

## Установка Arch Linux

1. Создаем новую виртуальную машину в VirtualBox со следующими параметрами:

Имя: ArchLinux

Тип: Linux

Версия: ArchLinux (64-bit)

Объем оперативной памяти: 4096 МБ

Создаем новый виртуальный жесткий диск (тип VDI)

Формат хранения: динамический виртуальный жесткий диск

Фиксированный виртуальный жесткий диск работает быстрее, но занимает больше места на хостовой машине

Размер жесткого диска: 15 ГБ

2. Далее необходимо настроить виртуальную машину. Выбираем нашу ВМ, заходим в настройки.

На вкладке Система:

Материнская плата: Ставим галочку Включить EFI (только специальные ОС)

Процессор: минимум 4 (лучше 6)

На вкладке Дисплей:

Экран: Видеопамять 128 МБ

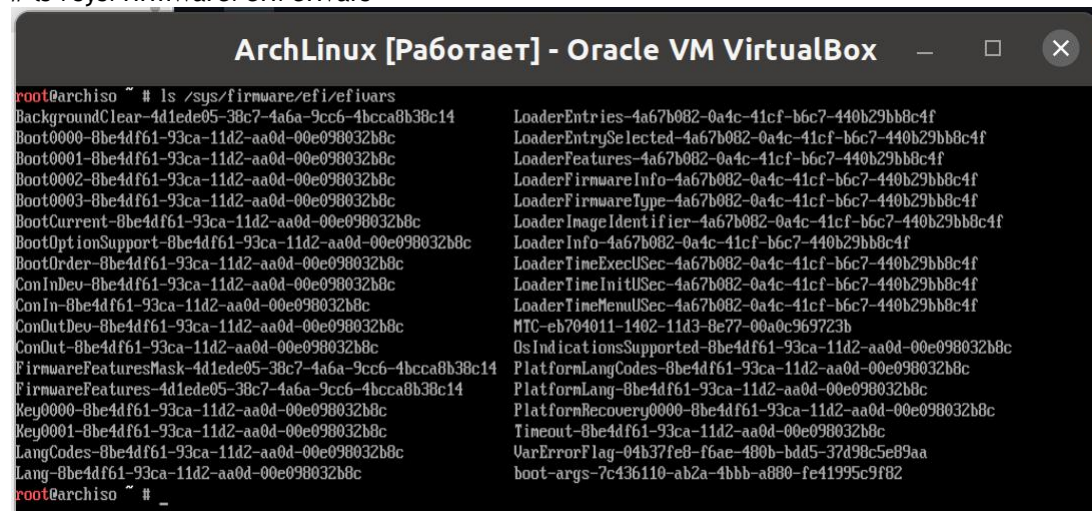
3. Скачиваем iso образ <https://mirror.yandex.ru/archlinux/iso/2025.03.01/>

4. Запускаем ВМ, выбираем загрузочный диск archlinux-2024.10.01-x86\_64.iso

5. В окне выбора системы выбираем Arch Linux install medium (x86\_64, UEFI)

6. Первое, что нужно сделать после загрузки командной оболочки - это проверить загрузились ли мы в режиме UEFI. Делается это следующей командой:

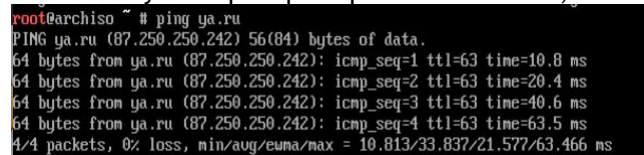
# ls /sys/firmware/efi/efivars



```
root@archiso ~ # ls /sys/firmware/efi/efivars
BackgroundClear-4d1ede05-38c7-4a6a-9cc6-4bcc8b38c14
Boot0000-8be4df61-93ca-11d2-aa0d-00e098032b8c
Boot0001-8be4df61-93ca-11d2-aa0d-00e098032b8c
Boot0002-8be4df61-93ca-11d2-aa0d-00e098032b8c
Boot0003-8be4df61-93ca-11d2-aa0d-00e098032b8c
BootCurrent-8be4df61-93ca-11d2-aa0d-00e098032b8c
BootOptionSupport-8be4df61-93ca-11d2-aa0d-00e098032b8c
BootOrder-8be4df61-93ca-11d2-aa0d-00e098032b8c
ConInDev-8be4df61-93ca-11d2-aa0d-00e098032b8c
ConIn-8be4df61-93ca-11d2-aa0d-00e098032b8c
ConOutDev-8be4df61-93ca-11d2-aa0d-00e098032b8c
ConOut-8be4df61-93ca-11d2-aa0d-00e098032b8c
FirmwareFeaturesMask-4d1ede05-38c7-4a6a-9cc6-4bcc8b38c14
Key0000-8be4df61-93ca-11d2-aa0d-00e098032b8c
Key0001-8be4df61-93ca-11d2-aa0d-00e098032b8c
LangCodes-8be4df61-93ca-11d2-aa0d-00e098032b8c
Lang-8be4df61-93ca-11d2-aa0d-00e098032b8c
LoaderEntries-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f
LoaderEntrySelected-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f
LoaderFeatures-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f
LoaderFirmwareInfo-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f
LoaderFirmwareType-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f
LoaderImageIdentifier-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f
LoaderInfo-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f
LoaderTimeExecUsec-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f
LoaderTimeInitUsec-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f
LoaderTimeMenuUsec-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f
MTC-e704011-1402-11d3-8e77-00a0c969723b
OsIndicationsSupported-8be4df61-93ca-11d2-aa0d-00e098032b8c
PlatformLangCodes-8be4df61-93ca-11d2-aa0d-00e098032b8c
PlatformLang-8be4df61-93ca-11d2-aa0d-00e098032b8c
PlatformRecovery0000-8be4df61-93ca-11d2-aa0d-00e098032b8c
Timeout-8be4df61-93ca-11d2-aa0d-00e098032b8c
VarErrorFlag-04b37fe8-f6ae-480b-bdd5-37d98c5e89aa
boot-args-7c436110-ab2a-4bbb-a800-fe41995c9f82
```

Если появилось сообщение об ошибке, то мы загрузились в режиме BIOS, а не UEFI.

7. Затем нужно проверить работает ли сеть, выполним команду ping



```
root@archiso ~ # ping ya.ru
PING ya.ru (87.250.250.242) 56(84) bytes of data:
64 bytes from ya.ru (87.250.250.242): icmp_seq=1 ttl=63 time=10.8 ms
64 bytes from ya.ru (87.250.250.242): icmp_seq=2 ttl=63 time=20.4 ms
64 bytes from ya.ru (87.250.250.242): icmp_seq=3 ttl=63 time=40.6 ms
64 bytes from ya.ru (87.250.250.242): icmp_seq=4 ttl=63 time=63.5 ms
4/4 packets, 0% loss, min/avg/max = 10.813/33.837/21.577/63.466 ms
```

команда ping будет выполняться бесконечно, чтобы ее остановить нажмите Ctrl+C

8. Следующим этапом необходимо обновить системные часы с помощью команды:

# timedatectl set-ntp true

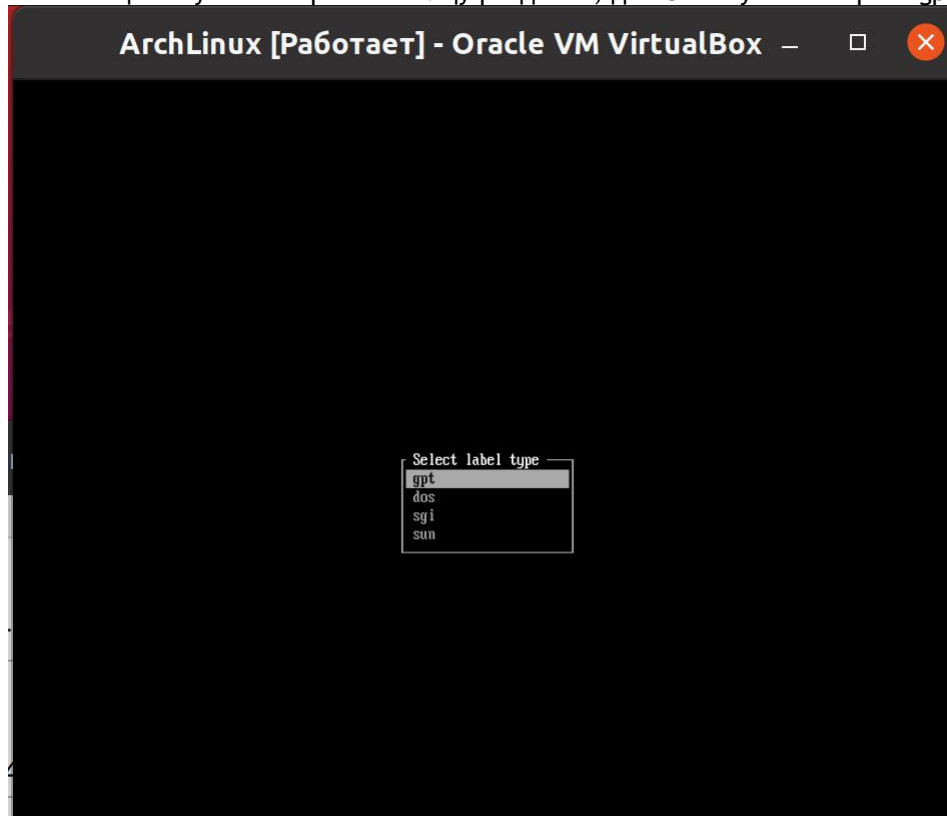
проверить выполнение команды можно с помощью команды:

# timedatectl status

```
ArchLinux [Работает] - Oracle VM VirtualBox

root@archiso ~ # timedatectl set-ntp true
root@archiso ~ #
root@archiso ~ # timedatectl status
          Local time: Sat 2021-10-09 13:41:51 UTC
          Universal time: Sat 2021-10-09 13:41:51 UTC
              RTC time: Sat 2021-10-09 13:41:52
              Time zone: UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
root@archiso ~ #
```

9. Дальше нужно разметить жесткий диск, делается это с помощью утилиты # cfdisk на этом шаге нужно выбрать таблицу разделов, для UEFI нужно выбрать gpt



Нам нужно создать раздел для загрузки UEFI. С помощью стрелочек выбираем создать новый раздел [NEW]



нам будет достаточно 500MB



Стрелкой вниз переместимся на Free space и создадим раздел под файл подкачки (SWAP). Рекомендуется создавать раздел, размеров в 2 раза больше, чем объем оперативной памяти, выделяемый на вашу VM. Мы выделим 4GB



Оставшееся пространство отдадим под корневую файловую систему

Disk: /dev/sda				
Size: 15 GiB, 16106127360 bytes, 31457280 sectors				
Label: gpt, identifier: BEC9FA1A-C85F-0B45-BD54-D523A782F9FC				
Device	Start	End	Sectors	Size Type
/dev/sda1	2048	978943	976896	477M Linux filesystem
/dev/sda2	978944	8792063	7813120	3.76 Linux filesystem
> /dev/sda3	8792064	31457246	22665183	10.86 Linux filesystem

10. Теперь необходимо выбрать тип для каждого раздела с помощью опции Type

Разделу /dev/sda1 назначаем Efi system

Разделу /dev/sda2 назначаем Linux swap

Раздел /dev/sda3 оставляем Linux filesystem

Затем необходимо зафиксировать все внесенные изменения с помощью опции Write

```
[ Delete ] [ Resize ] [ Quit ] [ Type ] [ Help ] [ Write ] [ Dump ]
```

Подтверждаем внесение изменений. Пишем yes

```
Are you sure you want to write the partition table to disk? yes
```

выходим из меню, выбрав пункт Quit

11. Наши разделы созданы, проверяем это с помощью команды

# lsblk

```
root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 673.1M 1 loop /run/archiso/airootfs
sda 8:0 0 15G 0 disk
├─sda1 8:1 0 477M 0 part
├─sda2 8:2 0 3.7G 0 part
└─sda3 8:3 0 10.8G 0 part
sr0 11:0 1 846.3M 0 rom /run/archiso/bootmnt
root@archiso ~ # _
```

12. Далее нам необходимо для раздела sda1 указать файловую систему FAT32

# mkfs.fat -F32 /dev/sda1

```
root@archiso ~ # mkfs.fat -F32 /dev/sda1
mkfs.fat 4.2 (2021-01-31)
root@archiso ~ # _
```

раздел sda2 у нас swap, его нужно инициализировать командой

# mkswap /dev/sda2

```
root@archiso ~ # mkswap /dev/sda2
Setting up swapspace version 1, size = 3.7 GiB (4000313344 bytes)
no label, UUID=54cb26ad-9e71-415c-8fb5-19ebb0428dfd
root@archiso ~ # _
```

sda3 у нас основной раздел и его мы будем форматировать в файловую систему btrfs

# mkfs.btrfs /dev/sda3

```
root@archiso ~ # mkfs.btrfs /dev/sda3
btrfs-progs v5.14.1
See http://btrfs.wiki.kernel.org for more information.

Label: (null)
UUID: a5e6323a-d6a5-4d99-83cc-3699caf7221b
Node size: 16384
Sector size: 4096
Filesystem size: 10.81GiB
Block group profiles:
  Data: single 8.00MiB
  Metadata: DUP 256.00MiB
  System: DUP 8.00MiB
SSD detected: no
Zoned device: no
Incompat features: extref, skinny-metadata
Runtime features:
Checksum: crc32c
Number of devices: 1
Devices:
  ID     SIZE  PATH
  1     10.81GiB /dev/sda3
root@archiso ~ # _
```

13. Теперь включим swap командой

# swapon /dev/sda2

```
root@archiso ~ # swapon /dev/sda2
root@archiso ~ # _
```

14. С помощью команды mount примонтируем наш основной раздел

# mount /dev/sda3 /mnt

```
root@archiso ~ # mount /dev/sda3 /mnt
root@archiso ~ # _
```

15. Проверяем предыдущие действия с помощью

# lsblk

```
root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 673.1M 1 loop /run/archiso/airootfs
sda 8:0 0 15G 0 disk
├─sda1 8:1 0 477M 0 part
├─sda2 8:2 0 3.7G 0 part [SWAP]
└─sda3 8:3 0 10.8G 0 part /mnt
sr0 11:0 1 846.3M 0 rom /run/archiso/bootmnt
root@archiso ~ #
```

16. Создаем каталог home в директории /mnt

# mkdir /mnt/home

```
root@archiso ~ # mkdir /mnt/home
root@archiso ~ # _
```

17. Теперь в нашу систему нужно установить ядро линукс, различные базовые пакеты и текстовый редактор neovim с помощью pacstrap

# pacstrap /mnt base linux linux-firmware sudo neovim

18. После установки пакетов нам необходимо создать файл fstab, он содержит в себе какие разделы монтируются при загрузке

```
# genfstab -U /mnt >> /mnt/etc/fstab
```

```
root@archiso ~ # genfstab -U /mnt >> /mnt/etc/fstab
root@archiso ~ #
```

19. Перейдем к корневому каталогу нашей системы

```
# arch-chroot /mnt
```

```
root@archiso ~ # arch-chroot /mnt
root@archiso /#
```

Теперь мы внутри системы, здесь у нас оболочка bash. Можем проверить это командой

```
# ps
```

```
root@archiso /# ps
  PID TTY          TIME CMD
   742 tty1      00:00:00 zsh
   5478 tty1      00:00:00 arch-chroot
   5490 tty1      00:00:00 unshare
   5491 tty1      00:00:00 bash
   5506 tty1      00:00:00 ps
root@archiso /# _
```

видим процесс bash

20. Первым делом создадим часовой пояс

```
# ln -sf /usr/share/zoneinfo/Europe/Moscow /etc/localtime
```

```
# hwclock --systohc
```

и проверим командой

```
# date
```

```
root@archiso /# ln -sf /usr/share/zoneinfo/Europe/Moscow /etc/localtime
root@archiso /# hwclock --systohc
root@archiso /# date
Sat Oct 16 12:13:46 MSK 2021
root@archiso /#
```

21. Далее займемся локализацией - установим российскую и английскую локаль. При помощи редактора nvim раскомментируем соответствующие строки файла local.gen

```
# nvim /etc/locale.gen
```

```
#en_US.UTF-8 UTF-8
#en_GB.UTF-8 UTF-8
#en_GB.ISO-8859-1
en_US.UTF-8 UTF-8
#en_US.ISO-8859-1
#en_ZA.UTF-8 UTF-8
#en_ZA.ISO-8859-1
```

```
#ru_RU.UTF-8
#ru_RU.UTF-8
#ru_RU.ISO-8859-2
#ru_RU.KOI8-R KOI8-R
ru_RU.UTF-8 UTF-8
#ru_RU.ISO-8859-5
#ru_UA.UTF-8 UTF-8
#ru_UA.KOI8-U
#ru_UA.UTF-8
```

удаляем символы # в нужных строчках, затем нажимаем Esc и пишем команду

```
:wq
```

, где w - означает write, а q - quit

```
etc/locale.gen [1]
root@archiso /# 11L, 9982C written
```

22. Запускаем скрипт locale-gen, который нам сгенерирует локаль

```
# locale-gen
```

```
root@archiso /# locale-gen
Generating locales...
  en_US.UTF-8... done
  ru_RU.UTF-8... done
Generation complete.
root@archiso /# _
```

23. Укажем язык в vconsole.conf

```
#nvim /etc/vconsole.conf
```

Запишем

```
LANG=en_US.UTF-8
```

24. Запишите имя компьютера (номер вашей группы и Ваш номер по списку в группе, например БББО-01-21-14) в файл hostname

```
# nvim /etc/hostname
```

25. Настроим файл hosts

```
# nvim /etc/hosts
```

Пропишем соответствие:

```
127.0.0.1    localhost
```

```
::1          localhost
```

127.0.1.1 5550-01-21-14.localdomain 5550-01-21-14

```
# Static table lookup for hostnames.
# See hosts(5) for details
127.0.0.1    localhost
::1         localhost
127.0.1.1    KBSP.localdomain KBSP_
```

26. Создадим пользователю root пароль с помощью команды

# passwd

Никаких букв и звездочек отображаться не будет

```
[root@archiso /]# passwd
New password:
Retype new password:
passwd: password updated successfully
[root@archiso /]# _
```

27. Теперь необходимо установить загрузчик Grub, для этого необходимо 2 пакета

# pacman -S grub efibootmgr

```
[root@archiso /]# pacman -S grub efibootmgr
resolving dependencies...
looking for conflicting packages...

Packages (3) efivar-37-4  efibootmgr-17-2  grub-2:2.06-2

Total Download Size:    6.89 MiB
Total Installed Size:  34.45 MiB

:: Proceed with installation? [Y/n] y
:: Retrieving packages...
grub-2:2.06-2-x86_64             6.8 MiB   422 KiB/s   00:16 [#####] 100%
efivar-37-4-x86_64             110.5 KiB 502 KiB/s   00:00 [#####] 100%
efibootmgr-17-2-x86_64         27.4 KiB  200 KiB/s   00:00 [#####] 100%
Total (3/3)                    6.9 MiB   413 KiB/s   00:17 [#####] 100%
(3/3) checking keys in keyring
(3/3) checking package integrity
(3/3) loading package files
(3/3) checking for file conflicts
(3/3) checking available disk space
:: Processing package changes...
(1/3) installing grub
:: Generate your bootloader configuration with:
grub-mkconfig -o /boot/grub/grub.cfg
Optional dependencies for grub
  freetype2: For grub-mkfont usage
  fuse2: For grub-mount usage
  dosfstools: For grub-mkrescue FAT FS and EFI support
  efibootmgr: For grub-install EFI support [pending]
  libisoburn: Provides xorriso for generating grub rescue iso using grub-mkrescue
  os-prober: To detect other OSes when generating grub.cfg in BIOS systems
  nttools: For grub-mkrescue FAT FS support
(2/3) installing efivar
(3/3) installing efibootmgr
:: Running post-transaction hooks...
(1/1) Arming ConditionNeedsUpdate...
[root@archiso /]# _
```

28. Создаем директорию /boot/efi

# mkdir /boot/efi

Примонтируем директорию sda1 в /boot/efi

# mount /dev/sda1 /boot/efi

```
[root@archiso /]# mount /dev/sda1 /boot/efi
[root@archiso /]# _
```

29. Выполним команду

# grub-install

```
[root@archiso /]# grub-install
Installing for x86_64-efi platform.
Installation finished. No error reported.
[root@archiso /]# _
```

Если ошибок не обнаружено, продолжаем дальше.

30. Запустим скрипт автоматической конфигурации загрузчика

# grub-mkconfig -o /boot/grub/grub.cfg

```
[root@archiso /]# grub-install
Installing for x86_64-efi platform.
Installation finished. No error reported.
[root@archiso /]# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-linux
Found initrd image: /boot/initramfs-linux.img
Found fallback initrd image(s) in /boot: initramfs-linux-fallback.img
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
[root@archiso /]#
```

31. Далее нам необходимо установить ПО для работы с сетью, установим пакет networkmanager

# pacman -S networkmanager

и подключим демон (аналог службы windows) для автоматической подгрузки при старте системы

# systemctl enable NetworkManager

```

root@archiso /]# systemctl enable NetworkManager
Created symlink /etc/systemd/system/multi-user.target.wants/NetworkManager.service → /usr/lib/systemd/system/NetworkManager.service.
Created symlink /etc/systemd/system/dbus-org.freedesktop.nm-dispatcher.service → /usr/lib/systemd/system/NetworkManager-dispatcher.service.
Created symlink /etc/systemd/system/network-online.target.wants/NetworkManager-wait-online.service → /usr/lib/systemd/system/NetworkManager-wait-online.service.
root@archiso /]# _

```

32. Итак, мы закончили с конфигурацией, вводим команду

# exit

```

root@archiso /]# exit
exit
arch-chroot /mnt 6.09s user 3.69s system 0% cpu 1:05:19.90 total
root@archiso ~ # _

```

отмонтируем sda 3 из /mnt

# umount -R /mnt

и перезагрузим систему командой

# reboot

33. После перезагрузки вводим логин root и ваш пароль

```

Arch Linux 5.14.12-arch1-1 (tty1)

KBSP login: root
password:
root@KBSP ~]#

```

Поздравляю, вы в системе!

34. Можем обновить пакеты командой

# pacman -Syuu

```

root@KBSP ~]# pacman -Syuu
:: Synchronizing package databases...
   core                               136.8 KiB   134 KiB/s   00:01 [#####] 100%
  extra                               1570.2 KiB  538 KiB/s   00:03 [#####] 100%
community                             5.8 MiB   458 KiB/s   00:13 [#####] 100%
:: Starting full system upgrade...
there is nothing to do
root@KBSP ~]# _

```

35. Установим программу neofetch, она показывает информацию о системе

# pacman -S neofetch

Запускаем

# neofetch

```

root@KBSP ~]# neofetch

      .o+
      /ooo/
      +ooooo:
      +ooooooo:
      -+ooooooo+:
      \:-:++oooo+:
      /+++//+++++:
      /++++//+++++:
      \++0000000000000000/
      /000ssso++osssso+/
      .osssso+"""/osssso+
      -osssso.      :ssssso.
      :osssso/      osssso+++
      /osssso/      +sssooo/-
      /osssso+/-    -:/+osssso+
      `sso+:-      .-/os0:
      ++:
      .

root@KBSP ~]# _

root@KBSP
OS: Arch Linux x86_64
Host: VirtualBox 1.2
Kernel: 5.14.12-arch1-1
Uptime: 5 mins
Packages: 149 (pacman)
Shell: bash 5.1.8
Resolution: 1024x768
Terminal: /dev/tty1
CPU: AMD Ryzen 7 5700U with Radeon Graphics (4) @ 1.796GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 103MiB / 3913MiB

```