Подсистема ввода-вывода

Курс «Операционные системы»

НГУ

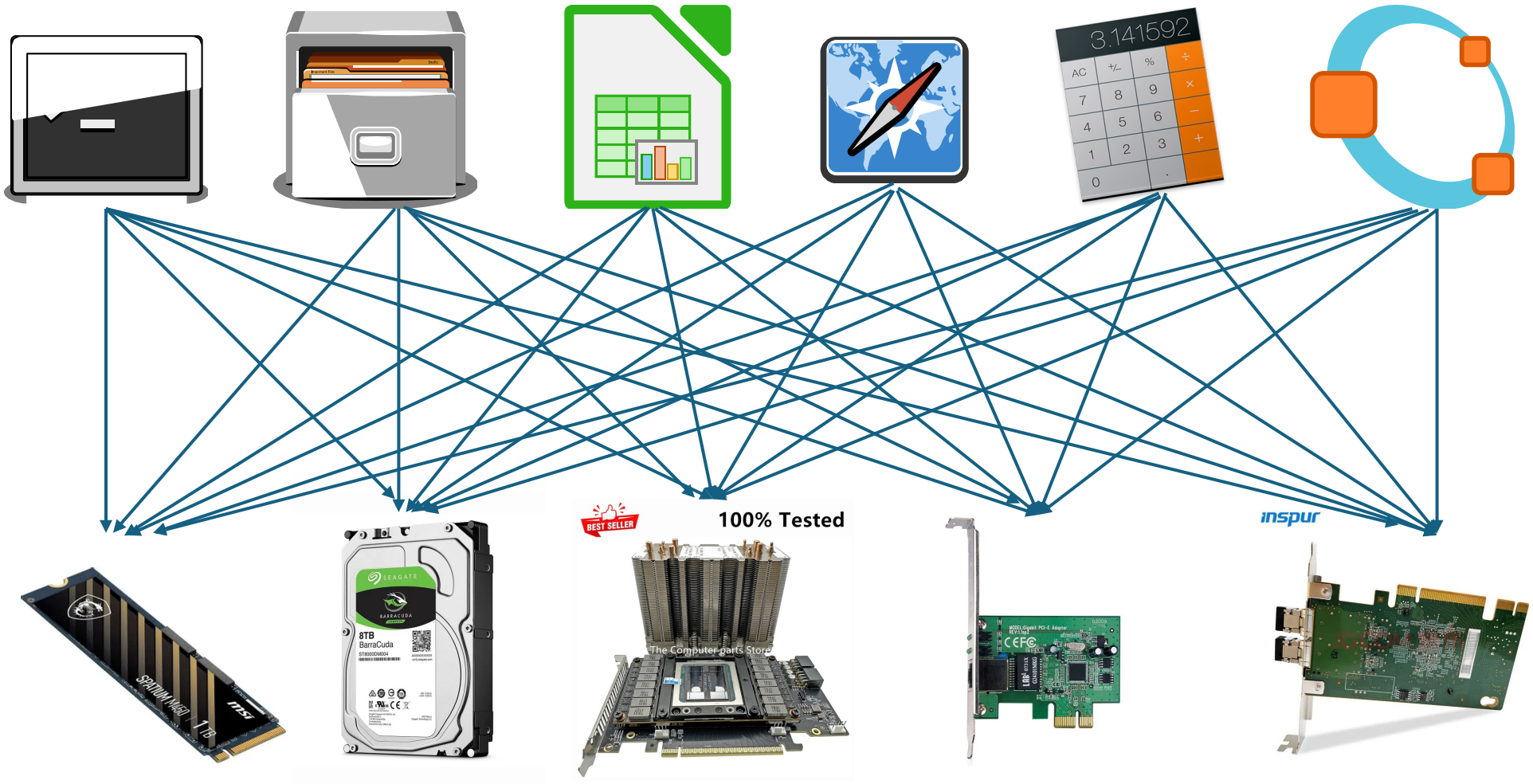
2026

Д.В. Иртегов

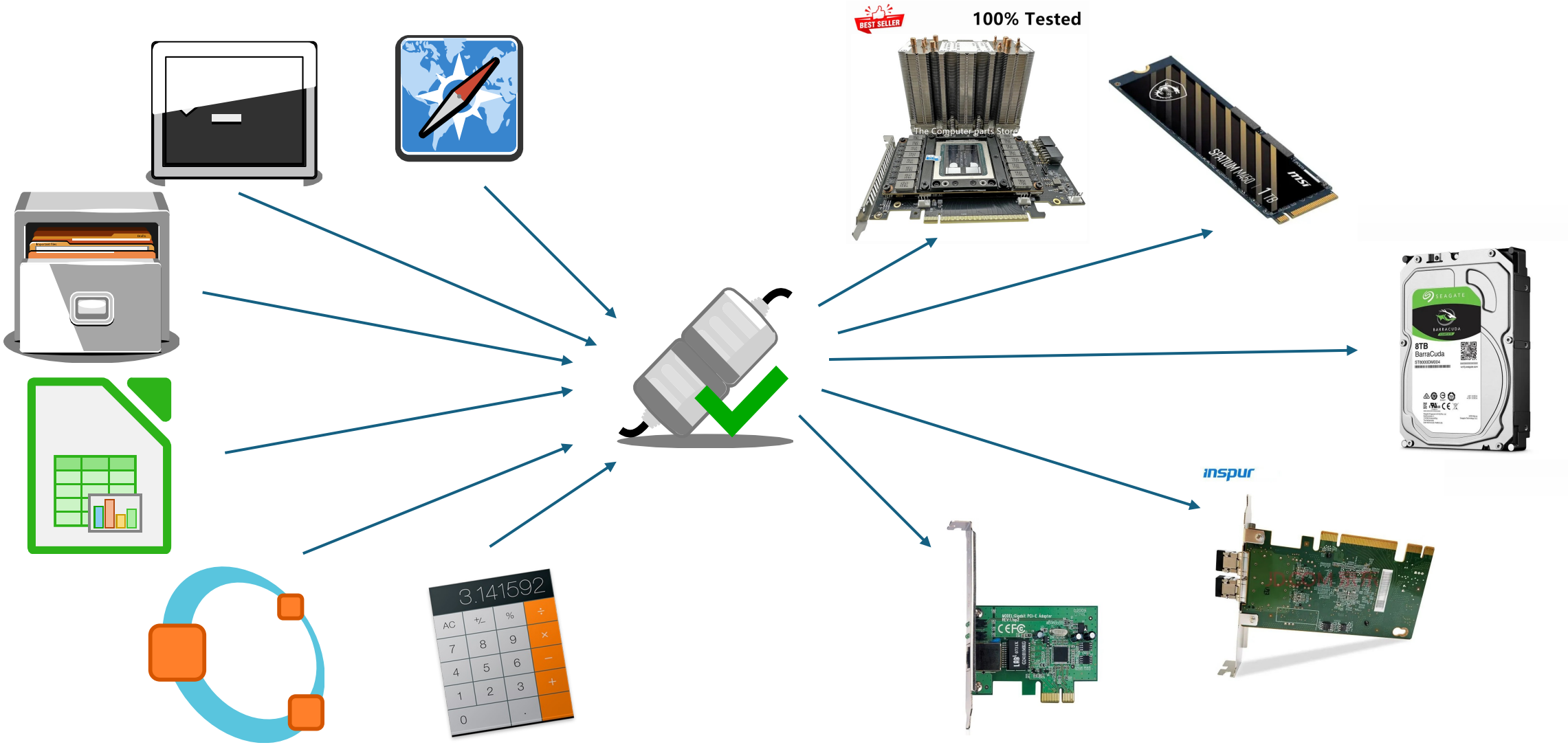
Что такое драйвер

- Программный компонент с определенным интерфейсом
- позволяет
  - ОС
  - загрузочному монитору
  - или прикладным программам
- взаимодействовать с чем-то еще (обычно с внешним устройством)
  - Слово «определенный» обычно означает
  - «документированный»
  - или даже «стандартный» (e.q. драйверы UEFI)

Зачем нужны драйверы



Зачем нужны драйверы



Зачем еще нужны драйверы

- ОС не пускает прикладные программы работать с оборудованием
  - Значит, ОС должна предоставлять интерфейс для этого
- Несколько программ могут захотеть работать с одним устройством
  - Драйвер должен обеспечить синхронизацию или сериализацию доступа

Как должен выглядеть интерфейс драйвера?

- Множество функциональных классов устройств
  - Видеоадаптеры
  - Принтеры
  - Сетевые адаптеры для различных протоколов канального уровня
  - Внешние носители
  - Звуковые адаптеры
  - Да тыщи их
- У каждого функционального класса могут быть свои операции
  - Видеоадаптер: установить видеорежим
  - Сетевой адаптер: привязать IP адрес
  - Принтер: проверить, есть ли бумага

Крайние варианты подхода

- Windows: свой интерфейс для каждого типа устройств
- 92 типа устройств
  https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/specifying-device-types

```
#define FILE_DEVICE_BEEP                    0x00000001
#define FILE_DEVICE_CD_ROM                  0x00000002
#define FILE_DEVICE_CD_ROM_FILE_SYSTEM      0x00000003
#define FILE_DEVICE_CONTROLLER              0x00000004
#define FILE_DEVICE_DATALINK                0x00000005
#define FILE_DEVICE_DFS                     0x00000006
#define FILE_DEVICE_DISK                    0x00000007
#define FILE_DEVICE_DISK_FILE_SYSTEM        0x00000008
#define FILE_DEVICE_FILE_SYSTEM             0x00000009
#define FILE_DEVICE_INPORT_PORT             0x0000000a
#define FILE_DEVICE_KEYBOARD                0x0000000b
#define FILE_DEVICE_MAILSLOT                0x0000000c
#define FILE_DEVICE_MIDI_IN                 0x0000000d
#define FILE_DEVICE_MIDI_OUT                0x0000000e
#define FILE_DEVICE_MOUSE                   0x0000000f
#define FILE_DEVICE_MULTI_UNC_PROVIDER      0x00000010
#define FILE_DEVICE_NAMED_PIPE              0x00000011
#define FILE_DEVICE_STORAGE_REPLICATION     0x00000055
#define FILE_DEVICE_TRUST_ENV               0x00000056
#define FILE_DEVICE_UCM                     0x00000057
#define FILE_DEVICE_UCMTCPCI                0x00000058
#define FILE_DEVICE_PERSISTENT_MEMORY       0x00000059
#define FILE_DEVICE_NVDIMM                  0x0000005a
#define FILE_DEVICE_HOLOGRAPHIC             0x0000005b
#define FILE_DEVICE_SDFXHCI                 0x0000005c
#define FILE_DEVICE_UCMUCSI                 0x0000005d
#define FILE_DEVICE_PRM                     0x0000005e
#define FILE_DEVICE_EVENT_COLLECTOR         0x0000005f
#define FILE_DEVICE_USB4                    0x00000060
#define FILE_DEVICE_SOUNDWIRE               0x00000061
#define FILE_DEVICE_FABRIC_NVME             0x00000062
#define FILE_DEVICE_SVM                     0x00000063
#define FILE_DEVICE_HARDWARE_ACCELERATOR    0x00000064
#define FILE_DEVICE_I3C                     0x00000065
```

Классический Unix

- Всего 2 типа устройств
  - Блочные (диски)
  - Символьные
  - Драйверы файловых систем вообще не считаются устройствами
- Блочные драйверы не доступны из userspace
  - Обращаться к ним могут только другие модули ядра
  - На практике, у каждого дискового устройства есть символьный интерфейс (выглядит как обычное символьное устройство)
- Символьные устройства:
  - read
  - write
  - ioctl

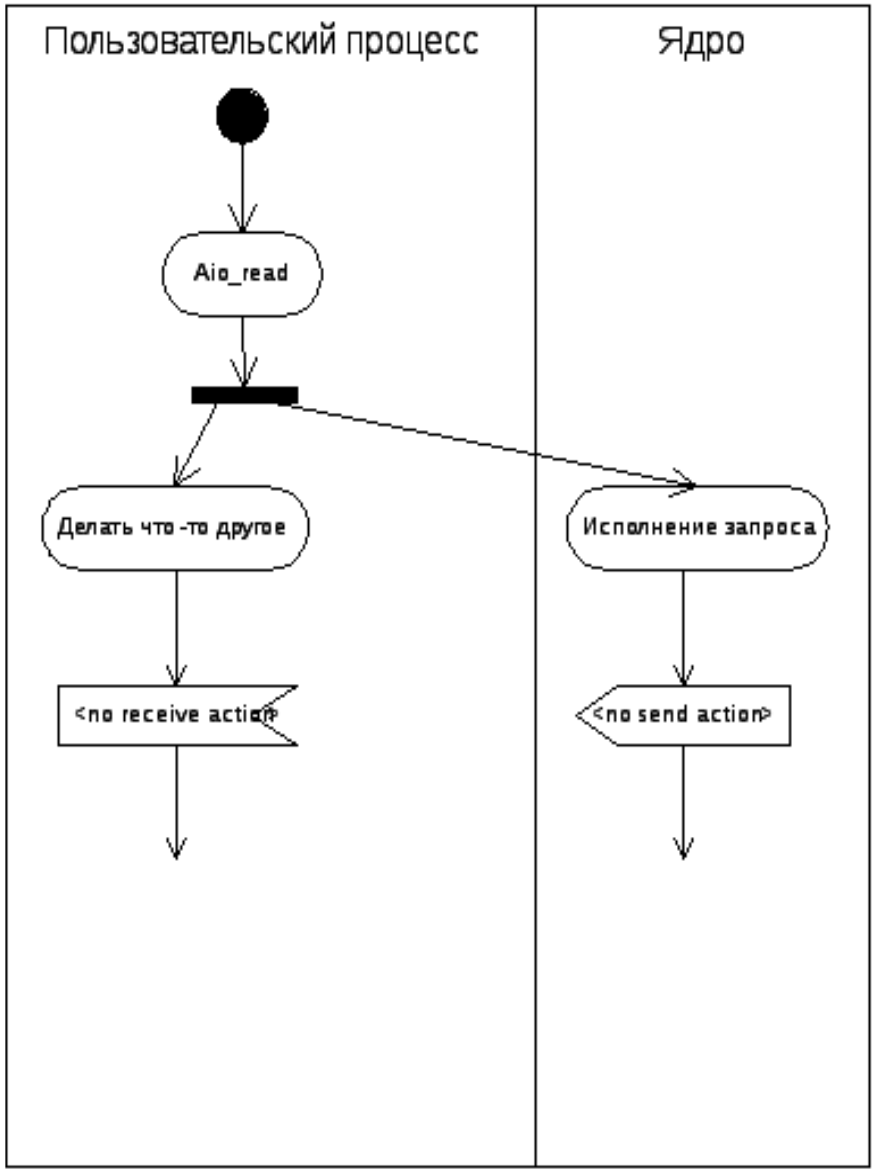
Современные Unix-системы

- Типов «устройств, которые не устройства» больше
  - Адаптеры SCSI, USB и других периферийных шин
  - В Linux - сетевые устройства
  - И др.
- Классы устройств по поддерживаемым командам ioctl
  - Терминалы
  - Звуковые адаптеры
  - Видеоадаптеры
  - Адаптеры WiFi
  - И др.
- У символьных устройств больше функций
  - lseek
  - mmap
  - Подписка на доступность для чтения/записи (для реализации select/poll)

Еще одна классификация интерфейсов

- Синхронные
  - Символьные устройства классического Unix и Linux
  - Функции драйвера исполняются в той же нити, которая делает запрос (например, в LWP, выполняющем системный вызов)
  - Под капотом могут делать асинхронные запросы, но дожидаются их завершения
- Асинхронные
  - Очереди запросов
    - Блочные устройства в классическом Unix, System V и Linux
    - STREAMS в System V
    - Драйверы SCSI, USB и других периферийных шин
    - Сетевые устройства в Linux
    - Все устройства в RSX-11 и VMS
    - Большинство устройств Windows
  - Потоковые порты обмена данными
    - QNX, Mach (современная MacOS)

Асинхронный ввод-вывод



Преимущества асинхронного ввода-вывода

- Большинство устройств медленные
- Но большинство работает асинхронно (в том числе с использованием DMA)
- Некоторые устройства - интерфейсы к чему-то еще более медленному (человек, сетевые сервисы)
- Можно решать эту проблему, создавая нити, но нить
  - Это дорого
  - Для них не гарантируется порядок выполнения
  - Возможны ошибки соревнования
- Асинхронный ввод-вывод похож на идиому Future

Отличия async I/O от select/poll

- Select/poll ждет готовности устройства
  - Применим только к некоторым устройствам: терминалы, трубы, сокеты
- Async I/O ждет завершения операции, а значит применим к устройствам, которые передают данные только по запросу (диски, регулярные файлы)
- Возможен callback: вместо вызова блокирующейся операции для ожидания можно зарегистрировать callback, который будет вызван при завершении (как сигнал или в отдельной нити)

Недостатки async I/O

- Непривычно, кажется что много писанины
- Во многих языках высокого уровня (Java, JavaScript, Python) не доступен или доступен как враппер над select/poll/event ports
- В Linux, многие устройства под капотом синхронные, поэтому async I/O с ними тоже реализован как враппер над select/poll
  - Проблемы с производительностью
- Много архаичных и нестандартных интерфейсов

Модели async I/O в Unix

- Архаичный: F_ASYNC + SIGIO
  (считается неудобным, не изучаем)
- Нестандартные: Solaris AIO, Linux Event ports
- Posix AIO
  (многие жалуются на производительность, особенно в Linux)