

Домашнее задание 1

Linux Архитектура и файловые системы

Суханов М.Ю.

Задание 1. Kernel and Module Inspection (15 баллов)

1. Продемонстрировать версию ядра вашей ОС:

```
ubuntu1@serverubuntu:~$ lsb_release -d && uname -r
Description:      Ubuntu 25.04
6.14.0-32-generic
```

Через команду `lsb_release` просмотрели дистрибутив линукс – Ubuntu 25.04

`Uname` показал информацию о ядре. Версия ядра – 6.14.0-32-generic

2. Показать все загруженные модули ядра.

```
ubuntu1@serverubuntu:~$ lsmod | awk '$3 > 0'
```

Module	Size	Used by
qrtr	53248	2
intel_rapl_common	53248	1 intel_rapl_msr
pmt_telemetry	16384	1 intel_pmc_core
pmt_class	16384	1 pmt_telemetry
intel_vsec	20480	1 intel_pmc_core
snd_hda_codec_generic	122880	1
binfmt_misc	24576	1
snd_intel_dspcfg	45056	1 snd_hda_intel
snd_intel_sdw_acpi	16384	1 snd_intel_dspcfg
nls_iso8859_1	12288	1
snd_hda_codec	204800	2 snd_hda_codec_generic,snd_hda_intel
snd_hda_core	147456	3 snd_hda_codec_generic,snd_hda_intel,snd_hda_codec
snd_hwdep	20480	1 snd_hda_codec
i2c_smbus	20480	1 i2c_i801
i2c_mux	16384	1 i2c_i801
snd_pcm	196608	3 snd_hda_intel,snd_hda_codec,snd_hda_core
snd_timer	53248	1 snd_pcm
snd	143360	6 snd_hda_codec_generic,snd_hwdep,snd_hda_intel,snd_hda_codec,snd_timer,snd_pcm
soundcore	16384	1 snd
sch_fq_codel	24576	2
nfnetlink	20480	2
vmw_vsock_virtio_transport_common	57344	1 vsock_loopback
vsock	61440	5 vmw_vsock_virtio_transport_common,vsock_loopback,vmw_vsock_vmci_transport
vmw_vmci	106496	1 vmw_vsock_vmci_transport
x_tables	65536	1 ip_tables
autofs4	57344	2
async_raid6_recov	20480	1 raid456
async_memcpy	16384	2 raid456,async_raid6_recov
async_pq	20480	2 raid456,async_raid6_recov
async_xor	16384	3 async_pq,raid456,async_raid6_recov
async_tx	16384	5 async_pq,async_memcpy,async_xor,raid456,async_raid6_recov
xor	20480	2 async_xor,btrfs
raid6_pq	126976	4 async_pq,btrfs,raid456,async_raid6_recov
hid	266240	2 usbhid,hid_generic
polyval_generic	12288	1 polyval_clmulni
ahci	49152	3
libahci	53248	1 ahci
virtio_dma_buf	12288	1 virtio_gpu
crypto_simd	16384	1 aesni_intel
cryptd	24576	2 crypto_simd,ghash_clmulni_intel

```
ubuntu1@serverubuntu:~$
```

Модулей ядра большое количество. Для демонстрации было принято решение отфильтровать выборку по только используемым модулям (колонка Used by).

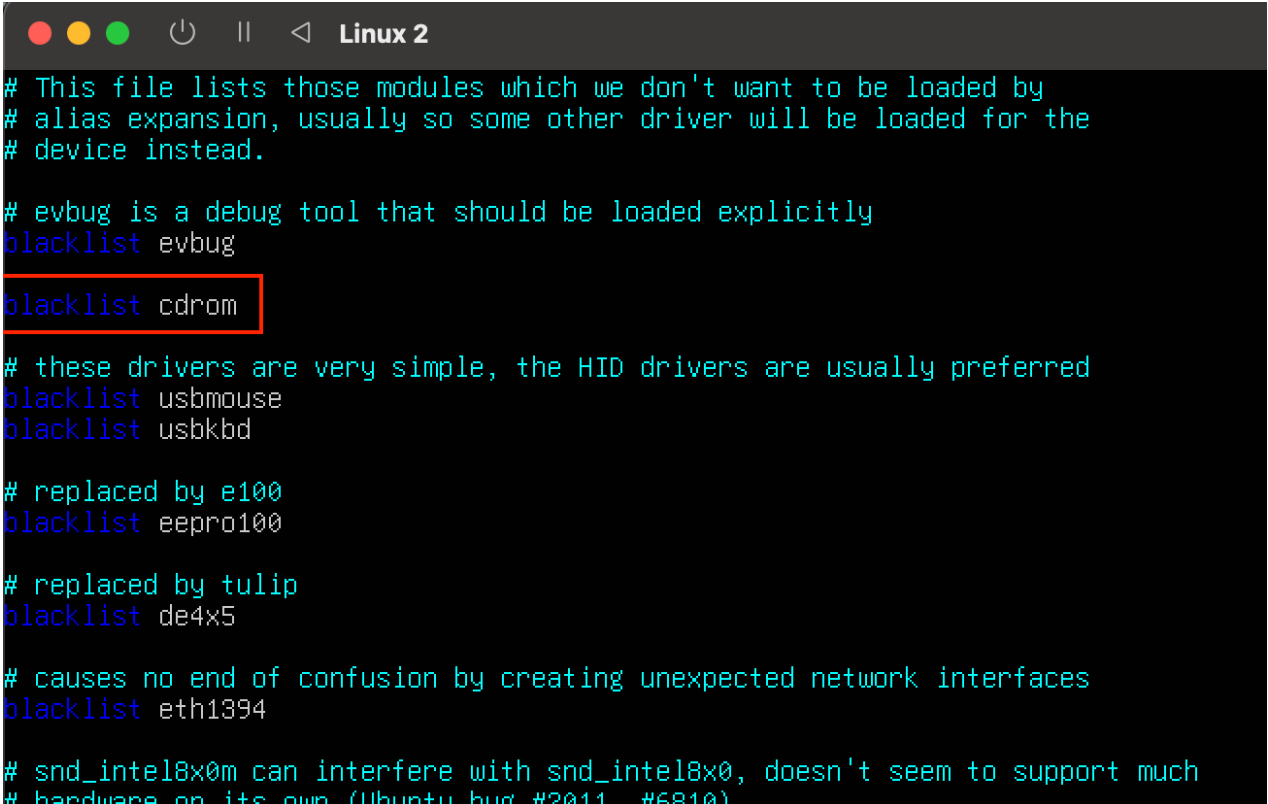
3. Отключить автозагрузку модуля cdrom.

Для отключения автозагрузки можно использовать blacklist.conf. Это конфигурационный файл в Linux, который используется для запрета загрузки определённых модулей ядра.

Откроем конфигурационный файл:

```
ubuntu1@serverubuntu:~$ sudo vi /etc/modprobe.d/blacklist.conf
```

Пропишем правило запрета автозагрузки cdrom:



```
# This file lists those modules which we don't want to be loaded by
# alias expansion, usually so some other driver will be loaded for the
# device instead.

# evbug is a debug tool that should be loaded explicitly
blacklist evbug

blacklist cdrom

# these drivers are very simple, the HID drivers are usually preferred
blacklist usbmouse
blacklist usbkbd

# replaced by e100
blacklist eeepro100

# replaced by tulip
blacklist de4x5

# causes no end of confusion by creating unexpected network interfaces
blacklist eth1394

# snd_intel8x0m can interfere with snd_intel8x0, doesn't seem to support much
# hardware on its own (Ubuntu bug #2011, #6810)
```

Сохраним файл и применим изменения:

```
ubuntu1@serverubuntu:~$ sudo update-initramfs -u
update-initramfs: Generating /boot/initrd.img-6.14.0-32-generic
ubuntu1@serverubuntu:~$ lsmod | grep cdrom
ubuntu1@serverubuntu:~$
```

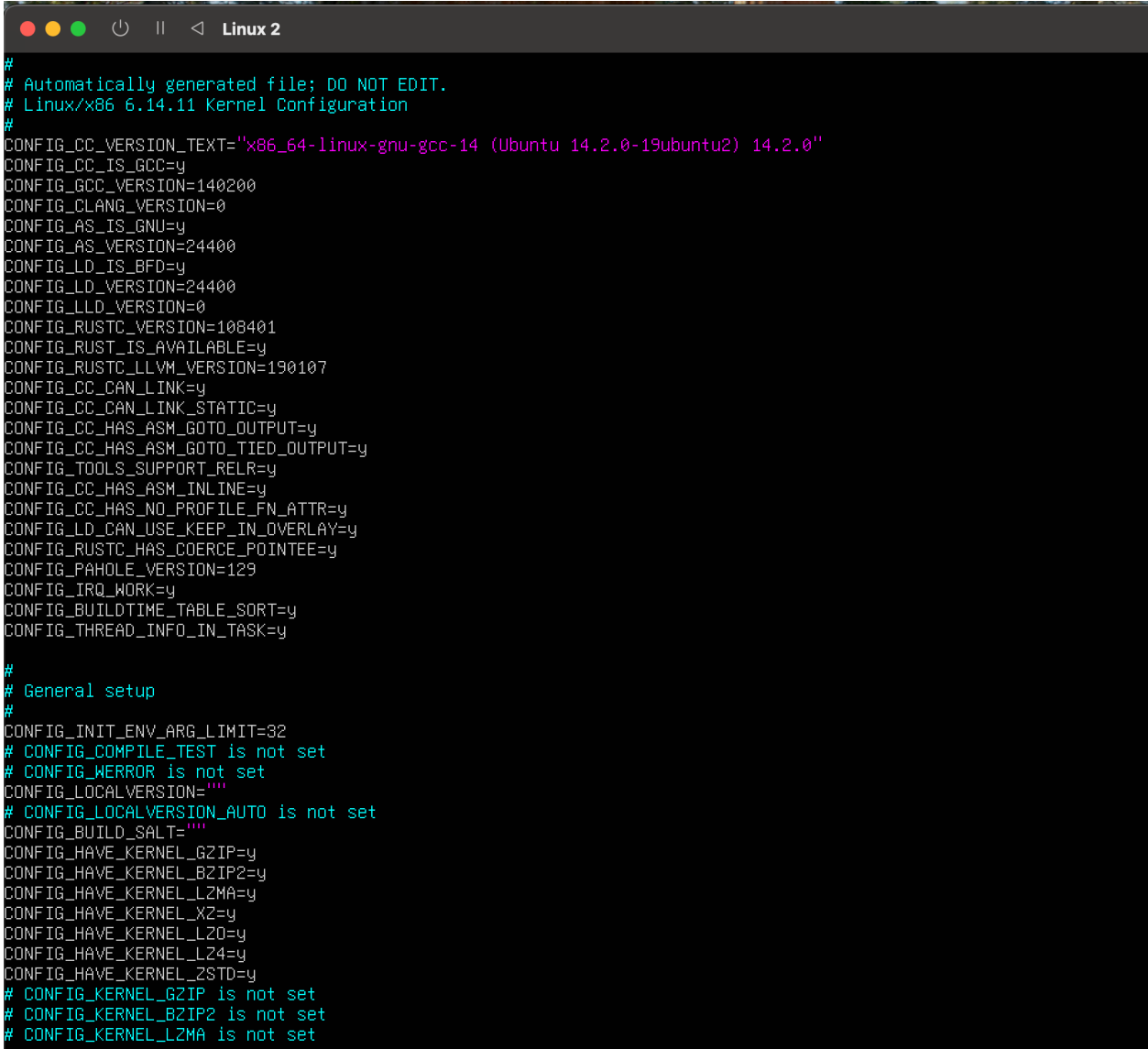
Для применения изменений пересоберем initramfs — временную файловую систему, которую ядро загружает при старте системы. А также проверим результат через просмотр всех загруженных модулей ядра с фильтрацией по cdrom.

4. Найти и описать конфигурацию ядра (файл конфигурации, параметр CONFIG_XFS_FS).

Конфигурация ядра расположена по следующему пути:

```
# CONFIG_KERNEL_LZMA is not set
ubuntu1@serverubuntu:/home$ vi /boot/config-6.14.0-32-generic _
```

Файл конфигурации довольно объемный:



```
#
# Automatically generated file; DO NOT EDIT.
# Linux/x86_64 6.14.11 Kernel Configuration
#
CONFIG_CC_VERSION_TEXT="x86_64-linux-gnu-gcc-14 (Ubuntu 14.2.0-19ubuntu2) 14.2.0"
CONFIG_CC_IS_GCC=y
CONFIG_GCC_VERSION=140200
CONFIG_CLANG_VERSION=0
CONFIG_AS_IS_GNU=y
CONFIG_AS_VERSION=24400
CONFIG_LD_IS_BFD=y
CONFIG_LD_VERSION=24400
CONFIG_LLD_VERSION=0
CONFIG_RUSTC_VERSION=108401
CONFIG_RUSTC_IS_AVAILABLE=y
CONFIG_RUSTC_LLVM_VERSION=190107
CONFIG_CC_CAN_LINK=y
CONFIG_CC_CAN_LINK_STATIC=y
CONFIG_CC_HAS_ASM_GOTO_OUTPUT=y
CONFIG_CC_HAS_ASM_GOTO_TIED_OUTPUT=y
CONFIG_TOOLS_SUPPORT_RELR=y
CONFIG_CC_HAS_ASM_INLINE=y
CONFIG_CC_HAS_NO_PROFILE_FN_ATTR=y
CONFIG_LD_CAN_USE_KEEP_IN_OVERLAY=y
CONFIG_RUSTC_HAS_COERCE_POINTEE=y
CONFIG_PAHOLE_VERSION=129
CONFIG_IRQ_WORK=y
CONFIG_BUILDTIME_TABLE_SORT=y
CONFIG_THREAD_INFO_IN_TASK=y

#
# General setup
#
CONFIG_INIT_ENV_ARG_LIMIT=32
# CONFIG_COMPILE_TEST is not set
# CONFIG_WERROR is not set
CONFIG_LOCALVERSION=""
# CONFIG_LOCALVERSION_AUTO is not set
CONFIG_BUILD_SALT=""
CONFIG_HAVE_KERNEL_GZIP=y
CONFIG_HAVE_KERNEL_BZIP2=y
CONFIG_HAVE_KERNEL_LZMA=y
CONFIG_HAVE_KERNEL_XZ=y
CONFIG_HAVE_KERNEL_LZ0=y
CONFIG_HAVE_KERNEL_LZ4=y
CONFIG_HAVE_KERNEL_ZSTD=y
# CONFIG_KERNEL_GZIP is not set
# CONFIG_KERNEL_BZIP2 is not set
# CONFIG_KERNEL_LZMA is not set
```

В первом блоке описываются инструменты для сборки ядра.

Первая строка – общая информация. Архитектура: x86_64, Версия ядра: 6.14.11 и т.д;

CONFIG_CC_CAN_LINK / CONFIG_CC_CAN_LINK_STATIC – проверки - может ли компилятор (CC) не просто скомпилировать код, но и создать из него готовый исполняемый файл;

CONFIG_CC_HAS_ASM_GOTO_OUTPUT — поддержка расширение синтаксиса ассемблера asm goto.

CONFIG_KERNEL_* — это опции выбора. Пользователь (или скрипт) должен выбрать ровно один из них, чтобы определить, каким алгоритмом будет сжат итоговый образ ядра.

```
ubuntu1@serverubuntu:/home$ grep CONFIG_XFS_FS /boot/config-6.14.0-32-generic
ubuntu1@serverubuntu:/home$ grep CONFIG_XFS_FS /boot/config-6.14.0-32-generic
CONFIG_XFS_FS=m
ubuntu1@serverubuntu:/home$
```

CONFIG_XFS_FS — это параметр конфигурации ядра Linux, который управляет поддержкой файловой системы XFS. На выходе получили флаг m. Флаг m (module) — означает, что драйвер файловой системы компилируется в виде загружаемого модуля.

[illegible]

В результате выполнения команды можно увидеть, что:

1. Команда `cat` выполняется через системный вызов `execve`. Происходит запуск бинарника `/usr/bin/cat` с аргументом `/etc/os-release`;
2. `Cat` ищет и открывает файл `/etc/os-release` через `openat`. Файл успешно найден и открыт;
3. Через `read` происходит считывание байтов из файла;
4. Для кодировок текста и корректности считывания, идет обращение к файлу локалей `/usr/lib/locale/locale-archive`;
5. Для выделения и освобождения памяти используются вызовы `mmap`, `brk`, `munmap`;
6. После чтения содержимого вызывается `close` и все файлы закрываются.

Таким образом, с помощью `strace` можно увидеть, какие системные вызовы делает та или иная команда в Linux.

Используемые файлы:

- `os-release` – целевой файл команды `cat`
- `locale/locale-archive` – файл для определения языка/кодировки
- `libc.so.6` – стандартная библиотека C для работы `cat`
- `ld.so.cache` – используется для поиска динамически подключаемых библиотек

```
ubuntu1@serverubuntu:~$ strace -e trace=openat,read,write,close cat /etc/os-release > /dev/null
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
close(3) = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\0\0\1\0\0\0@247\2\0\0\0\0"... , 832) = 832
close(3) = 0
openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
close(3) = 0
openat(AT_FDCWD, "/etc/os-release", O_RDONLY) = 3
read(3, "PRETTY_NAME=\"Ubuntu 25.04\"\nNAME=\"... , 262144) = 391
write(1, "PRETTY_NAME=\"Ubuntu 25.04\"\nNAME=\"... , 391) = 391
read(3, "", 262144) = 0
close(3) = 0
close(1) = 0
close(2) = 0
```

Также можно заметить, что в результате присутствует вызов `write`, однако результата вызова команды в терминале нет. Это объясняется тем, что `/dev/null` перенаправляет стандартный вывод в “пустоту”.

2. Создать раздел на /dev/sdb (в моем случае /dev/vda), используя fdisk или parted.

```
ubuntu1@serverubuntu:~$ sudo fdisk /dev/vda
Welcome to fdisk (util-linux 2.40.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS (MBR) disklabel with disk identifier 0x925a9430.

Command (m for help): n
Partition type
  p   primary (0 primary, 0 extended, 4 free)
  e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-4194303, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-4194303, default 4194303):

Created a new partition 1 of type 'Linux' and of size 2 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

В ходе создания раздела были пройдены следующие шаги:

1. Создаем раздел - Command (m for help): n;
2. Выбираем тип раздела - Select (default p): p # primary раздел
3. Выбираем номер раздела (Partition number) - 1
4. Выбираем сектор начала (First sector) – выбрали по умолчанию
5. Выбираем размер раздела (Last sector) – выбрали все свободное пространство
6. Записали изменения в конце.

```
ubuntu1@serverubuntu:~$ lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                                  8:0    0   20G  0 disk
├─sda1                              8:1    0   953M  0 part /boot/efi
├─sda2                              8:2    0    1,8G  0 part /boot
└─sda3                              8:3    0   17,3G  0 part
   └─ubuntu--vg-ubuntu--lv 252:0    0    10G  0 lvm  /
sr0                                  11:0    1    1,9G  0 rom
vda                                  253:0    0    2G   0 disk
└─vda1                              253:1    0    2G   0 part
```

В результате можно увидеть, что создание раздела прошло успешно.

3. Создать Physical Volume (PV) на этом разделе.

```
ubuntu1@serverubuntu:~$ sudo pvcreate /dev/vda1
Physical volume "/dev/vda1" successfully created.
ubuntu1@serverubuntu:~$ sudo pvdisplay
--- Physical volume ---
PV Name               /dev/sda3
VG Name               ubuntu-vg
PV Size               <17,32 GiB / not usable 0
Allocatable           yes
PE Size               4,00 MiB
Total PE              4433
Free PE               1873
Allocated PE          2560
PV UUID               9BzWm5-2Rvp-ARgq-iJot-zNw7-ZxSb-cwACBI

"/dev/vda1" is a new physical volume of "<2,00 GiB"
--- NEW Physical volume ---
PV Name               /dev/vda1
VG Name
PV Size               <2,00 GiB
Allocatable           NO
PE Size               0
Total PE              0
Free PE               0
Allocated PE          0
PV UUID               QeMwnj-5ueq-ywZp-8vyX-u3jC-E2om-43jDzw
```

Для создания Physical Volume мы использовали команду `pvcreate`, где также указали наш созданный ранее раздел - `/dev/vda1`.

4. Создать Volume Group (VG) с именем `vg_highload`.

```
ubuntu1@serverubuntu:~$ sudo vgcreate vg_highload /dev/vda1
Volume group "vg_highload" successfully created
ubuntu1@serverubuntu:~$ sudo vgs
VG                #PV #LV #SN Attr   VSize   VFree
ubuntu-vg         1   1   0 wz--n- <17,32g <7,32g
vg_highload       1   0   0 wz--n- <2,00g  <2,00g
ubuntu1@serverubuntu:~$ _
```

Для создания Volume Group мы использовали команду `vgcreate`, где в качестве имени группы указали `vg_highload`.

5. Создать два Logical Volume (LV): data_lv (1200 MiB) и logs_lv (оставшееся место).

```
ubuntu1@serverubuntu:~$ sudo lvcreate -L 1200M -n data_lv vg_highload
Logical volume "data_lv" created.
ubuntu1@serverubuntu:~$ sudo lvcreate -l 100%FREE -n log_lv vg_highload
Logical volume "log_lv" created.
ubuntu1@serverubuntu:~$ sudo lvs
LV          VG          Attr      LSize   Pool Origin Data%  Meta%   Move Log Cpy%Sync Convert
ubuntu-lv   ubuntu-vg    -wi-a---- 10,00g
data_lv     vg_highload -wi-a----- 1,17g
log_lv      vg_highload -wi-a----- 844,00m
ubuntu1@serverubuntu:~$
```

Logical Volume мы создаем через lvcreate. Флаг -l используется для указания размера тома, -n для указания названия тома.

6. Отформатировать data_lv как ext4 и примонтировать в /mnt/app_data.

Через mkfs.ext4 отформатируем наш том как ext4.

```
ubuntu1@serverubuntu:~$ sudo mkfs.ext4 /dev/vg_highload/data_lv
mke2fs 1.47.2 (1-Jan-2025)
Discarding device blocks: done
Creating filesystem with 307200 4k blocks and 76800 inodes
Filesystem UUID: c7a41abf-b30b-4b5a-bcad-00343d32ae64
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

Далее создадим точку монтирования и смонтируем наш том.

```
ubuntu1@serverubuntu:~$ sudo mkdir -p /mnt/app_data
ubuntu1@serverubuntu:~$ sudo mount /dev/vg_highload/data_lv /mnt/app_data
ubuntu1@serverubuntu:~$ df -h
Filesystem                                Size  Used Avail Use% Mounted on
tmpfs                                     340M  1,2M  339M   1% /run
efivarfs                                 256K   53K  199K  22% /sys/firmware/efi/efivars
/dev/mapper/ubuntu--vg-ubuntu--lv        9,8G  3,1G  6,2G  34% /
tmpfs                                     1,7G   0  1,7G   0% /dev/shm
tmpfs                                     5,0M   0  5,0M   0% /run/lock
tmpfs                                     1,7G   0  1,7G   0% /tmp
tmpfs                                     1,0M   0  1,0M   0% /run/credentials/systemd-resolved.service
/dev/sda2                                1,7G  122M  1,5G   8% /boot
/dev/sda1                                952M  6,2M  945M   1% /boot/efi
tmpfs                                     1,0M   0  1,0M   0% /run/credentials/getty@tty1.service
tmpfs                                     340M  12K  340M   1% /run/user/1000
tmpfs                                     1,0M   0  1,0M   0% /run/credentials/systemd-networkd.service
tmpfs                                     1,0M   0  1,0M   0% /run/credentials/systemd-journald.service
/dev/mapper/vg_highload-data_lv          1,2G  324K  1,1G   1% /mnt/app_data
```

Просмотрим информацию о файловых системах и увидим наш примонтированный том.

7. Отформатировать log_lv как xfs и примонтировать в /mnt/app_logs.

Через mkfs.xfs отформатируем наш том как xfs

```
<value> is xxx (512 byte blocks):
ubuntu1@serverubuntu:~$ sudo mkfs.xfs /dev/vg_highload/log_lv
meta-data=/dev/vg_highload/log_lv isize=512    agcount=4, agsize=54016 blks
        =                               sectsz=512    attr=2, projid32bit=1
        =                               crc=1        finobt=1, sparse=1, rmapbt=1
        =                               reflink=1     bigtime=1 inobtcount=1 nrext64=1
        =                               exchange=0    metadir=0
data      =                               bsize=4096   blocks=216064, imaxpct=25
        =                               sunit=0      swidth=0 blks
naming    =version 2                      bsize=4096   ascii-ci=0, ftype=1, parent=0
log       =internal log                   bsize=4096   blocks=16384, version=2
        =                               sectsz=512   sunit=0 blks, lazy-count=1
realtime  =none                          extsz=4096   blocks=0, rtextents=0
        =                               rgcount=0    rgsz=0 extents
Discarding blocks...Done.
```

Далее создадим точку монтирования и смонтируем наш том

```
ubuntu1@serverubuntu:~$ sudo mkdir -p /mnt/app_logs
ubuntu1@serverubuntu:~$ sudo mount /dev/vg_highload/log_lv /mnt/app_logs
ubuntu1@serverubuntu:~$ df -h
Filesystem                                Size  Used Avail Use% Mounted on
tmpfs                                     340M  1,2M  339M   1% /run
efivarfs                                  256K   53K  199K  22% /sys/firmware/efi/efivars
/dev/mapper/ubuntu--vg-ubuntu--lv        9,8G   3,1G   6,2G  34% /
tmpfs                                     1,7G   0   1,7G   0% /dev/shm
tmpfs                                     5,0M   0   5,0M   0% /run/lock
tmpfs                                     1,7G   0   1,7G   0% /tmp
tmpfs                                     1,0M   0   1,0M   0% /run/credentials/systemd-resolved.service
/dev/sda2                                 1,7G  122M   1,5G   8% /boot
/dev/sda1                                 952M   6,2M  945M   1% /boot/efi
tmpfs                                     1,0M   0   1,0M   0% /run/credentials/getty@tty1.service
tmpfs                                     340M   12K  340M   1% /run/user/1000
tmpfs                                     1,0M   0   1,0M   0% /run/credentials/systemd-networkd.service
tmpfs                                     1,0M   0   1,0M   0% /run/credentials/systemd-journald.service
/dev/mapper/vg_highload-data_lv          1,2G  324K   1,1G   1% /mnt/app_data
/dev/mapper/vg_highload-log_lv           780M   48M  733M   7% /mnt/app_logs
ubuntu1@serverubuntu:~$
```

Задание 4. Использование pseudo filesystem (25 баллов)

1. Извлечь из /proc модель CPU и объём памяти (KiB).

```
/dev/mapper/vg_highload-log_lv          780M   48M  733M   7% /mnt/app_logs
ubuntu1@serverubuntu:~$ grep "model name" /proc/cpuinfo | head -n 1
model name      : Intel Core Processor (Skylake)
ubuntu1@serverubuntu:~$ grep "MemTotal" /proc/meminfo
MemTotal:       3481104 kB
ubuntu1@serverubuntu:~$
```

Информация о процессоре лежит в /proc/cpuinfo, информация об объеме памяти в /proc/meminfo.

- Используя /proc/\$\$/status, найдите Parent Process ID (PPid) вашего текущего shell. что означает \$\$?

```
memTotal: 3481104 KB
ubuntu1@serverubuntu:~$ grep PPid /proc/$$/status
PPid: 894
```

PID родительского процесса равен 894. \$\$ - это PID (Process ID) текущего shell-процесса.

- Определить настройки I/O scheduler для основного диска /dev/sda.

```
dev          events          hidden          mq
ubuntu1@serverubuntu:~$ cat /sys/block/sda/queue/scheduler
none [mq-deadline]
```

В Linux у каждого блока устройства (например, диска /dev/sda) есть настройка I/O scheduler (планировщик ввода-вывода). В данном случае имя текущего активного планировщика – mq-deadline. Алгоритм mq-deadline - дедлайны для read/write операций.

- Определить размер MTU для основного сетевого интерфейса (например, eth0 или ens33).

```
device 'ens33' does not exist.
ubuntu1@serverubuntu:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether e2:e1:54:32:e4:7b brd ff:ff:ff:ff:ff:ff
    altname enxe2e15432e47b
ubuntu1@serverubuntu:~$
```

MTU (Maximum Transmission Unit) — это максимальный размер пакета данных, который может быть передан по сетевому интерфейсу за один раз. Из вывода ip link show можно увидеть, что mtu для lo (Виртуальный интерфейс для локальной связи внутри систем) = 65536, для enp0s1 (Физический Ethernet интерфейс) = 1500 пакетов.