LiveCD Customization

Creating your own Linux distribution

Background….

- Do you think that the Ubuntu/Arch/Debian/Fedora default programs and settings are wrong? You can take a base system and customize it to your liking! We know this already.
- What if you want to package up your system and settings to share with others?
    - Create your own ‘version’ or ‘distribution’ of Ubuntu/Arch/Debian/Fedora, etc.
    - Have your own Window Manager, Desktop Environment, spash screens, logos, whatever
- Some required tools
    - debootstrap, syslinux, squashfs-tools, genisoimage (at least for Ubuntu based systems)
    - Also you will need patience and lots of time to read documentation

Creating repkamOS v1.0.0

Say we want to make our own distro with the following features....

- Based on Ubuntu 15.04 Vivid Vervet
- i3wm-based desktop
- thunar for file management (from xfce)
- lxappearance for gtk themes/icon changing (from lxde)
- Installable on other systems via a standard Live CD

Getting Started….

Three parts to think about.

- Host System
  - Our real system
  - Install: syslinux, squashfs-tools and genisoimage
- Disk Image
  - A folder on our host system that we will build up the files required for the iso
  - Contains the isolinux bootloader, config files, kernel from the chroot, and the squashfs
  - This exists outside of the chroot
- Chroot Environment
  - This is the system that will eventually run from the disk
  - Requires the Casper package be installed and all your software customizations

Overall Goals

1) Create a chroot and install your packages there
2) Compress the chroot system into a squashfs file
3) Create and configure the disk image which will have the bootloader (isolinux), the kernel, the compressed file-system image
4) Burn the CD / Test in Virtual Machine

This doesn't sound that bad so far!

Creating our chroot environment

```
ARCH="amd64"
RELEASE="vivid"

sudo apt-get install debootstrap
mkdir -p chroot
sudo debootstrap --arch=$ARCH $RELEASE chroot
```

ARCH is going to be the processor architecture we want to target. In this case amd64. The RELEASE is the specific Ubuntu version we're going to use for the base system. In this case, vivid is Ubuntu 15.04.

Chroot Tricks

Some packages inside the chroot will expect certain directories to exist and be populated, such as /dev/ and /proc/

We need to bind mount these otherwise some packages will fail! We can also copy in our host systems resolv.conf and sources.list files to get network working in the chroot

```
sudo mount --bind /dev chroot/dev
sudo cp /etc/resolv.conf chroot/etc/resolv.conf
sudo cp /etc/apt/sources.list chroot/etc/apt/sources.list
```

Working inside the chroot

Now we can run ‘sudo chroot chroot’ and enter into our new system!

There are still a few important configuration steps that we need to do inside of the chroot in order for some packages to work properly.

We need to mount /proc, /sys, and /dev/pts as well as set a HOME and LC_ALL environment variables. Now, install the basic packages that are required for the Live environment:

```
apt-get install --yes ubuntu-standard casper lupin-casper
apt-get install --yes discover laptop-detect os-prober
apt-get install --yes linux-generic
```

What are these required packages for Live CD?

- lupin-casper
  - Provides hooks to find an ISO image on a hard disk and to read a preseed file from a hard disk
- ubuntu-standard
  - Metapackage that depends on all the packages in the standard Ubuntu system
- os-prober
  - Utility to detect other OSes on a set of drives
- discover
  - a hardware identification system based on the libdiscover2 library

Working in the chroot, optional

We can now choose to install a graphical installer for our system. For Ubuntu this is the ubiquity package. This, of course, also requires an appropriate window manager or desktop environment be installed.

```
apt-get install ubiquity-frontend-gtk
apt-get install ubiquity-frontend-kde
```

This step is only needed if we want to actually install our image on other systems. If we just wanted to run in a live environment, we would not need this package.

Working in the chroot, customization time!

Now that we’ve got all the useless boilerplate stuff out of the way we can do the fun stuff! This is where we decide what we really want our system to have.

- Install i3wm, install lightdm, install custom greeters, add your boot splash screens, set default desktop wallpapers, and much much more!
- For some ‘quality of life’ features, I will also include things like consolekit, htop, vim, feh, curl, suckless-tools and remove dunst - my personal preference.
- Any other software such as email clients, web browsers, file managers, must be installed by you now. Otherwise you will have a very boring system :)

You can also take this time to sudo apt-get update/upgrade to make sure all packages are running the ‘latest’ version from the repos.

Working in the chroot, potential issues

Because Ubuntu uses dbus for lots of desktop configuration and generally expects some dbus related values to exist, we may also need:

```
apt-get install --yes dbus
dbus-uuidgen > /var/lib/dbus/machine-id
dpkg-divert --local --rename --add /sbin/initctl
```

apt will try and start services automatically when they are installed. This is not really what we want in a chroot environment, we just want the packages. We can trick apt!

Create /root/fake directory that contains symlinks to /bin/true for initctl, invoke-rc.d, restart, start, stop, start-stop-daemon, service. (remember to remove later!)

```
PATH=/root/fake:$PATH apt-get install some-service
```

Cleaning up the chroot

Now that we're doing setting up our new system, we need to clean up some extra files before we can package everything up. For our Ubuntu example this includes...

```
rm /var/lib/dbus/machine-id
rm /sbin/initctl
dpkg-divert --rename --remove /sbin/initctl
apt-get clean
rm -rf /tmp/*
rm /etc/resolv.conf
umount -lf /proc, umount -lf /sys, umount -lf /dev/pts
```

Now we can safely 'exit' back into our host system!

Starting the Disk Image

There are three packages that need to be installed on the Host System which provide the tools to make the CD image

- Syslinux contains isolinux which makes the CD bootable
- Squashfs-tools will compress the chroot into the image
- Genisoimage provides mkisofs tool to turn a directory into a CD image.

So, we can simply run: sudo apt-get install syslinux squashfs-tools genisoimage
At this time, we should also make a new directory called ‘image’ to work in as well as a few subdirectories in image: casper, isolinux, install

Folders, files, and Kernels

We will need a kernel and an initrd file that was built with the Casper scripts inside of our chroot environment. Grab them from your chroot!

```
cp chroot/boot/vmlinuz-[VERSION]-generic image/casper/vmlinuz
```

This is extremely important. vmlinuz is the name of the Linux kernel executable. This is a compressed version of the kernel and is used to load the system into memory so that the system can actually run.

“Virtual Memory LINUx gZip”

Folders, files, and Kernels

We will need a kernel and an initrd file that was built with the Casper scripts inside of our chroot environment. Grab them from your chroot!

```
cp chroot/boot/initrd.img-[VERSION]-generic image/casper/initrd.lz
```

This is extremely important. initrd is the name of the initial ramdisk file. This is a scheme for loading a temporary root file system into memory which may be used as part of the Linux startup process.

initrd and initramfs are two commonly used methods.

Folders, files, and Kernels

The next most important files are syslinux isolinux binaries. We can take these directly from our host system since we installed syslinux in a previous step.

```
cp /usr/lib/syslinux/isolinux.bin image/isolinux/
cp /boot/memtest86+.bin image/install/memtest
```

I did discover that these paths change depending on your host system setup. So your results may be slightly different here! (If you want to cheat a bit, you can probably also take these files from an existing Linux livecd iso....)

Bootloader Configuration

If we want to provide some ‘boot-time instructions’ to the user, we can create an isolinux.txt file in ‘image/isolinux’ folder. Something like:

```
splash.rle
*****************************************************
Welcome to the repkamOS Live CD Image! Select an option.
For the default live system, enter "live".  To run memtest86+, enter "memtest"
*****************************************************
```

Bootloader Configuration

Create an isolinux.cfg file in image/isolinux/ to provide configuration settings for the boot-loader. Now the CD should be able to boot!

```
DEFAULT live
LABEL live
  menu label ^Start or install repkamOS
  kernel /casper/vmlinuz
  append  file=/cdrom/preseed/ubuntu.seed boot=casper initrd=/casper/initrd.lz quiet splash --
DISPLAY isolinux.txt
TIMEOUT 300
PROMPT 1
```

Much more information about this can be found in /usr/share/docs/syslinux on your host system, but this is an example.

Generate the Software Manifest

We need to generate a list of all the software installed in our chroot system. Thankfully dpkg has the ability to do this with a quick script. If we install this system, these are the packages that will get installed. We want to remove certain packages from this list.

```
sudo chroot chroot dpkg-query -W --showformat='${Package} ${Version}\n' | sudo tee image/casper/filesystem.manifest
sudo cp -v image/casper/filesystem.manifest image/casper/filesystem.manifest-desktop

REMOVE='ubiquity ubiquity-frontend-gtk ubiquity-frontend-kde casper lupin-casper live-initramfs user-setup discover1
xresprobe os-prober libdebian-installer4'

for i in $REMOVE
do
        sudo sed -i "/${i}/d" image/casper/filesystem.manifest-desktop
done
```

Compressing the chroot environment

mksquashfs will perform the squashing and print the resulting number of inodes and size of data written, as well as the average compression ratio, etc. We just need to end up with a compressed version of our chroot.

```
sudo mksquashfs chroot image/casper/filesystem.squashfs
printf $(sudo du -sx --block-size=1 chroot | cut -f1) > image/casper/filesystem.size
```

If we want to ONLY be a LiveCD and not an installable disk, we can exclude the /boot folder to save space: mksquashfs chroot image/casper/filesystem.squashfs -e boot

More disk boilerplate

There are a few more files that we can add if we want this disk image to be supported by the Ubuntu disk creator and be recognized as an “Ubuntu Remix” image.

- image/README.diskdefines
- touch image/ubuntu
- mkdir image/.disk && cd image/.disk
- touch base_installable
- echo “full_cd/single” > cd_type
- echo “repkamOS v1.0.0 - Ubuntu 15.04” > info
- echo “https//repkam09.com/repkamOS/” > release_notes_url
- find . -type f -print0 | xargs -0 md5sum | grep -v "\./md5sum.txt" > md5sum.txt

Finally, create the iso image!

We can now create our iso image from the image/ directory that we have been building!

```
cd image
sudo mkisofs -r -V "repkamOS" -cache-inodes -J -l -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -
boot-load-size 4 -boot-info-table -o ../repkamOS-v1.0.0.iso .
```

Now you have a nice packaged up iso file that you can copy to a CD or install directly in Virtualbox, QEMU, or whatever else

Optional, Persistent Data partition

We can optionally create an ext2, ext3 or ext4 partition named "casper-rw" as a separate partition and append the word "persistent" to your "append" line configuration in the syslinux configuration and all your session data will be stored there.

```
LABEL live
  menu label ^Start or install Ubuntu
  kernel /casper/vmlinuz
  append  file=/cdrom/preseed/ubuntu.seed boot=casper persistent initrd=/casper/initrd.gz quiet splash --
```

You will be able to keep your changes between boots of the USB drive. This will not work for CD-R for obvious reasons.