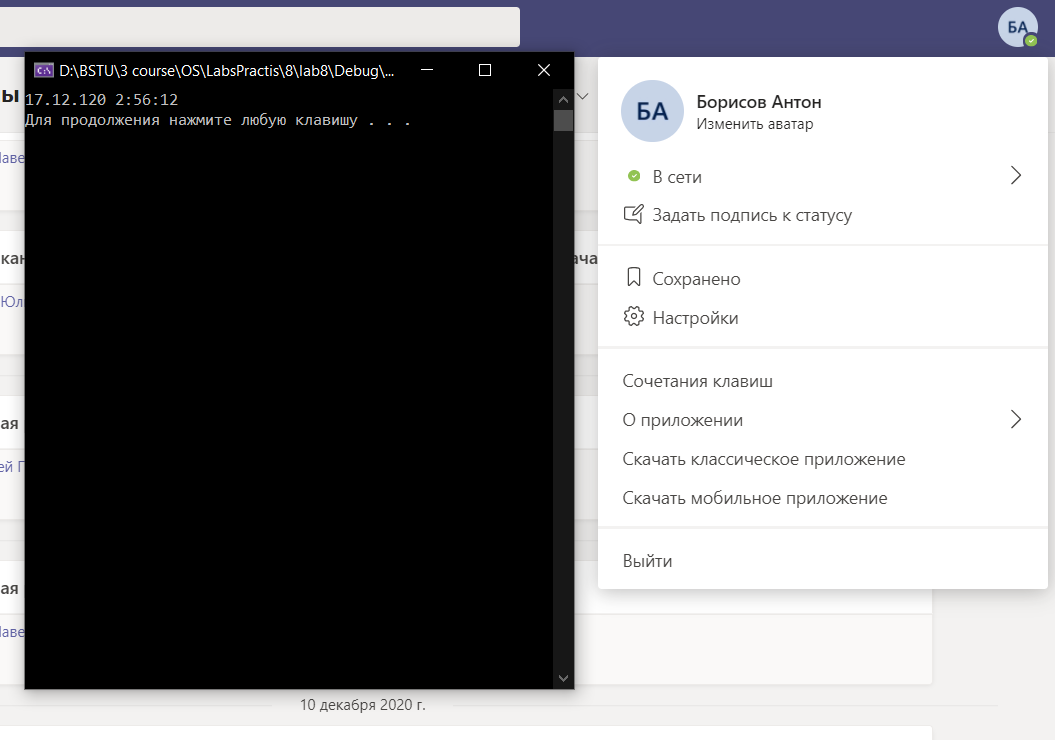
**Задание 1.**

- Скриншот с запуском приложения и результатом.



Расписать использумые функции для работы со временем.

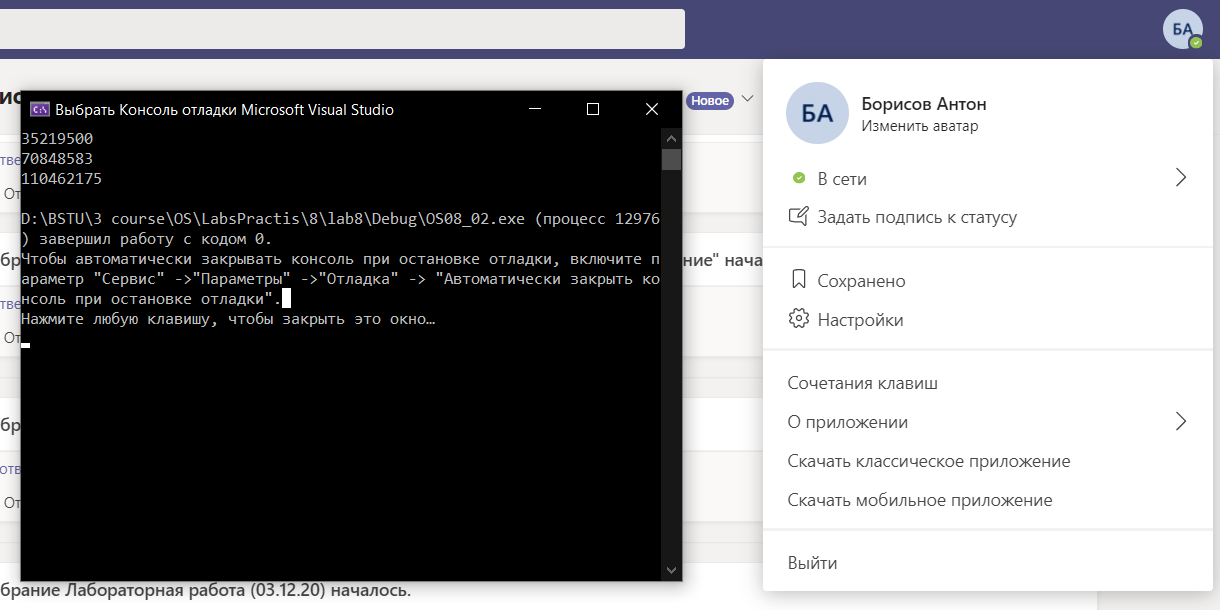
Функция time() возвращает текущее календарное время системы.

Функция time() может вызываться либо с указателем NULL, либо с указателем на переменную time\_t. В последнем случае этот аргумент также получает значение календарного времени.

localtime\_s(const tm\* tm, const time\_t \* time) - функция преобразовывает текущее значение времени, передаваемое как аргумент, через указатель time на time\_t в структуру tm. Время указанное в time используется для заполнения структуры tm значениями текущей даты и времени.

**Задание 2.**

- Скриншот с запуском приложения и результатом.

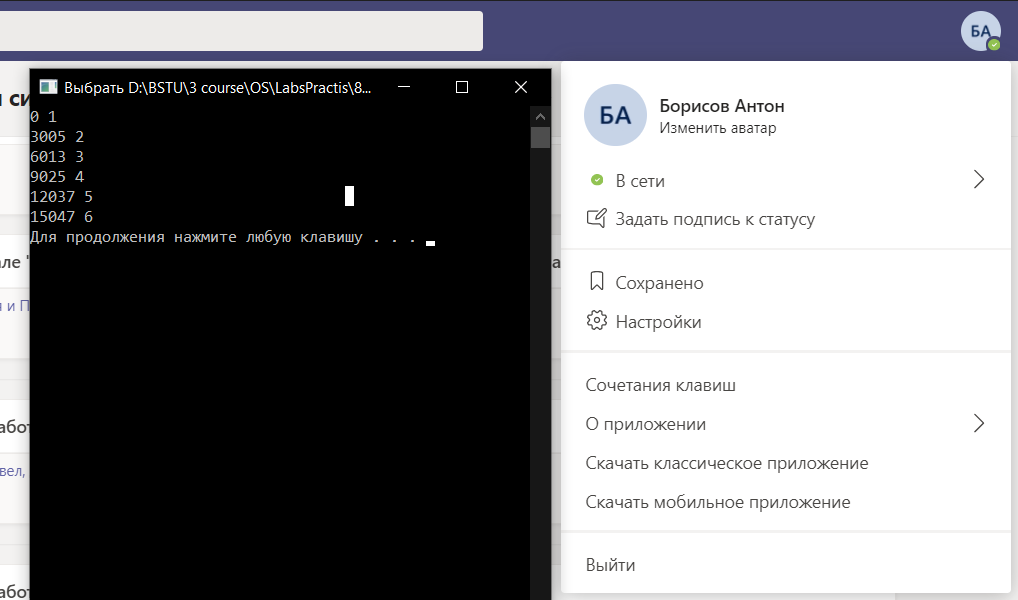


Расписать использумые функции для работы со временем.

clock() - Возвращает количество временных тактов, прошедших с начала запуска программы.

**Задание 3.**

- Скриншот с запуском приложения и результатом.



Расписать использумые функции для работы с таймерами.

**HANDLE CreateWaitableTimer(LPSECURITY\_ATTRIBUTES lpTimerAttributes, BOOL bManualReset, LPCTSTR lpTimerName )** - функция создает ожидающий таймер в занятом состоянии, из которого он выводится принудительно, т.е. после создания объект не активен. Таймер может быть двух типов с автосбросом или ручным сбросом, определяется параметром bManualReset. Если таймер с ручным сбросом, то при переходе в свободное состояние запускаются все потоки, которые его ожидали. Таймер с автосбросом запускает только один поток. Первый параметр обычно всегда NULL, последний - имя таймера. Имя используется для совместного разделения объекта между процессами.

**BOOL SetWaitableTimer(HANDLE hTimer, const LARGE\_INTEGER \*pDueTime,LONG lPeriod,PTIMERAPCROUTINE pfnCompletionRoutine,**  
**PVOID pvArgToCompletionRoutine, BOOL bResume)** - функция запускает таймер и определяет все его параметры

-  Первый параметр - дескриптор таймера.

- Второй параметр **pDueTime** определяет время перехода таймера в свободное состояние (signaled) или время срабатывания таймера

- Третий параметр **lPeriod** определяет режим работы и одновременно период повторения срабатываний ожидающего таймера.

- Четвертый параметр **pfnCompletionRoutine** определяет указатель на необязательную функцию асинхронного вызова (APC), которая помещается в очередь функций APC в момент срабатывания таймера и затем вызывается.

- Пятый параметр передает в функцию асинхронного вызова (APC) произвольный аргумент, например указатель на объект или структуру.  
- Последний параметр **bResume,**если он не нуль, выводит машину из спящего состояния по срабатыванию таймера.

**Задание 4.**

Все процессы при завершении должны выводить прошедшее с начала работы время.

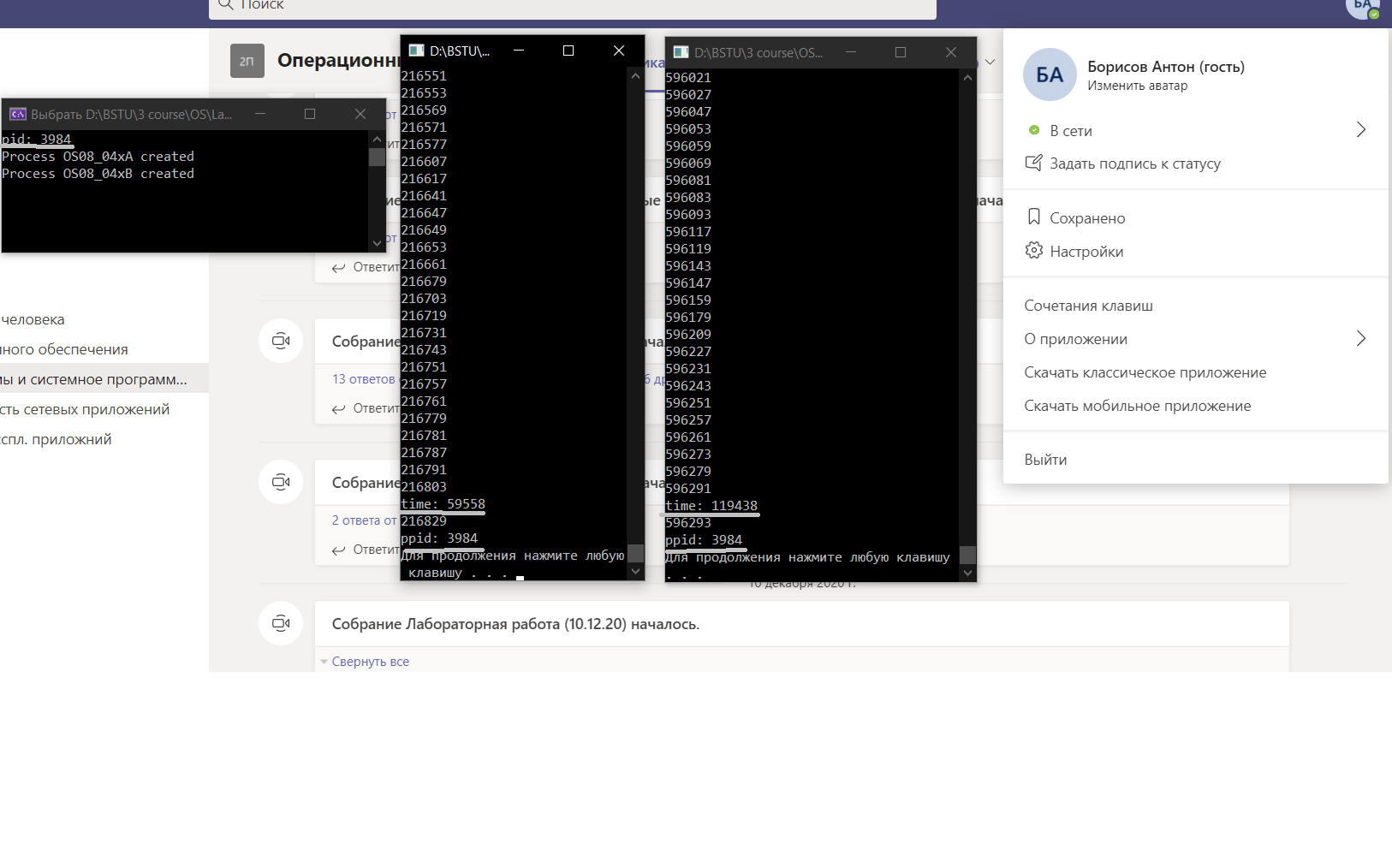
Главный процесс должен в начале вывести свой ID.

Дочерние процессы вначале работы должны вывести родительский PID.

Предыдущие 3 пункта должны быть выделены на скриншотах.

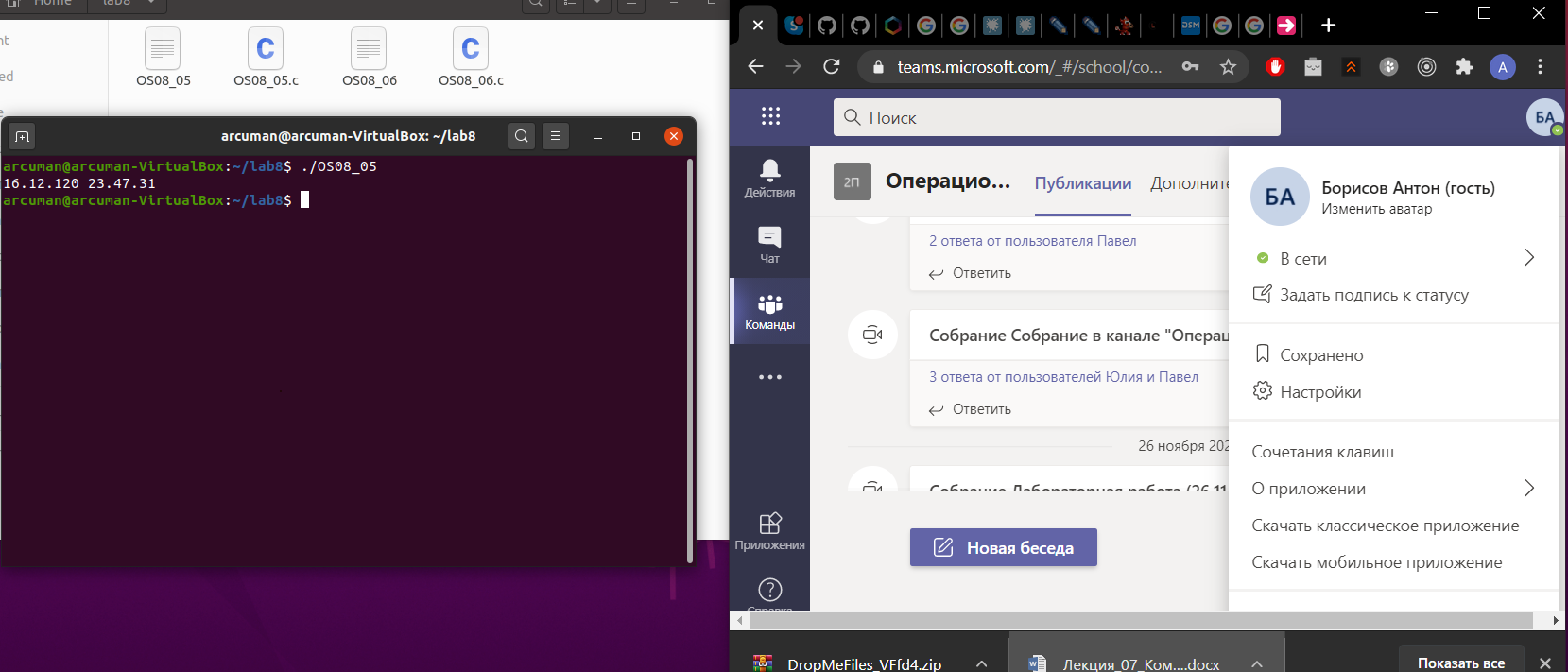
- Скриншот с запуском приложения и результатом.

- 2 скриншота с результатами дочерних процессов.



**Задание 5.**

- Скриншот с запуском приложения и результатом.



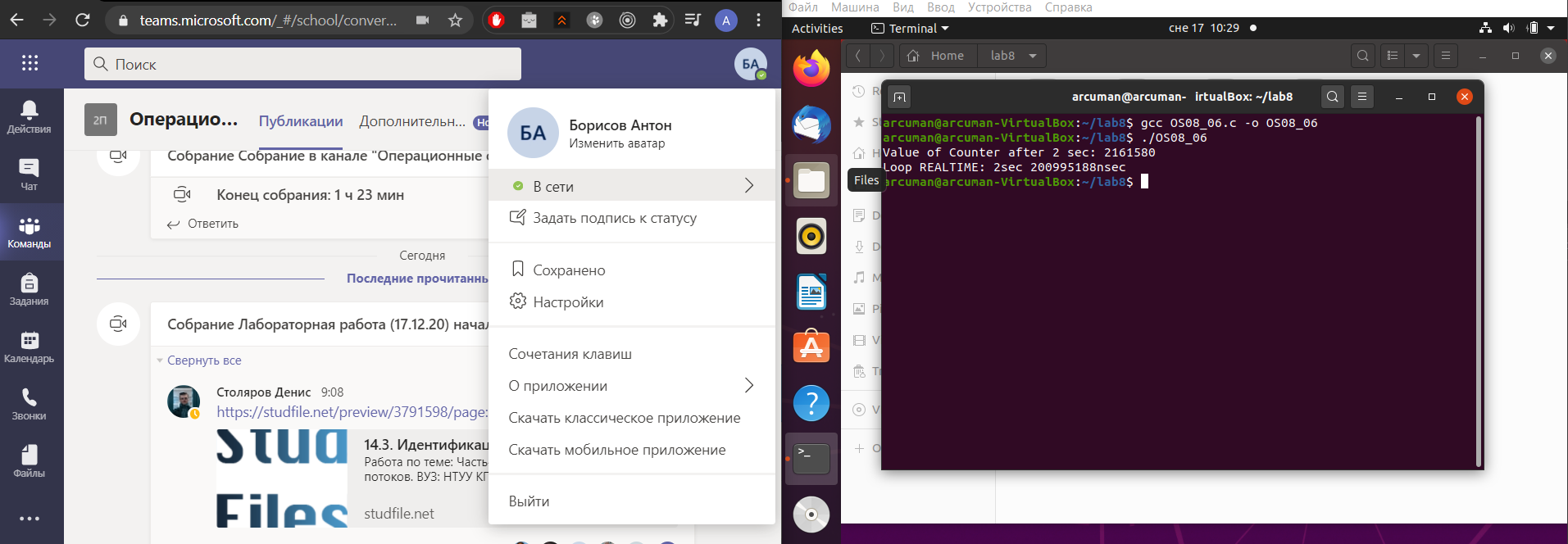
Расписать использумые функции для работы со временем.

time(time\_t \*second) – эта функция возвращает время с 00:00:00 UTC 1 января 1970 г. (временная метка Unix) в секундах. Если second не является нулевым указателем, возвращаемое значение также сохраняется в объекте, на который указывает second.

Библиотечная функция struct tm \* gmtime (const time\_t \* timer) использует значение, указанное таймером, для заполнения структуры tm значениями, которые представляют соответствующее время, выраженное в универсальном координированном времени (UTC) или часовом поясе GMT.

**Задание 6.**

- Скриншот с запуском приложения и результатом.



Расписать использумые функции для работы со временем.

Функция signal(int sig, void(\*func)) устанавливает функцию как обработчик сигнала.

Функция alarm(int sec) – послать от ядра процессу сигнал через заданный интервал времени.

Функция int clock\_gettime (clockid\_t clock, struct timespec \*ts) - получает текущее время по часам, идентифицированным часами, сохраняя его в секундах и наносекундах в \* ts.