Abstract

The goals of this project:

1. Show you how to build ChromeOS from scratch (ChromeOS is what runs on Chromebooks)
2. Boot it in a VM
3. Boot it on a real Chromebook
4. Modify partition ROOT-B of the ChromeOS image to have very few binaries and base it on u-root (u-root.tk)Q S S
5. Restore the KERNEL-B kernel so it has graphics embedded in the kernel once more
6. Install a different browser in tzntz.the KERNEL-B that uses X11 and lets us run other X-based__/-/?/-<///4toolstests/.-?ztzyzzyztzTts/-/.-tzzytztTStzYZtz YET YYZT
7. Add upspin client code and try to mount upspin server (see upspin.io)
8. Rtst2tszz stttStS3ztTttttztts YsttstsST5sZSSstTAaatSsstasstsTaysy5t1AX s35ttsT6statstttsyat55tttty5stss4tTzyzTts3ztstyzDtztztstszt3t2tz3t3tts3tt3sttttfzZTz2tz3tz

Yz3ys3ssyystzys3ytsyfyy3st3t]st3ts3y3zryz3yza00Make it easy to login to nextcloud (see nextcloud.com) -- they will help us with whatever we need, including server resources, I've already made these connections.

Resources

1. Student's desktop for builds
2. "Celes" chromebook
3. "Payne" chromebook
4. "Lumpy" chromebook
5. "Panther" chromebox

One-time tasks

Install the Chrome VNC viewer

To see the image from the VM, you need the Chrome VNC viewer, get it here: https://goo.gl/L7Trkl.

Pick a build machine and get it ready

This follows the instructions from chromium.org with a few changes. Last tested May 2017.

Install depot tools (the user in this case being me, rminnich)

1. cd /home/rminnich
2. Install tools
    a. sudo apt-get install build-essential git-core gitk git-gui subversion curl python2.7
    b. You need to make sure that if you run python, it gets python2, not 3. I made a file in my ~/bin which is a symlink from ~/bin/python to /usr/bin/python2.7, and ~/bin is first in my search path, but do whatever you think works best. You can also alias python to python2.7 or even uninstall python3!.
3. git clone https://chromium.googlesource.com/chromium/tools/depot_tools.git
4. PATH=$PATH:/home/rminnich/depot_tools/
5. NOTE: MAKE SURE that the depot_tools directory is on the SAME file system (i.e mount point) as the place you put the development tree. Else you'll hit a bug when you repo sync (below) and it won't be obvious.

Fix sudo[1]

[1] http://www.chromium.org/chromium-os/tips-and-tricks-for-chromium-os-developers#TOC-Making-sudo-a-little-more-permissive

Now you've got the tools, you need to do a little sudo housekeeping:
You need to create a file in

/etc/sudoers.d
Called
relax_requirements
That file needs to contain the following lines, exactly as written here:
Defaults !tty_tickets
Defaults timestamp_timeout=180
Just those two lines.

Configure git

Now it's time to set up git. Change these to suit.

1. git config --global user.email "me@mymail.com"
2. git config --global user.name "John Q. Smith"

Set up the chroot and init the repo in it[2]

Now get ready to set up your build universe.

1. cd /home/rminnich
2. umask 022
3. mkdir chromiumos [AS ABOVE, MAKE SURE YOU DO THIS ON THE SAME FILE SYSTEM YOU PUT depot_tools ON]
4. cd chromiumos/
5. repo init -u https://chromium.googlesource.com/chromiumos/manifest.git --repo-url https://chromium.googlesource.com/external/repo.git
6. repo sync
7. cros_sdk

Will drop you into a chroot; wait while it does its thing. You *may* see errors relating to locale_gen, as chromeos build system is not portable; do they matter? Not sure.
Go ahead and type exit and leave the chroot. It's good to practice going back and forth.

Things you'll repeat.

The previous steps were setup. These following steps you'll repeat as you make changes and build new images. Here we show it for amd64-generic, although these steps are also tested for falco and peppy.

[2] https://www.chromium.org/chromium-os/developer-guide

Note: always do a repo sync. Daily. It pays. Also, ALWAYS, when you do these steps, make sure you're on a network. Even build_packages can do lots of network IO.

1. cd /home/rminnich
2. cd chromiumos
3. cros_sdk
4. repo sync

OK, let's build! And accept a license[3].

1. ACCEPT_LICENSE="*" USE="kvm" ./build_packages --board=amd64-generic --nowithautotest
2. ACCEPT_LICENSE="*" USE="kvm" ./build_image --board=amd64-generic dev --boot_args="disablevmx=off noinitrd lsm.module_locking=0"

Note the first line is build_packages and the second is build_image. USE=kvm is so you can run VMs like windows.

Change amd64-generic or whichever board you want to build for (falco, peppy, amd64-generic, etc). The first time ./build_packages is run for a new board will take a long time.

Use the following command to setup your board. "./setup_board --board=(insert your overlay)"

The lsm.module_locking=0 is required to fix broken behavior in the chromium OS kernel.

Create the VM image

./image_to_vm.sh --from=../build/images/amd64-generic/latest --board=amd64-generic
Note that 'latest' is a symlink to the latest build directory, which has a very long name starting with R60. l

Running in QEMU

```
$ qemu-system-x86_64 -version
QEMU emulator version 2.6.0, Copyright (c) 2003-2008 Fabrice Bellard
```

[3] https://www.chromium.org/chromium-os/licensing/building-a-distro

```
$ qemu-system-x86_64 -smp 8 --enable-kvm -vnc 127.0.0.1:0,ipv4 -m
4096 -usbdevice tablet -vga virtio -net nic,model=virtio -net
user,hostfwd=tcp:127.0.0.1:9222-:22 -drive
format=raw,file=/path/to/vm/image/chromiumos_qemu_image.bin
```

At this point, qemu will be waiting for a VNC connection. Start up the VNC viewer in your Chrome browser and navigate to 127.0.0.1 and use the port that qemu printed when it started.

Real hardware

The chromebooks I am providing are set up to do USB boot, from the boot screen, when you hit control-U. You need to build a USB boot stick.

Taking celes, as an example:

1. ACCEPT_LICENSE="*" USE="kvm" ./build_packages --board=celes --nowithautotest
2. ACCEPT_LICENSE="*" USE="kvm" ./build_image --board=celes dev --boot_args="disablevmx=off noinitrd lsm.module_locking=0"
3. Plug in the USB stick, find the device name and use it in place of dev/sdx below
4. cros flash usb:////dev/sdx celes/latest
5. Plug USB stick intoy screen.

Modify the image on the USB stick

This is where it gets interesting. We'll need to

1. Learn how to do a "one shot" boot where we use KERN-B and ROOT-B partitions
2. Re enable kernel console
3. Build a root file system with just console mode and u-root
4. Add x11 and a browser to that
5. Try to get it to boot
6. Try to sign it

But let's start with step 1.

Appendix

Re-signing a partition

From a recent discussion on chromeos-firmware, this is how we can resign a kernel partition when we've changed it. This is only an issue when we get to step 6, above.
lopsetup -P /dev/loop0 image.bin
vbutil_kernel --repack /dev/loop0p2 \

```
--keyblock ${KEY_DIR}/recovery_kernel.keyblock \
--signprivate ${KEY_DIR}/recovery_kernel_data_key.vbprivk \
--version "${KERNEL_VERSION}" \
--oldblob /dev/loop0p2 \
--config ${new_kerna_config}
```

Re-signing the stick if we modify ROOT-B

From vapier:

"if you modify ROOT-B, you'd have to update a couple of things if you wanted to enforce verified boot. if you don't want verified boot, you shouldn't need to touch the kernel.

Assuming you want verified boot, you'd have to rebuild the hashes in the root partition that follow the rootfs, and you'd have to update the root hash in the kernel command line that's part of the KERN-B image.

if you look at sign_official_build.sh, the update_rootfs_hash does all of that. unfortunately we don't have a script/entry point atm that would just resign a rootfs/kern combo."

xOther resources

1. This document derives from https://docs.google.com/document/d/1VBLVWlFTyt0oWJu9sCZ1_Sd70FqcqBc3nkJz2VXPg28/edit?usp=sharing
2. "Be your own vendor:Build your own ChromeOS distro and image server"