GNU/Linux Administration - Support #410

GNet Developer User Arch Installation

07/08/2014 09:13 AM - Daniel Curtis

Status:	Closed	**Start date:**	07/08/2014
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Workstation	**Estimated time:**	8.00 hours
Target version:	Arch Linux	**Spent time:**	12.00 hours

Description

Now that I have made the switch to Arch Linux more permanently, I decided to run through how I set my laptop up. The primary hard drive consists of 3 partitions:

- Windows XP (Dummy OS)
- Arch / (root)
- Arch /home (home)

The Windows partition installs its bootloader on the primary hard drive. The intention is to have the Arch boot off of a USB drive, but will only boot into Windows if the drive is not present.

The general software load out consists of:

- **VirtualBox w/ Guest Additions**: Virtual computing software
- **LibreOffice**: Office suite
- ~~**TrueCrypt**: High-grade encryption tool~~
- **Tomb**: Modern open source encryption management tool, replaces TrueCrypt
- **Windows Network Browsing**: For connecting to Windows shares
- **Firefox**: Open Source Web browser
- **Chromium**: Open Source Chrome Web browser
- **Chrome**: Google Proprietary Chrome Web browser
- **Thunderbird**: Mail client
- **ownCloud Client**: Personal cloud client
- **Pidgin**: Instant messaging client
- **BleachBit**: Browser, mail, application cleaning application
- **GIMP**: Image editing
- **Filezilla**: FTP/SFTP Client
- **git**: Source code management
- **KeePass2**: Password management
- **VLC**: Media player
- **Flash**: Closed source media and content plugin
- **Arduino**: Arduino Integrated Development Environment
- **Fritzing**: Prototyping software
- **PlayOnLinux**: Front-end to Wine
- **Steam**: Digital gaming distributor
- **Komodo Edit**: Open Source IDE
- **HuluDesktop**: Hulu client
- **Wireshark** : Network traffic analysis tool

Preparing the two partitions

I decided to use LUKS on both the root and home partitions.

- Format the partitions, if any custom options are wanted, this is where you would specify them.:

```
cryptsetup luksFormat /dev/sdb5
cryptsetup luksFormat /dev/sdb6
```

1. **NOTE**: This will prompt you for a passphrase to use for encrypting the partition. If I were truly paranoid I would use a keyfile with the -d flag and generate a 1K random keyfile:

```
dd if=/dev/urandom of=/path/to/keyfile bs=1K
```

- Now map open the LUKS partition to tad them to the device mapper:

  ```
  cryptsetup luksOpen /dev/sdb5 root
  cryptsetup luksOpen /dev/sdb6 home
  ```

- Next, create the filesystem for the containers. I chose BTRFS, though the choice in filesystem is user-preferential; I would like to try ZFS at some point.

  ```
  mkfs.btrfs /dev/mapper/root
  mkfs.btrfs /dev/mapper/home
  ```

- Mount the new encrypted partitions:

  ```
  mount /dev/mapper/root /mnt
  mkdir /mnt/home
  mkdir /mnt/boot
  mount /dev/mapper/home /mnt/home
  ```

Prepare the USB bootloader

This is one layer in my defense-in-depth, needing a USB with the bootloader installed onto. If I were a tad more paranoid, I would include the usage of a keyfile.

I usually add a 512MB ext4 partition to the beginning of a USB drive, this will be enough room for a few kernels. Using cfdisk will simplify the task:

```
cfdisk /dev/sdc
```

- Once the partition is created and formatted to the appropriate filesystem, mount the USB drive to the installation path /boot folder:

  ```
  mount /dev/sdc1 /mnt/boot
  ```

Install the base system

- Generate an fstab:

  ```
  genfstab -p /mnt >> /mnt/etc/fstab
  ```

- Now its time to install the base system:

  ```
  pacstrap /mnt base grub openssh
  ```

  - chroot into the newly installed system:

    ```
    arch-chroot /mnt
    ```

  - Write your hostname to /etc/hostname:

```
echo 'archdev' >> /etc/hostname
```

  - Symlink /etc/localtime to /usr/share/zoneinfo/Zone/SubZone:

```
ln -s /usr/share/zoneinfo/America/Los_angeles /etc/localtime
```

  - Uncomment the selected locale in /etc/locale.gen and generate it with:

```
vi /etc/locale.gen
:%s/#en_US.UTF-8/en_US.UTF-8
:wq
locale-gen
```

  - Configure /etc/mkinitcpio.conf as needed and create an initial RAM disk with:

```
mkinitcpio -p linux
```

  1. **NOTE**: Make site to add the ***encrypt*** word to the mkinitcpio.conf HOOKS section:

HOOKS="... encrypt ... filesystems ..."

- Set a root password:

```
passwd
```

- Configure the network again for newly installed environment:

```
cp /etc/netctl/examples/ethernet-dhcp /etc/netctl/wired
netctl enable wired.service
```

- Add the encrypted mapping to /etc/crypttab:

  home   /dev/mapper/home

- Enable SSH

```
systemctl enable sshd.service
```

Install the bootloader

- Before installing the booloader to the USB drive, the bootloader must be configured for the encrypted root partition. This can be done by making the following modification to /etc/default/grub:

  GRUB_CMDLINE_LINUX_DEFAULT="root=/dev/mapper/root cryptdevice=/dev/sda5:root quiet"

- Now install GRUB onto the USB drive:

```
grub-install --target=i386-pc --recheck --debug /dev/sdc
grub-mkconfig -o /boot/grub/grub.cfg
```

Exit the install environment and reboot

At this point the system will be bootable from the USB drive. Exit and reboot the out of the installation environment:

```
exit
umount /mnt/home
umount /mnt/boot
umount /mnt
reboot
```

Install a desktop environment

There are many choices for desktop environments, I went through a few before I returned to my favorite (LXDE). Here are a few popular ones just for reference:

- GNOME

  ```
  pacman -S gnome
  systemctl enable gdm.service
  systemctl start gdm.service
  ```

- KDE

  ```
  pacman -S kde
  systemctl enable kdm.service
  systemctl start kdm.sercice
  ```

- XFCE

  ```
  pacman -S xfce4 xorg xorg-xinit
  echo 'exec startxfce4' >> ~/.xinitrc
  startx
  ```

- LXDE

  ```
  pacman -S lxde xorg xorg-xinit dbus gvfs gvfs-smb
  echo 'exec startlxde' >> ~/.xinitrc
  startx
  ```

Add an administrator user

It is generally a good idea not to run command directly as root, but rather as an administrative user using the sudo wrapper command.

- First install sudo:

  ```
  pacman -S sudo
  ```

And create a user:

- [redacted]

- Add bob to the sudoers file:

  ```
  visudo
  ```

  bob ALL=(ALL) ALL

Install Wifi

- Since my laptop has a weird broadcom wifi card in it, I needed to use b43-fwcutter to install wireless drivers:

```
curl -LO http://downloads.openwrt.org/sources/broadcom-wl-4.178.10.4.tar.bz2
tar xjf broadcom-wl-4.178.10.4.tar.bz2
cd broadcom-wl-4.178.10.4/linux
sudo b43-fwcutter -w /lib/firmware wl_apsta.o
```

- Then install the wpa_supplicant package:

```
sudo pacman -S wpa_supplicant
```

- And finally create the netctl configuration, start and enable the network profile:

```
cp /etc/netctl/examples/wireless-wpa /etc/netctl/wireless-net
netctl enable wireless-net
netctl start wireless-net
```

Install the packages

- For the packages I require through the Arch repositories, I will just run with one command:

```
sudo pacman -S chromium firefox filezilla keepass vlc base-devel wget bleachbit calibre cifs-utils epdfview flashplugin geany gimp git gparted gqrx leafpad libreoffice mpv nmap pidgin playonlinux remmina rsync steam thunderbird virtualbox virtualbox-host-modules virtualbox-guest-iso virtualbox-host-dkms linux-headers wpa_supplicant_gui wireshark-cli wireshark-gtk handbrake openshot kdenlive dvdauthor
```

- For the packages I require through the AUR, I need to download the compressed PKGBUILD files and compile each package from source:

```
mkdir ~/src && cd ~/src
```

1. Google Chome

```
wget https://aur.archlinux.org/packages/go/google-chrome/google-chrome.tar.gz
tar google-chrome.tar.gz
cd google-chome
makepkg -s PKGBUILD && makepkg -i PKGBUILD
```

2. Tomb

```
wget https://aur.archlinux.org/packages/to/tomb/tomb.tar.gz
tar xzf tomb.tar.gz
cd tomb
makepkg -s PKGBUILD && makepkg -i PKGBUILD
```

3. qtkeychain-git (This is a dependency for the ownCloud client)

```
wget https://aur.archlinux.org/packages/qt/qtkeychain-git/qtkeychain-git.tar.gz
tar xzf qtkeychain-git.tar.gz
```

```
cd qtkeychain-git
makepkg -s PKGBUILD && makepkg -i PKGBUILD
```

4. caffeine-systray

   ```
   wget https://aur.archlinux.org/packages/ca/caffeine-systray/caffeine-systray.tar.gz
   tar xzf caffeine-systray.tar.gz
   cd caffeine-systray
   makepkg -s PKGBUILD && makepkg -i PKGBUILD
   ```

5. owncloud-client

   ```
   wget https://aur.archlinux.org/packages/ow/owncloud-client/owncloud-client.tar.gz
   tar xzf owncloud-client.tar.gz
   cd owncloud-client
   makepkg -s PKGBUILD && makepkg -i PKGBUILD
   ```

6. fritzing

   ```
   wget https://aur.archlinux.org/packages/fr/fritzing/fritzing.tar.gz
   tar xzf fritzing.tar.gz
   cd fritzing
   makepkg -s PKGBUILD && makepkg -i PKGBUILD
   ```

7. arduino

   ```
   wget https://aur.archlinux.org/packages/ar/arduino/arduino.tar.gz
   tar xzf arduino.tar.gz
   cd fritzing
   makepkg -s PKGBUILD && makepkg -i PKGBUILD
   ```

8. komodo-edit

   ```
   wget https://aur.archlinux.org/packages/ko/komodo-edit/komodo-edit.tar.gz
   tar xzf komodo-edit.tar.gz
   cd komodo-edit
   makepkg -s PKGBUILD && makepkg -i PKGBUILD
   ```

9. huludesktop

   ```
   wget https://aur.archlinux.org/packages/hu/huludesktop/huludesktop.tar.gz
   tar xzf huludesktop
   cd huludesktop
   makepkg -s PKGBUILD && makepkg -i PKGBUILD
   ```

Caveats & Notes

Prepare VirtualBox host

- The vboxdrv kernel module needs to be loaded:

  ```
  sudo modprobe vboxdrv
  ```

- Users also need to be added to the vboxusers group in order to use VirtualBox:

  ```
  gpasswd --add bob vboxusers
  ```

- Build DKMS modules

  ```
  dkms install vboxhost/4.3.14
  ```

- Enable DKMS modules at boot

  ```
  systemctl enable dkms.service
  ```

NOTE: Intel Galileo uses a different Arduino IDE

Since I am developing on the Intel Galileo, I needed to grab Intel's IDE from https://communities.intel.com/docs/DOC-22226

Building the above arduino package from the AUR may resolve issues if there are any problems installing the Intel version of the arduino IDE.

NOTE: Arduino and Galileo users must be added to the lock and uucp groups

The arduino board communicates with the computer via a serial connection or a serial over USB connection. So the user needs read/write access to the serial device file. Udev creates files in /dev/tts/ owned by group uucp so adding the user to the uucp group gives the required read/write access:

```
gpasswd -a $USER uucp
gpasswd -a $USER lock
```

Note: You will have to logout and login again for this to take effect.

The arduino board appears as /dev/ttyACMx so if the above doesn't work try adding the user to the group tty:

```
gpasswd -a $USER tty
```

Before uploading to the Arduino, be sure to set the correct serial port, board, and processor from the Tools menu.

Fix the corrupt text with Steam

Steam needs to have its own fonts, this can be installed by doing the following:

```
mkdir ~/SteamFonts && cd ~/SteamFonts
wget https://support.steampowered.com/downloads/1974-YFKL-4947/SteamFonts.zip
unzip SteamFonts.zip
sudo cp * /usr/share/fonts/TTF/
sudo chown -R root.root /usr/share/fonts/TTF/
```

tmpfs

- It is sometimes useful to offload the /tmp folder to RAM by using tmpfs:

  ```
  echo 'tmpfs   /tmp         tmpfs   nodev,nosuid,size=2G          0  0' >> /etc/fstab
  ```

Resources

- http://zfsonlinux.org/
- http://en.wikipedia.org/wiki/ZFS
- http://www.zfsbuild.com/2010/05/26/zfs-raid-levels/
- http://nex7.blogspot.ch/2013/03/readme1st.html
- http://wintelguy.com/raidcalc.pl

- https://calomel.org/zfs_raid_speed_capacity.html
- https://pthree.org/2012/04/17/install-zfs-on-debian-gnulinux/
- http://docs.oracle.com/cd/E19253-01/819-5461/

Related issues:

Related to GNU/Linux Administration - Support #447: The Paranoid Developer Ar...	**Closed**	**08/31/2014**
Copied to GNU/Linux Administration - Support #642: GNet Developer Arch Deskto...	**Closed**	**07/08/2014**

History

#1 - 07/08/2014 11:32 AM - Daniel Curtis

- Description updated

#2 - 07/08/2014 12:14 PM - Daniel Curtis

- Description updated

- % Done changed from 10 to 90

#3 - 07/08/2014 12:41 PM - Daniel Curtis

- Description updated

#4 - 07/08/2014 04:06 PM - Daniel Curtis

- Description updated

- Status changed from New to Resolved

#5 - 07/09/2014 07:47 AM - Daniel Curtis

- Description updated

#6 - 07/09/2014 07:58 AM - Daniel Curtis

- Description updated

#7 - 07/09/2014 08:22 AM - Daniel Curtis

- Description updated

#8 - 07/09/2014 08:26 AM - Daniel Curtis

- Description updated

#9 - 07/09/2014 08:53 AM - Daniel Curtis

- Description updated

#10 - 07/09/2014 10:23 AM - Daniel Curtis

- Description updated

#11 - 07/09/2014 10:36 AM - Daniel Curtis

- Description updated

#12 - 07/09/2014 11:57 AM - Daniel Curtis

- Description updated

#13 - 07/09/2014 12:09 PM - Daniel Curtis

- Description updated

- Status changed from Resolved to Closed

- % Done changed from 90 to 100

#14 - 07/09/2014 12:17 PM - Daniel Curtis

- Description updated

#15 - 09/04/2014 10:16 AM - Daniel Curtis

- Description updated

#16 - 09/04/2014 11:20 AM - Daniel Curtis

- Related to Support #447: The Paranoid Developer Arch Linux Setup added

#17 - 09/05/2014 11:04 AM - Daniel Curtis

- Description updated

#18 - 09/05/2014 12:31 PM - Daniel Curtis

- Description updated

#19 - 09/09/2014 12:42 PM - Daniel Curtis

- Description updated

#20 - 04/05/2015 10:53 AM - Daniel Curtis

- Target version set to Arch Linux

- Category set to Workstation

#21 - 05/25/2015 10:32 AM - Daniel Curtis

- Description updated

#22 - 05/25/2015 10:33 AM - Daniel Curtis

- Description updated

#23 - 08/21/2015 05:44 PM - Daniel Curtis

- Copied to Support #642: GNet Developer Arch Desktop Installation added