Software

Tools to aid you

- Spell checking
- Fonts
- Polkit
- Firefox
- Google Chrome
- Apostrophe
- Discord
- Blu-ray
- Node.js (nvm)
- KVM
- Folding@Home
- Timeshift
- Snapper

Spell checking

Hunspell is a spell checker and morphological analyzer library used by Firefox, Thunderbird, Chromium, LibreOffice and more.

Install the following packages to enable system-wide spell checking and hyphenation support (add languages for `hunspell` and `hyphen` at your discretion):

```
pacman -S hunspell hunspell-de hunspell-en_US hyphen hyphen-de hyphen-en
```

Fonts

Emoji Fonts

```
pacman -S noto-fonts-emoji
```

Asian Language Fonts

```
pacman -S noto-fonts-cjk adobe-source-han-sans-otc-fonts adobe-source-han-serif-otc-fonts
```

Polkit

`polkit` is an application-level toolkit for defining and handling the policy that allows unprivileged processes to speak to privileged processes: It is a framework for centralizing the decision making process with respect to granting access to privileged operations for unprivileged applications.

Custom rules

Mount disks as user

Edit/create `/etc/polkit-1/rules.d/50-udisk.rules`

```
// Original rules: https://github.com/coldfix/udiskie/wiki/Permissions
// Changes: Added org.freedesktop.udisks2.filesystem-mount-system, as this is used by Dolphin.

polkit.addRule(function(action, subject) {
  var YES = polkit.Result.YES;
  // NOTE: there must be a comma at the end of each line except for the last:
  var permission = {
    // required for udisks1:
    "org.freedesktop.udisks.filesystem-mount": YES,
    "org.freedesktop.udisks.luks-unlock": YES,
    "org.freedesktop.udisks.drive-eject": YES,
    "org.freedesktop.udisks.drive-detach": YES,
    // required for udisks2:
    "org.freedesktop.udisks2.filesystem-mount": YES,
    "org.freedesktop.udisks2.encrypted-unlock": YES,
    "org.freedesktop.udisks2.eject-media": YES,
    "org.freedesktop.udisks2.power-off-drive": YES,
    // Dolphin specific
    "org.freedesktop.udisks2.filesystem-mount-system": YES,
    // required for udisks2 if using udiskie from another seat (e.g. systemd):
    "org.freedesktop.udisks2.filesystem-mount-other-seat": YES,
    "org.freedesktop.udisks2.filesystem-unmount-others": YES,
    "org.freedesktop.udisks2.encrypted-unlock-other-seat": YES,
```

```
    "org.freedesktop.udisks2.eject-media-other-seat": YES,
    "org.freedesktop.udisks2.power-off-drive-other-seat": YES
  };
  if (subject.isInGroup("storage")) {
    return permission[action.id];
  }
});
```

Firefox

Install Firefox via these packages (adjust for your desired locale):

```
pacman -S firefox firefox-i18n-de
```

Wayland

Wayland is not yet the default display manager for Firefox (it falls back to XWayland on Wayland). To force Firefox to use Wayland you can set the `MOZ_ENABLE_WAYLAND` environment variable to `1`. Use user specific systemd environment variable configs to set it:

```
echo "MOZ_ENABLE_WAYLAND=1" >> ~/.config/environment.d/moz_wayland.conf
```

Media Playback

Autoplay in background

Firefox prevents autoplay for media of tabs that aren't currently active, which causes apps like Plex to take very long to skip to the next track after the current one has ended. The following setting in `about:config` can be used to disable this behavior:

Setting key	Value	Description
`media.block-autoplay-until-in-foreground`	`false`	Enable autoplay when tab is not currently active

Hardware Decoding

Utilizing GPU hardware accelerated decoding of video content results in smoother playback of HD/4K content, while reducing CPU load and power draw (important to save on battery on laptops).

To ensure Firefox uses hardware decoding ensure the following:

- The necessary VA-API drivers are installed (see: Graphics Cards)

- Navigate to `about:support` and ensure that under *Compositing* it says *WebRender* (*WebRender Software* will **not** work)
- Navigate to `about:config` and set `media.ffmpeg.vaapi.enabled` to `true`
- If running Wayland, enable Wayland mode in Firefox (see above)

Verify hardware video decoding

To verify Firefox is actually using VA-API to decode video you can launch it with the following command:

```
MOZ_LOG="FFmpegVideo:5" firefox 2>&1 | grep 'VA-API'
```

Start playing some video in Firefox and watch the logs on your terminal. If your log output reads something like the following video decoding via VA-API is working.

```
[RDD 97685: MediaPDecoder #1]: D/FFmpegVideo FFVPX: Initialising VA-API FFmpeg decoder
[RDD 97685: MediaPDecoder #2]: D/FFmpegVideo FFVPX:  VA-API FFmpeg init successful
[RDD 97685: MediaPDecoder #2]: D/FFmpegVideo FFVPX: Choosing FFmpeg pixel format for VA-API video decoding.
[RDD 97685: MediaPDecoder #1]: D/FFmpegVideo FFVPX:  VA-API FFmpeg init successful
[RDD 97685: MediaPDecoder #2]: D/FFmpegVideo FFVPX: VA-API Got one frame output with pts=0 dts=0 duration=40000 opaque=-9223372036854775808
[RDD 97685: MediaPDecoder #1]: D/FFmpegVideo FFVPX: Initialising VA-API FFmpeg decoder
[RDD 97685: MediaPDecoder #1]: D/FFmpegVideo FFVPX:  VA-API FFmpeg init successful
[RDD 97685: MediaPDecoder #1]: D/FFmpegVideo FFVPX: VA-API Got one frame output with pts=40000 dts=40000 duration=40000 opaque=-9223372036854775808
[RDD 97685: MediaPDecoder #2]: D/FFmpegVideo FFVPX: VA-API Got one frame output with pts=80000 dts=80000 duration=40000 opaque=-9223372036854775808
[RDD 97685: MediaPDecoder #2]: D/FFmpegVideo FFVPX: VA-API Got one frame output with pts=120000 dts=120000 duration=40000 opaque=-9223372036854775808
```

KDE Plasma Integration

For better integration of Firefox into the KDE Plasma desktop, install the Plasma Integration add-on either via the Mozilla Add-on page. It enables rich notifications support and download progress integration into the notification area of KDE Plasma.

Media Playback Controls

To prevent duplicate entries in the Media Player widget or tray icon, set `media.hardwaremediakeys.enabled` to `false`. This disables the media entry from Firefox itself and only uses the one from the Plasma integration add-on.

KDE Dialogs

By default, Firefox uses GTK file and print dialogs, even on KDE. To change this to KDE native dialogs navigate to `about:config` and change the appropriate `widget.use-xdg-desktop-portal` settings to `1`.

Google Chrome

Install Google Chrome from AUR:

```
yay -S google-chrome
```

Tweaks

To enable hardware accelerated video decoding (with open source drivers) create a file at `~/.config/chrome-flags.conf` and add the following line in it:

```
--enable-features=VaapiVideoDecoder
```

Additionally, if you need to be able to share your screen wie WebRTC, you need to add the following line as well:

```
--enable-usermedia-screen-capturing
```

Furthermore, visit chrome://flags and set the following options to further tweak performance (use the search field to filter):

Setting key	Value	Description
`#enable-webrtc-pipewire-capturer`	`Enabled`	Uses PipeWire to capture the screen in Wayland sessions
`#enable-gpu-rasterization`	`Enabled`	Uses GPU for rasterization, boosting performance
`#enable-zero-copy`	`Enabled`	Accesses GPU memory directly, boosting performance
`#ozone-platform-hint`	`Auto`	Auto-detects which windowing system is currently in use (X11, Wayland)

Apostrophe

Apostrophe is a simple and distraction-free Markdown editor with a minimalistic interface, live preview and export capabilities into many different formats.

Install Apostrophe from the AUR:

```
yay -S apostrophe
yay -S --asdeps texlive-latexextra mathjax # for PDF export and formula rendering
```

Discord

Discord is a proprietary, cross-platform, all-in-one voice and text chat application.

Install Discord from the repositories:

```
pacman -S discord
```

Blu-ray

Playback

In order play Blu-Rays install the following packages:

```
sudo pacman -S libbluray libaacs
```

Additionally, a `KEYDB.cfg` file is needed. Download it from the FindVUK Online Database

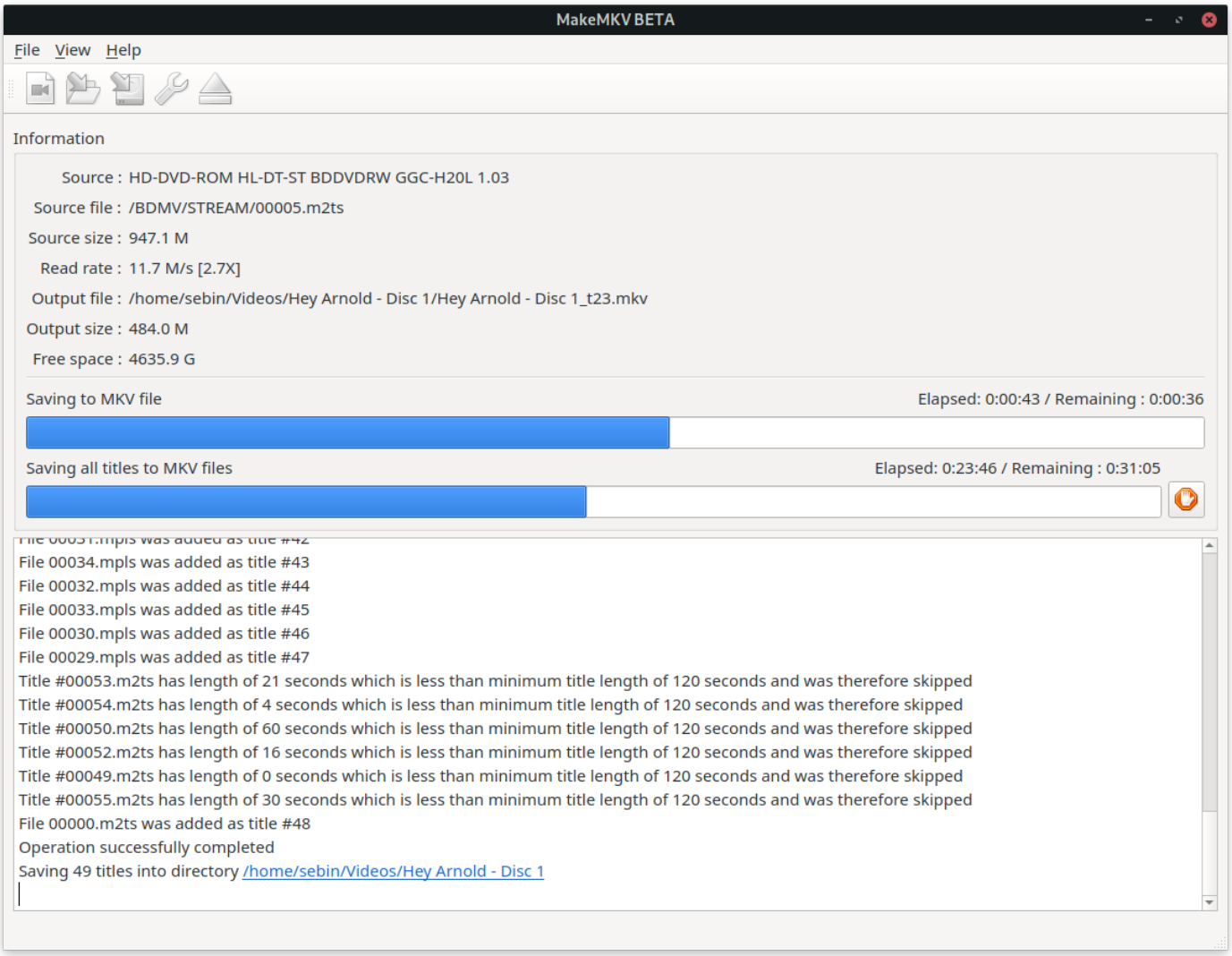
Extract the ZIP to `~/.config/aacs/`:

> **ATTENTION:** You may need to rename the `keydb.cfg` file to `KEYDB.cfg` (lower to upper case) for tooling to find it.

```
unzip keydb_eng.zip -d ~/.config/aacs/
```

After that use any Blu-Ray capable playback software, e.g. `vlc bluray:///dev/sr0` to play back Blu-Rays.

Ripping



In order to rip Blu-Rays install MakeMKV from the AUR:

```
yay -S makemkv
```

MakeMKV requires the `sg` (*SCSI generic (sg) driver*) kernel module to be loaded in order to recognize the drive. To load the module temporarily:

```
sudo modprobe sg
```

To have the kernel load the module on each boot:

```
sudo echo sg > /etc/modules-load.d/sg.conf
```

Node.js (nvm)

Use the Node Version Manager (`nvm`) to install Node.js into your current user's path and switch Node.js versions on the fly.

Install `nvm` via the AUR:

```
yay -S nvm
```

Include the init script `/usr/share/nvm/init-nvm.sh` into your shell configuration to load it each time you start your terminal:

```
# bash
echo 'source /usr/share/nvm/init-nvm.sh' >> ~/.bashrc


# zsh
echo 'source /usr/share/nvm/init-nvm.sh' >> ~/.zshrc
```

Restart your terminal to reload all init scripts and you should be able to use `nvm` to install a Node.js version of your choice:

```
nvm install 12
```

Migrating globally installed `npm` packages

When you install and switch to a different `nvm` managed version of Node.js (`nvm install 14` or `nvm use 16`) you may find that your globally installed `npm` packages (e.g. `svgo`) are no longer available until you switch back to the specific version of Node.js you have been using before the upgrade or switch.

This is because globally installed `npm` packages are installed for the specific version of Node.js you happen to be using at the time of installation and placed in a directory i.e. `~/.nvm/versions/node/v16.14.0/lib/node_modules`. When you install a different version, e.g. `17.2.0` the path to your Node.js installation changes to `~/.nvm/versions/node/v17.2.0/lib/node_modules`.

Use the `--reinstall-packages-from=<version>` option to carry over globally installed packages to the new Node.js installation.

You can either pass a specific version you want to reinstall globally installed packages from or use bash string expansion to reinstall from the currently active one in use:

```
nvm install <new version> --reinstall-packages-from=<old version>


nvm install 17 --reinstall-packages-from=$(node -v)
```

KVM

KVM (Kernel-based Virtual Machine) is a hypervisor built into the Linux kernel.

1. Install `libvirt` Packages

   ```
   yay -S qemu libvirt edk2-ovmf virt-manager nfs-utils virtio-win


   # optional dependencies
   iptables-nft dnsmasq   # for default NAT/DHCP networking
   bridge-utils           # for bridged networking
   openbsd-netcat         # for remote management over SSH
   ```

2. Add user to `libvirt` groups

   ```
   sudo usermod -aG libvirt $USER
   ```

3. Start `libvirtd` daemon

   ```
   sudo systemctl enable --now libvirtd
   ```

4. Create network bridge with `nmcli`

   ```
   nmcli connection add type bridge ifname br0 con-name "Netzwerkbrücke" stp no
   nmcli connection add type bridge-slave ifname enp39s0 con-name "Ethernet" master br0
   nmcli connection down "Kabelgebundene Verbindung 1"
   nmcli connection up "Netzwerkbrücke"
   ```

   - When using bonding of interfaces, disable IPv4 and IPv6 on the **bridge**

     ```
     nmcli con mod "Netzwerkbrücke" ipv4.method disabled ipv6.method ignore
     ```

5. Define bridge network XML file, e.g. as `br0.xml`

   ```
   <network>
     <name>br0</name>
     <forward mode='bridge'/>
     <bridge name='br0'/>
   </network>
   ```

6. Add bridge network to `virt-manager`

```
virsh -c qemu:///system net-define br0.xml
virsh -c qemu:///system net-autostart br0
```

7. Disable COW on Btrfs (optional, recommended)

```
sudo chattr +C /var/lib/libvirt/images
```

8. Define a remote storage pool (e.g. remote ISO images) `remote-iso.xml`

```
<pool type="netfs">
  <name>iso</name>
  <source>
    <host name="dragonhoard"/>
    <dir path="/Download/Software/ISOs"/>
    <format type="auto"/>
  </source>
  <target>
    <path>/var/lib/libvirt/images/iso</path>
  </target>
</pool>
```

9. Add storage pool to `virt-manager`

```
virsh -c qemu:///system pool-define remote-iso.xml
virsh -c qemu:///system pool-autostart iso
```

10. Create the storage pool mountpoint

```
sudo mkdir -p /var/lib/libvirt/images/iso
```

Folding@Home

Help scientists studying Alzheimer's, Huntington's, Parkinson's, and SARS-CoV-2 by simply running a piece of software on your computer. Add your computer to a network of millions of others around the world to form the world's largest distributed supercomputer.

Installlation

```
yay -S foldingathome opencl-amd
```

Configuration

Run `FAHClient --configure` as `root` to generate a configuration file at `/etc/foldingathome/config.xml`:

```
cd /etc/foldingathome
FAHClient --configure
```

Then start/enable the `foldingathome.service` systemd unit. NVIDIA users should also enable the `foldingathome-nvidia.service` systemd unit.

Example Configuration

```
<config>
  <!-- Slot Control -->
  <power v='FULL'/>

  <!-- User Information -->
  <passkey v='1234567890'/>
  <team v='45032'/>
  <user v='Registered_User_Name'/>

  <!-- Folding Slots -->
  <slot id='0' type='CPU'/>
  <slot id='1' type='GPU'/>
```

```
</config>
```

Timeshift

Timeshift is a backup and restore tool utilizing either `rsync` or `btrfs` snapshots.

Installation

> **WARNING:** Timeshift's `btrfs` mode is limited to an "Ubuntu-style" subvolume layout, i.e. only `@` and `@home` subvolumes can be present.

Install Timeshift from the AUR:

```
yay -S timeshift
```

Configuration

Cron

Timeshift relies on cronjobs to periodically create snapshots. Start the cron daemon via:

```
sudo systemctl enable --now cronie
```

Additionally, create an empty `crontab` file:

> **ATTENTION:** If the `EDITOR` environment variable is not set `crontab` automatically calls `vi` to edit files. Either `export` it or supply it temporarily when calling `crontab`, e.g. `EDITOR=nano crontab -e` to use `nano` instead of `vi`.

```
sudo crontab -e
```

Timeshift first time launch wizard

> **ATTENTION:** The GUI wizard is buggy around `btrfs` and `lvm` setups. It will say no `btrfs` partitions were detected but lets you continue regardless if you select the device with the root `btrfs` partition.

Open Timeshift via the desktop launcher and complete the configuration wizard.

Automatic snapshots on system changes

In addition to Timeshift's periodic spanshots, `timeshift-autosnap` provides a `pacman` hook to create a manual snapshot every time packages are installed, upgraded or removed. It works with either `rsync` or `btrfs` mode of Timeshift.

Install `timeshift-autosnap` from the AUR:

> **TIP:** When using GRUB as your bootloader consider also installing `grub-btrfs` to include `btrfs` snapshots in GRUB boot options.

```
yay -S timeshift-autosnap
```

By default `timeshift-autosnap` only keeps 3 snapshots and deletes older ones. To change this, edit `/etc/timeshift-autosnap.conf` and either set `deleteSnapshots` to `false` or increase the number of `maxSnapshots`, for example:

```
skipAutosnap=false
deleteSnapshots=true
maxSnapshots=7
updateGrub=true
snapshotDescription={timeshift-autosnap} {created before upgrade}
```

Prevent excessive snapshotting when using `yay`

By default, when installing or updating multiple packages from the AUR, `yay` first builds a package and immediately calls `pacman` to install it, before building and installing the next one on its list. This also means that the `timeshift-autosnap` hook is triggered **for each individual AUR package** built by `yay`, **including dependencies also installed from the AUR.**

This can have undesireable side-effects:

- `yay` will reach the `maxSnapshots` limit very quickly when installing multiple packages from the AUR, causing `timeshift-autosnap` to delete snapshots you may have wanted to keep
- if `deleteSnapshots` is set to `false` the available space on the `btrfs` partition may quickly be exhausted with an absurd amount of snapshots taken

To prevent this it is recommended to configure `yay` to:

1. not remove make dependencies after successfully built packages are installed
2. build all AUR packages first, install them all later
3. install AUR packages together with regular repo packages

By calling `yay` with the `--save` parameter, any options passed to it will be saved in a configuration file, e.g.:

```
yay --noremovemake --batchinstall --combinedupgrade --save
```

Next time you use `yay` to install, upgrade or remove packages it will read the generated config file at `~/.config/yay/config.json` and apply the options automatically without having to specify them during use.

Snapper

Install `snapper`

```
pacman -S snapper snap-pac
```

Generate root config

```
snapper -c root create-config /
```

Create pacman hook for `/boot`

File: `/etc/pacman.d/hooks/50-bootbackup.hook`

```
[Trigger]
Operation = Upgrade
Operation = Install
Operation = Remove
Type = Path
Target = usr/lib/modules/*/vmlinuz

[Action]
Depends = rsync
Description = Backing up /boot...
When = PreTransaction
Exec = /usr/bin/rsync -a --delete /boot /.bootbackup
```