Artix/Arch Install (Runit)

First of all this is a good video guide:[1)] https://youtube.com/watch?v=nCc_4fSYzRA

Secondly this guide assumes you'll be installing runit as your init of choice, of course you don't have to follow that recommendation, but note that some commands/actions may be different depending on your init choice.

(Also this guide was last updated 2023-12-25 so beware if its currently a few years after this.)

Pre-steps

For sake of completenes:

- Grab a iso from artixlinux.org downloads page
- Flash to a usb
- Disable secure boot on the target machine (backup keys too)
- Plugin & boot (live passwd is 'artix')
- `$ su root`

Now for some hardware checks, most modern systems will be running some form of EFI for the system firmware, if the following command *fails* you **do not** have a efi system.

```
artix-live:[root]:~# ls /sys/firmware/efi/efivars/
```

However if it spits out a lotta rubbish then you **do** have an efi system.

Finally, you want to make sure you are connected to the internet, if its a wired connection, you should be good to go, but if you are on wifi, youll have to manually turn it and connect (at least in the xfce image).

Partitioning

Find your target drive with `lsblk`, generally sepaking it will *most likely* be `/dev/sda`

```
artix-live:[root]:~# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0    7:0    0  73.2M  1 loop /run/artix/sfs/livefs
loop1    7:1    0   1.5G  1 loop /run/artix/sfs/rootfs
sda      8:0    0 931.5G  0 disk
sdb      8:16   1  29.3G  0 disk
├─sdb1   8:17   1   1.6G  0 part /run/artix/bootmnt
└─sdb2   8:18   1     4M  0 part
```

In the above example `/dev/sda` will be the target drive for the install.

Now most systems when partitioning wll have the following set of partitions:

- `/boot` boot partition, 512MiB is the minimum size it should be, fs is FAT32 (required for efi systems)
- `/` root partition (most likely ext4)
- `[SWAP]` should be the same size as your system memory[2)] but I usually just set it to half my total mem size.

Now some systems *also* have 1 more more of the following partitions:

- `/home` to make hopping easier, all the user files are here
- `/var` dunno why, but probably becuause it has just a high a count of read/write to justify placement on another disk in certain cases

However this is my *usual* partition scheme

- `/` (40-80GB)
- `[SWAP]` ({total mem}/2)
- `/home` rest of disk

Now run `fdisk` on the drive to format. THe following is a creation of the boot partition, but the process is the same for literally everything else.

```
artix-live:[root]:~# fdisk /dev/sda

Welcome to fdisk (util-linux 2.38.1).
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 disklabel with disk identifier 0xcfc381f3.

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):
First sector (2048-1953525167, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-1953525167, default
1953525167): +512M

Created a new partition 1 of type 'Linux' and of size 512 MiB.
```

tips:

- p to list partitions
- q to quit **without** writing changes
- w to quit **and write changes**

When you finish, doulb check the scheme, then write changes. The below is the scheme for a efi system.

```
Command (m for help): p
Disk /dev/sda: 931.51 GiB, 1000204886016 bytes, 1953525168 sectors
Disk model: Samsung SSD 870
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xcfc381f3

Device     Boot     Start        End    Sectors  Size Id Type
/dev/sda1            2048    1050623    1048576  512M 83 Linux
/dev/sda2         1050624  126879743  125829120   60G 83 Linux
/dev/sda3       126879744 1953525167 1826645424  871G 83 Linux
```

Filesystems

With the above scheme, you can make the following filesystems on the partitions:

```
# if your system is efi, you *will* need a boot partition
# it is also *required* that it is a FAT32 filesystem
mkfs.fat -F32 /dev/sda1

# the rest of the disks get a normal filesystem like ext4
mkfs.ext4 /dev/sda2
mkfs.ext4 /dev/sda3
```

Mounting & OS Bootstrap

Next we need to mount the filesystems so that we can install the os

```
# first the root partition
mount /dev/sda2 /mnt

# them make the mountpoints for the other 2 fs
mkdir -p /mnt/{boot,home}
mount /dev/sda1 /mnt/boot
mount /dev/sda3 /mnt/home
```

Now time for the install part, you will run `basestrap` with the path to the root and a list of packages along with `base`, `base-devel`, and `linux`. The following other packages I reccomend installing:

- `linux-firmware`: all of your hardware blobs so that your system can run
- `runit`: the only good init system[3)]
- `elogind-runit`: elogind for above init
- `vim`: good editor
- `git`: source control for installing aur pkgs
- `librewolf`: webbrowser

```
basestrap /mnt base base-devel linux linux-firmware runit elogind-runit
```

Finally, before we chroot, we need to generate our fstab for boot filsystems.

```
fstabgen -U /mnt >> /mnt/etc/fstab
```

Artix Chroot

Now that we have a system, lets chroot in:

```
artix-chroot /mnt
```

Pacman mirrors setup

Edit the mirrorlist for pacman to get slight faster speeds:

```
vim /etc/pacman.d/mirrorlist
```

Locale Setup

Depends on where you live, but you need to link the timezone info. For example

```
ln -sf /usr/share/zoneinfo/America/New_York /etc/localtime
```

Now set the system clock to hardware. **NOTE:** If youre time is incorrect even after setting both the timezone and the hwclock. Go into your BIOS and make sure your system clock is synced up to **GMT**. If it is not, set it, as if its on anything else your clock will be wrong.

```
hwclock --systohc
```

Now finally setup the locale, first edit and find your locale (the us for example is en_US) uncomment **both** `UTF-8` & `ISO-8859-1` variants. Then run `locale-gen`.

```
vim /etc/locale.gen
locale-gen
```

Now edit `/etc/locale.conf` (this should be a new file. For example from locale.gen

```
echo "LANG=en_US.UTF-8" > /etc/locale.conf
```

Networking

First install some networking packages

```
pacman -S networkmanager networkmanager-runit
```

Then enable it for boot, runit of course:

```
ln -s /etc/runit/sv/NetworkManager/ /etc/runit/runsvdir/current
```

Now edit your machines hostname:

```
vim /etc/hostname
```

Finally the hosts file.

```
vim /etc/hosts
```

People put a variation of things here, but the following are **highly reccomended**[4)]

```
127.0.0.1    localhost
::1          localhost
```

I also like to put the following line in as well (not `$hostname` this should be the hostname in `/etc/hostname`)

```
127.0.0.1    ${hostname}.local $hostname
```

Bootloader

Now, its finally time to make our machine bootable, we need to install out bootloader `grub`. `os-prober` is required if you want to dual-boot and `efibootmgr` is required if you are installing an efi system.

```
pacman -S grub os-prober efibootmgr
```

Now we install grub ang generate the grub config

```
grub-install --target=x86_64-efi --efi-directory=/boot --bootloader-id=GRUB

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

Finally, set the root password:

```
passwd
```

Post-install

Once you finish the main install, there may be a couple of other things that you want to do with your system. There are a couple of things that I do with all of my systems.

Primary User

You'll probably want to setup the primary user for the system, usually this is more than enough.

```
useradd -mU $username -s /bin/bash
# for sudo privs
usermod -aG wheel $username
```

Universe Repository

Artix has some nice community-maintained packages in the upstream `universe` repository. This often contains software not in the arch `community` repository (as we will cover later), so its nice to have. However, this is *not* enabled by default, you will have to add it yourself. Append the following snippet to `/etc/pacman.conf`:

```
[universe]
Server = https://mirror.pascalpuffke.de/artix-universe/$arch
Server = https://mirrors.qontinuum.space/artixlinux-universe/$arch
Server = https://mirror1.cl.netactuate.com/artix/universe/$arch
Server = https://ftp.crifo.org/artix-universe/$arch
Server = https://artix.sakamoto.pl/universe/$arch
Server = https://mirror1.artixlinux.org/universe/$arch
Server = https://universe.artixlinux.org/$arch
```

Arch Repos

You can setup artix to also pull from the arch `extra`[5] for additional software not present on artix. You can install the arch mirrorlist with the `arch-mirrorlist` in the `universe` repo (and it should work if you followed the previous section). Then append the following line **AFTER** all the artix repo declarations in `/etc/pacman.conf`:

```
[extra]
Include = /etc/pacman.d/mirrorlist-arch
```

1)
The arch manual sucks really bad
2)
Some people recommend double, but I've never seen that in the wild
3)
You can also install whatever you want for an init
4)
Required imo
5)
Community and extra were recently merged together, this is the only repo that needs to be enabled

From:
https://memex.kyaruc.moe/ - **kyaruc memex**

Permanent link:
https://memex.kyaruc.moe/guide:artix_arch_install

Last update: **2026-05-14 Thu 23:46**