + (rand() % 101));  }  return 0;  }  int main() {  int iterations = strlen(username);  cout << "Iterations number: " << iterations << endl;  cout << "Username: " << username << endl;  Lab\_04x(&iterations);  return 0;  } |

Листинг 2.1.1.1 – Код функции Lab-04x

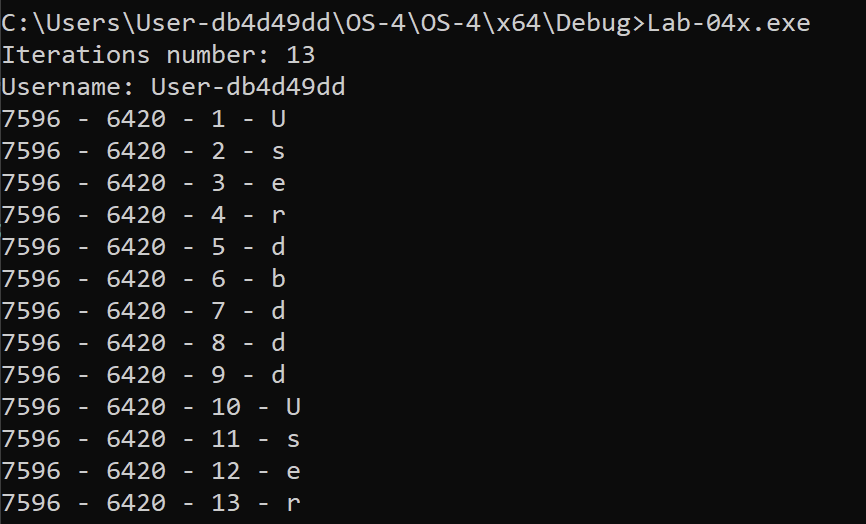


Рисунок 2.1.1.1 – Работа функции Lab-04x

### 2.1.2 Приложение Lab-04a

Программа создаёт два дочерних потока в которых выполняется функция Lab\_04x. При этом в первом и втором потоках выполняются 50 и 125 итераций соответственно.

В главном потоке также вызывается функция Lab\_04x так, чтобы выполнялось 100 итераций.

Главный поток должен ожидать завершения всех дочерних потоков перед тем, как завершать свою работу.

Пока процесс запущен, изучить информацию об имеющихся потоках через PowerShell и Process Explorer.

|  |
| --- |
| #include <windows.h>  #include <iostream>  #include <cstring>  #include <cctype>  using namespace std;  const char\* username = "User-db4d49dd";  bool isABS(char c) {  return (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z');  }  string getOnlyLetters(const char\* username) {  string letters = "";  for (int i = 0; i < strlen(username); i++) {  if (isABS(username[i])) {  letters += username[i];  }  }  return letters;  }  DWORD WINAPI Lab\_04x(LPVOID lpParam) {  int iterations = \*(int\*)lpParam;  string letters = getOnlyLetters(username);  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  cout << "Thread = " << tid << "was started the work (" << iterations << " iterations)" << endl;  for (int i = 0; i < iterations; i++) {  char letter = letters[i % letters.length()];  printf("%d - %d - %d - %c\n", pid, tid, i + 1, letter);  Sleep(300 + (rand() % 101));  }  cout << "Thread " << tid << " was ended the work" << endl;  return 0;  }  int main() {  string letters = getOnlyLetters(username);  cout << "Main thread PID: " << GetCurrentProcessId() << endl;  cout << "Username: " << username << endl;  int iter1 = 50;  int iter2 = 125;  int iterMain = 100;  HANDLE threads[2];  threads[0] = CreateThread(NULL, 0, Lab\_04x, &iter1, 0, NULL);  if (threads[0] == NULL) {  cout << "Error thread 1!" << endl;  return 1;  }  threads[1] = CreateThread(NULL, 0, Lab\_04x, &iter2, 0, NULL);  if (threads[1] == NULL) {  std::cout << "Error thread 2!" << std::endl;  CloseHandle(threads[0]);  return 1;  }  cout << "Threads created:" << endl;  cout << "Thread 1: " << iter1 << " iterations" << endl;  cout << "Thread 2: " << iter2 << " iterations" << endl;  cout << "Main thread: " << iterMain << " iterations" << endl;  Lab\_04x(&iterMain);  cout << "Main thread is ended. Wait threads 1 and 2" << endl;  WaitForMultipleObjects(2, threads, TRUE, INFINITE);  CloseHandle(threads[0]);  CloseHandle(threads[1]);  return 0;  } |

Листинг 2.1.2.1 – Исходный код приложения Lab-04a

Результат выполнения представлен на рисунке 2.1.2.1 и 2.1.2.2.



Рисунок 2.1.2.1 – Результат работы Lab-04a.exe

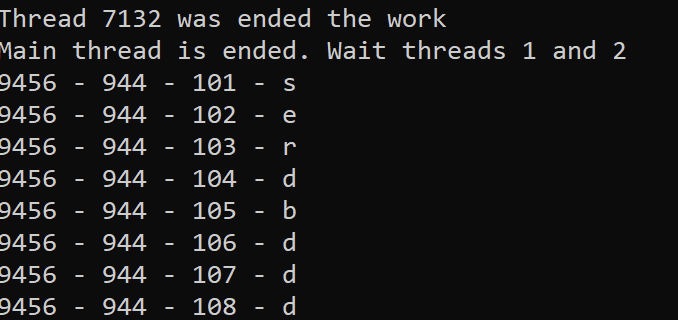


Рисунок 2.1.2.2 – Главный поток ожидает завершения всех дочерних потоков

Теперь выполним программу и проведем тест, который демонстрирует корректную работу с Proccess Explorer:

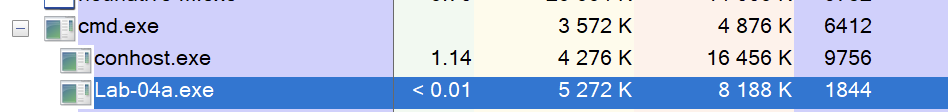


Рисунок 2.1.2.3 – Работа программы Lab-04а с Proccess Explorer

Просмотр информации о созданных потоках при помощи PowerShell:

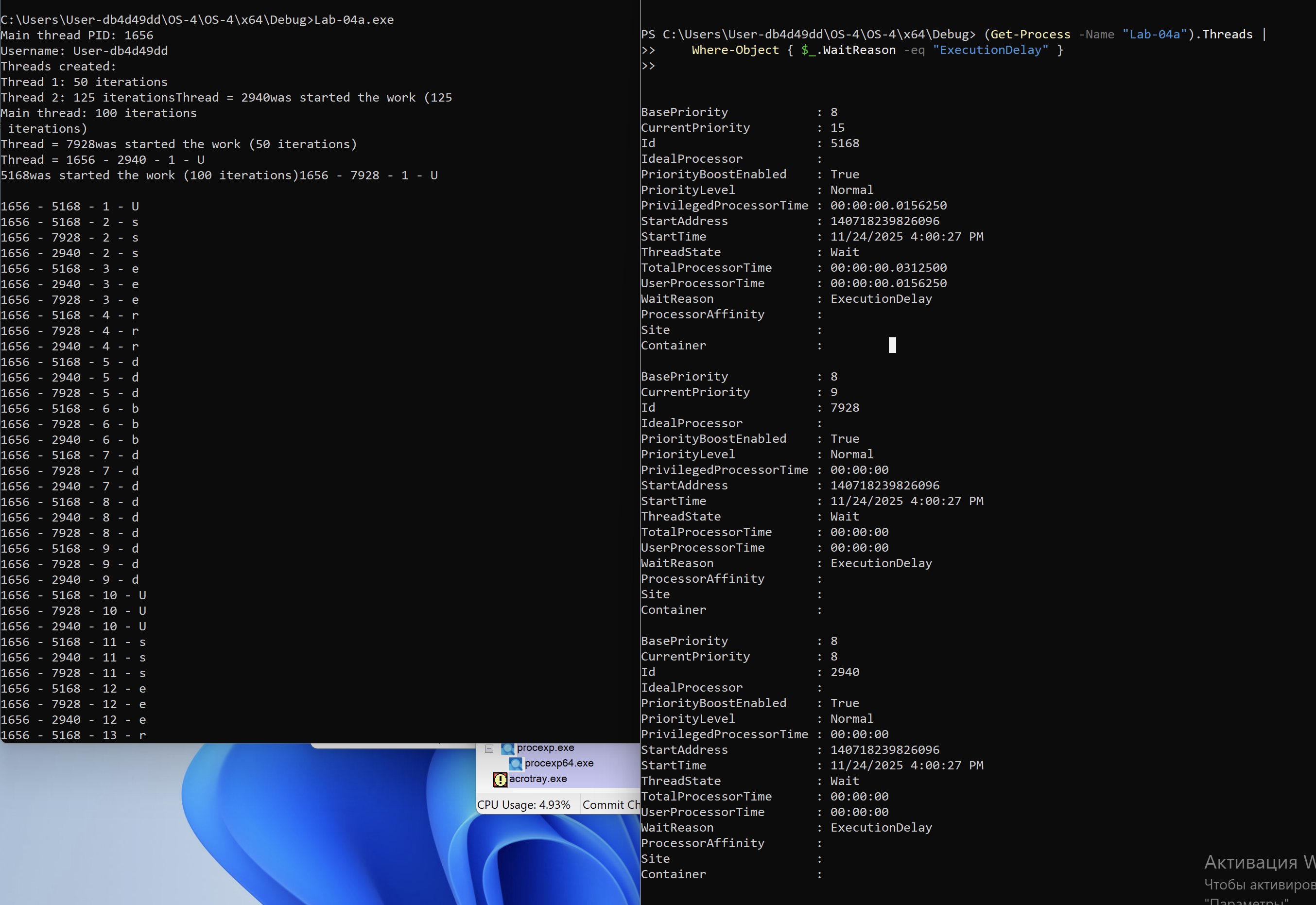


Рисунок 2.1.2.4 – Информация о процессах в PowerShell

### 2.1.3 Приложение Lab-04b

Программа в целом идентична Lab-04a, за исключением следующего:

• Вызов функции Lab\_04x в главном потоке заменить на аналогичный цикл;

• Добавить в цикл приостановку первого потока на 20 итерации и возобновление на 60 итерации;

• Добавить приостановку второго потока на 40 итерации, а возобновление после выполнения этого цикла.

Главный поток должен ожидать завершения всех дочерних потоков перед тем, как завершать свою работу.

Пока процесс запущен, изучить информацию об имеющихся потоках через PowerShell и Process Explorer.

|  |
| --- |
| #include <windows.h>  #include <iostream>  #include <cstring>  #include <cctype>  using namespace std;  const char\* username = "User-db4d49dd";  bool isABS(char c) {  return (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z');  }  string getOnlyLetters(const char\* username) {  string letters = "";  for (int i = 0; i < strlen(username); i++) {  if (isABS(username[i])) {  letters += username[i];  }  }  return letters;  }  DWORD WINAPI Lab\_04x(LPVOID lpParam) {  int iterations = \*(int\*)lpParam;  string letters = getOnlyLetters(username);  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  cout << "Thread " << tid << " was started (" << iterations << " iterations)" << endl;  for (int i = 0; i < iterations; i++) {  char letter = letters[i % letters.length()];  printf("%d - %d - %d - %c\n", pid, tid, i + 1, letter);  Sleep(300 + (rand() % 101));  }  cout << "Thread " << tid << " was ended" << endl;  return 0;  }  int main() {  string letters = getOnlyLetters(username);  cout << "Main thread PID: " << GetCurrentProcessId() << endl;  cout << "Username: " << username << endl;  int iter1 = 125;  int iter2 = 125;  int iterMain = 100;  HANDLE threads[2];  DWORD threadIds[2];  threads[0] = CreateThread(NULL, 0, Lab\_04x, &iter1, 0, &threadIds[0]);  if (threads[0] == NULL) {  cout << "Error creating thread 1!" << endl;  return 1;  }  threads[1] = CreateThread(NULL, 0, Lab\_04x, &iter2, 0, &threadIds[1]);  if (threads[1] == NULL) {  cout << "Error creating thread 2!" << endl;  CloseHandle(threads[0]);  return 1;  }  cout << "Threads created:" << endl;  cout << "Thread 1: " << iter1 << " iterations (TID: " << threadIds[0] << ")" << endl;  cout << "Thread 2: " << iter2 << " iterations (TID: " << threadIds[1] << ")" << endl;  cout << "Main thread: " << iterMain << " iterations" << endl;  DWORD mainPid = GetCurrentProcessId();  DWORD mainTid = GetCurrentThreadId();  cout << "Main thread started work" << endl;  for (int i = 0; i < iterMain; i++) {  char letter = username[i % strlen(username)];  printf("%d - %d - %d - %c\n", mainPid, mainTid, i + 1, letter);  if (i == 19) {  cout << "!!! SUSPENDING Thread 1 at iteration 20 !!!" << endl;  SuspendThread(threads[0]);  }  if (i == 59) {  cout << "!!! RESUMING Thread 1 at iteration 60 !!!" << endl;  ResumeThread(threads[0]);  }  if (i == 39) {  cout << "!!! SUSPENDING Thread 2 at iteration 40 !!!" << endl;  SuspendThread(threads[1]);  }  Sleep(300 + (rand() % 101));  }  cout << "!!! RESUMING Thread 2 after main thread completion !!!" << endl;  ResumeThread(threads[1]);  cout << "Main thread completed. Waiting for child threads." << endl;  WaitForMultipleObjects(2, threads, TRUE, INFINITE);  CloseHandle(threads[0]);  CloseHandle(threads[1]);  return 0;  } |

Листинг 2.1.3.1 – Исходный код программы Lab-04b

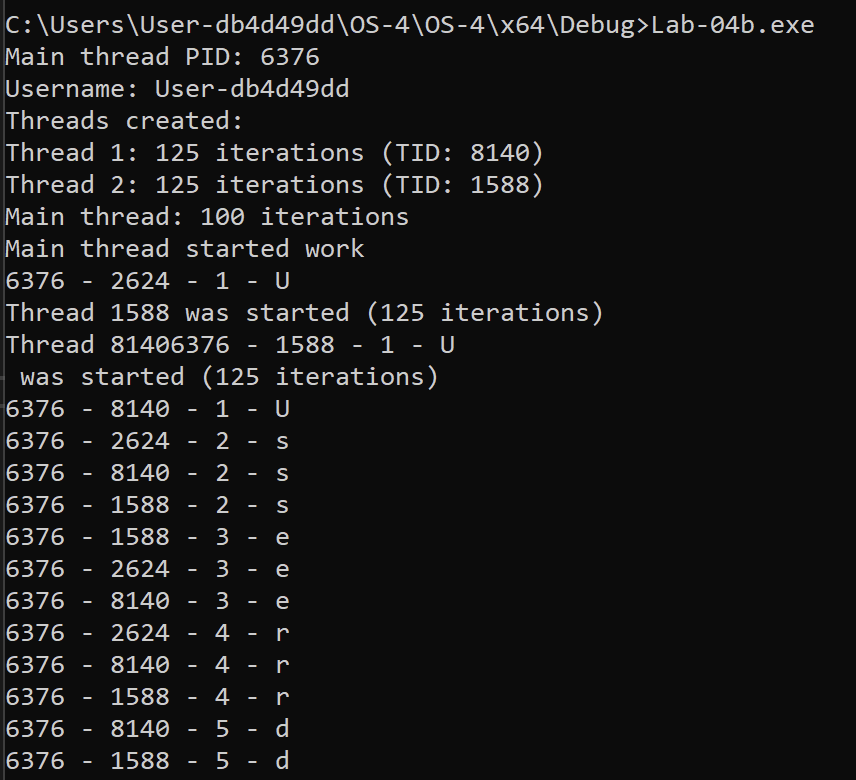


Рисунок 2.1.3.1 – Работа программы Lab-03b

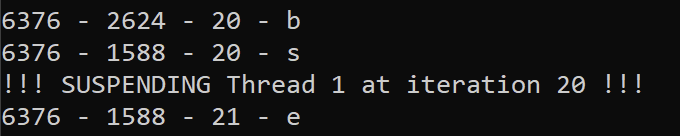


Рисунок 2.1.3.2 – Приостановка работы потока

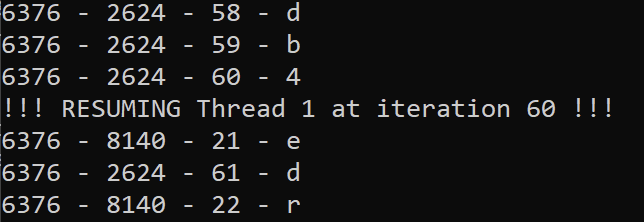


Рисунок 2.1.3.3 – Возобновление работы потока



Рисунок 2.1.3.4 – Главный поток ожидает завершения всех дочерних потоков

Теперь выполним программу и проведем тест, который демонстрирует корректную работу с Proccess Explorer:

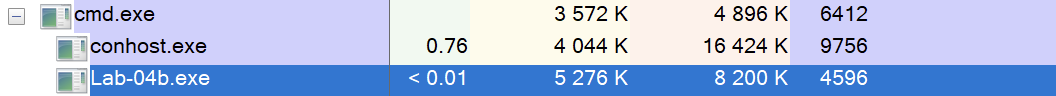


Рисунок 2.1.3.5 – Работа программы Lab-04b с Proccess Explorer

Просмотр информации о созданных потоках при помощи PowerShell:

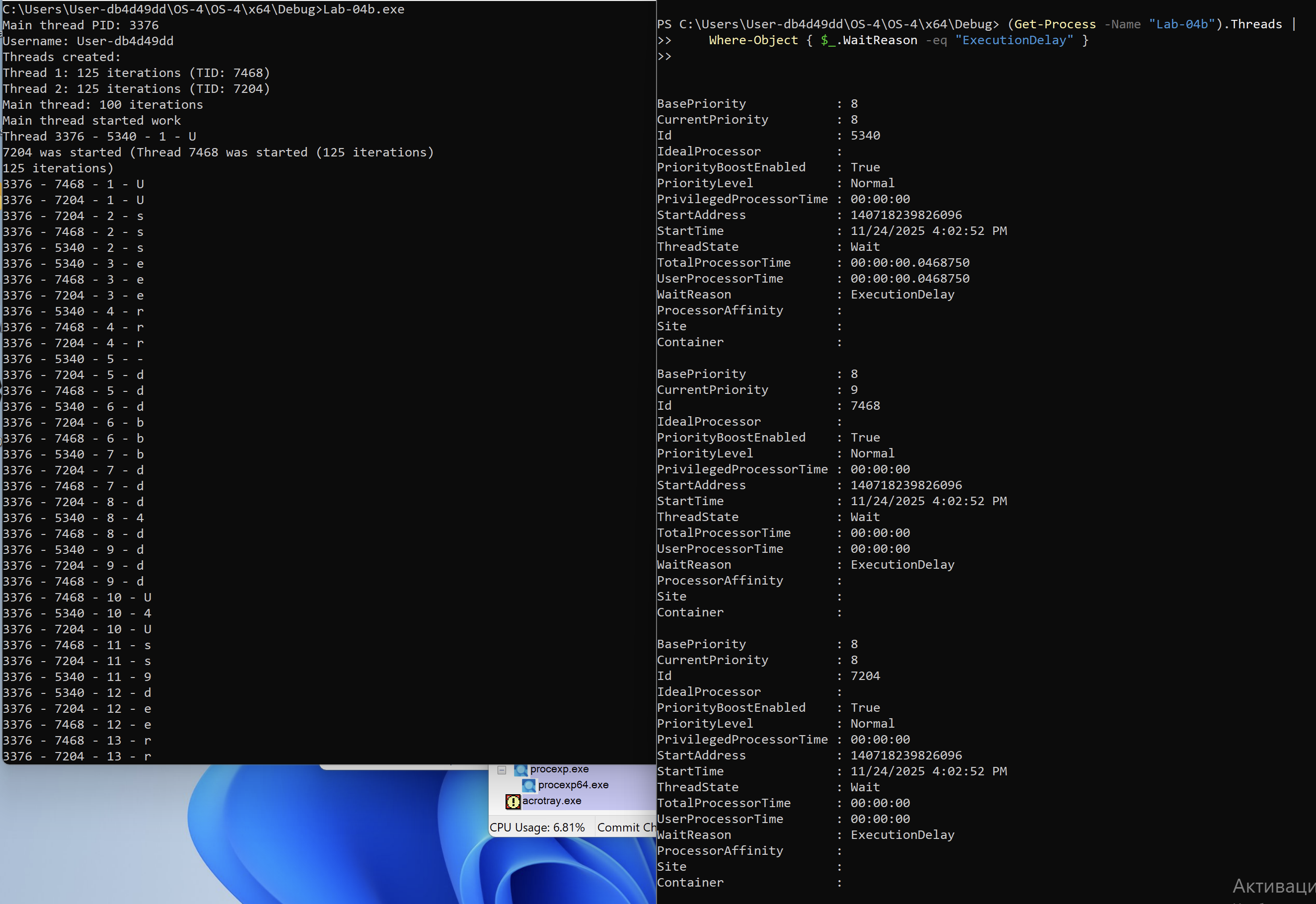


Рисунок 2.1.3.6 – Информация о процессах в PowerShell

### 2.1.4 Приложение Lab-04c

Программа в целом идентична Lab-04a, за исключением следующего:

• Вызов функции Lab-04x в главном потоке заменить на аналогичный цикл;

• Добавить в цикл завершение второго потока на 40 итерации.

Главный поток должен ожидать завершения всех дочерних потоков перед тем, как завершать свою работу.

Пока процесс запущен, изучить информацию об имеющихся потоках через PowerShell и Process Explorer.

|  |
| --- |
| #include <windows.h>  #include <iostream>  #include <string>  #include <cctype>  using namespace std;  const char\* username = "User-db4d49dd";  bool isAlpha(char c) {  return (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z');  }  string getOnlyLetters(const char\* username) {  string letters = "";  for (int i = 0; i < strlen(username); i++) {  if (isAlpha(username[i])) {  letters += username[i];  }  }  return letters;  }  struct ThreadData {  int iterations;  int threadNumber;  HANDLE threadHandle;  };  DWORD WINAPI Lab\_04x(LPVOID lpParam) {  ThreadData\* data = (ThreadData\*)lpParam;  int iterations = data->iterations;  int threadNumber = data->threadNumber;  string letters = getOnlyLetters(username);  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  cout << "Thread " << tid << " (Thread " << threadNumber << ") started (" << iterations << " iterations)" << endl;  for (int i = 0; i < iterations; i++) {  char letter = letters[i % letters.length()];  printf("%d - %d - %d - %c\n", pid, tid, i + 1, letter);  Sleep(300 + (rand() % 101));  if (threadNumber == 2 && i == 39) {  cout << "!!! TERMINATING Thread 2 at iteration 40 !!!" << endl;  TerminateThread(data->threadHandle, 0);  break;  }  }  cout << "Thread " << tid << " (Thread " << threadNumber << ") ended" << endl;  return 0;  }  int main() {  string letters = getOnlyLetters(username);  cout << "Main thread PID: " << GetCurrentProcessId() << endl;  cout << "Username: " << username << endl;  cout << "Letters only: " << letters << endl;  ThreadData thread1Data = { 125, 1, NULL };  ThreadData thread2Data = { 125, 2, NULL };  HANDLE threads[2];  threads[0] = CreateThread(NULL, 0, Lab\_04x, &thread1Data, 0, NULL);  if (threads[0] == NULL) {  cout << "Error creating thread 1!" << endl;  return 1;  }  thread1Data.threadHandle = threads[0];  threads[1] = CreateThread(NULL, 0, Lab\_04x, &thread2Data, 0, NULL);  if (threads[1] == NULL) {  cout << "Error creating thread 2!" << endl;  CloseHandle(threads[0]);  return 1;  }  thread2Data.threadHandle = threads[1];  cout << "Threads created:" << endl;  cout << "Thread 1: " << thread1Data.iterations << " iterations" << endl;  cout << "Thread 2: " << thread2Data.iterations << " iterations (will be terminated at iteration 40)" << endl;  cout << "Main thread: 100 iterations" << endl;  DWORD mainPid = GetCurrentProcessId();  DWORD mainTid = GetCurrentThreadId();  cout << "Main thread started work" << endl;  for (int i = 0; i < 100; i++) {  char letter = letters[i % letters.length()];  printf("%d - %d - %d - %c\n", mainPid, mainTid, i + 1, letter);  if (i == 39) {  cout << "!!! TERMINATING Thread 2 at main thread iteration 40 !!!" << endl;  TerminateThread(threads[1], 0);  }  Sleep(300 + (rand() % 101));  }  cout << "Main thread completed. Waiting for child threads..." << endl;  DWORD result = WaitForMultipleObjects(2, threads, TRUE, INFINITE);  if (result == WAIT\_FAILED) {  cout << "Wait failed! Error: " << GetLastError() << endl;  }  else {  cout << "All threads completed waiting!" << endl;  }  DWORD exitCode1, exitCode2;  GetExitCodeThread(threads[0], &exitCode1);  GetExitCodeThread(threads[1], &exitCode2);  cout << "Thread 1 exit code: " << exitCode1 << endl;  cout << "Thread 2 exit code: " << exitCode2 << endl;    CloseHandle(threads[0]);  CloseHandle(threads[1]);  cout << "All thread handles closed!" << endl;  return 0;  } |

Листинг 2.1.4.1 – Программа Lab-04c

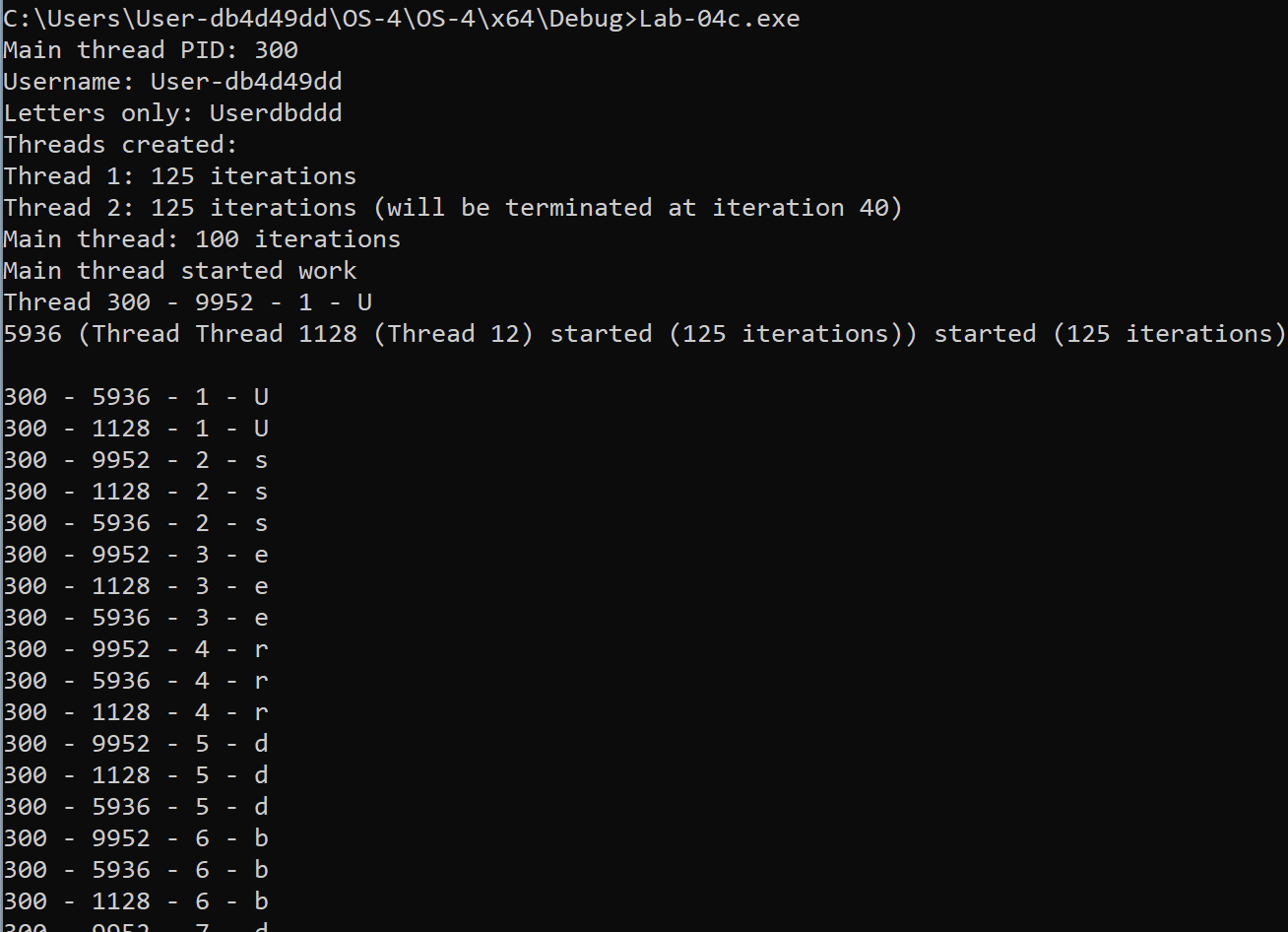


Рисунок 2.1.4.1 –Результат работы Lab-04c

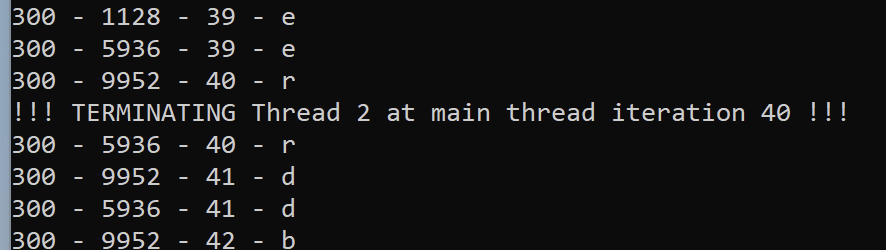


Рисунок 2.1.4.2 –Результат работы Lab-04c

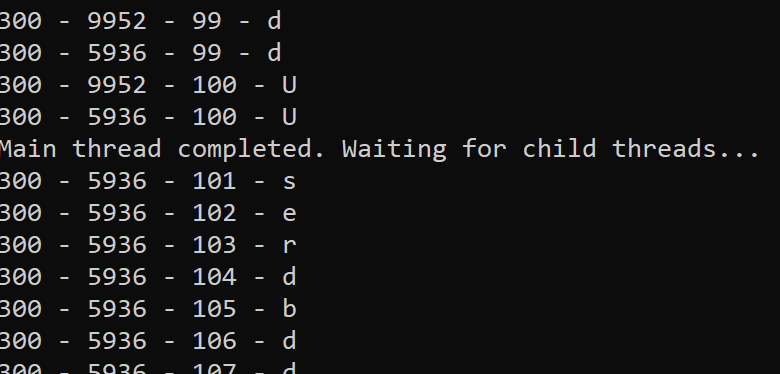


Рисунок 2.1.4.3 – Главный поток ожидает завершения всех дочерних потоков

Теперь выполним программу и проведем тест, который демонстрирует корректную работу с Proccess Explorer:

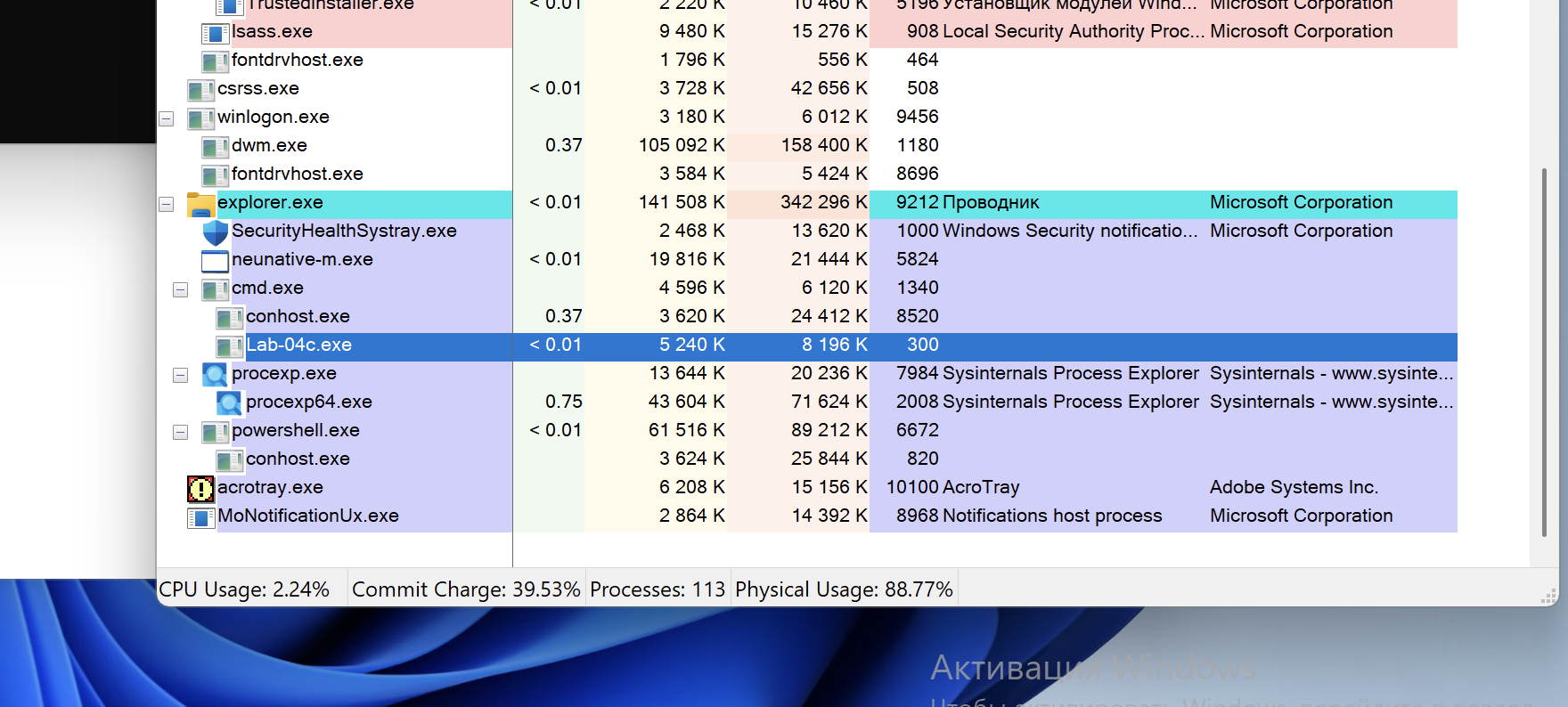


Рисунок 2.1.4.4 – Работа программы Lab-04b с Proccess Explorer

Просмотр информации о созданных потоках при помощи PowerShell:

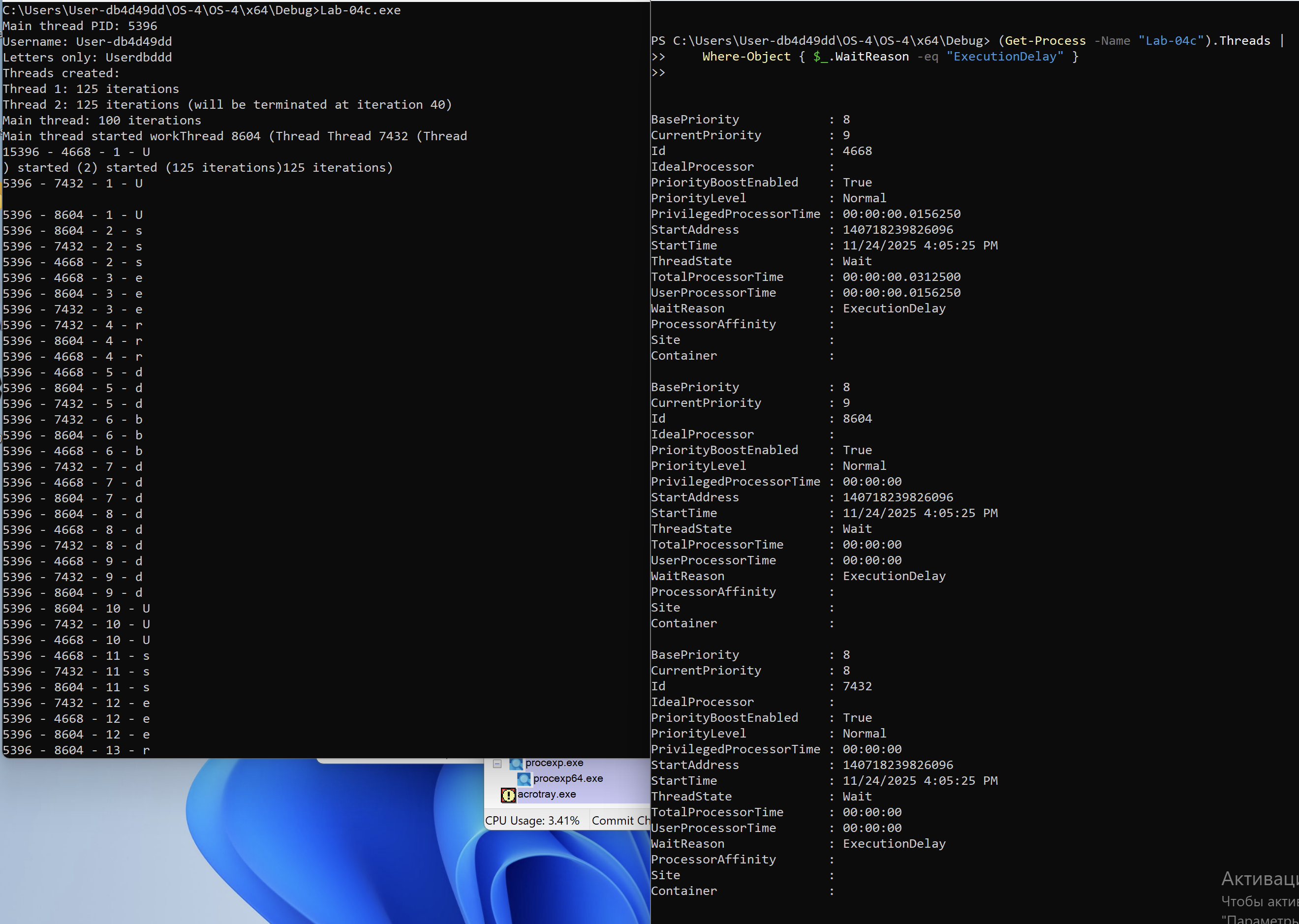


Рисунок 2.1.4.5 – Информация о процессах в PowerShell

### 2.1.5 Приложение Lab-04d

Создать программу, которая выводит список запущенных в процессе потоков. Для каждого потока выводится его идентификатор. (Tool Help Library).

|  |
| --- |
| #include <windows.h>  #include <iostream>  #include <tlhelp32.h>  #include <string>  #include <cctype>  using namespace std;  const char\* username = "User-db4d49dd";  bool isAlpha(char c) {  return (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z');  }  string getOnlyLetters(const char\* username) {  string letters = "";  for (int i = 0; i < strlen(username); i++) {  if (isAlpha(username[i])) {  letters += username[i];  }  }  return letters;  }  void PrintProcessThreads(DWORD processId) {  HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS\_SNAPTHREAD, 0);  if (hSnapshot == INVALID\_HANDLE\_VALUE) {  cout << "Error creating snapshot: " << GetLastError() << endl;  return;  }  THREADENTRY32 te32;  te32.dwSize = sizeof(THREADENTRY32);  cout << "=== Threads of process " << processId << " ===" << endl;  if (Thread32First(hSnapshot, &te32)) {  int threadCount = 0;  do {  if (te32.th32OwnerProcessID == processId) {  threadCount++;  cout << "Thread ID: " << te32.th32ThreadID  << " | Priority: " << (int)te32.tpBasePri  << endl;  }  } while (Thread32Next(hSnapshot, &te32));  cout << "Total threads: " << threadCount << endl;  }  else {  cout << "Thread32First failed: " << GetLastError() << endl;  }  CloseHandle(hSnapshot);  cout << "=========================" << endl;  }  DWORD WINAPI Lab\_04x(LPVOID lpParam) {  int iterations = \*(int\*)lpParam;  string letters = getOnlyLetters(username);  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  cout << "Thread " << tid << " started (" << iterations << " iterations)" << endl;  for (int i = 0; i < iterations; i++) {  char letter = letters[i % letters.length()];  printf("%d - %d - %d - %c\n", pid, tid, i + 1, letter);  Sleep(300 + (rand() % 101));  }  cout << "Thread " << tid << " ended" << endl;  return 0;  }  int main() {  string letters = getOnlyLetters(username);  DWORD currentPid = GetCurrentProcessId();  cout << "Current PID: " << currentPid << endl;  cout << "Username: " << username << endl;  cout << "Letters only: " << letters << endl;  cout << "BEFORE creating worker threads:" << endl;  PrintProcessThreads(currentPid);  int iter1 = 10;  int iter2 = 15;  HANDLE workerThreads[2];  workerThreads[0] = CreateThread(NULL, 0, Lab\_04x, &iter1, 0, NULL);  workerThreads[1] = CreateThread(NULL, 0, Lab\_04x, &iter2, 0, NULL);  if (workerThreads[0] == NULL || workerThreads[1] == NULL) {  cout << "Error creating worker threads!" << endl;  return 1;  }  Sleep(500);  cout << "AFTER creating worker threads:" << endl;  PrintProcessThreads(currentPid);  DWORD mainTid = GetCurrentThreadId();  cout << "Main thread working (TID: " << mainTid << ")" << endl;  for (int i = 0; i < 5; i++) {  char letter = letters[i % letters.length()];  printf("%d - %d - %d - %c\n", currentPid, mainTid, i + 1, letter);  Sleep(400);  }  WaitForMultipleObjects(2, workerThreads, TRUE, INFINITE);  cout << "AFTER worker threads completed:" << endl;  PrintProcessThreads(currentPid);  CloseHandle(workerThreads[0]);  CloseHandle(workerThreads[1]);  cout << "All threads completed and handles closed!" << endl;  return 0;  } |

Листинг 2.1.5.1 – Программа Lab-04d

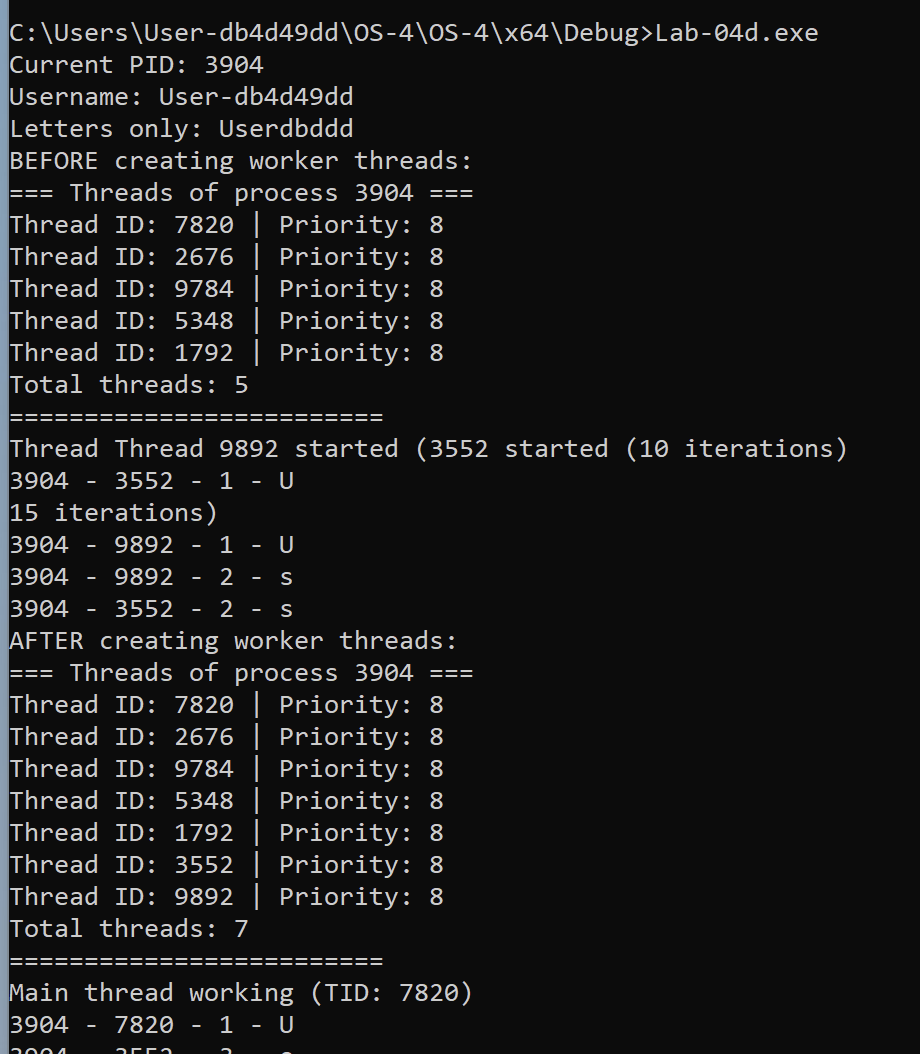


Рисунок 2.1.5.1 –Результат работы Lab-04d



Рисунок 2.1.5.2 –Результат работы Lab-04d

Теперь выполним программу и проведем тест, который демонстрирует корректную работу с Proccess Explorer:

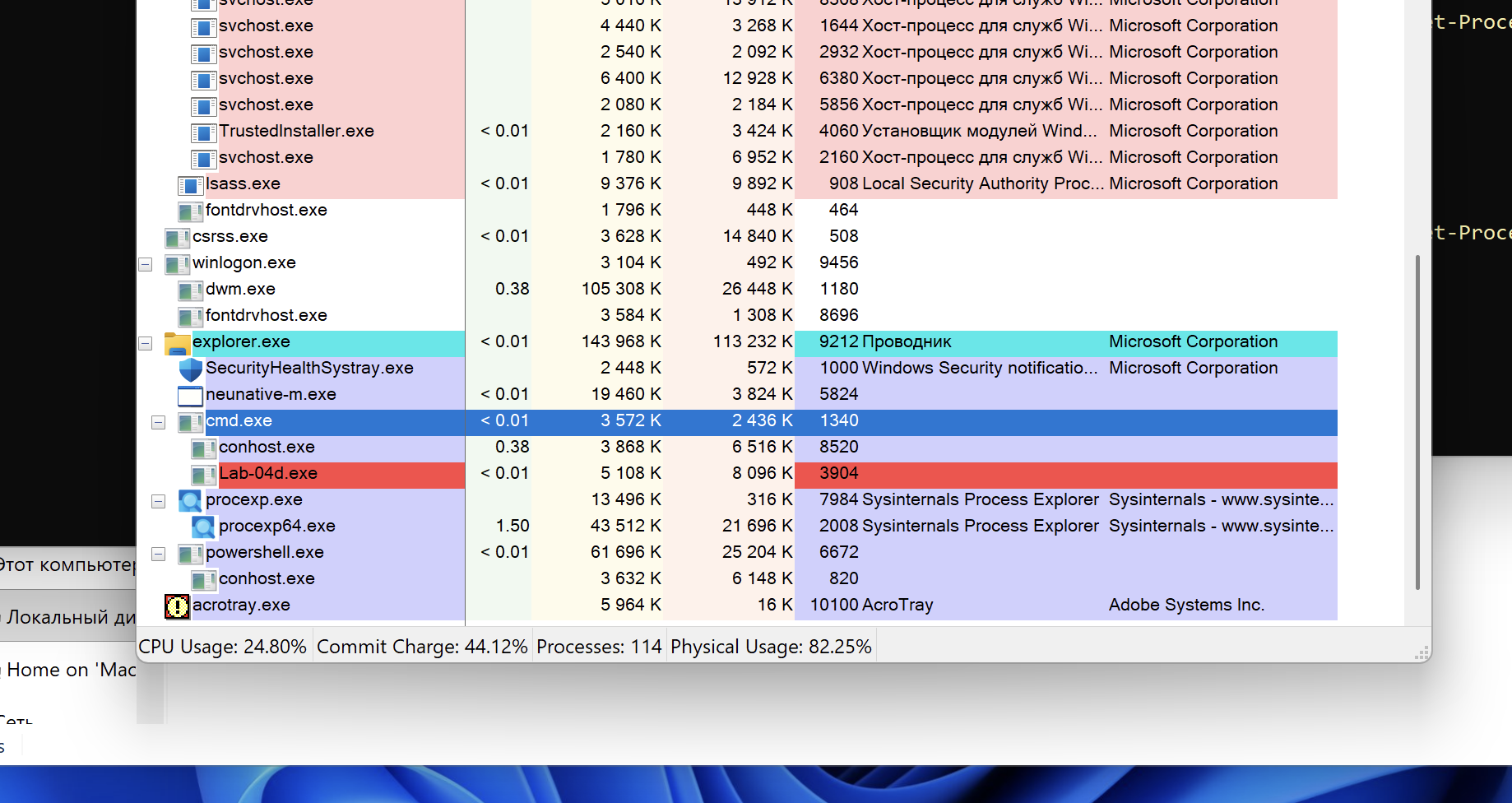


Рисунок 2.1.5.3 – Работа программы Lab-04b с Proccess Explorer

Просмотр информации о созданных потоках при помощи PowerShell:



Рисунок 2.1.5.4 – Информация о процессах в PowerShell

## 2.2 Выполнение заданий для Linux

### 2.2.1 Функция Lab-04x

Функция полностью идентична таковой из заданий по Windows.

|  |
| --- |
| #include <iostream>  #include <string>  #include <cctype>  #include <cstring>  #include <thread>  #include <chrono>  #include <unistd.h>  using namespace std;  const char\* username = "User-db4d49dd";  bool isABS(char c) {  return isalpha(static\_cast<unsigned char>(c));  }  string getOnlyLetters(const char\* username) {  string letters;  for (int i = 0; i < strlen(username); i++) {  if (isABS(username[i])) {  letters += username[i];  }  }  return letters;  }  void Lab\_04x(int iterations) {  string letters = getOnlyLetters(username);  if (letters.empty()) {  cout << "no letters" << endl;  return;  }  pid\_t pid = getpid();  auto tid = std::this\_thread::get\_id();  for (int i = 0; i < iterations; i++) {  char letter = letters[i % letters.length()];  cout << pid << " - " << tid << " - " << i + 1 << " - " << letter << endl;  this\_thread::sleep\_for(chrono::milliseconds(300 + rand() % 101));  }  }  int main() {  srand(time(nullptr));  int iterations = strlen(username);  cout << "Iterations number: " << iterations << endl;  cout << "Username: " << username << endl;  Lab\_04x(iterations);  return 0;  } |

Листинг 2.2.1.1 – Функция Lab-04x

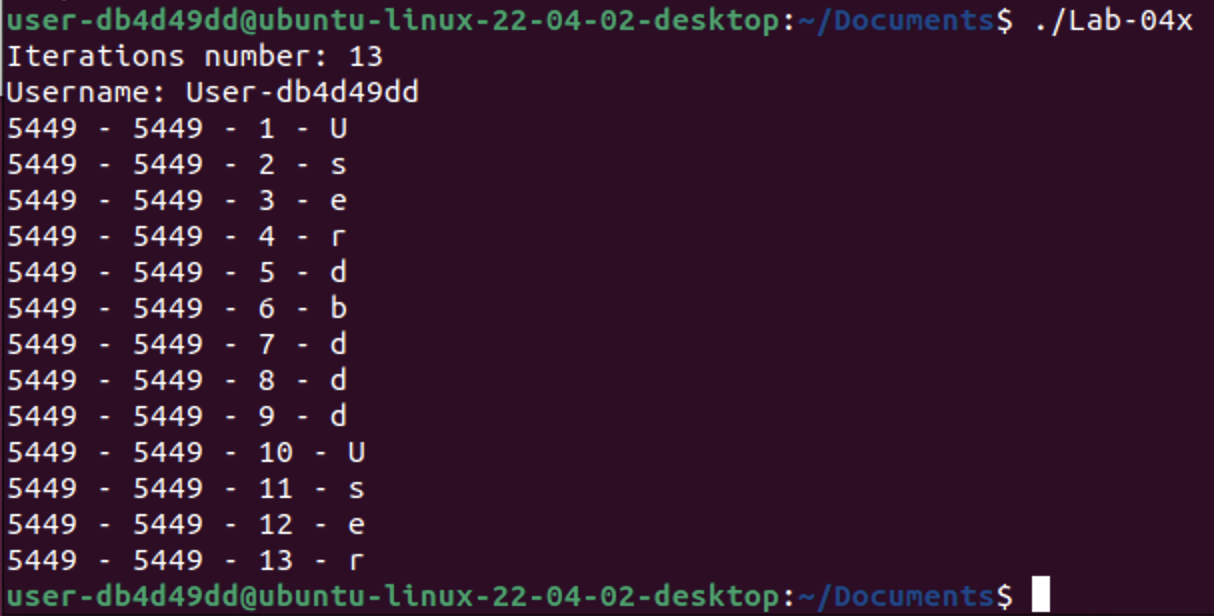


Рисунок 2.2.1.1 – Работа функции Lab-04x

### 2.2.2 Приложение Lab-04рх.exe

Программа создаёт два дочерних потока в которых выполняется функция Lab\_04x. При этом в первом и втором потоках выполняются 50 и 125 итераций соответственно.

В главном потоке также вызывается функция Lab\_04x так, чтобы выполнялось 100 итераций.

Главный поток должен ожидать завершения всех дочерних потоков перед тем, как завершать свою работу.

Изучить информацию о созданных потоках через файловую систему (каталог /proc) и утилиту ps.

|  |
| --- |
| #include <iostream>  #include <string>  #include <cctype>  #include <cstring>  #include <thread>  #include <chrono>  #include <unistd.h>  #include <vector>  #include <stdexcept>  using namespace std;  const char\* username = "User-db4d49dd";  bool isABS(char c) {  return isalpha(static\_cast<unsigned char>(c));  }  string getOnlyLetters(const char\* username) {  string letters;  for (int i = 0; i < strlen(username); i++) {  if (isABS(username[i])) {  letters += username[i];  }  }  return letters;  }  void Lab\_04x(int iterations, const string& threadName) {  try {  string letters = getOnlyLetters(username);  if (letters.empty()) {  cerr << "[" << threadName << "] error: no letters in username" << endl;  return;  }  pid\_t pid = getpid();  auto tid = this\_thread::get\_id();  cout << "[" << threadName << "] started (PID=" << pid  << ", TID=" << tid << ", iterations=" << iterations << ")" << endl;  for (int i = 0; i < iterations; i++) {  char letter = letters[i % letters.length()];  cout << pid << " - " << tid << " - " << i + 1 << " - " << letter << endl;  this\_thread::sleep\_for(chrono::milliseconds(300 + rand() % 101));  }  cout << "[" << threadName << "] completed" << endl;  }  catch (const exception& e) {  cerr << "[" << threadName << "] exception: " << e.what() << endl;  }  catch (...) {  cerr << "[" << threadName << "] unknown error" << endl;  }  }  int main() {  srand(time(nullptr));  cout << "Main process PID: " << getpid() << endl;  cout << "Username: " << username << endl;  cout << "Letters only: " << getOnlyLetters(username) << endl;  try {  thread t1(Lab\_04x, 50, "Thread-1");  thread t2(Lab\_04x, 125, "Thread-2");  cout << "Created threads:\n"  << "Thread-1 → 50 iterations\n"  << "Thread-2 → 125 iterations\n"  << "Main thread → 100 iterations\n" << endl;  Lab\_04x(100, "Main-Thread");  cout << "[Main] waiting for threads to complete..." << endl;  if (t1.joinable()) t1.join();  if (t2.joinable()) t2.join();  cout << "[Main] all threads completed successfully." << endl;  }  catch (const exception& e) {  cerr << "[Main] error: " << e.what() << endl;  }  catch (...) {  cerr << "[Main] unknown error occurred" << endl;  }  return 0;  } |

Листинг 2.2.2.1 – Код программы Lab-04рх

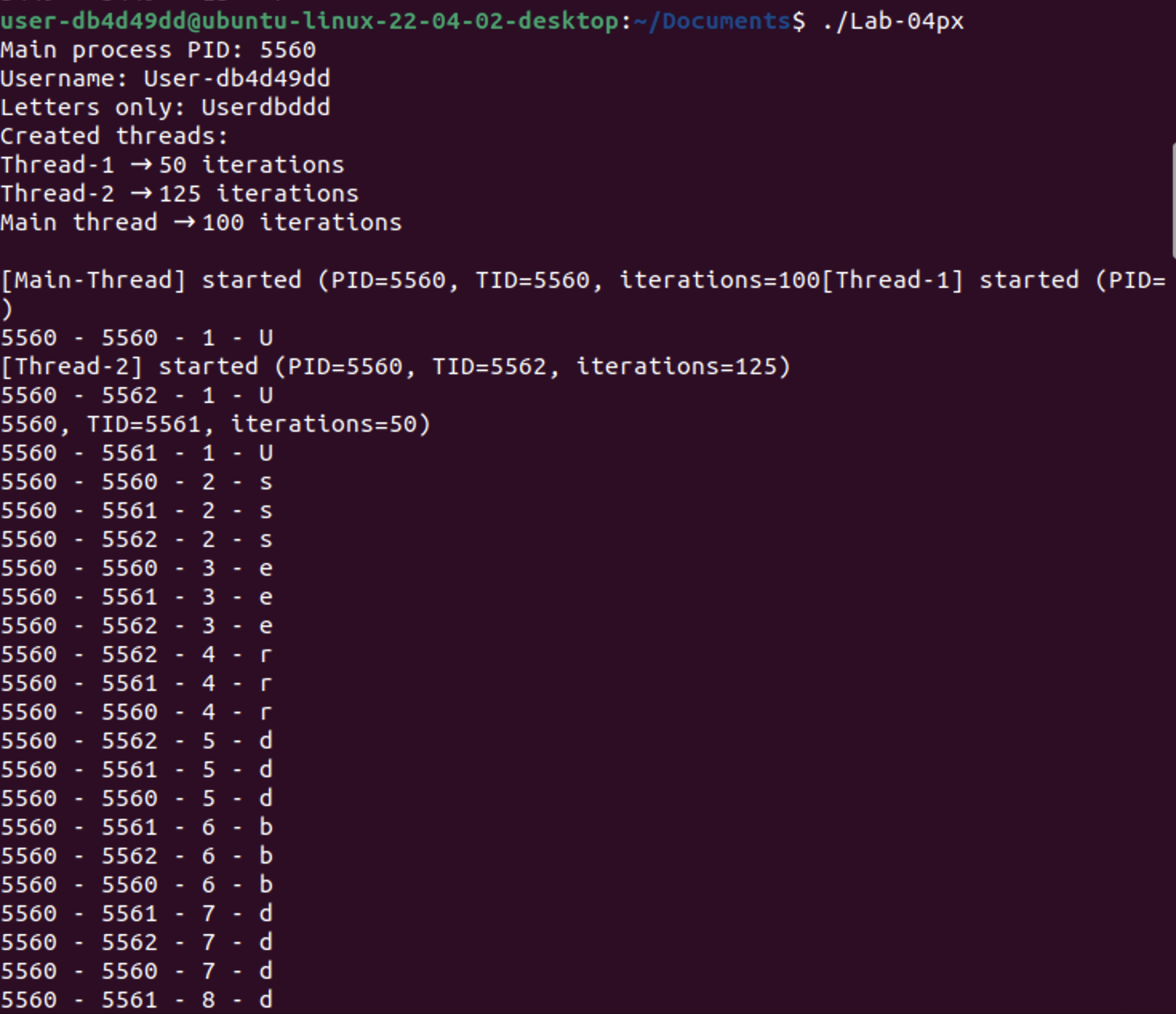


Рисунок 2.2.2.1 – Результат работы Lab-04рх.exe

После запуска можно проверить активные процессы и их PID следующими командами:

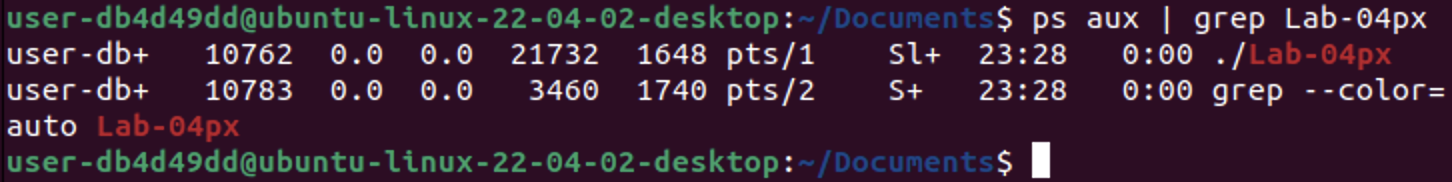


Рисунок 2.2.2.2 – Подтверждение запуска процессов через ps

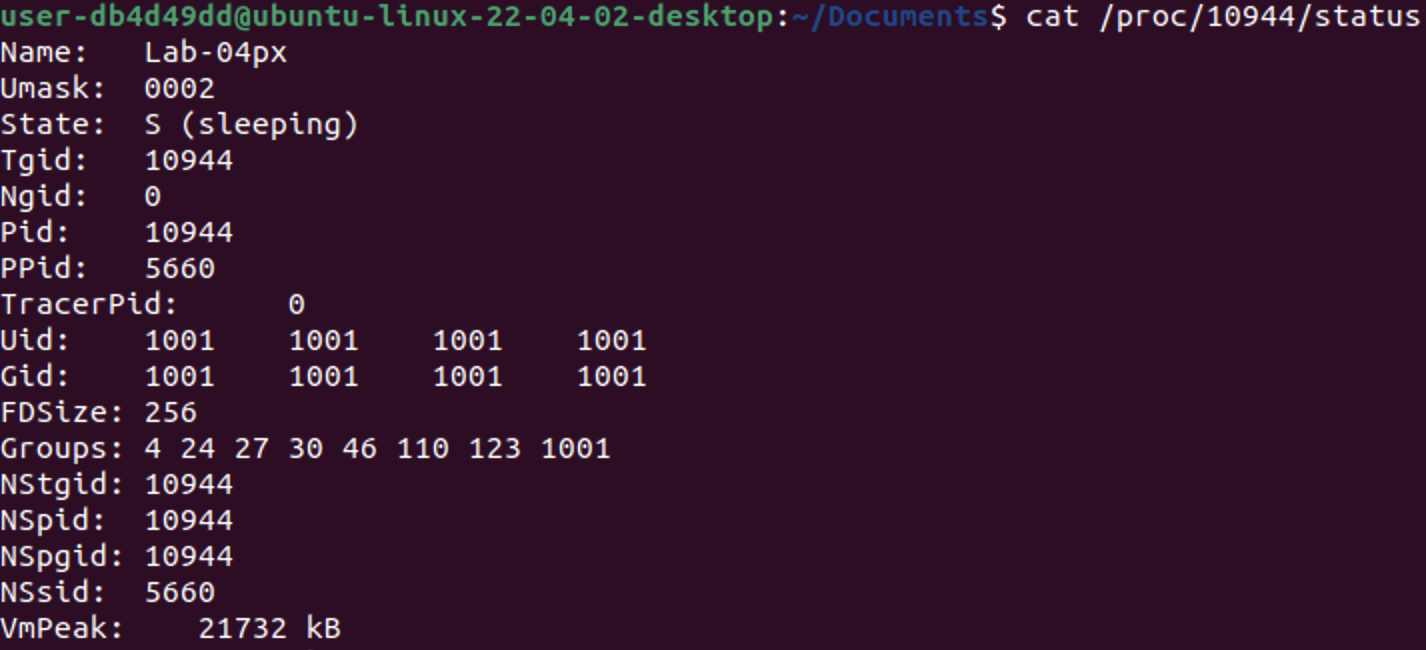


Рисунок 2.2.2.3 – Подтверждение запуска процессов через /proc

3 Вывод

В ходе лабораторной работы были получены практические навыки работы с потоками в ОС Windows и Linux. Были изучены механизмы создания потоков, их синхронизации, управления состояниями (приостановка, возобновление, принудительное завершение), а также мониторинга с использованием системных средств (PowerShell, Process Explorer, /proc, ps). Разработаны многопоточные приложения, обеспечивающие корректное выполнение задач, обработку ошибок и освобождение ресурсов.