Операционная система FreeBSD на ноутбуке Lenovo

В этой статье я хочу поделиться с общественностью некоторыми аспектами настройки и эксплуатации операционной системы FreeBSD при установке на современный ноутбук с целью использования его как основного рабочего места инженера (программиста, электронщика или конструктора). В своих предыдущих статьях я упоминал, что являюсь тонким ценителем этой ОС и с некоторых пор организовал своё рабочее место под управлением FreeBSD, о чем ни сколько не пожалел, и даже наоборот — мои волосы теперь по-настоящему мягкие и шелковистые.

Моя статья посвященная настройке САПР КОМПАС-3D под FreeBSD получила ряд одобрительных комментариев, поэтому мне захотелось продолжить тему «FreeBSD на десктопе». К тому же, есть добрые предпосылки — недавно я приобрел новый современный ноутбук **Lenovo Ideapad 3 Gaming** взамен окончательно рассыпавшегося на несколько частей Asus VX7, а с ним и массу приятного и затейливого опыта установки и настройки ОС FreeBSD для работы на новом «железе». В этой статье я не будут касаться установки и настройки специализированного ПО и прочих САПР, будет рассмотрен только системный вопрос: установка операционной системы, драйверов, патчей, библиотек, настройка и борьба с железом. Будет много выдержек из системного руководства (мануала - man) — уж сильно я к нему пристрастился за последние 130 лет.

Продолжить читать журнал

В этом выпуске:

1. Аппаратный вопрос

Для начала немного слов про выбор ноутбука. Опыт эксплуатации FreeBSD на моём старом ноутбуке Asus VX7 показал, что при приобретении нового ПК нужно тщательнейшим образом подходить к выбору аппаратуры на которой этой ОС предстоит работать. Ни для кого не является секретом то, что во FreeBSD с драйверами дела обстоят несколько кхм... похуже чем в Linux и на то есть свои причины — команда разработчиков этой ОС достаточно небольшая, вокруг системы нет такого количество сторонних комиттеров как в Linux, а система проходит почти незамеченной крупными игроками в IT индустрии и у производителей аппаратуры. В общем, драйвера к ней создаются либо отъявленными энтузиастами, либо по целевому финансированию от компаний бизнес которых выстроен вокруг FreeBSD. По большей части финансируются разработки драйверов сетевых и дисковых адаптеров, так как FreeBSD, в подавляющем большинстве случаев, используется на тяжело нагруженных серверах, а о поддержке «десктопа» мало кто думает.

Справедливости ради стоит заметить, что такой известный производитель видео карт с GPU как NVIDIA регулярно выпускает билды драйверов к своим видюхам в том числе и для FreeBSD amd64. Вот только вокруг изделий этой фирмы сложилась странная аура неприязни к качеству их продукции и многие фрюховоды и линуховоды настоятельно рекомендуют GPU от AMD в качестве альтернативы.

Так как новые ПК я приобретаю не часто, то хотелось взять сразу такой, чтоб навсегда. Ну или хотя бы лет на пять. И чтоб был очень очень производительный и при этом не грелся как чайник. И, разумеется, чтобы не дорогой. И удобный. Полистав глянцевые страницы интернет-журналов с обозреванием современных ноутбуков я пришел к выводу, что мне нужен ноутбук на процессоре AMD Ryzen 5, 7 или даже 9. Так получилось, что ноутбук я приобретал в марте 2022г, когда ценник на всю вычислительную технику резко взлетел вместе с курсом доллара. Прикинув остатки барышей я понял, что Ryzen 5 — мой выбор. В магазинах имелся на тот момент достаточно богатый выбор ноутбуков на этом процессоре, в том числе от известных брендов (Asus, Dell, MCI), но моё внимание привлек незатейливый ноут от Lenovo под названием Ideapad 3 Gaming. Привлёк он меня вот чем:

1. У Lenovo Ideapad 3 Gaming очень приятная на ощупь клавиатура (приятный клик) и с правильной раскладкой. Нет, кнопку «windows» на нём не ликвидировали, но в остальном расположение клавиш очень годное и мне привычное. В BIOS есть фича которая из бесполезной кнопки **Fn** делает еще один **LeftCtrl** — при работе в командной строке это благодать.

2. Процессор AMD Ryzen 5 5600H работающий на частоте 3293.96 MHz со встроенным видео ускорителем Radeon Graphics — всё как рекомендовано лучшими фрюховодами.

3. Второй (дискретный) GPU NVIDIA GeForce RTX 3050. Таким образом получается два GPU и вроде как оба поддерживаются во FreeBSD.

4. SSD диск объемом 238 GiB производства Samsung с интерфейсом NVMe — хоть и не большой по объему, но должен быть очень быстрый.

В остальном, достаточно стандартная начинка.

Не сильно долго размышляя, но и не торопясь я поехал в ближайший магазин и приобрел этот ноут, а к нему дополнительно оперативной памяти - суммарным объемом получилось 24 GiB. Так же приобрел запасной блок питания на 170 W. Во-первых, у ноута нестандартный разъем питания (прямоугольный, чем-то похож на USB type A), а блоки питания имеют свойство быстро дохнуть. Во-вторых, приятно иметь по отдельному БП дома и на работе, чтобы не носить с собой лишний груз.

Привез я это все домой, плотно поужинав и уложив детей спать, принялся в спокойной ночной обстановке под музыку в стиле Sovietwave устанавливать операционную систему. Повторюсь, моя цель — сделать на ноутбуке полноценное рабочее место с преферансом и куртизанками на базе ОС FreeBSD.

2. О версиях ОС FreeBSD

Первая версия ОС FreeBSD увидела свет в ноябре 1993 года и имела номер 1.0R. Это был «fork» версии ОС 386BSD разрабатываемой в калифорнийском университете в Berkley группой программистов Berkley Software Distribution (BSD), которая в свою очередь являлась переносом ОС 4.4BSD на платформу i386. Номер версии операционной системы FreeBSD некоторое время был многозначным (например 1.1.5.1 или 2.0.5), но с версии 6.0 разработчики перешли на двухзначное обозначение номера версии. Буква «R» в конце означает «release» («релиз» - выпуск) и чуть позже её заменили на суффикс обозначающий готовность системы к изнурительному труду. На данный момент имеется четыре суффикса, а значит четыре состояния готовности:

- Суффикс **-RCx** указывает на версию системы подготовленную к релизу (т. н. «release candidate X» - «релиз кандидат»), но еще не совсем дозревшую. Обычно, перед релизом может быть выпущено несколько кандидатов, которые тестируются широким кругом пользователей и вносятся незначительные корректировки. Например: **13.1-RC5** означает, пятый по счету кандидат в релизы перед выпуском версии **13.1**.
- Суффикс **-RELEASE** означает, что система стабильная и надежная как гранит и является очередным выпуском (релизом). Например, ОС FreeBSD 13.1-RELEASE рекомендована к установке на высоко нагруженные вычислительные системы.
- Суффикс **-STABLE** означает, что в системе имеются кое-какие изменения относительно релиза, не всегда сказывающиеся в лучшую сторону на её стабильности. Например, могут присутствовать новые, не до конца отлаженные драйверы или патчи к проблемам безопасности «на скорую руку». К версии -STABLE нужно относится с некоторым вниманием, потому как не все патчи одинаково полезны.
- И на конец суффикс **-CURRENT** указывает на то, что данная версия - это самое последнее достижение в разработке системы, т. е. экспериментальная версия системы со всеми нововведениями. Данная версия рекомендуется только для разработчиков системы, для экспериментаторов или бета-тестеров. На момент написания данной статьи, таковой являлась версия FreeBSD 14-CURRENT.

3. Установка ОС FreeBSD и первые звоночки

Инсталляционный образ FreeBSD (на тот момент 13.0-RELEASE) был мной уже ранее выкачан и записан на USB носитель, загрузка с которого прошла без особых проблем. Для желающих попробовать FreeBSD привожу ссылку на официальную инструкцию для версии 13.1-RELEASE: https://www.freebsd.org/releases/13.1R/announce/

Сам процесс инсталляции тривиальный. Инсталлятор задаст несколько вопросов: как распределить дисковое пространство, какие системные пакеты установить, предложит

настроить сеть и завести первого пользователя, а так же предложит установить пароль для системного пользователя **root**. Обычно от настройки сети на этом этапе я отказываюсь, так как могут быть нюансы.

Доверившись инсталлятору я разбил диск по-умолчанию, т. е. все файловые системы в одном разделе и запустил установку. Инсталлировал полный комплект вместе с портами, исходными текстами ядра и системных утилит - опыт подсказывал, что исходники обязательно понадобятся. Новой машине дал название **butterfly**, завел себе пользователя **rz**, добавил его в группу **wheel** и на этом - всё.

Тут следует заметить, что установка FreeBSD происходит целиком и полностью в консольном (текстовом) режиме. Более того, базовая установка не предполагает установки каких либо графических подсистем, т. е. после загрузки ОС по окончанию инсталляции Вы получаете сообщение «Добро пожаловать» и промпт «login:» от одноименной утилиты. В общем, эталон минимализма.

После перезагрузки и входа в систему сразу выяснилось, что система не обнаружила Wi-Fi карту MediaTek MT7961, но обнаружила Ethernet карту RealTek RTL8168 и это уже порадовало — значит я смогу продолжить доустановку системы. Вот, что об имеющихся в ноутбуке сетевых адаптерах говорит утилита **pciconf**:

```
rz@butterfly:~ % su
Password: <пароль root-а>

root@butterfly:/home/rz # pciconf -levc
...
re0@pci0:2:0:0:       class=0x020000 rev=0x15 hdr=0x00 vendor=0x10ec device=0x8168
subvendor=0x17aa subdevice=0x3909
    vendor     = 'Realtek Semiconductor Co., Ltd.'
    device     = 'RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller'
    class      = network
    subclass   = ethernet
    bar   [10] = type I/O Port, range 32, base 0x2000, size 256, enabled
    bar   [18] = type Memory, range 64, base 0xd1704000, size 4096, enabled
    bar   [20] = type Memory, range 64, base 0xd1700000, size 16384, enabled
...
none1@pci0:3:0:0:       class=0x028000 rev=0x00 hdr=0x00 vendor=0x14c3 device=0x7961
subvendor=0x17aa subdevice=0xe0bc
    vendor     = 'MEDIATEK Corp.'
    class      = network
    bar   [10] = type Prefetchable Memory, range 64, base 0xfc02000000, size 1048576, enabled
    bar   [18] = type Prefetchable Memory, range 64, base 0xfc02100000, size 16384, enabled
    bar   [20] = type Prefetchable Memory, range 64, base 0xfc02104000, size 4096, enabled
    cap 10[80] = PCI-Express 2 endpoint max data 128(128) FLR RO NS
                 max read 512
                 link x1(x1) speed 5.0(5.0) ASPM L1(L0s/L1) ClockPM enabled
    cap 05[e0] = MSI supports 32 messages, 64 bit, vector masks
    cap 01[f8] = powerspec 3  supports D0 D1 D2 D3  current D0
    ecap 000b[100] = Vendor [1] ID 1556 Rev 1 Length 8
    ecap 0018[108] = LTR 1
    ecap 001e[110] = L1 PM Substates 1
    ecap 0001[200] = AER 2 0 fatal 0 non-fatal 0 corrected
```

Немного погуглив, я выяснил, что поддержка адаптера MT7961 появилась в ядре Linux совсем недавно, с версии 5.13, а это давало понять, что во FreeBSD её можно не ждать в ближайшие годы. Очевидно, что в таком случае самый простой способ сделать поддержку Wi-Fi на этом ноутбуке это купить новый Wi-Fi адаптер (из списка подерживаемых) и попросту заменить, благо ноутбук легко разбирается и карта адаптера доступна для замены.

Тем временем решил двигаться далее. Легким движением руки настроил сеть путем добавления в системный конфигурационный файл **/etc/rc.conf** одной строки **ifconfig_re0="DHCP"**, для этого воспользовался следующими командами:

```
root@butterfly:/home/rz # echo 'ifconfig_re0="DHCP"' >> /etc/rc.conf
```

А чтобы система приняла мои изменения, я выполнил команду:

```
root@butterfly:/home/rz # /etc/netstart
```

по завершению которой убедился в готовности сетевого интерфейса командой **ifconfig re0:**

```
root@butterfly:/home/rz # ifconfig re0
re0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8209b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,WOL_MAGIC,LINKSTATE>
        ether 7c:8a:e1:a5:9e:03
        inet 192.168.172.110 netmask 0xffffff00 broadcast 192.168.172.255
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

и принялся доустанавливать систему и инсталлировать требуемые мне пакеты.

О системе пакетов я расскажу более подробно чуть позже, а сейчас установлю первый пакет - это будет утилита **sudo**, она позволяет избавиться от необходимости постоянно висеть в оболочке (шелле) от имени суперпользователя. Для этого используем следующую команду:

```
root@butterfly:/home/rz # pkg inst --yes sudo && rehash
```

После установки **sudo** необходимо подредактировать конфигурационный файл с помощью специальной утилиты **visudo** (устанавливается вместе с пакетом **sudo**), в файле требуется раскомментировать или добавить строку вида:

```
%wheel ALL=(ALL:ALL) ALL
```

После чего сохранить изменения и выйти из **visudo**. Если Вы используете редактор **vi**, что наиболее вероятно, то требуется нажать следующую последовательность клавиш: **<ESC>:wq<ENTER>**

Наличие данной строки в конфигурационном файле утилиты **sudo** позволит получать привилегии суперпользователя всем пользователям которые входят в группу **wheel**. Напомню, что я внес себя в группу **wheel** при инсталляции.

4. Апдейт и апгрейд ОС FreeBSD

Сразу после установки базового комплекта ОС FreeBSD, не плохо бы проапгрейдиться до последней версии, так как развернутый образ системы на моей USB флешки уже слегка залежавшийся. Для этих целей во FreeBSD имеется специальной средство — утилита **freebsd-update**, которая является универсальным средством для обновления системы. Данная утилита позволяет выполнять следующие задачи:

- Обновить текущую систему и установить критические обновления (патчи) без смены номера версии ОС.
- Обновить систему на более новую версию (minor version update).

- Обновить систему на более новую ветку (major version update).

Во все случаях утилита **freebsd-update** выкачивает и устанавливает набор обновлений к ядру ОС, к системным утилитам и системным библиотекам в *бинарном* виде. Происходит это в два этапа: 1) выкачивание патчей и подготовка новых версий ядра, утилит и библиотек к установке, и 2) собственно установка, и, по необходимости, перезагрузка системы и дальнейшая доустановка.

Небольшое замечание: устанавливать обновление к ядру ОС утилита **freebsd-update** будет только в том случае, если Вы используете конфигурацию ядра по-умолчанию (ядро **GENERIC**). Если же на Вашей машине используется своё, кастомное ядро, то сборку ядра и его установку после обновления придется выполнить вручную, но утилита выкачает и применит все необходимые патчи к исходным кодам ядра.

Выкачивание критических обновлений осуществляется по команде:

```
rz@butterfly:~ % sudo freebsd-update fetch
Password: <пароль пользователя rz>
Looking up update.FreeBSD.org mirrors... 2 mirrors found.
Fetching metadata signature for 13.0-RELEASE from update1.freebsd.org... done.
Fetching metadata index... done.
Inspecting system... done.
Preparing to download files... done.
The following files will be updated as part of updating to
13.1-RELEASE:
/boot/kernel/kernel
/usr/lib/debug/boot/kernel/kernel.debug
(END) q
```

Прежде всего утилита **sudo** попросит Вас авторизоваться - ввести пароль пользователя (**rz**), после чего она запустит утилиту **freebsd-update**. Далее утилита **freebsd-update** сообщит о том, какие патчи она выкачала и предложит просмотреть список. Меня в этом списке всё устраивает, поэтому я нажимаю **q** для выхода.

Установить выкачанные обновления можно командой:

```
rz@butterfly:~ % sudo freebsd-update install
```

Повторное использование утилиты **sudo** больше не просит ввода пароля — она запомнила его на некоторое время, таким образом избавив пользователя (меня) от необходимости постоянно вводить его при активной работе и настройке системы.

Если после установки обновлений что-то пошло не так, то по команде **freebsd-update rollback** можно откатиться к предыдущему состоянию системы, в том числе вернуть старую версию ядра ОС и выполнить перезагрузку (если требуется).

Так как утилита **freebsd-update** работает накопительным образом, т. е. постепенно выкачивает и складывает накопленные патчи, то рекомендуется добавить ежедневный вызов **freebsd-update** из **cron**-а. **Cron** это фоновый процесс который запускает другие процессы по расписанию. Запустить редактор таблицы **cron** можно командой:

```
rz@butterfly:~ % sudo crontab -e
```

Утилита **crontab** запустит редактор **vi** и предложит Вам отредактировать пустой файл, в него нужно добавить строку вида:

```
@daily<TAB>root<TAB>freebsd-update cron
```

Здесь <TAB> это символ табуляции. Сохранить изменения и выйти из редактора **vi**.

Это позволит ежедневно выкачивать все критические обновления и накапливать их в локальной системе. Установить накопленные обновления можно в любой удобный момент по команде **sudo freebsd-update install**.

Апгрейд операционной системы со сменой номера версии немного более сложен и происходит в три фазы. Для выполнения апгрейда нужно знать номер версии системы (с суффиксом) до которой предполагается выполнить этот апгрейд, при этом возможно делать не только повышение номера версии, но и понижение. Список поддерживаемых версий ОС FreeBSD можно подсмотреть на сайте проекта: https://www.freebsd.org, другой метод мне не известен.

В обоих случаях операция смены версии ОС достаточно рисковая и не плохо бы позаботиться о резервной копии важных данных (как минимум файла с паролями). Так как ОС FreeBSD только что установленная на мой ноутбук еще не представляет никакой ценности, то смело выполним апгрейд системы до версии 13.1-RELEASE следующей командой:

```
rz@butterfly:~ % sudo freebsd-update upgrade -r 13.1-RELEASE
```

По этой команде утилита **freebsd-update**, во-первых проверит наличие в репозитории указанной версии ОС, во-вторых проверит целостность установленной системы и подготовит список патчей, а так же сообщит какие компоненты отсутствуют в системы, т. е. не будут изменяться. После подтверждения пользователя, утилита выкачает все требуемые патчи и подготовит их к установке. На этом заканчивается первая фаза. Дальнейшая установка производится, как обычно, по команде:

```
rz@butterfly:~ % sudo freebsd-update install
```

По команде **install** утилита **freebsd-update** выполнит установку нового ядра и системных библиотек и потребуется перезагрузки операционной системы. Это конец второй фазы.

Выполним перезагрузку командой **reboot**, дождемся окончания загрузки ОС, войдем в систему и перейдем к третьей фазе - продолжим доустановку повторным вводом команды:

```
rz@butterfly:~ % sudo freebsd-update install
```

по завершению которой не плохо бы еще раз перезагрузить систему с цель проверки её работоспособности и отсутствия проблем с загрузкой модулей. Всякое бывает!

Проверить текущую версию установленной ОС FreeBSD можно командой:

```
rz@butterfly:~ % freebsd-version
13.1-RELEASE
```

либо традиционным способом:

```
rz@butterfly:~ % uname -a

FreeBSD butterfly 13.1-RELEASE FreeBSD 13.1-RELEASE releng/13.1-n250148-fc952ac2212 GENERIC
amd64
```

5. «Фатальная» особенность оболочки tcsh

Кто-то уже наверное обратил внимание, что в приводимых мной командах в качестве приглашения (промпта) командной строки используется символ **%** (процент) в вместо привычного многим символа **$** (денежной единицы) - это признак того, что мы имеем дело с оболочкой **tcsh**.

В ОС FreeBSD, как и у остальных операционных систем группы BSD, традиционно используется командная оболочка типа C-Shell (**csh**, **tcsh**), вместо Bourne и Bourne Again Shell (**bash**) распространенных в среде ОС Linux. Отличия этих оболочек весьма существенные и я почти на 100% уверен, что пользователи привыкшие к **bash** будут долго плеваться от **tcsh** и, в конечном счете, просто установят **bash** из пакетов. Так, как я вырос на BSD, то мне видится всё наоборот — **tcsh** гораздо удобнее и приятнее в работе. В частности, в **tcsh** «из коробки» работает авторасширение и поиск по истории команд с помощью клавиш перемещения курсора «вверх» и «вниз». При этом, в отличии от **bash**, где тоже есть такая функция, поиск идет по части уже введенной команды. В **bash** также присутствует данная фича, но реализована она крайне убого. Хотя, уверен, я просто плохо знаю **bash**.

Но в этой главе я хотел бы рассказать не о достоинствах и недостатках двух разных оболочек, в конце концов это дело привычки, а об одной особенности оболочки **tcsh** о которую буквально спотыкаются все новые пользователи FreeBSD. Дело в том, что **tcsh** по-умолчанию, в отличии от **bash**, ***не*** переиндексирует список исполняемых файлов в доступных путях описанных переменной окружения **$PATH** каждый раз при вызове внешней команды. Это приводит тому, что после окончания установки какого либо нового пакета, допустим компилятора C/C++, введя команду **cc** мы рискуем получить сообщение об ошибке типа:

```
cc: Command not found.
```

Для переиндексации внутренней hash-таблицы у оболочки **tcsh** есть встроенная команда **rehash**, которую следует вводить каждый раз при появлении в путях нового исполняемого файла, вот так:

```
rz@butterfly:~ % rehash

rz@butterfly:~ % cc
cc: error: no input files
```

Для пользователей Linux это может показаться странным и неудобным, но на самом деле это не так и я объясню почему. Представьте, что Вы работаете на относительно медленной аппаратуре и у Вас в файловой системе располагаются сотни тысяч файлов (привет embedded разработчикам). Каждый раз когда Вы вводите команду, или нажимаете клавишу TAB, оболочка **bash** проводит поиск исполняемого файла по всем подкаталогам указанным в переменной окружения $PATH. Если файлов много, то это выражается в явной задержке, иногда очень даже надоедливой. В оболочке **tsch** по-умолчанию включен поиск по hash-таблице и поэтому **tcsh** выглядит более динамично (особенно при активном использовании клавиши TAB). Поиск по таблице можно выключить командой **unhash** или

снова включить командой **hash**, а обновить таблицу командой **rehash**. О оболочке **bash** все строго наоборот.

6. Несколько слов о системной документации

Если верить словам Брайна Кернига из его книги «Время UNIX: A History and Memoir», то разработанная узким кругом программистов в Bell Labs ОС UNIX была преподнесена руководству компании как инструмент для обработки документации. Да, это был такой хитрый план Томпсона, Кернигана, Ричи и других обитателей «комнаты UNIX» для того, чтобы выбить финансирование на новую машину PDP-11 для их инициативной разработки, ибо старенькая PDP-7 уже порядком разваливалась и не удовлетворяла потребностям коллектива по развитию ОС UNIX. План сработал, а ОС UNIX дала миру такие средства как **roff/ntoff**, **troff** и **ditroff** для обращения с документацией как со структурированными взаимосвязанными данными. На базе языка **troff** была организована и внутрисистемная документация по командам и утилитам в самой ОС UNIX, для обращения к которой была введена команда **man** (от «manual») и расширенный язык для описания страниц руководства - **mdoc**.

Следуя лучшим традициям ОС UNIX в ОС FreeBSD на всё есть **man-страницы**, в том числе на системные и пользовательские утилиты, на приложения, на конфигурационные файлы, на API и ABI, на библиотеки и даже на драйверы устройств. Вообще, следует заметить, что по части системной документации и различных руководств для пользователя ОС FreeBSD нет равных. Важно и то, что в ОС FreeBSD непрерывно поддерживается целостность этой документации! Во многих ОС на базе Linux тоже есть команда **man**, но по большей части присутствует она там номинально и в некоторых случаях содержимое её страниц позаимствовано из FreeBSD. Среди пользователей группы операционных систем BSD есть негласное правило: если не знаешь с какой стороны подступиться к проблеме — начни с чтения различных man-ов и узкая тропинка будет становиться все шире и приведет тебя к решению проблемы. А легенда гласит, что известный термин RTFM возник именно в отношении к man-ам. ;)

Давайте выясним, какую информацию может предоставить нам **man** в ОС FreeBSD, для этого введем следующую команду:

```
rz@butterfly:~ % man man
MAN(1)                  FreeBSD General Commands Manual                  MAN(1)

NAME
     man – display online manual documentation pages

SYNOPSIS
     man [-adho] [-t | -w] [-M manpath] [-P pager] [-S mansect]
         [-m arch[:machine]] [-p [eprtv]] [mansect] page ...
     man -f keyword ...
     man -k keyword ...

DESCRIPTION
     The man utility finds and displays online manual documentation pages.  If
     mansect is provided, man restricts the search to the specific section of
     the manual.

     The sections of the manual are:
           1.   FreeBSD General Commands Manual
           2.   FreeBSD System Calls Manual
           3.   FreeBSD Library Functions Manual
           4.   FreeBSD Kernel Interfaces Manual
           5.   FreeBSD File Formats Manual
           6.   FreeBSD Games Manual
```

```
7.   FreeBSD Miscellaneous Information Manual
8.   FreeBSD System Manager's Manual
9.   FreeBSD Kernel Developer's Manual
```

Страница по **man** сообщает нам формат команды **man** и её параметров, а так же выдает нам ряд интересной справочной информации о номерах секций системной документации и их назначении. Видно, что в системе присутствует девять секций (категорий) с различными руководствами, в том числе по разработке ядра ОС (9) и по играм (6) - как же без них! :)

Давайте выясним, что системе известно про сетевой адаптер RealTek 8168 установленный в моём новом ноутбуке и как вообще этот драйвер называется. Для этого введем команду:

```
rz@butterfly:~ % apropos realtek
re, if_re(4) - RealTek 8139C+/8169/816xS/811xS/8168/810xE/8111 PCI/PCIe Ethernet adapter
driver
rgephy(4) - RealTek RTL8168/8169/8110/8211 series 10/100/1000 Gigabit Ethernet PHY driver
rl, if_rl(4) - RealTek 8129/8139 Fast Ethernet device driver
rsu, if_rsu(4) - Realtek RTL8188SU/RTL8192SU USB IEEE 802.11b/g/n wireless network device
rsufw(4) - Firmware Module for Realtek driver
rtsx(4) - Realtek SD card reader
rtwn(4) - Realtek IEEE 802.11 wireless network driver
rtwn_pci, if_rtwn_pci(4) - Realtek PCI device glue
rtwn_usb, if_rtwn_usb(4) - Realtek USB device glue
rtwnfw(4) - Firmware Module for Realtek Wireless driver
rue, if_rue(4) - RealTek RTL8150 USB to Fast Ethernet controller driver
ure, if_ure(4) - RealTek RTL8152/RTL8153 USB to Ethernet controller driver
urtw, if_urtw(4) - Realtek RTL8187B/L USB IEEE 802.11b/g wireless network device
```

Команда **apropos** производит поиск строки символов (без учета регистра) по всем man-страницам зарегистрированным в системе и выдает сокращенный список страниц на которых есть упоминание заданной строки текста. В данном случае мы получили список всех man-страниц с упоминанием про RealTek.

Из вывода команды **apropos** видно, что для имеющегося у меня сетевого адаптера используется два драйвера: драйвер **re** для самого Ethernet адаптера подключенного к шине PCIe и драйвер **rgephy** для управления трансивером (Ethernet PHY). В выводе команды **apropos** в скобках указан номер секции (категории) руководства.

Давайте посмотрим, что содержится в 4-й секции руководства по драйверу **re**:

```
rz@butterfly:~ % man 4 re
RE(4)                  FreeBSD Kernel Interfaces Manual                  RE(4)

NAME
     re - RealTek 8139C+/8169/816xS/811xS/8168/810xE/8111 PCI/PCIe Ethernet
     adapter driver

SYNOPSIS
     To compile this driver into the kernel, place the following lines in your
     kernel configuration file:

           device miibus
           device re

     Alternatively, to load the driver as a module at boot time, place the
     following line in loader.conf(5):

           if_re_load="YES"
```

```
DESCRIPTION
     The re driver provides support for various NICs based on the RealTek
     RTL8139C+, RTL8169, RTL816xS, RTL811xS, RTL8168, RTL810xE and RTL8111 PCI
     and PCIe Ethernet controllers.
...
```

Как видно, в руководстве приведена исчерпывающая информация о поддерживаемых моделях сетевых адаптеров и то, как этот драйвер установить. В данном случае предлагается два способа: 1) статический - вкомпилить драйвер в ядро операционной системы подправив конфигурационный файл ядра, и 2) динамический — дать указание загрузчику (bootloader-у) загружать соответствующий модуль. Второй вариант считается более предпочтительным, но так как в инсталлированном по-умолчанию ядре уже содержится этот драйвер, то никаких действий для его подключения от меня не потребовалось.

Давайте теперь спросим систему, что ей известно о сетевом адаптере MediaTek который не был автоматически обнаружен системой не смотря на то, что он присутствует в моём ноутбуке. Для этого введем команду:

```
rz@butterfly:~ % apropos mediatek
apropos: nothing appropriate
```

Система нам сообщает, что ничего конкретного про MediaTek она не знает. Печально. Что-ж, придется отказаться от MediaTek в пользу другого производителя.

7. Про FreeBSD Handbook

Проектом **FreeBSD Documentation Project** написана и постоянно актуализируется большая книга по ОС FreeBSD, которая называется «FreeBSD Handbook» (или просто «хэндбук»). Хендбук бесплатно распространяется с ОС FreeBSD в электронном виде в формате HTML, а так же доступен на сайте проекта, в том числе имеется версия на русском языке: https://docs.freebsd.org/ru/books/handbook/ которая немного отстает от англоязычной версии: https://docs.freebsd.org/en/books/handbook/

«FreeBSD Handbook» содержит огромный объём информации по установке, настройке и администрированию ОС FreeBSD. В отличии от внутрисистемной документации man-страниц, хэндбук не содержит справочной информации по командам и их ключам, а скорее представляет собой набор рецептов вида «как сделать то-то или как настроить что-то», при этом имеет четкую последовательность в повествовании - от простых тем к более сложным.

«FreeBSD Handbook» уникальная книга, написана понятным языком и легко читается как художественная литература «на ночь, перед сном» - настоятельно рекомендую!

Прошу обратить внимание на то, что если в очередном релизе операционной системы что-то меняется или какая-то из стандартных операций претерпевает изменения, то соответствующие изменения вносятся и в хэндбук для этой версии ОС. Поэтому будьте внимательны и обращайте внимание на версию хэндбука который Вы читаете.

Помимо хэндбука, по ОС FreeBSD написаны и другие книги, так же доступные в электронном виде, их список можно посмотреть тут: https://docs.freebsd.org/en/books/

Существует печатная версия хэндбука, а на сайте FreeBSD Mall в продаже находится 3-е издание хэндбука за $7 описывающее FreeBSD версии 4.x и 5.x — несколько старовато, но для начинающих рекомендую к прочтению, да бы полностью погрузиться в этот страшный подземный мир FreeBSD.

На этом же сайте есть в продаже еще одна, совсем свежая, книга - FreeBSD Mastery: ZFS за $23. Не читал, но одобряю.

Еще одна печатная книга по «потрохам» ОС FreeBSD: «Design and Implementation of the FreeBSD Operating System, 2nd Edition» за авторством МакКузика и сотоварищей. Цена кусачая - $65 на Амазоне, но книга очень стоящая, особенно для студентов изучающих операционные системы и системное программирование. К этой книге МакКузик выпустил курс видео-лекций в составе 12 видеороликов по одному часу. Лекции платные, но для затравки на Youtube доступна первая часть: FreeBSD Kernel Internals, Dr. Marshall Kirk McKusick

8. Менеджмент пакетов в ОС FreeBSD

Так же, как и у многих дистрибутивов ОС Linux, у ОС FreeBSD есть свой репозиторий с огромным количеством портированных под неё свободных (open source) программ, библиотек и всего прочего, что у нас называют аббревиатурой СПО. На момент написания данной статьи количество этого СПО составляло 38 487 единиц. Однако, в отличии от большинства Linux дистрибутивов, в ОС FreeBSD есть целых два репозитория: первый называется FreeBSD Ports Collection или в простонародье - «порты», содержит он исходные коды портированных на FreeBSD свободных программ, набор соответствующих патчей к ним и Makefile-ы для сборки (компиляции). Патчи прикладываются к оригинальным исходным кодам автоматически в процессе сборки соответствующего порта, чтобы он без проблем компилировался и исполнялся в среде ОС FreeBSD. Во FreeBSD, как минимум, своя специфическая структура каталогов и почти все программы требуется пропатчить на соответствие этой структуре, но об этом чуть позже. Все порты разбиты на категории и подкатегории и на машине пользователя представляют собой набор Makefile-ов который хранится в хорошо структурированном подкаталоге **/usr/ports/**. Набор этих файлов можно установить в процессе инсталляции системы либо скачать отдельно. Управление портами осуществляется с помощью утилиты **portsnap**, цитирую:

```
rz@butterfly:~ % man portsnap
PORTSNAP(8)             FreeBSD System Manager's Manual             PORTSNAP(8)

NAME
     portsnap - fetch and extract compressed snapshots of the ports tree

SYNOPSIS
     portsnap [-I] [-d workdir] [-f conffile] [-k KEY] [-l descfile]
              [-p portsdir] [-s server] command ... [path]

DESCRIPTION
     The portsnap tool is used to fetch and update compressed snapshots of the
     FreeBSD ports tree, and extract and update an uncompressed ports tree.

     In a normal update operation, portsnap will routinely restore modified
     files to their unmodified state and delete unrecognized local files.
```

Второй репозиторий содержит так называемые пакеты (**packages**). Это уже скомпилированные и готовые к работе пакеты программ, библиотек и всего того, что есть в портах, в *бинарном* виде. Пакет как правило содержит некоторое количество метаданных о себе: краткое и полное описание содержимого, адрес и имя мантейнера, а так же список зависимостей от других пакетов. Для управления пакетами предназначена утилита **pkg**, цитата:

```
rz@butterfly:~ % man pkg
PKG(8)                  FreeBSD System Manager's Manual                  PKG(8)

NAME
     pkg, pkg-static – manipulate packages

SYNOPSIS
     pkg [-v] [-d] [-l] [-N]
         [-j ⟨jail name or id⟩ | -c ⟨chroot path⟩ | -r ⟨root directory⟩]
         [-C ⟨configuration file⟩] [-R ⟨repository configuration directory⟩]
         [-4 | -6] ⟨command⟩ ⟨flags⟩

     pkg [--version] [--debug] [--list] [-N]
         [--jail ⟨jail name or id⟩ |
         --chroot ⟨chroot path⟩ | --rootdir ⟨root directory⟩]
         [--config ⟨configuration file⟩]
         [--repo-conf-dir ⟨repository configuration directory⟩] [-4 | -6]
         ⟨command⟩ ⟨flags⟩

DESCRIPTION
     pkg provides an interface for manipulating packages: registering, adding,
     removing and upgrading packages.  pkg-static is a statically linked
     variant of pkg typically only used for the initial installation of pkg.
```

Утилита **pkg** отслеживает список зависимостей, доустанавливает необходимые пакеты или удаляет неиспользуемые пакеты и файлы при деинсталляции какого-то пакета. Фактически это аналог RPM пакетов в ОС Linux, но есть нюансы. Один из них состоит в том, что репозиториев с пакетами не один, а целых два: **latest** и **quarterly** — первый содержит самые новые изменения в пакетах (новые пакеты, самые последние патчи и обновления к ним), а второй отстает от первого по времени на четверть года, т. е. содержит более стабильные и объезженные версии пакетов. Но и это еще не все. Для каждой поддерживаемой версии FreeBSD ведется свой набор репозиториев с пакетами (свой **latest** и **quarterly**) и они не совпадают друг с другом. Файл конфигурации **/etc/pkg/FreeBSD.conf** указывает какой именно репозиторий будет использован утилитой **pkg**:

```
rz@butterfly:~ % cat /etc/pkg/FreeBSD.conf
# $FreeBSD$
#
# To disable this repository, instead of modifying or removing this file,
# create a /usr/local/etc/pkg/repos/FreeBSD.conf file:
#
#   mkdir -p /usr/local/etc/pkg/repos
#   echo "FreeBSD: { enabled: no }" > /usr/local/etc/pkg/repos/FreeBSD.conf
#

FreeBSD: {
  url: "pkg+http://pkg.FreeBSD.org/${ABI}/quarterly",
  mirror_type: "srv",
  signature_type: "fingerprints",
  fingerprints: "/usr/share/keys/pkg",
  enabled: yes
}
```

При установке ОС FreeBSD «с нуля» соответствующий файл будет создан инсталлятором автоматически, он будет настроен на репозиторий **quarterly** для нашей версии FreeBSD. Так как меня интересует свежачок, то я создал аналогичный файл, но в другом месте файловой системы (обратите внимание, об этом сказано в комментарии) и в нём изменил значение параметра **url** на **"pkg+http://pkg.FreeBSD.org/${ABI}/latest"**:

```
rz@butterfly:~ % cat /usr/local/etc/pkg/repos/FreeBSD.conf
FreeBSD: {
  url: "pkg+http://pkg.FreeBSD.org/${ABI}/latest",
```

```
  mirror_type: "srv",
  signature_type: "fingerprints",
  fingerprints: "/usr/share/keys/pkg",
  enabled: yes
}
```

Здесь переменная **${ABI}** задает версию FreeBSD и подставляется автоматически утилитой **pkg**.

И так, краткий справочник по команде **pkg** от Кэпа:

1. Обновление репозитория:

% sudo pkg update

2. Поиск пакета по текстовому описанию:

% pkg search <some_text>

3. Установка пакета по его имени:

% sudo pkg install <package>
# переустановка:
% sudo pkg install -f <package>

4. Апгрейд пакета на его более новую версию:

% sudo pkg upgrade <package>

5. Снос пакета:

% sudo pkg delete <package>

6. Просмотр списка установленных пакетов:

% pkg info

7. Просмотр информации об установленном пакете:

% pkg info <package>

Пример использования:

```
rz@butterfly:~ % pkg info zip-3.0_1
zip-3.0_1
Name           : zip
Version        : 3.0_1
Installed on   : Mon Mar 14 03:48:02 2022 +05
Origin         : archivers/zip
Architecture   : FreeBSD:13:amd64
Prefix         : /usr/local
Categories     : archivers
Licenses       : BSD3CLAUSE
Maintainer     : ler@FreeBSD.org
WWW            : http://infozip.sourceforge.net/Zip.html
```

```
Comment        : Create/update ZIP files compatible with PKZIP
Options        :
        DOCS           : on
Annotations    :
        FreeBSD_version: 1300139
        cpe            : cpe:2.3:a:info-zip_project:zip:3.0:::::freebsd13:x64:1
        repo_type      : binary
        repository     : FreeBSD
Flat size      : 739KiB
Description    :
Zip is a compression and file packaging utility.  It is compatible with
PKZIP 2.04g (Phil Katz ZIP) for MSDOS systems.  There is a companion to zip
called unzip (of course) which you can also install from the ports/package
system.

WWW: http://infozip.sourceforge.net/Zip.html
```

К слову, порты тоже отслеживают зависимости, поэтому запустив сборку, скажем, **firefox** из портов Вы рискуете выкачать и скомпилировать весь богатый мир СПО на своей машине, будьте готовы к такому повороту событий. ;-)

Для прочтения хэндбука и посещения web сайтов из терминала нам понадобится утилита **lynx** которая представляет собой консольный web браузер. Установим её следующей командой:

```
rz@butterfly:~ % pkg search lynx
ja-lynx-2.8.9.r1               Console WWW client (browser) with multi-byte encoding support
ja-lynx-current-2.9.0.d4       Console WWW client (browser) with multi-byte encoding support
(development release)
libretro-beetle_lynx-0.20220327 Standalone port of Mednafen Lynx to libretro, itself a fork
of Handy
lynx-2.8.9.1_1,1               Non-graphical, text-based World-Wide Web client
lynx-current-2.9.0d10          Console-based web browser (current/development version)

rz@butterfly:~ % sudo pkg inst lynx-2.8.9.1_1,1
```

9. Замечание по структуре каталогов в ОС FreeBSD

Небольшое замечание по структуре дерева каталогов. В ОС FreeBSD идеологически постулируется, что всё, что устанавливается или изменяется суперпользователем должно располагаться в подкаталоге **/usr/local**. Так, бинарные файлы устанавливаемые утилитой **pkg** кладутся в подкаталог **/usr/local/bin** или **/usr/local/sbin**, библиотеки кладутся в **/usr/local/lib**, а файлы настроек и скриптов автозапуска в **/usr/local/etc/** и **/usr/local/etc/rc.d**. Иными словами, данные пользователя не перемешиваются с данными системы. На первый взгляд это кажется несущественным, но это важный принципиальный момент, который отличает ОС FreeBSD от большинства дистрибутивов Linux.

Следствие у этого принципа такое: почти все конфигурационные файлы от устанавливаемых отдельно утилит (пакетов или портов) находятся в подкаталоге **/usr/local/etc/**, а не **/etc/** как это принято в других подобных ОС.

10. Установка компилятора

Как любая UNIX-совместимая операционная система, операционные системы группы BSD разрабатывались программистами для программистов. А это означает, что основным инструментом пользователя, после командной оболочки, являются компиляторы с языков C, C++, FORTRAN и assembler, а так же утилиты для работы с ними. ОС FreeBSD унаследовала этот идеологический момент и преумножила его: предполагается, что всё, что окружает пользователь в системе должно быть собираемо из исходных кодов, от самого компилятора и

ядра ОС до библиотек, системных утилит и приложений пользователя. Существование портов, собственно, и подтверждает данный тезис.

Одной из давних особенностей системы является то, что Вы можете установить самый минимум операционной системы на поддерживаемую аппаратную платформу, далее зайти в каталог **/usr/src** и ввести команду **make buildworld**, после чего система выкачает базовый компилятор, затем самые последние исходники компилятора, соберет его, а с его помощью соберет ядро и все остальные части системы, т. е. воссоздаст себя буквально из ничего. Разумеется тут есть масса нюансов и даже продвинутому пользователю вряд ли когда либо придется «создавать мир», тем не менее такая возможность имеется и она используется для тестирования нового «железа» или новой версии операционной системы на собираемость, совместимость и выносливость - если система не упала в «panic» в процессе сборки своего мира, то все в полном поряде!

Другой момент состоит в том, что некоторые редко используемые особенности системы требуют включения в ядро ОС определенных участков кода которые отключены по-умолчанию. И не смотря на то, что разработчиками ОС проделан большой труд по разделению частей монолитного ядра на отдельные загружаемые модули (Kernel Loadable Modules), пользователю всё еще может потребоваться сборка кастомного ядра и пересборка отдельных модулей/драйверов. Пересобрать ядро так же придется и при наложении определенных патчей связанных с безопасностью ОС или её стабильностью. По мимо этого, драйверы устройств для ОС FreeBSD часто поставляются в виде исходных кодов (NVIDIA).

Поэтому любому пользователю FreeBSD потребуется компилятор, точнее целый набор компиляторов и инструментария разработчика (toolchain), а так же некоторые знания как с этим добром обращаться. Начиная с версии 10-RELEASE, в ОС FreeBSD используется C/C++ toolchain на базе компилятора Clang (LLVM) вместо привычного, для многих пользователей ОС Linux, GNU Compiler Collection (GCC). Связано это в первую очередь с лицензией: лицензия GPLv3, которой покрывается компилятор GCC, противоречит идеалам BSD и создавала много трудностей для коммерческих применений FreeBSD. Цитата с известного сайта:

> *The primary reason for switching from GCC to Clang is the incompatibility of GCC's GPL v3 license with the goals of the FreeBSD project. There are also political issues to do with corporate investment, as well as user base requirements. Finally, there are expected technical advantages to do with standards compliance and ease of debugging. Real world performance improvements in compilation and execution are code-specific and debatable; cases can be made for both compilers.*

Откровенно говоря, тезис на счет лицензионной несовместимости весьма спорный. Почему-то лицензия GPLv3 не мешает никому предлагать свои проприетарные разработки на основе Linux даже без открытия кода, почему же это вдруг стало проблемой для BSD ? По этому поводу в интернетах написано много статей, много флейма на форумах, но ясности они не придают. Короче, в ОС FreeBSD используем **Clang**.

С точки зрения меня как программиста и разработчика, Clang гораздо приятнее в обращении, но разрабатывать мне приходится всё равно под Linux и с использованием GCC.

Для установки Clang/LLVM 13.0.1 воспользуемся командой **pkg search** для поиска нужного нам пакета:

```
rz@butterfly:~ % pkg search llvm13
intel-compute-runtime-llvm13-22.24.23453 OpenCL implementation for Intel HD 5000 (Gen8) or
newer
```

```
intel-graphics-compiler-llvm13-1.0.11485 Intel Graphics Compiler for OpenCL
llvm13-13.0.1_3                LLVM and Clang
opencl-clang-llvm13-13.0.0     Clang wrapper to compile OpenCL C kernels to SPIR-V modules
spirv-llvm-translator-llvm13-13.0.0 Bi-directional translation between SPIR-V and LLVM IR
vc-intrinsics-llvm13-0.5.0     LLVM intrinsics for SIMD on GPU
```

Видно, что в репозитории имеется несколько пакетов с подобным именем. Нас интересует сам компилятор, а значит будем устанавливать пакет **llvm13-13.0.1_3**:

```
rz@butterfly:~ % sudo pkg install llvm13-13.0.1_3
```

Утилита **pkg** проведет анализ зависимостей и предложит нам установить (или удалить и переустановить) некоторое количество пакетов от которых зависит пакет llvm13-13.0.1_3. И после получения подтверждения от пользователя запустит процесс выкачивания и установки пакетов, последним из которых будет сам **llvm** (**Clang**). Если Вы видите, что что-то пошло «не так», не бойтесь нажать Ctrl-C — ничего не сломается. В крайнем случае Вы всегда можете переустановить пакет командой **pkg install -f**.

После окончания установки убедимся, что компилятор присутствует в путях поиска исполняемых файлов и может быть вызван, так же не забываем про команду **rehash**:

```
rz@butterfly:~ % rehash

rz@butterfly:~ % cc -v
FreeBSD clang version 13.0.0 (git@github.com:llvm/llvm-project.git llvmorg-13.0.0-0-
gd7b669b3a303)
Target: x86_64-unknown-freebsd13.1
Thread model: posix
InstalledDir: /usr/bin
```

Для того, чтобы у нас была возможность вести сборку СПО, в том числе и из коллекции портов, нам потребуется утилита **make**, точнее её GNU разновидность (или GNU make). В ОС FreeBSD есть своя, родная утилита **make** родом из Berkley и известная как «BSD make» и она, как Вы догадываетесь, не совместима с «GNU make» которая сейчас широко используется в среде Linux. Так же нам потребуется еще пара широко применяемых утилит, а именно **CMake** и **Git**. Поэтому установим соответствующие пакеты одним вызовом команды **pkg**:

```
rz@butterfly:~ % sudo pkg inst gmake cmake git && rehash
```

Обратите внимание на то, что в данном случае я указываю только начальные строки названий пакетов, а вместо команды **install** даю её сокращенный вид **inst**. Всё это фичи утилиты **pkg**.

11. Проверка работоспособности компилятора и системы сборки

Проверим на сколько наша система готова к компиляции портов. Для этого пойдем в каталог **/usr/ports/** и чего нибудь там соберем, например широко известную в узких кругах игру **NetHack**:

```
rz@butterfly:~ % cd /usr/ports/games/nethack33-nox11/

rz@butterfly:/usr/ports/games/nethack33-nox11 % sudo make BATCH=yes
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 You may use the following build options:
 WITH_TTY_GRAPHICS=yes              build with no GUI
 WITHOUT_X11=yes                    same as above
```

```
 By default, nethack port is built with X11 GUI.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
===>   nethack33-nox11-3.3.1_11 depends on file: /usr/local/sbin/pkg - found
=> nethack-331.tgz doesn't seem to exist in /usr/ports/distfiles/.
=> Attempting to fetch
https://downloads.sourceforge.net/project/nethack/nethack/3.3.1/nethack-331.tgz
nethack-331.tgz                                    2989 kB  848 kBps    04s
===> Fetching all distfiles required by nethack33-nox11-3.3.1_11 for building
===>  Extracting for nethack33-nox11-3.3.1_11
=> SHA256 Checksum OK for nethack-331.tgz.
===>  Patching for nethack33-nox11-3.3.1_11
===>  Applying FreeBSD patches for nethack33-nox11-3.3.1_11 from /usr/ports/games/nethack33-
nox11/../nethack33/files
===>   nethack33-nox11-3.3.1_11 depends on package: gmake>=4.3 - found
===>  Configuring for nethack33-nox11-3.3.1_11
Copying Makefiles.
===>  Building for nethack33-nox11-3.3.1_11
…
…
install  -m 444 nethack.6
/usr/ports/games/nethack33-nox11/work/stage/usr/local/man/man6/nethack33.6
install  -m 444 lev_comp.6
/usr/ports/games/nethack33-nox11/work/stage/usr/local/man/man6/lev_comp33.6
install  -m 444 dgn_comp.6
/usr/ports/games/nethack33-nox11/work/stage/usr/local/man/man6/dgn_comp33.6
install  -m 444 recover.6
/usr/ports/games/nethack33-nox11/work/stage/usr/local/man/man6/recover33.6
install  -m 444 dlb.6 /usr/ports/games/nethack33-nox11/work/stage/usr/local/man/man6/dlb33.6
install  -m 0644 /usr/ports/games/nethack33-nox11/work/nethack-3.3.1/doc/Guidebook.txt
/usr/ports/games/nethack33-nox11/work/stage/usr/local/share/doc/nethack33
====> Compressing man pages (compress-man)
```

Из вывода видно, что по команде **make BATCH=yes** были прослежены зависимости, выкачаны и распакованы соответствующие **distfiles** - архивы с исходными кодами затаренные гнутым зипом, выкачаны и приложены ряд патчей, после чего запущена сборка и подготовка к инсталляции собранного порта.

Параметр **BATCH=yes** указывает системе сборки на то, что на все глупые вопросы следует отвечать утвердительно (Yes Man) и не отвлекать пользователя от наслаждения чашечкой горячего чая.

Далее, для того чтобы установить собранный порт выполним команду:

```
rz@butterfly:/usr/ports/games/nethack33-nox11 % sudo make install BATCH=yes
===>  Installing for nethack33-nox11-3.3.1_11
===>  Checking if nethack33-nox11 is already installed
===>   Registering installation for nethack33-nox11-3.3.1_11
Installing nethack33-nox11-3.3.1_11...
===> SECURITY REPORT:
      This port has installed the following binaries which execute with
      increased privileges.
/usr/local/share/nethack33/recover
/usr/local/share/nethack33/nethack33

      If there are vulnerabilities in these programs there may be a security
      risk to the system. FreeBSD makes no guarantee about the security of
      ports included in the Ports Collection. Please type 'make deinstall'
      to deinstall the port if this is a concern.

      For more information, and contact details about the security
      status of this software, see the following webpage:
http://www.nethack.org/
```

В процессе инсталляции, утилита **install** выдает нам сообщение о том, что ряд исполняемых файлов в составе этого порта могут выполняться с привилегиями отличными от пользовательских, что может быть небезопасно в случае обнаружения уязвимостей.

Выясним, что за привилегии:

```
rz@butterfly:/usr/ports/games/nethack33-nox11 % ll /usr/local/share/nethack33/recover
/usr/local/share/nethack33/nethack33
-rwxr-sr-x  1 games  games  1713664 Jul  9 17:01 /usr/local/share/nethack33/nethack33*
-rwxr-sr-x  1 games  games    19744 Jul  9 17:01 /usr/local/share/nethack33/recover*
```

Видно, что установлен атрибут **SGID**, присвоены пользователь и группа **games**, т. е. игра **NetHack** исполняется от эффективного имени пользователя **games**. Будем считать что это относительно безопасно.

Если какой-то из установленных портов Вам покажется небезопасным, то его можно легко де-инсталлировать командой **make deinstall** из этого же каталога.

Любой порт можно установить повторно, дабы исправить поврежденные файлы, командой **make reinstall** (аналог команды **pkg install -f** для пакетов).

Проверим, запускается ли игра:

```
rz@butterfly:/usr/ports/games/nethack33-nox11 % rehash
rz@butterfly:/usr/ports/games/nethack33-nox11 % nethack

NetHack, Copyright 1985-2000
         By Stichting Mathematisch Centrum and M. Stephenson.
         See license for details.


Shall I pick a character for you? [ynq] q
```

Вроде бы все в порядке, игра запускается, а значит компилятор и система сборки у нас работают нормально. Выйдем из NetHack нажав клавишу **q** и продолжим заниматься установкой и настройкой ОС FreeBSD.

К игре NetHack мы еще вернемся когда начнем устанавливать **X11** и графическую оболочку **Xfce**. Так как процесс их установки не быстрый, то NetHack позволит нам немного расслабиться и провести время с пользой.

12. Утилита sysrc

Утилита **sysrc** позволяет безопасно (без внесения синтаксических ошибок) отредактировать системный конфигурационный файл из командной стоки. Вот, что сообщает нам страница системной документации по этой утилите:

```
rz@butterfly:~ % man sysrc
SYSRC(8)               FreeBSD System Manager's Manual               SYSRC(8)

NAME
     sysrc - safely edit system rc files

SYNOPSIS
     sysrc [-cdDeEFhinNqvx] [-s name] [-f file] [-j jail | -R dir]
           name[[+|-]=value] ...
     sysrc [-cdDeEFhinNqvx] [-s name] [-f file] [-j jail | -R dir] -a | -A
     sysrc [-E] [-s name] [-f file] -l
     sysrc [-eEqv] -L [name ...]

DESCRIPTION
```

```
     The sysrc utility retrieves rc.conf(5) variables from the collection of
     system rc files and allows processes with appropriate privilege to change
     values in a safe and effective manner.
```

Что-ж, очень полезно, возмём на вооружение!

13. Проблема с загрузчиком

В процессе установки одного из требуемых мне драйверов неожиданно обнаружилось, что во FreeBSD 13.0 и 13.1 есть баг приводящий к зависанию начального загрузчика (booloader-a) если в его конфиге (в файле **/boot/loader.conf**) присутствуют ссылки на любые модули. Честно сказать я не стал сильно долго разбираться с причинами такого поведения bootloader-a на моём ноутбуке, но сделал вывод, что загружать модули из **/boot/loader.conf** нельзя. Вместо этого я буду загружать их из **rc** скриптов, т. е. буду добавлять имена нужных мне для загрузки модулей в файл **/etc/rc.conf** в строку **kld_list** разделяя пробелом.

На случай если кто-то из будущих фрюховодов случайно наткнулся на эту проблему (то есть Ваша система намертво подвисла сразу после старта загрузчика или после загрузки какого-то «нерадивого» модуля), то сообщаю, что имеется способ загрузить «голое» ядро системы без попытки загрузки модулей загрузчиком. Делается это так:

1. При загрузке в меню выбрать «**3. Escape to loader prompt**».
2. В появившемся промпте **Ok** от загрузчика ввести следующие команды:

   unload

   load /boot/kernel/kernel

   или

   load /boot/GENERIC/kernel

Это проинструктирует загрузчик о том, что всё, что было записано и загружено через **loader.conf** нужно выгрузить и загрузить «голое» ядро GENERIC установленное при инсталляции ОС. Более подробно о процессе загрузки ОС FreeBSD и командах загрузчика читайте в статье хэндбука: https://docs.freebsd.org/en/books/handbook/boot/

14. Управление электропитанием

Управление электропитанием на ПК типа ноутбук является немаловажным компонентом системы и в ОС FreeBSD имеются все необходимые для этого средства. В частности, FreeBSD поддерживает промышленный стандарт управления электропитанием «Advanced Configuration and Power Management» (ACPI), для чего в ядре ОС имеется специальный драйвер — **acpi**.

14.1 Настройка режимов «сна»

Прежде всего ознакомимся со страницей руководства по драйверу **acpi**:

```
rz@butterfly:~ % man acpi
ACPI(4)                FreeBSD Kernel Interfaces Manual                ACPI(4)

NAME
     acpi – Advanced Configuration and Power Management support

SYNOPSIS
     device acpi
```

```
     options ACPI_DEBUG
     options DDB

DESCRIPTION
     The acpi driver provides support for the Intel/Microsoft/Compaq/Toshiba
     ACPI standard.  This support includes platform hardware discovery
     (superseding the PnP and PCI BIOS), as well as power management
     (superseding APM) and other features.  ACPI core support is provided by
     the ACPI CA reference implementation from Intel.

     Note that the acpi driver is automatically loaded by the loader(8), and
     should only be compiled into the kernel on platforms where ACPI is
     mandatory.

SYSCTL VARIABLES
     The acpi driver is intended to provide power management without user
     intervention.  If the default settings are not optimal, the following
     sysctls can be used to modify or monitor acpi behavior.  Note that some
     variables will be available only if the given hardware supports them
     (such as hw.acpi.acline).
```

Как следует из документации, драйвер **acpi** подгружается автоматически начальным загрузчиком (boot loader-ом), а текущие настройки и состояние подсистемы ACPI можно выяснить с помощью команды **sysctl** просмотрев дерево переменных MIB в ветке **hw.acpi**, как показано ниже:

```
rz@butterfly:~ % sysctl -a hw.acpi
hw.acpi.acline: 1                    # ←-- указывает на наличие внешнего электропитания.
hw.acpi.battery.info_expire: 5
hw.acpi.battery.units: 1
hw.acpi.battery.state: 0
hw.acpi.battery.rate: 0
hw.acpi.battery.time: -1
hw.acpi.battery.life: 95
hw.acpi.cpu.cx_lowest: C8
hw.acpi.thermal.tz0._TSP: 20
hw.acpi.thermal.tz0._TC2: 2
hw.acpi.thermal.tz0._TC1: 1
hw.acpi.thermal.tz0._ACx: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
hw.acpi.thermal.tz0._CRT: 125.1C
hw.acpi.thermal.tz0._HOT: -1
hw.acpi.thermal.tz0._PSV: -1
hw.acpi.thermal.tz0.thermal_flags: 0
hw.acpi.thermal.tz0.passive_cooling: 0
hw.acpi.thermal.tz0.active: -1
hw.acpi.thermal.tz0.temperature: 54.1C
hw.acpi.thermal.user_override: 0
hw.acpi.thermal.polling_rate: 10
hw.acpi.thermal.min_runtime: 0
hw.acpi.reset_video: 0
hw.acpi.handle_reboot: 1
hw.acpi.disable_on_reboot: 0
hw.acpi.verbose: 0
hw.acpi.s4bios: 0
hw.acpi.sleep_delay: 1
hw.acpi.suspend_state: S5
hw.acpi.standby_state: NONE
hw.acpi.lid_switch_state: NONE
hw.acpi.sleep_button_state: S3
hw.acpi.power_button_state: S5
hw.acpi.supported_sleep_state: S3 S4 S5
```

Данная команда предоставляет много интересной информации о поведении операционной системы при различных условиях. Видно, что система и аппаратура поддерживает ряд состояний энергосбережения для режима «suspend» (режима «сна»), а именно состояния **S3**, **S4** и **S5**. Из той же страницы руководства **man acpi** можно выяснить, что это за состояния:

```
hw.acpi.supported_sleep_state
        Suspend states (S1-S5) supported by the BIOS.

        S1      Quick suspend to RAM.  The CPU enters a lower power
                state, but most peripherals are left running.

        S2      Lower power state than S1, but with the same basic
                characteristics.  Not supported by many systems.

        S3      Suspend to RAM.  Most devices are powered off, and the
                system stops running except for memory refresh.

        S4      Suspend to disk.  All devices are powered off, and the
                system stops running.  When resuming, the system starts
                as if from a cold power on.  Not yet supported by FreeBSD
                unless S4BIOS is available.

        S5      System shuts down cleanly and powers off.
```

Т.е. операционная система и аппаратура на данном ноутбуке поддерживает состояния: «Suspend to RAM», «Suspend to disk» и «Power off» для команды перехода в режим «сна».

Настроим режим «сна» для нашей ОС. Состояние «Suspend to disk» (S4) в ОС FreeBSD не поддерживается если в ПК не установлен соответствующий BIOS, да и на системах с небольшим SSD это состояние мало интересно, так как потребуется зарезервировать огромную часть диска для образа RAM, а запись содержимого RAM на диск в процессе перехода в режим «сна» может занять достаточно длительное время. В общем, для режима «сна» у нас имеется практически только один вариант - «Suspend to RAM» (S3), поэтому установим значение MIB переменной **hw.acpi.suspend_state** в значение **S3** следующей командой:

```
rz@butterfly:~ % sudo sysctl -w hw.acpi.suspend_state=S3
hw.acpi.suspend_state: S5 -> S3
```

Для того, чтобы эта переменная устанавливалась в нужное значение каждый раз при загрузки ОС, добавим эту переменную в файл **/etc/sysctl.conf**:

```
rz@butterfly:~ % su
root@butterfly:/home/rz # echo hw.acpi.suspend_state=S3 >> /etc/sysctl.conf
```

Так же я хочу установить значение переменной **hw.acpi.power_button_state** равное **S3** чтобы мой ноутбук переходил в состояние «Suspend to RAM» при нажатии на кнопку питания (power button):

```
root@butterfly:/home/rz # echo hw.acpi.power_button_state=S3 >> /etc/sysctl.conf
root@butterfly:/home/rz # sysctl -w hw.acpi.power_button_state=S3
hw.acpi.power_button_state: S5 -> S3
```

Аналогичным образом можно установить значение переменной **hw.acpi.lid_switch_state** которая задает состояние системы при закрывании крышки ноутбука. Для себя я оставлю эту переменную в состоянии **NONE**, т. е. без реакции — люблю слушать музыку с закрытым ноутбуком, чтобы дети по кнопкам не давили. :-)

Проверим что у нас получилось. Сначала выполним два раза команду **sync —** ритуальное действие направленное на то, чтобы сбросить содержимое кэша файловой системы на диск на случай если операционная система решит «повеситься на мертво». После этого однократно и кратковременно нажмем кнопку «Power» на корпусе ноутбука. Примерно через 2 секунды мы увидим, что операционная система выдала ряд системных сообщений в **dmesg**, тут же погасила экран и подсветку клавиатуры, т. е. перешла в режим «сна».

Попытаемся пробудить систему еще одним кратковременным нажатием кнопки «Power» - и, о чудо, через 1-2 секунды операционная система полностью возобновила работу и вернулась к состоянию «до сна». Таким образом мы успешно проделали цикл операций **suspend/resume**. Скажу честно, проблема пробуждения после операции **suspend** во FreeBSD ранних версий работала крайне отвратительно, но с версии 13.0-RELEASE ситуация радикально поменялась в лучшую сторону. Правда, и тут не обходится без нюансов — некоторые проприетарные драйверы, например от NVIDIA, до сих пор не поддерживают операции **suspend/resume**, но об этом будет сказано далее.

Очень часто случается, что операционная система зависает (или вовсе не реагирует) при попытке перехода в состояние «сна». Чтобы выяснить какой из драйверов или компонентов системы ведет себя неподабающим образом можно установить значение переменной **hw.acpi.verbose** в **1**. В этом случае ОС при переходе в режим «сна» будет выводить в **dmesg** подробную информацию о том, как каждый из загруженных драйверов отреагировал на операцию **suspend**, что может помочь выявить проблемный элемент и, при необходимости, ликвидировать его. Если Вы столкнулись с такой проблемой, то рекомендую прочитать статью https://wiki.freebsd.org/SuspendResume.

Отправить систему в состояние «сна» можно и из командной строки выполнив команду: **sudo zzz**, вот что написано в странице руководства об этой команде:

```
rz@butterfly:~ % man zzz
ZZZ(8)              FreeBSD System Manager's Manual              ZZZ(8)

NAME
     zzz – suspend an ACPI or APM system

SYNOPSIS
     zzz

DESCRIPTION
     The zzz utility checks for ACPI or APM support and then suspends the
     system appropriately.  For APM,

          apm -z

     will be issued.  For ACPI, the configured suspend state will be looked
     up, checked to see if it is supported and,

          acpiconf -s <state>

     will be issued.
```

14.2 Настройка управления частотой центрального процессора

Подавляющее большинство современных микропроцессоров и систем-на-кристалле обладают возможностью динамического регулирования основных частот тактирования своих

подсистем, в том числе и частоты центрального процессора. Делается это для уменьшения энергопотребления — на меньших частотах кремниевые микро- и нано-структуры во-первых потребляют меньший ток, а во-вторых — могут быть запитаны от меньшего напряжение, что в сумме дает меньшую потребляемую мощность. Очевидно, что на устройствах с батарейным питанием данная возможность оказалась весьма к месту — зачем потреблять электроэнергию и зазря разряжать батарею, если пользователь уснул за компьютером.

Подавляющее большинство современных операционных систем имеют средства для непрерывного контроля нагрузки на ОС и динамического подстраивания тактовой частоты (или группы частот) с одновременным регулированием напряжения питания для всех подсистем на кристалле микропроцессора. В ОС FreeBSD этой задачей занимается отдельный фоновый процесс — демоном (от слова «daemon»), который непрерывно мониторит нагрузку на систему посредством анализа группы счетчиков **dev.cpu.*.cx_usage_counters** и подает в систему команды через драйвер **acpi**. По-умолчанию данная функция в операционной системе отключена, т. е. демон не запущен и вычислительная система работает на номинальной тактовой частоте. В базовый комплект ОС входит демон **powerd**, и, в целом, он неплохо справляется со своей задачей, но настоящие фрюховоды устанавливают **powerd++** (**powerdxx**) который имеет несколько больший функционал и содержит в комплекте две полезные утилиты: **loadrec** и **loadplay**. Эти утилиты позволяют произвести «запись» нагрузочных показателей системы за определенный интервал времени и в последствии «проиграть» (просимулировать) нагрузку, проанализировать поведение системы и оценить работу настроек демона **powerdxx**. Сейчас я не буду рассказывать о тонкостях этого процесса, просто покажу как установить демона:

```
rz@butterfly:~ % sudo pkg inst --yes powerdxx
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf powerdxx_enable="YES" powerdxx_flags="-a maximum
-b adaptive"
powerdxx_enable:  -> YES
powerdxx_flags:  -> -a maximum -b adaptive
```

Параметры запуска (флаги) **-a maximum** и **-b adaptive** задают поведение демона при работе от источника переменного тока (АС) и от батареи соответственно. В данном случае я хочу, чтобы при питании от АС мой ноутбук работал на максимальной частоте, а при питании от батареи переходил на адаптивную схему управления тактовой частотой. Под «адаптивной» подразумевается, что демон будет отслеживать наличие нагрузки и плавно понижать или повышать тактовую частоту в некоторых (заданных по-умолчанию) пределах.

После установки и внесения его настроек в системный конфиг, запустим демона командой:

```
rz@butterfly:~ % sudo service powerdxx start
Starting powerdxx.
```

после чего быстро пробежимся взглядом по страницам руководства:

```
rz@butterfly:~ % man powerdxx
powerd++(8)             FreeBSD System Manager's Manual            powerd++(8)

NAME
     powerd++ - CPU clock speed daemon

SYNOPSIS
     powerd++ -h
     powerd++ [-vfN] [-a mode] [-b mode] [-n mode] [-m freq] [-M freq]
              [-F freq:freq] [-A freq:freq] [-B freq:freq] [-H temp:temp]
```

```
              [-t sysctl] [-p ival] [-s cnt] [-P file]

DESCRIPTION
     The powerd++ daemon monitors the system load and adjusts the CPU clock
     speed accordingly.  It is a drop-in replacement for powerd(8) and
     supports two modes of operation, a load feedback control loop or fixed
     frequency operation.

…

TOOLS
     The loadrec(1) and loadplay(1) tools offer the possibility to record
     system loads and replay them.

IMPLEMENTATION NOTES
     This section describes the operation of powerd++.

     Both powerd(8) and powerd++ have in common, that they work by polling
     kern.cp_times via sysctl(3), which is an array of the accumulated loads
     of every core. By subtracting the last cp_times sample the loads over the
     polling interval can be determined. This information is used to set a new
     CPU clock frequency by updating dev.cpu.0.freq.

   Initialisation
     After parsing command line arguments powerd++ assigns a clock frequency
     controller to every core. I.e. cores are grouped by a common
     dev.cpu.%d.freq handle that controls the clock for all of them. Due to
     limitations of cpufreq(4) dev.cpu.0.freq is the controlling handle for
     all cores, even across multiple CPUs. However powerd++ is not built with
     that assumption and per CPU, core or thread controls will work as soon as
     the hardware and kernel support them.
```

Руководство по **powerdxx** ссылается на два других руководства: **man loadrec** и **man loadplay**. Предоставлю читателю удовольствие ознакомиться с ними и немного поэкспериментировать самостоятельно. Так же в руководстве по **powerdxx** сказано, что узнать текущую частоту тактирования вычислительных ядер можно следующей командой:

```
rz@butterfly:~ % sysctl dev.cpu.0.freq
dev.cpu.0.freq: 3300
```

15. Настройка шедулера и прочих параметров ядра ОС

Раз уж мы взялись за тонкую настройки системы, то продолжим в этом же духе. Для эксплуатации ОС FreeBSD на персональном рабочем месте я рекомендую установить перечисленные ниже MIB переменные в соответствующие значения.

1) Так как я не собираюсь заниматься отладкой приложения для ОС FreeBSD на этом ноутбуке, то особой надобности в «корках» у меня нет. Отключить запись «корки» при падении пользовательских и системных процессов можно установив в **ноль** значение переменной **kern.coredump**, ровно как и мгновенно включить эту фичу обратно можно установив переменную в значение **1**:
   sysctl -w kern.coredump=0

2) На системах с графическим интерфейсом пользователя рекомендуется увеличить порог срабатывания шедулера при принудительном вытеснении процессов, что увеличит отзывчивость системы:
   sysctl -w kern.sched.preempt_thresh=224

3) Выключим PC speaker — его назойливый звук при работе в консоли меня раздражает:
   sysctl -w kern.vt.enable_bell=0

4) На система с SSD рекомендуется увеличить «read-ahead» - параметр указывающий объем считываемых наперед данных при работе с диском:
   sysctl -w vfs.read_max=128

5) Так как я собираюсь монтировать сетевые диски, то имеет смысл разрешить всем пользователям системы выполнять операцию монтирования если они являются владельцем точки монтирования. Это позволит мне выполнять команду **mount** без **sudo**, не светя лишний раз свой пароль, да и просто меньше кнопок давить:
   sysctl -w vfs.usermount=1

6) Имеет смысл увеличить размеры входных сетевых буферов для того, чтобы видео-стриминговые сервисы и VoIP приложения не теряли пакеты:
   sysctl -w net.inet.udp.recvspace=128000
   sysctl -w kern.ipc.maxsockbuf=8388608

Разумеется, чтобы операционная система устанавливала значения этих переменных каждый раз при загрузке необходимо добавить их в файл **/etc/sysctl.conf**, вот как его содержимое выглядит в моей системе:

```
rz@butterfly:~ % cat /etc/sysctl.conf
# $FreeBSD$
#
#  This file is read when going to multi-user and its contents piped thru
#  ``sysctl'' to adjust kernel values.  ``man 5 sysctl.conf'' for details.
#

# Uncomment this to prevent users from seeing information about processes that
# are being run under another UID.
#security.bsd.see_other_uids=0

# Do not write cores
kern.coredump=0

# Context switching: higher - more switches, les GUI lag. good for Youtube watching.
kern.sched.preempt_thresh=224

# Disable system bell (PC speaker) on VT
kern.vt.enable_bell=0

# Allow user who own mount points to perform mount file systems
vfs.usermount=1

# Enable shared memory
kern.ipc.shm_allow_removed=1

# Increase VFS read-ahead (better disk performance - particularly for SSDs)
```

```
# FreeBSD Default: 64
vfs.read_max=128

# Suspend to RAM by default
hw.acpi.suspend_state=S3

# Suspend to RAM when power button pressed
hw.acpi.power_button_state=S3

# Increase UDP buffer space and IPC socket buf space for Video streaming
net.inet.udp.recvspace=128000
kern.ipc.maxsockbuf=8388608
```

16. Запуск Wi-Fi

В главе «3. Установка ОС FreeBSD и первые звоночки» я писал, что сетевой Wi-Fi адаптер MediaTek MT7961 в ОС FreeBSD на данный момент не поддерживается и единственное решение здесь — заменить этот адаптер на что-то более распространенное и от более известного производителя. В ноутбуке Lenovo Ideapad 3 Gaming Wi-Fi адаптер устанавливается в стандартный слот M.2, так что с заменой проблемы быть не должно — нужно только выяснить как разобрать корпус ноутбука не разворотив и не сломав нежные пластиковые детали. Ответ на данный вопрос был быстро получен через Youtube — вот видео с разборкой Lenovo Ideapad 3 Gaming.

С выбором Wi-Fi адаптера пришлось немного повозиться. В процессе чтения форумов я пришел к выводу, что нужно искать адаптер производства Intel — с ними менее всего шансов нарваться на проблемы. Пробежавшись по известным интернет-магазинам я обнаружил единственный доступный в наличии вариант: Intel(R) Dual Band Wireless AC 7265, который по счастливой случайности поддерживается во FreeBSD, поэтому тут же его заказал вместе с доставкой. Данный сетевой адаптер работает в двух диапазонах 2.4 ГГц и 5 ГГц, поддерживает стандарты **802.11a/b/g/n/ac**, что в теории могло позволить работать со скоростью 500 Mbit/s.

Пока приобретенный мной адаптер доставляется, я решил потратить один вечер на более плотное изучение «драйверного» вопроса. И вот, что я выяснил. Для этого адаптера в ОС FreeBSD существуют два разных драйвера: **iwm** и **iwlwifi**.

Драйвер **iwm** это «родной» фрюшный драйвер, достаточно старый и хорошо отлаженный, но есть одна ложка дёгтя — он не поддерживает стандарт 802.11ac, а значит скорость выше 54 Mbit/s на нём не получить. Вот, что написано в системном руководстве на сей счет:

```
rz@butterfly:~ % man iwm
IWM(4)                  FreeBSD Kernel Interfaces Manual                  IWM(4)

NAME
     iwm - Intel IEEE 802.11ac wireless network driver

...

DESCRIPTION
     The iwm driver provides support for:

           Intel Dual Band Wireless AC 3160
           Intel Dual Band Wireless AC 3165
           Intel Dual Band Wireless AC 3168
           Intel Dual Band Wireless AC 7260
           Intel Dual Band Wireless AC 7265
           Intel Dual Band Wireless AC 8260
```

```
     Intel Dual Band Wireless AC 8265
     Intel Dual Band Wireless AC 9260
     Intel Dual Band Wireless AC 9270
     Intel Dual Band Wireless AC 946X
     Intel Dual Band Wireless AC 9560

iwm supports station mode operation.  Only one virtual interface may be
configured at any time.  For more information on configuring this device,
see ifconfig(8).

This driver requires the firmware built with the iwmfw module to work.

Currently, iwm only supports 802.11b and 802.11g modes.  It will not
associate to access points that are configured to operate only in 802.11n
or 802.11ac modes.
```

Второй драйвер - **iwlwifi**, сначала показался мне более интересным, но позже выяснилось, что это не так. Этот драйвер появился в ОС FreeBSD путем портирования одноименного драйвера из ядра Linux посредством фреймворка **LinuxKPI** (об этом фреймворке я расскажу чуть более подробно при установке и настройки драйверов графических ускорителей и подсистемы DRM).

Драйвер **iwlwifi** появился совсем недавно, с версии FreeBSD 13.1-RELEASE, т. е. его разработка активно ведется и сам автор на страничке https://wiki.freebsd.org/WiFi/Iwlwifi утверждает, что драйвер сыроват, а следовательно возможен «спонтанный выброс глюка». Помимо этого в системном руководстве имеется следующее замечание:

```
rz@butterfly:~ % man iwlwifi
iwlwifi(4)             FreeBSD Kernel Interfaces Manual             iwlwifi(4)

NAME
     iwlwifi – Intel IEEE 802.11a/b/g/n/ac/ax wireless network driver

...

DESCRIPTION
     The iwlwifi driver is derived from Intel's Linux iwlwifi driver and
     provides support for all chipsets supported by the mvm part of that
     driver.  iwlwifi will be a successor to iwm(4) and may superseed that
     driver in the future.  It still complements the iwn(4) driver which
     supports older chipsets.

     The driver uses the linuxkpi_wlan and linuxkpi compat framework to bridge
     between the Linux and native FreeBSD driver code as well as to the native
     net80211(4) wireless stack.

     While iwlwifi supports all 802.11 a/b/g/n/ac/ax the compatibility code
     currently only supports 802.11 a/b/g modes.  Support for 802.11 n/ac is
     to come. 802.11ax and 6Ghz support are planned.

BUGS
     Certainly.

SEE ALSO
     iwlwififw(4), iwm(4), iwn(4), wlan(4), ifconfig(8), wpa_supplicant(8)

HISTORY
     The iwlwifi driver first appeared in FreeBSD 13.1.
```

Иными словами, получить соединение с сетью Wi-Fi на скоростях выше 54 Mbit/s у меня прямо сейчас с этим Wi-Fi адаптером не выйдет — оба драйвера не поддерживают стандарт 802.11ac.

Что-ж, дареному коню, как известно… В общем я решил попробовать оба драйвера и понять, какой из них более пригоден для повседневной работы.

16.1 Установка и настройка сетевого драйвера iwm

Эксперименты я начал с установки и настройки родного фрюшного драйвер для Wi-Fi адаптеров производства Intel. Установка в данном случае слишком громкое слово, данный драйвер установлен в ОС FreeBSD что называется «из коробки». Для его включения достаточно подгрузить модули **if_iwm** (сам драйвер) и **iwm7265fw** (прошивка для адаптера), после чего настроить соответствующим образом **wpa_supplicant** и сетевой интерфейс.

Проделаем это по-шагам.

Прежде всего добавим загрузку «ядрёных» модулей драйвера в системный конфигурационный файл **/etc/rc.conf** указав их имена в списке в переменной **kld_list**. Сделаем это с помощью утилиты **sysrc**:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf kld_list+="if_iwm iwm7265fw"
kld_list:  -> if_iwm iwm7265fw
```

После чего необходимо ассоциировать драйвер **iwm** с сетевым интерфейсом **wlan0**, для этого в этот же файл **/etc/rc.conf** необходимо добавить стоку **wlans_iwm0="wlan0"** . Сделаем это следующей командой:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf wlans_iwm0="wlan0"
wlans_iwm0:  -> wlan0
```

Далее нужно настроить IP адрес для интерфейса **wlan0** или указать системе, что данный интерфейс должен получать IP адрес по протоколу DHCP. Делается это путем задания строки в **/etc/rc.conf** вида: **ifconfig_wlan0="XXX"** где XXX — перечень параметров интерфейса в том виде, как они передаются утилите **ifconfig** (на всякий случай рекомендую к прочтению страницу системной документации **man ifconfig**).

В моё случае я хочу, чтобы система запрашивала IP адрес по протоколу DHCP, а вся работа с обслуживанием ассоциаций в сети Wi-Fi проводилась с помощью системного демона **wpa_supplicant**, поэтому еще раз воспользовавшись утилитой **sysrc** укажу следующее:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf ifconfig_wlan0="WPA DHCP"
ifconfig_wlan0:  -> WPA DHCP
```

Теперь займемся настройками непосредственно Wi-Fi, т. е. ассоциируем адаптер с Wi-Fi точкой доступа. Для этого необходимо создать конфигурационный файл **/etc/wpa_supplicant.conf** следующего содержания:

```
rz@butterfly:~ % cat /etc/wpa_supplicant.conf

ctrl_interface=/var/run/wpa_supplicant
eapol_version=2
ap_scan=1
fast_reauth=1
```

```
network={
        ssid="GUEST_NETWORK1"
        scan_ssid=1
        psk="password1"
        priority=5
}
network={
       ssid="GUEST_NETWORK2"
       scan_ssid=1
       psk="password2"
       priority=1
}
```

Здесь блоками **network={...}** описываются отдельные точки доступа или имена Wi-Fi сетей. Внутри блоков задается имя сети (**ssid**), пароль (**psk**), приоритет при выборе (**priority**) и ряд других опциональных параметров. Параметр **scan_ssid** указывает на то, что данную сеть нужно опрашивать по имени (а не просто слушать эфир), так как точка доступа организующая эту сеть настроена в режиме «радиомолчания».

Все готово к подъему сетевого интерфейса. Загрузить модули драйвера без перезагрузки ОС можно следующей командой:

```
rz@butterfly:~ % sudo kldload if_iwm iwm7265fw
```

При этом, в **dmesg** должны появится следующие сообщения:

```
pci3: <network> at device 0.0 (no driver attached)
iwm0: <Intel(R) Dual Band Wireless AC 7265> mem 0xd1600000-0xd1601fff at device 0.0 on pci3
iwm0: hw rev 0x210, fw ver 22.361476.0, address 10:02:b5:b7:b1:e6
```

Создать и поднять сетевой интерфейс **wlan0** можно командой:

```
rz@butterfly:~ % sudo service netif restart wlan0

Created wlan(4) interfaces: wlan0.
Starting wpa_supplicant.
Starting Network: wlan0.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 10:02:b5:b7:b1:e6
        groups: wlan
        ssid "" channel 1 (2412 MHz 11g)
        regdomain FCC country US authmode WPA1+WPA2/802.11i privacy MIXED
        deftxkey UNDEF txpower 30 bmiss 10 scanvalid 60 protmode CTS wme
        roaming MANUAL bintval 0
        parent interface: iwm0
        media: IEEE 802.11 Wireless Ethernet autoselect (autoselect)
        status: no carrier
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

Видно, что система создала новый сетевой интерфейс **wlan0** и запустила демона **wpa_supplicant** для обслуживания этого интерфейса. В **dmesg** наблюдаем сообщения:

```
wlan0: Ethernet address: 10:02:b5:b7:b1:e6
wlan0: link state changed to UP
```

Через несколько секунд с помощью утилиты **ifconfig** можно посмотреть, удалось ли **wpa_supplicant**-у ассоциироваться с какой либо из заданных Wi-Fi точек доступа и какой IP адрес нам выдали:

```
rz@butterfly:~ % ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 10:02:b5:b7:b1:e6
        inet 192.168.168.230 netmask 0xffffff00 broadcast 192.168.168.255
        groups: wlan
        ssid GUEST_NETWORK1 channel 4 (2427 MHz 11g) bssid 52:ff:20:58:f8:23
        regdomain FCC country US authmode WPA2/802.11i privacy ON
        deftxkey UNDEF AES-CCM 2:128-bit txpower 30 bmiss 10 scanvalid 60
        protmode CTS wme roaming MANUAL
        parent interface: iwm0
        media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
        status: associated
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

Из вывода утилиты **ifconfig** видно, что установлено соединение по протоколу 802.11g на частоте 2427 МГц с точкой доступа MAC адрес которой 52:ff:20:58:f8:23 и мне выдали IP адрес 192.168.168.230. Сетевой интерфейс **wlan0** связан с родительским интерфейсом драйвера **iwm0**.

Не смотря на то, что моя домашняя точка доступа поддерживает режим 802.11ac, видно что в данном случае этот режим не доступен. Попробуем использовать во благо то, что есть.

16.2 Тестирование сетевого интерфейса с драйвером iwm

Настало время немного протестировать получившееся сетевое соединение при работе с драйвером **iwm**. Для этого установим утилиту **iperf3**:

```
rz@butterfly:~ % sudo pkg inst --yes iperf3
```

На другом хосте в локальной сети установим и запустим эту же утилиту с параметром **-s** (в режиме сервера):

```
rz@devbox:~$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Запустим тест на ноутбуке:

```
rz@butterfly:~ % iperf3 -c devbox
Connecting to host devbox, port 5201
[  5] local 192.168.168.230 port 34461 connected to 192.168.169.6 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  3.00 MBytes  25.1 Mbits/sec    0    202 KBytes
[  5]   1.00-2.00   sec  2.69 MBytes  22.5 Mbits/sec    0    221 KBytes
[  5]   2.00-3.00   sec  2.63 MBytes  22.1 Mbits/sec    0    238 KBytes
[  5]   3.00-4.00   sec  2.70 MBytes  22.6 Mbits/sec    0    254 KBytes
[  5]   4.00-5.00   sec  2.72 MBytes  22.8 Mbits/sec    0    268 KBytes
[  5]   5.00-6.00   sec  2.60 MBytes  21.8 Mbits/sec    0    282 KBytes
[  5]   6.00-7.00   sec  2.65 MBytes  22.2 Mbits/sec    0    295 KBytes
[  5]   7.00-8.00   sec  2.65 MBytes  22.2 Mbits/sec    0    308 KBytes
[  5]   8.00-9.00   sec  2.65 MBytes  22.2 Mbits/sec    0    321 KBytes
[  5]   9.00-10.00  sec  2.66 MBytes  22.3 Mbits/sec    0    332 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  26.9 MBytes  22.6 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  26.6 MBytes  22.3 Mbits/sec                  receiver

iperf Done.
```

Результат на лицо — суммарная скорость передачи данных (туда+обратно) составляет 22.6 + 22.3 = 44.9 Mbits/s, что очень близко к теоритически возможным 54Mbit/s для режима 802.11g учитывая сетевой оверхэд.

Меня интересовало как драйвер **iwm** будет работать с сетями 5 ГГц, поэтому я подключился в админский интерфейс к своей домашней точке доступа и заблокировал у неё режим 2.4 ГГц, т. е. оставил только 5 ГГц. Через несколько секунд сетевой интерфейс на моём ноутбуке переподключился и вот что я наблюдал по команде **ifconfig wlan0**:

```
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 10:02:b5:b7:b1:e6
        inet 192.168.168.230 netmask 0xffffff00 broadcast 192.168.168.255
        groups: wlan
        ssid GUEST_NETWORK1 channel 149 (5745 MHz 11a) bssid 50:ff:20:78:f8:23
        regdomain FCC country US authmode WPA2/802.11i privacy ON
        deftxkey UNDEF AES-CCM 2:128-bit txpower 23 bmiss 10 mcastrate 6
        mgmtrate 6 scanvalid 60 wme roaming MANUAL
        parent interface: iwm0
        media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11a
        status: associated
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

Видно, что сетевой адаптер успешно ассоциировался с точкой доступа по протоколу 802.11**a** (не путать с 802.11**ac**) на скорости 54Mbit/s что, согласно руководству, является максимально возможным для драйвера **iwm**.

Запустим **iperf3** еще раз и протестируем сетевое соединение в таком режиме:

```
rz@butterfly:~ % iperf3 -c devbox
Connecting to host devbox, port 5201
[  5] local 192.168.168.230 port 22358 connected to 192.168.169.6 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  3.08 MBytes  25.8 Mbits/sec    0    205 KBytes
[  5]   1.00-2.00   sec  2.74 MBytes  23.0 Mbits/sec    0    224 KBytes
[  5]   2.00-3.00   sec  2.70 MBytes  22.6 Mbits/sec    0    241 KBytes
[  5]   3.00-4.00   sec  2.75 MBytes  23.0 Mbits/sec    0    257 KBytes
[  5]   4.00-5.00   sec  2.72 MBytes  22.8 Mbits/sec    0    272 KBytes
[  5]   5.00-6.00   sec  2.73 MBytes  22.9 Mbits/sec    0    286 KBytes
[  5]   6.00-7.00   sec  2.73 MBytes  22.9 Mbits/sec    0    299 KBytes
[  5]   7.00-8.00   sec  2.74 MBytes  23.0 Mbits/sec    0    312 KBytes
[  5]   8.00-9.00   sec  2.77 MBytes  23.2 Mbits/sec    0    325 KBytes
[  5]   9.00-10.00  sec  2.72 MBytes  22.8 Mbits/sec    0    336 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  27.7 MBytes  23.2 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  27.3 MBytes  22.9 Mbits/sec                  receiver

iperf Done.
```

Результат стабильно одинаковый, что не может не радовать.

А теперь самый сложный тест: проверка на то, как драйвер **iwm** поддерживает режим «сна» и на сколько успешно выходит из него (цикл операций suspend/resume).

Однократное нажатие кнопки «Power» и… через пару секунд экран и подсветка клавиатуры ноутбука потухла. Делаю вывод, что система ушла в спячку успешно, поэтому пытаюсь пробудить её повторным нажатием кнопки «Power», и… чудо-чудное, система мгновенно пробуждается и возвращается к исходному состоянию.

Посмотрим какие у нас появились сообщения в **dmesg** относительно сетевого интерфейса и драйвера его обслуживающего:

```
rz@butterfly:~ % dmesg | grep 'iwm\|wlan\|acpi'
...
wlan0: link state changed to DOWN
acpi0: cleared fixed power button status
wlan0: link state changed to UP
```

Никаких сообщений об ошибках или таймаутах нет, всё логично. Я еще пару раз проделал этот тест с кнопкой «Power», дабы убедиться в повторяемости эффекта — всё в норме и это прекрасно!

16.3 Установка, настройка и тестирование драйвера iwlwifi

Как я уже упомянул выше, драйвер **iwlwifi** появился в ОС FreeBSD с версии 13.1-RELEASE, а это означает, что его так же просто установить «из коробки» как и драйвер **iwm**, но есть нюанс. Так, как данный драйвер находится в активной разработке, вместе с фреймворком LinuxKPI, то с момента выхода релиза ОС драйвер претерпел ряд изменений. И действительно, заглянув Git репозиторий исходных кодов ОС я обнаружил с десяток коммитов в дерево этого драйвера с даты релиза ОС. Что же тут предпринять ?

Во-первых, можно игнорировать эти изменения и попробовать то, что поставляется с релизом ОС. Во-вторых, можно проапгрейдить ОС FreeBSD до версии **13-STABLE**, т. е. перейти на ветку **-STABLE**. Попробуем оба варианта последовательно.

Первый вариант осуществить достаточно просто. Для этого нужно добавить модуль ядра **if_iwlwifi** в список **lkd_list** в системном конфигурационном файле **/etc/rc.conf**, при этом не забыть убрать из него модули **if_iwm** и **iwm7265fw** от драйвера **iwm** - два драйвера хоть и могут сосуществовать вместе, но для чистоты эксперимента выполним:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf kld_list-="if_iwm iwm7265fw"
kld_list: if_iwm iwm7265fw ->

rz@butterfly:~ % sudo sysrc -f /etc/rc.conf kld_list+="if_iwlwifi"
kld_list:  -> if_iwlwifi
```

Далее, необходимо ассоциировать сетевой интерфейс **wlan0** с устройством **iwlwifi0**:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf wlans_iwlwifi0="wlan0"
wlans_iwlwifi0:  -> wlan0
```

Собственно на этом все настройки заканчиваются — интерфейс **wlan0** и **wpa_supplicant** уже были ранее законфигурированы при настройки драйвера **iwm**. Поэтому можно либо перезагрузить систему командой **sudo reboot**, либо выгрузить модули предыдущего драйвера вручную и загрузить новый, после чего перезапустить сетевой интерфейс. Я, пожалуй, в данном случае предпочту перезагрузиться.

После перезагрузки операционной системы в **dmesg** я обнаружил следующие сообщения от драйвера **iwlwifi**:

```
rz@butterfly:~ % dmesg | grep iwl
iwlwifi0: <iwlwifi> mem 0xd1600000-0xd1601fff at device 0.0 on pci3
iwlwifi0: successfully loaded firmware image 'iwlwifi-7265D-29.ucode'
iwlwifi0: Found debug destination: EXTERNAL_DRAM
```

```
iwlwifi0: Found debug configuration: 0
iwlwifi0: loaded firmware version 29.4063824552.0 7265D-29.ucode op_mode iwlmvm
iwlwifi0: Detected Intel(R) Dual Band Wireless N 7265, REV=0x210
iwlwifi0: Applying debug destination EXTERNAL_DRAM
iwlwifi0: Allocated 0x00400000 bytes for firmware monitor.
iwlwifi0: base HW address: 10:02:b5:b7:b1:e6, OTP minor version: 0x0
iwlwifi0: Applying debug destination EXTERNAL_DRAM
iwlwifi0: Applying debug destination EXTERNAL_DRAM
iwlwifi0: FW already configured (0) - re-configuring
```

из чего я сделал вывод, что драйвер успешно загрузился и проинициализировал сетевой адаптер.

И действительно, не успел я войти в систему, как сетевой интерфейс **wlan0** был уже ассоциирован с Wi-Fi сетью и IP адрес был назначен по DHCP. Вот, что я увидел в ответ от утилиты **ifconfig**:

```
rz@butterfly:~ % ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 10:02:b5:b7:b1:e6
        inet 192.168.168.230 netmask 0xffffff00 broadcast 192.168.168.255
        groups: wlan
        ssid GUEST_NETWORK1 channel 149 (5745 MHz 11a) bssid 50:ff:20:78:f8:23
        regdomain FCC country US authmode WPA2/802.11i privacy ON
        deftxkey UNDEF AES-CCM 3:128-bit txpower 23 bmiss 7 mcastrate 6
        mgmtrate 6 scanvalid 60 wme roaming MANUAL
        parent interface: iwlwifi0
        media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11a
        status: associated
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

Из вывода **ifconfig** видно, что сетевой интерфейс **wlan0** использует родительский интерфейс **iwlwifi0**, а связь с точкой доступа Wi-Fi произведена по протоколу 802.11a на скорости **36Mbits**. И тут встает вопрос - почему скорость всего 36Mbits ? Точка доступа находится от моего ноутбука в 2х метрах. Я пробовал крутить антенны, передвигать ноутбук в другое место, «стучать по колёсам и плевать под капот» - всё безрезультатно. Ок, спишем это на «электромагнитную несовместимость».

Запустив утилиту **iperf3** я наблюдал еще более странный результат:

```
rz@butterfly:~ % iperf3 -c devbox
Connecting to host devbox, port 5201
[  5] local 192.168.168.230 port 19754 connected to 192.168.169.6 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   622 KBytes  5.09 Mbits/sec    0   41.2 KBytes
[  5]   1.00-2.01   sec   567 KBytes  4.62 Mbits/sec    0   56.9 KBytes
[  5]   2.01-3.00   sec   623 KBytes  5.13 Mbits/sec    0   69.7 KBytes
[  5]   3.00-4.00   sec   525 KBytes  4.29 Mbits/sec    0   79.7 KBytes
[  5]   4.00-5.00   sec   628 KBytes  5.15 Mbits/sec    0   89.7 KBytes
[  5]   5.00-6.00   sec   569 KBytes  4.66 Mbits/sec    0   98.3 KBytes
[  5]   6.00-7.00   sec   574 KBytes  4.71 Mbits/sec    0    107 KBytes
[  5]   7.00-8.00   sec   569 KBytes  4.66 Mbits/sec    0    114 KBytes
[  5]   8.00-9.00   sec   580 KBytes  4.75 Mbits/sec    0    120 KBytes
[  5]   9.00-10.00  sec   580 KBytes  4.75 Mbits/sec    0    127 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  5.70 MBytes  4.78 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  5.56 MBytes  4.66 Mbits/sec                  receiver

iperf Done.
```

т. е. эффективная скорость передачи по Wi-Fi сети получается 5.7 + 5.56 = **11.26 Mbit/sec**. Я провел еще несколько вариантов тестов с утилитой **iperf3** и все они показывали, что сеть работает крайне медленно и наблюдается существенная ассиметрия — «на прием» сетевой интерфейс работает в два раза быстрее, чем «на передачу», но существенно медленней чем со старым фрюшным драйвером **iwm**.

Еще один малоприятеный эффект — сетевой адаптер периодически теряет связь с точкой доступа, а в **dmesg** появляются вот такие сообщения:

```
iwlwifi0: No beacon heard and the time event is over already...
wlan0: link state changed to DOWN
wlan0: link state changed to UP
iwlwifi0: No beacon heard and the time event is over already...
wlan0: link state changed to DOWN
wlan0: link state changed to UP
iwlwifi0: No beacon heard and the time event is over already...
wlan0: link state changed to DOWN
```

Попытка проверить драйвер на устойчивость к suspend/resume с помощью кнопки «Power» показала следующее: система успешно уходит в «сон» и возвращается из него, но сетевой интерфейс становится неработоспособным, а в **dmesg** наблюдается масса сообщений от драйвера:

```
timeout waiting for hardware access (CSR_GP_CNTRL 0x080403d8)
iwlwifi0: iwlwifi transaction failed, dumping registers
iwlwifi0: iwlwifi device config registers:
iwlwifi iwlwifi: 0000   86 80 5a 09 06 04 10 00 59 00 80 02 10 00 00 00  |..Z.....Y.......|
iwlwifi iwlwifi: 0010   04 00 60 d1 00 00 00 00 00 00 00 00 00 00 00 00  |..`.............|
iwlwifi iwlwifi: 0020   00 00 00 00 00 00 00 00 00 00 00 00 86 80 00 50  |...............P|
iwlwifi iwlwifi: 0030   00 00 00 00 c8 00 00 00 00 00 00 00 ff 01 00 00  |................|
...
iwlwifi0: iwlwifi device AER capability structure:
iwlwifi iwlwifi: 0000   01 00 01 14 00 00 00 00 00 00 00 00 31 20 46 00  |............1 F.|
iwlwifi iwlwifi: 0010   00 00 00 00 00 20 00 00 00 00 00 00 00 00 00 00  |..... ..........|
iwlwifi iwlwifi: 0020   00 00 00 00 00 00 00 00 00 00 00 00              |............    |
iwlwifi0: iwlwifi parent port (iwlwifi) config registers:
iwlwifi iwlwifi: 0000   86 80 5a 09 06 04 10 00 59 00 80 02 10 00 00 00  |..Z.....Y.......|
iwlwifi iwlwifi: 0010   04 00 60 d1 00 00 00 00 00 00 00 00 00 00 00 00  |..`.............|
iwlwifi iwlwifi: 0020   00 00 00 00 00 00 00 00 00 00 00 00 86 80 00 50  |...............P|
….
iwlwifi0: Queue 4 is active on fifo 2 and stuck for 10000 ms. SW [60, 62] HW [90, 90] FH
TRB=0x05a5a5a5a
```

Попытка выгрузить и и повторно загрузить модуль драйвера приводит к появлению еще большей «портянки» в **dmesg**, но сетевой интерфейс это не реанимирует. В общем, я сделал заключение, что данная версия драйвер **iwlwifi** - не жилец.

Но у нас есть еще один вариант — перейти на FreeBSD 13-STABLE, и мы обязаны его попробовать.

16.4 Апгрейд ядра ОС FreeBSD до 13-STABLE и обратно до 13.1-RELEASE

Переход на ветку **-STABLE** или **-CURRENT** возможен только путем выкачивания полного репозитория исходных кодов ОС и их последующей локальной сборки и установки. Для проверки драйвера **iwlwifi** мне будет достаточно собрать ядро и модули, всю остальную часть системы я менять не собираюсь.

Для того, чтобы осуществить задуманное, клонируем часть репозитория исходных кодов, укажем имя ветки **stable/13** и глубину в **1** коммит - дабы сильно не пригружать удаленный сервер и Сеть выкачиванием бесполезной информации:

```
rz@butterfly:~ % git clone --depth 1 -b stable/13 https://git.freebsd.org/src.git freebsd-stable-13
```

Далее зайдем в каталог с репозиторием, осмотримся:

```
rz@butterfly:~ % cd freebsd-stable-13/

rz@butterfly:~/freebsd-stable-13 % git status
On branch stable/13
Your branch is up to date with 'origin/stable/13'.

nothing to commit, working tree clean
```

и запустим сборку ядра:

```
rz@butterfly:~/freebsd-stable-13 % make -j8 buildkernel
…
```

...пока идет сборка ядра, а процесс этот долгий, переключимся в другой терминал нажатием **Ctrl-Alt-F2** и попробуем пройти пару уровней в игре **NetHack**, периодически переключаясь в первый терминал, нажимая **Ctrl-Alt-F1**, для наблюдения за процессом компиляции…]

Окончание сборки выглядит вот так:

```
--- kernel.full ---
linking kernel.full
ctfmerge -L VERSION -g -o kernel.full ...
      text      data       bss        dec          hex   filename
  23118796   1838765   4446976   29404537   0x1c0ad79   kernel.full
--- kernel.debug ---
objcopy --only-keep-debug kernel.full kernel.debug
--- kernel ---
objcopy --strip-debug --add-gnu-debuglink=kernel.debug  kernel.full kernel
--------------------------------------------------------------
>>> Kernel build for GENERIC completed on Wed Aug 17 05:41:58 +05 2022
--------------------------------------------------------------
>>> Kernel(s)  GENERIC built in 278 seconds, ncpu: 12, make -j12
--------------------------------------------------------------
```

Инсталлируем собранное ядро и модули:

```
rz@butterfly:~/freebsd-stable-13 % sudo make installkernel
…
--------------------------------------------------------------
>>> Installing kernel GENERIC completed on Wed Aug 17 05:42:56 +05 2022
--------------------------------------------------------------
```

Командой **sudo reboot** выполним перезагрузку системы и посмотрим что у нас получилось.

Сразу после перезагрузки, войдя в систему, посмотрим на версию ядра ОС:

```
rz@butterfly:~ % uname -a
```

```
FreeBSD butterfly 13.1-STABLE FreeBSD 13.1-STABLE #0 stable/13-66ea3876d: Wed Aug 17 05:41:50
+05 2022     rz@butterfly:/usr/obj/usr/home/rz/freebsd-stable-13/amd64.amd64/sys/GENERIC
amd64
```

Вроде бы замена ядра прошла успешно, а что же с драйвером **iwlwifi** ? Напомню, что все настройки у нас остались прежние и если с драйвером всё в порядке, то сетевой интерфейс должен подняться автоматически и без проблем. Выясним, так ли это:

```
rz@butterfly:~ % ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
       ether 10:02:b5:b7:b1:e6
       inet 192.168.168.230 netmask 0xffffff00 broadcast 192.168.168.255
       groups: wlan
       ssid FABMICRO_GUEST channel 149 (5745 MHz 11a) bssid 50:ff:20:78:f8:23
       regdomain FCC country US authmode WPA2/802.11i privacy ON
       deftxkey UNDEF AES-CCM 3:128-bit txpower 23 bmiss 7 mcastrate 6
       mgmtrate 6 scanvalid 60 wme roaming MANUAL
       parent interface: iwlwifi0
       media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11a
       status: associated
       nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

Действительно, интерфейс **wlan0** поднят и сетевой адаптер присоединился к точке доступа на скорости **54Mbits**, что как бы намекает на благополучный исход, но...

Очередной тест пропускной способности с помощью **iperf3** показал всё тот же вялый результат:

```
rz@butterfly:~ % iperf3 -c devbox
Connecting to host devbox, port 5201
[  5] local 192.168.168.230 port 15978 connected to 192.168.169.6 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   871 KBytes  7.12 Mbits/sec    0    156 KBytes
[  5]   1.00-2.00   sec   609 KBytes  5.00 Mbits/sec    0    161 KBytes
[  5]   2.00-3.00   sec   518 KBytes  4.24 Mbits/sec    0    166 KBytes
[  5]   3.00-4.00   sec   600 KBytes  4.91 Mbits/sec    0    171 KBytes
[  5]   4.00-5.00   sec   585 KBytes  4.79 Mbits/sec    0    176 KBytes
[  5]   5.00-6.00   sec   553 KBytes  4.52 Mbits/sec    0    180 KBytes
[  5]   6.00-7.00   sec   601 KBytes  4.94 Mbits/sec    0    184 KBytes
[  5]   7.00-8.00   sec   574 KBytes  4.70 Mbits/sec    0    188 KBytes
[  5]   8.00-9.00   sec   568 KBytes  4.66 Mbits/sec    0    193 KBytes
[  5]   9.00-10.00  sec   576 KBytes  4.72 Mbits/sec    0    197 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  5.91 MBytes  4.96 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  5.71 MBytes  4.79 Mbits/sec                  receiver

iperf Done.
```

Хотя, по ощущениям сеть Wi-Fi работать стала стабильнее — самопроизвольно не отваливается от точки доступа и ругательств в **dmesg** не производит.

Проверим как «улучшенный» вариант драйвера относится к циклу suspend/resume нажав кнопку «Power»... Система уходит в «сон» и пробуждается нормально, но в **dmesg** присутствует та же портянка с регистрами и сообщениями об ошибках:

```
Timeout waiting for hardware access (CSR_GP_CNTRL 0x080403d8)
…
iwlwifi0: Queue 4 is active on fifo 2 and stuck for 10000 ms. SW [52, 74] HW [90, 90] FH
TRB=0x05a5a5a5a
iwlwifi0: Error sending MCAST_FILTER_CMD: enqueue_hcmd failed: -5
```

```
iwlwifi0: mcast filter cmd error. ret=-5
iwlwifi0: Error sending ECHO_CMD: enqueue_hcmd failed: -5
iwlwifi0: Failed to synchronize multicast groups update
iwlwifi0: Error sending MCAST_FILTER_CMD: enqueue_hcmd failed: -5
iwlwifi0: mcast filter cmd error. ret=-5
iwlwifi0: Error sending ECHO_CMD: enqueue_hcmd failed: -5
iwlwifi0: Failed to synchronize multicast groups update
```

Что-ж, мы с Вами на практике убедились, что драйвер **iwlwifi** пока еще не достаточно хорош и пора откатиться обратно на ядро версии **13.1-RELEASE** и вернуться к драйверу **iwm**.

Для того, чтобы вернуться на ядро ветки **-RELEASE** перейдем в каталог **/usr/src/** - тут находятся исходные коды ядра ОС устанавливаемые при инсталляции системы и поддерживаемые утилитой **freebsd-update**, из них и соберем себе ядро**:**

```
rz@butterfly:~ % cd /usr/src
rz@butterfly:/usr/src % sudo make -j12 buildkernel && sudo make installkernel
```

По окончанию сборки и установки ядра вернём настройки интерфейса **wlan0** на использование драйвера **iwm**:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf kld_list-="if_iwlwifi"
kld_list: if_iwlwifi ->

rz@butterfly:~ % sudo sysrc -f /etc/rc.conf kld_list+="if_iwm iwm7265fw"
kld_list:  ->  if_iwm iwm7265fw

rz@butterfly:~ % sudo sysrc -x -f /etc/rc.conf wlans_iwlwifi0

rz@butterfly:~ % sudo sysrc -f /etc/rc.conf wlans_iwm0="wlan0"
wlans_iwm0:  -> wlan0
```

И отправим систему на перезагрузку:

```
rz@butterfly:/usr/src % sudo reboot
```

После загрузки зайдем в систему и убедимся, что загружено ядро нужной нам ветки **-RELEASE**:

```
rz@butterfly:~ % uname -a
FreeBSD butterfly 13.1-RELEASE-p1 FreeBSD 13.1-RELEASE-p1 GENERIC amd64
```

На этом изыскания с драйверами к Wi-Fi адаптеру я хочу закончить и плавно перейти к более интересным делам — к настройке графической подсистемы.

17. Установка графической подсистемы X11

В ОС FreeBSD, следуя лучшим традициям, используется графическая подсистема **X Window System** на базе протокола **X11**. Безусловно, ведутся эксперименты с запуском Wayland (и даже с Mir), но его использование во многих смыслах не является каноническим и по этому мной не рассматривается принципиально.

Прежде чем мы перейдем непосредственно к установке требуемых пакетов, я хотел бы немного рассказать о том, что такое X Window System и почем это есть хорошо и правильно, а всё остальное — от лукавого.

Графический интерфейс пользователя в UNIX системах появился не сразу, где-то спустя полтора десятка лет, а именно в 1984 году в стенах MIT был запущен проект под названием «X Window System» или просто «X» (в простонародье «Иксы»). До этого конечно же были и другие потуги в создании GUI для UNIX систем, но все они не оказались столь популярными по причине своей мелкотравчатости, т. е. не глубокой концептуальной продуманности. А еще, по причине того, что Иксы с нулевого дня были с открытым исходным кодом и к проекту могли подключаться другие разработчики по мери его развития.

X Window System предполагает концепцию типа «клиент-сервер» в составе которой имеются следующие сущности:

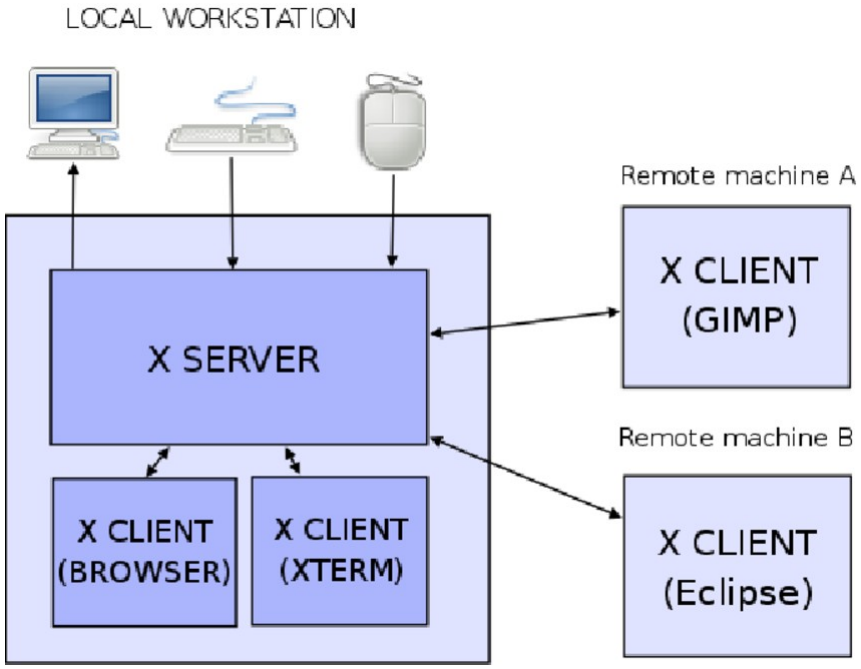

Рис. 1. Упрощенная архитектура X Window System.

1) **X сервер (X Server) —** устройство ввода и отображения информации. Является отдельной, независимой от основной (хост) системы сущностью. X сервер в рамках X Window System может быть как отдельно стоящим аппаратным устройством (в конце 80^{-x} и 90^{-x} так оно и было), так и просто процессом на хост системе.

2) **Устройство (Device)** — физическая сущность которой располагает и оперирует X сервер: графический дисплей, графический акселератор, клавиатура, мышь, манипуляторы «joystick», и т. д.

3) **Ресурс (Resource)** — блок информации который можно загрузить на X сервер или попросить X сервер загрузить самостоятельно из файла. К ресурсам относятся графические файлы (битмапы), различные шрифты (fonts), файлы данных и т. д.

4) **X клиент (X Client)** — пользовательский процесс, который функционирует на хост системе и с помощью специального сетевого протокола может иметь доступ к X серверам, отправлять в них запросы на манипуляцию ресурсами и устройствами, т. е. отображать графическую информацию на один и более X серверов, получать от X серверов ввод пользователя (нажатия клавиш клавиатуры, перемещение мыши, отклонение манипуляторов типа «joystick» и т. д.).

5) **X протокол (X protocol)** — сетевой протокол, позволяет вести обмен данными между X клиентами и X серверами. X протокол позволяет одному клиенту общаться сразу с несколькими X серверами (несколькими физическими дисплеями), а одному X серверу — обслуживать запросы сразу от нескольких X клиентов независимо от того где они физически располагаются. В качестве транспорта в X протоколе используется либо TCP для разнесенных клиента и сервера, либо UNIX-сокеты - в случае, когда клиент и сервер исполняются на одном хосте. Разумеется, что в X протоколе предусматривается протокол авторизации (XAuth) позволяющий серверам регламентировать доступ клиентов к своим устройствам и ресурсам. Стандартный протокол авторизации XAuth достаточно примитивен и поэтому считается небезопасным, но существует воз и маленькая тележка расширений под общим названием X Access Control Extension (XACE). Более того, всеми любимый протокол SSH имеет «от рождения» возможность пробрасывать (туннелировать) X сеанс через себя и эту его фичу часто используют для доступа к вычислительным системам в крупных корпоративных сетях. Здесь вся авторизация ложится на плечи SSH.

Для полноты следует еще упомянуть про существование такой сущности как X Font Server (сервер шрифтов или **xfs**), но в этой статье мне не хочется так глубоко погружаться в X11. К тому же, подавляющее большинство приложений сейчас рендерят шрифты на стороне клиента.

Первая спецификация X Window System вышедшая в июне 1984 года имела протокол версии X1, далее протокол быстро развивался и к 1987 году достиг версии **X11**. Так номер версии с тех пор и закрепился в его названии, а настоящий номер версия (номер релиза) добавляется через букву **R**. На данный момент самая последняя версия протокола X11 имеет номер **X11R7.7** и была она выпущена в июне 2012 года.

Среди обывателей бытует мнение, что протокол X11 не развивается, давно устарел и его нужно списать на свалку истории, но я считаю что это не так! Графическая подсистема X Window System, выстроенная за несколько десятков лет на его основе, очень мощная и с очень объемной экосистемой, хоть местами и с изъянами. Любые попытки «переосмыслить» и изобрести что-то новое и более идеологически правильное, в конечном счете напарываются на все те же сложности, и в результате становятся еще более убогими и конкретно недопиленными (привет Wayland).

Статья и дискуссия: https://ajaxnwnk.blogspot.com/2020/10/on-abandoning-x-server.html

Еще дискуссия: https://news.ycombinator.com/item?id=24921806

Таким образом, для того чтобы в нашей системе появился графический интерфейс нам необходимо установить следующее:

- Набор библиотек реализующих протокол X11. На данный момент широко используются **Xlib** и **XCB**.
- Одну из немногих реализаций X сервера. На данный момент это **Xorg**. Подробнее о нём чуть ниже.
- Менеджер окон (Window Manager - WM) - X клиент реализующий одну из множества оконных систем.
- Менеджер дисплеев (Display Manager - DM) — графическая утилита (тоже X клиент), которая обеспечивает авторизацию пользователя у конкретного X сервера и организует сеанс. Чтобы было более понятно, DM это то первое графическое приложение с которым взаимодействует пользователь при входе (логине) в систему.

Пара замечаний про свободный X сервер.

С 1991 по 2004 год на платформе x86 широко использовался X сервер от проекта XFree86, но в феврале 2004 у проекта возникли проблемы с Free Software Foundation из-за кусков кода закрытых лицензией GPL. В результате был сделан «fork» под названием X.org и создана новая некоммерческая организация разрабатывающая и поддерживающая эту реализацию X сервера и по сей день. X.org так же ведет разработку и поддерживает набор библиотек Xlib и XCB (X for C bindings) и на данный момент является стандартом «де-факто», причем не только на x86.

X сервер Xorg состоит из двух частей: аппаратно независимой (Device Independent X) и части зависимой от конкретной видео карты или видео ускорителя (Device Dependent X). В свою очередь DDX часть требует наличия в системе драйвера для видео карты реализующего Direct Rendering Infrastructure (DRI). DRI был введен в ОС Linux для быстрого доступа пользовательским процессам к аппаратным ускорителям и, в частности для OpenGL. Из Linux-а DRI перекочевал во многие другие свободные ОС, в том числе и во FreeBSD. Получается, что перед тем как устанавливать Xorg сервер нам нужно озаботиться видео драйверами. Напомню, что в моём новом ноутбуке две видео карты: встроенная от AMD и дискретная от NVIDIA (и хочу заметить, что тут кроется серьезный подвох, но пока не буду раскрывать какой).

Тут так же следует сказать, что в пакете Xorg имеется «универсальный» видео драйвер позволяющей использовать видео карту без режимов аппаратного ускорения с помощью стандарта VESA — его поддерживают подавляющее большинство видеокарт. Этот драйвер (или скорее режим работы) называется **scfb**. Дабы заранее не усложнять себе жизнь с поиском и установкой правильных видео драйверов, я решил для начала настроить Xorg в этом режиме и получить хоть какой-то визуальный эффект радующий глаз, но об этом чуть далее.

Запустим установку требуемых нам пакетов. Сначала установим библиотеки:

```
rz@butterfly:~ % sudo pkg inst libX11 libxcb qt5
```

Указанные в данной команде имена пакетов на самом деле являются мета-пакетами которые содержат огромное количество зависимостей. Утилита **pkg** проследит, пересчитает и предложит пользователю установить их все и здесь важно, чтобы в файловой системе оказалось достаточно свободного места.

Следующим этапом установим Xorg сервер:

```
rz@butterfly:~ % sudo pkg inst --yes xorg
```

Xorg вытянет за собой оставшуюся половину интернета.

Разумеется все пакеты можно установить одной командой, последовательно перечислив их имена через пробел. Параметр **--yes** проинструктирует утилиту **pkg** не задавать лишних вопросов и со всем соглашаться.

Процесс установки этих пакетов может занять некоторое время, поэтому заварим чашку чая и пойдем в другой виртуальный видео терминал играть в NetHack — вдруг повезет раздобыть Амулет Йендора.

Чтобы переключаться между виртуальными видео терминалами (VTY) необходимо нажать **Ctrl-Alt-F1** или F2, или F3 и так до F8 — всего по-умолчанию включено восемь виртуальных видео терминалов.

18. Настройка Xorg в режиме scfb

Теоретически, сразу после окончания установки Xorg можно попытаться запустить X сервер без всякой предварительной настройки - Xorg достаточно интеллектуален для того, чтобы самостоятельно определить тип видео адаптера, мыши и клавиатуры и подгрузить соответствующие аппаратно зависимые модули. Но, к сожалению, на моём ноутбуке Lenovo Ideapad 3 Gaming оказалось предостаточно аппаратных нововведений с которыми Xorg самостоятельно справится не может.

Тем не менее, посмотрим как это выглядит. Выполним команду **X** и посмотрим на результат:

```
rz@butterfly:~ % X
```

В моём случае Xorg сразу завершился с сообщением об ошибке. Заглянув в файл **/var/log/Xorg.0.log** я обнаружил следующие сообщения:

```
X.Org X Server 1.20.14
X Protocol Version 11, Revision 0
[    41.505] Build Operating System: FreeBSD 13.1-RELEASE amd64
[    41.505] Current Operating System: FreeBSD butterfly 13.1-RELEASE FreeBSD 13.1-RELEASE
releng/13.1-n250148-fc952ac2212 GENERIC amd64
[    41.505] Build Date: 19 June 2022  12:55:14AM
[    41.505]
[    41.505] Current version of pixman: 0.40.0
[    41.505]   Before reporting problems, check http://wiki.x.org
       to make sure that you have the latest version.
[    41.505] Markers: (--) probed, (**) from config file, (==) default setting,
       (++) from command line, (!!) notice, (II) informational,
       (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
[    41.506] (==) Log file: "/var/log/Xorg.0.log", Time: Wed Jul 13 04:29:00 2022
[    41.506] (==) Using system config directory "/usr/local/share/X11/xorg.conf.d"
[    41.506] (==) No Layout section.  Using the first Screen section.
[    41.506] (==) No screen section available. Using defaults.
[    41.506] (**) |-->Screen "Default Screen Section" (0)
[    41.506] (**) |   |-->Monitor "<default monitor>"
[    41.506] (==) No monitor specified for screen "Default Screen Section".
       Using a default monitor configuration.
[    41.506] (==) Automatically adding devices
```

```
[    41.506] (==) Automatically enabling devices
[    41.506] (==) Not automatically adding GPU devices
[    41.506] (==) Max clients allowed: 256, resource mask: 0x1fffff
[    41.507] (==) FontPath set to:
       /usr/local/share/fonts/misc/,
       /usr/local/share/fonts/TTF/,
       /usr/local/share/fonts/OTF/,
       /usr/local/share/fonts/Type1/,
       /usr/local/share/fonts/100dpi/,
       /usr/local/share/fonts/75dpi/,
       catalogue:/usr/local/etc/X11/fontpath.d
[    41.507] (==) ModulePath set to "/usr/local/lib/xorg/modules"
[    41.507] (II) The server relies on udev to provide the list of input devices.
       If no devices become available, reconfigure udev or disable AutoAddDevices.
[    41.507] (II) Loader magic: 0x433270
[    41.507] (II) Module ABI versions:
[    41.507]   X.Org ANSI C Emulation: 0.4
[    41.507]   X.Org Video Driver: 24.1
[    41.507]   X.Org XInput driver : 24.1
[    41.507]   X.Org Server Extension : 10.0
[    41.507] (--) PCI: (1@0:0:0) 10de:25a2:17aa:3a5d rev 161, Mem @ 0xd0000000/16777216,
0xfb00000000/4294967296, 0xfc00000000/33554432, I/O @ 0x00003000/128
[    41.507] (--) PCI:*(5@0:0:0) 1002:1638:17aa:3a5d rev 198, Mem @ 0xfc10000000/268435456,
0xfc20000000/2097152, 0xd1400000/524288, I/O @ 0x00001000/256, BIOS @ 0x????????/65536
...
[    41.517] (EE) open /dev/dri/card0: No such file or directory
[    41.517] (WW) Falling back to old probe method for modesetting
[    41.517] (EE) open /dev/dri/card0: No such file or directory
[    41.517] (WW) Falling back to old probe method for scfb
[    41.517] scfb trace: probe start
[    41.517] (II) scfb(1): using default device
[    41.517] scfb trace: probe done
[    41.517] (WW) VGA arbiter: cannot open kernel arbiter, no multi-card support
[    41.517] (EE) Screen 0 deleted because of no matching config section.
[    41.517] (II) UnloadModule: "modesetting"
[    41.517] (EE) Fatal server error:
[    41.517] (EE) Cannot run in framebuffer mode. Please specify busIDs       for all
framebuffer devices
[    41.517] (EE)
[    41.517] (EE)
Please consult the The X.Org Foundation support
         at http://wiki.x.org
 for help.
[    41.518] (EE) Please also check the log file at "/var/log/Xorg.0.log" for additional
information.
[    41.518] (EE)
[    41.523] (EE) Server terminated with error (1). Closing log file.
```

Из лога следует, что Xorg попытался что-то предпринять: перебрал несколько вариантов видео драйверов, обнаружил целых два видео адаптера, но не смог определить их принадлежность к какому-то видео драйверу, а самих аппаратных драйверов не нашел (отсутствуют файлы устройств **/dev/dri/cardX**). Xorg попытался использовать драйвер **scfb**, но с ним тоже не срослось из-за того, что в системе два видео адаптера и драйвер **scfb** просит указать **BusID** того адаптера, который будет использован для переключения видеорежима (процедура переключения называется «**modesetting**»).

Раз так, попытаемся указать Xorg-у, что для переключения видеорежима следует использовать встроенную видео карту от AMD. Для этого создадим файл **/etc/X11/xorg.conf** следующего содержания:

```
Section "Device"
       Identifier "vga0"
       Driver "scfb"
       BusID "PCI:5:0:0"
EndSection
```

Здесь мы сообщаем Xorg-у, что следует использовать видео драйвер **scfb** с видео адаптером который находится на шине **PCI** по адресу **5:0:0 -** этот адрес подсказал нам сам Xorg в логе и он совпадает с тем, что выдает утилита **pciconf**:

```
rz@butterfly:~ % sudo pciconf -levc | grep -A 7 vga
vgapci0@pci0:1:0:0:   class=0x030200 rev=0xa1 hdr=0x00 vendor=0x10de device=0x25a2
subvendor=0x17aa subdevice=0x3a5d
    vendor     = 'NVIDIA Corporation'
    device     = 'GA107M [GeForce RTX 3050 Mobile]'
    class      = display
    subclass   = 3D
    cap 01[60] = powerspec 3  supports D0 D3  current D0
    cap 05[68] = MSI supports 1 message, 64 bit enabled with 1 message
    cap 10[78] = PCI-Express 2 endpoint max data 256(256) FLR RO NS
--
vgapci1@pci0:5:0:0:                   class=0x030000 rev=0xc6 hdr=0x00 vendor=0x1002
device=0x1638 subvendor=0x17aa subdevice=0x3a5d
    vendor     = 'Advanced Micro Devices, Inc. [AMD/ATI]'
    device     = 'Cezanne'
    class      = display
    subclass   = VGA
    cap 09[48] = vendor (length 8)
    cap 01[50] = powerspec 3  supports D0 D3  current D0
    cap 10[64] = PCI-Express 2 legacy endpoint max data 128(256) RO NS
```

Запустим еще раз команду **X** и обнаружим абсолютно черный экран без какой либо реакции на внешние раздражители, посылка сигнала SIGINT нажатием комбинации клавиш **Ctrl-C** ни к чему не приводит. Попав в такую ситуацию новичку может показаться, что система полностью зависла, однако это не так. Всё работает так, как мы этого и просили: Xorg запустился, включил видеорежим по-умолчанию и ждет запросов от X клиента, при этом клавиатура (и прочие устройства ввода) находится в монопольном владении X сервера.

Для выхода из сложившейся ситуации нажмем комбинацию клавиш **Ctrl-Alt-F1**, которую операционная система (точнее драйвер видеотерминала — VTY) обработает как переключение виртуального видео терминала. Это нажатие приведет к переключению видеорежима обратно в текстовый и вернет нас в нулевой терминал (**ttyv0**) из которого мы запустили X сервер). Далее привычным нажатием **Ctrl-C** остановим исполнение X сервера.

Замечание: запущенный процесс X сервера Xorg добавляет в систему еще один, девятый, видео терминал — графический, с идентификатором **ttyv8**. Переключиться в него можно обратно из текстового нажатием **Alt-F9** или (**Ctrl-Alt-F9**).

Чтобы продолжить эксперименты с «Иксами» нам теперь требуется какой-то X клиент, который мы должны заставить общаться с нашим X сервером. В качестве такого X клиента может быть, и часто выступает, утилита **xterm** — графический эмулятор телетайпа (терминала). Установим её из пакетов командой:

```
rz@butterfly:~ % sudo pkg inst --yes xterm
```

и посмотрим, что написано про **xterm** в системной документации:

```
rz@butterfly:~ % man xterm
XTERM(1)                        X Window System                        XTERM(1)



NAME
```

```
       xterm - terminal emulator for X

SYNOPSIS
       xterm [-toolkitoption ...] [-option ...] [shell]

DESCRIPTION
       The xterm program is a terminal emulator for the X Window System.  It
       provides DEC VT102/VT220 and selected features from higher-level
       terminals such as VT320/VT420/VT520 (VTxxx).  It also provides
       Tektronix 4014 emulation for programs that cannot use the window system
       directly.  If the underlying operating system supports terminal
       resizing capabilities (for example, the SIGWINCH signal in systems
       derived from 4.3BSD), xterm will use the facilities to notify programs
       running in the window whenever it is resized.
```

Теперь встает вопрос, как запустить одновременно X сервер и X клиент (xterm) и как клиенту объяснить где находится сервер. Ответ на вторую часть этого вопроса прост: переменная окружения **$DISPLAY** указывает X клиенту адрес (IP или host name) и идентификатор сеанса. По-умолчанию можно использовать значение **:0.0** — это говорит X клиенту о том, что сервер находится на этой же машине и к нему можно подключиться на нулевой дисплей и нулевой экран. Полный формат этой переменной окружения и назначение полей можно почерпнуть из системной документации по команде **man X** в главе **DISPLAY NAMES**.

Чтобы сильно не заморачиваться с написанием своих скриптов запуска X сервера и требуемых клиентов, в системе, после установки пакета **Xorg**, уже имеется специальный скрипт для организации пользовательского сеанса, вызывается он командой **startx**. Ниже цитата из системной документации по этой команде:

```
rz@butterfly:~ % man startx
STARTX(1)               FreeBSD General Commands Manual              STARTX(1)



NAME
       startx - initialize an X session

SYNOPSIS
       startx [ [ client ] options ... ] [ -- [ server ] [ display ] options
       ... ]

DESCRIPTION
       The startx script is a front end to xinit(1) that provides a somewhat
       nicer user interface for running a single session of the X Window
       System.  It is often run with no arguments.
```

Выполним команду **startx** и… обнаружим много интересного, а именно:

1) запустился X сервер и переключился в графический режим;
2) на черном экране видно несколько окон с терминалом — это утилита **xterm**;
3) в верхнем правом углу видны аналоговые часы — это утилита **xclock**;
4) ввод с клавиатуры отрабатывает нормально, а вот мышь (в моём случае TouchPad) не работает — в центре экрана висит «примерзший» указатель в виде символа X.

По нажатию комбинации **Ctrl-Alt-F1** вернемся в текстовый терминал и завершим сеанс нажатием **Ctrl-C** остановив исполнение команды **startx**.

На данном этапе мы имеем рабочую графическую подсистему X11 с одним неприятным нюансом — не работает TouchPad. Решением этой проблемы займемся завтра, а сейчас пора спать!

19. Заставляем работать TouchPad

[прошло пару дней]

В подавляющем большинстве ноутбуков устанавливаются тачпады производства известной калифорнийской фирмы Synaptics и такие тачпады работают с подавляющим большинством операционных систем, в том числе и с ОС FreeBSD, что называется «из коробки». Тачпады Synaptics имеют популярный интерфейс PS/2, что по сути является последовательным портом, к ним написана масса драйверов, а сами драйвера весьма примитивны в реализации. Но на моём Lenovo Ideapad 3 Gaming установлен совершенно другой тачпад.

Перед тем, как решать проблему с драйверами к тачпаду, необходимо выяснить «что за зверь перед нами». Данный тачпад никак не обнаруживается системой: ни на шине PCI, ни на шине USB его не видно. Гугление тоже не дало какого либо вразумительного ответа о модели тачпада, кроме имени производителя — ELAN. Честно говоря, я ранее и не слышал о таком производителей. Гугление по имени производителя и по имени ноутбука дало противоречивые результаты — у «Елань», как выяснилось, имеется много различных тачпадов с разными видами интерфейсов, в том числе и с PS/2. В общем, печаль — не ставить же мне сейчас на этот ноутбук «винду», чтобы выяснить модель тачпада. И тут пришла одна мысль.

Где-то в глубине сайта Lenovo мне удалось раздобыть пакет с драйверами озаглавленный как *ELAN Touchpad Driver*, выкачанный файл имел названием **2tlf0faf40.exe** — самораспаковывающийся архив и инсталлятор в одном флаконе для ОС Windows. Сначала я попробовал распаковать его утилитой **7z** — ресурсы из EXE файла изъялись, но большой блоб данных в получившемся файле оказался в неизвестном мне сжатом формате. Тогда я прибег к простой хитрости — запустил **2tlf0faf40.exe** в среде Wine32 на другой машине, дождался пока отобразится окошко с подтверждением инсталлятора, запустил «инсталляцию» и остановил процесс командой **kill -9**. В результате, в каталоге *~/.wine32/drive_c/DRIVERS* обнаружился подкаталог *./Elan\ Touchpad\ Driver*, а в нём и сам драйвер со всеми сопутствующими .INF файлами.

Немного покопавшись в добытых .INF файлах я понял, что в моём ноутбуке Lenovo установлено устройство сенсорного ввода типа HIDI2C (HID over I2C) — стандарт придуманный в стенах небезызвестной корпорации Microsoft. Команда **pciconf** показала, что в ноутбуке действительно имеется I2C (SMBus) контроллер:

```
rz@butterfly:~ % sudo pciconf -levb | grep -A 5 -i 'smb\|i2c'
intsmb0@pci0:0:20:0:  class=0x0c0500 rev=0x51 hdr=0x00 vendor=0x1022 device=0x790b
subvendor=0x17aa subdevice=0x384c
    vendor     = 'Advanced Micro Devices, Inc. [AMD]'
    device     = 'FCH SMBus Controller'
    class      = serial bus
    subclass   = SMBus
```

Этой информации оказалось более чем достаточно. По команде **apropos i2c** я быстро выяснил, что в ОС FreeBSD имеется поддержка шины I2C/SMBus, а так же HID устройств подключенных к этой шине. Из чтения предложенных man-страниц выяснилось какие

именно драйвера требуется подключить, это: **ig4**, **iicbus** и **iichid**. Для проверки теории я вручную подгрузил соответствующие модули командой:

```
rz@butterfly:~ % sudo kldload ig4 iicbus iichid
```

и тут же увидел в **dmesg** следуюшие сообщения:

```
ig4iic0: <Designware I2C Controller> iomem 0xfedc2000-0xfedc2fff irq 10 on acpi0
iicbus0: <Philips I2C bus (ACPI-hinted)> on ig4iic0
iicbus0: <unknown card> at addr 0x2c
iichid0: <MSFT0001:01 06CB:CE78 I2C HID device> at addr 0x2c on iicbus0
iichid0: Interrupt setup failed. Fallback to sampling
hidbus0: <HID bus> on iichid0
```

Видно, что HID устройство обнаружено и в системе появилось новое устройство ввода:

```
rz@butterfly:~ % evtest /dev/input/event9

Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x6cb product 0xce78 version 0x103
Input device name: "MSFT0001:01 06CB:CE78 TouchPad"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 272 (BTN_LEFT)
    Event code 325 (BTN_TOOL_FINGER)
    Event code 328 (BTN_TOOL_QUINTTAP)
    Event code 330 (BTN_TOUCH)
    Event code 333 (BTN_TOOL_DOUBLETAP)
    Event code 334 (BTN_TOOL_TRIPLETAP)
    Event code 335 (BTN_TOOL_QUADTAP)
  Event type 3 (EV_ABS)
    Event code 0 (ABS_X)
      Value    683
      Min        0
      Max     1404
      Resolution      12
    Event code 1 (ABS_Y)
      Value    208
      Min        0
      Max      864
      Resolution      12
    Event code 47 (ABS_MT_SLOT)
      Value      1
      Min        0
      Max        4
    Event code 53 (ABS_MT_POSITION_X)
      Value      0
      Min        0
      Max     1404
      Resolution      12
    Event code 54 (ABS_MT_POSITION_Y)
      Value      0
      Min        0
      Max      864
      Resolution      12
    Event code 57 (ABS_MT_TRACKING_ID)
      Value      0
      Min       -1
      Max    65535
Properties:
```

```
        Property type 0 (INPUT_PROP_POINTER)
```

Ура! Осталось объяснить X серверу, что у нас есть устройство ввода типа тачпад, для этого добавим в файл конфигурации X сервера **/etc/X11/xorg.conf** следующие секции:

```
Section "InputDevice"
    Identifier     "Touchpa0"
    Driver         "evdev"
EndSection

Section "InputClass"
        Identifier "libinput touchpad catchall"
        MatchIsTouchpad "on"
        MatchDevicePath "/dev/input/event*"
        Driver "libinput"
        Option "Tapping" "on"
        Option "MiddleEmulation" "on"
        Option "AccelProfile" "adaptive"
EndSection
```

Запустим «Иксы» командой **startx** и убедимся, что X сервер обнаружил устройство ввода указательного типа, а сам указатель перемещается и графическая подсистема реагирует на нажатия кнопок тачпада и жесты прокрутки.

Единственное, что пока смущает, так это сообщение от драйвера **iichid**:

```
iichid0: Interrupt setup failed. Fallback to sampling
```

Это сообщение говорит нам о том, что драйвер не знает на какой линии GPIO находится линия прерывания от микросхемы HIDI2C интерфейса, поэтому, чтобы получать данные с микросхемы драйвер будет её непрерывно опрашивать («сэмплировать»). Это не есть хорошо, так как будет создавать лишнюю нагрузку на систему, но пока поживем в таком режиме.

Чтобы нужные нам драйверы загружались автоматически, запишем в файл **/etc/rc.conf** в переменную **kld_list** их имена. Сделаем это традиционно утилитой **sysrc**:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf kld_list+="ig4 iicbus iichid"
```

20. Установка WM и DM

Как я уже упоминал выше, для того, чтобы в графической подсистеме появились красивые окошки, кнопки и прочие элементы современного GUI, нам необходимо установить какой либо из доступных Window Manager-ов (WM). Мой выбор в данном случае упал на **Xfce** потому, что он легкий и быстрый, модульный, очень гибко настраиваемый и при этом весьма неплохо выглядит. Все настройки **Xfce** можно выполнить через графический интерфейс, но при этом сохраняются они в текстовые файлы, которые можно подправить вручную. Каждый модуль ведет свой конфигурационный файл. В общем, Xfce соответствует всем канонам. А еще, Xfce является клоном старого доброго Common Desktop Environment (CDE) который мне довелось увидеть на 386$^{-й}$ машине под управлением SCO UNIX в далеком 1993 году. Вот как это выглядело в 90$^{-х}$:

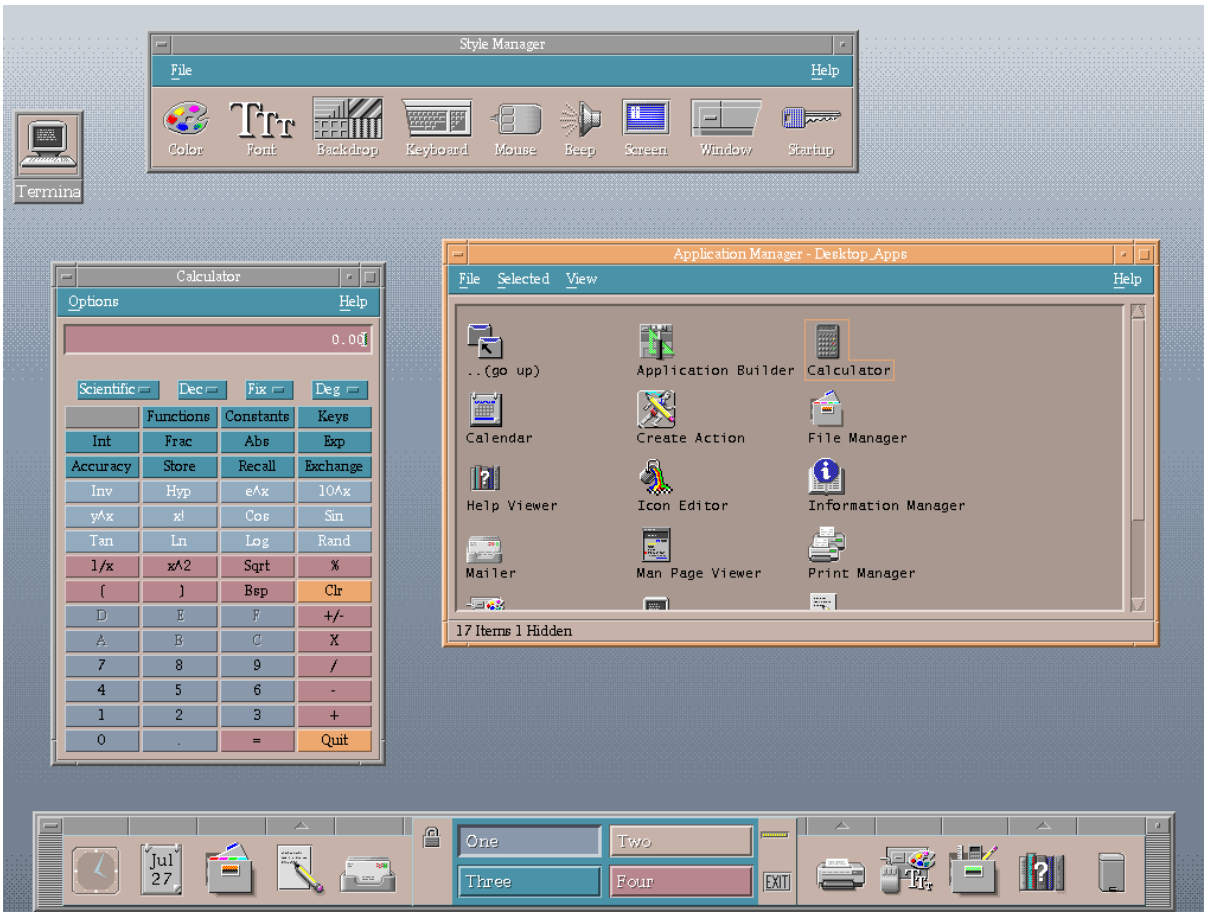

Рис. 2. Common Desktop Environment образца конца 90^{-x}.

Просто прелесть, правда ?

Установим оконную оболочку **Xfce версии 4**:

```
rz@butterfly:~ % sudo pkg search xfce-4
xfce-4.16                      Meta-port for the Xfce Desktop Environment

rz@butterfly:~ % sudo pkg inst --yes xfce-4
```

Помимо Window Manager-а нам еще не плохо бы установить Display Manager — приложение, которое проводит авторизацию пользователя, создает сеанс и запускает набор X клиентов для организации графического окружения (собственно, запускает Xfce). В качестве DM я выбрал для себя Simple Display Desktop Manager (SDDM) без особых на то причин.

Установим **SDDM**:

```
rz@butterfly:~ % sudo pkg search sddm-0
sddm-0.19.0_7                  QML based login manager

rz@butterfly:~ % sudo pkg inst --yes sddm-0
```

Для того, чтобы SDDM стартовал сразу после загрузки операционной системы необходимо добавить в системный конфигурационный файл **/etc/rc.conf** следующую строку: **sddm_enable="YES"**. Сделать это можно всё тем же способом:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf sddm_enable="YES"
Password:
sddm_enable:  -> YES
```

Ну и последний штрих, запустим SDDM не перезагружая систему следующей командой:

```
rz@butterfly:~ % sudo service sddm start
```

Сразу по исполнению этой команды мы увидим как запустится X сервер, переключит видео карту в графический режим, к нему подключится SDDM и отобразит нам окно с предложением пройти авторизацию. Перед тем как вводить login и пароль, выберем тип сеанса **Session Xfce** в верхнем левом углу экрана в выпадающем списке **Session**.

Очевидно, что SDDM позволяет создавать сеанс используя любой из установленных в системе WM. Вы можете самостоятельно установить KDE5 или FVWM и переключаться между ними каждый раз выбирая желаемый тип сеанса. К тому же, в репозитории FreeBSD уже имеются соответствующие пакеты:

```
rz@butterfly:~ % pkg search kde5
kde5-5.24.6.22.04.3            KDE Plasma Desktop and Applications (current)

rz@butterfly:~ % pkg search fvwm
fvwm-2.6.9_2                   Popular virtual window manager for X
```

21. Настройка раскладки клавиатуры в Xfce4

Как я уже отмечал, почти все настройки **Xfce** можно выполнить просто клацая мышью по экрану, но тут ключевое слово «почти». Как выяснилось, утилита для конфигурирования раскладок клавиатуры **xfce4-keyboard-settings** не позволяет указать дополнительные параметры к раскладке, в частности желанный многими пользователями параметр **numpad:microsoft** который делает поведение клавиатуры более привычным на ноутбуках с клавиатурой с цифровым блоком (мой Lenovo как раз тот самый случай). Поэтому просто создадим файл **keyboard-layout.xml** следующего содержания:

```
rz@butterfly:~ % cat ~/.config/xfce4/xfconf/xfce-perchannel-xml/keyboard-layout.xml
<?xml version="1.0" encoding="UTF-8"?>

<channel name="keyboard-layout" version="1.0">
  <property name="Default" type="empty">
    <property name="XkbDisable" type="bool" value="false"/>
    <property name="XkbModel" type="string" value="asus_laptop"/>
    <property name="XkbOptions" type="empty">
      <property name="Group" type="string" value="grp:caps_toggle,numpad:microsoft"/>
    </property>
    <property name="XkbLayout" type="string" value="us,ru"/>
    <property name="XkbVariant" type="string" value=",legacy"/>
  </property>
</channel>
```

В этом конфигурационном файле мы устанавливаем модель клавиатуры (**asus_laptop**), добавляем пару раскладок (**US, RU**), устанавливаем клавишу переключения между ними (клавиша **CapsLock** в режиме «toggle») и задаем поведение цифрового блока клавиатуры «как на винде» (параметр **numpad:microsoft**).

Чтобы данные изменения вступили в силу необходимо завершить сеанс и войти снова (Logout/Login). И тут сразу сделаю замечание: если Вы попытаетесь отредактировать настройки клавиатуры запустив стандартное средство **xfce4-keyboard-settings** (**Applications → Settings → Keyboard**), то после сохранения новых настроек, все неподдерживаемые опции и параметры будут удалены, в том числе параметр **numpad:microsoft**, поэтому не лишним будет отнять у этого файла атрибут разрешающий запись:

```
rz@butterfly:~ % ll ~/.config/xfce4/xfconf/xfce-perchannel-xml/keyboard-layout.xml
-rw-r--r--  1 rz  rz  552 Jul 26 02:19 /home/rz/.config/xfce4/xfconf/xfce-perchannel-xml/keyboard-layout.xml

rz@butterfly:~ % chmod -w ~/.config/xfce4/xfconf/xfce-perchannel-xml/keyboard-layout.xml

rz@butterfly:~ % ll ~/.config/xfce4/xfconf/xfce-perchannel-xml/keyboard-layout.xml
-r--r--r--  1 rz  rz  552 Jul 26 02:19 /home/rz/.config/xfce4/xfconf/xfce-perchannel-xml/keyboard-layout.xml
```

22. Проблема потери настроек клавиатуры после suspend/resume

Поигравшись немного с другими настройками Xfce я решил проверить как «Иксы» относятся к процедуре «suspend/resume». Выполнив команду **sudo zzz** я дождался когда система уйдет в режим «сна» и тут же разбудил её. На удивление, процесс пробуждения всей системы занял всего 2-3 секунды и я почти мгновенно получил «десктоп» в том же состоянии, в котором он был до «засыпания».

Однако, покликав мышью в разные меню и понажимав кнопки я тут же обнаружил, что клавиша CapsLock больше не переключает раскладки между RU и US, а её функция вернулась к исходной. Догадываясь в чем дело, я решил сначала восстановить требуемые мне настройки из командной строки используя утилиту **setxkbmap**:

```
rz@butterfly:~ % setxkbmap -layout us,ru -option grp:caps_toggle,numpad:microsoft
```

а затем проверить какие настройки «Иксы» сохраняют для устройства ввода до и после «сна»:

```
# До сна:
rz@butterfly:~ % setxkbmap -print
xkb_keymap {
        xkb_keycodes  { include "evdev+aliases(qwerty)"     };
        xkb_types     { include "complete+numpad(microsoft)"       };
        xkb_compat    { include "complete"   };
        xkb_symbols   { include "pc+us+ru(legacy):2+inet(evdev)+capslock(grouplock)"     };
        xkb_geometry  { include "pc(pc104)"  };
};

# Засыпаем:
rz@butterfly:~ % sudo zzz

# После сна:
rz@butterfly:~ % setxkbmap -print
xkb_keymap {
        xkb_keycodes  { include "evdev+aliases(qwerty)"     };
        xkb_types     { include "complete"   };
        xkb_compat    { include "complete"   };
```

```
        xkb_symbols   { include "pc+us+inet(evdev)" };
        xkb_geometry  { include "pc(pc105)"  };
};
```

Видно, что все настройки сбросились в состояние по-умолчанию. Не долго разбираясь с чем связано такое странное поведение системы X11 на ОС FreeBSD, я имплементировал гарантированное решение — вставил требуемые мне настройки клавиатуры прямо в конфигурационный файл X сервера **/etc/X11/xorg.conf** так, что секция **InputClass** теперь выглядит следующим образом:

```
Section "InputClass"
        Identifier "keyboard defaults"
        MatchIsKeyboard "on"
        Option "XkbLayout" "us,ru"
        Option "XKbOptions" "terminate:ctrl_alt_bksp,numpad:microsoft,grp:caps_toggle"
EndSection
```

Фактически здесь всё то же самое, что и выполняет утилита **xfce4-keyboard-settings**, но применяются эти настройки в момент процедуры инициализации X сервера, в том числе и при «пробуждении» системы.

В дополнении к уже имеющимся настройкам клавиатуры я добавил еще одну опцию **terminate:ctrl_alt_bksp** которая позволяет аварийно завершить выполнение X сервера по нажатию комбинации клавиш **Ctrl-Alt-Backspace** и вернуться в текстовый терминал или в Display Manager в случае зависания X сервера или блокировки ввода. Эта древняя и очень полезная фича всегда была включена в X серверах, но в **Xorg** её по-умолчанию отключили — непорядок!

После внесения каких либо изменений в файл **/etc/X11/xorg.conf**, графическую подсистему требуется перезапустить. Сделать это можно следующей командой предварительно переключившись в текстовый терминал с помощью **Ctrl-Alt-F1**:

```
rz@butterfly:~ % sudo service sddm restart
```

Замечание:

Строго говоря, если настройки клавиатуры заданы в системном конфигурационном файле, то локальный файл **~/.config/xfce4/xfconf/xfce-perchannel-xml/keyboard-layout.xml** можно смело удалить, но делать я этого не буду из соображений порядка: вероятно, что проблема с пропаданием настроек клавиатуры в Xorg будет исправлена в ближайшее время, и тогда можно будет вернуться к прежнему, конвенциальному варианту.

23. Запуск 3D акселераторов

Очевидно, что на одной растровой графике в современном мире прожить невозможно, почти все САПР требуют аппаратной поддержки OpenGL и EGL — прямого доступа к аппаратному ускорителю 3D графики, чего драйвер **scfb** обеспечить не в состоянии. А значит настала пора озаботиться установкой драйверов для двух видео адаптеров. Напомню, что в ноутбуке Lenovo Ideapad 3 Gaming присутствует встроенный в микросхему микропроцессора (СнК) видео ускоритель от AMD совместимый с Radeon Graphics и внешняя микросхема GeForce RTX3050 от NVIDIA.

23.1 Технология «PRIME GPU Offloading»

Изначально я предполагал, что смогу переключать монитор между двумя видеоадаптерами либо через BIOS, либо каким-то иным программным средством, но после небольшого изучения стало понятно, что монитор намертво прибит к встроенному видеоадаптеру (AMD GPU). Дальнейшее изучение вопроса поведало мне о новой технологии устройства видео карт в современных ПК — PRIME GPU Offloading, которой собственно и оснащен мой ноутбук Lenovo. Если описывать в двух словах, то данная технология работает так. В вычислительной системе всегда присутствует основная видеокарта (в моём случае это AMD GPU) и дополнительная (NVIDIA). При этом, оба видео монитора, **eDP** (Ж/К монитор) и **HDMI-A-0** намертво прибиты к основной видеокарте и переключение мониторов между видеокартами невозможно. Вместо этого, допускается передача вычислительной нагрузки с основной видеокарты (видеоускорителя) на вспомогательный с последующим отображением полученного результата на один из мониторов подключенных к основной видеокарте. Если копнуть чуть глубже, то нужно сказать, что PRIME это фича драйвера Direct Rendering Management (DRM) позволяющая передавать блоки данных между аппаратурой видеоакселератора и видеоадаптера без копирования, т. е. путём обмена указателями на физические блоки памяти. О самом DRM я расскажу чуть ниже.

Понятное дело, что такая сложная технология требует специальной поддержки как на уровне драйверов видео адаптеров, так и в самой графической подсистеме (X11). Прежде всего я попытался понять есть ли поддержка PRIME в сервере **Xorg**. Выяснилось, что требуемая поддержка для функционирования GPU Offloading в Xorg добавлена достаточно давно, но в README к драйверу от NVIDIA указывается, что требуется минимум версия **1.20.7**. Текущая версия Xorg в репозитории FreeBSD имеет номер 1.20.14, что обнадеживало, но вот дальше всё было гораздо печальнее. Короче, запустить оба акселератора мне удалось, но только в режиме «hybrid setup» («гибридный режим»). С режимом «PRIME GPU Offloading» у меня ничего не получилось и, подозреваю, связано это с тем, что Xorg 1.20.14 несколько староват - не содержит каких-то важных патчей и багфиксов. Сначала Xorg у меня крэшился на доступе к AMD GPU. Устранив эту проблему я обнаружил тот же самый крэш с доступом но уже к NVIDIA GPU. Прошли месяцы пока я нашел способ обойти это крэш и заставить Xorg работать с «безголовым» (headless) GPU, коим является NVIDIA GeForce RTX3050. Победив и эту проблему я столкнулся с тем, что Xorg не желает обнаруживать двух «провайдеров» в моей системе, что требуется для конфигурирования «PRIME GPU Offloading», а каждый GPU работает только как отдельный «экран». Гугление и анализ коммитов к Xorg убедил меня том, что без **Xorg** версии **21.1** эта фича PRIME попросту не работоспособна. Работы над портированием Xorg 21.1 на FreeBSD активно ведутся и есть шанс, что все эти привороты рано или поздно заработают и праздник случится на нашей улице. Ну, а чтобы не ждать, пока будем использовать оба GPU в «гибридном режиме» - по мне, так даже немного более удобней, о я чем поведаю ниже.

Прежде всего расскажу какие есть режимы и варианты использования GPU, а их несколько:

1. Задействовать только один GPU встроенный в микропроцессор (AMD GPU Radeon Graphics). Схема полностью рабочая, поддерживается всем доступным программным обеспечением, в том числе и требующим EGL. Недостаток только один — не сильно быстрый этот GPU. Зато экономный до электропитания.
2. Режим «PRIME Render Offload». В этом режиме основным видеоадаптером является AMD GPU, а часть расчета 3D графики от приложений отправляется на внешний (дискретный) GPU NVIDIA. Результат расчета в виде растровой картинки возвращается назад в основной видеоадаптер и отрисовывается на экране монитора

или его части (в окне). Управление тем, какой GPU использовать, производится с помощью переменных окружения. Данный режим мне запустить не удалось.

3. Режим «PRIME и Reverse PRIME через RandR». Тут два подрежима: «прямой» и «обратный».
   - В «прямом» подрежиме основным видеоадаптером является дискретный NVIDIA GPU который полностью рассчитывает 3D и 2D графику, а результат отправляет на встроенный AMD GPU который всего лишь отображает картинку на мониторе. AMD GPU при этом работает в режиме «modesetting» - то есть как средство отображения плоской картинки и переключения видеорежимов, его GPU не задействован.
   - В «обратном» подрежиме основным видеоадаптером является встроенный AMD GPU который работает с слоской (2D) графикой, а нагрузка по просчету 3D отправляется на дискретный NVIDIA GPU. Результат расчета возвращается и отображается с помощью основного видеоадаптера. Основной видеоадаптер AMD GPU так же как и в «прямом» подрежиме работает в режиме «modestting», т. е. его GPU тоже не используется.

   Всё управление в «прямом» и «обратном» подрежимах производится с помощью утилиты **xrandr**, которая с версии **1.4** позволяет коммутировать «источники» изображения (sources) и его «потребителей» (sink). Использование **xrandr** несколько проще, чем длинные и замысловатые переменные окружения. Недостаток — один GPU не используется и висит «мертвым грузом». Запустить этот режим мне так же не удалось.

- «Гибридны режим» является оберткой к VirtualGL. В этом режиме в операционной системе запущено два процесса X сервера (два **Xorg**-а), по одному на каждый видеоадаптер/акселератор. Каждому X серверу обеспечивается своим уникальным конфигурационным файлом (xorg.conf). При этом, X сервер работающий с дискретным NVIDIA GPU настраивается так, чтобы работать без монитора (режим «headless»), не использовать и не перехватывать средства ввода. X сервер который взаимодействует со встроенным AMD GPU видеоадаптером и к которому подключен монитор является основным, с ним взаимодействует пользователь и номер его X дисплея по которому X клиенты его адресуют равен :**0.0**. «Безголовый» X сервер работает в фоне и его номер дисплея равен :**8.0** (или любой другой отличный от основного). С помощью определенных манипуляций с переменными окружения и подменой библиотек можно объяснить любому X клиенту на какой из двух X серверов (X дисплеев) отрисовывать его окно, а следовательно каким GPU пользоваться. И тут тоже имеется два подрежима: «безголовый» и «с отображением графики».
  - «Безголовый» подрежим подразумевает, что все выполненные расчеты не попадают на экран и не отображаются пользователю. Данный подрежим может быть полезен для выполнения каких-то математических расчетов «в фоне».
  - Подрежим «с отображением графики» работает примерно так же как и «PRIME» - результат расчета передается на основной (нулевой) X сервер и отображается на нём в окне. Таким образом пользователь видит приложение и взаимодействует с ним.

Чисто теоретически, на практике это проверить сложно, но «гибридный режим» должен работать несколько медленнее чем режим «PRIME» из-за того, что большой объём данных циркулирует между двумя X серверами через системную оперативную память, а не напрямую межу двумя устройствами по DMA. Тем не менее, с моей точки зрения, «гибридный режим» более интересен нежели «PRIME» по следующим соображениям:

1. Задействованы оба GPU и пользователь может выбрать каким приложениям к какому GPU обращаться.
2. Имеется набор простых утилит (**nvrun** и **nvrun-vgl**), позволяющих легко переключать приложения между двумя GPU.
3. Внешний (дискретный) GPU NVIDIA существенно мощнее, а значит потребляет существенно больше электроэнергии. Возможность использовать одно и то же приложение на разных GPU в зависимости от наличия внешнего электропитания является немаловажной фичей при работе на ноутбуках.
4. Утилита **nvidia-smi** позволяет мониторить текущую загрузку дискретного GPU по-процессно.
5. Возможность в любой момент «потушить» любую активность на дискретном GPU одной командой **kill**, при этом не потеряв всю графическую подсистему — многого стоит.

Из недостатков «гибридного режиме»:

1. Через VirtualGL не работает технология Vulkan (этот вопрос пока открыт).
2. Не работает Linuxulator (эмулятор Linux) и Wine.

Не смотря на недостатки, «гибридный режим» мне како-то показался более интересным и я, пожалуй, останусь на нём даже если в Xorg под FreeBSD полноценно заработает технология PRIME.

Начнем настройку GPU со скачивания и установки двайверов для двух видеоадаптеров.

23.2 Установка драйвера для NVIDIA GPU

Драйвер для NVIDIA под FreeBSD выкачиваем с официального сайта: http://download.nvidia.com/XFree86/FreeBSD-x86_64/ , качаем самую последнюю версию (на момент написания статьи это была версия 515.57). Распаковываем, собираем и устанавливаем:

```
rz@butterfly:~ % wget http://download.nvidia.com/XFree86/FreeBSD-x86_64/515.57/NVIDIA-
FreeBSD-x86_64-515.57.tar.xz
rz@butterfly:~ % tar -zxpf NVIDIA-FreeBSD-x86_64-515.57.tar.xz
rz@butterfly:~ % cd NVIDIA-FreeBSD-x86_64-515.57
rz@butterfly:~/NVIDIA-FreeBSD-x86_64-515.57 % make && sudo make install
```

И тут мы наталкиваемся на одну серьезную засаду. Скрип инсталлятора автоматически добавит в конфигурацию bootloader-а (в файл **/boot/loader.conf**) несколько строк для загрузки соответствующих модулей ядра ОС. Этот файл нужно отредактировать вручную и убрать следующие строки:

```
nvidia_load="YES"
nvidia_name="nvidia"
nvidia_modeset_load="YES"
nvidia_modeset_name="nvidia-modeset"
```

Вместо этого имена модулей нужно поместить в строку переменной **kld_list** файла **/etc/rc.conf**:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf kld_list+="nvidia nvidia-modeset"
```

Если Вы по какой-то причине забыли или не успели «почистить» свой **/boot/loader.conf**, то при следующей перезагрузки Ваша система превратится в… нет не в тыкву, просто зависнет намертво после запуска загрузчика. Но ничего страшного в этом нет, воспользуйтесь инструкцией по загрузке «голого» ядра из раздела «**12. Проблема с загрузчиком**».

Загрузить только что инсталлированные модули драйвера можно следующей командой:

```
rz@butterfly:~ % sudo kldload nvidia nvidia-modeset
```

Результатом успешной установки и загрузки модулей драйвера для NVIDIA GPU является следующая информация выводимая драйвером в **dmesg**:

```
nvidia0: <NVIDIA GeForce RTX 3050 Laptop GPU> on vgapci0
nvidia-modeset: Loading NVIDIA Kernel Mode Setting Driver for UNIX platforms  515.57  Wed Jun
22 22:18:53 UTC 2022
```

Данный пакет драйверов от NVIDIA содержит в себе не только сам драйвер для работы с аппаратурой ускорителя в составе подсистемы DRM, но и DDX драйвер для X сервера **Xorg** (подкаталог **./x11/driver**) который инсталлируется в виде динамически загружаемой библиотеки **/usr/local/lib/xorg/modules/drivers/nvidia_drv.so** и будет задействован X сервером.

23.3 Установка DRM драйвера для AMD GPU

С драйвером для AMD GPU всё гораздо интересней. Драйвер для AMD GPU Radeon Graphics (а он так и называется «amdgpu»), придется позаимствовать у ОС Linux, так как родного во FreeBSD нет и, судя по всему, уже никогда не будет.

Дело в том, что с некоторых пор в ядре ОС FreeBSD появился фреймворк под названием **LinuxKPI** или Linux Kernel Programming Interface — средство позволяющее брать исходные коды драйверов из ядра Linux и, применив к ним ряд патчей, получать драйвера для ОС FreeBSD. Сразу замечу, что далеко не все драйвера из Linux можно таким вот чудесным мановением руки превратить в драйвера для ОС FreeBSD и без рукопригладства тут не обходится. На сколько мне известно фреймворк **LinuxKPI** на данный момент поддерживает только драйвера сетевых адаптеров и драйвера подсистемы DRM (Direct Rendering Management). Данный фреймворк появился в ОС FreeBSD с версии 11-RELEASE и работы над ним все еще активно ведутся, в частности по улучшению поддержки драйверов сетевых адаптеров.

Для того, чтобы задействовать какой либо драйвер из Linux посредством **LinuxKPI**, необходимы исходные коды самого драйвера, а так же исходные коды ядра ОС FreeBSD. Исходники ядра ОС я установил в самом начале при выполнении базовой инсталляции с USB Flash носителя. С исходниками драйверов все сложнее — выкорчевать их из Linux-а не просто, но добрые люди постарались за нас Вами и сделали соответствующие репозитории на Github-е.

По этому клонируем репозиторий под названием **freebsd/drm-kmod** — в нём содержатся адаптированные исходные коды драйверов для подсистемы DRM, в том числе исходники драйверов видеоадаптеров и видеоускорителей:

```
rz@butterfly:~ % git clone https://github.com/freebsd/drm-kmod.git
```

Замечу, что поддержка встроенного видеоускорителя для системы-на-кристалле AMD Cezanne (ChipID = 0x1638) появилась в DRM начиная с версии 5.7, но реально рабочей она стала только с версии **5.10**. Поэтому, зайдем в репозиторий и переключимся на ветку **5.10-lts**:

```
rz@butterfly:~ % cd drm-kmod
rz@butterfly:~/drm-kmod % git checkout 5.10-lts
branch '5.10-lts' set up to track 'origin/5.10-lts'.
Switched to a new branch '5.10-lts'
```

после чего проведем сборку и установку драйвера:

```
rz@butterfly:~/drm-kmod % make -j8

rz@butterfly:~/drm-kmod % sudo make install
```

и добавим модуль ядра **amdgpu** в переменную **kld_list** для загрузки в при старте системы:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf kld_list+="amdgpu"
```

Для того, чтобы включить систему динамического энергосбережения для AMD GPU рекомендуется изменить значения следующих MIB переменных. Так, как эти переменные имеют статус «read-only», то изменять их значения можно только в момент инициализации загрузчика ОС, следовательно их нужно добавить в файл конфигурации загрузчика **/boot/loader.conf**:

```
rz@butterfly:~ % grep amd /boot/loader.conf
compat.linuxkpi.amdgpu_dpm=1
compat.linuxkpi.amdgpu_bapm=1
compat.linuxkpi.amdgpu_runpm=1
compat.linuxkpi.amdgpu_dc=1
```

После чего выполнить перезагрузку ОС (команда **sudo reboot**). Да, к нашему всеобщему сожалению, иного способа ввести в действие эти переменные нет:

```
rz@butterfly:~ % sudo sysctl -w compat.linuxkpi.amdgpu_dpm=1
sysctl: oid 'compat.linuxkpi.amdgpu_dpm' is a read only tunable
sysctl: Tunable values are set in /boot/loader.conf
```

После того, как ОС FreeBSD загрузилась, следует убедиться, что модуль ядра **amdgpu** и все его сопутствующие части успешно загрузились. Для этого следует внимательно просмотреть сообщения в **dmesg** от подсистемы **drm** на наличии сообщений об ошибках. Сделать это можно командой:

```
rz@butterfly:~ % dmesg | grep drm
```

Ключевыми моментами здесь являются следующие сообщения:

```
drmn1: Fetched VBIOS from VFCT
drmn1: successfully loaded firmware image 'amdgpu/green_sardine_sdma.bin'
[drm] VCN decode is enabled in VM mode
[drm] VCN encode is enabled in VM mode
[drm] JPEG decode is enabled in VM mode
[drm] vm size is 262144 GB, 4 levels, block size is 9-bit, fragment size is 9-bit
drmn1: VRAM: 4096M 0x000000F400000000 - 0x000000F4FFFFFFFF (4096M used)
```

```
drmn1: GART: 1024M 0x0000000000000000 - 0x000000003FFFFFFF
drmn1: AGP: 267419648M 0x000000F800000000 - 0x0000FFFFFFFFFFFF
[drm] Detected VRAM RAM=4096M, BAR=4096M
[drm] RAM width 128bits DDR4
[drm] amdgpu: 4096M of VRAM memory ready
[drm] amdgpu: 4096M of GTT memory ready.
[drm] GART: num cpu pages 262144, num gpu pages 262144
[drm] PCIE GART of 1024M enabled (table at 0x000000F400900000).
drmn1: successfully loaded firmware image 'amdgpu/green_sardine_asd.bin'
drmn1: successfully loaded firmware image 'amdgpu/green_sardine_pfp.bin'
drmn1: successfully loaded firmware image 'amdgpu/green_sardine_me.bin'
drmn1: successfully loaded firmware image 'amdgpu/green_sardine_ce.bin'
drmn1: successfully loaded firmware image 'amdgpu/green_sardine_rlc.bin'
drmn1: successfully loaded firmware image 'amdgpu/green_sardine_mec.bin'
drmn1: successfully loaded firmware image 'amdgpu/green_sardine_mec2.bin'
drmn1: successfully loaded firmware image 'amdgpu/green_sardine_dmcub.bin'
[drm] Loading DMUB firmware via PSP: version=0x01010000A
drmn1: successfully loaded firmware image 'amdgpu/green_sardine_vcn.bin'
[drm] Found VCN firmware Version ENC: 1.7 DEC: 4 VEP: 0 Revision: 17
…
drmn1: SMU is initialized successfully!
…
[drm] Display Core initialized with v3.2.104!
[drm] DMUB hardware initialized: version=0x0101000A
[drm] VCN decode and encode initialized successfully(under DPG Mode).
[drm] JPEG decode initialized successfully.
…
vgapci1: child drmn1 requested pci_get_powerstate
[drm] Initialized amdgpu 3.40.0 20150101 for drmn1 on minor 0
```

Если Вы совсем не видите никаких сообщений от **drm**, это значит что либо Вы некорректно инсталлировали драйвер **drm-kmod**, либо модуль **amdgpu** не добавлен в список загрузки (см переменную **kld_list** файл **/etc/rc.conf**).

Узнать загружен ли модуль в данный момент можно командой **kldstat** следующим образом:

```
rz@butterfly:~ % kldstat | grep amdgpu
 9    1 0xffffffff82e00000   414220 amdgpu.ko
14    1 0xffffffff82ddf000     64d8 amdgpu_green_sardine_sdma_bin.ko
15    1 0xffffffff83215000    2e2d8 amdgpu_green_sardine_asd_bin.ko
16    1 0xffffffff82de6000    16fd8 amdgpu_green_sardine_pfp_bin.ko
17    1 0xffffffff83244000    12fd8 amdgpu_green_sardine_me_bin.ko
18    1 0xffffffff83257000     afd8 amdgpu_green_sardine_ce_bin.ko
19    1 0xffffffff83262000     b8d0 amdgpu_green_sardine_rlc_bin.ko
20    1 0xffffffff8326e000    437e8 amdgpu_green_sardine_mec_bin.ko
21    1 0xffffffff832b2000    437e8 amdgpu_green_sardine_mec2_bin.ko
22    1 0xffffffff832f6000    1d3d0 amdgpu_green_sardine_dmcub_bin.ko
23    1 0xffffffff83314000    71d58 amdgpu_green_sardine_vcn_bin.ko
```

Если же Вы наблюдаете какие-то сообщения об ошибках, при этом модуль не рапортует о том, что он успешно загрузился и проинициализировался, то Ваше дело «швах» и Вам прямая дорога на форум поддержки пользователей FreeBSD: https://forums.freebsd.org/

23.4 Установка драйвера к AMD GPU для X сервера Xorg

Как я отмечал выше, для того чтобы X сервер мог взаимодействовать с видеокартой или видеоускорителем ему требуется прослойка в виде DDX драйвера — модуль, который X сервер подгружает в процессе своей инициализации и использует для взаимодействия с DRM видеоподсистемой. Для Xorg таким модулем является **xf86-video-amdgpu** (напомню, что Xorg является наследником проекта **XFree86**), установить который можно как из пакетов, так и из коллекции портов.

Разнообразия ради выполним установку этого DDX драйвера из портов. Для этого зайдем в каталог **/usr/ports/x11-drivers/xf86-video-amdgpu**, прочитаем краткую информацию о нём:

```
rz@butterfly:~ % cd /usr/ports/x11-drivers/xf86-video-amdgpu

rz@butterfly:/usr/ports/x11-drivers/xf86-video-amdgpu % cat pkg-descr
This package contains the X.Org xf86-video-amdgpu driver.

The amdgpu driver supports AMD Radeon chipsets: OLAND, HAINAN, TAHITI, PITCAIRN,
VERDE, BONAIRE, KABINI, MULLINS, KAVERI, HAWAII, TOPAZ, TONGA, CARRIZO, FIJI,
STONEY, POLARIS11, POLARIS10

On FreeBSD requires amdgpu KMS driver from graphics/drm-kmod.

WWW: https://www.x.org/wiki/RadeonFeature/
```

Информация в этом описании несколько устарела, в списке поддерживаемых чипсетов отсутствует **RENOIR** — это тот самый AMD GPU «чипсет» с идентификатором ChipID = 0x1638 присутствующий в СнК AMD Ryzen 5600H на моём ноутбуке.

Замечу, что у AMD царит полный бардак с именованием своих изделий — одна и та же сущность (в данном случае GPU) имеет несколько различных названий: «Radeon Graphics», «Cenzanne», «RENOIR». Если Вы будете искать информацию в Сети, то гуглите сразу по всем трем словам.

Запустим сборку:

```
rz@butterfly:/usr/ports/x11-drivers/xf86-video-amdgpu % sudo make -j4
```

И установку:

```
rz@butterfly:/usr/ports/x11-drivers/xf86-video-amdgpu % sudo make install
```

На этом установка драйверов видеоускорителей завершена и можно переходить к водным процедурам, т. е. выпить чаю, пройти еще один уровень в NetHack и, собравшись с силами, смело приступать к настройке X сервера.

23.5 Настройка Xorg на работу с 3D ускорителем AMD GPU

Я уже отмечал, что в теории, Xorg достаточно интеллектуален чтобы при запуске без всякого конфига и настроек обнаружить всё доступное оборудование и приступить к работе. К сожалению, конкретно на моём «железе» такой подход не работает даже при наличии всех необходимых драйверов. Мне пришлось потратить не мало времени (несколько месяцев изысканий и общения на форумах, а так же дождаться выхода соответствующих патчей),

чтобы выяснить все тонкости с подбором драйверов и опций в конфиге Xorg и заставить его работать. Не буду мучить читателя подробностями этой увлекательной истории, сразу приступлю к делу.

И так, напомню, что у меня уже имеется рабочий Xorg который работает с VESA совместимым драйвером **scfb** и использует встроенную видеокарту в режиме «modesetting», т. е. без какого либо ускорения. Для того, чтобы перейти на драйвер с аппаратным ускорителем, необходимо указать его имя **amdgpu** в параметре **Driver** в секции **Device** в файле конфигурации **/etc/X11/xorg.conf**, вот таким образом:

```
Section "ServerLayout"
      Identifier "layout"
      Screen 0 "iGPU"
      InputDevice    "Keyboard0" "CoreKeyboard"
      InputDevice    "Mouse0" "CorePointer"
EndSection

Section "Screen"
      Identifier "iGPU"
      Device "iGPU"
EndSection

Section "Device"
      Identifier "iGPU"
      Driver "amdgpu"
      BusID "PCI:5:0:0" p
EndSection
```

Так же очень желательно указать параметр **BusID** с идентификатором PCI шины на которой находится данный GPU чип, иначе чуда может не случиться.

Подправив соответствующим образом конфигурационный файл, следует перейти в VT консоль нажатием сочетания клавишь **Ctrl-Alt-F1**, войти в систему и перезапустить сервис дисплейного менеджера SDDM:

```
rz@butterfly:~ % sudo service sddm restart
```

При этом произойдет перезапуск X сервера и, если всё в порядке, то видеоподсистема перейдет в графический режим и на экране появится обычное окно для ввода логина и пароля.

В случае, если X сервер не смог перезапуститься по какой либо причине, необходимо внимательно прочитать содержимое лог файла **/var/log/Xorg.0.log** в котором будут описаны все шаги которые предпринимал Xorg при запуске и что именно его не устроило — с большой вероятностью могу сказать, что Вы допустили опечатку в конфиге. Подправив конфиг нужно еще раз перезапустить SDDM.

Если у Вас произошел нормальный запуск X сервера и Вы видите окно логина, я всё равно рекомендую заглянуть в лог файл и убедиться в том, что в нём присутствует инициализация AMD GPU:

```
rz@butterfly:~ % grep AMDGPU /var/log/Xorg.0.log
[    18.664] (II) AMDGPU: Driver for AMD Radeon:
[    18.678] (II) AMDGPU(0): [KMS] Kernel modesetting enabled.
[    20.021] (II) AMDGPU(0): Creating default Display subsection in Screen section
[    20.021] (==) AMDGPU(0): Depth 24, (--) framebuffer bpp 32
```

```
[    20.021] (II) AMDGPU(0): Pixel depth = 24 bits stored in 4 bytes (32 bpp pixmaps)
[    20.021] (==) AMDGPU(0): Default visual is TrueColor
[    20.022] (==) AMDGPU(0): RGB weight 888
[    20.022] (II) AMDGPU(0): Using 8 bits per RGB (8 bit DAC)
[    20.022] (--) AMDGPU(0): Chipset: "Unknown AMD Radeon GPU" (ChipID = 0x1638)
[    20.308] (II) AMDGPU(0): glamor X acceleration enabled on AMD RENOIR (DRM 3.40.0, 13.1-
RELEASE, LLVM 13.0.1)
[    20.308] (II) AMDGPU(0): glamor detected, initialising EGL layer.
[    20.308] (==) AMDGPU(0): TearFree property default: auto
[    20.308] (==) AMDGPU(0): VariableRefresh: disabled
[    20.308] (==) AMDGPU(0): AsyncFlipSecondaries: disabled
[    20.308] (II) AMDGPU(0): KMS Pageflipping: enabled
[    20.308] (II) AMDGPU(0): Output eDP has no monitor section
[    20.308] (II) AMDGPU(0): Output HDMI-A-0 has no monitor section
[    20.323] (II) AMDGPU(0): EDID for output eDP
[    20.323] (II) AMDGPU(0): Manufacturer: AUO  Model: 4a99  Serial#: 0
[    20.323] (II) AMDGPU(0): Year: 2021  Week: 1
[    20.323] (II) AMDGPU(0): EDID Version: 1.4
[    20.323] (II) AMDGPU(0): Digital Display Input
…
```

После чего можно смело входить в систему через DM и проводить тесты. Прежде всего запустим окно **xterm** (или любого другого эмулятора терминала) и установим полезные утилиты:

```
rz@butterfly:~ % sudo pkg inst --yes glx-utils
rz@butterfly:~ % rehash
```

Теперь можно просмотреть информацию о подсистеме OpenGL утилитой **glxinfo**:

```
rz@butterfly:~ % glxinfo -B
name of display: :0.0
display: :0  screen: 0
direct rendering: Yes
Extended renderer info (GLX_MESA_query_renderer):
    Vendor: AMD (0x1002)
    Device: AMD RENOIR (DRM 3.40.0, 13.1-RELEASE, LLVM 13.0.1) (0x1638)
    Version: 21.3.8
    Accelerated: yes
    Video memory: 4096MB
    Unified memory: no
    Preferred profile: core (0x1)
    Max core profile version: 4.6
    Max compat profile version: 4.6
    Max GLES1 profile version: 1.1
    Max GLES[23] profile version: 3.2
Memory info (GL_ATI_meminfo):
    VBO free memory - total: 3842 MB, largest block: 3842 MB
    VBO free aux. memory - total: 3863 MB, largest block: 3863 MB
    Texture free memory - total: 3842 MB, largest block: 3842 MB
    Texture free aux. memory - total: 3863 MB, largest block: 3863 MB
    Renderbuffer free memory - total: 3842 MB, largest block: 3842 MB
    Renderbuffer free aux. memory - total: 3863 MB, largest block: 3863 MB
Memory info (GL_NVX_gpu_memory_info):
    Dedicated video memory: 4096 MB
    Total available memory: 8192 MB
    Currently available dedicated video memory: 3842 MB
OpenGL vendor string: AMD
OpenGL renderer string: AMD RENOIR (DRM 3.40.0, 13.1-RELEASE, LLVM 13.0.1)
OpenGL core profile version string: 4.6 (Core Profile) Mesa 21.3.8
OpenGL core profile shading language version string: 4.60
OpenGL core profile context flags: (none)
OpenGL core profile profile mask: core profile

OpenGL version string: 4.6 (Compatibility Profile) Mesa 21.3.8
OpenGL shading language version string: 4.60
OpenGL context flags: (none)
OpenGL profile mask: compatibility profile
```

```
OpenGL ES profile version string: OpenGL ES 3.2 Mesa 21.3.8
OpenGL ES profile shading language version string: OpenGL ES GLSL ES 3.20
```

На этом можно констатировать тот факт, что 3D ускоритель AMD GPU встроенный в СнК полностью функционирует и его можно смело использовать.

Но что же делать с если что-то пошло не так и всё не так как хочется: «иксы» не запускаются, модули не загружаются, и т. д. и т.п. ? Прежде всего нужно собрать как можно больше информации о Вашей системе, в том числе: версию ОС, версии Xorg и устанавливаемых драйверов, а так же вывод **dmesg** и лог Xorg-а — Вы четко должны представлять что, когда и куда Вы устанавливали, какие конфиги изменяли и какой результат при этом получили. После чего можно задать квалифицированный вопрос на форуме https://forums.freebsd.org/ указав всю собранную Вами информацию и детально описав проблему. С очень большой вероятностью, что Вам помогут.

Что еще могу посоветовать — не бойтесь экспериментировать, пробуйте разные варианты, версии, патчи и коммиты, при этом всегда сохраняйте предыдущие версии конфигов и драйверов и исходников, чтобы можно было откатиться к исходной точке.

23.6 «Гибридный режим» - запуск вспомогательного Xorg для работы с акселератором NVIDIA GeForce

Для того, чтобы задействовать второй (дискретный) 3D ускоритель NVIDIA GeForce RTX 3050 в «гибридном режиме» необходимо решить две задачи: 1) запустить еще одну копию X сервера Xorg со своим отдельным конфигурационным файлом, и 2) решить вопрос маршрутизации запросов по протоколу X11 от X клиентов к нужному X серверу. И тут обнаружилось, что для ОС FreeBSD давно существует порт (и пакет) под названием **x11/nvidia-hybrid-graphics**, который решает именно эти задачи.

Зайдем в каталог **/usr/ports/x11/nvidia-hybrid-graphics** и прочитаем описание этого порта:

```
rz@butterfly:~ % cd /usr/ports/x11/nvidia-hybrid-graphics/

rz@butterfly:/usr/ports/x11/nvidia-hybrid-graphics % cat pkg-descr
This port integrates the Nvidia graphics driver and supporting utilities with
VirtualGL, enabling use of Nvidia acceleration on a system with hybrid graphics
hardware configuration, aka "Optimus".

Applications to be run with Nvidia acceleration should be started using
`nvrun-vgl`.

WWW: https://github.com/therontarigo/freebsd-gpu-headless
```

Прежде чем перейти непосредственно к установке, я хочу внести несколько замечаний:

1) Порт **nvidia-hybrid-graphics** предназначен для видеоадаптеров производства NVIDIA с поддержкой технологии «Optimus». В двух словах «Optimus» это технология передачи вычислительной нагрузки (GPU Offloading) между «безголовым» ускорителем от NVIDIA.
2) Данный порт/пакет представляет собой обертку для старой и доброй технологии VirtualGL — способ передачи вычислительной нагрузки на отдельно стоящие 3D ускорители с последующим отображением результата на X дисплее пользователя используя стандартный X11 протокол.

3) В процессе работы VirtualGL осуществляет передачу больших объемов данных между видеокартами. Как конкретно это реализовано в **nvidia-hybrid-graphics** я точно сказать не могу, но после ряда экспериментов я могу с большой вероятностью утверждать, что встроенный в DRM механизм передачи указателей на аппаратные буфера (PRIME) в VirtualGL не используется, а следовательно между двумя процессами Xorg циркулирует большой объём трафика (готовые отрендеренные битмапы), что может сказываться на производительности всей ОС.
4) VirtualGL не поддерживает технологию Vulkan, а значит приложения которые используют OpenGL и Vulkan одновременно работать не будут. Но отдельно стоящие приложения для Vulkan-а скорее всего заработают.

Не смотря на всё вышесказанное, «гибридный режим» позволяет достаточно эффективно задействовать вычислительные ресурсы дискретного ускорителя от NVIDIA, что может быть полезно для некоторых специфических приложений, как то при работе в различных САПР (FreeCAD) или при рендеринге 3D анимации (Blender).

Установим данный порт следующей командой:

```
rz@butterfly:/usr/ports/x11/nvidia-hybrid-graphics % sudo make install
```

Добавим запуск вспомогательного **Xorg** сервера в файл **/etc/rc.conf**:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf nvidia_xorg_enable="YES"
nvidia_xorg_enable:  -> YES
```

И тут я натолкнулся на еще один неприятный нюанс — дело в том, что в Xorg версии 1.20.14 есть баг, который приводит к его крэшу при запуске в процессе инициализации «безголового» акселератора. Багфикса к этой проблеме для ОС FreeBSD на данный монет не существует, но есть обход. Я подозреваю, что эта проблема устранена в более поздних версиях Xorg которые пользователям FreeBSD на данный момент недоступны.

Короче, нужно явно объяснить Xorg-у, что он имеет дело с «безголовым» ускорителем с помощью опции **Option "UseDisplayDevice" "None"** в секции **Device** и так же, как для AMD GPU указать параметр **BusID** с расположением устройства NVIDIA GeForce на шине PCI. Сделать это нужно в файле конфигурации вспомогательного Xorg, который находится в **/usr/local/etc/X11/xorg-nvidia-headless.conf**.

Ниже привожу полное содержимое этого файла в моей системе:

```
rz@butterfly:~ % cat /usr/local/etc/X11/xorg-nvidia-headless.conf

Section "ServerLayout"
    Identifier     "nvidia"
    Screen      0  "Screen0"
    InputDevice    "fake" "CorePointer" "CoreKeyboard"
    Option         "AutoAddDevices" "false"
EndSection

Section "Files"
    ModulePath      "/usr/local/lib/xorg/modules-NVIDIA"
    ModulePath      "/usr/local/lib/xorg/modules"
EndSection

Section "Module"
    Load           "dri3"
```

```
    Load           "glx"
    Disable        "efifb"
EndSection

Section "InputDevice"
    Identifier     "fake"
    Driver         ""
EndSection

Section "Monitor"
    Identifier     "Monitor0"
EndSection

Section "Device"
    Identifier     "Device0"
    Driver         "nvidia"
    BusID      "PCI:1:0:0
    Option     "UseDisplayDevice" "None"
EndSection

Section "Screen"
    Identifier     "Screen0"
    Device         "Device0"
    Monitor        "Monitor0"
EndSection
```

После того как внесены все правки в конфигурационный файл, запустить вспомогательный X сервер можно следующей командой:

```
rz@butterfly:~ % sudo service nvidia_xorg start
```

Никаких видимы эффектов после этой команды ожидать не следует. Для того, чтобы выяснить запущен ли вспомогательный Xorg можно запросить список процессов и посмотреть сколько одновременных процессов Xorg присутствует на данный момент и с какими параметрами они запущены:

```
rz@butterfly:~ % ps auxww | grep 'Xorg '
root       11547    2.3  1.0   524716 206228 v0  S    05:05       2:41.63 /usr/local/bin/Xorg
-nolisten tcp -auth /var/run/sddm/{ae1a94b0-8017-4c27-9767-8aa0fd9b5849} -background none -
noreset -displayfd 17 -seat seat0 vt9
root        1514    0.0  0.1 25285412  28156  -  I    Wed02       0:01.54 /usr/local/bin/Xorg
-sharevts -novtswitch -noreset -config xorg-nvidia-headless.conf -configdir xorg-nvidia-
headless.conf.d :8
```

Из вывода команды **ps** видно, что в моей системе сейчас запущено два процесса **Xorg**, для второго явно задано номер дисплея **:8** — именно к этому дисплею должны подключаться X клиенты если им требуется работать с NVIDIA GPU.

Войдем в систему через DM и проверим что нам выдает утилита **glxinfo**. Для того, чтобы настроить клиента на работу со вспомогательным X сервером в состав установленного порта **nvidia-hybrid-graphics** входит две утилиты-скрипта: **nvrun** и **nvrun-vgl** которые запускаю X клиента предварительно настроив переменные **DISPLAY** и **LD_PRELOAD** на работу с оберткой VirtualGL. Отличие между ними состоит в том, что первая запускает клиента в «безголовом» режим — без отображения вывода, а вторая делает так, что весь графический вывод клиента транслируется на основной X сервер и отображается пользователю.

Запустим **glxinfo** в безголовом режиме:

```
rz@butterfly:~ % nvrun glxinfo -B
```

```
name of display: :8
display: :8  screen: 0
direct rendering: Yes
Memory info (GL_NVX_gpu_memory_info):
    Dedicated video memory: 4096 MB
    Total available memory: 4096 MB
    Currently available dedicated video memory: 3904 MB
OpenGL vendor string: NVIDIA Corporation
OpenGL renderer string: NVIDIA GeForce RTX 3050 Laptop GPU/PCIe/SSE2
OpenGL core profile version string: 4.6.0 NVIDIA 515.57
OpenGL core profile shading language version string: 4.60 NVIDIA
OpenGL core profile context flags: (none)
OpenGL core profile profile mask: core profile

OpenGL version string: 4.6.0 NVIDIA 515.57
OpenGL shading language version string: 4.60 NVIDIA
OpenGL context flags: (none)
OpenGL profile mask: (none)

OpenGL ES profile version string: OpenGL ES 3.2 NVIDIA 515.57
OpenGL ES profile shading language version string: OpenGL ES GLSL ES 3.20
```

Как видно, NVIDIA GeForce RTX 3050 распознан и доступен для работы. На этом можно сказать, что второй (дискретный) GPU работает и можно провести небольшие тесты двух ускорителей, о чем я поведаю далее.

24. Тестирование 3D ускорителей

Приведенные ниже тесты 3D ускорителей являются очень поверхностными и ни коим образом не претендуют на роль бенчмарков, единственная их цель — понять работает ли аппаратура, дает ли она какой-то выигрыш в производительности и с какими последствиями.

Самый простой способ оценить производительность аппаратного 3D ускорителя это запустить утилиту **glxgears** на полный экран и наблюдать параметр FPS на стандартном выводе.

Запустим **glxgears** на AMD GPU:

```
rz@butterfly:~ % glxgears -fullscreen -info
Running synchronized to the vertical refresh.  The framerate should be
approximately the same as the monitor refresh rate.
GL_RENDERER   = AMD RENOIR (DRM 3.40.0, 13.1-RELEASE, LLVM 13.0.1)
GL_VERSION    = 4.6 (Compatibility Profile) Mesa 21.3.8
GL_VENDOR     = AMD
…
VisualID 1276, 0x4fc
304 frames in 5.0 seconds = 60.601 FPS
301 frames in 5.0 seconds = 60.059 FPS
301 frames in 5.0 seconds = 60.036 FPS
301 frames in 5.0 seconds = 60.051 FPS
^C
```

Этот же самый тест на NVIDIA GPU — запускаем **glxgears** через утилиту **nvrun-vgl**:

```
rz@butterfly:~ % nvrun-vgl glxgears -fullscreen -info
GL_RENDERER   = NVIDIA GeForce RTX 3050 Laptop GPU/PCIe/SSE2
GL_VERSION    = 4.6.0 NVIDIA 515.57
GL_VENDOR     = NVIDIA Corporation
...
VisualID 33, 0x21
1605 frames in 5.0 seconds = 320.869 FPS
1656 frames in 5.0 seconds = 331.177 FPS
1609 frames in 5.0 seconds = 321.657 FPS
```

```
1626 frames in 5.0 seconds = 325.063 FPS
^C
```

Видно, что на такой примитивной задаче (вращение зубчатого колеса) GPU NVIDIA выдает почти на порядок большее количество FPS чем её собрат от AMD, но я не стал доверять этим цифрам и следом провел ряд замеров производительности с помощью утилиты **glmark2** которая всё разложила по полочкам.

И так, установим утилиту **glmark2** из пакетов:

```
rz@butterfly:~ % sudo pkg inst --yes glx-utils glmark2
rz@butterfly:~ % rehash
```

и запустим тест **glmark2** на 3D ускорителе AMD RENOIR, сначала в «безголовом» режиме чтобы оценить чистую производительность ускорителя. В приведенной ниже команде параметр **--fullscreen** указывает на размер битмапа который нужно формировать на выходе ускорителя — во весь экран (в моём случае 1920x1080), а **--off-screen** сообщает утилите о том, то битмап отображать на экране не следует - «безголовый» режим.

```
rz@butterfly:~ % glmark2 --fullscreen --off-screen
=======================================================
    glmark2 2021.12
=======================================================
    OpenGL Information
    GL_VENDOR:     AMD
    GL_RENDERER:   AMD RENOIR (DRM 3.40.0, 13.1-RELEASE, LLVM 13.0.1)
    GL_VERSION:    4.6 (Compatibility Profile) Mesa 21.3.8
=======================================================
[build] use-vbo=false: FPS: 9868 FrameTime: 0.101 ms
[build] use-vbo=true: FPS: 12158 FrameTime: 0.082 ms
[texture] texture-filter=nearest: FPS: 8086 FrameTime: 0.124 ms
[texture] texture-filter=linear: FPS: 8179 FrameTime: 0.122 ms
[texture] texture-filter=mipmap: FPS: 7714 FrameTime: 0.130 ms
[shading] shading=gouraud: FPS: 5961 FrameTime: 0.168 ms
[shading] shading=blinn-phong-inf: FPS: 5450 FrameTime: 0.183 ms
[shading] shading=phong: FPS: 5339 FrameTime: 0.187 ms
[shading] shading=cel: FPS: 5014 FrameTime: 0.199 ms
[bump] bump-render=high-poly: FPS: 5350 FrameTime: 0.187 ms
[bump] bump-render=normals: FPS: 13093 FrameTime: 0.076 ms
[bump] bump-render=height: FPS: 12481 FrameTime: 0.080 ms
[effect2d] kernel=0,1,0;1,-4,1;0,1,0;: FPS: 3438 FrameTime: 0.291 ms
[effect2d] kernel=1,1,1,1,1;1,1,1,1,1;1,1,1,1,1;: FPS: 1363 FrameTime: 0.734 ms
[pulsar] light=false:quads=5:texture=false: FPS: 7082 FrameTime: 0.141 ms
[desktop] blur-radius=5:effect=blur:passes=1:separable=true:windows=4: FPS: 1469 FrameTime:
0.681 ms
[desktop] effect=shadow:windows=4: FPS: 2936 FrameTime: 0.341 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-
method=map: FPS: 1766 FrameTime: 0.566 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-
method=subdata: FPS: 1500 FrameTime: 0.667 ms
[buffer] columns=200:interleave=true:update-dispersion=0.9:update-fraction=0.5:update-
method=map: FPS: 2266 FrameTime: 0.441 ms
[ideas] speed=duration: FPS: 4917 FrameTime: 0.203 ms
[jellyfish] <default>: FPS: 2514 FrameTime: 0.398 ms
[terrain] <default>: FPS: 151 FrameTime: 6.623 ms
[shadow] <default>: FPS: 3660 FrameTime: 0.273 ms
[refract] <default>: FPS: 297 FrameTime: 3.367 ms
[conditionals] fragment-steps=0:vertex-steps=0: FPS: 6552 FrameTime: 0.153 ms
[conditionals] fragment-steps=5:vertex-steps=0: FPS: 6025 FrameTime: 0.166 ms
[conditionals] fragment-steps=0:vertex-steps=5: FPS: 6499 FrameTime: 0.154 ms
[function] fragment-complexity=low:fragment-steps=5: FPS: 6015 FrameTime: 0.166 ms
[function] fragment-complexity=medium:fragment-steps=5: FPS: 5938 FrameTime: 0.168 ms
[loop] fragment-loop=false:fragment-steps=5:vertex-steps=5: FPS: 6004 FrameTime: 0.167 ms
```

```
[loop] fragment-steps=5:fragment-uniform=false:vertex-steps=5: FPS: 5996 FrameTime: 0.167 ms
[loop] fragment-steps=5:fragment-uniform=true:vertex-steps=5: FPS: 6044 FrameTime: 0.165 ms
=======================================================
                                  glmark2 Score: 5488
=======================================================
```

Теперь запустим тест **glmark2** на 3D ускорителе AMD RENOIR с отображением битмапа на экран:

```
rz@butterfly:~ % glmark2 --fullscreen
=======================================================
    glmark2 2021.12
=======================================================
    OpenGL Information
    GL_VENDOR:     AMD
    GL_RENDERER:   AMD RENOIR (DRM 3.40.0, 13.1-RELEASE, LLVM 13.0.1)
    GL_VERSION:    4.6 (Compatibility Profile) Mesa 21.3.8
=======================================================
[build] use-vbo=false: FPS: 6217 FrameTime: 0.161 ms
[build] use-vbo=true: FPS: 6282 FrameTime: 0.159 ms
[texture] texture-filter=nearest: FPS: 4561 FrameTime: 0.219 ms
[texture] texture-filter=linear: FPS: 4618 FrameTime: 0.217 ms
[texture] texture-filter=mipmap: FPS: 4511 FrameTime: 0.222 ms
[shading] shading=gouraud: FPS: 4222 FrameTime: 0.237 ms
[shading] shading=blinn-phong-inf: FPS: 3953 FrameTime: 0.253 ms
[shading] shading=phong: FPS: 3937 FrameTime: 0.254 ms
[shading] shading=cel: FPS: 3775 FrameTime: 0.265 ms
[bump] bump-render=high-poly: FPS: 3934 FrameTime: 0.254 ms
[bump] bump-render=normals: FPS: 6312 FrameTime: 0.158 ms
[bump] bump-render=height: FPS: 6187 FrameTime: 0.162 ms
[effect2d] kernel=0,1,0;1,-4,1;0,1,0;: FPS: 2748 FrameTime: 0.364 ms
[effect2d] kernel=1,1,1,1,1;1,1,1,1,1;1,1,1,1,1;: FPS: 1227 FrameTime: 0.815 ms
[pulsar] light=false:quads=5:texture=false: FPS: 4108 FrameTime: 0.243 ms
[desktop] blur-radius=5:effect=blur:passes=1:separable=true:windows=4: FPS: 1382 FrameTime:
0.724 ms
[desktop] effect=shadow:windows=4: FPS: 2526 FrameTime: 0.396 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-
method=map: FPS: 1969 FrameTime: 0.508 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-
method=subdata: FPS: 2465 FrameTime: 0.406 ms
[buffer] columns=200:interleave=true:update-dispersion=0.9:update-fraction=0.5:update-
method=map: FPS: 2369 FrameTime: 0.422 ms
[ideas] speed=duration: FPS: 3828 FrameTime: 0.261 ms
[jellyfish] <default>: FPS: 1905 FrameTime: 0.525 ms
[terrain] <default>: FPS: 153 FrameTime: 6.536 ms
[shadow] <default>: FPS: 2908 FrameTime: 0.344 ms
[refract] <default>: FPS: 318 FrameTime: 3.145 ms
[conditionals] fragment-steps=0:vertex-steps=0: FPS: 4321 FrameTime: 0.231 ms
[conditionals] fragment-steps=5:vertex-steps=0: FPS: 4035 FrameTime: 0.248 ms
[conditionals] fragment-steps=0:vertex-steps=5: FPS: 4324 FrameTime: 0.231 ms
[function] fragment-complexity=low:fragment-steps=5: FPS: 4028 FrameTime: 0.248 ms
[function] fragment-complexity=medium:fragment-steps=5: FPS: 4028 FrameTime: 0.248 ms
[loop] fragment-loop=false:fragment-steps=5:vertex-steps=5: FPS: 4040 FrameTime: 0.248 ms
[loop] fragment-steps=5:fragment-uniform=false:vertex-steps=5: FPS: 4044 FrameTime: 0.247 ms
[loop] fragment-steps=5:fragment-uniform=true:vertex-steps=5: FPS: 4038 FrameTime: 0.248 ms
=======================================================
                                  glmark2 Score: 3614
=======================================================
```

Этот же тест **glmark2** на 3D ускорителе NVIDIA GeForce RTX 3050 в «безголовом» режиме — используем утилиту **nvrun**:

```
rz@butterfly:~ % nvrun glmark2 --fullscreen
=======================================================
```

```
    glmark2 2021.12
=======================================================
    OpenGL Information
    GL_VENDOR:     NVIDIA Corporation
    GL_RENDERER:   NVIDIA GeForce RTX 3050 Laptop GPU/PCIe/SSE2
    GL_VERSION:    4.6.0 NVIDIA 515.57
=======================================================
[build] use-vbo=false: FPS: 7266 FrameTime: 0.138 ms
[build] use-vbo=true: FPS: 32034 FrameTime: 0.031 ms
[texture] texture-filter=nearest: FPS: 31400 FrameTime: 0.032 ms
[texture] texture-filter=linear: FPS: 31230 FrameTime: 0.032 ms
[texture] texture-filter=mipmap: FPS: 31596 FrameTime: 0.032 ms
[shading] shading=gouraud: FPS: 27793 FrameTime: 0.036 ms
[shading] shading=blinn-phong-inf: FPS: 27753 FrameTime: 0.036 ms
[shading] shading=phong: FPS: 26840 FrameTime: 0.037 ms
[shading] shading=cel: FPS: 26844 FrameTime: 0.037 ms
[bump] bump-render=high-poly: FPS: 17968 FrameTime: 0.056 ms
[bump] bump-render=normals: FPS: 33425 FrameTime: 0.030 ms
[bump] bump-render=height: FPS: 33011 FrameTime: 0.030 ms
[effect2d] kernel=0,1,0;1,-4,1;0,1,0;: FPS: 27867 FrameTime: 0.036 ms
[effect2d] kernel=1,1,1,1,1;1,1,1,1,1;1,1,1,1,1;: FPS: 20865 FrameTime: 0.048 ms
[pulsar] light=false:quads=5:texture=false: FPS: 31515 FrameTime: 0.032 ms
[desktop] blur-radius=5:effect=blur:passes=1:separable=true:windows=4: FPS: 9110 FrameTime:
0.110 ms
[desktop] effect=shadow:windows=4: FPS: 11403 FrameTime: 0.088 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-
method=map: FPS: 1551 FrameTime: 0.645 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-
method=subdata: FPS: 2459 FrameTime: 0.407 ms
[buffer] columns=200:interleave=true:update-dispersion=0.9:update-fraction=0.5:update-
method=map: FPS: 1800 FrameTime: 0.556 ms
[ideas] speed=duration: FPS: 14879 FrameTime: 0.067 ms
[jellyfish] <default>: FPS: 21732 FrameTime: 0.046 ms
[terrain] <default>: FPS: 1501 FrameTime: 0.666 ms
[shadow] <default>: FPS: 18275 FrameTime: 0.055 ms
[refract] <default>: FPS: 3422 FrameTime: 0.292 ms
[conditionals] fragment-steps=0:vertex-steps=0: FPS: 30862 FrameTime: 0.032 ms
[conditionals] fragment-steps=5:vertex-steps=0: FPS: 30607 FrameTime: 0.033 ms
[conditionals] fragment-steps=0:vertex-steps=5: FPS: 30708 FrameTime: 0.033 ms
[function] fragment-complexity=low:fragment-steps=5: FPS: 30769 FrameTime: 0.033 ms
[function] fragment-complexity=medium:fragment-steps=5: FPS: 30040 FrameTime: 0.033 ms
[loop] fragment-loop=false:fragment-steps=5:vertex-steps=5: FPS: 30692 FrameTime: 0.033 ms
[loop] fragment-steps=5:fragment-uniform=false:vertex-steps=5: FPS: 30694 FrameTime: 0.033 ms
[loop] fragment-steps=5:fragment-uniform=true:vertex-steps=5: FPS: 30442 FrameTime: 0.033 ms
=======================================================
                                  glmark2 Score: 22374
=======================================================
```

Ну и крайний тест - **glmarg2** на 3D ускорителе NVIDIA GeForce RTX 3050 в режиме с отображением картинки на экран. Для этого используем утилиту **nvrun-vgl**:

```
rz@butterfly:~ % nvrun-vgl glmark2 --fullscreen
=======================================================
    glmark2 2021.12
=======================================================
    OpenGL Information
    GL_VENDOR:     NVIDIA Corporation
    GL_RENDERER:   NVIDIA GeForce RTX 3050 Laptop GPU/PCIe/SSE2
    GL_VERSION:    4.6.0 NVIDIA 515.57
=======================================================
[build] use-vbo=false: FPS: 380 FrameTime: 2.632 ms
[build] use-vbo=true: FPS: 362 FrameTime: 2.762 ms
[texture] texture-filter=nearest: FPS: 356 FrameTime: 2.809 ms
[texture] texture-filter=linear: FPS: 359 FrameTime: 2.786 ms
[texture] texture-filter=mipmap: FPS: 359 FrameTime: 2.786 ms
[shading] shading=gouraud: FPS: 362 FrameTime: 2.762 ms
[shading] shading=blinn-phong-inf: FPS: 360 FrameTime: 2.778 ms
```

```
[shading] shading=phong: FPS: 360 FrameTime: 2.778 ms
[shading] shading=cel: FPS: 359 FrameTime: 2.786 ms
[bump] bump-render=high-poly: FPS: 360 FrameTime: 2.778 ms
[bump] bump-render=normals: FPS: 356 FrameTime: 2.809 ms
[bump] bump-render=height: FPS: 355 FrameTime: 2.817 ms
[effect2d] kernel=0,1,0;1,-4,1;0,1,0;: FPS: 355 FrameTime: 2.817 ms
[effect2d] kernel=1,1,1,1,1;1,1,1,1,1;1,1,1,1,1;: FPS: 347 FrameTime: 2.882 ms
[pulsar] light=false:quads=5:texture=false: FPS: 350 FrameTime: 2.857 ms
[desktop] blur-radius=5:effect=blur:passes=1:separable=true:windows=4: FPS: 285 FrameTime:
3.509 ms
[desktop] effect=shadow:windows=4: FPS: 288 FrameTime: 3.472 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-
method=map: FPS: 258 FrameTime: 3.876 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-
method=subdata: FPS: 299 FrameTime: 3.344 ms
[buffer] columns=200:interleave=true:update-dispersion=0.9:update-fraction=0.5:update-
method=map: FPS: 281 FrameTime: 3.559 ms
[ideas] speed=duration: FPS: 344 FrameTime: 2.907 ms
[jellyfish] <default>: FPS: 342 FrameTime: 2.924 ms
[terrain] <default>: FPS: 210 FrameTime: 4.762 ms
[shadow] <default>: FPS: 348 FrameTime: 2.874 ms
[refract] <default>: FPS: 311 FrameTime: 3.215 ms
[conditionals] fragment-steps=0:vertex-steps=0: FPS: 355 FrameTime: 2.817 ms
[conditionals] fragment-steps=5:vertex-steps=0: FPS: 353 FrameTime: 2.833 ms
[conditionals] fragment-steps=0:vertex-steps=5: FPS: 357 FrameTime: 2.801 ms
[function] fragment-complexity=low:fragment-steps=5: FPS: 355 FrameTime: 2.817 ms
[function] fragment-complexity=medium:fragment-steps=5: FPS: 353 FrameTime: 2.833 ms
[loop] fragment-loop=false:fragment-steps=5:vertex-steps=5: FPS: 354 FrameTime: 2.825 ms
[loop] fragment-steps=5:fragment-uniform=false:vertex-steps=5: FPS: 353 FrameTime: 2.833 ms
[loop] fragment-steps=5:fragment-uniform=true:vertex-steps=5: FPS: 351 FrameTime: 2.849 ms
=======================================================
                                  glmark2 Score: 338
=======================================================
```

Сводная таблица результатов тестирования укорителей выглядит следующим образом:

Таблица 1. Сравнение результатов тестирования двух GPU на Lenovo Ipeadap 3 Gaming

Наименование теста	glmark2 Score
AMD RENOIR (DRM 3.40.0, 13.1-RELEASE, LLVM 13.0.1) - «безголовый»	5488
AMD RENOIR (DRM 3.40.0, 13.1-RELEASE, LLVM 13.0.1) - с отображением на экран	3614
NVIDIA GeForce RTX 3050 Laptop GPU/PCIe/SSE2 (NVIDIA 515.57) - «безголовый»	22374
NVIDIA GeForce RTX 3050 Laptop GPU/PCIe/SSE2 (NVIDIA 515.57) - с передачей и отображением изображение на основной GPU	338

Из таблицы видно, что NVIDIA GeForce в «безголовом» варианте выигрывает у AMD RENOIR в 4 раза (22374 единицы против 5488 единиц), что не удивительно. Но при передачи результата рендеринга и отображении его на основном видеоадаптере производительность падает до 338 единиц — всё упирается в производительность оперативной памяти на конкретной машине. Очевидно, что использовать дискретный ускоритель от NVIDIA в «гибридном режиме» на данном ноутбуке как основное средство отображения 3D графики, в том числе при работе в различных САПР, смысла нет — работать будет не шибко быстро. Но вот как фоновая «числодробилка» данный 3D акселератор весьма не плох и он будет полезен

для таких приложений как например **Blender** для пакетного рендеринга, а так же во **FreeCAD** при расчетах в FEM Workbench. Что-ж, будем использовать с умом и по назначению!

Для того, чтобы оценить текущую загрузку AMD GPU в репозитории с пакетами имеется утилита **radeontop**. Установим её, запустим и посмотрим как выглядит нагрузка при выполнении теста **glmark2** в безголовом режиме на AMD GPU:

```
rz@butterfly:~ % sudo pkg inst --yes radeontop
rz@butterfly:~ % glmark2 --fullscreen --off-screen > /dev/null &
[1] 25617
rz@butterfly:~ % radeontop
```

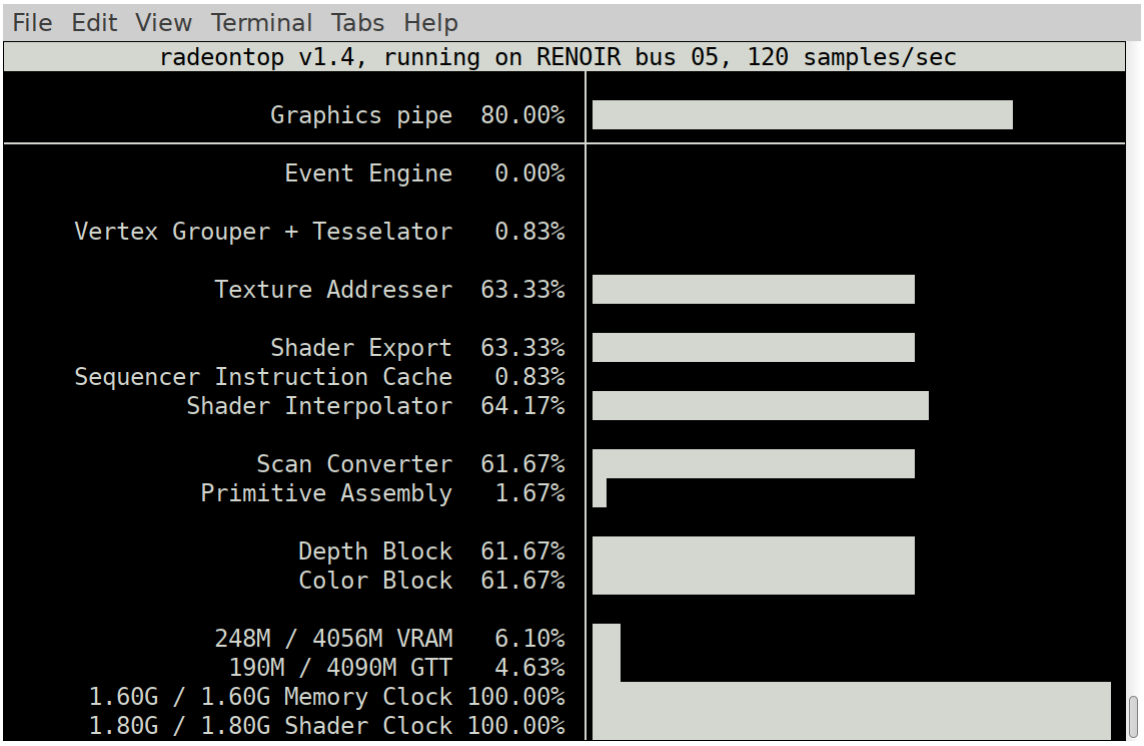

Рис. 3. Мониторинг нагрузки на AMD GPU утилитой radeontop.

Аналогичная утилита для NVIDIA GeForce устанавливается вместе с пакетом проприетарных драйверов и называется **nvidia-smi**. Запустив её будем наблюдать следующую картину:

```
rz@butterfly:~ % nvidia-smi -l
```

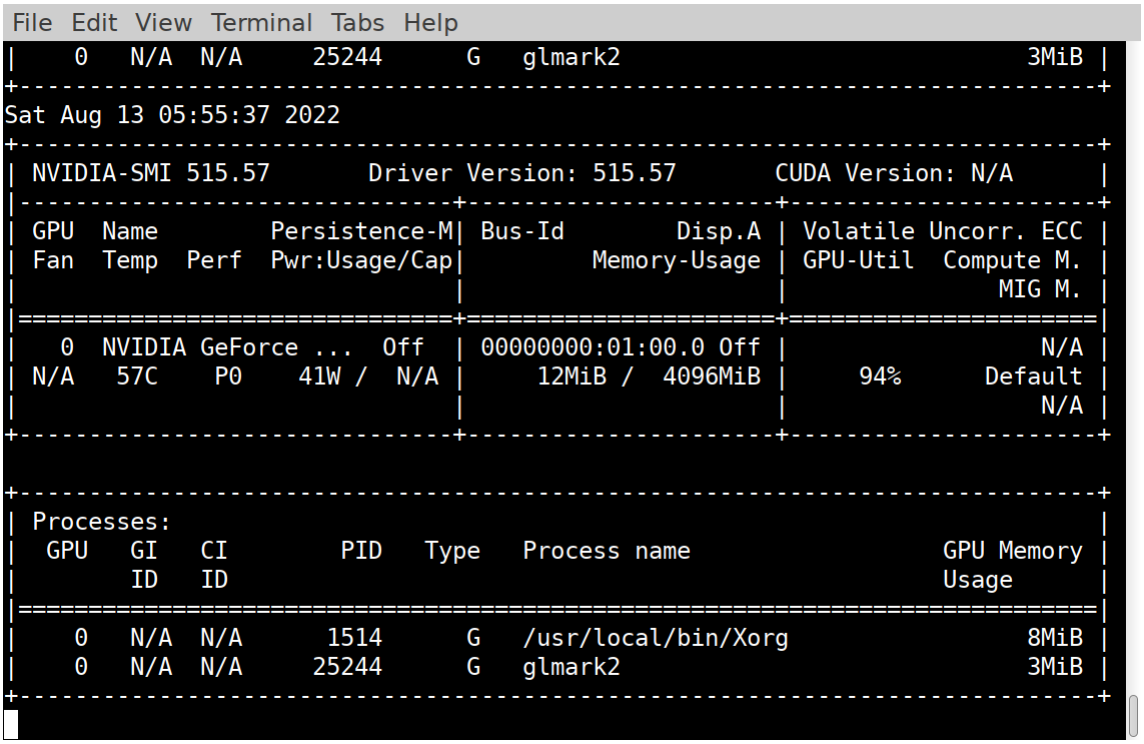

Рис. 4. Мониторинг нагрузки на NVIDIA GPU утилитой nvidia-smi.

Замечание:

Выяснилось, что драйвер для GPU NVIDIA до сих пор не поддерживает нормальную работу suspend/resume — попытка «усыпить» систему в процессе работы NVIDIA GPU приводит либо к крэшу системы, либо к её тотальному зависанию. В то же время у драйверов от AMD такой проблемы не выявилось — AMD GPU засыпает и пробуждается исправно. Поэтому следует обращать внимание на то, с каким из двух акселераторов работают Ваши приложения перед тем, как закрыть крышку ноутбука.

25. Настройка web-камеры

В эпоху когда всё большее число работников сферы «АйТи» переходят на «удаленку», а студенты на «дистанционку», работать в системе не позволяющей организовать видеоконференцсвязь становится неприемлемо и неприлично, поэтому далее я расскажу как в ОС FreeBSD настроить поддержку встроенной в ноутбук web-камеры. Для любопытствующих скажу сразу — Zoom из браузера Firefox на FreeBSD работает! Более подробную информация о работоспособности популярных «он-лайн» средств ВКС на ОС FreeBSD можно почерпнуть тут: https://freebsdfoundation.org/blog/status-of-online-conference-software-on-freebsd/

В ОС FreeBSD имеется возможность эмулировать популярный в Linux API под названием Video4Linux 2 (V4L2). Осуществляется это с помощью демона **webcamd** — процесс который запускается в фоне и предоставляет доступ к большому спектру поддерживаемых USB камера через V4L2 API. По сути, **webcamd** представляет собой коллекцию «камерных» драйверов позаимствованную из ядра Linux и оформленную в виде единого «user-space» драйвера, а следовательно для работы **webcamd** требуется поддержка CUSE. CUSE - драйвер-обертка позволяющий реализовывать символьные устройства в «user space», а не в ядре ОС, что сильно облегчает разработку простых драйверов устройств. Поддержка CUSE в ОС FreeBSD осуществляется драйвером **cuse**, который либо внедряется статически (вкомпилируется) в ядро ОС, либо загружается в виде модуля.

Настройка CUSE конвенциальна - требуется добавить модуль **cuse** в список загружаемых в переменную **kld_list** в файле **/etc/rc.conf**. Как обычно воспользуемся для этого утилитой **sysrc**:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf kld_list+="cuse"
kld_list: if_iwm amdgpu nvidia-modeset ig4 iichid -> if_iwm amdgpu nvidia-modeset ig4 iichid
cuse
```

и вручную загрузим этот модуль:

```
rz@butterfly:~ % sudo kldload cuse
```

Далее установим из пакетов **webcamd**:

```
rz@butterfly:~ % sudo pkg inst --yes webcamd
```

и добавим его запуск в системный конфигурационный файл **/etc/rc.conf**:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf webcamd_enable="YES"
webcamd_enable:  -> YES
```

Но это еще не всё! Для того, чтобы **webcamd** опознал камеру, необходимо сообщить ему имя камеры через параметр **-N**. Как правило все встроенные web-камеры в современных ноутбуках являются USB устройствами. Поэтому имя камеры не сложно узнать с помощью утилиты **usbconfig**:

```
rz@butterfly:~ % sudo usbconfig dump_device_desc | grep -i camera
ugen0.2: <Azurewave Integrated Camera> at usbus0, cfg=0 md=HOST spd=HIGH (480Mbps) pwr=ON
(500mA)
  iProduct = 0x0001  <Integrated Camera>
```

Еще один способ — запустить сам **webcamd** с параметром **-l** и он сообщит нам список доступных устройств:

```
rz@butterfly:~ % sudo webcamd -l
Available device(s):
Show webcamd usage:
webcamd -h
webcamd 25325 - - webcamd: No USB device match found
rz@butterfly:~ % sudo webcamd
Available device(s):
webcamd [-d ugen1.1] -N AMD-XHCI-root-HUB -S unknown -M 0
webcamd [-d ugen0.1] -N AMD-XHCI-root-HUB -S unknown -M 1
webcamd [-d ugen1.2] -N Logitech-USB-Receiver -S unknown -M 0
webcamd [-d ugen0.2] -N Azurewave-Integrated-Camera -S unknown -M 0
webcamd [-d ugen1.3] -N ITE-Tech--Inc--ITE-Device-8176 -S unknown -M 0
webcamd [-d ugen0.3] -N vendor-0x8087-product-0x0a2a -S unknown -M 0
Show webcamd usage:
webcamd -h
```

Находим в списке устройств камеру и копируем её название. Тут нужно обратить внимание на то, что все пробелы в названии камеры должно быть заменены на символ дефиса (-). В моём случае команда для установки параметров для **webcamd** выглядит следующим образом:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf webcamd_flags="-N Azurewave-Integrated-Camera"
webcamd_flags:  -> -N Azurewave-Integrated-Camera
```

Осталось запустить демона командой:

```
rz@butterfly:~ % sudo service webcamd start
Starting webcamd.
```

Важно замечание: выяснилось, что при некоторых обстоятельствах демон **webcamd** отказывается стартовать через стандартных механизм сервисов (через утилиту **service** как в команде выше), в этом случае придется перезагрузить систему.

В случае успешного запуска демона **webcamd** в фоне появится процесс вида:

```
rz@butterfly:~ % ps auxwww | grep webcam
root        727    0.0  0.0    20972   7672  -  I<s  04:45   0:00.00 /usr/local/sbin/webcamd
-i 0 -d ugen0.2 -N Azurewave-Integrated-Camera -B -U webcamd -G webcamd
```

А в файловой системе **/dev** появится одно или несколько символьных устройств V4L2:

```
rz@butterfly:~ % ll /dev/video*
crw-rw----  1 webcamd  webcamd  0x18c Aug 18 04:45 /dev/video0
crw-rw----  1 webcamd  webcamd  0x18d Aug 18 04:45 /dev/video1
```

Обратите внимание на то, что данные файлы устройств принадлежат группе **webcamd** и не доступны для чтения/записи всем остальным пользователям, а это значит, что пользователя который будет работать с камерой необходимо добавить в эту группу. Вот как я добавляю себя (пользователь **rz**):

```
rz@butterfly:~ % pw groupmod webcamd -m rz
```

Проверить работоспособность камеры можно старым добрым VLC плейером. Установим VLC:

```
rz@butterfly:~ % sudo pkg inst --yes vlc
```

Запустим VLC и понаблюдаем себя родимого, не забыв передвинуть шторку камеры:

```
rz@butterfly:~ % vlc v4l2:///dev/video0
VLC media player 3.0.17.4 Vetinari (revision 3.0.13-8-g41878ff4f2)
[0000000800c5d060] main libvlc: Running vlc with the default interface. Use 'cvlc' to use vlc
without interface.
```

26. Установка automount для съёмных носителей и настройка поддержки кириллицы

Любому пользователю ПК постоянно приходится иметь дело со съёмными носителями информации — всякого рода «USB флэшки» и SD/mmc карты. Во FreeBSD, как и в любой UNIX-подобной системе, файловые системы должны быть примонтированы до использования и корректно отмонтированы после использования, для чего в операционной системе присутствует две утилиты **mount** и **umount**. Для большинства пользователей UNIX систем вызов утилиты **mount** при подключении к машине нового носителя не является чем-то экстраординарным, так же как и обязательный вызов утилиты **umount** для корректного закрытия файловой системы перед отключением носителя. Тем не менее, у новых и неопытных пользователей данная последовательность действий может создать проблемы. Одна из сложностей состоит в том, что набор параметров для команды **mount** зависит от типа

устройства и файловой системы развернутой на нём, что не всегда очевидно для неопытного глаза.

Для того, чтобы упростить работу со съёмными носителями, в ОС FreeBSD имеется специальное средство — **automout**. В своей сути **automount** это небольшой shell-скрипт который можно вызывать как вручную, так и автоматически из **devd** — системного демона, который мониторит появление (attach) и исчезновение (detach) устройств в системе и вызывает внешний процесс для обработки таких событий. Таким образом, демону **devd** можно задать набор нехитрых правил, чтобы он вызывал скрип **automount** каждый раз при подключении или отключении съёмного носителя.

Скрипт **automount** можно установить из портов (**sysutils/automount**) или пакетов, вместе с ним установится и необходимый набор правила для **devd**. Вот как это можно сделать с помощью утилиты **pkg**:

```
rz@butterfly:~ % pkg search automount
automount-1.7.8                FreeBSD's devd(8) based automount solution
…

rz@butterfly:~ % sudo pkg inst --yes automount
```

Вместе с **automount** так же будут установлены модули поддержки различных эзотерических файловых систем, а так же модуль **fusefs** через который реализуется поддержка этих «не родных» файловых систем. На данный момент **automount** поддерживает следующие файловые системы: NTFS/FAT/exFAT/EXT2/EXT3/EXT4/UFS/HFS/XFS/ISO9660, но не все из них доступны для записи.

После установки, модуль **fusefs** следует добавить в список **kld_list** в файле **/etc/rc.conf**, как обычно:

```
rz@butterfly:~ % sudo sysrc -f /etc/rc.conf kld_list+="fusefs"
kld_list: if_iwm amdgpu nvidia-modeset cuse ig4 iichid -> if_iwm amdgpu nvidia-modeset cuse
ig4 iichid fusefs
```

Далее следует перезапустить демона **devd** для того, чтобы он подгрузил новый набор правил:

```
rz@butterfly:~ % sudo service devd restart
Stopping devd.
Waiting for PIDS: 19556.
Starting devd.
```

В общем-то, можно приступать к использованию, но не спешите вставлять свою USB флэшку если файловая система на ней был создана в ОС Windows. С подачи небезызвестной фирмы из Редмонда, в файловых системах типа FAT и FAT32 используется кодировка кириллических символов либо CP866 (MS-DOS), либо CP1251 (Windows). Поэтому, если сейчас примонтировать такое устройство, то вместо кириллических символов в именах файлов будут символы знака вопрос (?), потому, что утилита **mount_msdosfs**, которая в конченом счете монтирует носитель с FAT/FAT32 ничего не знает о том, какая на этом носителе использована кодировка и все символы с кодом выше 127 будут заменяться системой на знаки вопроса.

Чтобы исправить это недоразумение, утилите **mount_msdosfs** необходимо передавать два параметра: **-D** и **-L**. Вот, что по этому поводу указано в системном руководстве:

```
rz@butterfly:~ % man mount_msdosfs
MOUNT_MSDOSFS(8)        FreeBSD System Manager's Manual       MOUNT_MSDOSFS(8)

NAME
     mount_msdosfs - mount an MS-DOS file system

SYNOPSIS
     mount_msdosfs [-9ls] [-D DOS_codepage] [-g gid] [-L locale] [-M mask]
                   [-m mask] [-o options] [-u uid] [-W table] special node

DESCRIPTION
     The mount_msdosfs utility attaches the MS-DOS file system residing on the
     device special to the global file system namespace at the location
     indicated by node.  This command is normally executed by mount(8) at boot
     time, but can be used by any user to mount an MS-DOS file system on any
     directory that they own (provided, of course, that they have appropriate
     access to the device that contains the file system).

     The options are as follows:

…

     -L locale
             Specify locale name used for file name conversions for DOS and
             Win'95 names.  By default ISO 8859-1 assumed as local character
             set.

     -D DOS_codepage
             Specify the MS-DOS code page (aka IBM/OEM code page) name used
             for file name conversions for DOS names.

     -W table
             This option is preserved for backward compatibility purpose only,
             and will be removed in the future.  Please avoid using this
             option.

             Specify text file name with conversion table: iso22dos, iso72dos,
             koi2dos, koi8u2dos.

EXAMPLES
     To mount a Russian MS-DOS file system located in /dev/ada1s1:

           mount_msdosfs -L ru_RU.KOI8-R -D CP866 /dev/ada1s1 /mnt

     To mount a Japanese MS-DOS file system located in /dev/ada1s1:

           mount_msdosfs -L ja_JP.eucJP -D CP932 /dev/ada1s1 /mnt
```

В руководстве выше показан пример набор параметров для файловых систем созданных в MS-DOS и с использованием локальной кодировки KOI8-R для UNIX систем, которая давно вышла из употребления. Для файловых систем созданных в ОС Windows необходимо указать кодовую страницу **cp1251**, а в качестве локальной кодировки — **ru_RU.UTF-8**.

Но как же передать эти параметры в утилиту **mount_msdosfs**, которая вызывается при автоматическом монтировании из **automount** ? Оказывается, создатель скрипта **automount** позаботился об этом и сделал так, что утилита **automount** при запуске считывает конфигурационный файл **/usr/local/etc/automount.conf** в котором могу присутствовать различные опции, в том числе интересующие нас опции **FAT_ENCODING** и **FAT_CODEPAGE**.

Установим их значение используя утилиту **sysrc**:

```
rz@butterfly:~ % sudo sysrc -f /usr/local/etc/automount.conf FAT_CODEPAGE="cp1251"
FAT_ENCODING="ru_RU.UTF-8"
```

```
FAT_CODEPAGE:  -> cp1251
FAT_ENCODING:  -> ru_RU.UTF-8
```

Если заглянуть внутрь скрипта **/usr/local/sbin/automount**, то можно обнаружить много интересного, в том числе ряд полезных опций и комментарий к ним. Например:

```
NOTIFY (set to NO by default)
  Use 'notify-send' and 'libnotify' to show notifications
  of mounting and unmounting devices on the desktop.

  example: NOTIFY='YES'

WALL (set to NO by default)
  Use wall(1) to show notifications of mounting and
  unmounting devices on terminals of logged in users.

  example: WALL='YES'
```

Как следует из описания, эти опции позволяют получать нотификации от утилиты **automount** в графическую систему и в терминал (на «стену») при монтировании или отмонтировании устройств. Попробуем установить их:

```
rz@butterfly:~ % sudo sysrc -f /usr/local/etc/automount.conf NOTIFY="YES" WALL="YES"
NOTIFY:  -> YES
WALL:  -> YES
```

После таких изменений ничего перезапускать не надо - как я уже отметил, скрипт **automount** сам перечитает свой конфигурационный файл при запуске, а запускается он каждый раз из **devd** при подключении или отключении нового носителя.

Вот теперь вставим USB флешку и посмотрим, что же у нас получилось.

После подключения носителя, через секунду на нём начинаем мигать синий светодиод, что говорит нам о том, что с носителем идет обмен данными — т. е. **automount** запустился и пытается примонтировать файловые системы. Еще через несколько секунд на всех открытых у меня терминалах появляется сообщение:

```
Broadcast Message from rz@butterfly
        (no tty) at 6:10 +05...

automount: Device '/dev/da0' mounted on '/media/da0' directory.
```

- это **automount** сообщает, что новое устройство **/dev/da0** успешно примонтировано в подкаталог **/media/da0/**. Просмотрим его содержимое:

```
rz@butterfly:~ % ll /media/da0/
total 2400
drwxrwxr-x  1 root  wheel    16384 Jan  6  2020 AMIGA/
drwxrwxr-x  1 root  wheel    16384 Sep  1  2019 LOST.DIR/
drwxrwxr-x  1 root  wheel    16384 Oct 29  2015 System Volume Information/
-rwxrwxr-x  1 root  wheel  2403803 Oct 29  2015 Файл_для_примера.zip*
```

Видно, что файлы с кириллицей в именах отлично отображаются. Так же видно, что файлы доступны для записи только пользователю **root** и пользователям входящим в группу **wheel**. Так как пользователь **rz** находится в этой группе с самого начала инсталляции ОС, то у меня не должно возникнуть каких-то проблем с записью в файлы на этом носителе. Однако, иногда требуется предоставить такую возможность отдельной группе пользователей или

конкретному пользователю не входящиму в группу администраторов. Для это цели у скрипта **automount** есть еще ряд полезных опций:

```
MNT_GROUP (wheel by default)
  If set to some group name, the mount command will
  chown(1) the mount directory with the group.

  example: group='operator'

MNT_MODE (set to 775 by default)
  Value for chmod on mount point.

USER (root by default)
  If set to some username, the mount command will
  chown(1) the mount directory with the user and
  its primary user group. If used with FM option
  allows to launch the specified file manager after
  a successful mount.

  example: USER="vermaden"
```

Предлагаю читателю поэкспериментировать с ними самостоятельно.

Чтобы отмонтировать съёмное устройство нужно выдать команду **umount /media/da0**, но это не всегда удобно, так как требуется запомнить либо имя устройства (/dev/da0) либо точку монтирования (/media/da0). Гораздо проще выдать подряд две команды на синхронизацию дискового кеша: **sync && sync,** после чего устройство можно смело вынимать — процедуру демонтирования **automount** выполнит самостоятельно уже в отсутствии устройства. При таком способе изъятия носителя файловая система на нём не будет логически закрыта (битик «dirty» не будет снят), но и повреждений тоже не будет, так что смело пользуйтесь этим «лайф-хаком». Более того, на своей системе я забайндил эту команду на сочетание клавиш **Ctrl+Alt+S** через утилиту **xfce4-keyboard-settings** и выработал у себя привычку нажимать её каждый раз перед изъятием съёмного носителя.

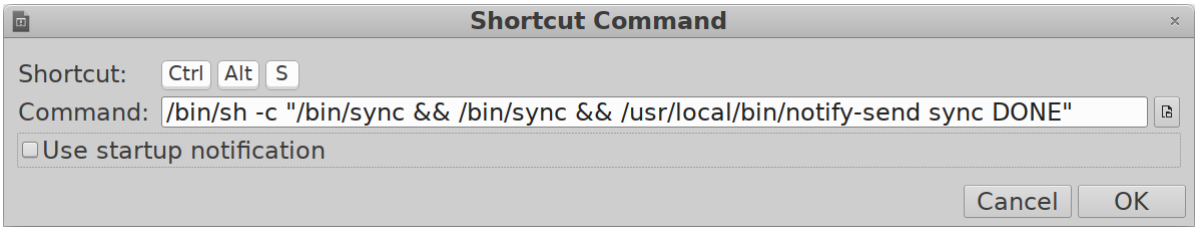

Рис. 5. Пример настройки «горячей клавиши» в xfce4-keyboard-settings.

Бывает, что при записи на носитель больших объёмов данных в дисковом кэше остается очень много несохраненной на диск информации. В этом случае выполнение команды **sync** занимает какое-то время - до нескольких десятков секунд при записи на медленный SD/mmc носитель. Приведенный выше «шорт-кат» выводит сообщение «sync DONE» только после того, как дисковый кэш полностью записан и носитель можно отключать.

На этом, пожалуй, я сделаю остановку в повествовании и предоставлю своей клавиатуре возможность немного охладиться.

27. Разное про FreeBSD и не только

Для тех, кто смог дочитать до этого места привожу немного прочей информации о проекте и ОС FreeBSD.

Про игровые консоли.

Под управлением ОС FreeBSD работают такие известные игровые приставки как Sony PlayStation и Nintento Switch.

Про RISC-V.

В 2016 году ОС FreeBSD была портирована на архитектуру RISC-V (RV64CG), работа была полностью выполнена нашим соотечественником Русланом Букиным, который ныне трудится в Кембриджском Университете (UK) и, на сколько я понимаю, продолжает тянуть на себе эту тяжелую ношу.

Про чертёнка «Beastie».

Справа приведено изображение талисмана всех операционных систем группы BSD. Зовут этого зверька с вилами «Beastie» - игра слов на английскую аббревиатуру Би-Эс-Ди (BSD). В простонародье его называют «BSD демоном» от анг. «daemon» - фоновый процесс в UNIX-подобных ОС. Авторские права на изображение принадлежат Маршаллу Кирку МакКьюзику. МакКузик стоял у истоков BSD и по сей день является активным коммиттером в код операционной системы FreeBSD.

Про проект The FreeBSD Project.

Проект The FreeBSD Project занимается администрированием процесса разработки ОС FreeBSD и её технической поддержкой. В отличие от ядра Linux, где всем заправляет единолично товарищ Торвальдс, проект FreeBSD управляется кругом единомышленников который называют «**core team**». Эта команда решает принципиальные задачи и определяет направление развития ОС. Всего в команде девять должностей (девять человек), которые являются выборными. Выборы проходят раз в два года согласно Уставу Проекта FreeBSD. На должности могут претендовать активные коммиттеры, а голосовать - все разработчики, которые внесли хоть какой-то вклад в проект (коммиттеры, бетатестеры, багрепортеры, люди занимающиеся поддержкой на форуме). Последние выборы проходили в мае 2022 года.

Про фонд The FreeBSD Foundation.

Некоммерческий фонд The FreeBSD Foundation занимается продвижением ОС FreeBSD в массы, взаимодействует с корпоративным сектором и финансирует тяжелые разработки в рамках проекта FreeBSD. Согласно отчету за 2021 год, основным источником дохода фонда являются частные пожертвования (более 90%). За весь период своего существования Фонд собрал чуть более 5 млн долл. США — сущие копейки по сравнению с тем, сколько зарабатывают дистрибьюторы Linux систем. На самом деле это не вся сумма которая была потрачена на разработку ОС FreeBSD, многие корпорации эксплуатанты ОС FreeBSD, такие как Apple, Netflix, Cisco, Juniper, NetApp, Sony и т.д, предпочитают делать выплаты непосредственно разработчикам за реализацию тех или иных задач. Ну и надо понимать, что как и в ядре Linux, большая часть кода ОС FreeBSD создается разработчиками не за деньги, а за зарплату. ;)

28. Основные сайты для посещения

1. https://www.freebsd.org/
2. https://forums.freebsd.org/
3. https://docs.freebsd.org/en/
4. https://bugs.freebsd.org/bugzilla/
5. https://freebsdfoundation.org/ <-- пожертвования тут!

Призываю всех дееспособных граждан поучаствовать в проекте FreeBSD по принципу «кто чем может» - бета-тестирование, баг-репорты, устранения багов, разработка драйверов, сопровождение портов, техническая поддержка и привлечение новых пользователей. Финансовые пожертвования тоже приветствуются.