

Московский Авиационный Институт  
(Национальный Исследовательский Университет)  
Факультет информационных технологий и прикладной математики  
Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу  
«Операционные системы»**

Студент: Хомяков Иван Андреевич.  
Группа: М8О-207Б-21  
Вариант: -  
Преподаватель: Миронов Евгений Сергеевич  
Оценка: \_\_\_\_\_  
Дата: \_\_\_\_\_  
Подпись: \_\_\_\_\_

Москва, 2022

## **Содержание**

1. Репозиторий
2. Постановка задачи
3. Системные вызовы
4. Демонстрация работы
5. Выводы

## Репозиторий

[https://github.com/EbumbaE/OS\\_LAB/lab1](https://github.com/EbumbaE/OS_LAB/lab1)

## Постановка задачи

### Цель работы

Приобретение практических навыков диагностики работы программного обеспечения.

### Задание

При выполнении последующих лабораторных работ необходимо продемонстрировать ключевые системные вызовы, которые в них используются и то, что их использование соответствует варианту ЛР.

## Системные вызовы

1. `CreateFileMapping` – создает или открывает именованный или безымянный объект сопоставления файлов для указанного файла. Нам для задания необходимо указывать имя, размер буфера и флаг `INVALID_HANDLE_VALUE` для поддержки системным файлом подкачки, а не файлом в файловой системе.
2. `MapViewOfFile` – сопоставляет представление сопоставления файла с адресным пространством вызывающего процесса, передаем в нее наш файловый дескриптор, `FILE_MAP_ALL_ACCESS` – сопоставляем представление файла для чтения/записи.
3. `OpenFileMapping` - открывает именованный объект сопоставления файлов. Указываем имя и тип разрешения.
4. `_tcsncpy` – копирует значение строки в переменную типа `TCHAR`.
5. `CreateEvent` – создает именованный или неименованный объект события. В нашем случае указываем имя и стандартный дескриптор безопасности.
6. `OpenEvent` – открывает существующий именованный объект события. Указываем имя и право доступа `EVENT_ALL_ACCESS` для всех прав доступа к ивенту.
7. `SetEvent` – устанавливает указанный объект события в сигнальное состояние. Нам эта функция нужна для синхронизации и общения ребенка и родителя.
8. `WaitForSingleObject` - ожидает, пока указанный объект не перейдет в сигнальное состояние или пока не истечет время ожидания. Указываем handler ивента и время ожидания `INFINITE`. Используем эту функцию, чтобы дождаться завершения нашего ребенка.
9. `WaitForMultipleObjects` - ожидает, пока один или все указанные объекты не перейдут в сигнальное состояние или пока не истечет время ожидания. Указываем количество, массив дескрипторов событий и время ожидания `INFINITE`. Используем эту функцию, чтобы




ребенок определился, выполняться ему или сразу закрыться. Т.к. мы указали дескриптор запуска ребенка первым, достаточной проверкой будет 'возвращаемый функцией код != WAIT\_OBJECT\_0'. Т.е., если нам вернется все, кроме кода запуска ребенка, мы закрываем его.

10. CreateProcess – создает процесс, принимает кучу параметров, важные из них (szCmdline – командная строка, запускающая child.exe; TRUE – наследуем pipe; siStartInfo – сами pipe; siProcInfo – устанавливаем информацию о процессе)
11. CloseHandle – закрываем handle

## Демонстрация работы

### CMD:

```
C:\Users\Ivanh\Documents\OS_LAB\lab4\build>drstrace -- OS_LAB
<Using system call file C:\Users\Ivanh\AppData\Roaming\Dr.
Memory\symcache\syscalls_x64.txt>
<drstrace log file is .\drstrace.OS_LAB.exe.02588.0000.log>
<Using system call file C:\Users\Ivanh\AppData\Roaming\Dr.
Memory\symcache\syscalls_x64.txt>
<drstrace log file is .\drstrace.CHILD1.exe.14932.0000.log>
<Using system call file C:\Users\Ivanh\AppData\Roaming\Dr.
Memory\symcache\syscalls_x64.txt>
<drstrace log file is .\drstrace.CHILD2.exe.06456.0000.log>
all is ok
```

 drstrace.CHILD1.exe.14932.0000.log	05.01.2023 20:15	Текстовый докум...	5 КБ
 drstrace.CHILD2.exe.06456.0000.log	05.01.2023 20:15	Текстовый докум...	18 КБ
 drstrace.OS_LAB.exe.02588.0000.log	05.01.2023 20:15	Текстовый докум...	88 КБ

### OS\_LAB:

#### NtQueryVirtualMemory

```
arg 0: 0xffffffffffffffff (type=HANDLE, size=0x8)
arg 1: 0x00000000004023e0 (type=void *, size=0x8)
arg 2: 0x0 (type=int, size=0x4)
arg 3: 0x0000000000061fb50 (type=<struct>*, size=0x8)
arg 4: 0x30 (type=unsigned int, size=0x8)
arg 5: 0x0000000000061fb00 (type=unsigned int*, size=0x8)
```

succeeded =>

```
arg 3: <NYI> (type=<struct>*, size=0x8)
arg 5: 0x0000000000061fb00 => 0x30 (type=unsigned int*, size=0x8)
retval: 0x0 (type=NTSTATUS, size=0x4)
```

.....

#### NtMapViewOfSection

```
arg 0: 0xd8 (type=HANDLE, size=0x8)
arg 1: 0xffffffffffffffff (type=HANDLE, size=0x8)
arg 2: 0x0000000000061fcb8 => 0x0000000000000000 (type=void **, size=0x8)
arg 3: 0x0 (type=unsigned int, size=0x4)
arg 4: 0x0 (type=unsigned int, size=0x4)
```

```

    arg 5: 0x0 (type=LARGE_INTEGER*, size=0x8)
    arg 6: 0x0000000000061fcc0 => 0x100 (type=unsigned int*, size=0x8)
    arg 7: 0x1 (type=int, size=0x4)
    arg 8: 0x0 (type=named constant, size=0x4)
    arg 9: PAGE_READWRITE (type=named constant, value=0x4, size=0x4)
succeeded =>
    arg 2: 0x0000000000061fcb8 => 0x00000000000180000 (type=void **, size=0x8)
    arg 5: 0x0 (type=LARGE_INTEGER*, size=0x8)
    arg 6: 0x0000000000061fcc0 => 0x1000 (type=unsigned int*, size=0x8)
    retval: 0x0 (type=NTSTATUS, size=0x4)
NtCreateEvent
    arg 0: 0x0000000000061fc80 (type=HANDLE*, size=0x8)
    arg 1: 0x1f0003 (type=unsigned int, size=0x4)
    arg 2: len=0x30, root=0x6c, name=26/28 "WaitWorkChild", att=0x80,
sd=0x0000000000000000, sqos=0x0000000000000000 (type=OBJECT_ATTRIBUTES*,
size=0x8)
    arg 3: 0x1 (type=int, size=0x4)
    arg 4: 0x0 (type=bool, size=0x1)
succeeded =>
    arg 0: 0x0000000000061fc80 => 0xe0 (type=HANDLE*, size=0x8)
    retval: 0x0 (type=NTSTATUS, size=0x4)
.....
NtMapViewOfSection
    arg 0: 0xec (type=HANDLE, size=0x8)
    arg 1: 0xfffffffffffffff (type=HANDLE, size=0x8)
    arg 2: 0x00000000000758d10 => 0x00000000000000000 (type=void **, size=0x8)
    arg 3: 0x0 (type=unsigned int, size=0x4)
    arg 4: 0x0 (type=unsigned int, size=0x4)
    arg 5: <null> (type=LARGE_INTEGER*, size=0x8)
    arg 6: 0x00000000000758c70 => 0x0 (type=unsigned int*, size=0x8)
    arg 7: 0x1 (type=int, size=0x4)
    arg 8: MEM_ROTATE (type=named constant, value=0x800000, size=0x4)
    arg 9: PAGE_EXECUTE_WRITECOPY (type=named constant, value=0x80, size=0x4)
succeeded =>
    arg 2: 0x00000000000758d10 => 0x00007ffcc9dd0000 (type=void **, size=0x8)
    arg 5: <null> (type=LARGE_INTEGER*, size=0x8)
    arg 6: 0x00000000000758c70 => 0x9c000 (type=unsigned int*, size=0x8)
    retval: 0x0 (type=NTSTATUS, size=0x4)
.....
NtProtectVirtualMemory
    arg 0: 0xfffffffffffffff (type=HANDLE, size=0x8)
    arg 1: 0x0000000000061d690 => 0x00007ffccb6b1000 (type=void **, size=0x8)
    arg 2: 0x0000000000061d688 => 0x3520 (type=unsigned int*, size=0x8)
    arg 3: PAGE_READONLY (type=named constant, value=0x2, size=0x4)
    arg 4: 0x0000000000061d680 (type=unsigned int*, size=0x8)
succeeded =>
    arg 1: 0x0000000000061d690 => 0x00007ffccb6b1000 (type=void **, size=0x8)
    arg 2: 0x0000000000061d688 => 0x4000 (type=unsigned int*, size=0x8)
    arg 4: 0x0000000000061d680 => 0x4 (type=unsigned int*, size=0x8)
    retval: 0x0 (type=NTSTATUS, size=0x4)

```

.....

#### NtCreateUserProcess

arg 0: 0x0000000000061e4f8 (type=HANDLE\*, size=0x8)  
arg 1: 0x0000000000061e560 (type=HANDLE\*, size=0x8)  
arg 2: 0x2000000 (type=unsigned int, size=0x4)  
arg 3: 0x2000000 (type=unsigned int, size=0x4)  
arg 4: <null> (type=OBJECT\_ATTRIBUTES\*, size=0x8)  
arg 5: <null> (type=OBJECT\_ATTRIBUTES\*, size=0x8)  
arg 6: 0x204 (type=unsigned int, size=0x4)  
arg 7: 0x1 (type=bool, size=0x1)  
arg 8: <NYI> (type=<struct>\*, size=0x8)  
arg 9: <NYI> (type=<unknown>\*, size=0x8)  
arg 10: <NYI> (type=<struct>\*, size=0x8)

succeeded =>

arg 0: 0x0000000000061e4f8 => 0xf8 (type=HANDLE\*, size=0x8)  
arg 1: 0x0000000000061e560 => 0xf4 (type=HANDLE\*, size=0x8)  
retval: 0x0 (type=NTSTATUS, size=0x4)

#### CHILD1:

##### NtQueryVirtualMemory

arg 0: 0xfffffffffffffff (type=HANDLE, size=0x8)  
arg 1: 0x00000000000402310 (type=void \*, size=0x8)  
arg 2: 0x0 (type=int, size=0x4)  
arg 3: 0x0000000000061fb50 (type=<struct>\*, size=0x8)  
arg 4: 0x30 (type=unsigned int, size=0x8)  
arg 5: 0x0000000000061fb00 (type=unsigned int\*, size=0x8)

succeeded =>

arg 3: <NYI> (type=<struct>\*, size=0x8)  
arg 5: 0x0000000000061fb00 => 0x30 (type=unsigned int\*, size=0x8)  
retval: 0x0 (type=NTSTATUS, size=0x4)

##### NtQueryVirtualMemory

arg 0: 0xfffffffffffffff (type=HANDLE, size=0x8)  
arg 1: 0x00000000000402310 (type=void \*, size=0x8)  
arg 2: 0x3 (type=int, size=0x4)  
arg 3: 0x0000000000061fb80 (type=<struct>\*, size=0x8)  
arg 4: 0x30 (type=unsigned int, size=0x8)  
arg 5: 0x00000000000000000 (type=unsigned int\*, size=0x8)

succeeded =>

arg 3: <NYI> (type=<struct>\*, size=0x8)  
arg 5: 0x00000000000000000 (type=unsigned int\*, size=0x8)  
retval: 0x0 (type=NTSTATUS, size=0x4)

.....

#### CHILD2:

.....

##### NtReadFile

arg 0: 0xec (type=HANDLE, size=0x8)  
arg 1: 0x0 (type=HANDLE, size=0x8)  
arg 2: <null> (type=<function>, size=0x8)  
arg 3: 0x00000000000000000 (type=void, size=0x8)

```

    arg 4: 0x00000000000061fba0 (type=IO_STATUS_BLOCK*, size=0x8)
    arg 5: 0x000000000000b5e00 (type=void*, size=0x8)
    arg 6: 0x1000 (type=unsigned int, size=0x4)
    arg 7: <null> (type=LARGE_INTEGER*, size=0x8)
    arg 8: 0x0000000000000000 (type=unsigned int*, size=0x8)
succeeded =>
    arg 4: status=0x0, info=0xa (type=IO_STATUS_BLOCK*, size=0x8)
    arg 5: 0x000000000000b5e00 => 0x76760a0d64646161 (type=void*, size=0x8)
    retval: 0x0 (type=NTSTATUS, size=0x4)
NtReadFile
    arg 0: 0xec (type=HANDLE, size=0x8)
    arg 1: 0x0 (type=HANDLE, size=0x8)
    arg 2: <null> (type=<function>, size=0x8)
    arg 3: 0x0000000000000000 (type=void, size=0x8)
    arg 4: 0x00000000000061fba0 (type=IO_STATUS_BLOCK*, size=0x8)
    arg 5: 0x000000000000b5e00 (type=void*, size=0x8)
    arg 6: 0x1000 (type=unsigned int, size=0x4)
    arg 7: <null> (type=LARGE_INTEGER*, size=0x8)
    arg 8: 0x0000000000000000 (type=unsigned int*, size=0x8)
failed (error=0xc0000011) =>
    arg 4: status=0xc0000011, info=0x0 (type=IO_STATUS_BLOCK*, size=0x8)
    arg 5: 0x000000000000b5e00 => 0x6576760a64646161 (type=void*, size=0x8)
    retval: 0xc0000011 (type=NTSTATUS, size=0x4)
NtSetInformationProcess
    arg 0: 0xffffffffffffffff (type=HANDLE, size=0x8)
    arg 1: 0x47 (type=int, size=0x4)
    arg 2: 0x00000000000061fc30 (type=<struct>*, size=0x8)
    arg 3: 0x4 (type=unsigned int, size=0x4)
succeeded =>
    retval: 0x0 (type=NTSTATUS, size=0x4)
.....

```

## Выводы

drstrace – инструмент отслеживания системных вызовов для Windows. Он использует Dr. Memory Framework для мониторинга всех системных вызовов, выполняемых целевым приложением. drstrace может быть очень полезен при отладке программ. Также Dr. Memory предоставляет режим нечеткого тестирования, который многократно выполняет одну функцию в целевом приложении, изменяя значение одного аргумента перед каждой итерацией.