

Тестирование Task Manager

1. Цель тестирования

Цель тестирования – проверить корректную интеграцию DLL с GUI, правильное обновление информации, стабильность работы фонового потока сбора данных и корректное освобождение ресурсов при завершении работы приложения.

2. Область применения тестирования

Тестированию подлежат следующие компоненты:

- **Статическая информация:** получение и отображение информации о CPU, памяти и дисках.
- **Динамическая информация:** запуск фонового потока, сбор и обновление данных о процессах в таблице (Treeview).
- **Управление памятью:** корректное освобождение памяти после получения данных.
- **Завершение работы:** остановка фонового потока и корректное закрытие окна приложения.

3. Тестовое окружение

- **Операционная система:** macOS или Windows (с корректно собранной DLL/библиотекой)
- **Версия Python:** 3.7 и выше
- **Библиотеки:**
 - Tkinter (стандартная поставка Python)
 - ctypes (стандартная поставка Python)
- **Путь к библиотеке:** Убедиться, что путь к файлу библиотеки корректен
(/Users/sip/Yandex.Disk.localized/Learning/Тестирование программных модулей/Менеджер задач/sys_info_fn/target/release/libsys_info_fn.dylib)
- **Исходные данные о системе:**
 - Процессор: Intel core i5-5350U 1,8 ГГц 4 ядра
 - Оперативная память: DDR3 8 ГБ
 - Диск: APPLE SSD: Всего 84 ГБ, Свободно 48 ГБ

4. Тестируемые функции DLL

1. **get_cpu_static_info**

Описание: Возвращает указатель на структуру `CpuStaticInfo` с информацией о процессоре.

Ожидаемые данные:

- **brand:** "Intel Core i5-5350U" (или строка, содержащая модель)
- **usage:** Значение загрузки CPU (может варьироваться)
- **frequency:** 1,8 ГГц (после деления на 1000)
- **core_count:** 4

2. **free_cpu_static_info**

Описание: Освобождает память, выделенную для структуры `CpuStaticInfo`.

Тестирование: Вызывается после получения данных, чтобы убедиться, что не происходит утечек памяти.

3. **get_memory_static_info**

Описание: Возвращает структуру `MemoryStaticInfo` с информацией о памяти.

Ожидаемые данные:

- **total:** 8 (ГБ)
- **used:** Значение, зависящее от текущего состояния
- **available:** Значение, зависящее от текущего состояния

4. **get_disk_static_info_array**

Описание: Возвращает массив структур `DiskStaticInfo`, содержащих информацию о дисках.

Ожидаемые данные:

- Для APPLE SSD:
 - **total_space:** 84 (ГБ)
 - **available_space:** 48 (ГБ)

5. **free_disk_static_info_array**

Описание: Освобождает память, выделенную для массива дисковой информации.

6. **start_process_collector**

Описание: Запускает фоновый поток для сбора информации о процессах.

Ожидаемый результат: При первом вызове возвращает `true` (если поток не был запущен ранее).

7. **get_process_info_array**

Описание: Возвращает массив структур `ProcessInfo` с информацией о запущенных процессах.

Тестирование: Количество процессов должно быть больше 0 (при нормальной работе системы).

8. **free_process_info_array**

Описание: Освобождает память, выделенную для массива информации о процессах.

9. `stop_process_collector`

Описание: Останавливает фоновый поток сбора информации о процессах.

Ожидаемый результат: При корректном завершении возвращает `true`.

10. `free_string`

Описание: Освобождает память, выделенную для C-строки.

Тестирование: При передаче нулевого указателя (NULL) функция не должна вызывать ошибок.

11. `kill_process`

Описание: Завершает процесс с указанным идентификатором (PID).

Ожидаемый результат:

- При попытке завершить несуществующий процесс (например, PID 999999) функция должна вернуть `-1`.
- Если процесс найден и успешно завершён, функция возвращает `0`.

4. Тестирование функций

Тестирование функции

PID	Имя	CPU %	Память (МБ)	Чтение (КБ)	Запись (КБ)
2425	Python	18.2	47.30	12.0	8.0
2430	screencapture	8.5	13.20	140.0	0.0
2431	screencapturel	1.1	14.21	40.0	0.0
1290	Code Helper	0.4	50.50	792.0	0.0
1283	Code Helper (P	0.2	468.71	76548.0	4.0
599	Yandex	0.2	193.57	542992.0	204060.0
1269	Code Helper (R	0.1	306.23	48208.0	0.0
1285	Code Helper	0.1	34.30	812.0	0.0
1260	Electron	0.1	132.49	150952.0	13960.0
1261	Obsidian	0.1	87.04	103312.0	320.0

Сценарий 4.1.

Тестирование `get_cpu_static_info` и `free_cpu_static_info`

Шаги:

1. Вызвать функцию `get_cpu_static_info` из DLL.
2. Извлечь данные из полученной структуры.
3. Проверить, что:
 - Строка `brand` содержит подстроку "Intel Core i5-5350U"
 - Значение `frequency` примерно равно 1,8.
 - Значение `core_count` равно 4.
4. Вызвать `free_cpu_static_info` для освобождения памяти.

Ожидаемые результаты:

- Данные соответствуют исходным:
 - Бренд процессора: "Intel Core i5-5350U"
 - Частота: 1,8 ГГц
 - Количество ядер: 4

Сценарий 4.2. Тестирование `get_memory_static_info`

Шаги:

1. Вызвать функцию `get_memory_static_info`.
2. Извлечь значения полей: `total`, `used`, `available`.
3. Проверить, что `total` равно 8 (с учетом перевода в ГБ).

Ожидаемые результаты:

- `total` = 8 (ГБ).
- Значения `used` и `available` находятся в диапазоне [0, 8] и соответствуют реальному состоянию системы.

Сценарий 4.3.

Тестирование `get_disk_static_info_array` и `free_disk_static_info_array`

Шаги:

1. Вызвать функцию `get_disk_static_info_array`.
2. Проверить, что возвращаемый массив содержит хотя бы одну запись.
3. Для каждого диска проверить, что:
 - Имя диска соответствует "APPLE SSD" (или содержит эту подстроку).
 - `total_space` примерно равно 84.
 - `available_space` примерно равно 48.

4. Вызвать `free_disk_static_info_array` для освобождения памяти.

Ожидаемые результаты:

- Диск APPLE SSD отображается с данными:
 - Общий объём: 84 ГБ
 - Свободное место: 48 ГБ

Сценарий 4.4. Тестирование функций, связанных с процессами

4.4.1 Запуск сборщика: `start_process_collector`

Шаги:

1. Вызвать `start_process_collector`.
2. Проверить, что функция возвращает `true`.

Ожидаемый результат:

- Функция возвращает `true` при первом запуске.

4.4.2 Получение информации о процессах: ``get_process_info_array``

Шаги:

1. После запуска сборщика вызвать `get_process_info_array`.
2. Проверить, что:
 - Количество процессов (поле `len`) больше 0.
 - Для каждого процесса поля `pid` и `name` не равны NULL..

PID	Имя	CPU %	Память (МБ)	Чтение (КБ)	Запись (КБ)
2450	Python	17.6	47.52	20.0	8.0
1269	Code Helper (R	0.7	311.29	48512.0	0.0
1290	Code Helper	0.4	55.76	792.0	0.0
1283	Code Helper (P	0.2	468.91	76556.0	4.0
599	Yandex	0.1	195.63	543060.0	204672.0
1260	Electron	0.1	136.71	151672.0	14732.0
1398	cpptools	0.0	53.00	95876.0	0.0
1965	Yandex Helper	0.0	77.50	84.0	0.0
740	Yandex Helper	0.0	7.87	9752.0	0.0
533	AMPLibraryAge	0.0	1.08	26448.0	0.0

Ожидаемый результат:

- Список процессов заполнен данными

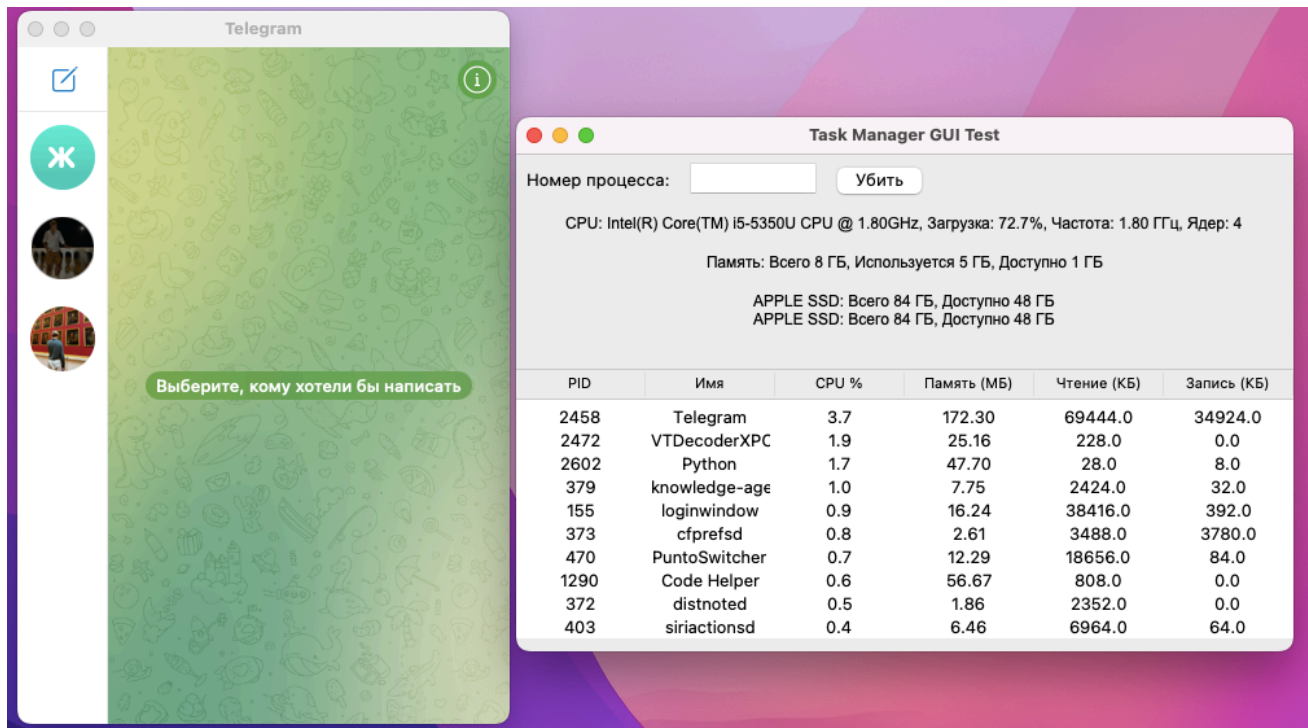
Сценарий 4.5. Тестирование `kill_process`

Шаги:

1. Вызвать `kill_process` с PID, который не существует (например, 999999).
2. Проверить, что функция возвращает `-1`.
3. (Опционально) Для тестирования успешного завершения процесса можно запустить тестовый процесс, получить его PID, затем вызвать `kill_process` и проверить, что возвращается `0`.

Ожидаемый результат:

- При передаче несуществующего PID функция возвращает `-1`.
- В случае успешного завершения процесса возвращается `0`.



- PID Telegram = 2548

Номер процесса:

- Процесс Успешно завершён

PID	Имя	CPU %	Память (МБ)	Чтение (КБ)	Запись (КБ)
2634	screencapture	9.5	13.25	0.0	0.0
2602	Python	5.7	58.81	1060.0	8.0
2635	screencapture	1.3	14.25	0.0	0.0
1290	Code Helper	0.4	56.67	808.0	0.0
599	Yandex	0.3	180.83	548236.0	212052.0
1283	Code Helper (P	0.2	455.02	76576.0	4.0
1266	Obsidian Helpe	0.1	44.96	15932.0	0.0
1260	Electron	0.1	135.18	153068.0	15720.0
1285	Code Helper	0.1	34.63	812.0	0.0
1268	Obsidian Helpe	0.0	261.43	60300.0	1856.0

Баг-репорты

Кейс #0

Машина тестирования: Windows x64

Сценарий использования: Запуск программы. Ожидание загрузки раздела "Процессы" загрузки процессов. Переход в раздел "Производительность". Переход в раздел "Службы".

Проблема: Программа потребляет слишком много ресурсов. Ограничение 5%, наша программа скачет от 0 до 11 и более процентов:

> Python	10,1%	24,6 МБ
> Python	8,5%	46,4 МБ
> Python	7,6%	26,4 МБ
> Python	0,1%	26,3 МБ

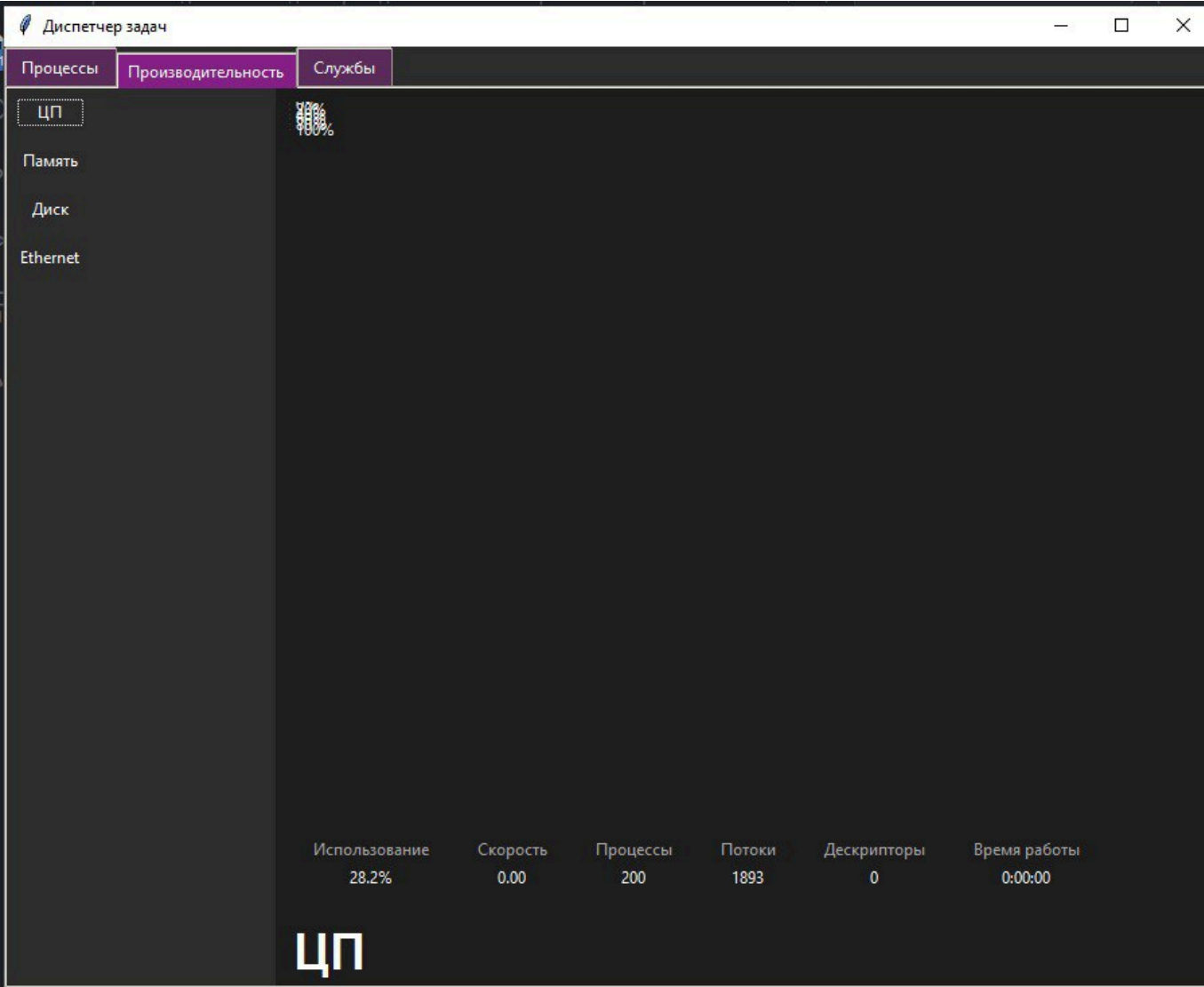
Примечание: переписать функцию `"_update_performance"`, на использование функций из DLL

```
def _update_performance(self):
    system_info = {
        'cpu_percent': psutil.cpu_percent(),
        'memory': {
            'total': psutil.virtual_memory().total,
            'available': psutil.virtual_memory().available
        },
        'disk_usage': psutil.disk_usage('/').percent,
        'network_usage': 0, # Заглушка
        'process_count': len(psutil.pids()),
        'thread_count': sum(p.num_threads() for p in psutil.process_iter()),
        'uptime': time.time() - psutil.boot_time()
    }
    self.performance_tab.update_data(system_info)
```

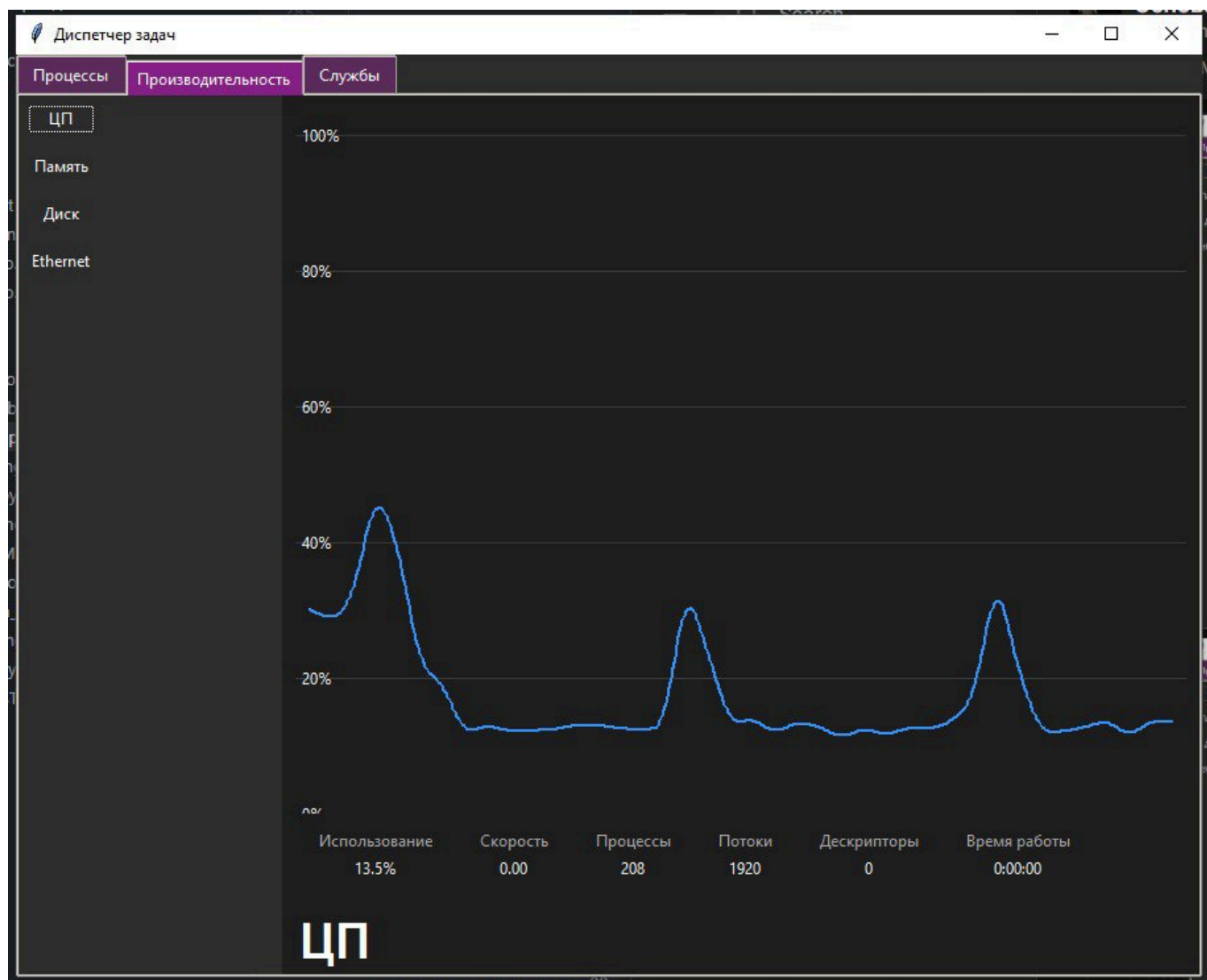
Кейс #1

Машина тестирования: Windows x64

Сценарий использования: Запуск программы. Ожидание загрузки раздела "Процессы" загрузки процессов. Переход в раздел "Производительность" и видим сбившийся график:



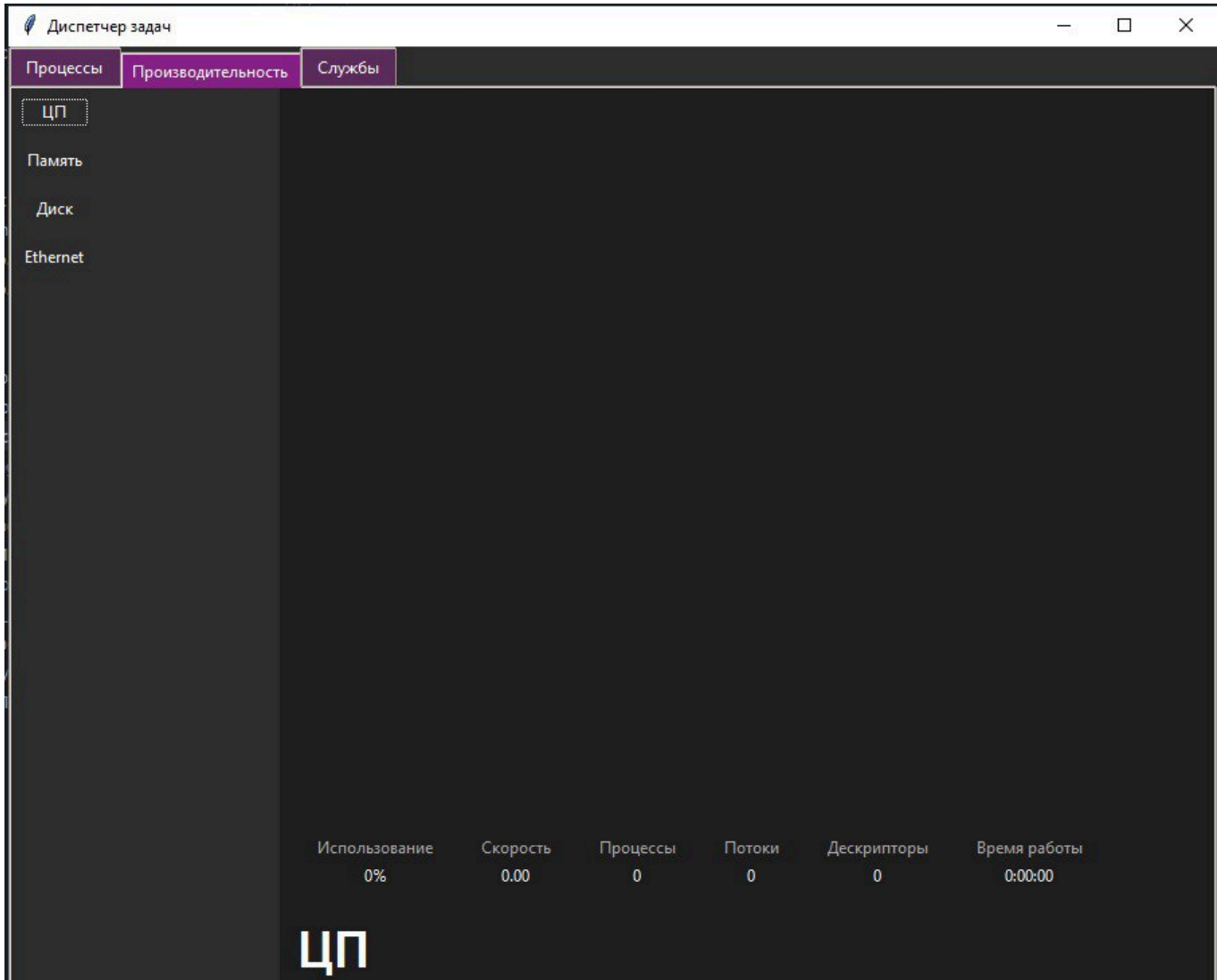
Примечание_1: Позже график начинает выглядеть как нужно



Примечание_2:

Сценарий использования: Запуск программы. Не ожиданем загрузки раздела "Процессы".

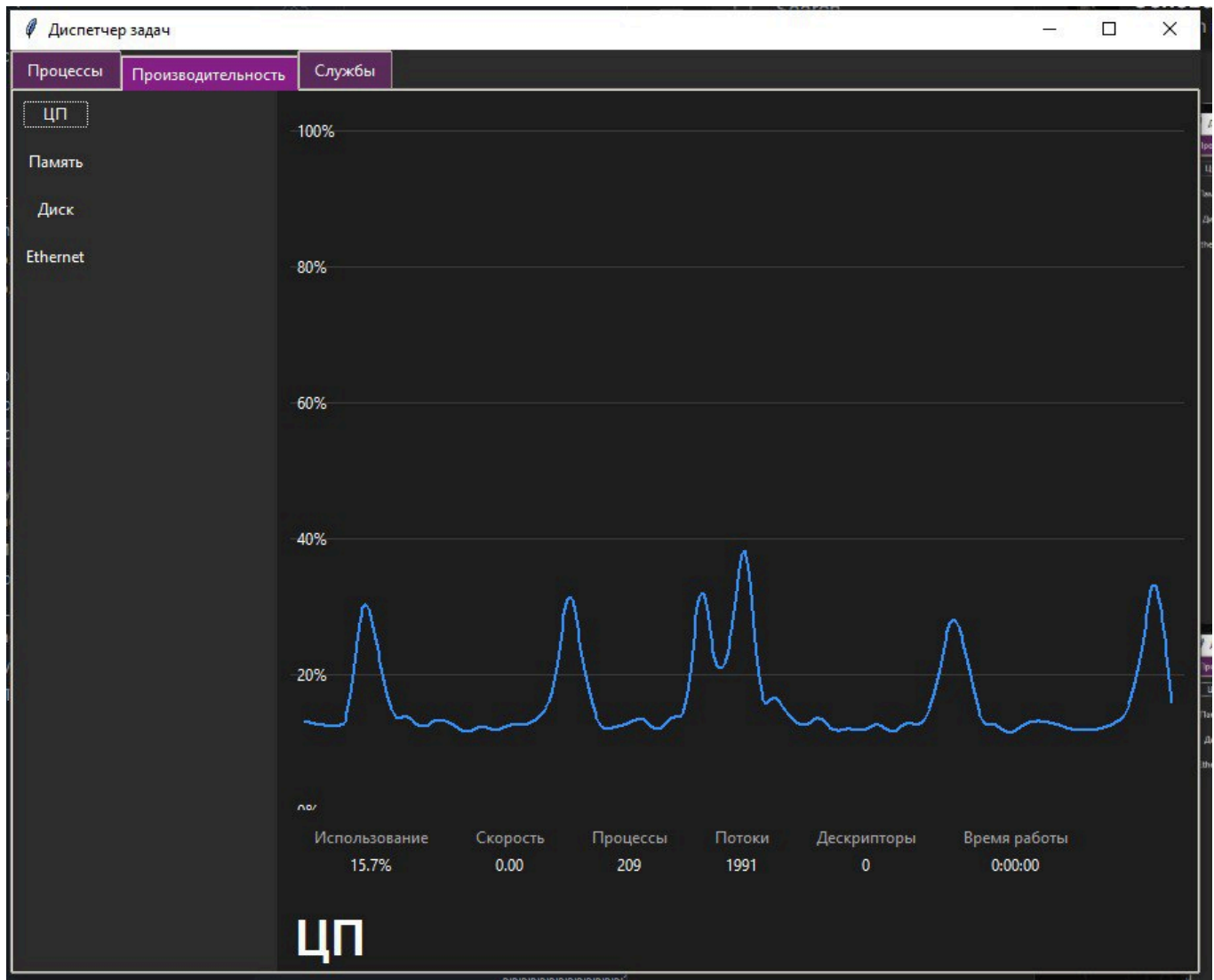
Переход в раздел "Производительность" и видим нормальный график:

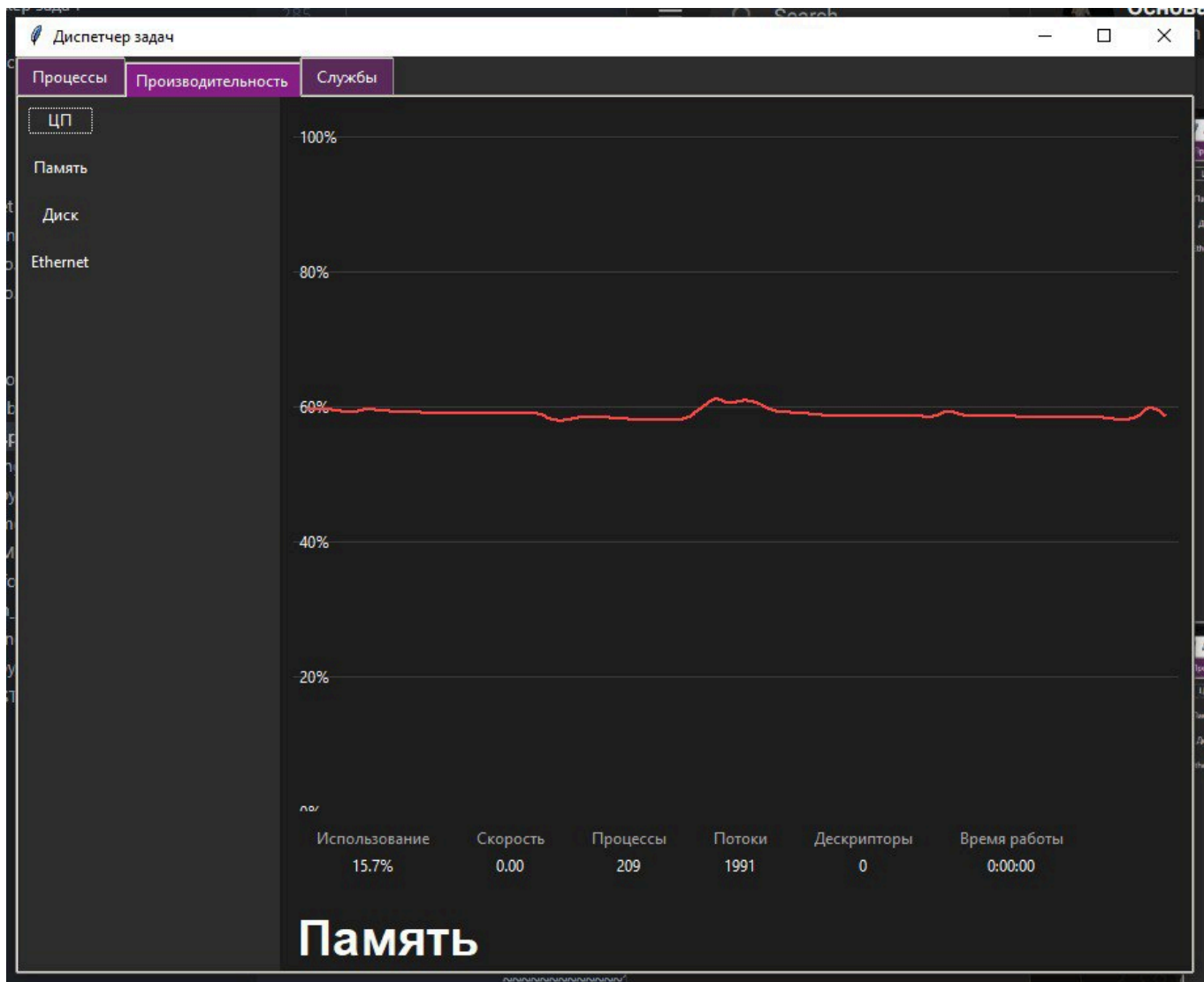


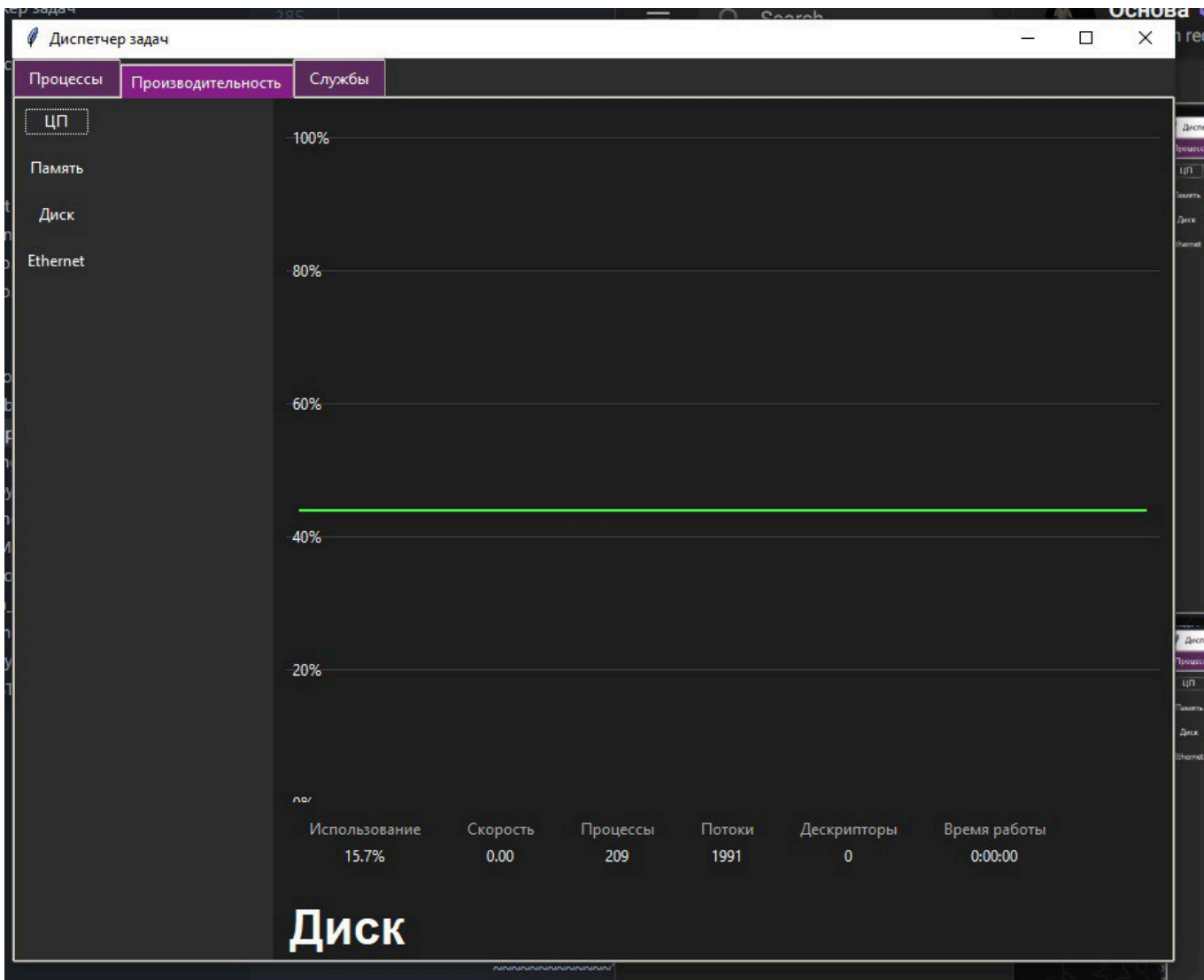
Кейс #2

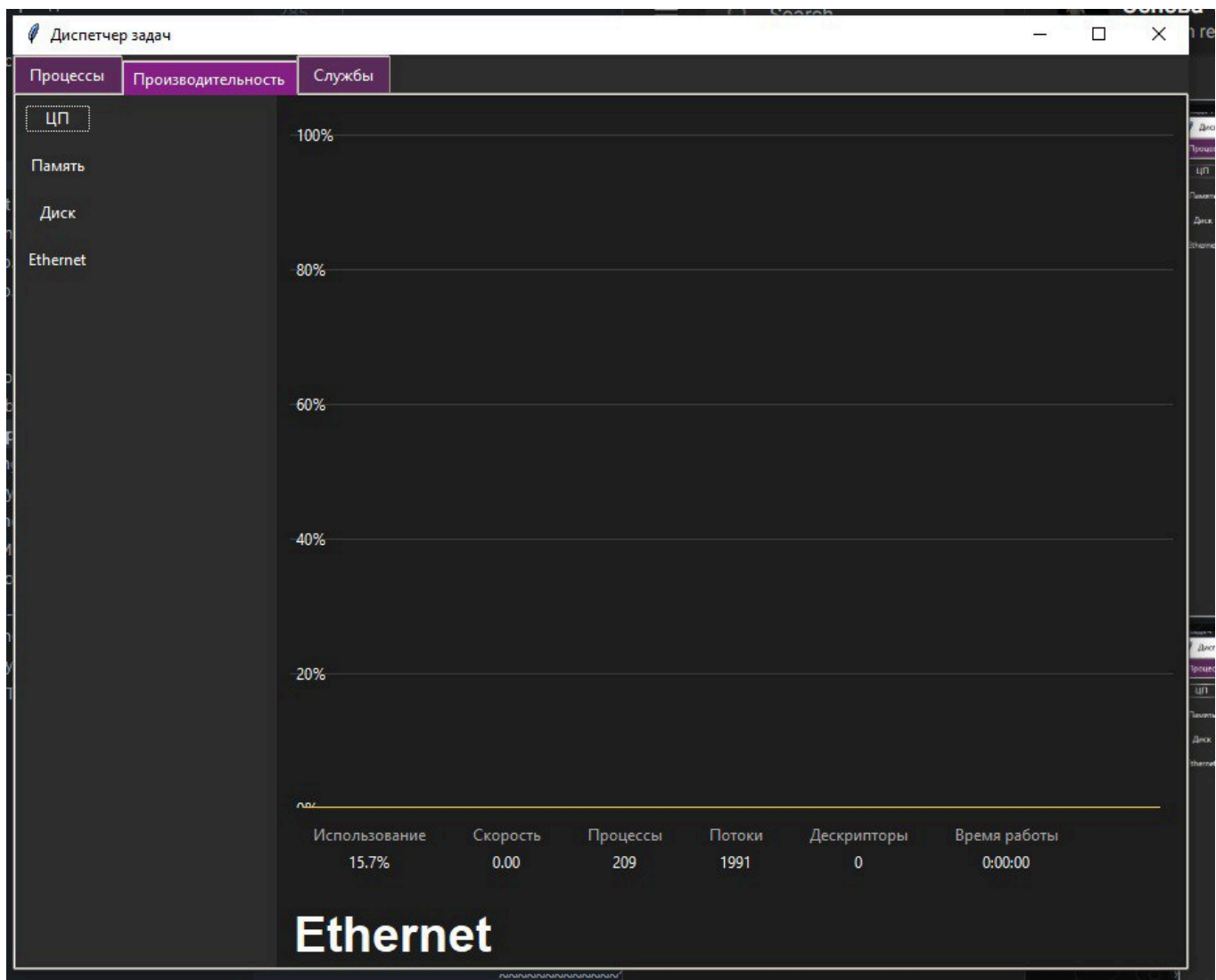
Машина тестирования: Windows x64

Не реализованная функция: Под графиком расположен информационный раздел, который должен изменяться при переходе от вкладки к вкладке. В нынешней версии программы смены информации об производительных модулей не происходит:









Кейс #3

Машина тестирования: Windows x64

Не реализованная функция: Не все блоки модуля "Информация производительных модулей" обновляется: "Скорость", "Время работы."

Использование	Скорость	Процессы	Потоки	Дескрипторы	Время работы
15.7%	0.00	209	1991	0	0:00:00

Примечания: Удалить "Дескрипторы",

Примечания: Добавить эти параметры для процессора(Которая есть в DLL)

Использование	Скорость	Базовая скорость:	3,30 ГГц
29%	3,57 ГГц	Сокетов:	1
Процессы	Потоки	Ядра:	4
205	1979	Логических процессоров:	4
Дескрипторы	Виртуализация:	Включено	
79663	Кэш L1:	256 КБ	
Время работы	Кэш L2:	1,0 МБ	
0:01:51:11	Кэш L3:	6,0 МБ	

Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz

Добавить эти параметры для Памяти (Которая есть в DLL)

Используется (сжатая)	Доступно	Скорость:	1600 МГц
4,6 ГБ (419 МБ)	3,3 ГБ	Использовано гнезд:	2 из 4
Выделено	Кэшировано	Форм-фактор:	DIMM
6,3/9,2 ГБ	3,0 ГБ	Зарезервировано аппаратно:	93,6 МБ
Выгружаемый пул	Невыгружаемый пул		
420 МБ	198 МБ		

Добавить эти параметры для Диска (Которая есть в DLL)

Активное время	Среднее время ответа	Емкость:	477 ГБ
0%	0 мс	Формат:	477 ГБ
Скорость чтения	Скорость записи	Системный диск:	Да
0 КБ/с	0 КБ/с	Файл подкачки:	Да
		Тип:	SSD

Диск 0 (C:)

Netac SSD 512GB

Добавить эти параметры для Ethernet (В DLL пока не реализована, но добавьте заглушки)

Отправка	Адаптер:	Ethernet 8
2,0 Мбит/с	Тип подключения:	Ethernet
Получение	IPv4-адрес:	192.168.9.104
72,0 кбит/с	IPv6-адрес:	-


Ethernet

Intel(R) Ethernet Connection I217-LM

Кейс #4


Машина тестирования: Windows x64

Не реализованная функционал: Добавить раздел с GPU




ЦП

47% 3,56 ГГц




Память

4,7/7,9 ГБ (59%)




Диск 0 (C:) SSD

0%




Ethernet Ethernet 2

0: 0 Гб: 0 кбит/с




Ethernet Ethernet 8

0: 1,0 Гб: 0,1 Мбит/с



Ethernet vEthernet (Default ...)

0: 0 Гб: 0 кбит/с



Графический про Intel(R) HD Graphics 4600

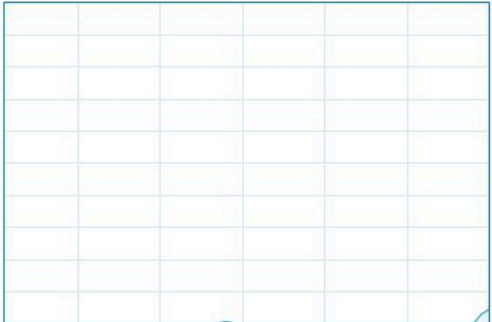
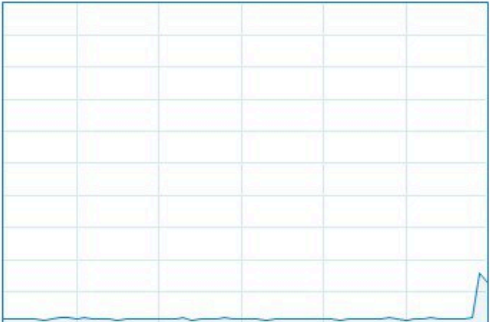
13%

GPU

Intel(R) HD Graphics 4600

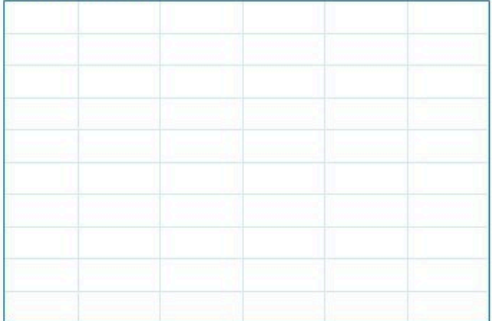
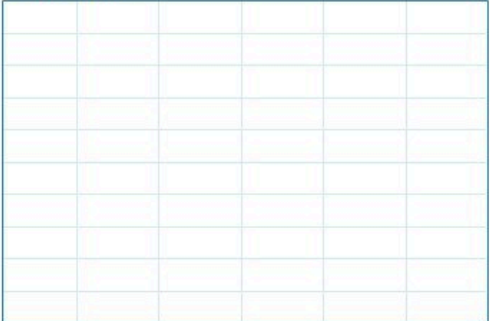
3D

13% Copy 5%



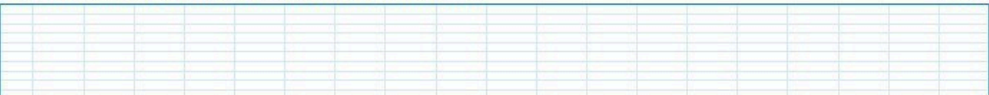
Video Decode

0% Video Processing 0%



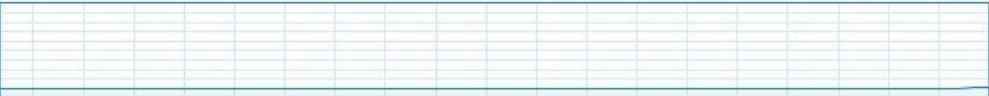
Использование выделенной памяти графического процессора

128 МБ



Использование общей памяти графического процессора

2,0 ГБ



Использование

13%

Оперативная память графического процессора

0,2/2,1 ГБ

Выделенная память графического процессора

0,0/128 МБ

Общая память графического процессора

0,2/2,0 ГБ

Версия драйвера:

Дата разработки:

Версия DirectX:

Физическое располож...

Зарезервированная ап...