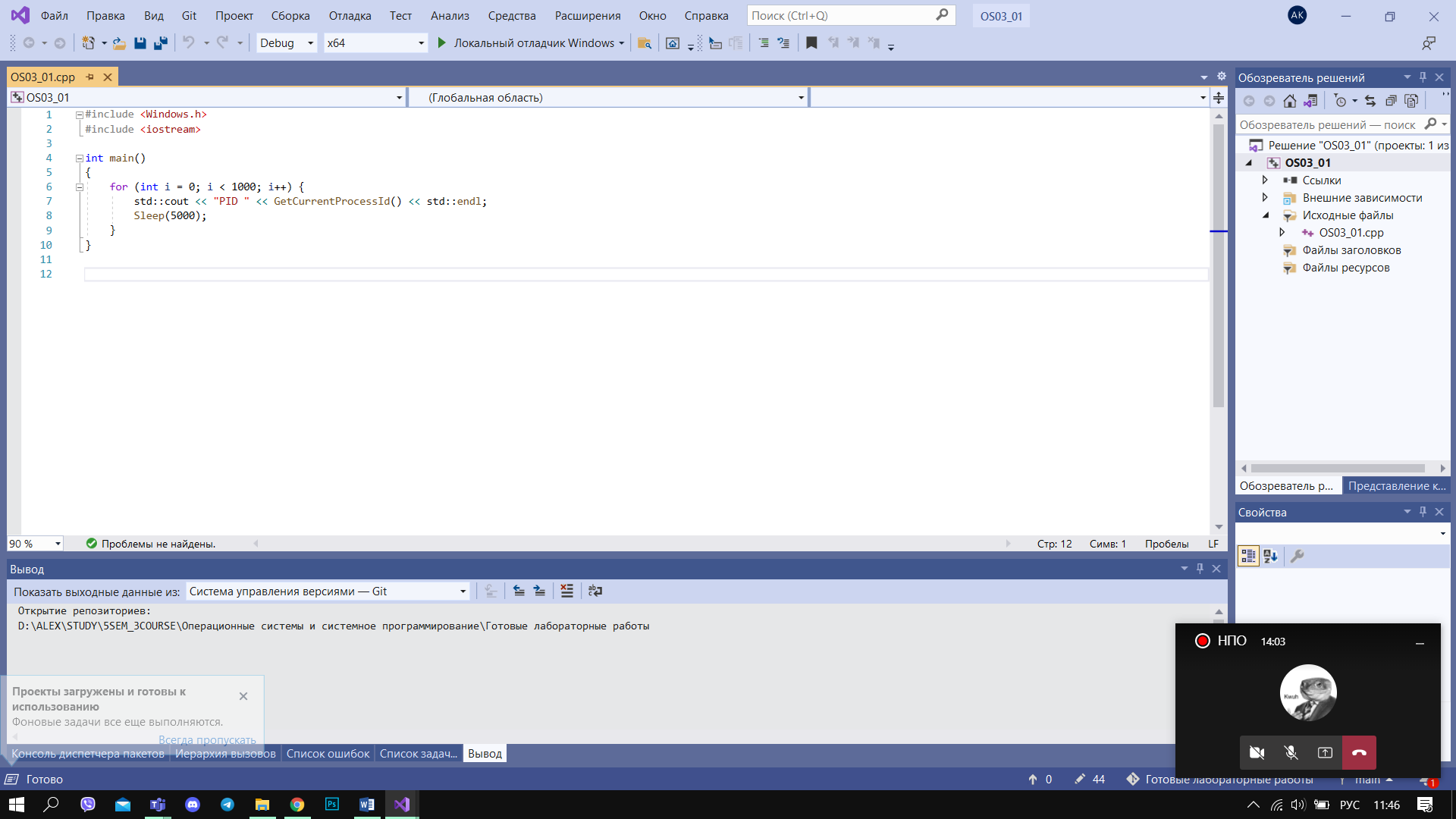
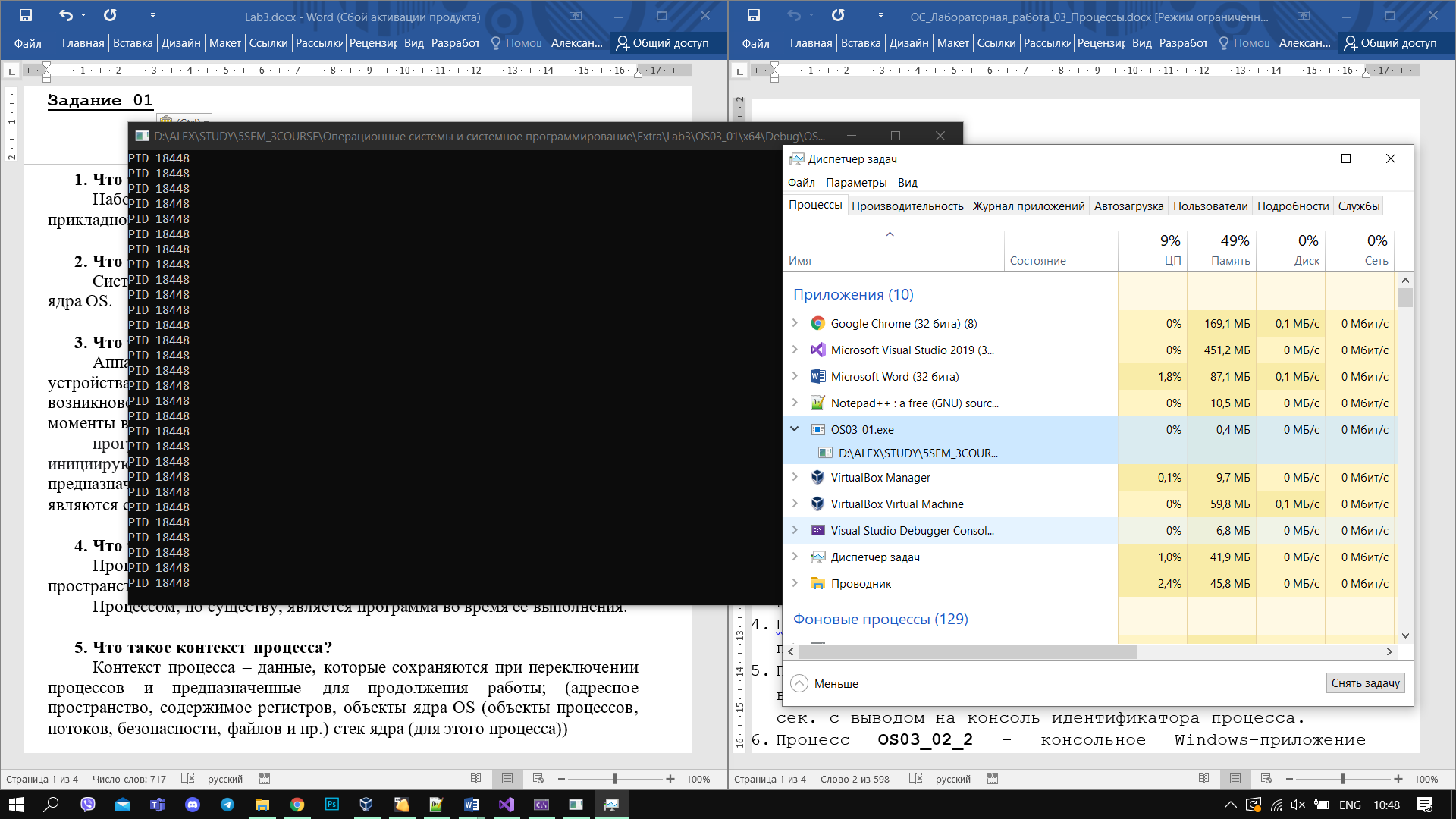
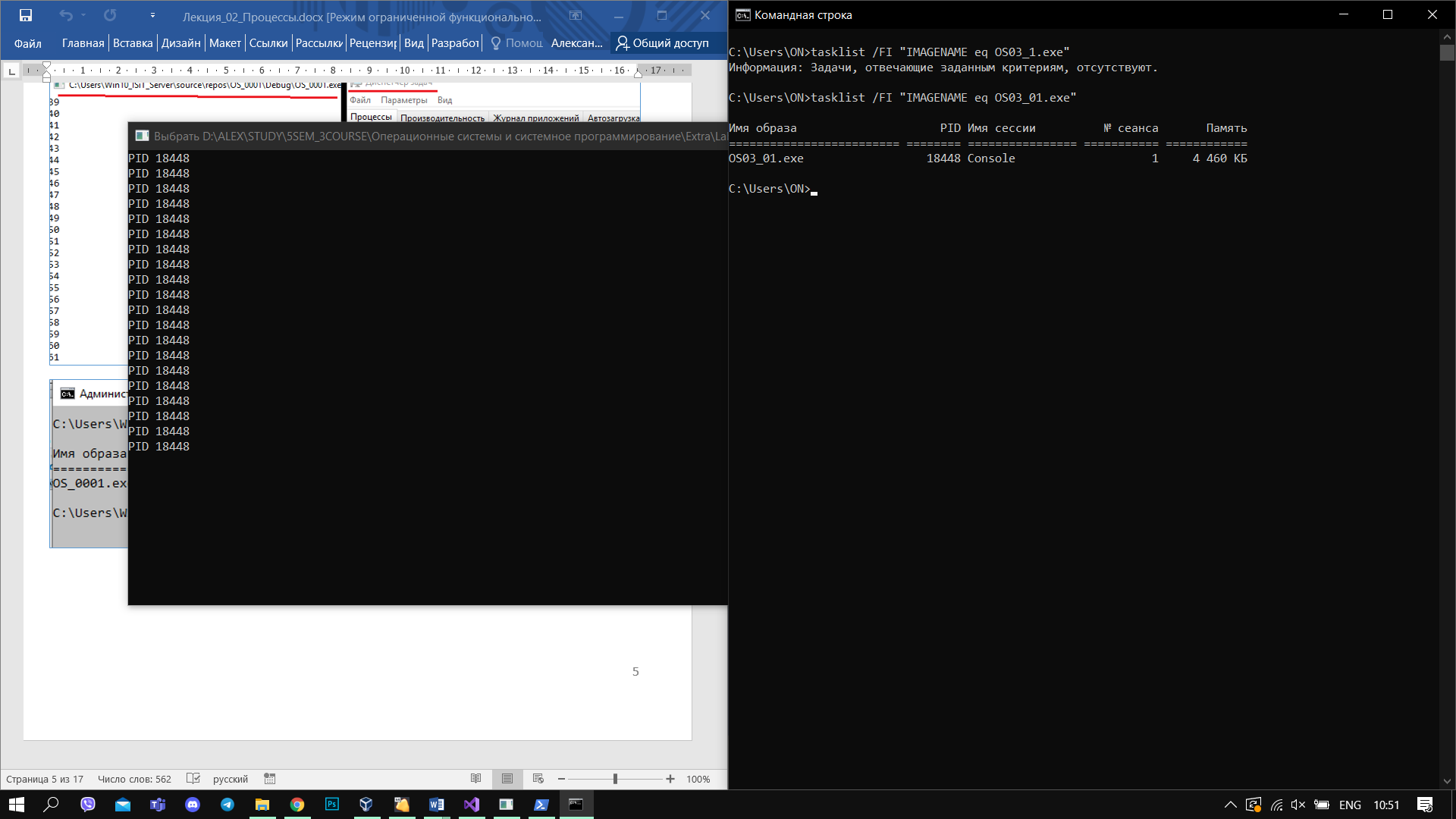
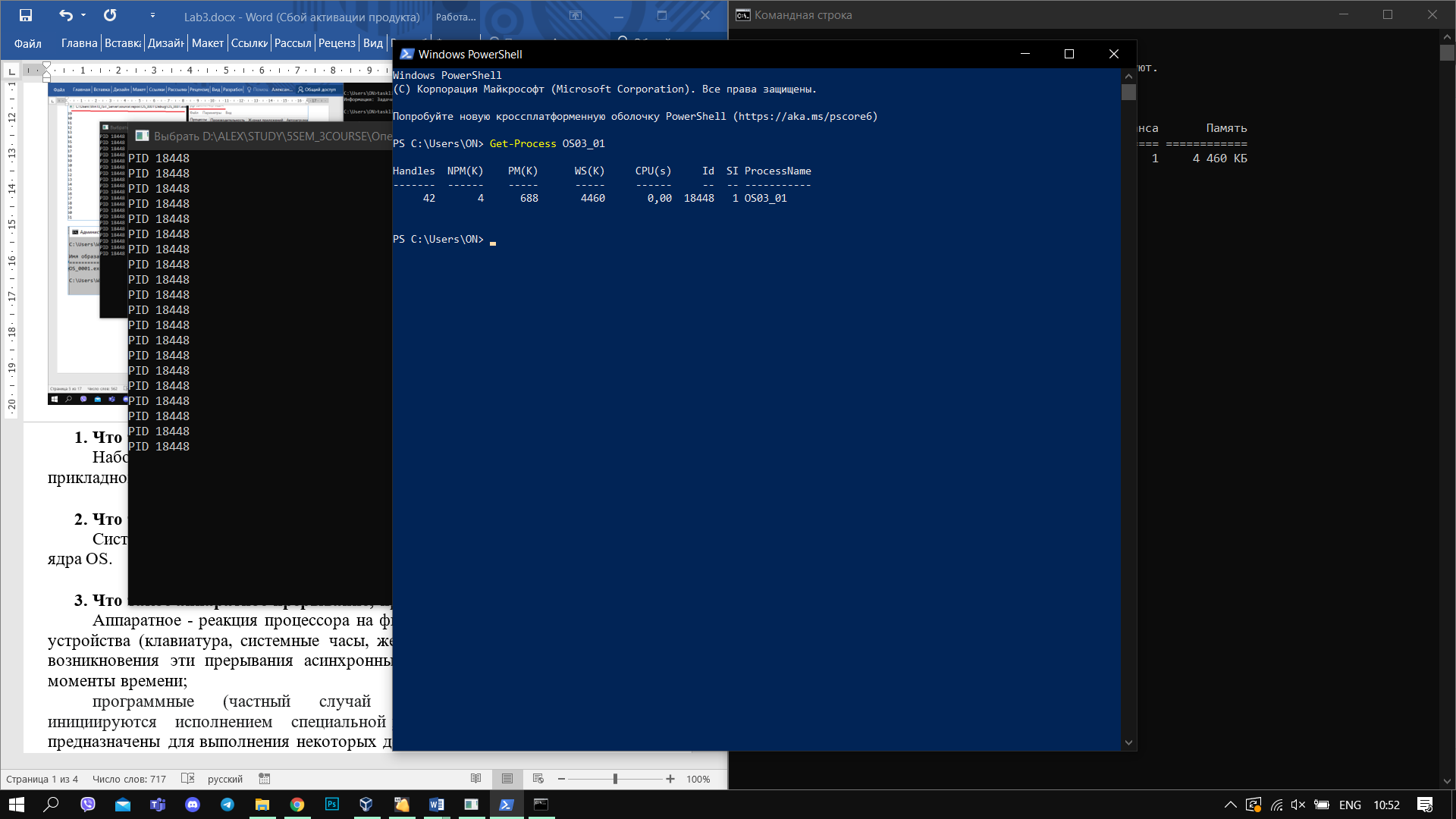
**Задание 01**

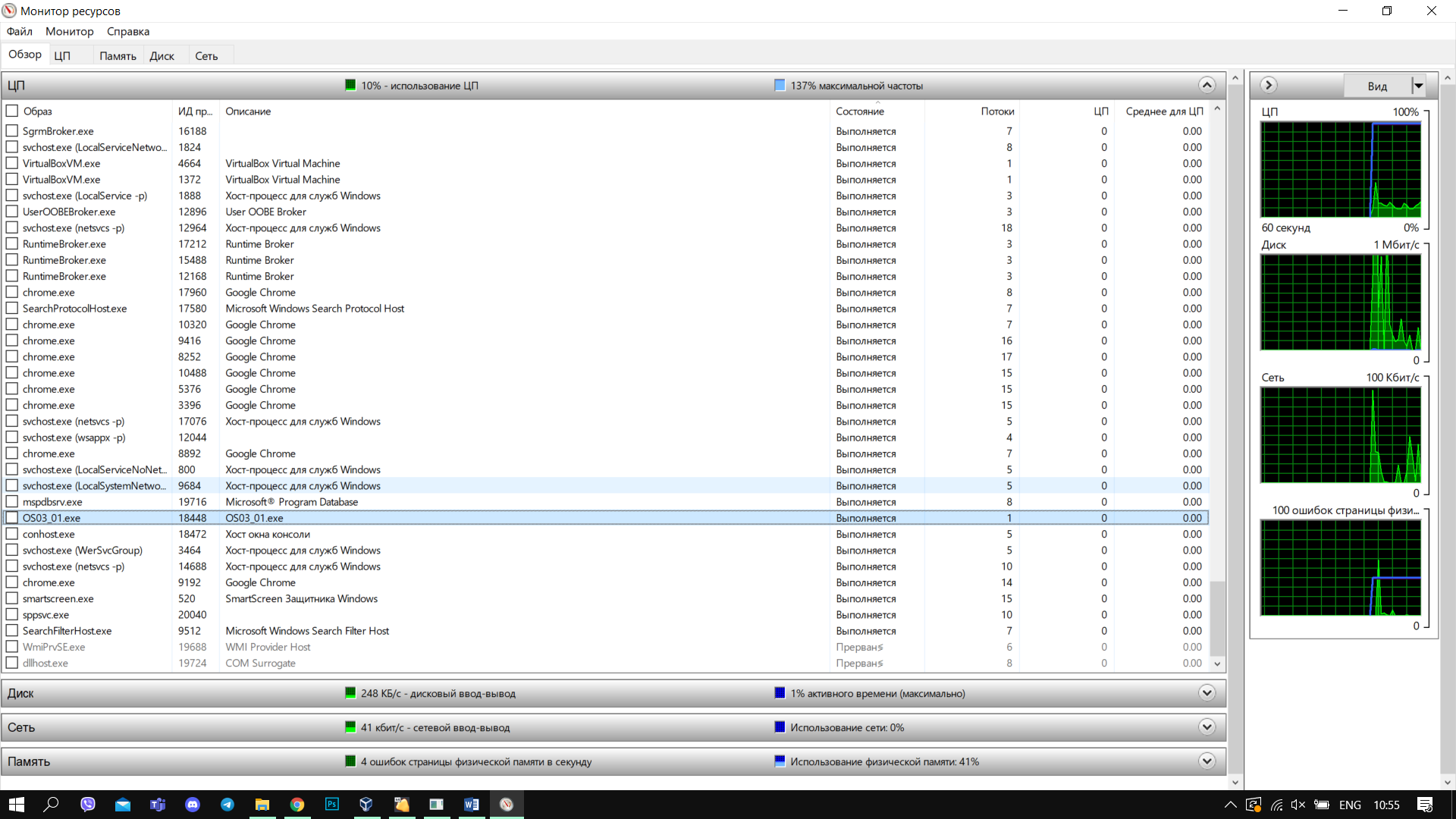


Листинг приложения «OS03\_01»









**Задание 02**

#include <Windows.h>

#include <iostream>

using namespace std;

int main()

{

LPCWSTR an1 = L"OS03\_02\_1.exe";

LPCWSTR an2 = L"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);

si2.cb = sizeof(STARTUPINFO);

if (CreateProcess(an1, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si1, &pi1)) {

cout << "--Process OS03\_02\_1 created\n";

}

else {

cout << "--Process OS03\_02\_1 not created\n";

}

if (CreateProcess(an2, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si2, &pi2)) {

cout << "--Process OS03\_02\_2 created\n";

}

else {

cout << "--Process OS03\_02\_2 not created\n";

}

for (int i = 0; i < 1000; i++) {

std::cout << "OS03\_02 PID " << GetCurrentProcessId() << std::endl;

Sleep(1000);

}

WaitForSingleObject(pi1.hProcess, INFINITE);

WaitForSingleObject(pi2.hProcess, INFINITE);

CloseHandle(pi1.hProcess);

CloseHandle(pi2.hProcess);

return 0;

}

Листинг приложения «OS03\_02»

#include <Windows.h>

#include <iostream>

int main()

{

for (int i = 0; i < 50; i++) {

std::cout << "OS03\_02\_1 PID " << GetCurrentProcessId() << std::endl;

Sleep(1000);

}

return 0;

}

Листинг приложения «OS03\_02\_1»

#include <Windows.h>

#include <iostream>

int main()

{

for (int i = 0; i < 125; i++) {

std::cout << "OS03\_02\_2 PID " << GetCurrentProcessId() << std::endl;

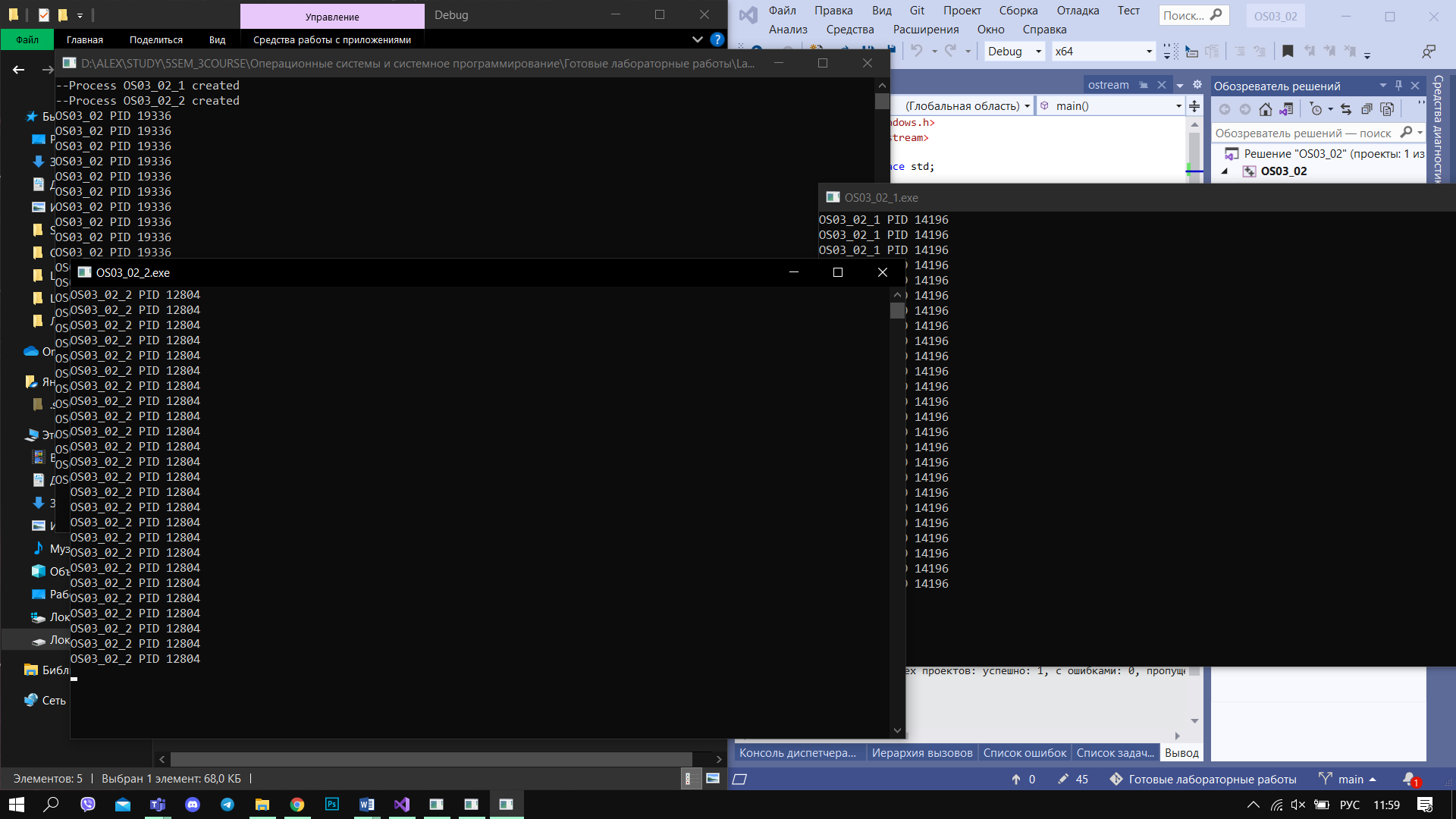
Sleep(1000);

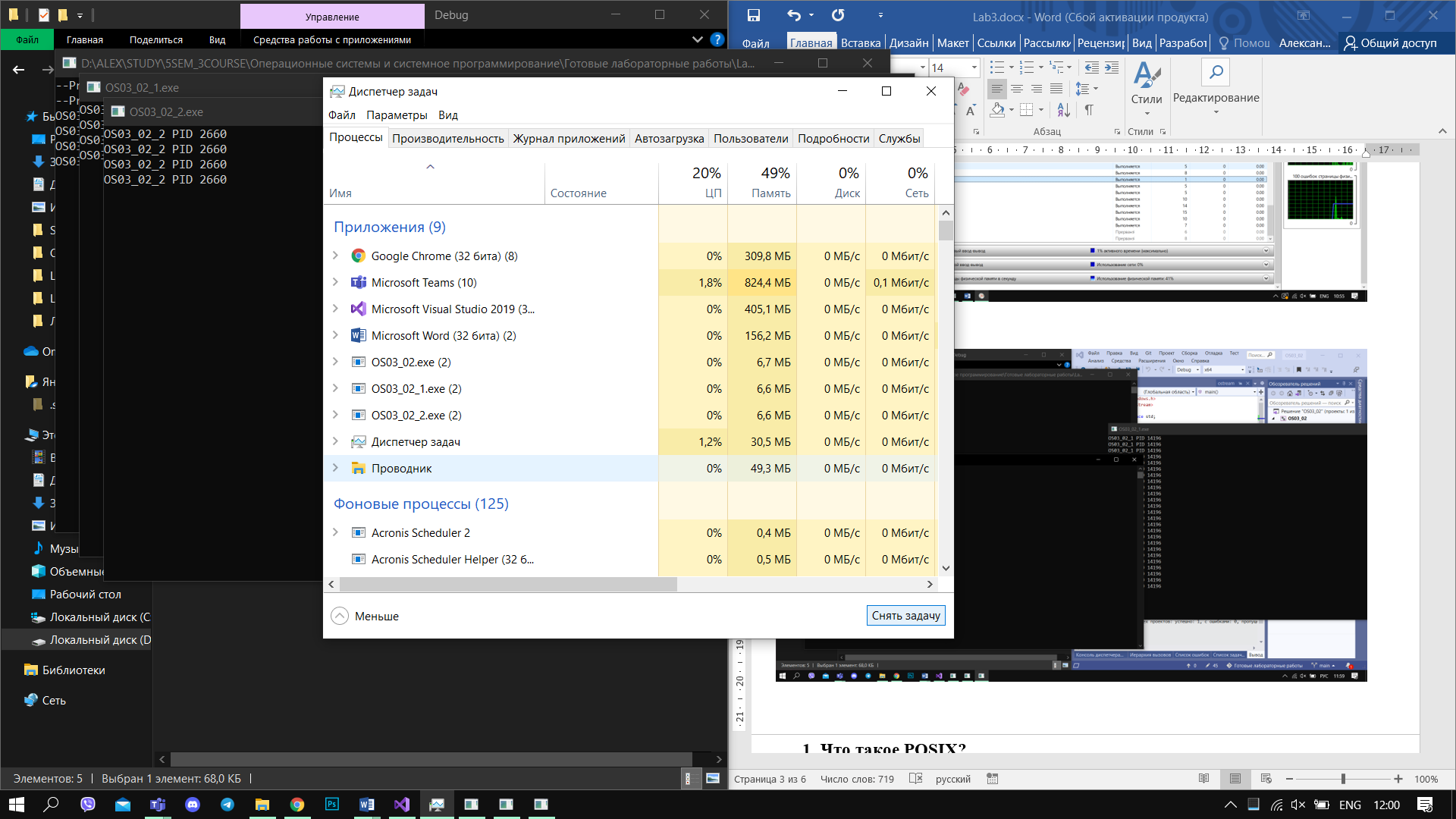
}

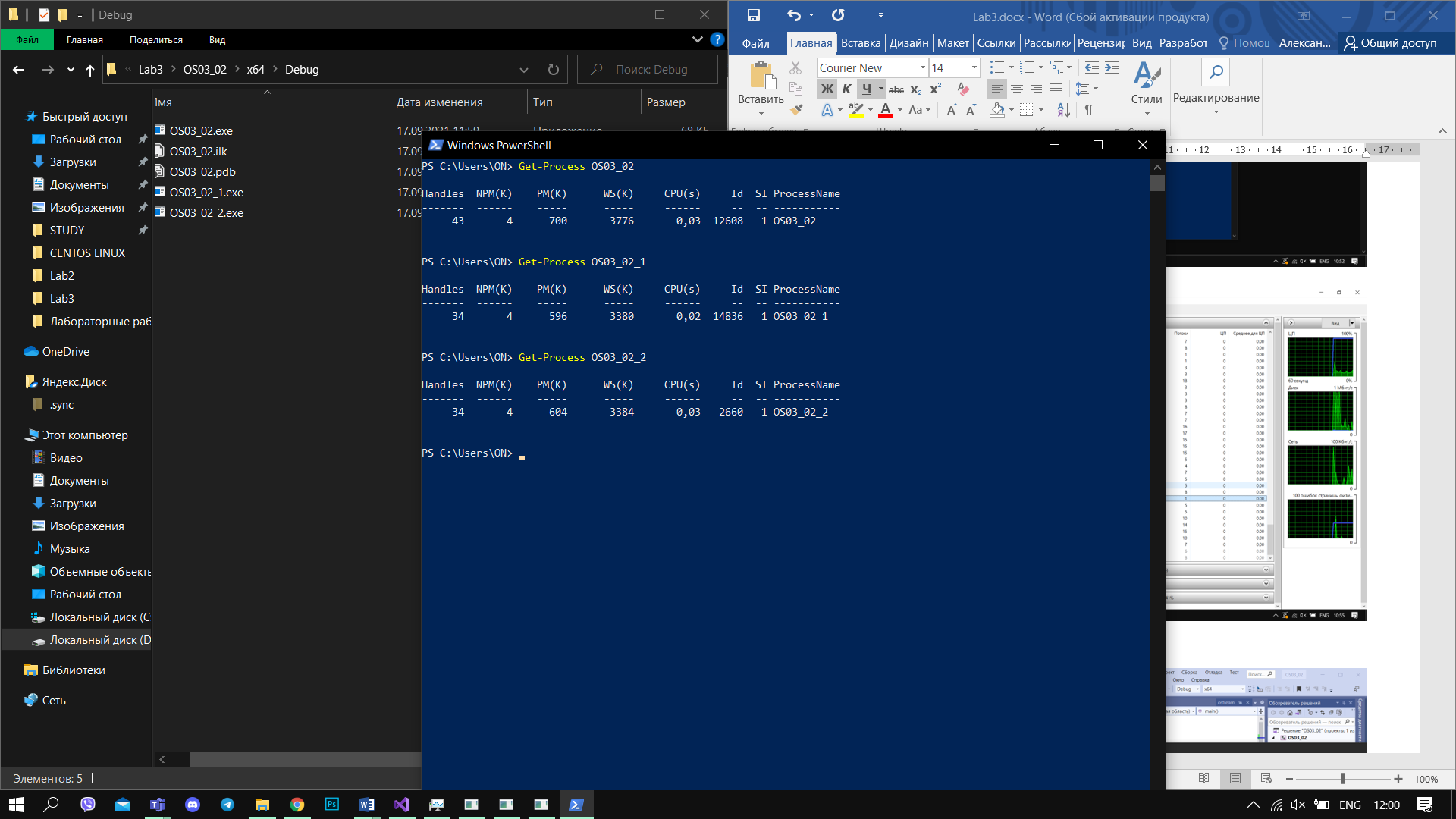
return 0;

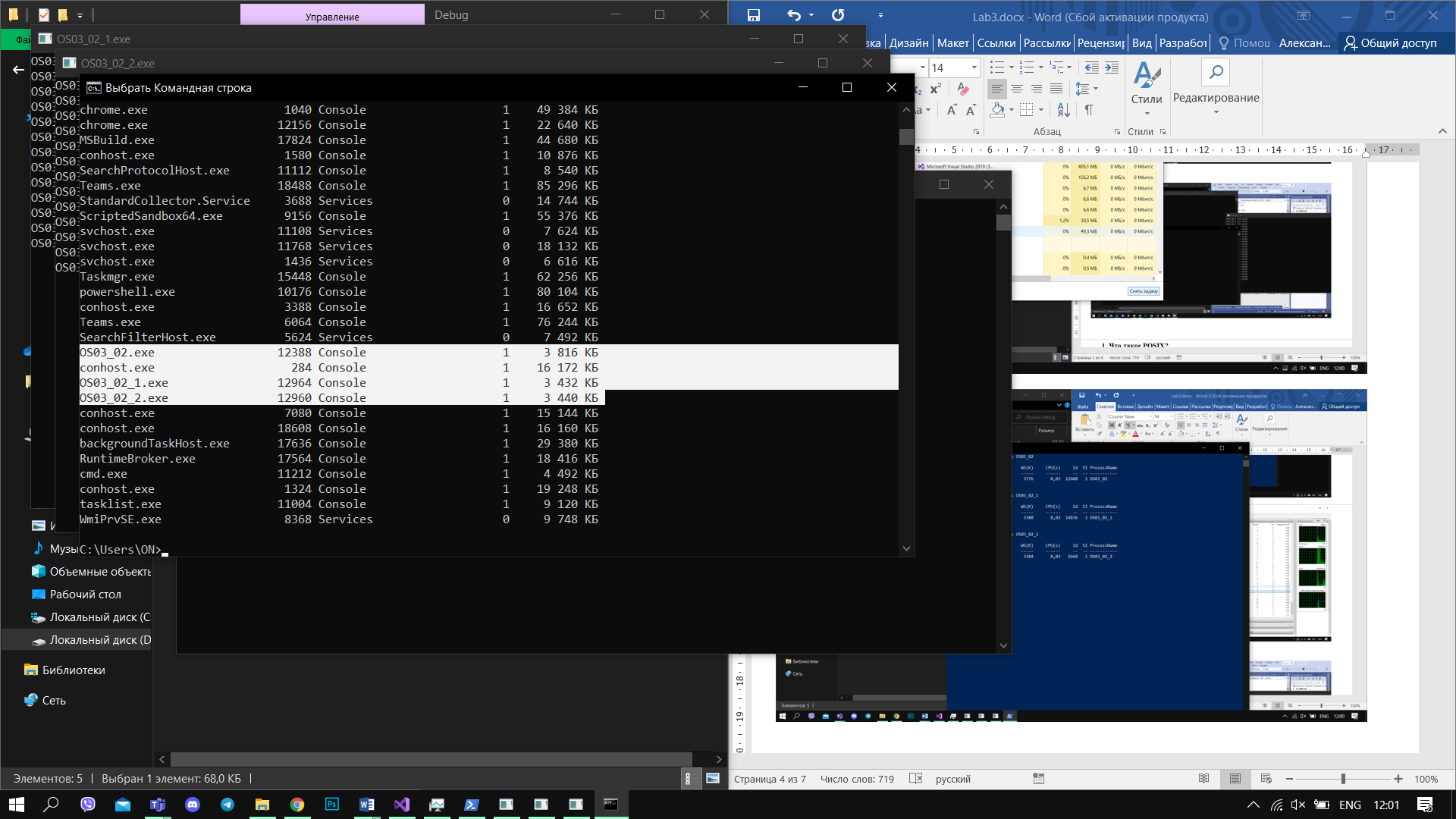
}

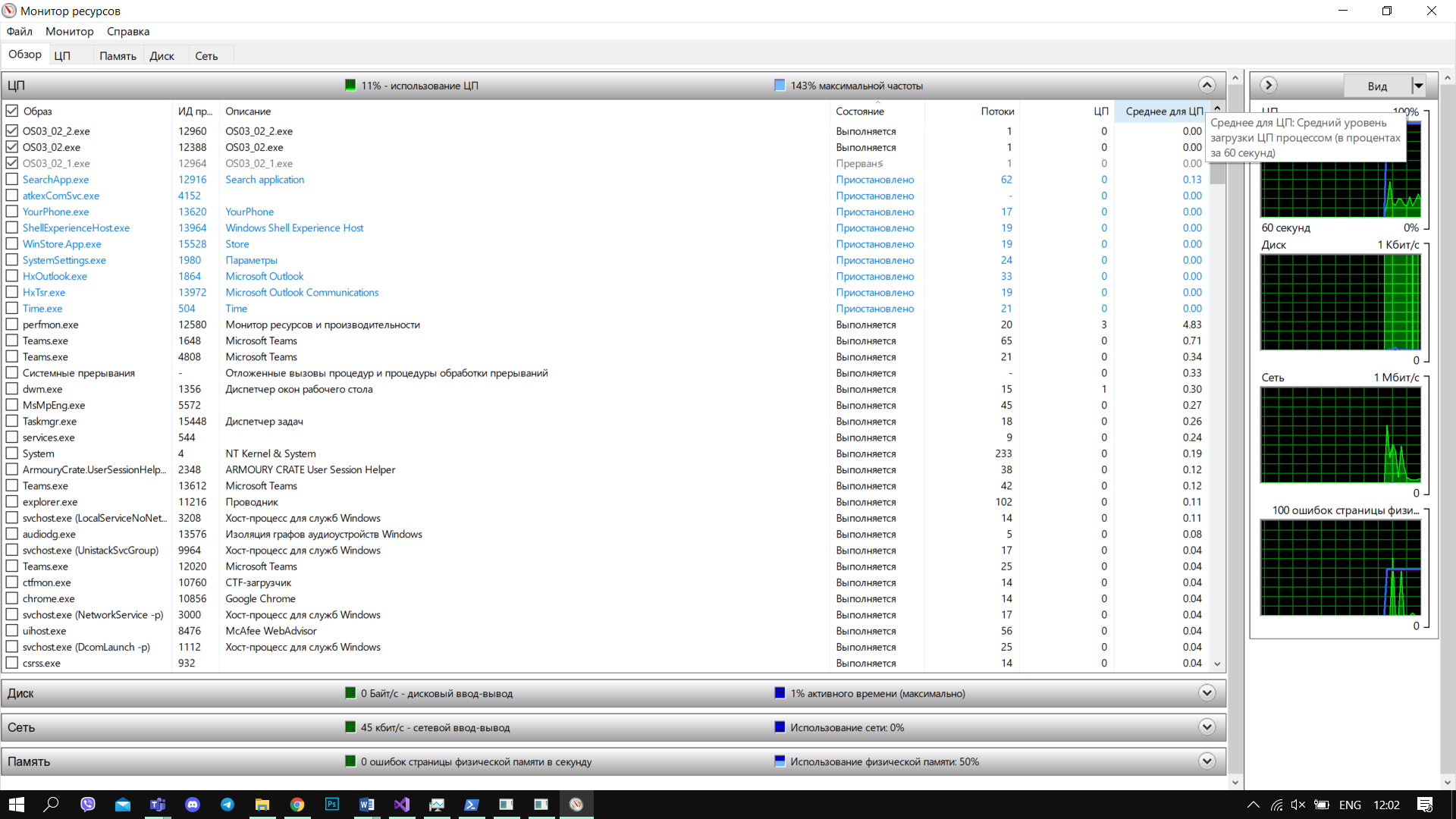
Листинг приложения «OS03\_02\_2»



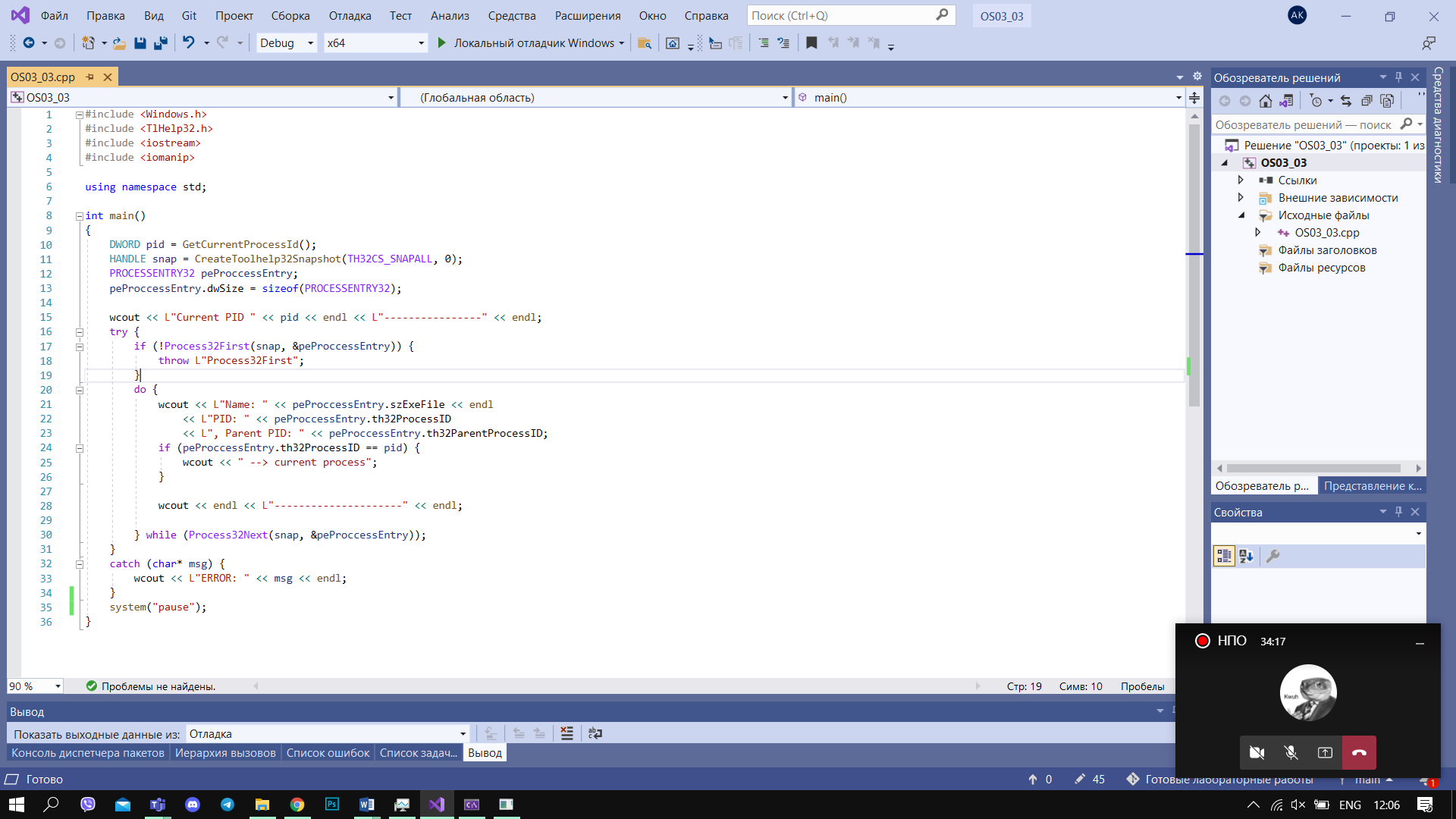




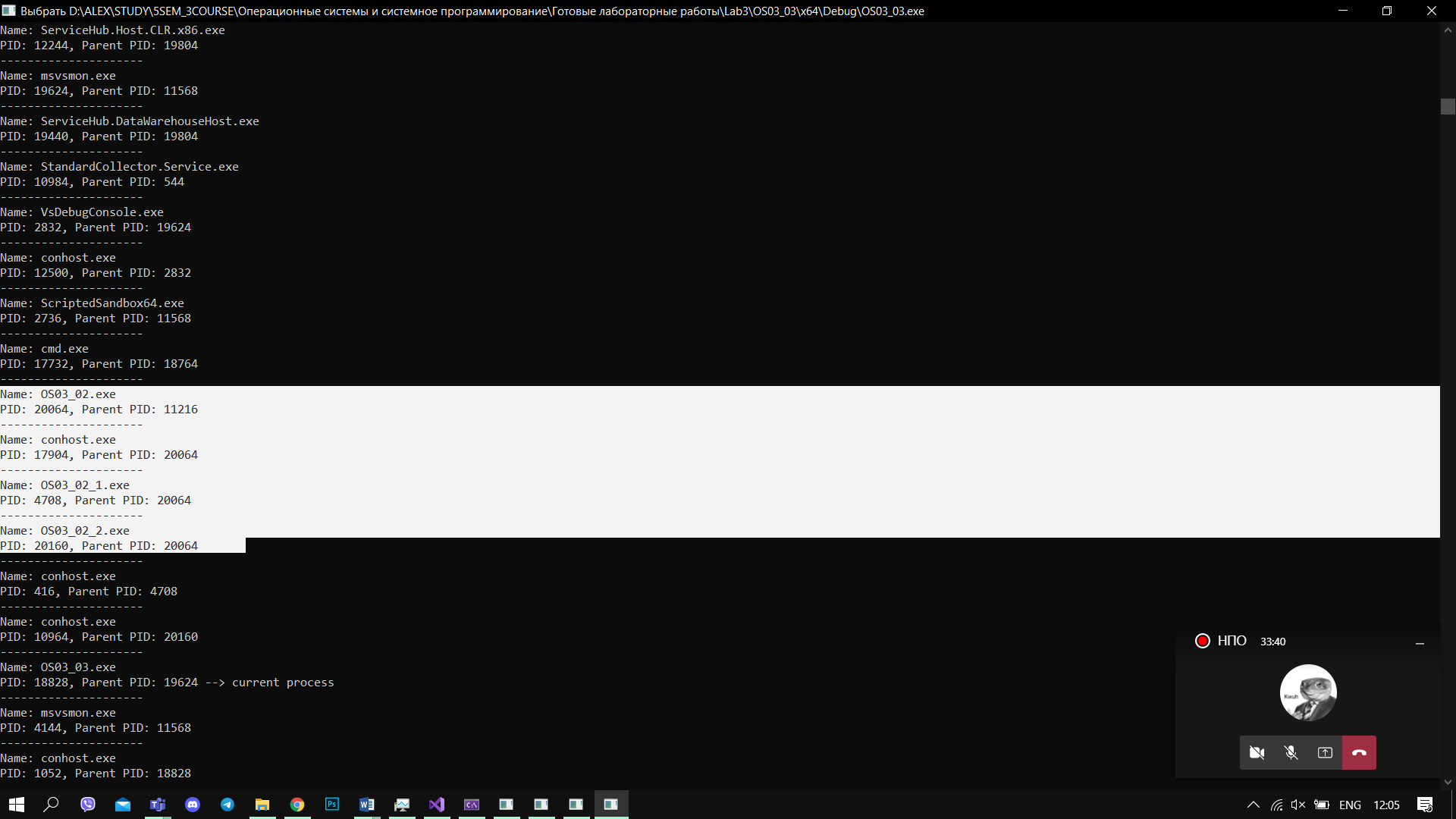




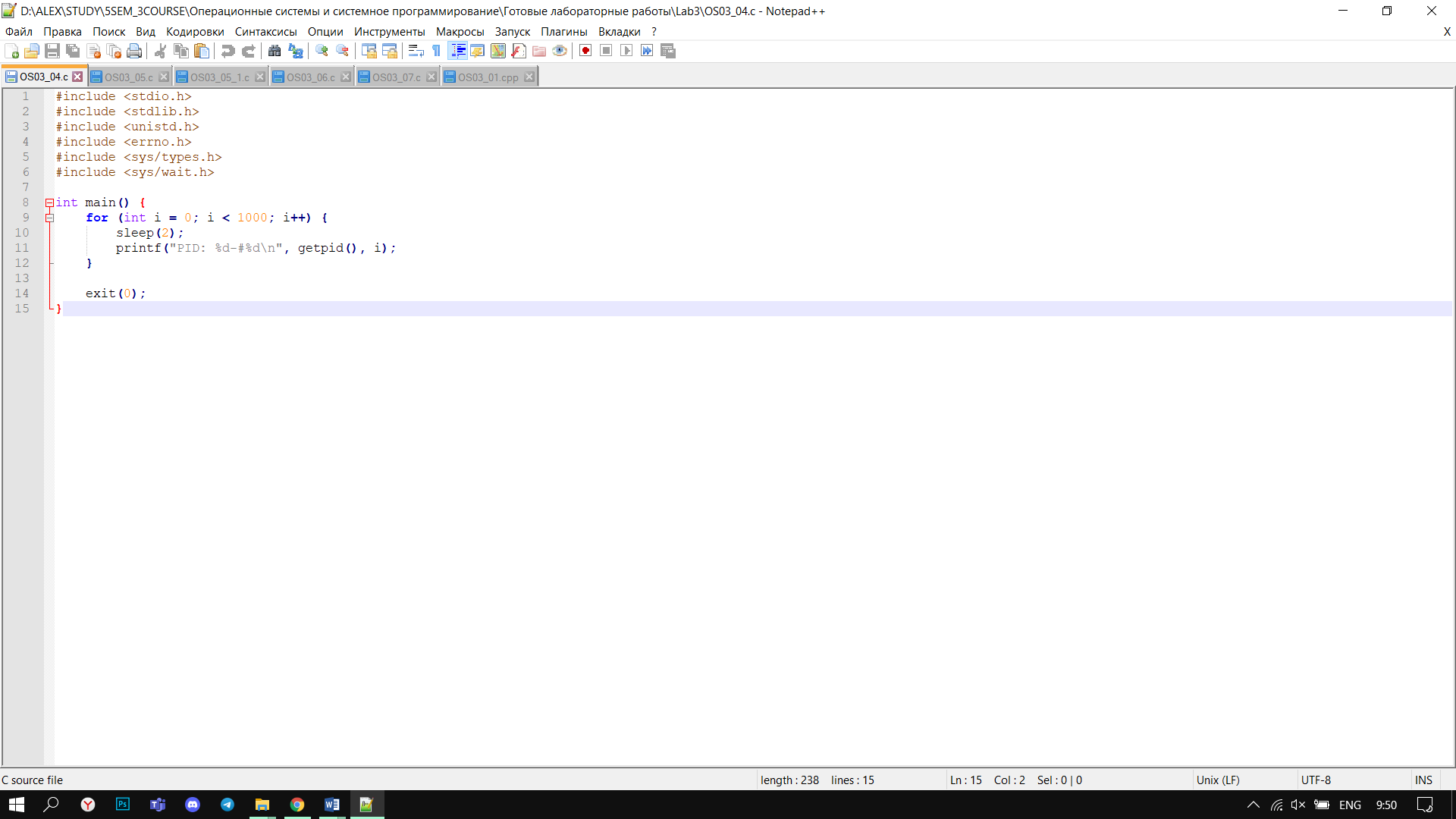
**Задание 03**

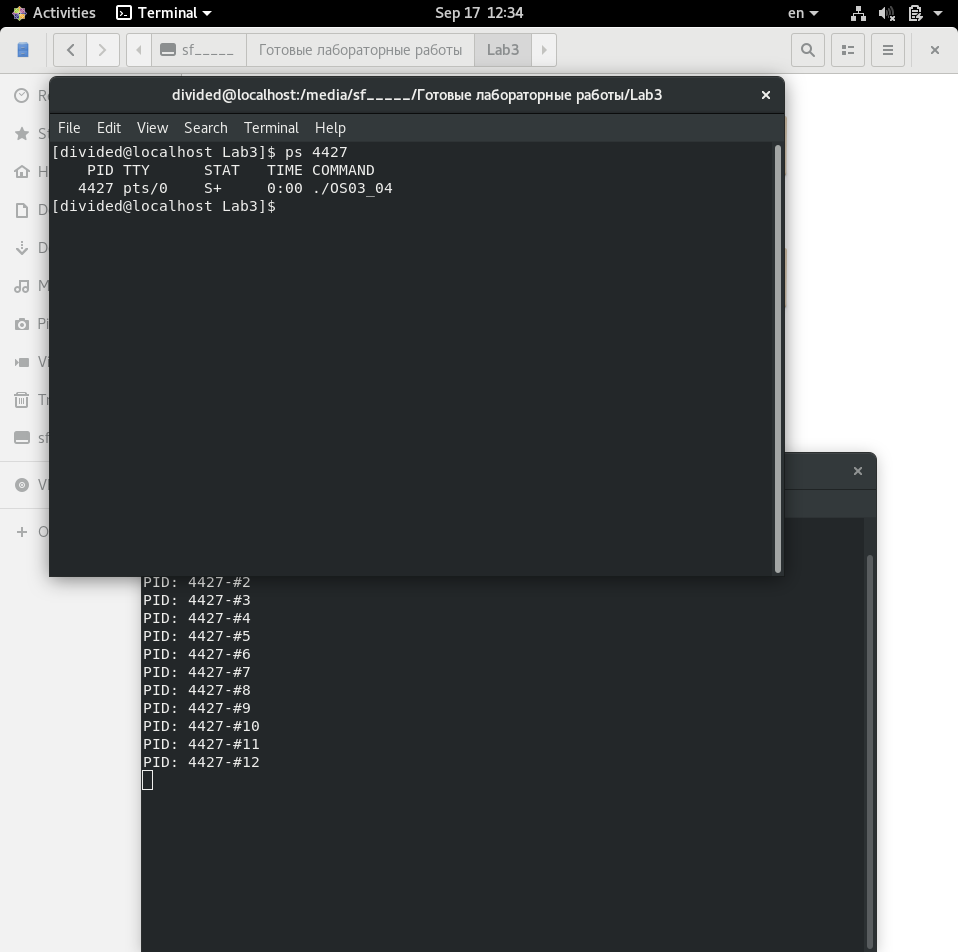
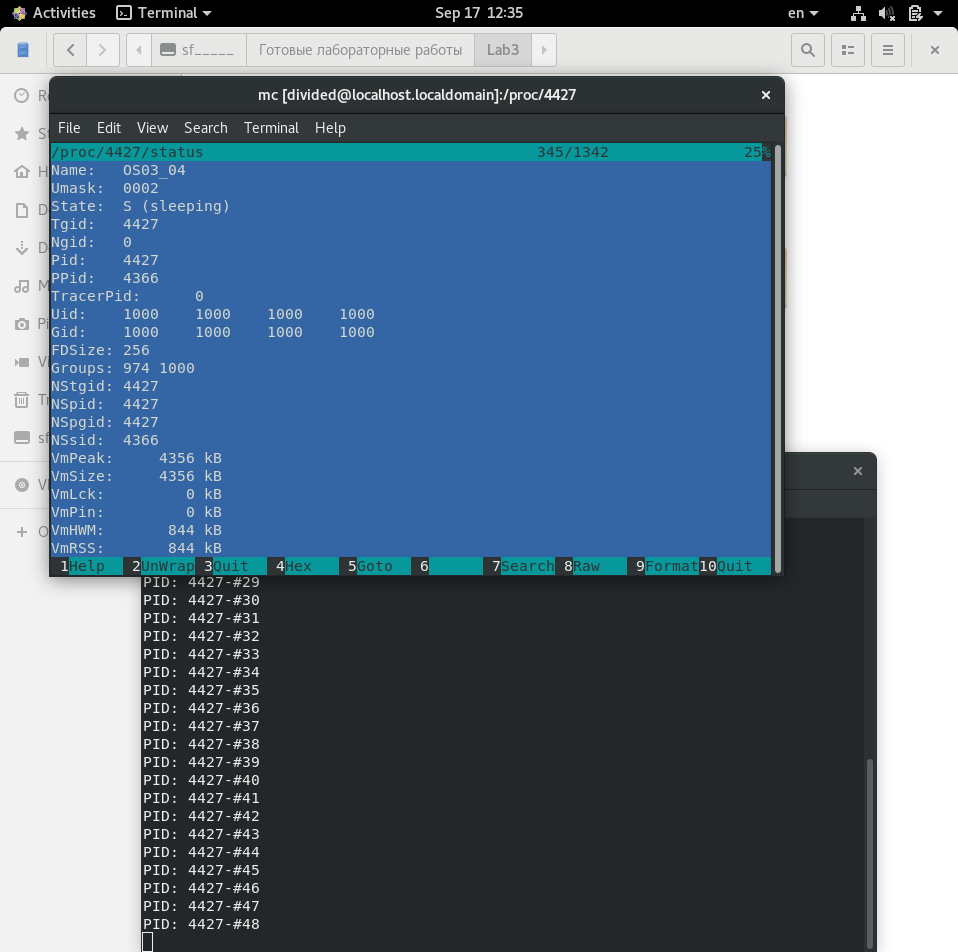


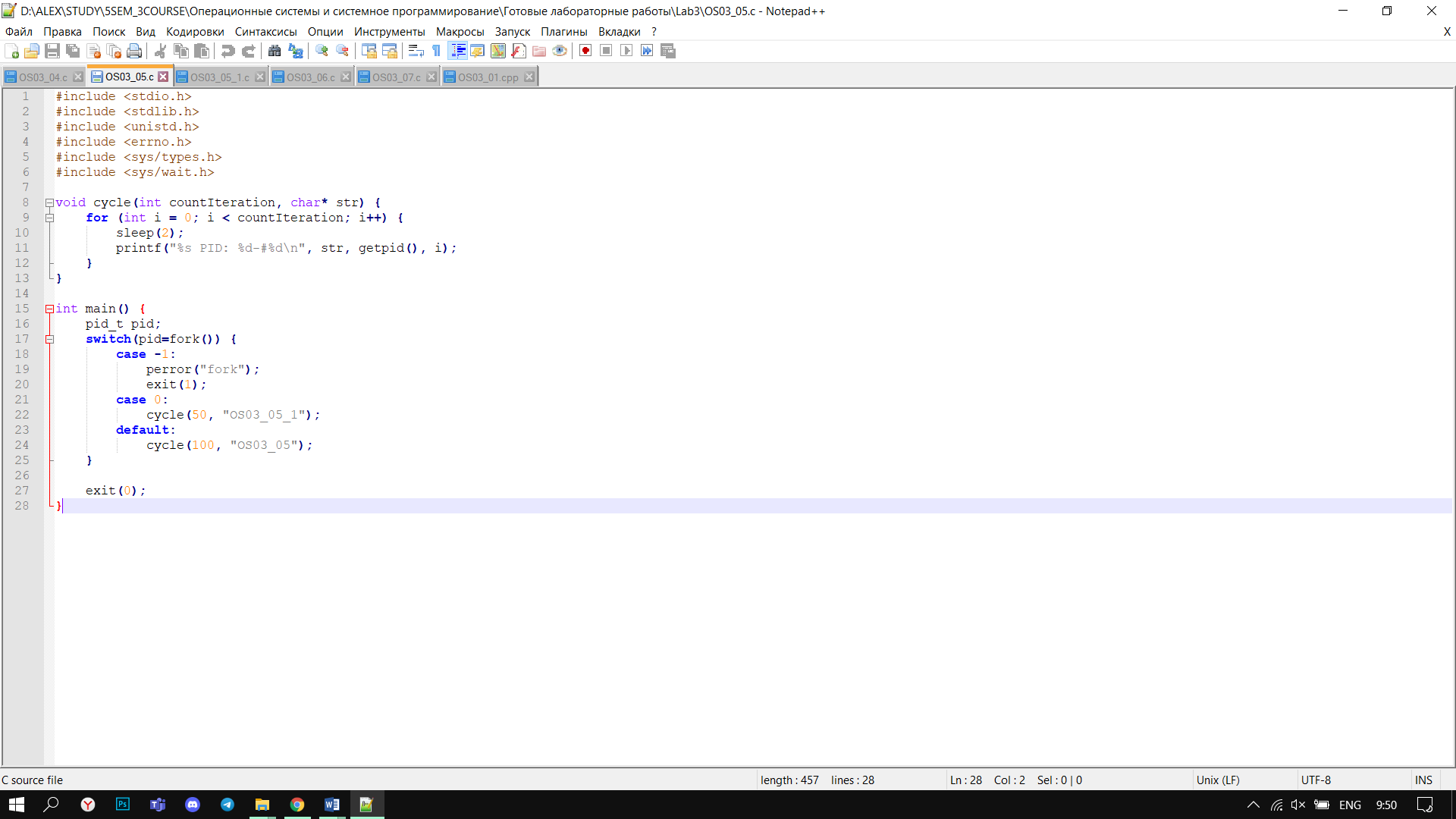
Листинг приложения «OS03\_03»

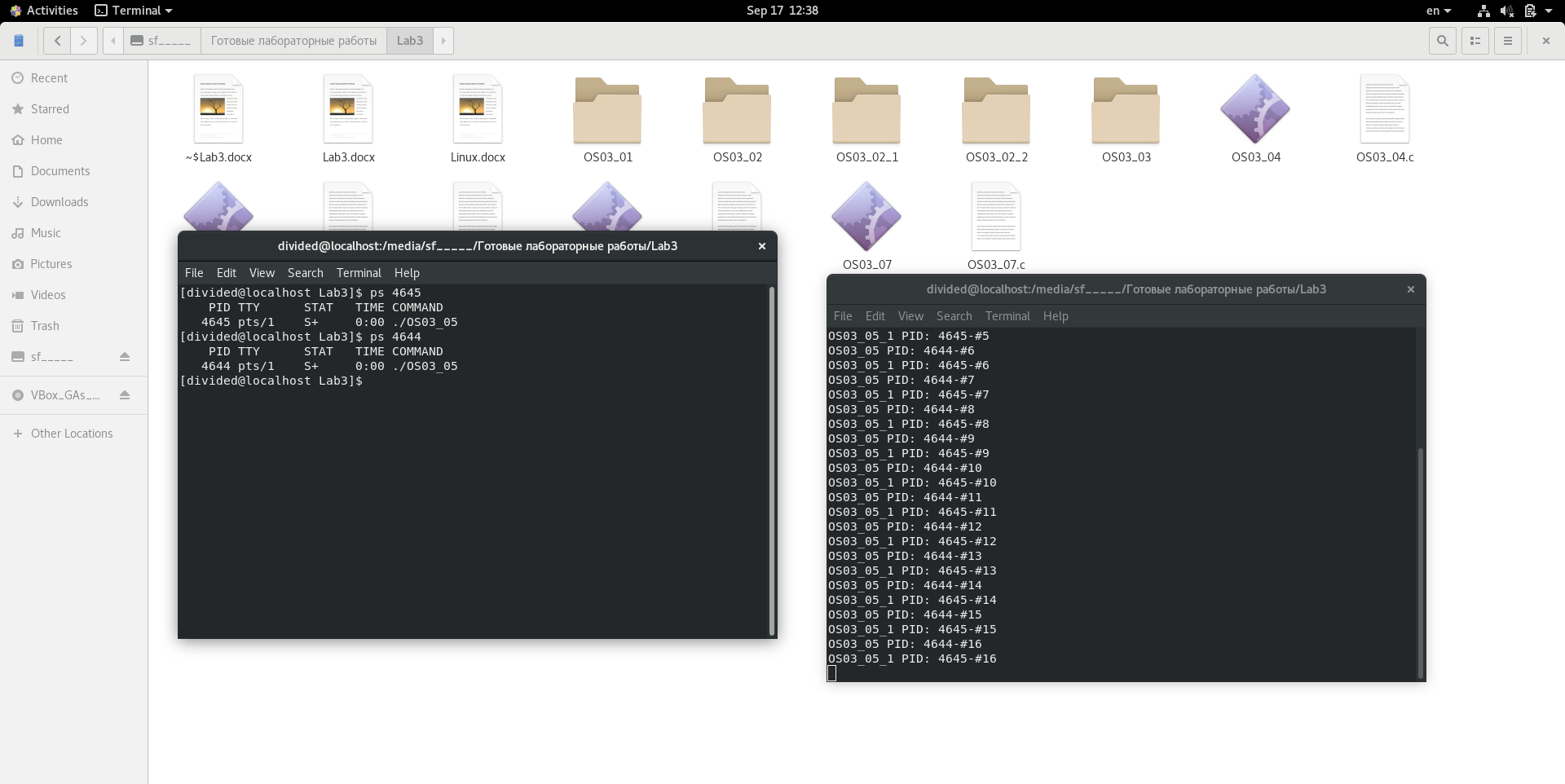
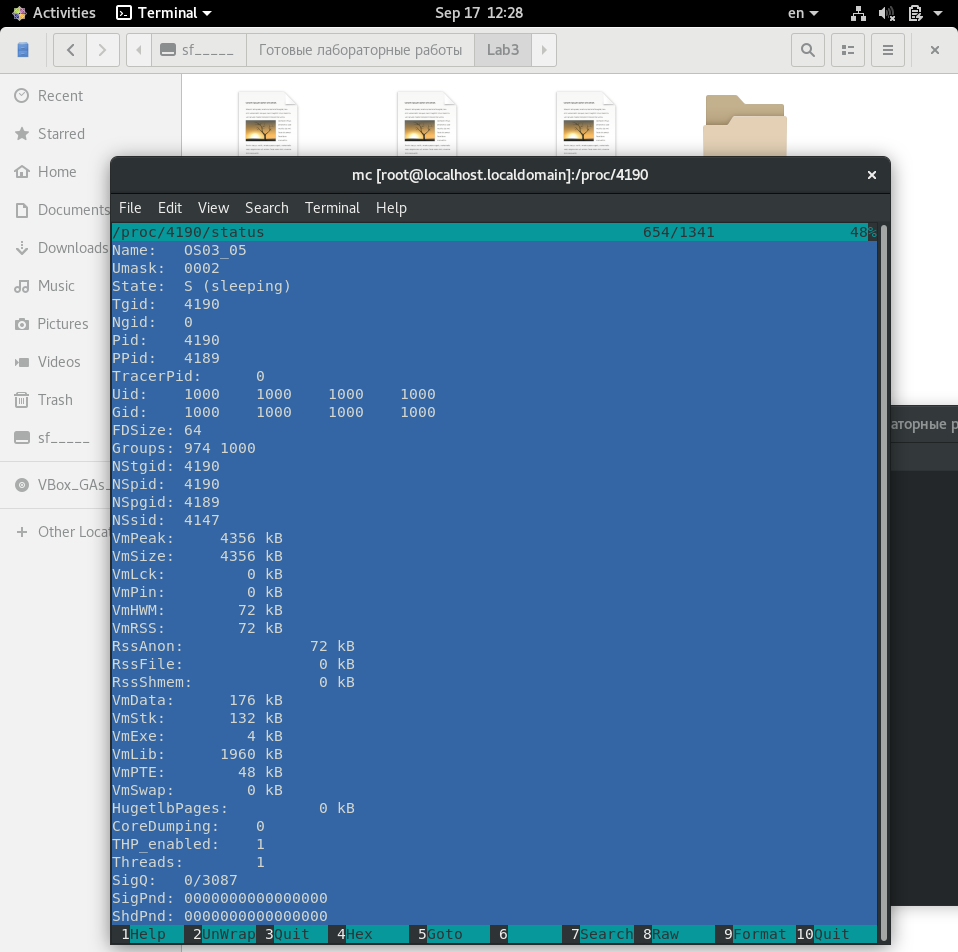
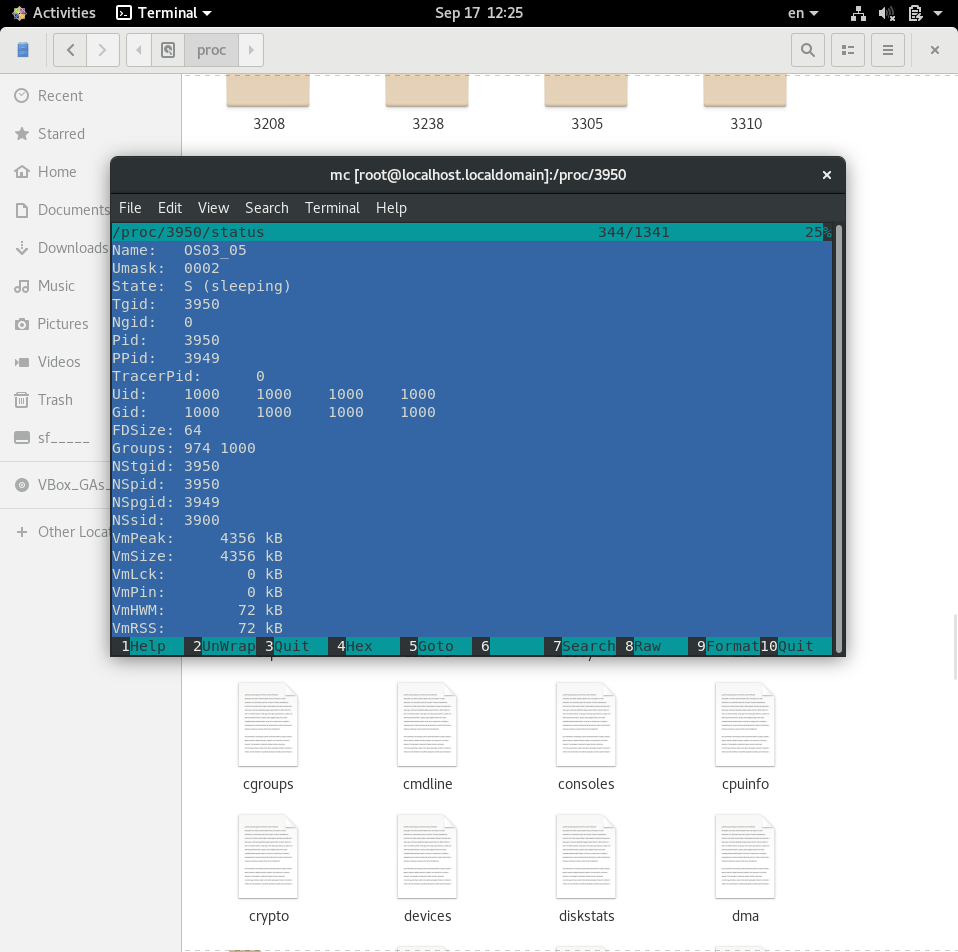


**Задание 04**

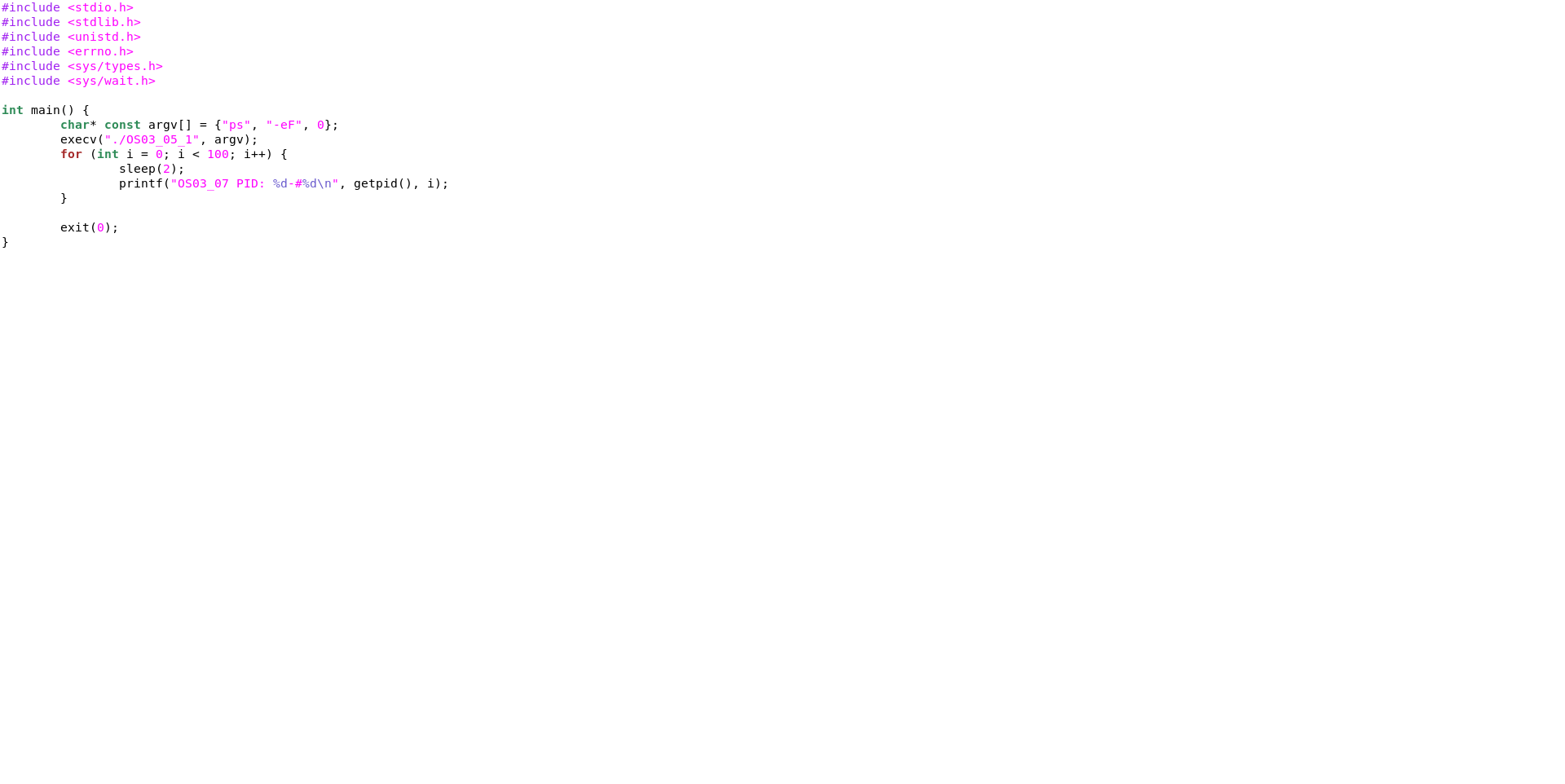
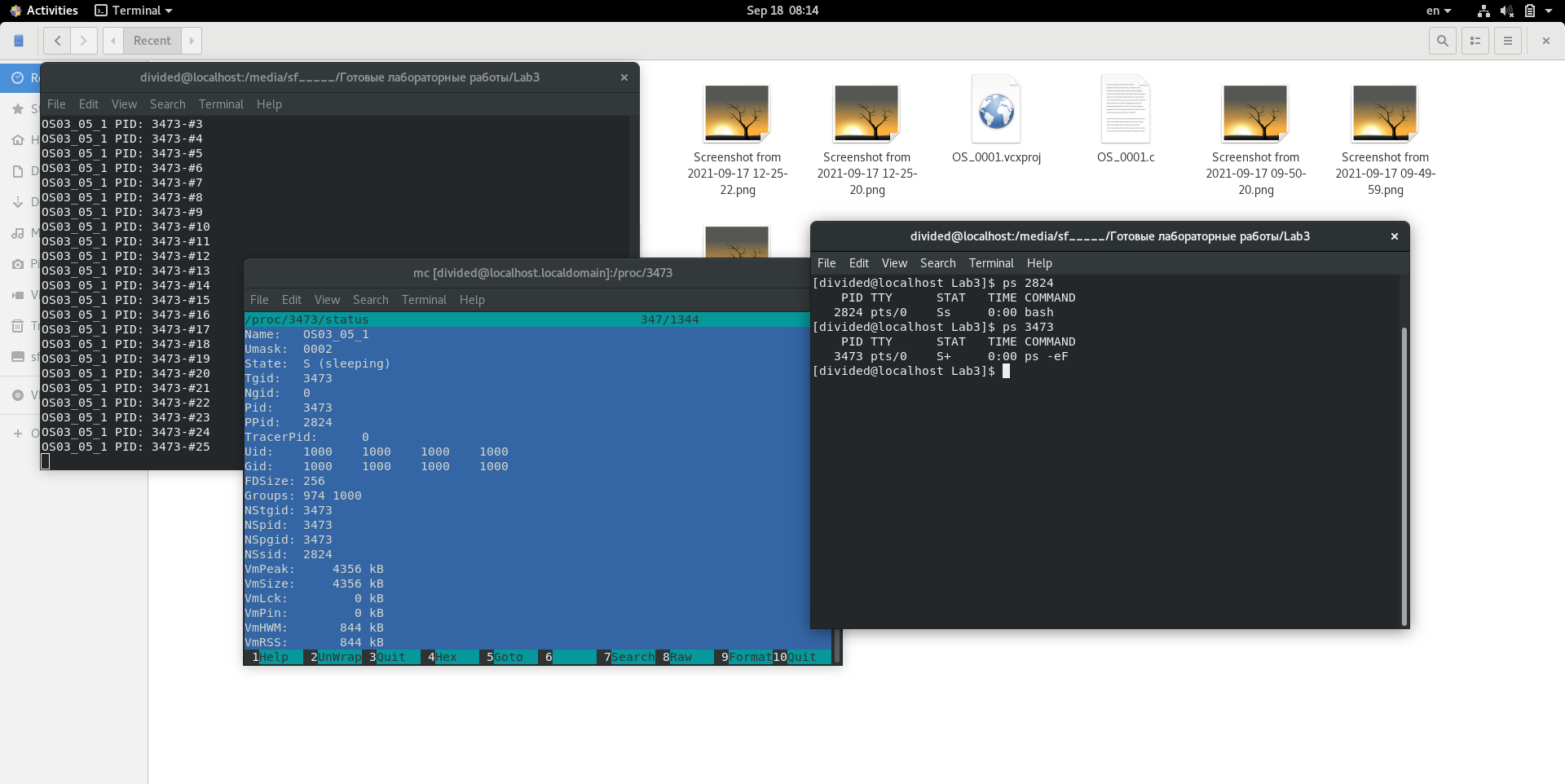
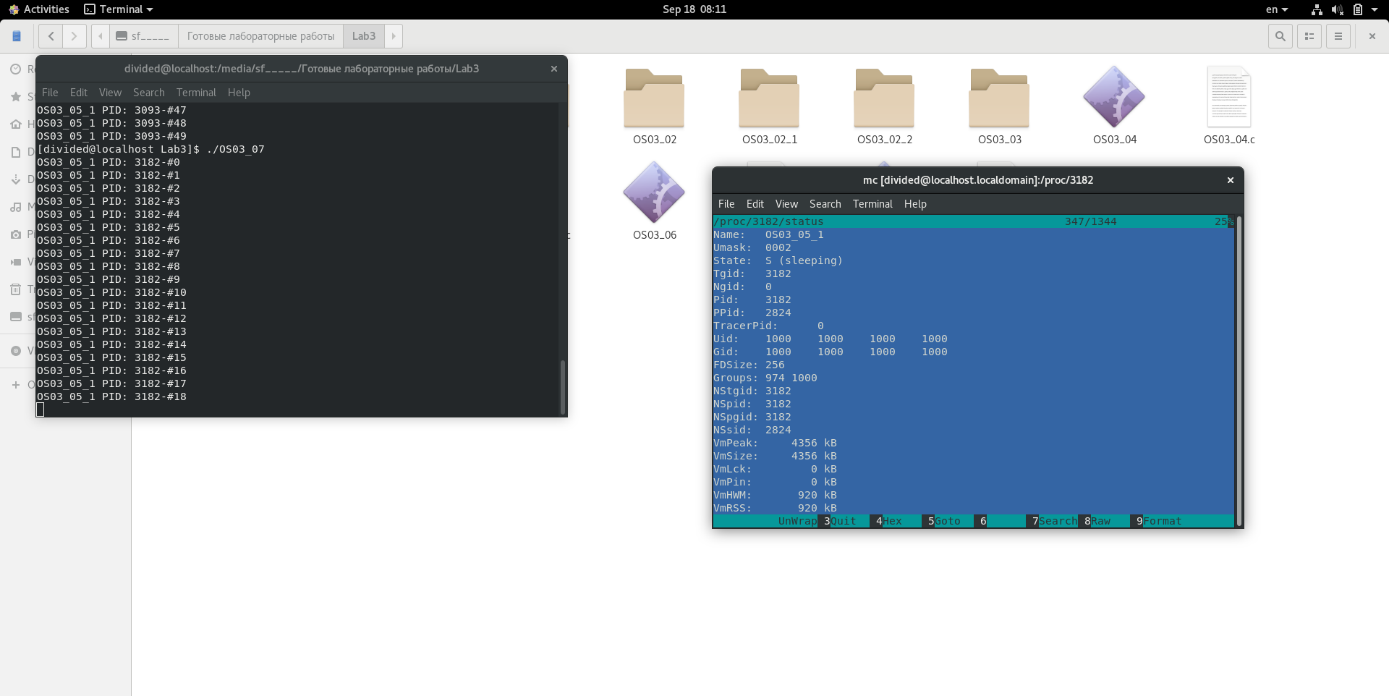
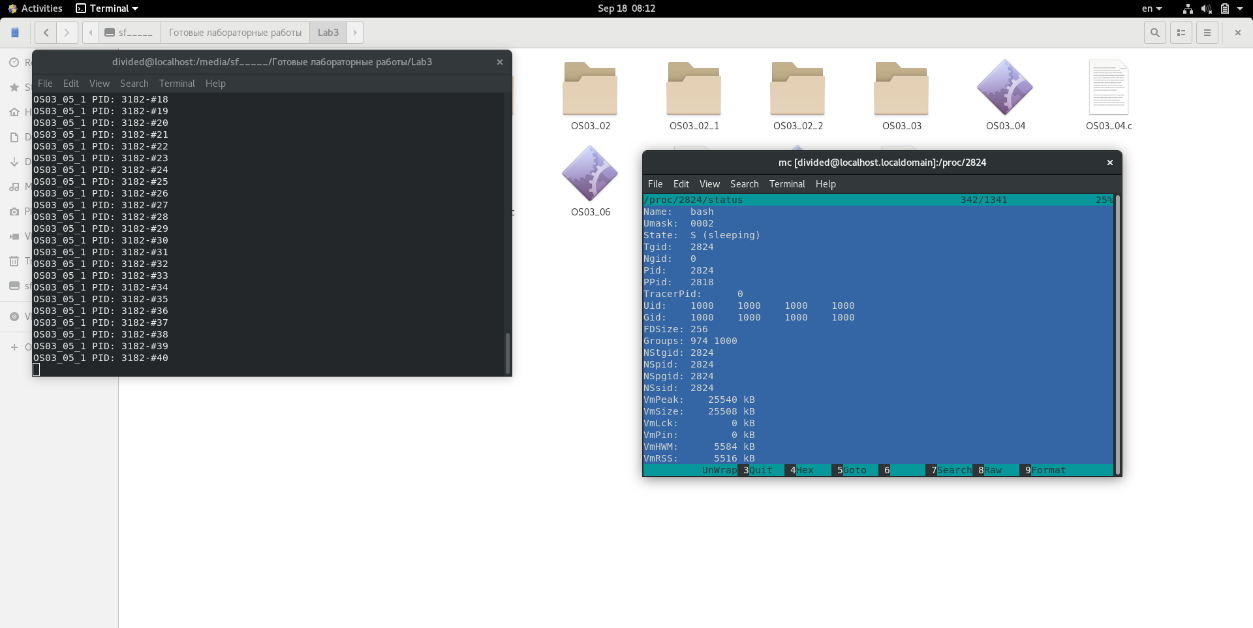




**Задание 05**



**Задание 07**

****

1. **Что такое POSIX?**

**POSIX** — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов.

1. **Что такое системный вызов?**

Системный вызов - механизм вызова прикладной программой функции ядра OS.

1. **Что такое аппаратное прерывание, программное прерывание?**

Аппаратное - реакция процессора на физический сигнал от некоторого устройства (клавиатура, системные часы, жесткий диск и т.д.), по времени возникновения эти прерывания асинхронны, т.е. происходят в случайные моменты времени;

программные (частный случай внутреннего прерывания) — инициируются исполнением специальной [инструкции](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8) в коде [программы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), предназначены для выполнения некоторых действий операционной системы, являются синхронными;

1. **Что такое процесс?**

Процесс OS – единица работы OS - объект ядра OS + адресное пространство:

Процессом, по существу, является программа во время ее выполнения.

1. **Что такое контекст процесса?**

Контекст процесса – данные, которые сохраняются при переключении процессов и предназначенные для продолжения работы; (адресное пространство, содержимое регистров, объекты ядра OS (объекты процессов, потоков, безопасности, файлов и пр.) стек ядра (для этого процесса))

1. **Что такое родительский и дочерний процесс?**

Системный вызов *fork* создает точную копию исходного процесса, называемого **родительским процессом (parent process)**. Новый процесс называется **дочерним процессом (child process)**. У родительского и у дочернего процессов есть свои собственные (приватные) образы памяти. Если родительский процесс впоследствии изменяет какие-либо свои переменные, то эти изменения остаются невидимыми для дочернего процесса (и наоборот).

1. **Что такое процесс инициализации OS?**

Порождение нового процесса — это длительная процедура, так как ОС должна выполнить множество действий:

1. Выделить процессу необходимые ресурсы (адресное пространство, файлы, устройство и т.д.);
2. Произвести инициализацию этих ресурсов (загрузить выполняемую программу в ОП, инициализировать первое начальное значение регистров и стеков, открыть файлы и т.д.);
3. Занести всю необходимую информацию в специальную таблицу, описывающую процессу в системе;
4. Передать управление новому процессу.
5. **Перечислите области памяти процесса и поясните их назначение.**

**Адресное пространство** — это диапазон адресов, обозначающих определенное место в памяти.

С каждым процессом связано его адресное пространство — список адресов ячеек памяти от нуля до некоторого максимума, откуда процесс может считывать данные и куда может записывать их.

Адресное пространство содержит выполняемую программу, данные этой программы и ее стек. Кроме этого, с каждым процессом связан набор ресурсов, который обычно включает регистры (в том числе счетчик команд и указатель стека), список открытых файлов, необработанные предупреждения, список связанных процессов и всю остальную информацию, необходимую в процессе работы программы.

Таким образом, процесс — это контейнер, в котором содержится вся информация, необходимая для работы программы.

**сегмент heap** – структура данных, с помощью которой реализована динамически распределяемая память приложения;

**сегмент кода** – содержит машинные команды, адресуется регистром CS;

**сегмент данных** – содержит данные, то есть константы и рабочие области, необходимые программе. Адресуется регистром DS;

**сегмент стека** – содержит адреса возврата в точку вызова подпрограмм. Адресуется регистром SS.

1. **Чем отличаются системные процессы от пользовательских?**

Системные процессы практически ничем не отличаются от любых написанных пользователем программ. Единственный критерий, по которому отличают прикладные процессы и системные сервисные процессы, состоит в том, что процесс операционной системы управляет каким-либо ресурсом в интересах прикладного процесса.

1. **Что такое Windows-сервисы, Linux-демоны?**

Процессы, запускаемые автоматически при запуске OS; Windows: windows-сервисы; Linux-демоны.

1. **С помощью каких системных вызовов можно создать дочерний процесс в Windows? Поясните разницу**.

Функция **CreateProcess**создает новый процесс и его главный поток. Новый процесс выполняет указанный исполняемый файл. Функция CreateProcess предназначена для запуска новой программы. Функции WinExec и LoadModule также работают, но не предоставляют таких возможностей, как CreateProcess.

Функция **LoadModule** загружает и исполняет приложение или создает новый экземпляр существующего приложения.

Самый простой способ запустить какую-то программу из своего приложе­ния — использовать функцию **WinExec.**

1. **С помощью каких системных вызовов можно создать дочерний процесс в Linux? Поясните разницу.**

Fork : Вызов fork в основном создает дубликат текущего процесса, идентичный почти во всех отношениях (не все копируется, например, ограничения ресурсов в некоторых реализациях, но идея состоит в том, чтобы создать как можно более близкую копию).

Новый процесс (потомок) получает другой процесс ID (PID) и имеет PID старого процесса (родителя) в качестве своего родителя PID (PPID). Поскольку два процесса теперь работают с одним и тем же кодом, они могут определить, какой из них какой, по коду возврата fork - дочерний получает 0, родительский получает PID дочернего. Это все, конечно, при условии, что вызов fork работает - если нет, дочерний элемент не создается, и родитель получает код ошибки.

Vfork : Основное различие между vfork и fork заключается в том, что при создании нового процесса с помощью vfork() родительский процесс временно приостанавливается, и дочерний процесс может занять адресное пространство родителя. Это странное положение дел продолжается до тех пор, пока дочерний процесс либо не завершит работу, либо не вызовет execve(), после чего родительский процесс продолжится.

Это означает, что дочерний процесс vfork() должен быть осторожен, чтобы избежать неожиданного изменения переменных родительского процесса. В частности, дочерний процесс не должен возвращаться из функции, содержащей вызов vfork(), и он не должен вызывать exit() (если ему нужно выйти, он должен использовать \_exit(); на самом деле это также верно для дочернего элемента обычного fork()).

Exec : Вызов exec - это способ в основном заменить весь текущий процесс новой программой. Он загружает программу в текущее пространство процесса и запускает ее из точки входа. exec() заменяет текущий процесс исполняемым файлом, на который указывает функция. Управление никогда не возвращается к исходной программе, если только не возникает ошибка exec().

Clone : Клон, как fork, создает новый процесс. В отличие от fork, эти вызовы позволяют дочернему процессу совместно использовать части контекста выполнения с вызывающим процессом, такие как пространство памяти, таблица файловых дескрипторов и таблица обработчиков сигналов.

1. **Какие потоки данных доступны любому процессу автоматически?**

потоки имеющие зарезервированные номера - дескрипторы (номера), поток ввода (0), поток вывода (1), поток вывода ошибок (2).

1. **Поясните назначение системного вызова WaitForSingleObject в Windows-приложении.**

Вызов *WaitForSingleObject* используется для ожидания события. Ожидание может касаться множества возможных событий. Если в параметре указан процесс, то вызывающая программа дожидается окончания конкретного процесса.

1. **Поясните назначение системного вызова wait в Linux-приложении.**

Она блокирует вызывающий процесс до тех пор, пока один из его дочерних процессов не завершится (или не произойдет ошибка).

1. **Дайте развернутое определение процесса OS.**

Процесс OS – единица работы OS - объект ядра OS + адресное пространство.

Процессом, по существу, является программа во время ее выполнения.