|  |  |
| --- | --- |
|  | **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»***

|  |  |
| --- | --- |
| Студент *ИУ9-41Б* | *Ионов Т. Р.* |
| Преподаватель | *Брагин А.В.* |

Москва, 2021 г.

**Содержание**

1.Постановка задачи.

2.Практическая реализация.

2.1 Reactos.

2.2 NetBSD.

3.Результаты.

4.Вывод.

5.Используемая литература.

**Постановка задачи**

Разработать простой загружаемый модуль ядра (драйвер), который выводит в отладочный лог список процессов в системе

**Практическая реализация**

**2.1. ReactOS**

1. Изучение общей структуры загружаемого модуля ядра в системе ReactOS в официальной документации. Изучение библиотек, позволяющим получить доступ к списку процессов в системе
2. Модификация исходных файлов операционной системы, путём добавления в папку drivers файла с исходным кодом драйвера, а также служебных файлов, необходимых для компиляции
3. Пересборка iso-образа операционной системы и его установка на виртуальную машину Oracle VirtualBox.
4. Запуск полученного драйвера с помощью встроенной утилиты sc start.

**2.2. NetBSD**

1. Изучение общей структуры загружаемого модуля ядра в системе NetBSD в официальной документации. Изучение библиотек, позволяющим получить доступ к списку процессов в системе
2. Добавление исходного кода драйвера в папку /usr/src/sys/dev и makefile в папку /usr/src/sys/modules/lab3.
3. Сборка драйвера с помощью утилиты make и его установка с помощью встроенной утилиты modload.

**Результаты**

1.1) Исходный код драйвера:

/\*

\* PROJECT: ReactOS Kernel

\* LICENSE: GPL - See COPYING in the top level directory

\* FILE: drivers/base/mydriver/mydriver.c

\* PURPOSE: Mydriver Device Driver

\* PROGRAMMERS: Alex Ionescu (alex.ionescu@reactos.org)

\* Eric Kohl

\*/

/\* INCLUDES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <ntddk.h>

#ifndef NDEBUG

#define NDEBUG

#endif

#include <debug.h>

#include <ntifs.h>

#include <ndk/exfuncs.h>

/\* FUNCTIONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DRIVER\_UNLOAD DriverUnload;

VOID

NTAPI

DriverUnload(IN PDRIVER\_OBJECT DriverObject)

{

PDEVICE\_OBJECT DeviceObject;

/\* Get DO and DE \*/

DeviceObject = DriverObject->DeviceObject;

/\* Delete the object \*/

IoDeleteDevice(DeviceObject);

}

VOID

NTAPI

PrintProcesses(IN PSYSTEM\_PROCESS\_INFORMATION ProcInfo)

{

DPRINT1("------------------PROCESSES-LIST--------------------\n");

**while** (ProcInfo->NextEntryOffset) {

ProcInfo = (PSYSTEM\_PROCESS\_INFORMATION)((ULONG\_PTR)ProcInfo + (ULONG\_PTR)ProcInfo->NextEntryOffset);

DPRINT1("Image name: %wZ, Unique PID: %d, Inherited PID: %d, Base priority: %d\n", &ProcInfo->ImageName, ProcInfo->UniqueProcessId, ProcInfo->InheritedFromUniqueProcessId, ProcInfo->BasePriority);

}

}

NTSTATUS

NTAPI

Processes()

{

NTSTATUS Status;

PSYSTEM\_PROCESS\_INFORMATION ProcInfo;

/\* Creating buffer \*/

ULONG BufferSize = **sizeof**(SYSTEM\_PROCESS\_INFORMATION);

ULONG ReturnLength;

PVOID Buffer = ExAllocatePool(NonPagedPool, BufferSize);

**if** (Buffer == **NULL**)

{

DPRINT1("ERROR: NOT ENOUGH MEMORY\n");

**return** STATUS\_INSUFFICIENT\_RESOURCES;

}

/\* Getting processes information \*/

Status = ZwQuerySystemInformation(SystemProcessInformation, Buffer, BufferSize, &ReturnLength);

/\* Reallocating buffer if not it's enough and getting process information again \*/

**if** (Status == STATUS\_INFO\_LENGTH\_MISMATCH)

{

ExFreePool(Buffer);

BufferSize = ReturnLength;

Buffer = ExAllocatePool(NonPagedPool, BufferSize);

**if** (Buffer == **NULL**)

{

DPRINT1("ERROR: NOT ENOUGH MEMORY\n");

**return** STATUS\_INSUFFICIENT\_RESOURCES;

}

Status = ZwQuerySystemInformation(SystemProcessInformation, Buffer, BufferSize, &ReturnLength);

}

**if** (!NT\_SUCCESS(Status))

{

ExFreePool(Buffer);

DPRINT1("ERROR: Status = %Х\n", Status);

**return** Status;

}

/\* Printing processes information \*/

ProcInfo = (PSYSTEM\_PROCESS\_INFORMATION)Buffer;

PrintProcesses(ProcInfo);

/\* Freeing buffer \*/

ExFreePool(Buffer);

**return** Status;

}

NTSTATUS

NTAPI

DriverEntry(IN PDRIVER\_OBJECT DriverObject,

IN PUNICODE\_STRING RegistryPath)

{

PDEVICE\_OBJECT DeviceObject;

UNICODE\_STRING DeviceName = RTL\_CONSTANT\_STRING(L"\\Device\\Mydriver");

NTSTATUS Status;

UNREFERENCED\_PARAMETER(RegistryPath);

/\* Create the device \*/

Status = IoCreateDevice(DriverObject,

0,

&DeviceName,

FILE\_DEVICE\_UNKNOWN,

0,

**FALSE**,

&DeviceObject);

**if** (!NT\_SUCCESS(Status)) **return** Status;

/\* Make it use buffered I/O \*/

/\* Setup the Driver Object \*/

DriverObject->DriverUnload = DriverUnload;

/\* Page the entire driver \*/

MmPageEntireDriver(DriverEntry);

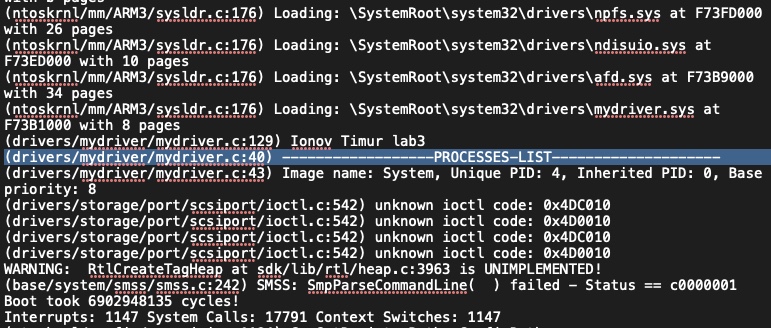
DPRINT1("Ionov Timur lab3\n");

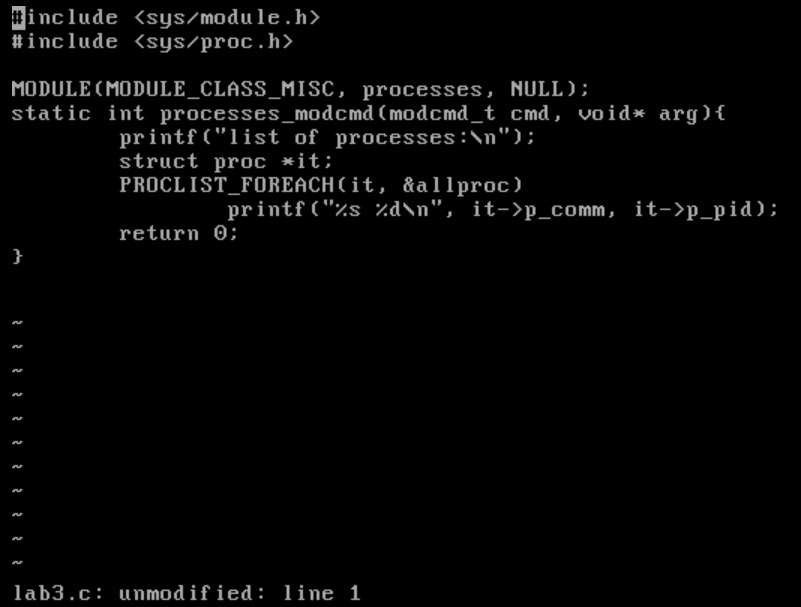
Processes();

**return** STATUS\_SUCCESS;

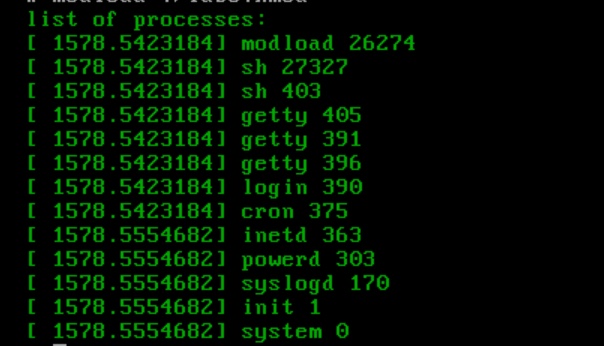
}

/\* EOF \*/

1.2) Вывод процессов системы с помощью драйвера ReactOS в файл out.txt в основной ОС:

2.1) Исходный код драйвера для NetBSD:

2.2) Вывод после запуска драйвера NetBSD процессов системы:



**Вывод**

В ходе выполнения были изучены библиотеки, позволяющие получить процессы системы. Написание драйвера для ReactOS показалось задачей сложнее за счет более сложной архитектуры. Драйвер для NetBSD оказалось писать намного легче и не вызвало каких-либо трудностей.

**Используемая литература**

1) Официальный вики операционной системы ReactOS - reactos.org/wiki.

2) Официальная документация к системе NetBSD - www.netbsd.org/docs.

3) Введение в сборку загружаемых модулей для NetBSD - http://www.home.unix-ag.org/bmeurer/NetBSD/howto-lkm.html.