Migration

Contents

- Migration guide to Artix
  - Prepare the repositories
  - Clean up the packages cache
  - Lower Pacman security levels
  - Install the Artix PGP keyring
  - Restore Pacman security levels
  - Backup a list of your active daemons
  - Download the Artix packages
  - Remove systemd
  - Install your chosen init
  - ReInstall packages from Artix repositories
  - Install init scripts
  - Enable services
    - OpenRC
    - Runit
    - s6
    - dinit
  - Configure networking
  - LVM setups
    - OpenRC
    - Runit
    - s6
    - dinit
  - Remove more systemd cruft
  - Update the bootloader and the kernel initramfs
    - Recreate your initramfs and grub configuration file
    - Reinstall GRUB
  - Reboot

Migration guide to Artix

Note: Migration is for advanced users that know what they are doing. In general, doing a fresh installation is the preferred way to get started on Artix.

All commands below must be run as root.

Prepare the repositories

To begin, replace `pacman.conf` and the mirrorlist with Artix's:

```
 mv -vf /etc/pacman.conf /etc/pacman.conf.arch
 curl
https://gitea.artixlinux.org/packages/pacman/raw/branch/master/pac
man.conf -o /etc/pacman.conf
 mv -vf /etc/pacman.d/mirrorlist /etc/pacman.d/mirrorlist-arch
 curl https://gitea.artixlinux.org/packages/artix-
mirrorlist/raw/branch/master/mirrorlist -o
/etc/pacman.d/mirrorlist
 cp -vf /etc/pacman.d/mirrorlist /etc/pacman.d/mirrorlist.artix
```

If you had `multilib` enabled, don't forget to uncomment the following sets of lines in `/etc/pacman.conf`:

```
 [lib32]
 Include = /etc/pacman.d/mirrorlist

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

Clean up the packages cache

Clean all cache, because some of our own packages will have a different signature and pacman will complain. If you have a large cache and want to keep it, don't -Scc but you'll be asked to remove some corrupted packages and re-download. Then, force sync:

```
pacman -Scc && pacman -Syy
```

Lower Pacman security levels

To install the `artix-keyring` package, you need to temporarily lower pacman's security levels for the installation in your `/etc/pacman.conf`.

```
#SigLevel = Required DatabaseOptional
SigLevel = Never
```

With `Required DatabaseOptional` being the original options.

Install the Artix PGP keyring

To install artix-keyring, you must sign a master key manually:

```
pacman -S artix-keyring
pacman-key --populate artix
pacman-key --lsign-key 95AEC5D0C1E294FC9F82B253573A673A53C01BC2
```

Restore Pacman security levels

Do it now, or you may very well forget it:

```
SigLevel = Required DatabaseOptional
#SigLevel = Never
```

Backup a list of your active daemons

Save a list of your running systemd services, you'll need to install the respective init scripts afterwards.

```
systemctl list-units --state=running | grep -v systemd | awk
'{print $1}' | grep service > daemon.list
```

You are now ready for the installation of the Artix components, as well as the replacement init scripts for your systemd services.

Download the Artix packages

You should fetch the basic packages in the pacman cache (not really mandatory, but if you lose your internet connection in the middle of the update, you might be left without an init). Base and base-devel groups must also be reinstalled in order to be replaced by the ones in the [system] repo. Finally, choose one of Artix's inits (`openrc` , `runit` , `s6` , or `dinit`), and its init scripts.

```
pacman -Sw base base-devel grub linux linux-headers mkinitcpio \
  rsync lsb-release esysusers etmpfiles artix-branding-base
```

And run one of the following, depending on your init:

```
pacman -Sw openrc elogind-openrc openrc-system
```

or

```
pacman -Sw runit elogind-runit runit-system
```

or

```
pacman -Sw s6-base elogind-s6 s6-system
```

or

```
pacman -Sw dinit elogind-dinit dinit-system
```

Remove systemd

Now that the packages are cached, you can send systemd and its family to oblivion (answer 'yes' to all pacman questions below).

```
pacman -Rdd --noconfirm systemd systemd-libs systemd-sysvcompat
pacman-mirrorlist dbus
rm -fv /etc/resolv.conf
```

Restore the Artix mirrorlist, deleted by the previous operation:

```
cp -vf /etc/pacman.d/mirrorlist.artix /etc/pacman.d/mirrorlist
```

After you've done this you MUST also complete the next step, otherwise you're left without an init system. If you're doing this remotely (e.g. through ssh), please keep another ssh session open, just in case the first one freezes (it has happened, thank you systemd).

Install your chosen init

Finally, you can install the packages previously downloaded with `pacman -Sw`. Simple enough:

```
pacman -S base base-devel grub linux linux-headers mkinitcpio \
  rsync lsb-release esysusers etmpfiles artix-branding-base
```

And:

```
pacman -S openrc elogind-openrc openrc-system
```

or

```
pacman -S runit elogind-runit runit-system
```

or

```
pacman -S s6-base elogind-s6 s6-system
```

or

```
pacman -S dinit elogind-dinit dinit-system
```

Note: don't forget to install a network manager (e.g. connman or netifrc), a text editor and any other software that you consider vital.

ReInstall packages from Artix repositories

First, make sure your locale is C for `grep(1)` :

```
export LC_ALL=C
```

Replace all Arch packages with Artix ones:

```
pacman -Sl system | grep installed | cut -d" " -f2 | pacman -S -
```

```
pacman -Sl world | grep installed | cut -d" " -f2 | pacman -S -
pacman -Sl galaxy | grep installed | cut -d" " -f2 | pacman -S -
```

And in case you had 32-bit support enabled:

```
pacman -Sl lib32 | grep installed | cut -d" " -f2 | pacman -S -
```

Install init scripts

- Note: these init scripts are all optional and can be skipped.

Also, install and enable the initscripts for your running services, an example follows (replace `init` with either `openrc`, `runit`, `s6`, or `dinit`):

```
pacman -S --needed acpid-init alsa-utils-init cronie-init cups-init fuse-init haveged-init hdparm-init openssh-init samba-init syslog-ng-init
```

Enable services

OpenRC

```
for daemon in acpid alsasound cronie cupsd xdm fuse haveged hdparm smb sshd syslog-ng; do rc-update add $daemon default; done
```

You almost certainly need to enable *udev* unless you know what you're doing; *dbus* and *elogind* are no longer required to be explicitly enabled, as they are started on demand (OpenRC only).

```
rc-update add udev sysinit
```

Runit

```
for daemon in acpid alsasound cronie cupsd dbus elogind xdm fuse haveged hdparm smb sshd syslog-ng; do ln -s /etc/runit/sv/$daemon /etc/runit/runsvdir/default; done
```

s6

```
touch /etc/s6/adminsv/default/contents.d/{acpid,cronie,cupsd,elogind,xdm,fuse,haveged,hdparm,smbd,sshd,syslog-ng}
s6-db-reload
```

dinit

```
for daemon in acpid alsasound cronie cupsd xdm fuse haveged
```

```
hdparm smb sshd syslog-ng; do dinitctl enable $daemon; done
```

Configure networking

Edit your network configuration file, it's `/etc/conf.d/net`. This is especially important if you're converting a remote box, or you may very well be locked out of it. Depending on whether you use persistent device naming or not, you must symlink `/etc/init.d/net.lo` to `net.enp0s3` or `net.eth0` respectively. The interface names are used only as examples, make sure you find out what applies to you system. If in doubt (and have only got one ethernet interface), you can disable persistent device naming with a kernel command line (GRUB_CMDLINE_LINUX below) and go with *net.eth0*. OpenRC has its own network manager, **netifrc** which by default uses DHCP for the wired interface.

```
vi /etc/conf.d/net
echo 'GRUB_CMDLINE_LINUX="net.ifnames=0"' >>/etc/default/grub  # disable persistent device naming
ln -s /etc/init.d/net.lo /etc/init.d/net.eth0
rc-update add net.eth0 boot
pacman -S --needed netifrc
```

Create a basic `/etc/resolv.conf`, in the example we use Cloudflare's DNS:

```
nano /etc/resolv.conf
```

```
nameserver 1.1.1.1
```

LVM setups

If you have an LVM setup, you must install the *lvm2-init* and *device-mapper-init* packages, otherwise the logical volumes will be inactive after reboot. Both of these packages are part of the *init-system* group, so you probably have it installed already. Enable the services at boot:

OpenRC

```
rc-update add lvm boot
rc-update add device-mapper boot
```

Runit

```
ln -s /etc/runit/sv/lvm2 /etc/runit/runsvdir/default
ln -s /etc/runit/sv/device-mapper /etc/runit/runsvdir/default
```

s6

```
touch /etc/s6/adminsv/default/contents.d/device-mapper
s6-db-reload
```

(lvm2-s6 is already enabled on boot during install)

dinit

```
 dinitctl enable lvm2
 dinitctl enable dmeventd
```

Remove more systemd cruft

You can optionally remove some systemd junk accounts too:

```
 for user in journal journal-gateway timesync network bus-proxy
journal-remote journal-upload resolve coredump; do
   userdel systemd-$user
 done
 rm -vfr /{etc,var/lib}/systemd
```

Make sure you remove any 'init=/usr/lib/systemd/systemd' or similar directives from your bootloader config, as the linux kernel by default launches `/sbin/init`. Also, remove any 'x-systemd' directives from `/etc/fstab`.

Update the bootloader and the kernel initramfs

New *mkinitcpio* and *grub* were installed, so you better reconfigure them. Copy the new `/etc/mkinitcpio.pacnew` over to `/etc/mkinitcpio`, `/etc/default/grub.pacnew` to `/etc/default/grub`, recreate the kernel's initramfs with mkinitcpio and reinstall grub. If you had committed any changes to the original files (i.e. resume hook in `/etc/mkinitcpio.conf` or a custom kernel cmdline parameter in *grub*), you should merge them now.

Recreate your initramfs and grub configuration file

```
 mkinitcpio -p linux (the default Artix kernel) or mkinitcpio -P
(all installed kernels)
 update-grub
```

Reinstall GRUB

- For UEFI:

```
 grub-install --target=x86_64-efi --efi-directory=/boot/efi --
bootloader-id=grub (default for most setups)
 grub-install --target=x86_64-efi --efi-directory=esp_mount --
bootloader-id=grub (some users reported success with this one)
```

- For BIOS:

```
 grub-install /dev/sdX (replace sdX with sda, sdb, or whatever
your disk is)
```

Reboot

You won't be able to reboot normally, as the bloated PID1 binary is missing now. Do it with the kernel **SysRq trigger**.

```
sync
umount -a
mount -f / -o remount,ro
echo s >| /proc/sysrq-trigger
echo u >| /proc/sysrq-trigger
echo b >| /proc/sysrq-trigger
```

Some additional configuration may still be needed, especially with regards to desktop functionality. The configuration section at systemd-free.org will give you some ideas although elogind is used instead of consolekit now.