Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Операционные системы

Студент: Трубач Д.С.

ФИТ 3 курс 5 группа

Преподаватель: Савельева М.Г.

Минск 2023

**Лабораторная работа №7**

Задание 1. Разработайте приложение OS07\_01.

Приложение OS07\_01 выводит на консоль текущую локальную дату и время в формате *дд.мм.ггг чч:мин:сек*.

|  |  |
| --- | --- |
| Код программы OS07\_01 | #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <time.h>  int main() {  time\_t rawtime;  struct tm info;  char buffer[80];  time(&rawtime);  localtime\_s(&info, &rawtime);  strftime(buffer, sizeof(buffer), "%d.%m.%Y %H:%M:%S", &info);  printf("%s\n", buffer);  return 0;  } |
| Скриншот запуска |  |

Задание 2. Разработайте приложение OS07\_02, выполняющее бесконечный цикл.

В теле цикла подсчитывается количество итераций.

Выведите на консоль значения счетчика итераций через 5 сек. и 10 сек.

Корректно завершите работу цикла и приложения через 15 сек., выведите итоговое значение счетчика итераций.

|  |  |
| --- | --- |
| Код программы OS07\_02 | #include <Windows.h>  #include <iostream>  #include <ctime>  using namespace std;  int main()  {  clock\_t start = clock();  int k = 0;  bool flag5 = true, flag10 = true;  while (true)  {  k++;  if ((clock() - start) / CLOCKS\_PER\_SEC == 5 && flag5) {  cout << "Iterations after 5s: " << k << '\n';  flag5 = false;  }  if ((clock() - start) / CLOCKS\_PER\_SEC == 10 && flag10) {  cout << "Iterations after 10s: " << k << '\n';  flag10 = false;  }  if ((clock() - start) / CLOCKS\_PER\_SEC == 15) {  cout << "Iterations after 15s: " << k << '\n';  break;  }  }  return 0;  } |
| Скриншот запуска |  |

Задание 3. Разработайте приложение OS07\_03, выполняющее бесконечный цикл.

В теле цикла с задержкой подсчитывается количество итераций.

Выведите на консоль значения счетчика итераций каждые 3 сек.

Корректно завершите работу цикла и приложения через 15 сек., выведите итоговое значение счетчика итераций.

|  |  |
| --- | --- |
| Код программы OS07\_03 | #include <iostream>  #include <Windows.h>  using namespace std;  HANDLE createThread(LPTHREAD\_START\_ROUTINE func);  void close\_timer\_watcher();  void print\_timer\_watcher();  HANDLE print\_timer = CreateWaitableTimer(NULL, FALSE, L"os07\_04v2\_print");  HANDLE close\_timer = CreateWaitableTimer(NULL, FALSE, L"os07\_04v2\_close");  bool print = false, iterate = true;  int main()  {  LPTHREAD\_START\_ROUTINE funcs[] = { (LPTHREAD\_START\_ROUTINE)close\_timer\_watcher, (LPTHREAD\_START\_ROUTINE)print\_timer\_watcher };  const int size = sizeof(funcs) / sizeof(LPTHREAD\_START\_ROUTINE);  HANDLE threads[size];  LARGE\_INTEGER close\_timeout;  close\_timeout.QuadPart = -150000000LL;  LARGE\_INTEGER print\_timeout;  print\_timeout.QuadPart = -30000000LL;  clock\_t current\_time, start\_time = clock();  unsigned long int i;  for (i = 0; i < size; i++)  threads[i] = createThread(funcs[i]);  SetWaitableTimer(print\_timer, &print\_timeout, 3000, NULL, NULL, FALSE);  SetWaitableTimer(close\_timer, &close\_timeout, 0, NULL, NULL, 0);  for (i = 0; iterate; i++)  {  current\_time = ((clock() - start\_time) / CLOCKS\_PER\_SEC);  if (print)  {  cout << "Time: " << current\_time << "\tIterations: " << i << '\n';  print = false;  }  }  cout << "Final: " << i << '\n';  for (i = 0; i < size; i++)  CloseHandle(threads[i]);  system("pause");  return 0;  }  HANDLE createThread(LPTHREAD\_START\_ROUTINE func)  {  DWORD thread\_id = NULL;  HANDLE thread = CreateThread(NULL, 0, func, NULL, 0, &thread\_id);  if (thread == NULL) {  throw "Error creating child thread";  }  return thread;  }  void close\_timer\_watcher()  {  WaitForSingleObject(close\_timer, INFINITE);  iterate = false;  CloseHandle(close\_timer);  }  void print\_timer\_watcher()  {  for (;;)  {  WaitForSingleObject(print\_timer, INFINITE);  print = true;  }  CloseHandle(print\_timer);  } |
| Скриншот запуска |  |

Задание 4. Разработайте приложение OS07\_04, запускающее два одинаковых дочерних процесса OS07\_04\_X.

Процессы OS07\_04\_X вычисляют и выводят на консоль (каждый в свою) пронумерованный ряд простых положительных чисел (простое число делится нацело только на себя и 1).

Первый дочерний процесс должен выполняться 1 минуту и корректно завершаться.

Первый дочерний процесс должен выполняться 2 минуты и корректно завершаться.

Приложение OS08\_04 завершается после завершения дочерних процессов.

|  |  |
| --- | --- |
| Код программы OS07\_04 | #include <Windows.h>  #include <iostream>  #include <ctime>  using namespace std;  int main() {  DWORD pid = GetCurrentProcessId();  cout << "Main process ID: " << pid << '\n';  long long it = -60 \* 10000000;  HANDLE htimer = CreateWaitableTimer(NULL, FALSE, L"os07\_04");  if (!SetWaitableTimer(htimer, (LARGE\_INTEGER\*)&it, 60000, NULL, NULL, FALSE))  throw "SetWaitableTimer Error";  LPCWSTR path = L"E:\\3course\\5sem\\OS\\Lab7\\Debug\\OS07\_04x.exe";  PROCESS\_INFORMATION pi1, pi2;  clock\_t start = clock();  pi1.dwThreadId = 1; pi2.dwThreadId = 2;  {  STARTUPINFO si; ZeroMemory(&si, sizeof(STARTUPINFO)); si.cb = sizeof(STARTUPINFO);  CreateProcess(path, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi1) ?  cout << "Process OS07\_04x #1 was created\n" : cout << "Process OS07\_04x #1 wasn't created\n";  }  {  STARTUPINFO si; ZeroMemory(&si, sizeof(STARTUPINFO)); si.cb = sizeof(STARTUPINFO);  CreateProcess(path, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi2) ?  cout << "Process OS07\_04x #2 was created\n" : cout << "Process OS07\_04x #2 wasn't created\n";  }  WaitForSingleObject(pi1.hProcess, INFINITE);  WaitForSingleObject(pi2.hProcess, INFINITE);  CloseHandle(htimer);  cout << "Elapsed time = " << clock() - start << " clocks\n";  system("pause");  return 0;  } |
| Код программы OS07\_04x | #include <Windows.h>  #include <iostream>  #include <ctime>  #include <TlHelp32.h>  #include <stdio.h>  using namespace std;  bool stop = false;  int simple(int n)  {  for (int i = 2; i <= n / 2; i++)  if ((n % i) == 0)  return 0;  return 1;  }  DWORD ChildThread()  {  DWORD tid = GetCurrentThreadId();  for (long long i = 2; ; i++)  {  if (stop) break;  if (simple(i)) cout << i << '\n';  }  return 0;  }  int main()  {  clock\_t start = clock();  HANDLE timerHandle = OpenWaitableTimer(TIMER\_ALL\_ACCESS, FALSE, L"os08\_04");  DWORD childId = NULL;  HANDLE childHandle = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildThread, NULL, 0, &childId);  WaitForSingleObject(timerHandle, INFINITE);  stop = true;  WaitForSingleObject(childHandle, INFINITE);  CloseHandle(childHandle);  cout << "Elapsed time = " << clock() - start << " clocks\n";  system("pause");  return 0;  } |
| Скриншот запуска |  |

Задание 5. Разработайте приложение OS07\_05.

Приложение OS07\_05 выводит на консоль текущую локальную дату и время в формате *дд.мм.ггг чч:мин:сек*.

|  |  |
| --- | --- |
| Код программы OS07\_05 | #include <stdio.h>  #include <time.h>  int main() {  time\_t rawtime;  struct tm \*info;  time(&rawtime);  info = localtime(&rawtime);  printf("%02d.%02d.%04d %02d:%02d:%02d\n",  info->tm\_mday, info->tm\_mon + 1, info->tm\_year + 1900,  info->tm\_hour, info->tm\_min, info->tm\_sec);  return 0;  } |
| Скриншот запуска | E:\3course\5sem\OS\Lab7\Images\linux.png |

Задание 6. Разработайте приложение OS07\_06, выполняющее бесконечный цикл.

В теле цикла подсчитывается количество итераций.

Выведите на консоль значения счетчика итераций через 2 сек. процессорного времени и корректно завершите цикл.

Приложение OS07\_06 должно выполнять замер реального затраченного на работу цикла времени и выводить его значения на консоль.

|  |  |
| --- | --- |
| Код программы OS07\_06 | #include <stdio.h>  #include <time.h>  int main() {  clock\_t start\_time, current\_time;  int counter = 0;    start\_time = clock();  while (1) {  counter++;  current\_time = clock();  if ((double)(current\_time - start\_time) / CLOCKS\_PER\_SEC >= 2.0) {  printf("Iterations: %d\n", counter);  printf("Load time: %d\n", current\_time - start\_time);  start\_time = current\_time;  break;  }  }  return 0;  } |
| Скриншот запуска | E:\3course\5sem\OS\Lab7\Images\linux.png |