Installation

It is possible to install any init system (openrc, runit, s6, or dinit) from any ISO. Also, you can perform a base installation (as described in this article) from within any non-base ISO image, using the terminal; this way you can benefit from the auto-configured networking and web browser availability of our DE installation media. In fact, unless you are very experienced or doing a headless server installation, there is little reason to use the base images.

Contents

Fresh Artix installation from bootable media

Artix can either be installed through the console or the GUI installer. The GUI install is quite straightforward, we'll focus on the console installation procedure here. The installation images are confirmed to work on both BIOS and UEFI systems.

Set the keyboard layout

To check the available layout types:

```
ls -R /usr/share/kbd/keymaps
```

Then type the name of the layout without the extension. For example, to set the Spanish(Spain) layout, type:

```
loadkeys es
```

This only sets the selected keyboard layout in the current tty and only until reboot. To make the setting permanent, you have to edit `/etc/conf.d/keymaps` in the installed system. It's also useful to set up `/etc/vconsole.conf` - it may look like this:

```
FONT_MAP=8859-1_to_uni
FONT=lat1-16
KEYMAP=de-latin1
```

Partition your disk

Partition your hard drive (`/dev/sda` will be used in this guide) with **fdisk** or **cfdisk**, the partition numbers and order are at your discretion:

```
cfdisk /dev/sda
```

If you want to install side-by-side with other operating systems, you must make some space on the disk by resizing the existing partitions. You may use **gparted** for this, however detailed instructions are out of the scope of this guide. See the ArchWiki

NOTE: On UEFI systems with a GPT-partitioned disk, there must be an EFI system partition (ESP). The suggested size is around 512 MiB.

Format partitions

Next, format the new partitions, we will use ext4 in this example:

```
mkfs.ext4 -L ROOT /dev/sda2        <- root partition
mkfs.ext4 -L HOME /dev/sda3        <- home partition, optional
mkfs.ext4 -L BOOT /dev/sda1        <- boot partition, optional
mkswap -L SWAP /dev/sda4           <- swap partition
```

The -L switch assigns labels to the partitions, which helps referring to them later through `/dev/disk/by-label` without having to remember their numbers

If you are doing a UEFI installation, the ESP needs to be formatted as fat32.

```
mkfs.fat -F 32 /dev/sda1
fatlabel /dev/sda1 ESP
```

Mount Partitions

Now, activate your swap space and mount your partitions:

```
swapon /dev/disk/by-label/SWAP                  (if created)
mount /dev/disk/by-label/ROOT /mnt
mkdir /mnt/boot
mkdir /mnt/home
mount /dev/disk/by-label/HOME /mnt/home         (if created)
mount /dev/disk/by-label/BOOT /mnt/boot         (if created)
mkdir /mnt/boot/efi                             (UEFI)
mount /dev/disk/by-label/ESP /mnt/boot/efi      (UEFI)
```

Connect to the Internet

A working Internet connection is required and assumed. A wired connection is setup automatically, if found. Wireless ones must be configured by the user.

In contrast to systemd-based distributions, Artix uses traditional interface names, which were used before systemd. Here is an overview:

Description	Interface name
Loopback	lo
Ethernet	eth0, eth1...
Wireless	wlan0, wlan1...

If you want to connect through a wireless interface you should use a wireless connection manager, most likely ConnMan, which is already running. Alternatively, use wpa_supplicant (supporting WPA/WPA2) or iw (supporting WEP) and dhcpcd to set it up. See Network configuration/Wireless in the Arch wiki, substituting any systemd features with those pertaining to the used init. Verify your connection before you proceed:

```
ping artixlinux.org
```

Update the system clock

Activate the NTP daemon to synchronize the computer's real-time clock:

```
rc-service ntpd start
```

or

```
sv up openntpd
```

or

```
s6-rc -u change openntpd
```

or

```
dinitctl start ntpd
```

Install base system

Use **basestrap** to install the **base** and optionally the **base-devel** package groups and your preferred init (currently available: `openrc`, `runit`, `s6`, and `dinit`):

```
basestrap /mnt base base-devel openrc elogind-openrc
```

or

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

or

```
basestrap /mnt base base-devel s6-base elogind-s6
```

or

```
basestrap /mnt base base-devel dinit elogind-dinit
```

If you encounter errors, you can use the **-i** flag of **basestrap** ('interactive'). Example:

```
basestrap -i /mnt base
```

and you will be prompted to choose the respective **elogind**.

Install a kernel

Artix provides three kernels: `linux`, `linux-lts` and `linux-zen`, but you can use any other kernel you like ('-ck, -pf' etc). It is very recommended to install `linux-firmware` too. You can try not installing it, but some devices such as network cards may not work.

```
basestrap /mnt linux linux-firmware
```

or

```
basestrap /mnt linux-lts linux-firmware
```

Use **fstabgen** to generate `/etc/fstab`, use -U for UUIDs as source identifiers and -L for partition labels:

```
 fstabgen -U /mnt >> /mnt/etc/fstab        <- edit and verify,
also set root, swap, home and etc..
```

Check the resulting fstab for errors before rebooting. Now, you can chroot into your new Artix system with:

```
artix-chroot /mnt # formerly artools-chroot
```

Configure the base system

Configure the system clock

Set the time zone:

```
ln -sf /usr/share/zoneinfo/Region/City /etc/localtime
```

Run hwclock to generate /etc/adjtime:

```
hwclock --systohc
```

Note that this will default to UTC. If you use Windows and you want the time to be synchronized in both Artix and Windows, follow System_time#UTC_in_Windows:ArchWiki for instructions to enable UTC in there also.

Localization

Install a text editor of your choice (let's use **nano** here) and edit `/etc/locale.gen`, uncommenting the locales you desire:

```
pacman -S nano
nano /etc/locale.gen
```

Generate your desired locales running:

```
locale-gen
```

To set the locale systemwide, create or edit `/etc/locale.conf` (which is sourced by `/etc/profile`) or `/etc/bash/bashrc.d/artix.bashrc` or `/etc/bash/bashrc.d/local.bashrc`; user-specific changes may be made to their respective `~/.bashrc`, for example:

```
export LANG="en_US.UTF-8"     <-- localize in your languages
export LC_COLLATE="C"
```

Boot Loader

First, install **grub** and **os-prober** (for detecting other installed operating systems):

```
pacman -S grub os-prober efibootmgr
grub-install --recheck /dev/sda
# for BIOS systems
grub-install --target=x86_64-efi --efi-directory=/boot/efi --
bootloader-id=grub   # for UEFI systems
grub-mkconfig -o /boot/grub/grub.cfg
```

NOTE: On GPT-partitioned drives, refer to ArchWiki:Grub article.

Alternatively, you can use some other bootloaders or even the own kernel using EFISTUB. Bear in mind that we won't cover them here since most are not supported officially and the Arch Wiki already provides enough documentation. See the ArchWiki:Bootloader article if you want to view the alternatives. If you have an Intel or AMD CPU, enable microcode updates in addition.

Add user(s)

First, set the root passwd:

```
passwd
```

Second, create a regular user and password:

```
useradd -m user
passwd user
```

Network configuration

Create the hostname file:

```
nano /etc/hostname
myhostname
```

Now add matching entries to hosts:

```
nano /etc/hosts
127.0.0.1        localhost
::1              localhost
127.0.1.1        myhostname.localdomain  myhostname
```

If the system has a permanent IP address, it should be used instead of 127.0.1.1.

If you use OpenRC you should add your hostname to /etc/conf.d/hostname too:

```
hostname='myhostname'
```

And install your prefered DHCP client

```
pacman -S dhcpcd or dhclient
```

If you want to use a wireless connection, be sure to also install **wpa_supplicant**.

Note: In Runit, s6, and dinit, enabling a service by default at this step requires a *different* command than the normal one because the init systems rely on a /run (a tmpfs) to be created. That creation occurs when you actually boot into the new system. You can choose to skip these steps and and enable the services *after* a reboot if you

wish using the commands listed on the Runit, s6, and dinit pages. Just be sure you have an internet daemon and its respective service script installed first.

In this example, we'll use **connman** and assume a GTK-based DE. For Qt-based DEs, the GUI is not included in the Artix repositories. Instead, there is a program called **cmst** (for LXQt, also **lxqt-connman-applet**) in AUR. It is the user's responsibility to manage installing programs from AUR, if desired. Plasma has a built-in network indicator, but unfortunately it only supports NetworkManager. Using several different network management programs at the same time is discouraged, as it can only lead to network issues.

OpenRC

Install **connman** and optionally a front-end:

```
pacman -S connman-openrc connman-gtk
rc-update add connmand
```

Runit

Install **connman** and optionally a front-end:

```
pacman -S connman-runit connman-gtk
ln -s /etc/runit/sv/connmand /etc/runit/runsvdir/default
```

s6

Install **connman** and optionally a front-end:

```
pacman -S connman-s6 connman-gtk
touch /etc/s6/adminsv/default/contents.d/connmand
s6-db-reload
```

dinit

install **connman** and optionally a front-end:

```
pacman -S connman-dinit connman-gtk
ln -s ../connmand /etc/dinit.d/boot.d/
```

Alternatively, if you will use `openrc`, Gentoo's **netifrc** modules can be used, these are located in /etc/init.d/ and work on a script-per-interface basis.

```
ip -s link                <- Get the exact name of your interface
nano /etc/conf.d/net        <- Add config_<interface>="dhcp"
```

Now the parent script `/etc/init.d/net.lo` should be symlinked to create additional scripts for each network interface and then loaded into an openrc runlevel.

```
ln -s /etc/init.d/net.lo /etc/init.d/net.<interface>
```

```
rc-update add net.<interface> default
```

Reboot the system

Now, you can reboot and enter into your new installation:

```
exit                          <- exit chroot environment
umount -R /mnt
reboot
```

Post installation configuration

Once shutdown is complete, remove your installation media. If all went well, you should boot into your new system. Log in as your root to complete the post-installation configuration. See Archlinux's general recommendations for system management directions and post-installation tutorials.

To get a graphical environment you need to install the xorg group:

```
pacman -S xorg
```

Read the Archlinux's Xorg wiki, for information on how Xorg chooses the best available video driver and which one is optimal for your hardware and how properly set Xorg server.

Desktop Environment

Install your favorite desktop environment, for example **KDE**, **GNOME**, **MATE**, **XFCE4** or **LXQt**:

```
pacman -S plasma kde-applications

pacman -S mate mate-extra system-config-printer blueman connman-gtk

pacman -S xfce4 xfce4-goodies

pacman -S lxqt

pacman -Sy moksha-artix  See Artix Linux: Moksha Desktop for Artix
```

See also: Artix Linux: Installation of XFCE4 *(might be outdated!)*

Display Login Manager

Each DM has its own openrc package, which brings openrc's DM setup in line with runit and s6 counterparts. Currently we support XDM, LightDM, GDM, SDDM and LXDM.

for example:

```
pacman -S sddm-openrc or sddm-runit or sddm-s6 or sddm-dinit
```

Be sure to add the service to start automatically at boot time.

Or you can use .xinitrc to launch your DE manually; edit (or create) `~/.xinitrc` and add **exec mate-session**. Notice: **mate-session** and quite a few other packages from the Arch repositories are compiled against systemd even if they don't actually use it, at least not as PID1; to satisfy the library link you may install **elogind** and its services files **elogind-openrc**, **elogind-runit**, **elogind-s6**, or **elogind-dinit**.

```
pacman -S elogind
```
