Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №3**

Выполнил:

Студент 3 курса 4 группы ФИТ

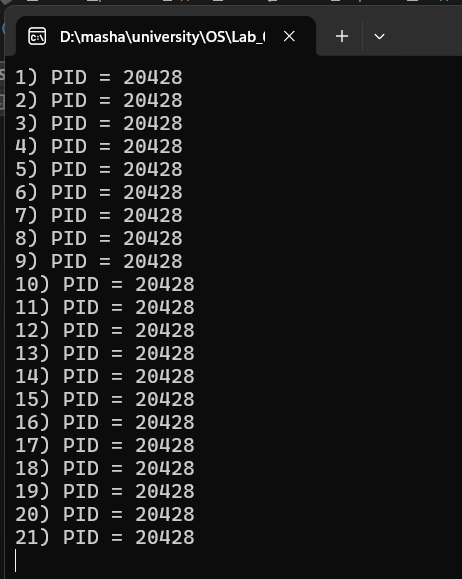
Сосновец Мария Игоревна

2024 г.

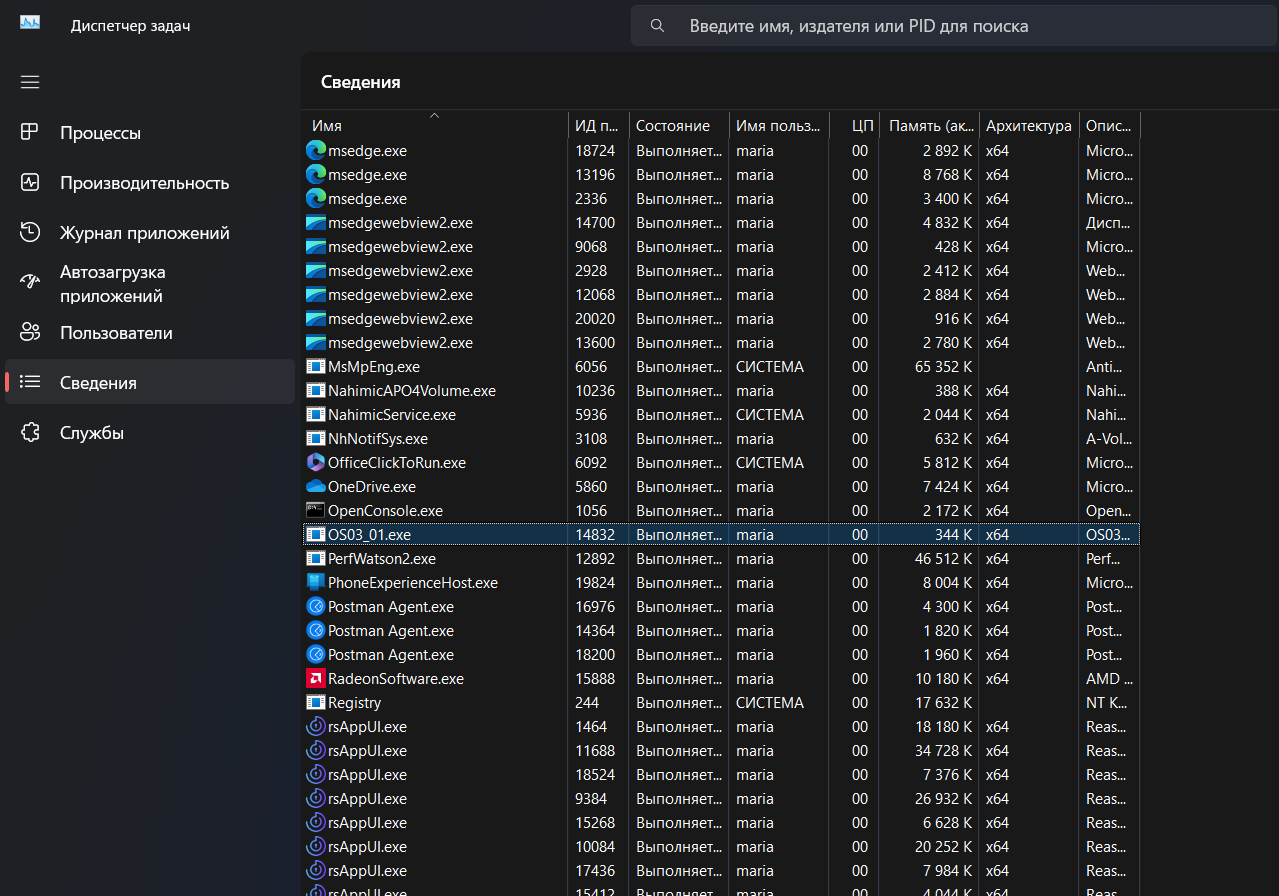
**Задание 01**

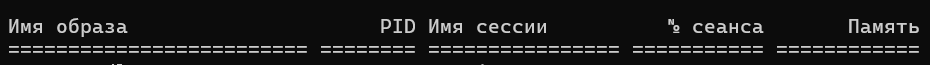
1. Разработайте консольное Windows-приложение OS03\_01 на языке С++, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификатора процесса.

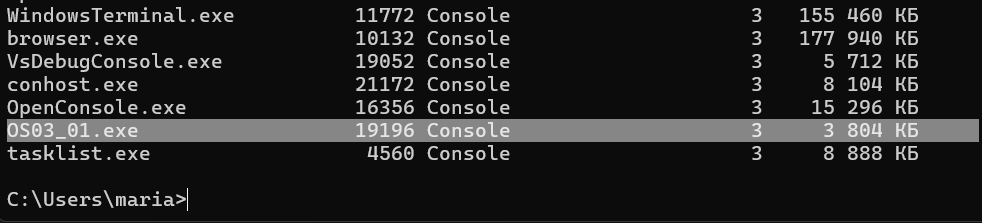
|  |
| --- |
| #include <iostream>;  #include <Windows.h>  using namespace std;  void main(){  for (short i = 1; i <= 1000; ++i) {  cout << i << ") PID = " << GetCurrentProcessId() << "\n";  Sleep(1000);  }  } |

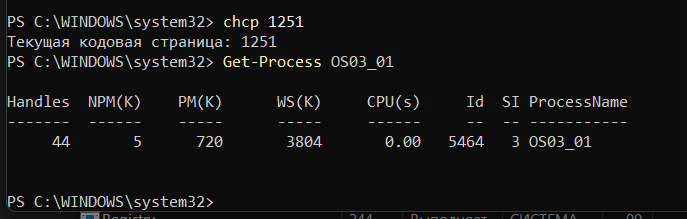


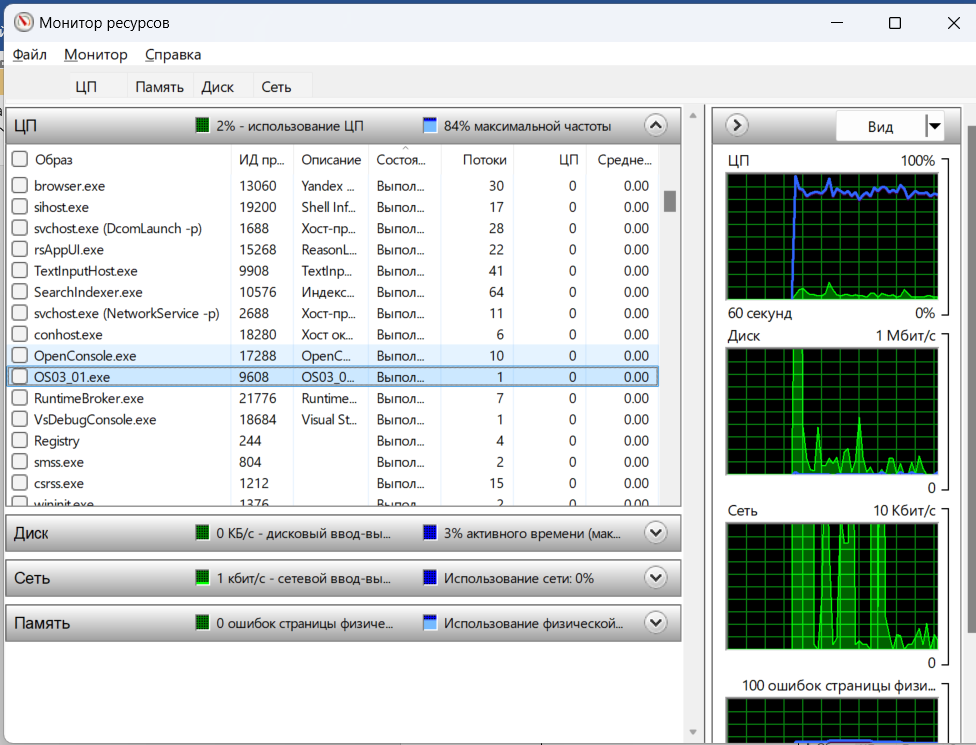
1. Продемонстрируйте информацию о процессе OS03\_01 с помощью утилит Task Manager, tasklist, PowerShell ISE и Performance Monitor.











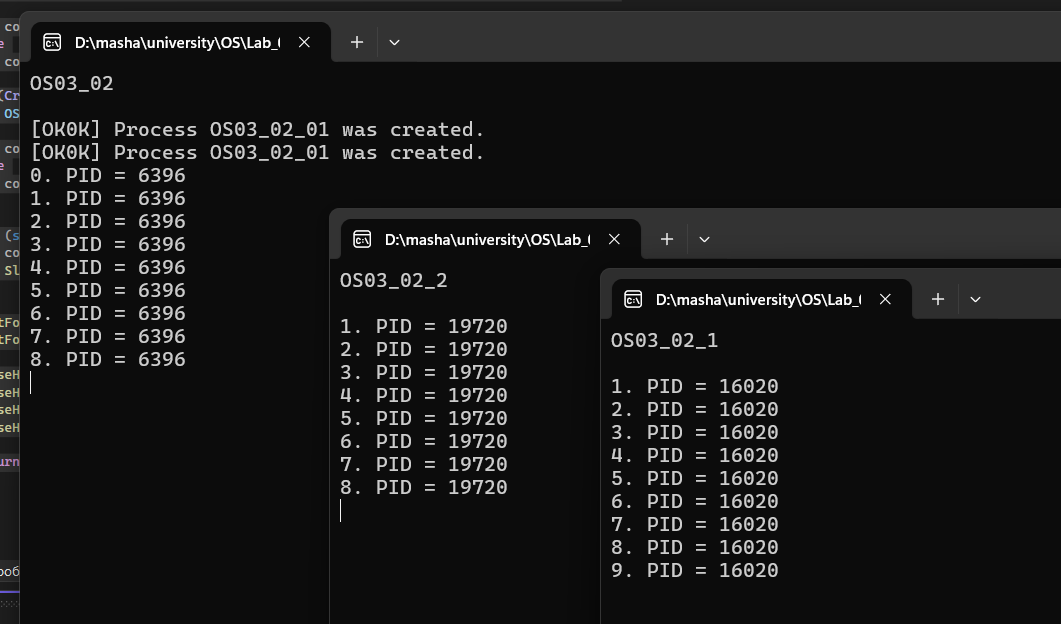
**Задание 02**

1. Разработайте консольное Windows-приложение OS03\_02 на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

|  |
| --- |
| #include <iostream>;  #include <Windows.h>;  using namespace std;  void main() {  for (short i = 0; i <= 100; ++i) {  cout << i + 1 << ") PID = " << GetCurrentProcessId() << "\n";  Sleep(1000);  }  } |

1. Приложение OS03\_02 должно создавать два дочерних процесса OS03\_02\_1 и OS03\_02\_2.

|  |
| --- |
| #include <iostream>;  #include <Windows.h>;  using namespace std;  int main() {  cout << "OS03\_02\n\n";  LPCWSTR OS03\_02\_1 = L"D:\\masha\\university\\OS\\Lab\_03\\OS03\_01\\x64\\Debug\\OS03\_02\_1.exe";  LPCWSTR OS03\_02\_2 = L"D:\\masha\\university\\OS\\Lab\_03\\OS03\_01\\x64\\Debug\\OS03\_02\_2.exe";  STARTUPINFO si1;  STARTUPINFO si2;  PROCESS\_INFORMATION pi1;  PROCESS\_INFORMATION pi2;  ZeroMemory(&si1, sizeof(STARTUPINFO));  ZeroMemory(&si2, sizeof(STARTUPINFO));  si1.cb = sizeof(STARTUPINFO);  si1.dwFlags = STARTF\_USEFILLATTRIBUTE;  si1.dwFillAttribute = (DWORD)FOREGROUND\_RED;  si1.lpReserved = NULL;  si2.cb = sizeof(STARTUPINFO);  si2.dwFlags = STARTF\_USEFILLATTRIBUTE;  si2.dwFillAttribute = (DWORD)FOREGROUND\_BLUE;  si1.lpReserved = NULL;  if (CreateProcess(  OS03\_02\_1, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si1, &pi1  ))  cout << "[OK0K] Process OS03\_02\_01 was created. \n";  else  cout << "[ERROR] Process OS03\_02\_01 was not created. \n";    if (CreateProcess(  OS03\_02\_2, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si2, &pi2  ))  cout << "[OK0K] Process OS03\_02\_01 was created. \n";  else  cout << "[ERROR] Process OS03\_02\_01 was not created. \n";  for (short i = 0; i <= 100; ++i) {  cout << i << ". PID = " << GetCurrentProcessId() << "\n";  Sleep(1000);  }  WaitForSingleObject(pi1.hProcess, INFINITE);  WaitForSingleObject(pi2.hProcess, INFINITE);  CloseHandle(pi1.hThread);  CloseHandle(pi2.hThread);  CloseHandle(pi1.hProcess);  CloseHandle(pi2.hProcess);  return 0;  } |



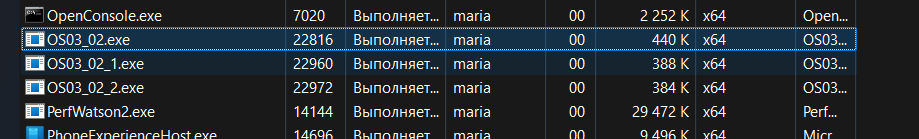
1. Процесс OS03\_02\_1 - консольное Windows-приложение, выполняющее цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

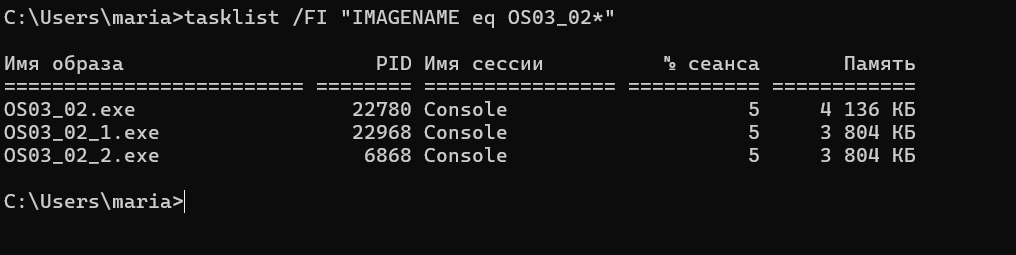
|  |
| --- |
| #include <iostream>  #include <Windows.h>  using namespace std;  void main()  {  cout << "OS03\_02\_1\n\n";  for (short i = 1; i <= 50; ++i)  {  cout << i << ". PID = " << GetCurrentProcessId() << "\n";  Sleep(1000);  }  system("pause");  } |

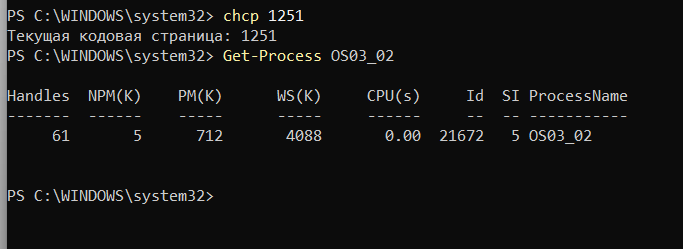
1. Процесс OS03\_02\_2 - консольное Windows-приложение выполняющее цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

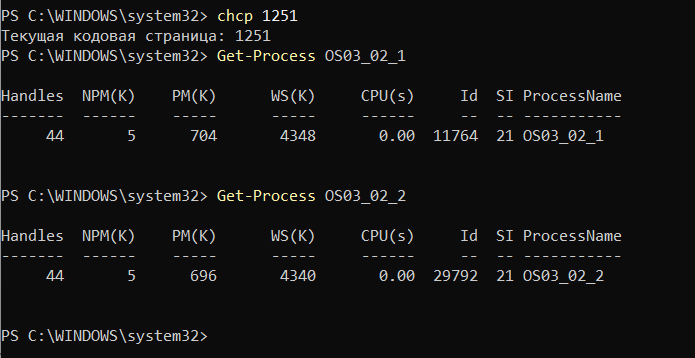
|  |
| --- |
| #include <iostream>  #include <Windows.h>  using namespace std;  void main()  {  cout << "OS03\_02\_2\n\n";  for (short i = 1; i <= 125; ++i)  {  cout << i << ". PID = " << GetCurrentProcessId() << "\n";  Sleep(1000);  }  system("pause");  } |

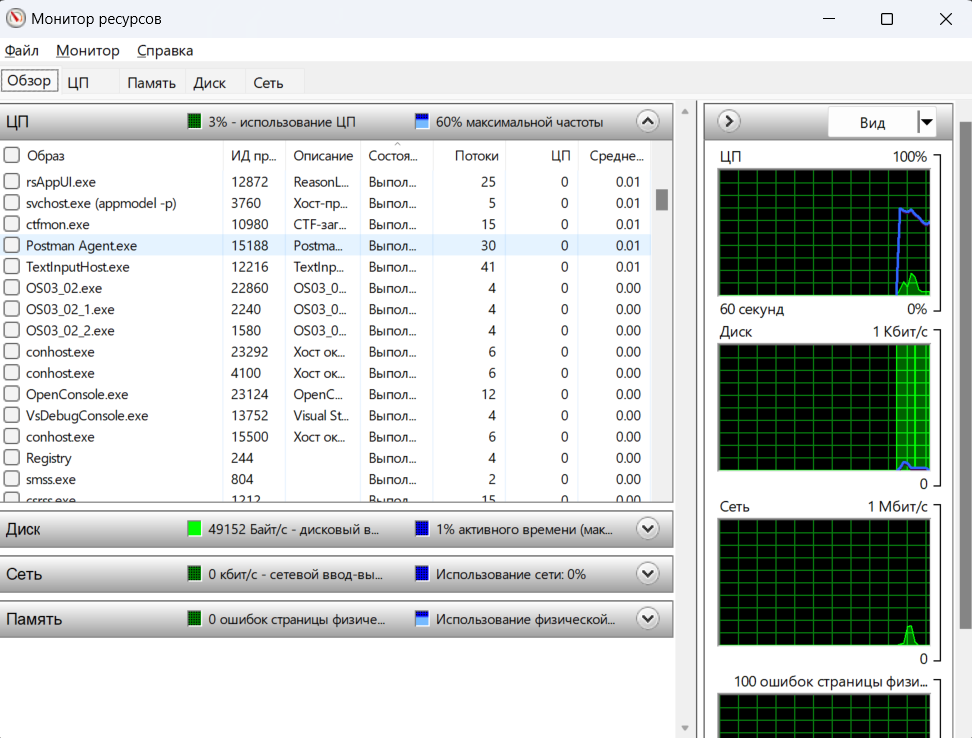
7. Продемонстрируйте информацию о процессах OS03\_02, OS03\_02\_1 и OS03\_02\_2 с помощью утилит Task Manager, tasklist, PowerShell ISE и Performance Monitor.









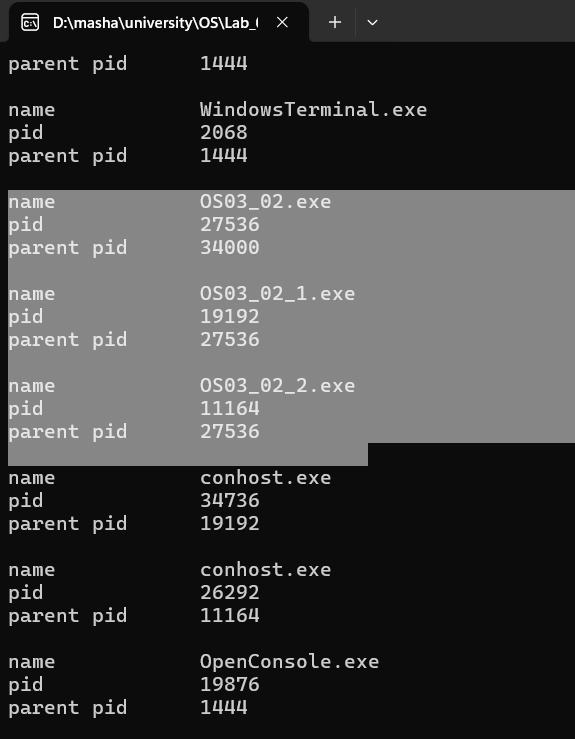


**Задание 03.**

1. Разработайте консольное Windows-приложение OS03\_03 на языке С++, выводящее на консоль перечень выполняющихся процессов в данный момент в OS.

|  |
| --- |
| #include <iostream>  #include <Windows.h>  #include "TlHelp32.h"  #include <iomanip>  using namespace std;  int main()  {  DWORD pid = GetCurrentProcessId();  HANDLE snapshot = CreateToolhelp32Snapshot(TH32CS\_SNAPALL, 0);  PROCESSENTRY32 peProcessEntry;  peProcessEntry.dwSize = sizeof(PROCESSENTRY32);  wcout << L"Current PID: " << pid << endl;  try  {  if (!Process32First(snapshot, &peProcessEntry))  throw L"Process32First";  do  {  if (peProcessEntry.th32ProcessID == pid)  wcout << " ! CURRENTLY RUNNING PROCESS !\n";  wcout << L"Name\t\t" << peProcessEntry.szExeFile << "\n";  wcout << L"PID\t\t" << peProcessEntry.th32ProcessID << "\n";  wcout << L"Parent PID\t" << peProcessEntry.th32ParentProcessID << "\n";  } while (Process32Next(snapshot, &peProcessEntry));  }  catch (char\* errMessage)  {  wcout << L"[ERROR] " << errMessage;  }  system("pause");  return 0;  } |

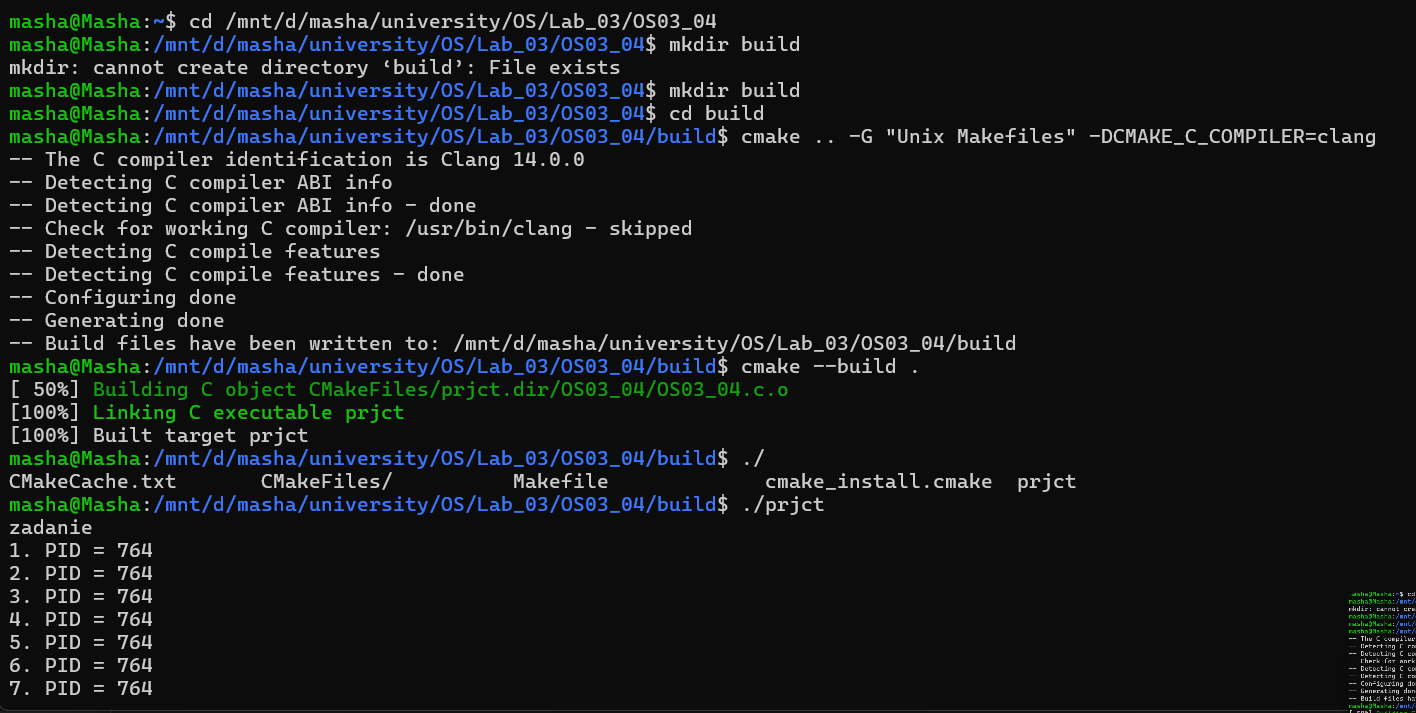
1. Запустите приложение OS03\_02 и продемонстрируйте с помощью приложения OS03\_03 в перечне процессов OS03\_02, OS03\_02\_1, OS03\_02\_2 и OS03\_03.



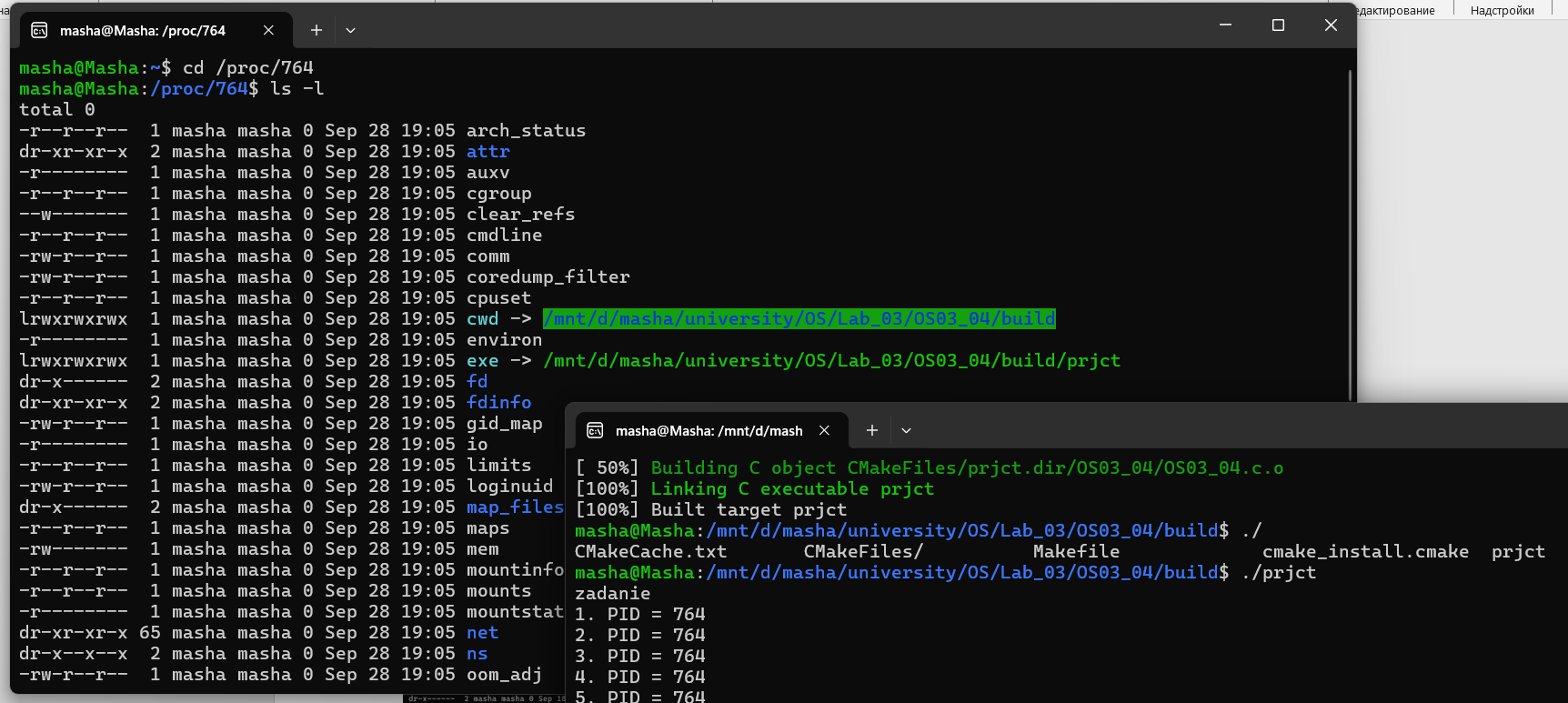
**Задание 04**

1. Разработайте консольное Linux-приложение OS03\_04 на языке С, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификатора процесса.

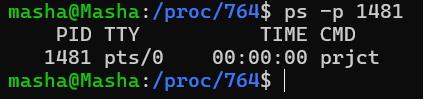
|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #ifdef \_WIN32  #include <Windows.h>  #else  #include <unistd.h> // Для функции sleep()  #include <sys/types.h> // Для типа pid\_t  #endif  int main() {  #ifdef \_WIN32  DWORD pid = GetCurrentProcessId(); // Получаем PID на Windows  Sleep(1000); // Задержка на 1000 миллисекунд (1 секунда)  #else  pid\_t pid = getpid(); // Получаем PID на Linux  // Задержка на 1 секунду  #endif  printf("zadanie\n");  for (short i = 1; i <= 10000; ++i) {  printf("%d. PID = %d\n", i, pid);  #ifdef \_WIN32  Sleep(1000); // Задержка на 1000 миллисекунд (1 секунда) на Windows  #else  sleep(1); // Задержка на 1 секунду на Linux  #endif  }  exit(0);  } |



1. Продемонстрируйте информацию о процессе OS03\_04 с помощью файловой системы /proc.



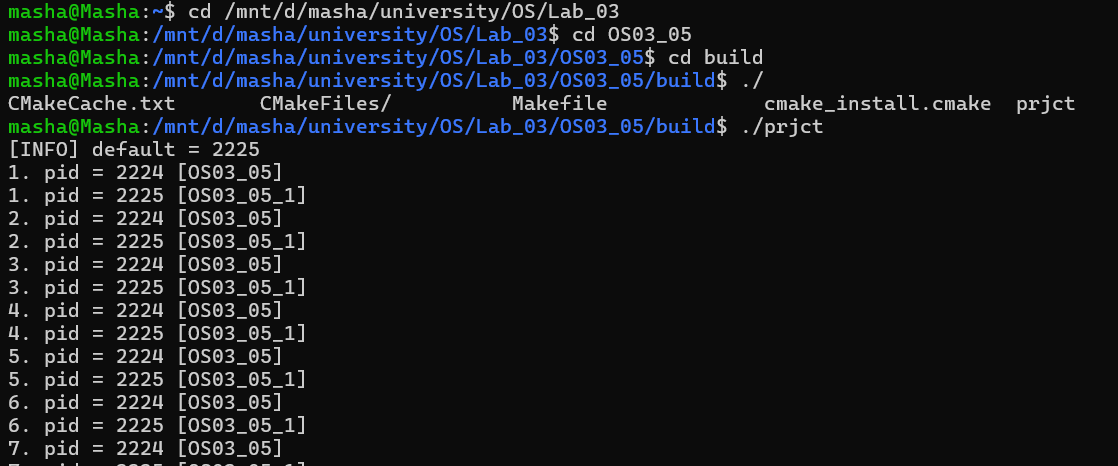
1. Продемонстрируйте информацию о процессе OS03\_04 с помощью утилиты ps.



**Задание 05**

1. Разработайте консольное Linux-приложение OS03\_05 на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <errno.h>  #ifdef \_WIN32  #include <windows.h>  #define getpid() GetCurrentProcessId()  #define sleep(seconds) Sleep((seconds) \* 1000)  #define pid\_t int  #else  #include <unistd.h>  #include <sys/types.h>  #include <sys/wait.h>  #endif  void cycle(int counter, char\* message)  {  for (int i = 1; i <= counter; ++i)  {  printf("%d. PID = %d [%s]\n", i, getpid(), message);  sleep(2);  }  }  int main()  {  #ifndef \_WIN32    pid\_t pid;  switch (pid = fork())  {  case -1:  perror("[ERROR] Fork() returned -1");  exit(-1);  case 0:  cycle(50, "OS03\_05\_1");  exit(0);  default:  printf("[INFO] default = %d\n", pid);  cycle(100, "OS03\_05");  wait(0);  }  #else  printf("[INFO] Running in single process mode for Windows.\n");  cycle(100, "OS03\_05");  #endif  return 0;  } |

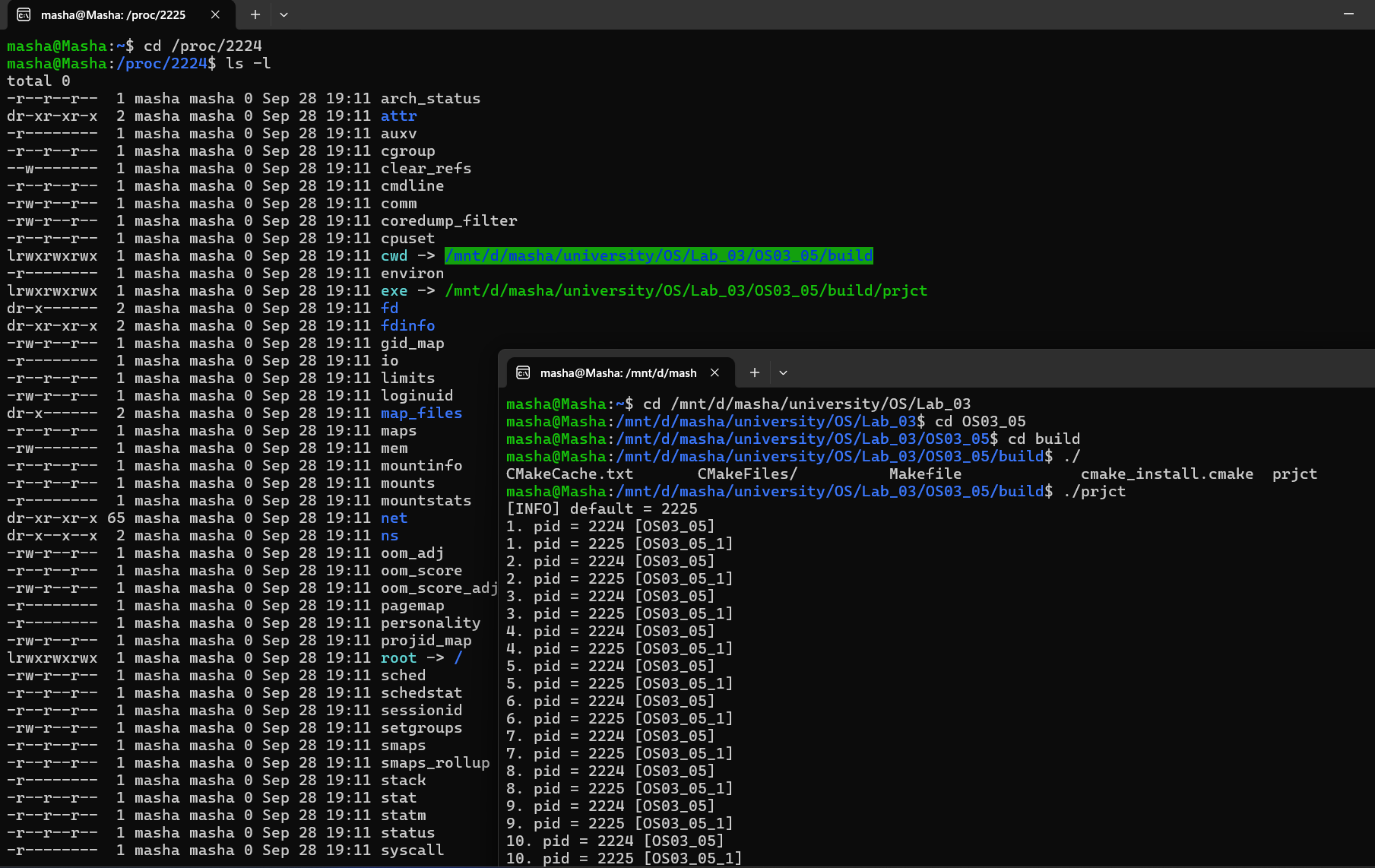


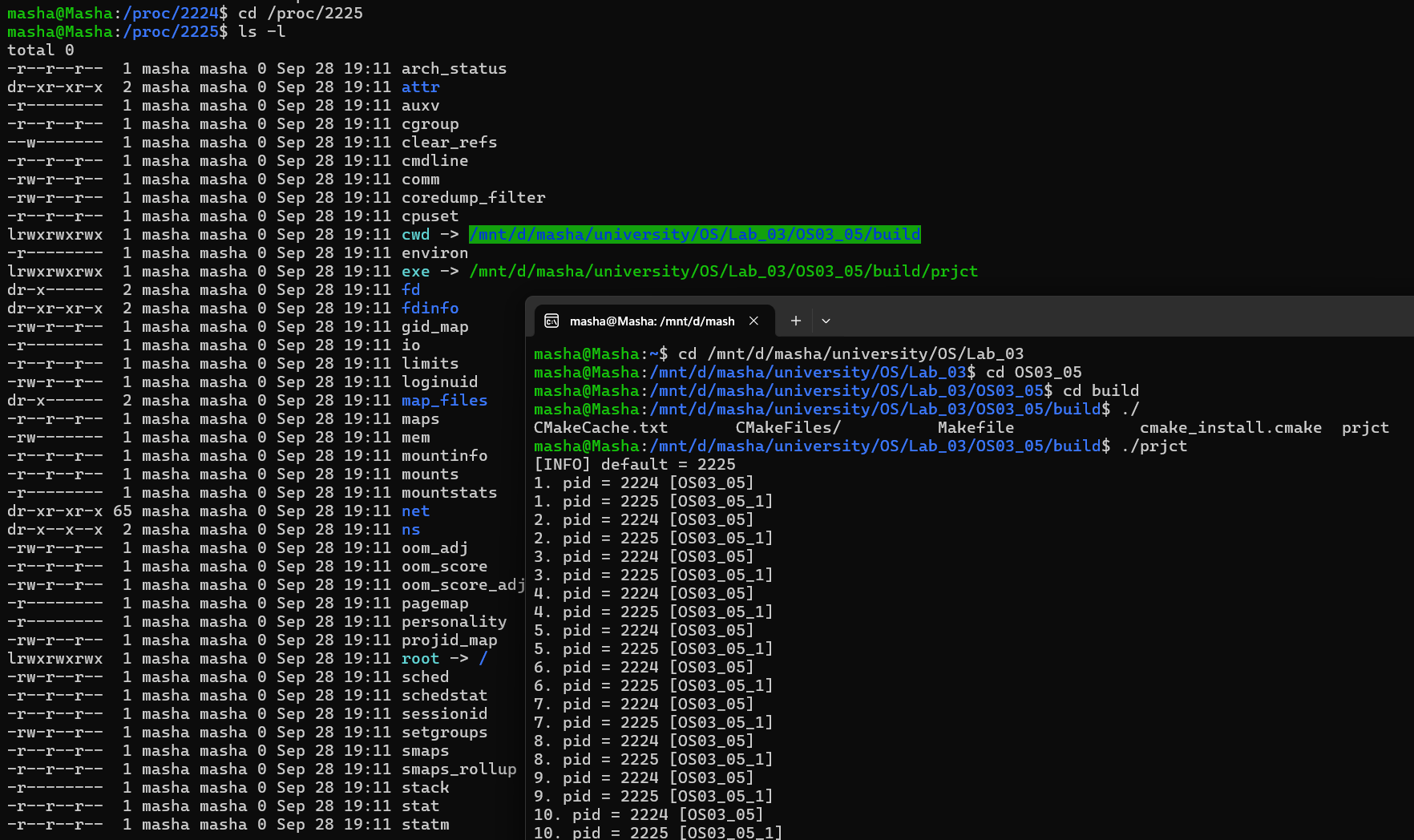
14. Приложение OS03\_05 должно создавать один дочерний процесс OS03\_05\_1 с помощью системного вызова fork. Процесс OS03\_05\_1 в этом случае не является отдельным модулем, а встроен (fork) в программный модуль OS03\_05.

15. Процесс OS03\_05\_1 - консольное Linux-приложение, выполняющее цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

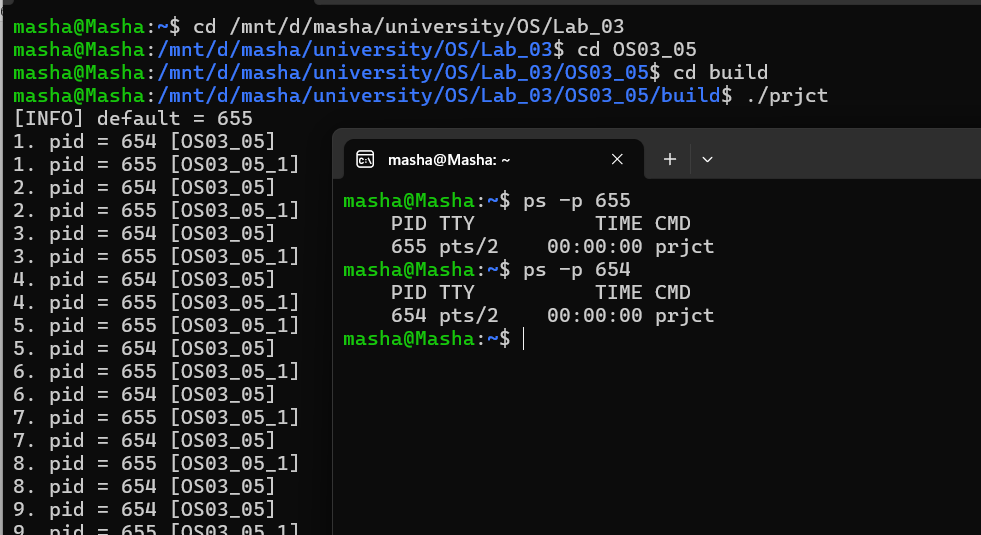
|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <errno.h>  #ifdef \_WIN32  #include <windows.h>  #else  #include <unistd.h>  #include <sys/types.h>  #include <sys/wait.h>  #endif  int main()  {  for (int i = 1; i <= 50; ++i)  {  printf("%d. PID = %d [OS03\_05\_1]\n", i, getpid());  #ifdef \_WIN32  Sleep(1000);  sleep(1);  #endif  }  exit(0);  } |

16. Продемонстрируйте информацию о процессах OS03\_05 и OS03\_05\_1 с помощью файловой системы /proc.





17. Продемонстрируйте информацию о процессах OS03\_05 и OS03\_05\_1 с помощью утилиты ps.

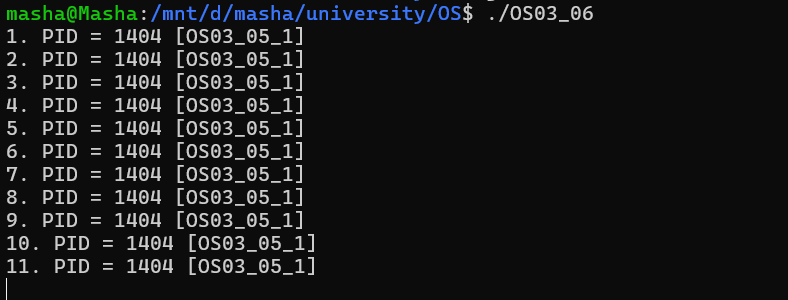


**Задание 06**

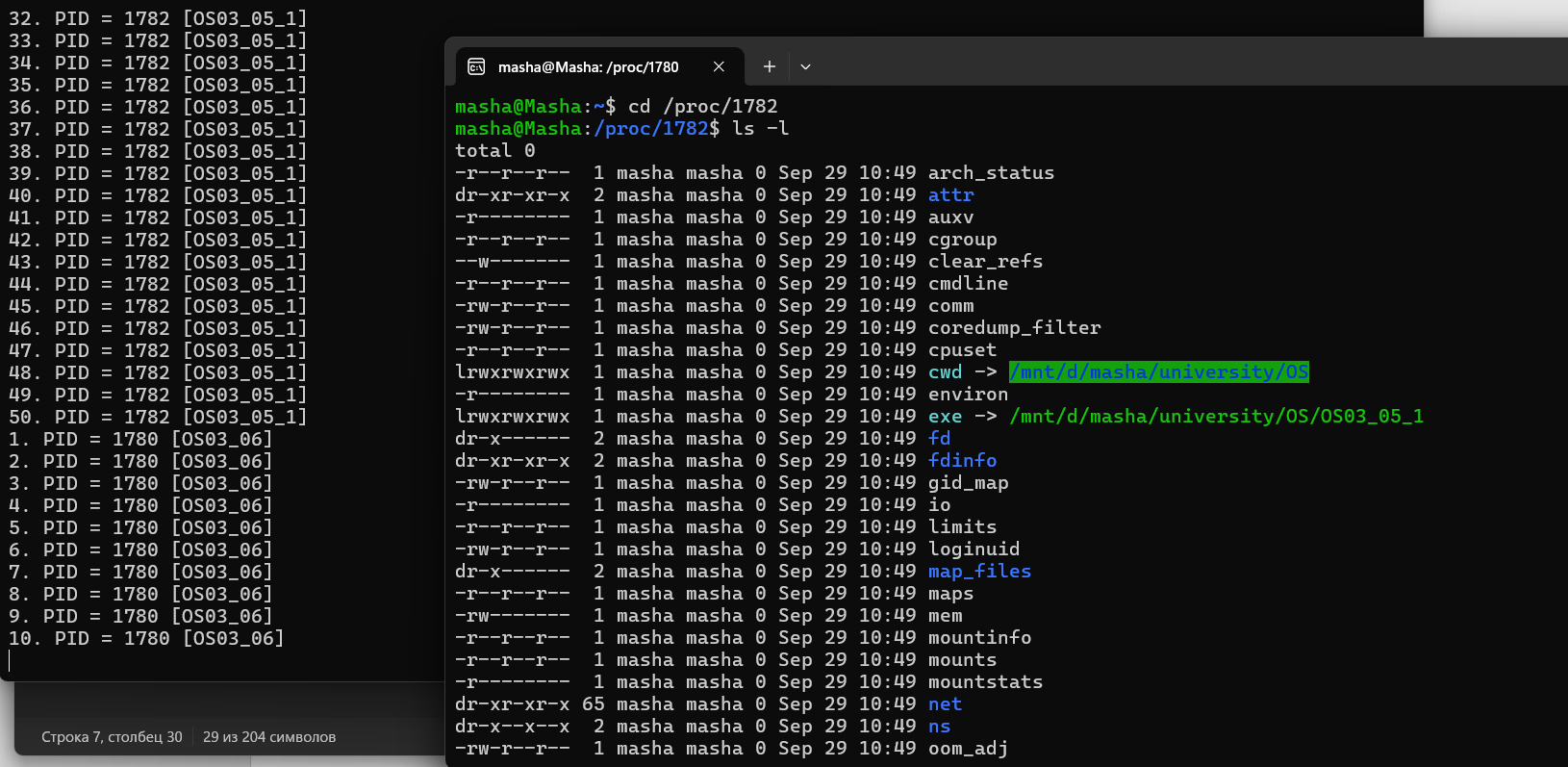
18. Разработайте консольное Linux-приложение OS03\_06 на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

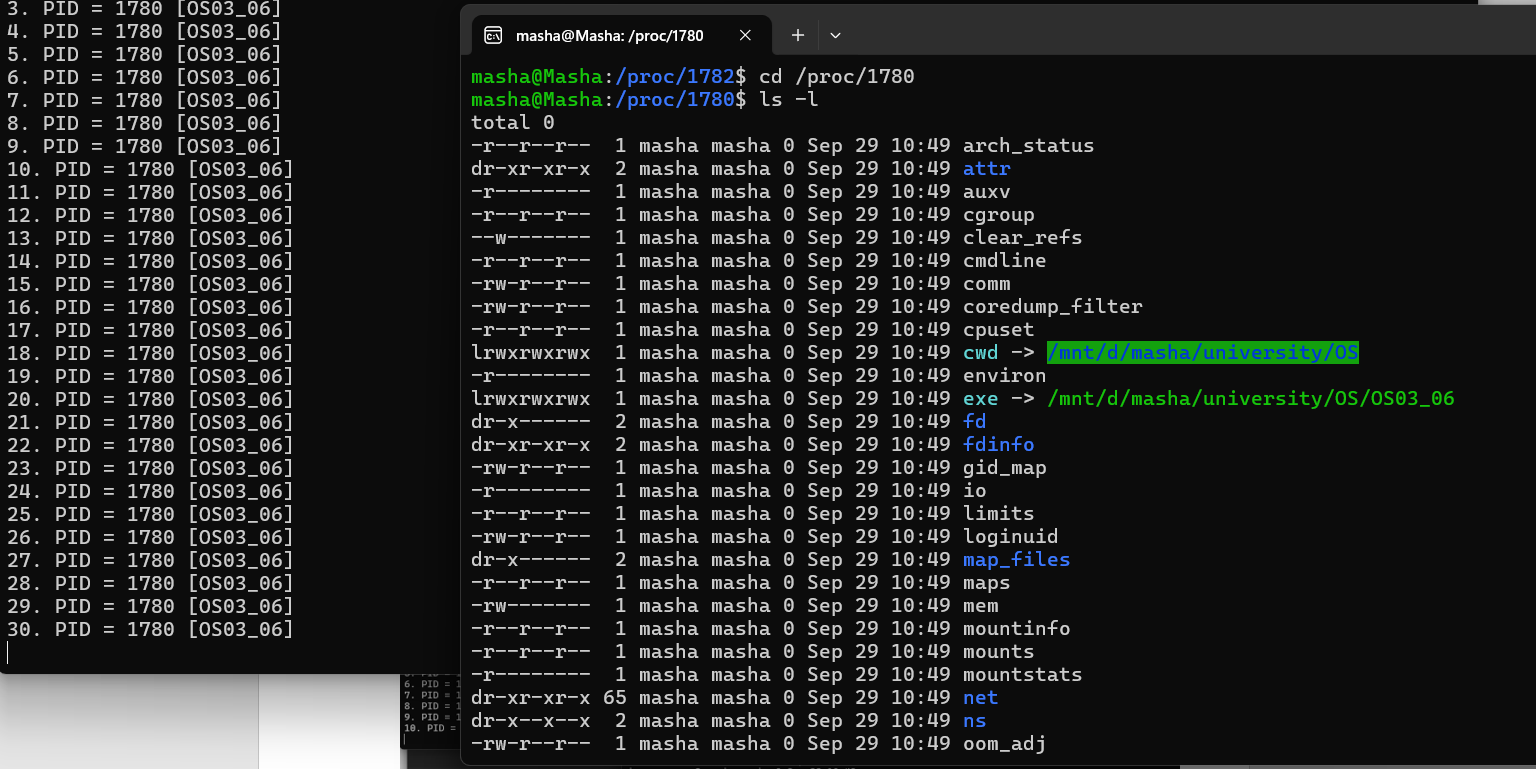
|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <errno.h>  #include <unistd.h>  #include <sys/types.h>  #include <sys/wait.h>  int main()  {  system("./OS03\_05\_1");  for (int i = 1; i <= 100; i++)  {  printf("%d. PID = %d [OS03\_06]\n", i, getpid());  sleep(1);  }  exit(0);  } |

19. Приложение OS03\_06 должно создавать один дочерний процесс OS03\_05\_1 (отдельный модуль) с помощью системного вызова system.

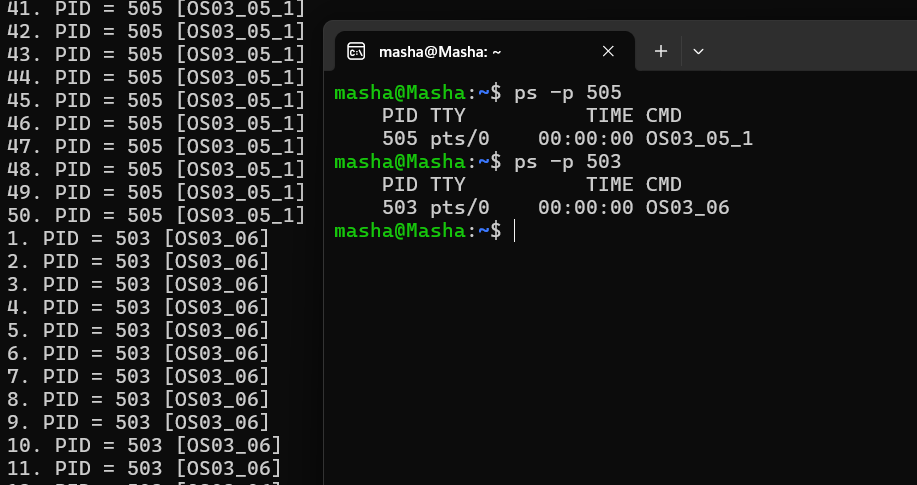


20. Продемонстрируйте информацию о процессах OS03\_06 и OS03\_05\_1 с помощью файловой системы /proc.





21. Продемонстрируйте информацию о процессах OS03\_06 и OS03\_05-1 с помощью утилиты ps.

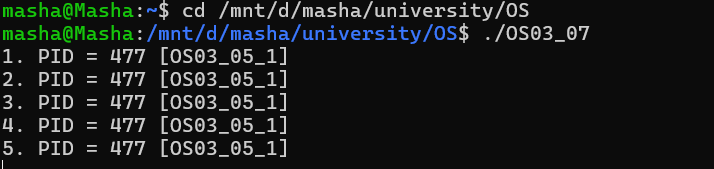


**Задание 07**

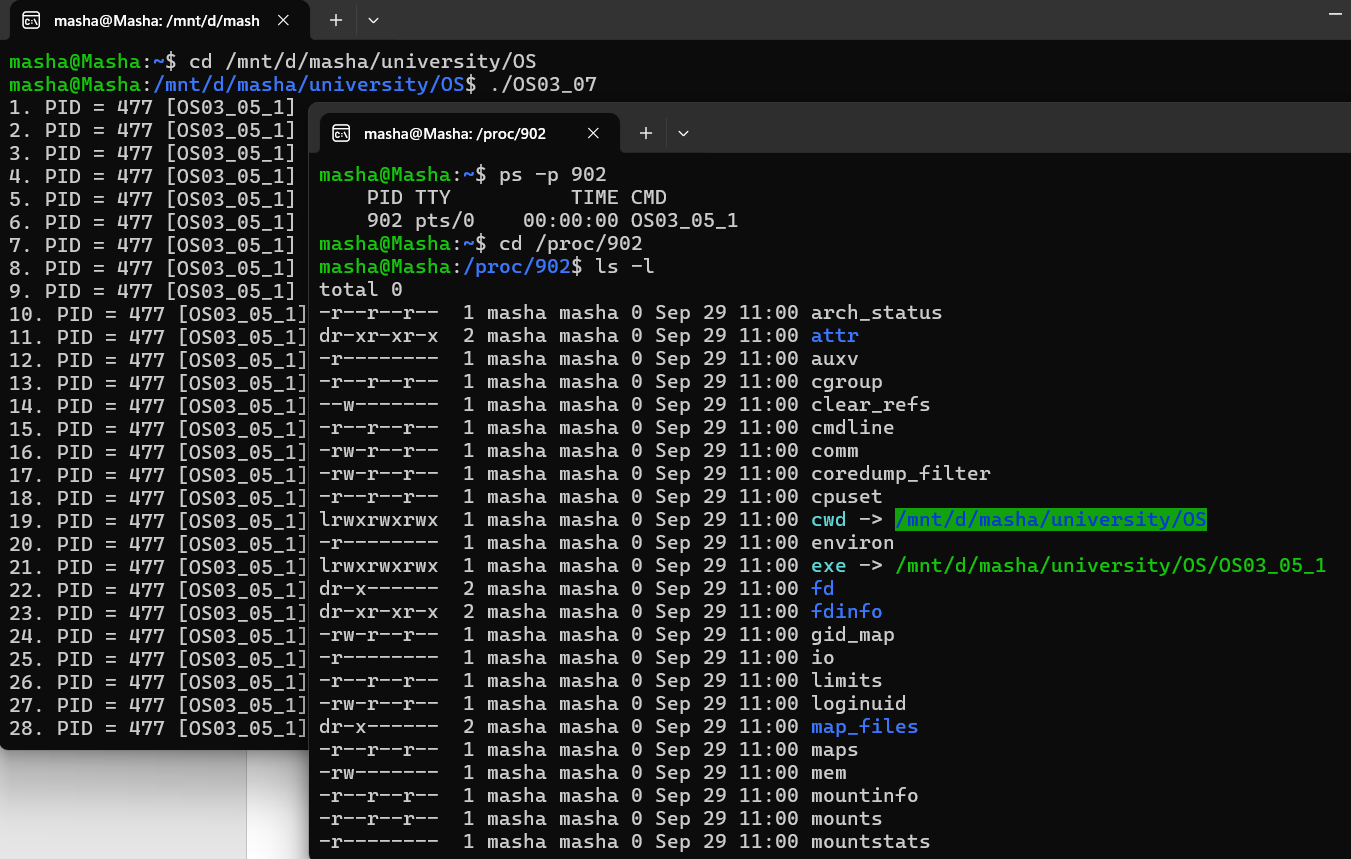
22. Разработайте консольное Linux-приложение OS03\_07 на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <errno.h>  #include <unistd.h>  #include <sys/types.h>  #include <sys/wait.h>  int main()  {    char\* const argv[] = { "ps", "-ef", 0 }; // массив параметров  execv("./OS03\_05\_1", argv);  for (int i = 1; i <= 100; ++i)  {  printf("%d. PID = %d [OS03\_07]\n", i, getpid());  sleep(1);  }  exit(0);  } |

23. Приложение OS03\_07 должно создавать один дочерний процесс OS03\_05\_1 (отдельный модуль) с помощью системного вызова exec.



24. Продемонстрируйте информацию о процессах OS03\_07 и OS03\_05\_1 с помощью файловой системы /proc.



25. Продемонстрируйте информацию о процессах OS03\_07 и OS03\_05-1 с помощью утилиты ps.

26. Продемонстрируйте разницу системных вызовов system и exec.