Harmonic CableOS VCMTS Image Documentation

Customize the vendor-provided Debian In-Ram-Filesystem .iso

Variables and Defaults, Required package installation.

```
requiredPkgs=( "genisoimage" "mkisofs" "makefs" "mkinitramfs" "livecd-rootfs" "fakeroot" "live-build" )
: "${USERDATA:=/opt/userdata}"
: "${APOLLO_ISO:=APOLLO_PLATFORM-release-3.21.3.0-7+auto15.iso}"
: "${INSTALL_SCRIPT:=cableos-installer.sh}"
: "${ELTORITO:=stage2_eltorito}"
: "${LIVEIMG_URL:=https://gemmei.ftp.acc.umu.se/debian-cd/current/amd64/iso-cd/debian-12.5.0-amd64-netinst.iso}"
: "${LIVEIMG_ISO:=$(basename $LIVEIMG_URL)}"
: "${DEBIRF_ISO:=debirf-live_bullseye_6.0.0-0.deb11.6-amd64.iso}"
: "${WORKDIR:=${HOME}/cableos-live}"
: "${LIVEFS_DIR:=${WORKDIR}/${DEBIRF_ISO%.*}}"
: "${ROOTFS_DIR:=${LIVEFS_DIR}/rootfs}"

sudo apt-get -y install "${!requiredPkgs[@]}"
```

Step 1: Extract the ISO file

- Create a working directory.
- Unpack debirf minimal.tgz into working directory
- Mount the .iso file at /mnt .
- Copy the contents at /mnt/* to the newly created working directory.
- Unmount the .iso from /mnt .
- Change directories to the newly created working directory

```
mkdir "${WORKDIR}"
tar -xzvf "${DEBIRF_MINIMAL}" -C "${WORKDIR}"
sudo mount -o loop "${USERDATA}/${DEBIRF_ISO}" /mnt
cp -r /mnt/* "${WORKDIR}/"
sudo umount /mnt
cd "${WORKDIR}"
```

Step 2: Extract debirf-live.cgz

- Create the target folder for the live filesystem.
- Change directories to the previously created livefs directory.
- Extract the filesystem archive using zcat to read the contents of the .cgz file and pipe the output to cpio to extract them.
- Remove the .cgz file

```
(mkdir "${LIVEFS_DIR}" && cd "${LIVEFS_DIR}" && zcat
"${WORKDIR}/${DEBIRF_ISO%.*}.cgz" | cpio -idvm && rm -f
"${WORKDIR}/${DEBIRF_ISO%.*}.cgz")
```

Step 3: Extract roots.cxz

- Create a target folder for rootfs.
- Change directories to the previously created rootfs folder.
- Extract the rootfs using xzcat to read the contents of the .cxz file and pipe the output to cpio to extract them.
- Remove the .cxz file

```
(mkdir "${ROOTFS_DIR}" && cd "${ROOTFS_DIR}/" && xzcat
"${LIVEFS_DIR}/rootfs.cxz" | cpio -idvm && rm -f
"${LIVEFS_DIR}/rootfs.cxz")
```

Step 4: Add custom files

Here we are delaring an array of of files that will be copied into the rootfs directory with the following structure:

`key`: /path/to/source/file.ext

`value`: /path/to/rootfs/destination/dir/

The script below performs the following checks and actions.

For each file in the array :

- Check that the destination directory is not present

  AND

  - Create the destination directory

- Check that the source file exists

  AND

- (The destination file does not) **OR** (The source and destination files are not identical)

  AND

  - Copy the source file to the destination directory

```
declare -A filePaths
FILE1="${USERDATA}/${APOLLO_ISO}"
FILE2="${USERDATA}/${INSTALL_SCRIPT}"
filePaths["${FILE1}"]="${ROOTFS_DIR}/data"
filePaths["${FILE2}"]="${ROOTFS_DIR}/root"

for fileName in "${!filePaths[@]}"; do

    [[ ! -d "${filePaths[${fileName}]}" ]] \
    && echo -e "
    ${filePaths[${fileName}]} does not exist.
    mkdir -p ${filePaths[${fileName}]}
    " \
    && mkdir -p "${filePaths[${fileName}]}"

    [[ -e ${fileName} ]] && ([[ ! -e "${filePaths[${fileName}]}/$(basename
${fileName})" ]] || ! ( diff -q "${fileName}"
"${filePaths[${fileName}]}/$(basename ${fileName})" )) \
    && echo -e "
    cp ${fileName} ${filePaths[${fileName}]}/ \
    " \
    && cp "${fileName}" "${filePaths[${fileName}]}/"
done
```

Step 5: Repack the root filesystem

- Recreate the .cpio archive.
- Compress it back to .cxz archive

```
( cd "${ROOTFS_DIR}" && find . | cpio -o -H newc | xz -z -T0 >
"${LIVEFS_DIR}/rootfs.cxz" && rm -rf "${ROOTFS_DIR}" )
```

Step 6: Repack debirf-live.cgz

- Recreate the .cpio archive.

- Compress it back to .cgz archive.

```
( cd "${LIVEFS_DIR}" && find . | cpio -o -H newc | gzip -6 >
"${WORKDIR}/${DEBIRF_ISO%.*}.cgz" && rm -rf "${LIVEFS_DIR}" )
```

Step 7: Recreate the ISO

- Rebuild the .iso image from the updated working directory contents
- If successful, print image and md5sum
- If unsuccessful, print error notice and remove failed image

```
nsuccessful, print error notice and remove failed image

( mkisofs -R -b boot/grub/bios.img -no-emul-boot -boot-load-size 4 -boot-
info-table -c boot/grub/boot.cat -input-charset utf-8 -o
"${USERDATA}/REPACK-${DEBIRF_ISO}" . ) \
&& ( echo -e "
ISO Repack Completed Successfully.
New Image: ${USERDATA}/REPACK-${DEBIRF_ISO}
" \
&& md5sum ${USERDATA}/REPACK-${DEBIRF_ISO} | tee -a
${USERDATA}/REPACK-${DEBIRF_ISO}.md5sum ) \
|| ( echo -e "
ISO Repack Failed... Removing..
" && rm -f ${USERDATA}/REPACK-${DEBIRF_ISO} )
```

Complete Auto-Build Script

CableOS Auto-Build Script

The complete script will perform all of the steps listed above to:

- Create the directory structures
- Mount the original .iso,
- Copy the files into the proper locations and extrack theew op
- Extract them to the proper locations
- Add custom ISO and install script
- Repack the archives
- Recreate the ISO

Packer Conversion HCL2

```
// packer.pkr.hcl
  packer {
    required_plugins {
      qemu = {
        version = ">= 0.0.1"
        source  = "github.com/hashicorp/qemu"
} }
  }
  source "qemu" "debirf-live" {
Feedback
Help Settings
1
"type": "file",
"source": "/initrd.img",
"destination": "output-debirf-live/initrd.img"
"type": "shell-local",
"inline": [
  "qemu-img convert -f qcow2 -O raw new.qcow new.img",
  "maas admin boot-resources create name=custom/new name_title='New Image'
architecture=amd64/generic content@=new.img"
make build
iso_url = var.iso_url
iso_checksum = "none"
disk_size = "4096"
output_directory = var.output_directory
vm_name
format
accelerator http_directory = "http" boot_command =[

= var.iso_checksum
= 10240
= var.vm_name
= "qcow2"
= "kvm"
"<enter><wait>",
"linux /install/vmlinuz auto hostname=${var.vm_name} <wait>",
"initrd /install/initrd.gz <wait>",
"boot<enter>"
]
ssh_username
ssh_password
ssh_port
ssh_wait_timeout = "10000s"
headless        = false
= var.ssh_username
= var.ssh_password
= 22
build {
  sources = [
    "source.qemu.debirf-live"
  ]
  provisioner "shell" {
    inline = [
      "sudo ostree-production install --source=Apollo.iso --destination=/",
      "sudo cp /boot/vmlinuz* /vmlinuz",
      "sudo cp /boot/initrd.img* /initrd.img"
    ]
  }
```

```
    provisioner "file" {
      source      = "/vmlinuz"
      destination = "${var.output_directory}/vmlinuz"
  }
    provisioner "file" {
      source      = "/initrd.img"
      destination = "${var.output_directory}/initrd.img"
    }
    post-processor "qemu" {
      only
      output
      format
      disk_interface = "virtio"
      = ["qemu"]
      = var.new_qcow
      = "qcow2"
    }
    post-processor "shell-local" {
      inline = [
        "qemu-img convert -f qcow2 -O raw ${var.new_qcow} ${var.new_img}",
        "maas admin boot-resources create name=custom/new name_title='New
  Image' architecture=amd64/generic content@=${var.new_img}"
      ]
    }
  }
```