МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ:

по дисциплине «Операционные системы»

Исполнитель

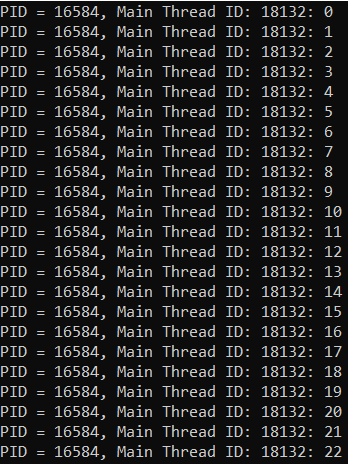
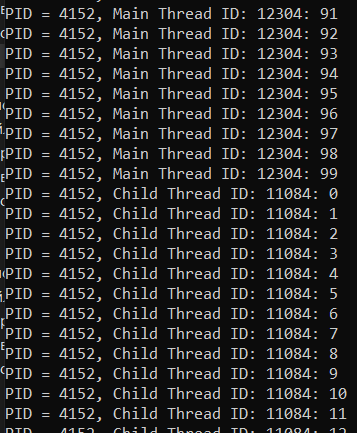
студент (ка) 3 курса группы 6 Розель Станислав Александрович

(Ф.И.О.)

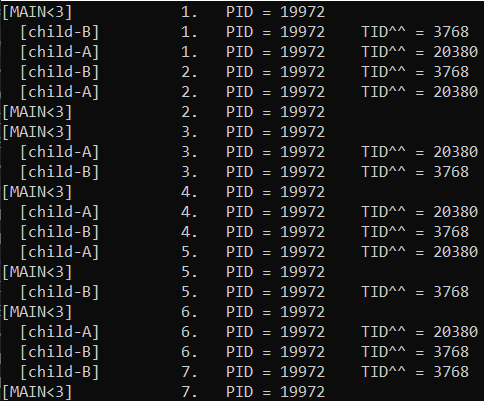
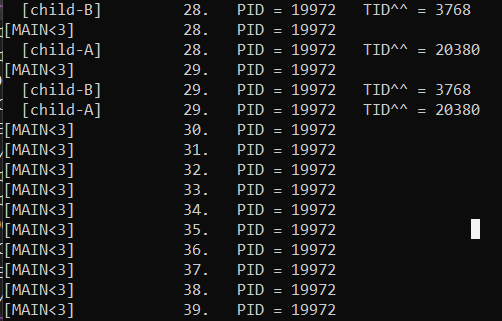
Минск 2024

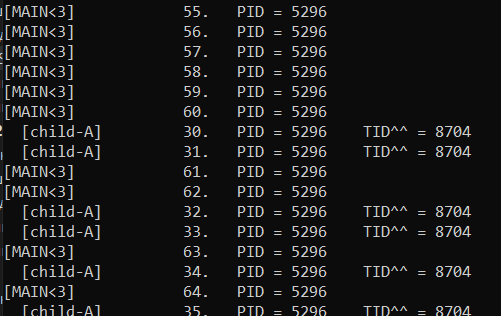
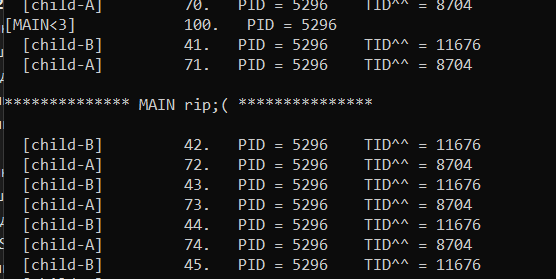
Задание 1

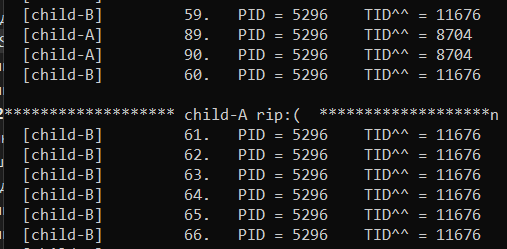
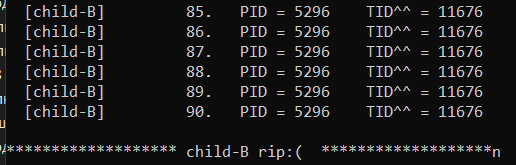
|  |
| --- |
| #include <Windows.h>  #include <iostream>  using namespace std;  DWORD pid = NULL;  int check;  void EnterCriticalSectionAsm()  {  \_asm  {  CriticalSection:  lock bts check, 0;  jc CriticalSection  }  }  void LeaveCriticalSectionAsm()  {  \_asm lock btr check, 0  }  DWORD WINAPI ChildThread()  {  DWORD tid = GetCurrentThreadId();  EnterCriticalSectionAsm();  for (short i = 0; i < 100; i++)  {  std::cout << "PID = " << pid << ", Child Thread ID: " << tid << ": " << i << std::endl;  Sleep(40);  }  LeaveCriticalSectionAsm();  return 0;  }  int main()  {  pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  DWORD ChildId = NULL;  HANDLE hChild = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildThread, NULL, 0, &ChildId);  EnterCriticalSectionAsm();  for (short i = 0; i < 100; i++)  {  std::cout << "PID = " << pid << ", Main Thread ID: " << tid << ": " << i << std::endl;  Sleep(40);  }  LeaveCriticalSectionAsm();  WaitForSingleObject(hChild, INFINITE);  CloseHandle(hChild);  return 0;  } |

Задание 2

|  |
| --- |
| #include <iostream>  #include <windows.h>  using namespace std;  CRITICAL\_SECTION critical\_section;  HANDLE createThread(LPTHREAD\_START\_ROUTINE func, char\* thread\_name)  {  DWORD thread\_id = NULL;  HANDLE thread = CreateThread(NULL, 0, func, thread\_name, 0, &thread\_id);  if (thread == NULL)  throw "[ERROR] CreateThread";  return thread;  }  void WINAPI loop(char\* displayed\_name)  {  int pid = GetCurrentProcessId();  int tid = GetCurrentThreadId();  for (int i = 1; i <= 90; ++i)  {  if (i == 30)  EnterCriticalSection(&critical\_section);  printf(" [%s]\t %d. PID = %d\tTID^^ = %u\n", displayed\_name, i, pid, tid);  if (i == 60)  LeaveCriticalSection(&critical\_section);  Sleep(100);  }  cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* " << displayed\_name << " rip:(" << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*n\n";  }  int main()  {  int pid = GetCurrentProcessId();  const int size = 2;  HANDLE threads[size];  threads[0] = createThread((LPTHREAD\_START\_ROUTINE)loop, (char\*)"child-A");  threads[1] = createThread((LPTHREAD\_START\_ROUTINE)loop, (char\*)"child-B");  InitializeCriticalSection(&critical\_section);  for (int i = 1; i <= 100; ++i)  {  if (i == 30)  EnterCriticalSection(&critical\_section);  printf("[MAIN<3]\t %d. PID = %d\n", i, pid);  if (i == 60)  LeaveCriticalSection(&critical\_section);  Sleep(100);  }  cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN rip;( \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";  WaitForMultipleObjects(size, threads, TRUE, INFINITE);  for (int i = 0; i < size; ++i)  CloseHandle(threads[i]);  DeleteCriticalSection(&critical\_section);  return 0;  } |

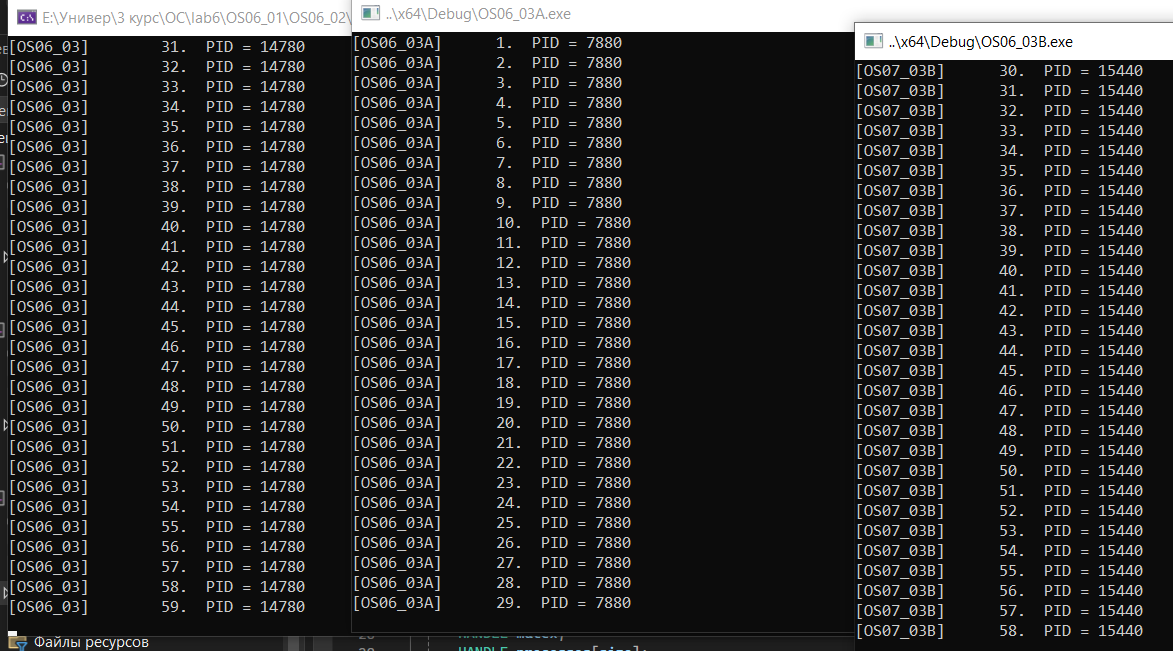
 

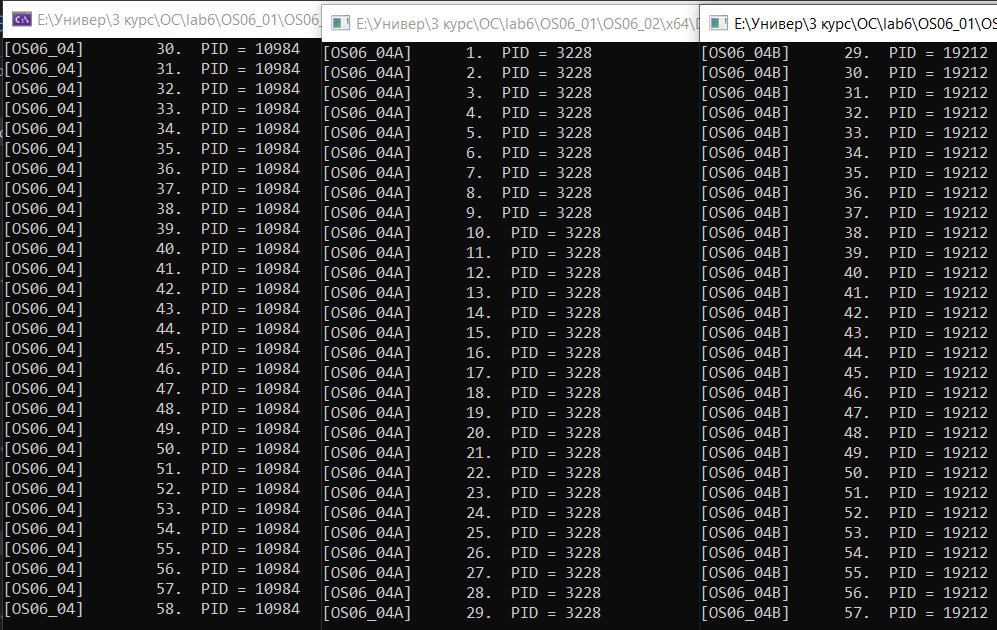
Задание 3

|  |
| --- |
| #include <iostream>  #include <Windows.h>  using namespace std;  PROCESS\_INFORMATION createProcess(LPCWSTR path)  {  STARTUPINFO startupInfo;  PROCESS\_INFORMATION processInfo;  ZeroMemory(&startupInfo, sizeof(STARTUPINFO));  startupInfo.cb = sizeof(STARTUPINFO);  if (CreateProcessW(path, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &startupInfo, &processInfo))  cout << "Process created\n";  else  cout << "[ERROR] CreateProcessW\n";  return processInfo;  }  int main()  {  int pid = GetCurrentProcessId();  const int size = 2;  HANDLE mutex;  HANDLE processes[size];  processes[0] = createProcess(L"..\\x64\\Debug\\OS06\_03A.exe").hProcess;  processes[1] = createProcess(L"..\\x64\\Debug\\OS06\_03B.exe").hProcess;  mutex = CreateMutex(NULL, FALSE, L"OS06\_03");  for (int i = 1; i <= 90; ++i)  {  if (i == 30)  WaitForSingleObject(mutex, INFINITE);  else if (i == 60)  ReleaseMutex(mutex);  printf("[OS06\_03]\t %d. PID = %d\n", i, pid);  Sleep(100);  }  WaitForMultipleObjects(size, processes, TRUE, INFINITE);  for (int i = 0; i < size; i++)  CloseHandle(processes[i]);  CloseHandle(mutex);  cout << '\n';  system("pause");  return 0;  } |



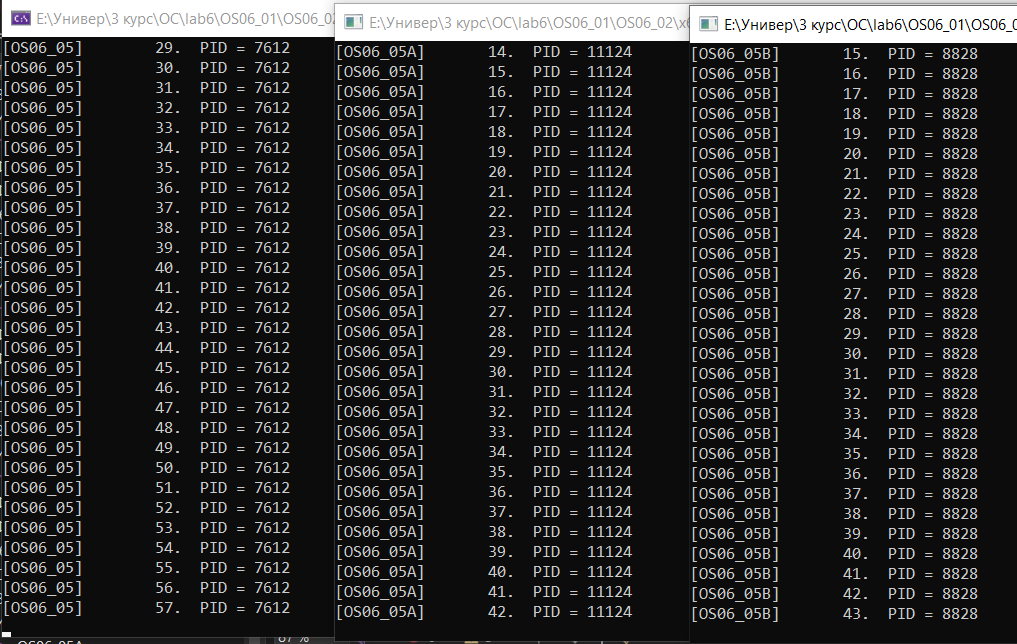
Задание 4

|  |
| --- |
| #include <iostream>  #include <Windows.h>  using namespace std;  PROCESS\_INFORMATION createProcess(LPCWSTR path)  {  STARTUPINFO startupInfo;  PROCESS\_INFORMATION processInfo;  ZeroMemory(&startupInfo, sizeof(STARTUPINFO));  startupInfo.cb = sizeof(STARTUPINFO);  if (CreateProcessW(path, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &startupInfo, &processInfo))  cout << "Process created\n";  else  cout << "[ERROR] CreateProcessW\n";  return processInfo;  }  int main()  {  int pid = GetCurrentProcessId();  const int size = 2;  HANDLE semaphore;  HANDLE processes[size];  processes[0] = createProcess(L"E:\\Универ\\3 курс\\OC\\lab6\\OS06\_01\\OS06\_02\\x64\\Debug\\OS06\_04A.exe").hProcess;  processes[1] = createProcess(L"E:\\Универ\\3 курс\\OC\\lab6\\OS06\_01\\OS06\_02\\x64\\Debug\\OS06\_04B.exe").hProcess;  semaphore = CreateSemaphore(NULL, 2, 2, L"OS06\_04");  for (int i = 1; i <= 90; i++)  {  if (i == 30)  WaitForSingleObject(semaphore, INFINITE);  else if (i == 60)  ReleaseSemaphore(semaphore, 1, NULL);  printf("[OS06\_04]\t %d. PID = %d\n", i, pid);  Sleep(100);  }  WaitForMultipleObjects(size, processes, TRUE, INFINITE);  for (int i = 0; i < size; i++)  CloseHandle(processes[i]);  CloseHandle(semaphore);  cout << '\n';  system("pause");  return 0;  } |



Задание 5

|  |
| --- |
| #include <iostream>  #include <Windows.h>  using namespace std;  PROCESS\_INFORMATION createProcess(LPCWSTR path)  {  STARTUPINFO startupInfo;  PROCESS\_INFORMATION processInfo;  ZeroMemory(&startupInfo, sizeof(STARTUPINFO));  startupInfo.cb = sizeof(STARTUPINFO);  if (CreateProcessW(path, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &startupInfo, &processInfo))  cout << "Process created.\n";  else  cout << "[ERROR]\n";  return processInfo;  }  int main()  {  int pid = GetCurrentProcessId();  const int size = 2;  HANDLE event;  HANDLE processes[size];  processes[0] = createProcess(L"E:\\Универ\\3 курс\\OC\\lab6\\OS06\_01\\OS06\_02\\x64\\Debug\\OS06\_05A.exe").hProcess;  processes[1] = createProcess(L"E:\\Универ\\3 курс\\OC\\lab6\\OS06\_01\\OS06\_02\\x64\\Debug\\OS06\_05B.exe").hProcess;  event = CreateEvent(NULL, FALSE, FALSE, L"OS06\_05");  for (int i = 1; i <= 90; i++)  {  if (i == 15)  SetEvent(event);  printf("[OS06\_05]\t %d. PID = %d\n", i, pid);  Sleep(100);  }  WaitForMultipleObjects(size, processes, TRUE, INFINITE);  for (int i = 0; i < size; i++)  CloseHandle(processes[i]);  CloseHandle(event);  system("pause");  return 0;  } |



Задание 6

|  |
| --- |
| #include <stdio.h>  #include <time.h>  #include <unistd.h>  #include <pthread.h>  #define DELAY\_MS 100  pthread\_mutex\_t mutex;  pthread\_t create\_thread(void\* function, void\* args)  {      pthread\_t thread;      pthread\_create(&thread, NULL, function, args);      return thread;  }  void\* loop(char\* name)  {      int pid = getpid();      struct timespec timespec;      timespec.tv\_sec = DELAY\_MS / 1000;      timespec.tv\_nsec = (DELAY\_MS % 1000) \* 1000000;      for (int i = 1; i <= 90; i++)      {          if (i == 30)              pthread\_mutex\_lock(&mutex);          else if (i == 60)              pthread\_mutex\_unlock(&mutex);          printf("  \t  %d.  PID^^ = %d\n", name, i, pid);          nanosleep(&timespec, &timespec);      }      printf("\nmeow\_meow\_meow  %s finished<3  meow\_meow\_meow n\n", name);      pthread\_exit("done");  }  int main(int argc, char\* argv[])  {      int pid = getpid();      int size = 2;      char\* names[] = { "A", "B" };      pthread\_t threads[size];      struct timespec timespec;      timespec.tv\_sec = DELAY\_MS / 1000;      timespec.tv\_nsec = (DELAY\_MS % 1000) \* 1000000;      pthread\_mutex\_init(&mutex, NULL);      for (int i = 0; i < size; i++)          threads[i] = create\_thread(loop, (void\*)names[i]);      for (int i = 1; i <= 90; i++)      {          if (i == 30)              pthread\_mutex\_lock(&mutex);          else if (i == 60)              pthread\_mutex\_unlock(&mutex);          printf("[MAIN]\t  %d.  PID = %d\n", i, pid);          nanosleep(&timespec, &timespec);      }      printf("\n<3<3<3<3<3 MAIN tread finished meow  <3<3<3<3<3\n");      for (int i = 0; i < size; i++)          pthread\_join(threads[i], NULL);      pthread\_mutex\_destroy(&mutex);      return 0;  } |

