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 № 1

по курсу «Операционные системы»

ReactOS и NetBSD. Среда сборки, установки и тестирование в виртуальной машине

Студент группы ИУ9-42Б Нащекин Н. Д.

Преподаватель: Брагин А. В.

Москва, 2024

1 Содержание

2 Цель

Приобретение навыков по самостоятельной сборке «с нуля» дистрибутивов операционных систем ReactOS и NetBSD с последующей их установкой в виртуальную машину.

3 Постановка задачи

ЧАСТЬ 1. СБОРКА ДИСТРИБУТИВА REACTOS

Настроить среду сборки и тестирования ReactOS на своём компьютере (или на компьютере в учебной аудитории университета), собрать установочный образ и произвести его установку в виртуальную машину с помощью программных средств виртуализации.

ЧАСТЬ 2. ОПЕРАЦИОННАЯ СИСТЕМА NETBSD: СРЕДА СБОРКИ, УСТАНОВКА И ТЕСТИРОВАНИЕ

Установить новый выпуск NetBSD из дистрибутива с официального сайта проекта NetBSD https://www.netbsd.org/ в виртуальную машину. Настроить среду сборки и тестирования NetBSD в виртуальной машине на своём компьютере (или на компьютере в учебной аудитории университета). Пересобрать ядро в этой виртуальной машине, добавив вывод фамилии обучающегося в debug log.

4 Практическая реализация

ЧАСТЬ 1

Сначала мной была установлена ReactOS Build Environment (RosBE)[1]. После установки среды сборки я получил исходный код операционной системы[2]. Используя среду сборки, я собрал ISO-образ ReactOS и установил его в вирутальной машине. В настройках виртуальной машины я включил последовательный порт COM-1 и перенаправил вывод в файл для удобного изучения логов.

Далее сразу в нескольких файлах ядра (ntoskrnl) я добавил вывод своих фамилии и имени в лог, например в файле ntoskrnl/ex/init.c:

```
/* Initialize keyed events */
    if (ExpInitializeKeyedEventImplementation() == FALSE)
    {
        DPRINT1("Executive: Keyed event initialization failed\n");
        return FALSE;
    }

    /* Initialize Win32K */
    if (ExpWin32kInit() == FALSE)
    {
        DPRINT1("Executive: Win32 initialization failed\n");
        return FALSE;
    }

    DPRINT1("NASHCHEKIN NIKITA\n");


    return TRUE;
```

В файле ntoskrnl/io/iomgr/driver.c:

```
IopDeleteDriver(IN PVOID ObjectBody)
{
    PDRIVER_OBJECT DriverObject = ObjectBody;
    PIO_CLIENT_EXTENSION DriverExtension, NextDriverExtension;
    PAGED_CODE();

    DPRINT1("Deleting driver object '%wZ'\n", &DriverObject->DriverName);
    DPRINT1("NASHCHEKIN NIKITA!!!\n");

    /* There must be no device objects remaining at this point */
```

```
11      ASSERT(!DriverObject->DeviceObject);
12
13      /* Get the extension and loop them */
14      DriverExtension = IoGetDrvObjExtension(DriverObject)->ClientDriverExtension;
15      while (DriverExtension)
16      {
17          /* Get the next one */
18          NextDriverExtension = DriverExtension->NextExtension;
19          ExFreePoolWithTag(DriverExtension, TAG_DRIVER_EXTENSION);
20
21          /* Move on */
22          DriverExtension = NextDriverExtension;
23      }
24
25      /* Check if the driver image is still loaded */
26      if (DriverObject->DriverSection)
27      {
28          /* Unload it */
29          MmUnloadSystemImage(DriverObject->DriverSection);
30      }
31
32      /* Check if it has a name */
33      if (DriverObject->DriverName.Buffer)
34      {
35          /* Free it */
36          ExFreePool(DriverObject->DriverName.Buffer);
37      }
38
39      /* Check if it has a service key name */
40      if (DriverObject->DriverExtension->ServiceKeyName.Buffer)
41      {
42          /* Free it */
43          ExFreePool(DriverObject->DriverExtension->ServiceKeyName.Buffer);
44      }
45  }
```

Аналогичные строки я добавил в нескольких других файлах (для надежности!). Далее я пересобрал ядро и переустановил систему.

ЧАСТЬ 2

Сначала я скачал образ дистрибутива ОС NetBSD[3] и установил его в виртуальную машину. При установке системы я не настроил использование com-порта, поэтому пришлось добавить в файл /boot.cfg строку

consdev=com0,115200

Аналогичным образом настроил вывод отладки в файл. Затем по ftp (из

NetBSD) скачал архив с ядром системы[4], распаковал его, изменил несколько файлов, добавив вывод своей фамилии, и собрал ядро[5]. Заменив ядро, убедился в наличии моих фамилии и имени в логах. Фрагмент кода из файла usr/src/sys/kern/init_main.c:

```
/* Create the aiodone daemon kernel thread. */
        if (workqueue_create(&uvm.aiodone_queue, "aiodoned",
            uvm_aiodone_worker, NULL, PRI_VM, IPL_NONE, WQ_MPSAFE))
                panic("fork aiodoned");

        printf("NASHCHEKIN NIKITA!!!");
        /* Mount the root file system. */
        do {
                domountroothook(root_device);
                if ((error = vfs_mountroot())) {
                        printf("cannot mount root, error = %d\n", error);
                        boothowto |= RB_ASKNAME;
                        setroot(root_device,
                            (rootdev != NODEV) ? DISKPART(rootdev) : 0);
                }
        } while (error != 0);
        mountroothook_destroy();
```

5 Результаты

Был реализован вывод моих фамилии и имени в лог при запуске операционных систем ReactOS и NetBSD.

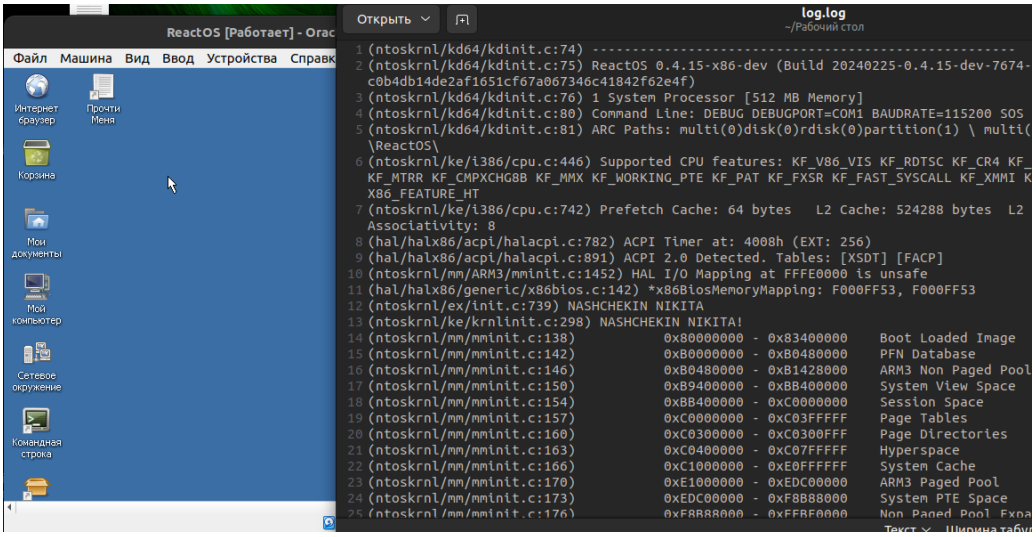

Рис. 1 — Скриншот 1

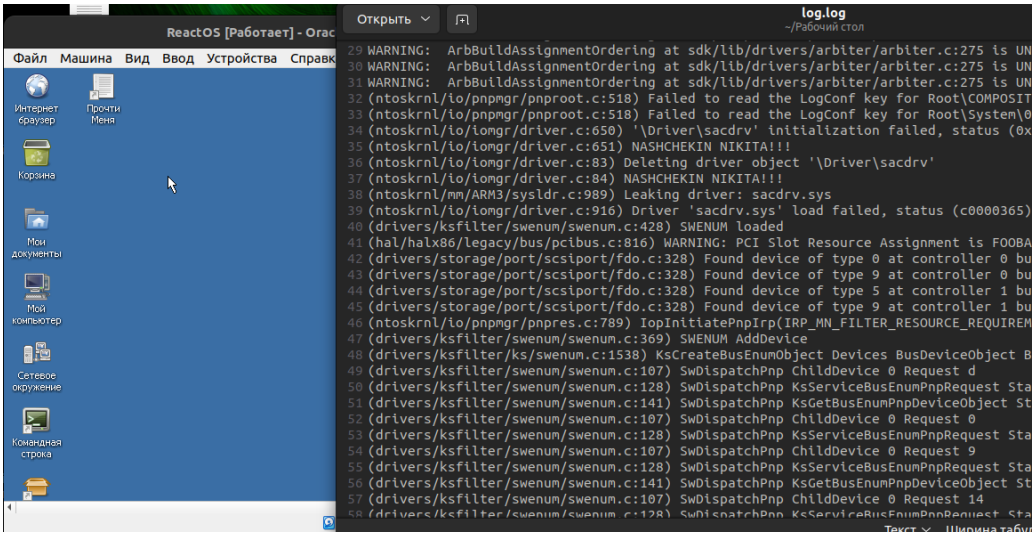

Рис. 2 — Скриншот 2

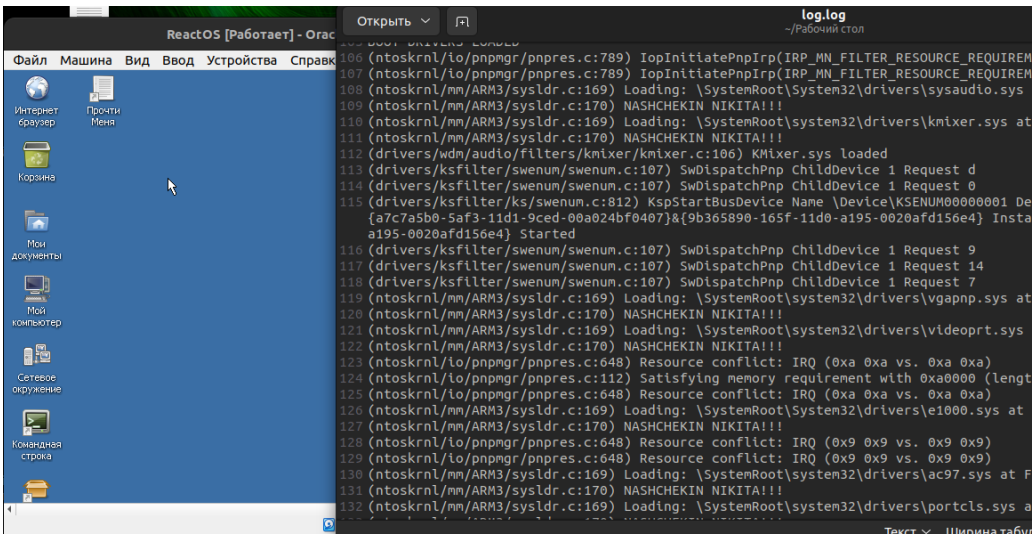

Рис. 3 — Скриншот 3

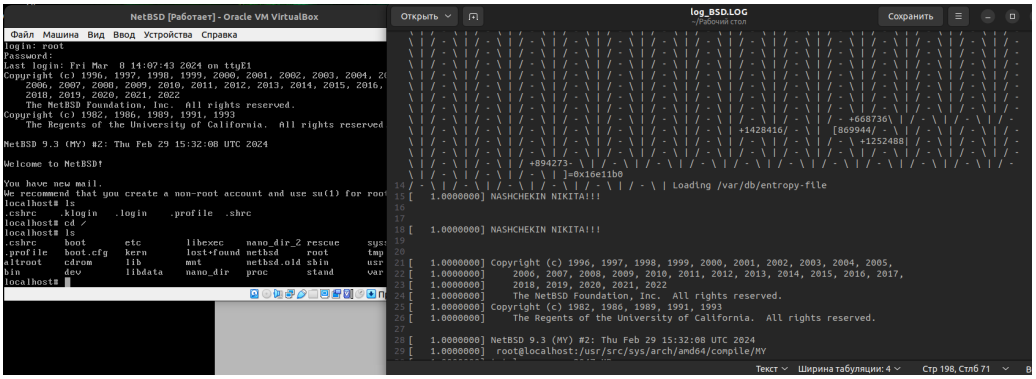

Рис. 4 — Скриншот 4

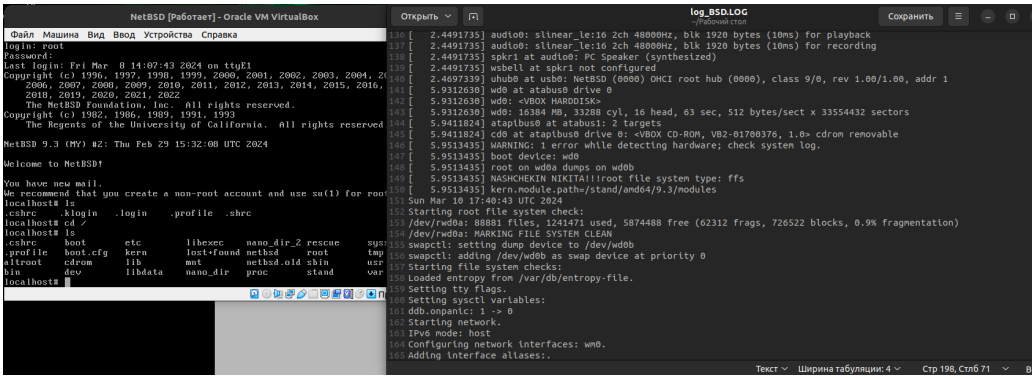

Рис. 5 — Скриншот 5

6 Выводы

Выполнив данную лабораторную работу, я приобрёл навыки сборки с нуля дистрибутивов операционных систем и проведения их установки. Я смог найти файлы, отвечающие за инициализацию операционных систем, и добавил туда свои строки. Выполнять первую часть работы на ReactOS было проще: я мог изменить файлы из старшей системы, провести компиляцию ядра удобной средой сборки и просто переустановить систему. Во второй части мне пришлось вручную компилировать ядро внутри NetBSD, что немного сложнее.

7 Список литературы

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

[2] - https://github.com/reactos/reactos.git

[3] - https://www.netbsd.org/

[4] - http://ftp.NetBSD.org/pub/NetBSD/NetBSD-9.3/source/sets/syssrc.tgz

[5] -
https://netbsd.org/docs/guide/en/chap-kernel.html#chap-kernel-building-manually