Linux Block Layer Diagram
Схема дисковой системы Linux
https://www.thomas-krenn.com/de/wikiDE/images/e/e0/Linux-storage-stack-diagram_v4.10.png
https://www.thomas-krenn.com/en/wiki/Linux_Storage_Stack_Diagram

Virtual File System (VFS)

Системные вызовы read, write, open…
Приведите примеры команд которая генерирует системный вызов read?
Пример strace dd .., strace cat
В тестах например iozone, fio - там тесты называются по имени системного вызова. Нужно для понимания того что вы тестируете.

VFS находится транслирует системные вызовы в обращения к реальным физическим дискам, через слой блочных устройств.

Драйвера устройств

Контроллеры дисков

Находим адаптеры дисков

```
lspci
lspci -v -vv -vvv # more details
lspci -tv  # treelike structure which reflects the actual physical
structure of the PCI buses
lspci -D # shows domain
lspci -nn  # show vendor ID
lspci -nnA
lspci -v -s 00:1f.2  # show info about only one device
```

0000:00: 1f.2
При подключении PCI устройства ядро присваивает устройству четыре числа.
Каждая шина может содержать до 32 устройств, а устройство PCI может иметь до восьми функций. Местоположение устройства определяется

- 16-битным номером домена,
- 8-битным номером шины, максимально 256 шин
- 5-битным номером устройства, до 32 устройств
- 3-битным номером функции; максимум 8 функций

последние три цифры обычно называются **BDF** или **B/D/F (bus/device/function)** (шина / устройство / функции).
Спецификация PCI позволяет системе размещать до 256 шин, ненулевые доменные номера используются только для группировки шин PCI в очень больших системах.
Пример:
00:1f.2 SATA controller: Intel Corporation 6 Series/C200 Series Chipset Family 6 port Desktop SATA AHCI Controller (rev 04) (prog-if 01 [AHCI 1.0]

https://diego.assencio.com/?index=649b7a71b35fc7ad41e03b6d0e825f07

Загрузка драйверов

udevadm примеры

Пример с подключением диска. Как добавляется в систему.

```
journalctl -k -f
```

```
udevadm monitor
udevadm info --query=path --name=/dev/sdc
udevadm info /dev/sdc
```

Перечитать диски можно так же
```
echo 0 0 0 | tee /sys/class/scsi_host/host*/scan
```

Пример добавляем свое правило в udev.
где находятся файлы с описанием правил правил можно найти здесь `man 7 udev`

- system rules directory /usr/lib/udev/rules.d,
- volatile runtime directory /run/udev/rules.d
- local administration directory /etc/udev/rules.d.

```
ls /usr/lib/udev/rules.d/
udevadm info -n /dev/sda
# cat /etc/udev/rules.d/69-disk.rules
ACTION=="add", KERNEL=="sd[a-z]",
ENV{ID_SERIAL_SHORT}=="VBca5e42b9-32e027c5",
SYMLINK+="my_virtual_drive"
udevadm test $(udevadm info --query=path --name=/dev/sda)
udevadm control --reload-rules && udevadm trigger
udevadm control --reload
sudo udevadm trigger --action=add
```

Примеры именования блочных устройств
```
cat /usr/lib/udev/rules.d/60-persistent-storage.rules
```

Дополнительные примеры для настройки системы udev
https://sites.google.com/site/itmyshare/system-admin-tips-and-tools/udevadm---useage-examples

Блочные устройства

Имена дисков

Файлы устройств содержат данные, необходимые операционной системе для взаимодействия с физическими устройствами, такими как диски и дисководы и т. п. Фактически, специальные файлы устройств являются указателями на драйверы устройств, и когда процесс обращается к файлу устройств, он по сути работает с драйвером этого устройства

Есть два типа файлов устройств:

- блочные (block special files)
- символьные (character special files).

Блочные файлы устройств используются для передачи данных, разделённых на пакеты фиксированной длины — блоки.
символьные файлы устройств используются для не буферизованного обмена данными
`cat /proc/devices`

Метаданные файла содержат 3 специальных поля: **класс устройства**, **старший номер устройства** и **младший номер устройства**.
Класс устройства сообщает символьное устройство или блочное.

```
# ls -l /dev/sda1
brw-rw----. 1 root disk 8, 1 Jan 28 20:03 /dev/sda1
```

Пример: обращение к файлу /dev/sda1 (блочное устройство со старшим номером 8 и младшим номером 1) будет адресовано на 1-й раздел жесткого диска SATA
Файлы устройств размещаются в каталоге **/dev**

При попытке обращения к специальному файлу ядро переадресует обращение через нужный драйвер на устройство в соответствии со значением полей.

Команды для поиска информации о дисках

```
lsblk
blkid
```

Примеры устройств NVME

- /dev/nvme0n1 - device 1 on controller 0, the first discovered device on the first discovered controller.
- /dev/nvme0n1p1 - partition 1 on device 1 on controller 0.
- /dev/nvme2n5 - device 5 on controller 2, the fifth discovered device on the third discovered controller.
- /dev/sr0 - optical disc drive 0, the first discovered optical disc drive.
- /dev/sr4 - optical disc drive 4, the fifth discovered optical disc drive.
- /dev/cdrom - a symbolic link to /dev/sr0.

NVMe drive (Non-Volatile Memory Express)

Non-Volatile Memory express (энергонезависимая память)

```
cat /proc/partitions |grep -e nvme -e major
ls /sys/block/|grep nvme
lsmod|grep nvme
```

Фишки NVMe

- Namespaces
- NVMe over Fabrics
- NVMe over Fabrics Using TCP

NVMe предоставляют функцию пространства имен. (namespaces) Namespace это часть энергонезависимой памяти которое может быть отформатировано логическими блоками. Можно определить больше одного пространства имен. Каждое может иметь свой размер блока (512 байт, 4 килобайта). На хосте каждое пространство имен видно как отдельное физическое устройство.

```
Посмотрим сколько namespaces определено в системе
yum search nvme
yum install nvme-cli
nvme list  # покажет все устройства
nvme list-ns /dev/nvme0 # покажет сколько ns определено на устройстве

namespace - позволяет разделить диск на логический части (например
установить размер блока) (аналог разделов)
можно управлять namespaces удалять, добавлять, но в VM это не работает
```

- NVMe over Fabrics - можно подключать диски на удаленный хост через (Infiniband, Fibre Channel, Ethernet)
- начиная с ядра >5 можно подключать диски на удаленный хост через TCP

Настройки осуществляются утилитой nvme, например
`nvme discovery` - поиск устройства в сети

Дополнительная информация
https://www.linuxjournal.com/tag/nvme
https://github.com/linux-nvme/nvme-cli
https://www.thomas-krenn.com/en/wiki/Linux_Multi-Queue_Block_IO_Queueing_Mechanism_(blk-mq)
Namespaces explanation
this NVM Express tutorial presentation

Эмуляция NVME средствами ядра
https://unix.stackexchange.com/questions/336365/use-a-file-to-emulate-nvme-device

Эмулируем NVME раздачу по сети
https://www.linuxjournal.com/content/data-flash-part-ii-using-nvme-drives-and-creating-nvme-over-fabrics-network
https://community.mellanox.com/s/article/howto-configure-soft-roce

Утилиты для администрирования дисков

- hdparam
- sdparm (SCSI)
- smartctl

hdparm - позволяет получить/установить параметры устройств SATA/IDE и тестировать производительность
Параметры

- кэши дисков (drive caches)
- режим сна (sleep mode)
- управлять питанием (power management)
- уровнем шума (acoustic management)
- настройки DMA (Direct Memory Access)

Пример запуска утилиты hdparm для теста производительности

```
hdparm -Tt --direct /dev/nvme1n1

/dev/nvme1n1:
 Timing O_DIRECT cached reads:   2688 MB in  2.00 seconds = 1345.24
MB/sec
 Timing O_DIRECT disk reads: 4672 MB in  3.00 seconds = 1557.00 MB/sec

[root@4 www]# hdparm -Tt /dev/nvme1n1

/dev/nvme1n1:
 Timing cached reads:   18850 MB in  1.99 seconds = 9452.39 MB/sec
 Timing buffered disk reads: 4156 MB in  3.00 seconds = 1385.08 MB/sec
```

Пример отключения энергосбережения

```
ACTION=="add", SUBSYSTEM=="block", KERNEL=="sda",
RUN+="/usr/bin/hdparm -B 254 -S 0 /dev/sda"
```

http://gentoo.theserverside.ru/book/secrets_of_dev.html

Планировщики ввода/вывода (I/O Schedulers)

I/O Scheduling — общее название метода управления очередью операций ввода-вывода к жесткому диску и планировании данных операций компьютерных операционных систем.
Зачем знать ? Можно повысить производительность системы.

- Приоритизация некоторых запросов ввода-вывода
- Увеличение производительности дисков
- Гарантия исполнения важных запросов в кратчайшие сроки

Способы

- слияние запросов. Суть его заключается в объединении нескольких запросов к физическим разделам жесткого диска в один и тем самым уменьшении операций ввода-вывода.
- установка приоритет выполнения - очень старые запросы имеют высший приоритет выполнения перед вновь поступившими

Non-multiqueue I/O schedulers
NOTE: Non-multiqueue have been deprecated

- noop
- dealine
- cfq

Смена планировщика

```
/sys/block/<device>/queue/iosched
echo SCHEDNAME > /sys/block/DEV/queue/scheduler
# cat /sys/block/sda/queue/scheduler
[mq-deadline] kyber bfq none
# echo none >/sys/block/sda/queue/scheduler
# cat /sys/block/sda/queue/scheduler
[none] mq-deadline kyber bfq
```

Что выбрать
SSD или NVME диски

При использовании SSD с несколькими очередями или устройств NVME существует небольшая разница в пропускной способности между планировщиками ввода-вывода mq-deadline / none / bfq. В этих случаях может быть предпочтительнее использовать планировщик ввода-вывода **none** для уменьшения нагрузки на процессор.

Жесткий диск

Избегайте использования планировщиков ввода / вывода **none / noop** для жесткого диска, поскольку запросы сортировки по адресам блоков уменьшают задержки времени поиска, и ни один из этих планировщиков ввода / вывода не поддерживает эту функцию.
Показано, что **mq-deadline** выгоден для более требовательных операций ввода-вывода, связанных с сервером, однако пользователи настольных компьютеров могут

поэкспериментировать **с bfq,** поскольку было показано, что некоторые приложения загружаются быстрее.

Дополнительно
https://wiki.ubuntu.com/Kernel/Reference/IOSchedulers
https://www.kernel.org/doc/html/latest/block/switching-sched.html

https://ru.wikipedia.org/wiki/I/O_scheduling
http://www.opennet.ru/base/sys/ioshedulers.txt.html

Создаем разделы GPT и MBR

Таблица разделов - логи□чески в□ыделенна□я □част□ь жесткого диска фиксированного размера видима□я ОС как отдел□ьное бло□чное устройство
Физически информация о разделах находится□ на жестком диске в разных местах в зависимости от типа таблицы.

Системы адресации дисков накладывают ограничения на размер.
CHS (Cylinder Head Sector) - номер цилиндра (дорожки), номер головки (поверхности) и номер сектора. **Ограничение 8ГБ.**
LBA (Logical Block Addressing) - каждый блок, адресуемый на жёстком диске, имеет свой номер. Например LBA 0 = 0/0/1 (Цилиндр 0/Головка 0/Сектор 1)

Ограничение размера диска обусловлено разрядностью LBA.
LBA 32 bit **Ограничение: 2 Тб.**
LBA 48 bit. **Ограничение: 144.1Пб (**при размере блока 512 байт)
Пример расчета: (2^{48}) 281 474 976 710 656 блоков. 2^{48} × 512 байт приблизительно (Петабайт)
LBA64 максимальный размер адресуемого диска составляет **2 ZiB.**

В структуре MBR используется LBA32
В структуре GPT используется LBA48 -адресация. (зарезервировано 64 бита)

Преимущества MBR
- Совместима с большинством систем.

Недостатки MBR
- Допускает только четыре раздела, с возможностью создания дополнительных подразделов на одном из основных разделов.
- Ограничивает размер раздела двумя терабайтами.

- Информация о разделе хранится только в одном месте — в главной загрузочной записи. Если она повреждена, то весь диск становится нечитаемым.

Преимущества GPT

- Допускает неограниченное количество разделов. Зависит от места, выделенного для таблицы разделов. по умолчанию 128 разделов.
- Нет необходимости в расширенных и логических разделах.
- Контрольные суммы CRC32 позволяют обнаружить ошибки и повреждения заголовка и таблицы разделов.
- Сохраняет заголовок резервной копии и таблицу разделов в конце диска. Есть откуда восстанавливаться.
- Предоставляет уникальный GUID диска и уникальный GUID раздела (PARTUUID) для каждого раздела. Независимый от файловой системы способ ссылки на разделы и диски.
- Предоставляет независимое от файловой системы имя раздела (PARTLABEL).

Недостатки GPT

- Может быть несовместима со старыми системами.

Дополнительная литература
https://habr.com/ru/post/347002/ - подробное побайтовое описание структур MBR и GPT
https://habr.com/ru/post/327572/ -

Утилиты для разбивки диска

Интерактивные
`fdisk`
`gdisk`
`parted`

Для скриптов
sfdisk
sgdisk
parted /dev/sda mklabel gpt mkpart P1 ext3 1MiB 8MiB

Получить список разделов
`fdisk -l /dev/sda`
`gdisk -l /dev/sda`

```
sfdisk -l /dev/sda
parted -l /dev/sda
sgdisk -p /dev/sda

 gdisk /dev/nvme0n1
```

Примеры разбивки дисков на разделы
Утилиты

- parted
- sfdisk
- sgdisk

Создаем

```
for i in {1..40} ; do sgdisk -n ${i}:0:+10M /dev/nvme0n1 ; done
sgdisk -o /dev/nvme0n2
lsblk
```

Делаем копию разделов с одного диска не другой

```
sgdisk -R /dev/sdb /dev/sda
```

Генерируем случайные GUID на всех разделах

```
sgdisk -G /dev/sdb
```

```
parted mklabel gpt
mkpart LVM ext4 2048s 5%

(parted) print
Model: NVMe Device (nvme)
Disk /dev/nvme0n3: 100MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
```

Make partition from script

```
parted /dev/sdc -s  mklabel gpt
parted /dev/sdc -s print
parted /dev/sdc -s mkpart primary 1MiB 1000MiB \
      mkpart primary 1000MiB 2000Mib \
      mkpart primary 2000MiB 3000Mib
```

or

```
for i in $(seq 1 8) ; do parted /dev/sdc -s mkpart primary ${i}000Mib
$((i+1))000Mib ; done
```

Backup and restore partition table

```
sfdisk -d /dev/sda > sda.dump
sfdisk /dev/sda < sda.dump
```

sfdisk examples

```
echo 'label: gpt' | sfdisk /dev/sd
sfdisk -l
sfdisk -V
```

Store dump to file

```
sfdisk -d /dev/sda
sfdisk -d /dev/sda  > sda_partitions
sfdisk /dev/sdb < sda_partitions
```

https://www.gnu.org/software/parted/manual/html_node/unit.html

Создадим на разделах файловую систему

```
for i in $(seq 1 8) ; do parted /dev/sdc -s mkpart primary ${i}000Mib
$((i+1))000Mib ; done
for drive in /dev/sdc[1-9] ; do mkfs.ext4 $drive ; done
for drive in /dev/sdc[1-9] ; do mkdir /mnt/$(basename $drive) ; done
for drive in /dev/sdc[1-9] ; do mount $drive /mnt/$(basename $drive) ;
done
for n in $(seq 1 $RANDOM ) ; do     dd if=/dev/urandom of=file$(
printf %03d "$n" ).bin bs=1 count=1; done

for part in /mnt/sd*
do
  cd $part
  for n in $(seq 1 $RANDOM ) ;
    do
     dd if=/dev/urandom of=file$( printf %03d "$n" ).bin bs=1 count=1
    done
done
```

Массив дисков

Типы RAID

Как выбрать RAID
5 Which raid is for me?

- 2 диска
- 3 - 8 дисков
- больше 8 дисков

- RAID0 (Performance) - данные распределены между дисками. (striping)
- RAID1 (Redundancy) - полная копия данных между двумя или более дискам (mirrored)
- RAID4 (Error Checking) - данные распределены между дисками и один диск используется для хранения контрольных данных (parity information).
- RAID5 (Distributed Error Checking) - данные и контрольные данные распределены между дисками.
- RAID6 (Redundant Error Checking) - данные и два набора контрольных данных распределены между дисками.
- RAID10 (Performance, Redundancy) - гибридный массив. данные распределены между дисками и сделана их копия. (striping + mirroring)

Принимаем решение
https://en.wikipedia.org/wiki/Standard_RAID_levels#Comparison

Аппаратный против software RAID

- BBU (Battery Backup Unit)
- read/write cache

Hardware RAID external controllers vs Internal controllers

Внешний технологии

- FibreChannel,
- NVMe over Fabrics (Ethernet, Infiniband)

Обслуживание BBU
https://www.thomas-krenn.com/en/wiki/Battery_Backup_Unit_(BBU/BBM)_Maintenance_for_RAID_Controllers

Error Recovery Control

Error Recovery Control - это технология которая позволяет системному администратору конфигурировать количество времени которое контроллеру диска разрешено потратить на восстановление от ошибки чтения или записи.

Две стратегии поведения НЖМД при обнаружении ошибки:

- standalone/desktop — пытаться прочитать до последнего
- raid — не ждать помечать диск как сбойный

Разные производители называют технологию по разному:

- **SCT ERC (SMART Command Transport Error Recovery Control) or simply ERC** (Western Digital)
- **TLER (Time-Limited Error Recovery) ,** Samsung/Hitachi
- **CCTL (Command Completion Time Limit)** Samsung/Hitachi

Узнать значение ERC

```
smartctl -l scterc /dev/sda
     SCT Error Recovery Control:
                Read: Disabled
                Write: Disabled
Установить значение ERC
smartctl -l scterc,150,150 /dev/sda
    SCT Error Recovery Control:
                Read:    150 (15.0 seconds)
                Write:    150 (15.0 seconds)
```

https://habr.com/ru/post/92701/
https://en.wikipedia.org/wiki/Error_recovery_control
https://raid.wiki.kernel.org/index.php/Timeout_Mismatch

Программный RAID

Синтаксис команды mdadm

mdadm <command --help
where <command> can be several options, including

- create,
- assemble,
- build,
- stop, etc.

7.1.2 Basic Usage
The table below lists the available command parameters.

-c Specifies the strip size in kilobytes.
-l Specifies the RAID level (i.e., 0, 1, 5, 10).
-n Number of devices to be used in a RAID volume.
-x Number of spare devices in the initial RAID array.
-z Specifies the size (in kilobytes) of space dedicated on each disk to the RAID volume. This must be a multiple of the chunk size. A suffix of ‘M’ or ‘G’ can be given to indicate megabytes or gigabytes, respectively (this also applies to the –c parameter).

```
mdadm --create<newDevice><devicesToUse>
--raid-devices=<numberActiveDisks> -- level=<raidLevel>
```
либо короткая запись
mdadm –c <newDevice> <devicesToUse> -n <numberActiveDisks> l <raidLevel>

Note: For the remainder of this document, all examples will assume the creation of a RAID 1 device using /dev/nvmeXn1 and /dev/nvmeXn1.
mdadm --detail <mdDevice>
mdadm --stop <mdDevice>
mdadm --zero-superblock <device>

Создаем RAID 1 уровня - зеркало. Примеры обслуживания.

```
mdadm --create --verbose /dev/md/raid1_example --level=1
--raid-devices=2 /dev/sdb /dev/sdc
cat /proc/mdstat
mdadm /dev/md/raid1_example
mdadm --detail /dev/md/raid1_example
```

Добавим диск. Какой тип у диска?
```
mdadm --add /dev/md/raid1_example /dev/sdd
mdadm --detail /dev/md/raid1_example
```

Выход из строя диска с hot spare
```
mdadm /dev/md/raid1_example --fail /dev/sdd
mdadm --detail /dev/md/raid1_example
mdadm /dev/md/raid1_example --remove /dev/sdd # удалили сбойный диск
```

Выход из строя диска без hot spare
```
mdadm /dev/md/raid1_example --fail /dev/sdc
mdadm --detail /dev/md/raid1_example
mdadm /dev/md/raid1_example --add /dev/sdd
```

```
mdadm --detail /dev/md/raid1_example
mdadm /dev/md/raid1_example --remove /dev/sdc
```

Остановка и старт массива

```
mdadm /dev/md/raid1_example_0
reboot
cat /proc/mdstat
```

Диск собрался автоматически?

```
Остановим без перезагрузки
mdadm --stop /dev/md5
cat /proc/mdstat
mdadm --assemble --scan
```

Старт когда один диск полностью поврежден

```
wipefs -a /dev/sd[dc]
reboot
cat /proc/mdstat
mdadm --assemble --scan
mdadm --stop /dev/md127
mdadm --examine
mdadm --assemble --run /dev/md/raid1_example_0 /dev/sdb
mdadm --assemble --run /dev/md5  /dev/sdb /dev/sdc
mdadm --assemble --update=name --name=newname /dev/md/newname
mdadm --examine /dev/sdc /dev/sdb | grep  UUID
```

Как найти устройства которые входят в массив?
Как стартануть?
Что хранится в суперблоке?

Пример создания RAID10

```
wipefs -a /dev/sd[b-e]
mdadm --create /dev/md0 --level=10 --raid-devices=4 /dev/sd[b-e]
mdadm --detail /dev/md0
mdadm /dev/md0 --fail /dev/sdc
```

Что происходит если добавляем диски разного размера?
Сколько устройств может выйти из строя?
Как определить какие устройства могут выйти из строя?

Пример создания RAID5

```
wipefs -a /dev/nvme0n[1-3]
mdadm --create --verbose /dev/md/raid5 --level=5  --raid-devices=3
/dev/nvme0n[1-3]
mdadm --add /dev/md/raid5 /dev/nvme0n4
mdadm --grow  --raid-devices=4  /dev/md/raid5
```

или

```
mdadm --grow  --raid-devices=4  /dev/md/raid5
--backup-file=raid5backup
```

Что происходит при добавлении нового устройства?

Сколько устройств может выйти из строя?

Зачем файл бэкапа? (The process can take many hours or even days. There is a critical section at start, which cannot be backed up. To allow recovery after unexpected power failure,)

TODO:

Пример переноса дисков на другую систему:

когда - отказала материнская плата, контроллер дисков

VM1 - добавить 3 диска
- собрать raid5
VM2 - добавить эти 3 диска
стартануть RAID

```
mdadm --examine /dev/sdb /dev/sdc /dev/sdd /dev/sde
```

Пример создание RAID на loopback устройствах

```
for num in {0..4} ; do
 dd if=/dev/zero of=raid_drive_${num} bs=1M count=200
losetup /dev/loop${num} raid_drive_${num}
; done
mdadm --create --verbose /dev/md5 --level=5  --raid-devices=3
/dev/loop0 /dev/loop1 /dev/loop2
mdadm --detail /dev/md5
```

Добавим диск: spare диск
mdadm --add /dev/md5 /dev/loop3

Добавим диск: увеличим размер
```
mdadm --grow /dev/md5 -n4 --backup-file=raid5backup
mdadm --add /dev/md5 /dev/loop3
```

Удаляем блочные устройства
mdadm --stop /dev/md5
for num in {0..4} ; do
losetup -d /dev/loop${num
done

Примеры создания RAID1 разное число дисков
```
mdadm --create /dev/md/raid1example /dev/sda /dev/sdb  --level=1
mdadm --create /dev/md/raid1example /dev/loop{0..2}  --level=1
--raid-devices=3
mdadm --create --verbose /dev/md0 --level=mirror --raid-devices=2
/dev/sdb1 /dev/sdc1 --spare-devices=1 /dev/sdd1
```

Примеры создания RAID5 разное число дисков

```
mdadm --create --raid-devices=3 --level=5 /dev/md/raid5 /dev/sdb
/dev/sdc /dev/sdd

mdadm --create --raid-devices=4 --level=5 /dev/md/nvmeraid5
/dev/nvme0n1p{1..4}
mdadm --add /dev/md/nvmeraid5 /dev/nvme0n1p5
cat /proc/mdstat
mdadm --detail /dev/md/nvmeraid5

mdadm --fail /dev/md/nvmeraid5 /dev/nvme0n1p1
cat /proc/mdstat
mdadm --detail /dev/md/nvmeraid5
```

Удалим сбойный диск

mdadm --remove /dev/md/nvmeraid5 /dev/nvme0n1p1

Тест блочного устройства

fio --time_based --name=benchmark --runtime=30 --filename=/dev/nvme0n1 --nrfiles=1 --ioengine=libaio --iodepth=32 --direct=1 --invalidate=1 --verify=0 --verify_fatal=0 --numjobs=4 --rw=randread --blocksize=4k --randrepeat=false