Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 3

по курсу «Операционные системы»

СПИСОК ПРОЦЕССОВ В СИСТЕМЕ

Студент группы ИУ9-42Б Нащекин Н. Д.

Преподаватель: Брагин А. В.

Москва, 2024

1 Содержание

2 Цель

Разработать загружаемые модули ядра (драйвера) для операционных систем Windows / ReactOS и NetBSD, которые выводят в отладочный журнал список всех процессов в системе.

3 Постановка задачи

ЧАСТЬ 1. ВЫВОД СПИСКА ПРОЦЕССОВ ИЗ ДРАЙВЕРА В REACTOS

На основе драйвера из лабораторной работы № 2 создать новый драйвер, который выводит в отладочный лог список процессов в системе, используя API ядра. Список должен состоять из имени процесса (имя образа – Image Name), идентификатора процесса (PID) и идентификатора родительского процесса. Драйвер должен загружаться и выгружаться по требованию так, чтобы можно было проверить соответствие списка процессов, выведенного драйвером и с помощью утилиты Task Manager (taskmgr.exe).
Получение информации о использованием процессах из ядра осуществить API с функции ZwQuerySystemInformation(SystemProcessInformation,. . .).
Выделение и освобождение памяти для информации о процессах реализовать используя API функции ExAllocatePool(PagedPool,. . .) и ExFreePool().
Предусмотреть ситуацию, в которой запрошенный необходимый размер буфера для информации о процессах может измениться в большую сторону между вызовами ZwQuerySystemInformation и в этом случае произвести новое выделение памяти с освобождением старого буфера.

ЧАСТЬ 2. ВЫВОД СПИСКА ПРОЦЕССОВ ИЗ ДРАЙВЕРА В NETBSD

На основе драйвера из лабораторной работы № 2 создать новый драйвер, который выводит в отладочный лог список процессов в системе, используя API ядра. Список должен состоять из имени процесса (имя образа – Image Name), идентификатора процесса (PID) и идентификатора родительского процесса. Драйвер должен загружаться и выгружаться по требованию так, чтобы можно было проверить соответствие списка процессов, выведенного драйвером и с помощью утилиты ps.
Получение информации о процессах из ядра осуществить используя глобальную переменную allproc. Для этого необходимо включить заголовочные файлы:

```
1 #include <sys/module.h>
2 #include <sys/kernel.h>
3 #include <sys/proc.h>
```

4 Практическая реализация

ЧАСТЬ 1

Аналогично лабораторной работе 2, в скачанном ранее каталоге с исходным кодом ReactOS[1] в папке reactos/drivers я создал папку lab3_driver. В drivers/CMakeLists.txt добавил строку

```
add_subdirectory(lab3_driver)
```

для того, чтобы мой драйвер участвовал в сборке системы. В папке drivers/lab3_driver были созданы три файла: CMakeLists.txt, lab3_driver.c и lab3_driver.rc. Содержимое этих файлов было создано на основе моего драйвера из прошлой лабораторной. Содержимое CMakeLists.txt:

```
add_library(lab3_driver MODULE lab3_driver.c lab3_driver.rc)
set_module_type(lab3_driver kernelmodedriver)
add_importlibs(lab3_driver ntoskrnl)
add_cd_file(TARGET lab3_driver DESTINATION reactos/system32/drivers FOR all)
```

В основном файле выделяется 32 килобайта паймяти под массив, в котором будут хранится структуры, описывающие процесс. Затем, после выполнения функции ZwQuerySystemInformation[2], если размера буфера не хватает, он увеличивается в 8 раз. После этого программа проходит по полученному массиву структур и для каждого процесса выводит информацию о нём в лог системы. lab3_driver.c:[3],[4]

```
#include <ntddk.h>
#include <debug.h>
#include <ntifs.h>
#include <ndk/exfuncs.h>
#include <ndk/ketypes.h>
#include <ntstrsafe.h>

static DRIVER_UNLOAD DriverUnload;
static VOID NTAPI
DriverUnload(IN PDRIVER_OBJECT DriverObject)
{
    IoDeleteDevice(DriverObject->DeviceObject);
        DPRINT1("NASHCHEKIN NIKITA'S Driver unloaded\n");
```

```
}

NTSTATUS NTAPI DriverEntry(IN PDRIVER_OBJECT DriverObject,
        IN PUNICODE_STRING RegistryPath) {
    /* For non-used parameter */
    UNREFERENCED_PARAMETER(RegistryPath);

    /* Setup the Driver Object */
    DriverObject->DriverUnload = DriverUnload;

        ///Основная функция
    DPRINT1("NASHCHEKIN NIKITA!!! From second driver\n");

    ULONG return_length = 1024*32;
    PVOID pBuffer = ExAllocatePool(NonPagedPool, return_length);
    NTSTATUS status;

    if (pBuffer == NULL) {
        DPRINT1("Ошибка при выделении памяти\n");
        goto end;
    }

    status = ZwQuerySystemInformation(SystemProcessInformation, pBuffer, return_length, &return_length);

    if (!NT_SUCCESS(status)) {
        DPRINT1("Ошибка при выполнении ZwQuery...\n");
        goto end;
    }

    while (ZwQuerySystemInformation(SystemProcessInformation, pBuffer, return_length, &return_length) == STA
        ExFreePool(pBuffer);
        return_length = return_length * 8;
        pBuffer = ExAllocatePool(NonPagedPool, return_length);
    }

    PSYSTEM_PROCESS_INFORMATION process_info = (PSYSTEM_PROCESS_INFORMATION)pBuffer;

    while (process_info) {
        DPRINT1("Имя процесса: %.*ls, ", process_info->ImageName.Length / sizeof(WCHAR), process_info->

        DPRINT1("PID: %lu, ", (ULONG)process_info->UniqueProcessId);

        DPRINT1("Родительский процесс: %lu\n", (ULONG)process_info->InheritedFromUniqueProcessId);

        if (process_info->NextEntryOffset == 0) {
        break;
      }

      process_info = (PSYSTEM_PROCESS_INFORMATION)((PUCHAR)process_info +
       process_info->NextEntryOffset);
```

```
    }

    end:

    /* Return success. */
    return STATUS_SUCCESS;
}
```

lab3_driver.rc:

```
#define REACTOS_VERSION_DLL
#define REACTOS_STR_FILE_DESCRIPTION  "my second driver"
#define REACTOS_STR_INTERNAL_NAME     "lab3_driver"
#define REACTOS_STR_ORIGINAL_FILENAME "lab3_driver.sys"
#include <reactos/version.rc>
```

Далее в среде сборки был пересобран образ, а затем переустановлена система. В работающей системе были выполнены команды:

```
sc create lab3driver binPath=C:\ReactOS\system32\drivers\lab3_driver.sys type= kernel
sc start lab3driver
```

В логи был выведен список процессов, запущенных в системе. Для каждого процесса было напечатано его название, process ID и process ID родительского процесса. Содержимое этого списка аналогично списку, предоставляемому программой taskmgr.exe

ЧАСТЬ 2

Как и в предыдущей лабораторной, сначала был создан файл /usr/src/sys/dev/lab3.c и добавлена реализация драйвера:[5]

```
#include <sys/module.h>
MODULE(MODULE_CLASS_MISC, lab3, NULL);

static int lab3_modcmd(modcmd_t cmd, void* arg) {
        printf("NASHCHEKIN NIKITA From second NetBSD driver!!!");

        struct proc *process;

        PROCLIST_FOREACH(process, &allproc) {
```

```
10                      printf("Process name: %s, ", process->p_comm);
11                      printf("PID: %d, ", process->p_pid);
12                      printf("Process parent: %d\n", process->p_ppid);
13              }
14
15              return 0;
16      }
```

Далее был создан файл /usr/src/sys/modules/lab3/Makefile со следующим содержимым:

```
1   .include "../Makefile.inc"
2   KMOD=lab3
3   .PATH: ${S}/dev
4   SRCS=lab3.c
5   .include <bsd.kmodule.mk>
```

Затем была выполнена команда make и modload ./lab3.kmod

В логи был выведен список всех процессов системы.

5 Результаты

Были реализованы драйверы, выводящие список системных процессов в лог при их запуске на операционных системах ReactOS и NetBSD.

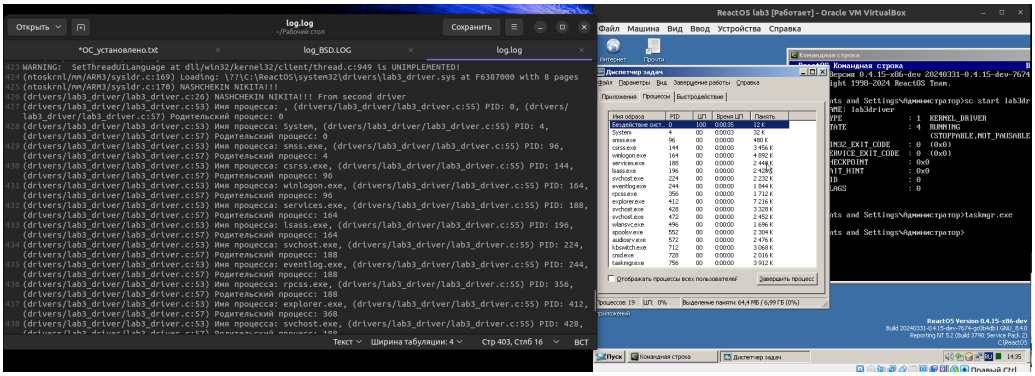

Рис. 1 — Скриншот 1

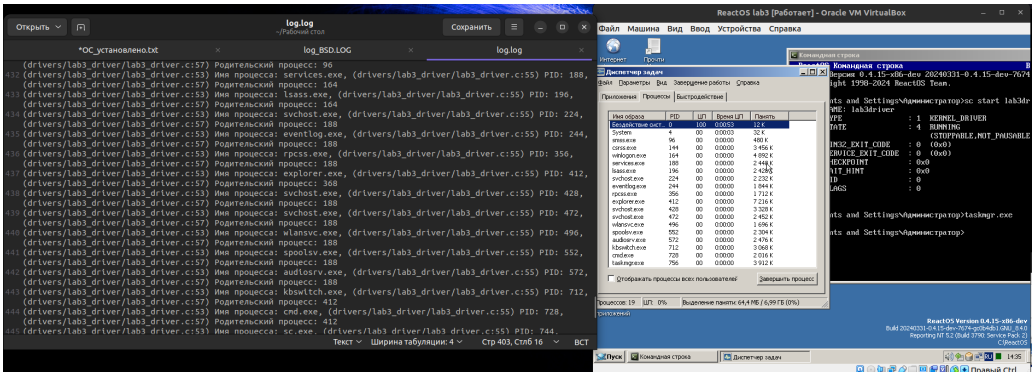

Рис. 2 — Скриншот 2

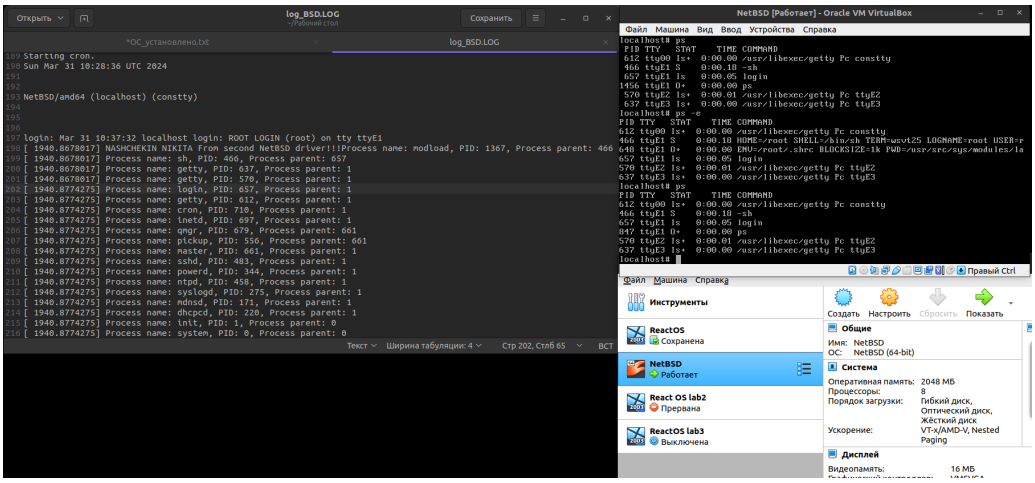

Рис. 3 — Скриншот 3

6 Выводы

Выполнив данную лабораторную работу, я разработал два драйвера для ReactOS и NetBSD. На ReactOS список, выводимый моим драйвером, аналогичен списку, который предоставляет утилита taskmgr.exe. На NetBSD мой драйвер выводит даже больше процессов, чем утилита ps. Видимо, ps, вызываемая без аргументов, не отображает некоторые системные процессы. Как и в предыдущей лабораторной, на NetBSD разработка драйвера оказалась проще благодаря готовому макросу PROCLIST_FOREACH, производящему удобный перебор всех процессов в передаваемом списке, и глобальной переменной allproc, которая сразу хранит в себе информацию обо всех системных процессах.

7 Список литературы

[1] - https://reactos.org/wiki/Building_ReactOS

[2] - https://cpp.hotexamples.com/examples/-/-/ZwQuerySystemInformation/cpp-zwquerysysteminformation-function-examples.html

[3] - https://reactos.org/forum/viewtopic.php?t=16944

[4] - https://learn.microsoft.com/en-us/windows/win32/sysinfo/zwquerysysteminformation

[5] - https://ftp.netbsd.org/pub/NetBSD/NetBSD-current/src/sys/sys/proc.h