Packages - Bug #2372

[linux-libre-chromebook] not booting on C201

2019-07-12 06:21 PM - bill-auger

Status:	fixed	**% Done:**	100%
Priority:	critical		
Assignee:	Megver83		
Category:			

Description

heard it through the grapevine - the original feature request ticket is marked 'fixed'; and it was working at that time - so this should have a new ticket to try getting it working again

i leave this unconfirmed for now - because i can not confirm it; but ive added danielp3344, CBotulinum, and oaken-source as watchers because they are their only ones i know who have one of these computers

History

#1 - 2019-07-14 12:59 AM - Megver83

- Subject changed from {linux-libre-chromebook]: not booting on C201 to [linux-libre-chromebook] not booting on C201

OK, I'll be pending

#2 - 2019-08-16 12:26 AM - mai

Hi, I started messing with this since i got tired of waiting for somebody else to do it. the flashable kernel (vmlinux.kpart) that comes from the current linux-libre-chromebook package provided by parabola doesnt boot. when i try to boot it, after the libreboot screen, the lcd turns off and the pc hangs or something, and i am forced to pull the power cord. also full disclaimer: i have my battery disconnected. i have tried rebuilding the package without any alterations and the same thing happens. on the next try i made it so i did not use the config.armv7h provided with the linux-libre package source files and instead edited the PKGBUILD in the prepare() section as follows:

```
168   cp ../config.$CARCH .config
169   make olddefconfig
```

changes to

```
168   rm .config
169   make defconfig
```

and then run makepkg like normal. native build takes around 2 hours (on a usb hdd).

then i install the build packages with

```
pacman -U --assume-installed linux-libre=5.2.5 linux-libre-*.pkg.tar.xz
```

also another hint i want to share is how to partition a sd card or usb drive to use with the c201.

this is to wipe the drive and create a new gpt partition table

```
# DISK=/dev/sdX
# sgdisk --zap-all $DISK
# sgdisk -o $DISK
```

this is to make the kernel partition with a size of 32MB

```
# sgdisk --new=1:8192:73727 --typecode=1:7f00 --change-name=1:Kernel $DISK
# sgdisk --attributes=1:=:015A000000000000 $DISK
```

this is to use the rest of the space as the rootfs partition

```
# TOTAL_SECTORS=$(cat /sys/block/$(echo $DISK | awk -F'/' '{print $NF}')/size)
# FINAL_SECTOR=$(echo "(($TOTAL_SECTORS - 1 - 32 - 1) / 2048) * 2048) - 1"| bc)
# sgdisk --new=2:75776:$FINAL_SECTOR --typecode=2:7f01 --change-name=2:Root $DISK
```

then just continue install like normal. i can give my process if anybody wants it, i intend on writing a wiki article for the c201 eventually

anyway, the kernel i built will boot SOMETIMES, maybe 50% of the time, when it doesnt, it does the black lcd screen that doesnt power on thing, just like the stock linux-libre-chromebook kernel.

when it does boot successfully, the screen should go black after libreboot passes things off, stay off for 3 - 5 seconds, then the screen powers on and booting continues. one thing i have noticed is when the lcd turns off during early boot, there is a high pitch noise coming from the screen and you can hear the pitch change a few times if its going to boot properly, otherwise i will just powercycle the machine and try again.

so now once im booted up, some modules might be missing (for example fuse) and to get them i go back to the directory where i originally ran makepkg and edit ./src/linux-5.2/.config to enable a module (find CONFIG_FUSE_FS and set it to CONFIG_FUSE_FS=m)

then run

```
$ makepkg --noextract
```

then

```
# pacman -U linux-libre-5.2.5_gnu-1-armv7h.pkg.tar.xz
```

now where im stuck is installing parabola to the internal mmc. using the sgdisk commands mentioned above, or even gdisk or cgdisk, the partition files in /dev/ do not appear, unless i run partx -a /dev/mmcblk1

at this point i can format the mmcblk1p2 partition and use pacstrap to install to it, and dd my kernel to mmcblk1p1, but when i try to boot it, i just get that black (turned off) screen again...

ok so i know this was a lot but hopefully it helps somebody or we can use this info to move forward with a proper fix. using the sd card is a bit laggy so i would like to figure out how to get the internal mmc to work.

(sorry about edits, i am having issues with middle mouse button on thinkpad usb keyboard pasting when i try to scroll)

#3 - 2019-08-16 02:59 PM - danielp3344

For what it's worth my functional config from linux 5.1.8 does not work on mainline.

(Tip for kernel work on the C201, pressing the refresh and power buttons at the same time does a hard reset, which is nice when stuff hangs)

#4 - 2019-09-02 09:59 PM - bill-auger

megver -

the dragora folks just got 4.19.67-gnu booting on the C201 - maybe their config will work for parabola

https://notabug.org/dragora/dragora/src/master/archive/kernel

#5 - 2019-09-12 12:31 PM - bill-auger

as an update, they have all of the hardware working now, except for the radios

#6 - 2019-09-12 05:35 PM - danielp3344

- File .config added

I have a config (attached) that works for 5.1.x, but I cannot get anything later to boot.

#7 - 2019-09-12 05:56 PM - bill-auger

daniel -

have you tried the 4.19.6 kernel in [libre-testing]? - did it ever work? - does it still work? - maybe this bug report would not exist if the kernel for the C201 was LTS

#8 - 2019-09-12 06:03 PM - danielp3344

I seem to recall it working fine, but there are some ***huge*** improvements in 5.2 and 5.3, such as working gpu drivers. I really want to run GNOME :P

#9 - 2019-09-17 02:52 PM - bill-auger

ok so daniel has determined that the current situation is that the existing parabola config work for kernels up to v5.1; but is broken for kernels v5.2 and above

kelsoo has started collecting information about the C201 that would be helpful to anyone trying to use it with linux-libre

https://notabug.org/kelsoo/scribbles/wiki/Links+to+all+things+libre+for+the+c201+asus+chromebook

#10 - 2019-09-30 05:48 PM - bill-auger

i should mention, as an update, that daniel claimed to have made a booting v5.2 kernal - im not sure if it was with the vanilla kernel or linux-libre though - i was hoping he would update this ticket eventually with some detail on how to, or (+100) a working PKGBUILD

#11 - 2019-10-06 08:13 PM - bill-auger

- Assignee changed from Megver83 to oaken-source

- Status changed from unconfirmed to in progress

#12 - 2019-10-13 12:07 AM - bill-auger

there is a new 'linux-libre-chromebook' 5.3.1 in [libre] now

#13 - 2019-10-13 06:04 AM - oaken-source

it's in [libre-testing], not libre ;)

package is called linux-libre-veyron, after the linux-veyron from archarm. It's not terribly up to date, but in my experience it works pretty well.

#14 - 2019-10-13 06:41 AM - bill-auger

ok i am totally confused then - it looks like megver upgraded it https://www.parabola.nu/packages/?q=chromebook - that was on 2019-09-23, well after this issue was opened - has anyone tried that one?

#15 - 2019-10-13 06:42 AM - oaken-source

briefly. It didn't boot for me.

#16 - 2019-10-13 06:48 AM - bill-auger

i think we should pull it then, this is a confusing situation; if the broken kernel was simply replaced with another broken one - this problems reported in this BR were about kernels 5.2 and above - i think 5.3 is the next LTS, so i guess there is more to do

#17 - 2019-10-13 06:50 AM - oaken-source

linux-libre-chromebook is a split package from linux-libre. What it does, as far as I can tell, is use our generic arm kernel and repackage and sign it in a way the C201 can load it at boot time. However, it does seem like we need a custom config for the C201 to boot. We need to have a discussion whether we need to tie all these together into a single PKGBUILD or whether it's ok to have certain kernels such as the veyron in a separate PKGBUILD.

Keeping them together might make it easier to apply common changes across the board, while keeping them apart would allow us to be more specific in the details, and have different kernel versions that work best for certain devices for example. I'd tend towards the latter.

#18 - 2019-10-15 06:29 AM - CBotulinum

- File log.txt added

So, I've tried the kernels in libre-testing (referring to linux-libre-veyron) both before and after "config-disable-otg.patch" was applied, the result is the same:
Any USB device plugged in before boot gets power and sometimes even works (E.g.: A USB mouse works flawlessly, networking hardware shows up in lsusb but not in ip link), I've tried switching ports, same result, all testing done while also plugged into external power.
This is the log: http://ix.io/1YHB
I've also attached it here.
Note that plugging a device in (after boot) might supply it power, but doesn't generate any event in dmesg (checked also with dmesg -w).

#19 - 2019-12-08 08:07 PM - Megver83

sorry for arriving late to the party. I've been reading and studying a bit more this situation. I do not own a C201, but thanks to all the feedback from this issue I've been thinking on possible solutions, and there are some things I'll explain here to avoid more confusions.

Everything I've done has been without any testing by my side (only by users and some devs). So, as I read this email from a user that claimed to use this guide to install Parabola (by migrating ALARM), I read there that a rootfs with linux-veyron is used, but the second alternative is using the mainline kernel (linux-armv7) with linux-armv7-chromebook, so I replicated that in linux-libre.

However, ppl barely booted their devices, so I developed a theory: the C201 can work with vanilla kernel, but not as well as CrOS kernel, but with Linux-libre does worst.

oaken-source created linux-libre-veyron, but as bill-auger said, it ended up replacing a non-working kernel with another one. And it's true because there's almost no difference with what I do with linux-libre.

Then I saw linux-veyron PKGBUILD, and it builds the CrOS kernel (while linux-libre-veyron builds the normal linux-libre kernel). So I thought on a crazy idea: what about deblobbing that kernel, and test it in the C201? I'm currently working on that, but first, I need to know some things that are not very clear to me:

1) does ALARM with linux-armv7-chromebook works as expected?
2) is it does, how different is it than using linux-veyron? or linux-veyron is the only compatible kernel?
3) is linux-libre-chromebook a complete piece of crap or it helps a bit? what problems does it have? (besides #2161)

maybe I could start a mail thread in the mailing list to talk about this with the guys who own a C201, to see what can we get

P.S.: I read a Gentoo Wiki article about configuring the kernel in the C201 and they do not use the CrOS kernel source. Just for the record.

#20 - 2019-12-09 02:19 AM - bill-auger

the 'linux-libre-veyron' kernel in libre-testing may not be a total loss - IIRC only one build was attempted, and oaken-source stopped progress in that direction, not because he thought it was a dead end, but just as a matter of priorities

you can ask oaken-source to try the kernels that you make on his C201 - also probably danielp3344, CBotulinum, and others are still interested in experimenting

when i opened this ticket, i added every parabola user who i know has a C201, to the list of watchers - maybe there are more people interested, who are reading the mailing list, but it is probably a very small club of those who are willing to help - so far, those people are all watching this ticket already

#21 - 2019-12-09 03:50 AM - mai

Hi, I havnt tried rebuilding the kernel since my last post back in
august, but my suggestion would be to add an extra config file to the
linux-libre package source files, maybe named config.chromebook. same
should go for linux-libre-lts. in the next few weeks if time allows I
will try rebuilding it against the latest version and also try to get
proper module / initramfs working and also installing to the inernal
storage. if a package is created it would be simple enough for me to
just try it. also it should be noted that the way i bootstrapped my
installation was to take a working os on the c201 and chrooting into the
armv7 parabola release tarball (i used prawnos).

#22 - 2019-12-09 04:41 PM - Megver83

OK, so I created an initial CrOS Linux-libre kernel (5.4 currently)
https://gitlab.com/libreforks/linux-libre-cros

This is what Paul Kocialkowski suggested once, but was never done afaik
https://lists.parabola.nu/pipermail/dev/2016-April/003940.html

> Some work would be needed to write up instructions regarding how to install
> Parabola on the C201. (...) The device needs a kernel with specific drives for its hardware,
> that are not integrated into Linus Torvald's version of Linux, the upstream
> Linux kernel.
>
> Instead, CrOS systems use a downstream version of Linux, with changes for
> hardware support. The source code for it is available as part of the
> ChromiumOS project and it can be rebuilt from source. (...)
>
> One might be concerned about what comes with it, since it is a modified
> version of the upstream kernel. I am not aware of any addition that would conflict
> with users' privacy and security, (...) the
> **firmwares for Wi-Fi are not part of the kernel, but are out of the tree, as**
> **far as I recall. Thus, this kernel could be deblobbed and packaged in Parabola,**
> **to make the installation easier.**

I'm not 100% sure if this kernel is fully-free, but looks like it is, I read the Licensing from their Kernel Design and it says the following:

> All Linux kernel code, from Google and from partners, must be released under GPLv2 and must be released under the DCO (documented certificate of ownership) as a signoff of ownership. Each commit will contain the signoff of the code author/committer and the ACK of the patch approver. Code should be submitted under the Chromium contributor license agreement.

So what I understand from here is that Google and Chromium stuff in the kernel must be free software, so deblobbing their kernel like the vanilla kernel makes it fully-free.

Now, they sometimes backport stuff, I have to be pending if they backport anything that's potentially nonfree, but the advantage with using the latest stable releases is that backports from -rc releases are less likely to happen.

So, AFAIK, they won't add blobs because that should be done by the specific vendor who needs it, and if the Chromium project does it will be in temporary (and separated) branches till they are merged into mainline.

I will build this kernel in [libre-testing] and tell oaken-source to test it.

#23 - 2019-12-19 02:09 AM - mai

Hi, I got to test the linux-libre-cros package.

it works!

starting with the parabola-systemd-cli-armv7h-tarball-2018-02-06.tar.gz
rootfs archive you need to chroot into it and do a system upgrade, then
remove all entries from /etc/fstab.
also setting the root password helps ;)

this is booting via a 2GB usb stick or micro-sd

still having trouble with installing to the internal emmc,

if i can find the time, i will try to capture serial console logs from
the normal linux-libre-chromebook package from usb and also from trying to boot
the linux-libre-cros kernel when trying to boot via the internal storage

havnt tested anything else yet

edit: also the post install script https://git.parabola.nu/abslibre.git/tree/libre-testing/linux-libre-cros/linux-libre-cros.install
has a tiny bug where it doesnt print "Do you want to do this now? [y|N] " until after you press enter

#24 - 2019-12-29 01:04 AM - Megver83

mai wrote:

> Hi, I got to test the linux-libre-cros package.
>
> it works!

this is good news :D

> starting with the parabola-systemd-cli-armv7h-tarball-2018-02-06.tar.gz
> rootfs archive you need to chroot into it and do a system upgrade, then
> remove all entries from /etc/fstab.
> also setting the root password helps ;)
>
> this is booting via a 2GB usb stick or micro-sd

a rootfs for the C201 could be created, although I'm a pro-migration-from-ALARM rather than creating rootfs ourselves but meh, less work for the end user

> still having trouble with installing to the internal emmc,

If you can get it working into the internal eMMC, that would be awesome. That would deserve a wiki article too.

> if i can find the time, i will try to capture serial console logs from
> the normal linux-libre-chromebook package from usb and also from trying to boot
> the linux-libre-cros kernel when trying to boot via the internal storage
>
> havnt tested anything else yet

don't worry about linux-libre-chromebook anymore, if linux-libre-cros works it's fine. I'll probably rename it to linux-libre-veyron, but that's another thing.

> edit: also the post install script https://git.parabola.nu/abslibre.git/tree/libre-testing/linux-libre-cros/linux-libre-cros.install
> has a tiny bug where it doesnt print "Do you want to do this now? [y|N] " until after you press enter

I think I didn't understand this, do you have the fix?
btw, besides this "tiny bug", the script worked as expected? because the post-install script from linux-libre-chromebook had some issues in finding the correct kernel partition but I changed it in linux-libre-cros to see if it fixed it

#25 - 2020-01-13 09:49 PM - mai

- File linux-libre-cros-5.4.2-1-armv7h.bootlog added

- File linux-libre-cros.install added

- File linux-libre-chromebook-5.4.8-1-armv7h.bootlog added

hi, so i was able to solder some wires to the servo connector pads.

also a fun note, the cros kernel DOES work with internal emmc! i havnt built the latest cros kernel as it takes more than 8 hours to compile, im still using the linux-libre-cros-5.4.2-1 version

with the linux-libre-cros-5.4.2-1 kernel the screen doesnt come on unless you have the package (linux-libre-cros-5.4.2-1-armv7h.pkg.tar.xz) installed too, but i did get it boot completely to a login prompt via serial console. when i tested this before for the emmc, i just dd-ed the kernel to the first partition and i got the black (turned off) screen. i tried it again after doing a fresh parabola install and it worked.

i did try running the newest cros kernel (linux-libre-cros 5.4.6-1) and yes it works via usb and emmc. i did notice the kernel name was changed to veyron-dirty... i guess using the linux-libre deblob script isnt cleaning well enough?

so i have attached logs for the linux-libre-cros-5.4.2-1 and the linux-libre-chromebook-5.4.8-1

i had to rebuild these on my own to change the cmdline to be
console=ttyS2,115200n8 earlyprintk=ttyS2,115200n8 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 rootwait rw noinitrd loglevel=8 console_msg_format=syslog

also about the dd command when installing the cros kernel, for some reason the read command prints the "Do you want to do this now? [y|N] " string AFTER you press enter, not sure why as when i run that command in a terminal it acts as expected. i attached linux-libre-cros.install with the change that should be a work around.

if the cros kernel is truly dirty, it would be best to try and figure out how to make the chromebook kernel work. im not experienced enough to know how to interpret the logs as hints towards the next step. I am willing to mess around more if anybody has any suggestions.

#26 - 2020-01-14 12:09 AM - Megver83

Awesome logs, thanks for sharing them!

The "dirty" suffix appears when scripts/setlocalversion is ran, but don't worry, it's clean. It has nothing to do with the kernel being blob-free or not.

I've been talking with two ChromiumOS kernel devs (here is the conversation) and one of them told me that with 5.4 I won't get GPU, but with 4.19 yes, so I'm planning to update linux-libre-veyron (the one from oaken-source) to the latest LTS version from the CrOS kernel, and keep it in 4.19 till its lifecycle ends unless the next LTS works without problems.

I'll notify here when I do that, I've to be more careful with the deblobbing of that branch (chromeos-4.19) because they backport stuff from newer versions which might have blobs. Hopefully a 4.19 CrOS kernel should work better than latest stable/mainline

#27 - 2020-01-14 12:27 AM - Megver83

- % Done changed from 0 to 70

- Assignee changed from oaken-source to Megver83

#28 - 2020-01-15 11:50 PM - mai

hi again, I got the chromebook kernel to work!

long story short, you just need to add the 'gpt' kernel parameter to the cmdline

longer explanation:

after posting the logs in my previous post, i realized there was a kernel parameter that would provide even more information, its 'initcall_debug'

here is what my cmdline became:
console=ttyS2,115200n8 earlyprintk=ttyS2,115200n8 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 rootwait rw noinitrd loglevel=8

console_msg_format=syslog initcall_debug

what i noticed is the "Waiting for root device" message. this is from the 'rootwait' parameter. so i changed it to rootdelay=10

here is what my cmdline became:
console=ttyS2,115200n8 earlyprintk=ttyS2,115200n8 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 rootdelay=10 rw noinitrd loglevel=8 console_msg_format=syslog initcall_debug

after it waits for 10 seconds it says:
<6>[19.942403] Waiting 10 sec before mounting root device...
<4>[30.178588] VFS: Cannot open root device "PARTUUID=89b31cdb-1147-5241-8271-c1adbb9bbb44/PARTNROFF=1" or unknown-block(0,0): error -6
<4>[30.192210] Please append a correct "root=" boot option; here are the available partitions:
<4>[30.201850] 1f00 4096 mtdblock0
<4>[30.201851] (driver?)
<4>[30.209741] b300 15392768 mmcblk2
<4>[30.209743] driver: mmcblk
<0>[30.217916] Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0)

15392768 mmcblk2 is the emmc, but for some reason its not reading the partition table... i figured the way i partitioned the drive might have been the fault.
im using parabola installed to a sdcard as to install to the emmc, so i booted that to try repartition with cgpt instead of sfdisk.

when i ran sfdisk -d /dev/mmcblk0

it returns:

The primary GPT table is corrupt, but the backup appears OK, so that will be used.
label: gpt
label-id: EBA5A923-2F33-7C4E-AC9A-1555FD600D19
device: /dev/mmcblk0
unit: sectors
first-lba: 34
last-lba: 30785502

/dev/mmcblk0p1 : start= 20480, size= 65536, type=FE3A2A5D-4F32-41A7-B725-ACCC3285A309, uuid=89B31CDB-1147-5241-8271-C1ADBB9BBB44, name="Kernel", attrs="GUID:49,51,52,54,56"
/dev/mmcblk0p2 : start= 86016, size= 30699486, type=EBD0A0A2-B9E5-4433-87C0-68B6B72699C7, uuid=63DB8E49-63C4-984E-90A0-8AC3222C4771, name="Root"

so seeing the message about the primary GPT table being corrupt, i tried to use gdisk to recover it from the backup.
so when i run gdisk /dev/mmcblk0 then press p then enter, it shows no table... if i press r then enter (for recovery and transformation options) then b then enter (for use backup GPT header (rebuilding main)) then p then enter, it still didnt show the partition table.

i wondered how prawnos got the emmc to boot and so i looked at their install script
https://github.com/SolidHal/PrawnOS/blob/master/scripts/InstallScripts/InstallPrawnOS.sh
in the emmc_partition() fuction there is a comment saying:
#disable dmesg, writing the partition map tries to write the the first gpt table, which is unmodifiable

so it occurred to me that the linux-libre-chromebook kernel is only reading the primary partition table. so i looked to see if there is a way to inform the kernel to use the backup partition table

this is when i found the 'gpt' kernel parameter
i tried it and the system booted, backlight, lcd, and all, from the emmc.

so now if we go back to the parabola linux-libre-chromebook stock cmdline and just add gpt to it, it becomes
console=tty0 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 gpt rootwait rw noinitrd

it works!

I can provide logs for any stage of this if you want. i have no idea why / how the cros / alarm /other c201 distro kernels get past this "corrupt / unmodifiable primary partition table" issue without the gpt parameter, but im not sure how to figure that out.

ok so closing requests are to add the gpt parameter to the cmdline for the PKGBUILD for linux-libre-chromebook

and it would also be awesome to include the file kernel.itb in the package so you can change the cmdline parameters without having to recompile the package just to get this file.
the other files are optional (bootloader.bin, cmdline, kernel.keyblock, kernel_data_key.vbprivk), as you can get them from the linux-libre package source files,
but probably not a bad idea to include them as well.

wooo! parabola kernel with panfrost!

i will start writing up a wiki page for setting up the c201 once the pkgbuild is updated for linux-libre-chromebook and the updated package is in the repos.

thanks

#29 - 2020-01-16 01:08 AM - Megver83

mai wrote:

> hi again, I got the chromebook kernel to work!
>
> long story short, you just need to add the 'gpt' kernel parameter to the cmdline

awesome news! this is MUCH better than having to fork the CrOS kernel because it's just too time-consuming

> longer explanation:
>
> after posting the logs in my previous post, i realized there was a kernel parameter that would provide even more information, its 'initcall_debug'
>
> here is what my cmdline became:
> console=ttyS2,115200n8 earlyprintk=ttyS2,115200n8 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 rootwait rw noinitrd loglevel=8 console_msg_format=syslog initcall_debug
>
> what i noticed is the "Waiting for root device" message. this is from the 'rootwait' parameter. so i changed it to rootdelay=10
>
> here is what my cmdline became:
> console=ttyS2,115200n8 earlyprintk=ttyS2,115200n8 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 rootdelay=10 rw noinitrd loglevel=8 console_msg_format=syslog initcall_debug
>
> after it waits for 10 seconds it says:
> <6>[19.942403] Waiting 10 sec before mounting root device...
> <4>[30.178588] VFS: Cannot open root device "PARTUUID=89b31cdb-1147-5241-8271-c1adbb9bbb44/PARTNROFF=1" or unknown-block(0,0): error -6
> <4>[30.192210] Please append a correct "root=" boot option; here are the available partitions:
> <4>[30.201850] 1f00 4096 mtdblock0
> <4>[30.201851] (driver?)
> <4>[30.209741] b300 15392768 mmcblk2
> <4>[30.209743] driver: mmcblk
> <0>[30.217916] Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0)
>
> 15392768 mmcblk2 is the emmc, but for some reason its not reading the partition table... i figured the way i partitioned the drive might have been the fault.
> im using parabola installed to a sdcard as to install to the emmc, so i booted that to try repartition with cgpt instead of sfdisk.
>
> when i ran sfdisk -d /dev/mmcblk0
>
> it returns:
>
> The primary GPT table is corrupt, but the backup appears OK, so that will be used.
> label: gpt
> label-id: EBA5A923-2F33-7C4E-AC9A-1555FD600D19
> device: /dev/mmcblk0
> unit: sectors
> first-lba: 34
> last-lba: 30785502
>
> /dev/mmcblk0p1 : start= 20480, size= 65536, type=FE3A2A5D-4F32-41A7-B725-ACCC3285A309, uuid=89B31CDB-1147-5241-8271-C1ADBB9BBB44, name="Kernel", attrs="GUID:49,51,52,54,56"
> /dev/mmcblk0p2 : start= 86016, size= 30699486, type=EBD0A0A2-B9E5-4433-87C0-68B6B72699C7, uuid=63DB8E49-63C4-984E-90A0-8AC3222C4771, name="Root"
>
> so seeing the message about the primary GPT table being corrupt, i tried to use gdisk to recover it from the backup.
> so when i run gdisk /dev/mmcblk0 then press p then enter, it shows no table... if i press r then enter (for recovery and transformation options) then b then enter (for use backup GPT header (rebuilding main)) then p then enter, it still didnt show the partition table.
>
> i wondered how prawnos got the emmc to boot and so i looked at their install script
> https://github.com/SolidHal/PrawnOS/blob/master/scripts/InstallScripts/InstallPrawnOS.sh
> in the emmc_partition() fuction there is a comment saying:
> #disable dmesg, writing the partition map tries to write the the first gpt table, which is unmodifiable
>
> so it occurred to me that the linux-libre-chromebook kernel is only reading the primary partition table. so i looked to see if there is a way to inform the kernel to use the backup partition table
>
> this is when i found the 'gpt' kernel parameter
> i tried it and the system booted, backlight, lcd, and all, from the emmc.

just found it (posting here for documenting this)
https://www.kernel.org/doc/html/v5.4/admin-guide/kernel-parameters.html

```
gpt             [EFI] Forces disk with valid GPT signature but
                      invalid Protective MBR to be treated as GPT. If the
                      primary GPT is corrupted, it enables the backup/alternate
                      GPT to be used instead.
```

> so now if we go back to the parabola linux-libre-chromebook stock cmdline and just add gpt to it, it becomes
> console=tty0 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 gpt rootwait rw noinitrd
>
> it works!
>
> I can provide logs for any stage of this if you want. i have no idea why / how the cros / alarm /other c201 distro kernels get past this "corrupt / unmodifiable primary partition table" issue without the gpt parameter, but im not sure how to figure that out.

well, makes sense, but yea I also have the doubt of why does it work with other kernels, including linux-libre-cros, but not linux-libre-chromebook. And yes, please share the logs

btw, could you share the default /proc/cmdline from linux-libre-cros? I'll dig a bit to see why our linux-libre needs the gpt option

> ok so closing requests are to add the gpt parameter to the cmdline for the PKGBUILD for linux-libre-chromebook
>
> and it would also be awesome to include the file kernel.itb in the package so you can change the cmdline parameters without having to recompile the package just to get this file.
> the other files are optional (bootloader.bin, cmdline, kernel.keyblock, kernel_data_key.vbprivk), as you can get them from the linux-libre package source files,
> but probably not a bad idea to include them as well.

what kernel.itb file? and why do you say those files are optional? they are necessary to make the kernel bootable

> wooo! parabola kernel with panfrost!

regarding this, I'll update linux-libre with the gpt cmdline option. If it's confirmed to work perfectly, I'll probably keep it simple and leave it as the only kernel to work with the C201

> i will start writing up a wiki page for setting up the c201 once the pkgbuild is updated for linux-libre-chromebook and the updated package is in the repos.
>
> thanks

thank to you, your time and well-done testing!

#30 - 2020-01-16 02:49 AM - Megver83

I'd like to know more about the difference between both linux-libre and linux-libre-cros in terms of performance and compatibility in the C201, are they notably different?

#31 - 2020-01-16 05:39 PM - Megver83

- File 0001-CHROMIUM-block-partitions-efi-Add-support-for-IGNORE.patch added

mai wrote:

> (...) i have no idea why / how the cros / alarm /other c201 distro kernels get past this "corrupt / unmodifiable primary partition table" issue without the gpt parameter, but im not sure how to figure that out.

I haven't check their kernels, but did a quick check on a diff I made between the CrOS kernel and vanilla, and looked for "gpt", then I found a file they modify, and got this:
https://chromium.googlesource.com/chromiumos/third_party/kernel/+/2b4a8bfd9379fd9431d517e4db20c57468ecf3e6%5E%21/

> This patch adds support for a special GPT header signature marker (using
> the string 'IGNOREME' instead of the spec's 'EFI PART'). This tells the
> kernel to ignore this GPT completely and look at the other one instead.
> **Since the kernel always prefers the primary GPT anyway, all we really need to do effectively is to check whether the primary GPT is marked 'IGNOREME' and force evaluation of the secondary one in that case.**

I'll build linux-libre with this patch, if it doesn't work, then I'll add 'gpt' to the cmdline (without this patch)

#32 - 2020-01-16 07:13 PM - Megver83

So I built the LTS CrOS kernel (v4.19.94), which has panfrost according to the CrOS kernel devs. I's an update to libre-testing/linux-libre-veyron, please try it and tell me if it's worthy or the same thing as linux-libre-cros and linux-libre

#33 - 2020-01-17 06:03 AM - mai

- File veyronlog added

Megver83 wrote:

> So I built the LTS CrOS kernel (v4.19.94), which has panfrost according to the CrOS kernel devs. I's an update to libre-testing/linux-libre-veyron, please try it and tell me if it's worthy or the same thing as linux-libre-cros and linux-libre

linux-libre-veyron 4.19.94-1 hangs at boot somewhere. it gives a blank black screen with the backlight on.

i did mess with futility a bit and discovered there is some functionality that is kinda helpful

futility show vmlinux.kpart
will show the cmdline thats cooked inside.

also you can extract the vmlinuz from vmlinux.kpart:
futility --debug vbutil_kernel --get-vmlinuz vmlinux.kpart --vmlinuz-out vmlinuz

i did this with my last linux-libre-chromebook vmlinux.kpart
the cmdline for that was with the gpt fix:
console=tty0 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 gpt rootwait rw noinitrd

an interesting thing to note is that the "vmlinuz" extracted is not the same size as the uimage created by mkimage in the PKGBUILD (kernel.itb) that was the vmlinuz input file when it was originally wrapped when built with makepkg. whats the difference? idk...

anyway, i then re-wrapped the vmlinuz, changing the rootwait to rootdelay=10 and setting up serial output just to see if everything is working as it should.

i am using parabola installed on a 8GB sdcard with the linux-libre-cros kernel as my recovery os
i have a directory where i was testing the different cmdline parameters

```
# cd /path/to/last_linux-libre-chromebook_rewrap
# ls
bootloader.bin  cmdline  kernel.itb  kernel.keyblock  kernel_data_key.vbprivk  vmlinux.kpart  wrap.sh
# cat wrap.sh
futility --debug vbutil_kernel --pack vmlinux.kpart --version 1 --vmlinuz kernel.itb --arch arm --keyblock ker
nel.keyblock --signprivate kernel_data_key.vbprivk --config cmdline --bootloader bootloader.bin
# cat cmdline
console=tty0 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 gpt rootwait rw noinitrd
# futility show vmlinux.kpart
Kernel partition:        ../vmlinux.kpart
Key block:
  Signature:             ignored
  Size:                  0x4b8
  Flags:                 7  !DEV DEV !REC
  Data key algorithm:    4 RSA2048 SHA256
  Data key version:      1
  Data key sha1sum:      d6170aa480136f1f29cf339a5ab1b960585fa444
Kernel Preamble:
  Size:                  0xfb48
  Header version:        2.2
  Kernel version:        1
  Body load address:     0x100000
  Body size:             0x77c000
  Bootloader address:    0x87b000
  Bootloader size:       0x1000
  Flags:                 0x0
Body verification succeeded.
Config:
console=tty0 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 gpt rootwait rw noinitrd
# mkdir re-re-wrap
# cd re-re-wrap
# futility --debug vbutil_kernel --get-vmlinuz ../vmlinux.kpart --vmlinuz-out vmlinuz
# wc -c vmlinuz
7847936 vmlinuz
# wc -c ../kernel.itb
7832676 ../kernel.itb
# echo "console=ttyS2,115200n8 earlyprintk=ttyS2,115200n8 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 gpt roo
```

```
tdelay=10 rw noinitrd" > cmdline
# futility --debug vbutil_kernel --pack vmlinux.kpart --version 1 --vmlinuz vmlinuz --arch arm --keyblock ../k
ernel.keyblock --signprivate ../kernel_data_key.vbprivk --config cmdline --bootloader ../bootloader.bin
DEBUG: run_command("vbutil_kernel") ...
DEBUG:   argv[0] = "vbutil_kernel"
DEBUG:   argv[1] = "--pack"
DEBUG:   argv[2] = "vmlinux.kpart"
DEBUG:   argv[3] = "--version"
DEBUG:   argv[4] = "1"
DEBUG:   argv[5] = "--vmlinuz"
DEBUG:   argv[6] = "vmlinuz"
DEBUG:   argv[7] = "--arch"
DEBUG:   argv[8] = "arm"
DEBUG:   argv[9] = "--keyblock"
DEBUG:   argv[10] = "../kernel.keyblock"
DEBUG:   argv[11] = "--signprivate"
DEBUG:   argv[12] = "../kernel_data_key.vbprivk"
DEBUG:   argv[13] = "--config"
DEBUG:   argv[14] = "cmdline"
DEBUG:   argv[15] = "--bootloader"
DEBUG:   argv[16] = "../bootloader.bin"
DEBUG: Reading cmdline
DEBUG:  config file size=0x7c
DEBUG: Reading ../bootloader.bin
DEBUG:  bootloader file size=0x200
DEBUG: Reading vmlinuz
DEBUG:  vmlinuz file size=0x77c000
DEBUG: g_kernel_blob_size  0x77c00000001000
DEBUG: g_kernel_size       0xb640300800000000 ofs 0x4ec9e700000000
DEBUG: g_config_size       0xb64000080077c000 ofs 0x4ec9e700000000
DEBUG: g_param_size        0xb64010080077d000 ofs 0x4ec9e700000000
DEBUG: g_bootloader_size   0xb64020080077e000 ofs 0x4ec9e700000000
DEBUG: g_ondisk_bootloader_addr   0x87e000
DEBUG: end of kern_blob at kern_blob+0x10004317f900
DEBUG:  kernel32_start=0x4317f9004317f900
DEBUG:  kernel32_size=0x4317f9004317f900
DEBUG: kblob_size = 0x77f000
DEBUG: vblock_size = 0x10000
DEBUG: writing vmlinux.kpart with 0x77f00000010000, 0x2260ba8b5c84008
# futility show vmlinux.kpart
Kernel partition:        vmlinux.kpart
Key block:
  Signature:             ignored
  Size:                  0x4b8
  Flags:                 7  !DEV DEV !REC
  Data key algorithm:    4 RSA2048 SHA256
  Data key version:      1
  Data key sha1sum:      d6170aa480136f1f29cf339a5ab1b960585fa444
Kernel Preamble:
  Size:                  0xfb48
  Header version:        2.2
  Kernel version:        1
  Body load address:     0x100000
  Body size:             0x77f000
  Bootloader address:    0x87e000
  Bootloader size:       0x1000
  Flags:                 0x0
Body verification succeeded.
Config:
console=ttyS2,115200n8 earlyprintk=ttyS2,115200n8 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 gpt rootdelay=1
0 rw noinitrd
# lsblk
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mtdblock0     31:0    0    4M  0 disk
mmcblk1      179:0    0  7.4G  0 disk
|-mmcblk1p1  179:1    0   32M  0 part
`-mmcblk1p2  179:2    0  7.4G  0 part /
mmcblk0      179:256  0 14.7G  0 disk
|-mmcblk0p1  179:257  0   32M  0 part
`-mmcblk0p2  179:258  0 14.7G  0 part
mmcblk0boot0 179:512  0    4M  1 disk
mmcblk0boot1 179:768  0    4M  1 disk
# dd if=vmlinux.kpart of=/dev/mmcblk0p1
15480+0 records in
15480+0 records out
```

```
7925760 bytes (7.9 MB, 7.6 MiB) copied, 0.661896 s, 12.0 MB/s
# poweroff
```

i take out the sdcard and power on the c201
i now have serial console during boot and it waited 10 seconds to mount the root partition.
rewrapping this way worked and i shouldnt have to recompile kernels anymore...

i wrote all this out essentially to explain this next part. i log in and do the following

```
# pacman -Syy
# pacman -S linux-libre-veyron
# pacman -Ql linux-libre-veyron | grep vmlinux.kpart
linux-libre-veyron /boot/vmlinux.kpart
# cd /boot
# futility show vmlinux.kpart
Kernel partition:        vmlinux.kpart
Key block:
  Signature:             ignored
  Size:                  0x4b8
  Flags:                 7  !DEV DEV !REC
  Data key algorithm:    4 RSA2048 SHA256
  Data key version:      1
  Data key sha1sum:      d6170aa480136f1f29cf339a5ab1b960585fa444
Kernel Preamble:
  Size:                  0xfb48
  Header version:        2.2
  Kernel version:        1
  Body load address:     0x100000
  Body size:             0x6cd000
  Bootloader address:    0x7cc000
  Bootloader size:       0x1000
  Flags:                 0x0
Body verification succeeded.
Config:
console=tty0 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 rootwait rw noinitrd
# mkdir veyron-rewrap
# cd veyron-rewrap
### i copy over bootloader.bin kernel.keyblock kernel_data_key.vbprivk to this directory
# futility --debug vbutil_kernel --get-vmlinuz ../vmlinux.kpart --vmlinuz-out vmlinuz
DEBUG: run_command("vbutil_kernel") ...
DEBUG:   argv[0] = "vbutil_kernel"
DEBUG:   argv[1] = "--get-vmlinuz"
DEBUG:   argv[2] = "../vmlinux.kpart"
DEBUG:   argv[3] = "--vmlinuz-out"
DEBUG:   argv[4] = "vmlinuz"
DEBUG: ../vmlinux.kpart size is 0x6dd000
DEBUG: Reading ../vmlinux.kpart
DEBUG: Keyblock is 0x4b8 bytes
DEBUG: Preamble is 0xfb48 bytes
DEBUG:  kernel_version = 1
DEBUG:  bootloader_address = 0x7cc000
DEBUG:  bootloader_size = 0x1000
DEBUG:  kern_blob_size = 0x6cd000
DEBUG:  flags = 0x0
DEBUG: kernel blob is at offset 0x10000
DEBUG: bootloader_size     = 0x1000
DEBUG: bootloader_ofs      = 0x6cc000
DEBUG: param_ofs           = 0x6cb000
DEBUG: config_ofs          = 0x6ca000
DEBUG: kernel_size         = 0x6ca000
# echo "console=ttyS2,115200n8 earlyprintk=ttyS2,115200n8 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 gpt roo
tdelay=10 rw noinitrd" > cmdline
# futility --debug vbutil_kernel --pack vmlinux.kpart --version 1 --vmlinuz vmlinuz --arch arm --keyblock kern
el.keyblock --signprivate kernel_data_key.vbprivk --config cmdline --bootloader bootloader.bin
DEBUG: run_command("vbutil_kernel") ...
DEBUG:   argv[0] = "vbutil_kernel"
DEBUG:   argv[1] = "--pack"
DEBUG:   argv[2] = "vmlinux.kpart"
DEBUG:   argv[3] = "--version"
DEBUG:   argv[4] = "1"
DEBUG:   argv[5] = "--vmlinuz"
DEBUG:   argv[6] = "vmlinuz"
DEBUG:   argv[7] = "--arch"
DEBUG:   argv[8] = "arm"
DEBUG:   argv[9] = "--keyblock"
```

```
DEBUG:   argv[10] = "kernel.keyblock"
DEBUG:   argv[11] = "--signprivate"
DEBUG:   argv[12] = "kernel_data_key.vbprivk"
DEBUG:   argv[13] = "--config"
DEBUG:   argv[14] = "cmdline"
DEBUG:   argv[15] = "--bootloader"
DEBUG:   argv[16] = "bootloader.bin"
DEBUG: Reading cmdline
DEBUG:  config file size=0x7c
DEBUG: Reading bootloader.bin
DEBUG:  bootloader file size=0x200
DEBUG: Reading vmlinuz
DEBUG:  vmlinuz file size=0x6cd000
DEBUG: g_kernel_blob_size  0x6cd00000001000
DEBUG: g_kernel_size       0xb652400800000000 ofs 0x4ba9e700000000
DEBUG: g_config_size       0xb6521008006cd000 ofs 0x4ba9e700000000
DEBUG: g_param_size        0xb6522008006ce000 ofs 0x4ba9e700000000
DEBUG: g_bootloader_size   0xb6523008006cf000 ofs 0x4ba9e700000000
DEBUG: g_ondisk_bootloader_addr   0x7cf000
DEBUG: end of kern_blob at kern_blob+0x1000f10d4b00
DEBUG:  kernel32_start=0xf10d4b00f10d4b00
DEBUG:  kernel32_size=0xf10d4b00f10d4b00
DEBUG: kblob_size = 0x6d0000
DEBUG: vblock_size = 0x10000
DEBUG: writing vmlinux.kpart with 0x6d000000010000, 0x1c04ba8b5e54008
# futility show vmlinux.kpart
Kernel partition:        vmlinux.kpart
Key block:
  Signature:             ignored
  Size:                  0x4b8
  Flags:                 7  !DEV DEV !REC
  Data key algorithm:    4 RSA2048 SHA256
  Data key version:      1
  Data key sha1sum:      d6170aa480136f1f29cf339a5ab1b960585fa444
Kernel Preamble:
  Size:                  0xfb48
  Header version:        2.2
  Kernel version:        1
  Body load address:     0x100000
  Body size:             0x6d0000
  Bootloader address:    0x7cf000
  Bootloader size:       0x1000
  Flags:                 0x0
Body verification succeeded.
Config:
console=ttyS2,115200n8 earlyprintk=ttyS2,115200n8 init=/sbin/init root=PARTUUID=%U/PARTNROFF=1 gpt rootdelay=1
0 rw noinitrd
# lsblk
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mtdblock0     31:0    0    4M  0 disk
mmcblk2      179:0    0 14.7G  0 disk
├─mmcblk2p1  179:1    0   32M  0 part
└─mmcblk2p2  179:2    0 14.7G  0 part /
mmcblk2boot0 179:256  0    4M  1 disk
mmcblk2boot1 179:512  0    4M  1 disk
# dd if=vmlinux.kpart of=/dev/mmcblk2p1
14080+0 records in
14080+0 records out
7208960 bytes (7.2 MB, 6.9 MiB) copied, 0.508708 s, 14.2 MB/s
# reboot
```

and when it reboots, libreboot / depthcharge finds the kernel partition and reads the dtbs, but the kernel doesnt print anything...

ive attached serial console log from powering on the c201 until the point that it hangs.

its way past my bedtime and all of a sudden paying work is picking up again. i can do more testing / debugging this weekend if time allows.

#34 - 2020-01-17 05:24 PM - Megver83

If it libre-testing/linux-libre-veyron doesn't work, let's leave it like that. There's already the workaround for linux-libre-chromebook and linux-libre-cros works as expected.

This is what I plan to do now:

1) build linux-libre with the patch I attached, to see if we can live without 'gpt' in cmdline, this will be in linux-libre v5.4.12. If it fails again, I'll remove

that patch and add 'gpt' in cmdline
2) update (to v5.4.12 or higher) and move linux-libre-cros to [libre], and rename it to 'linux-libre-veyron' to make it a replacement for Arch ARM's linux-veyron
3) remove libre-testing/linux-libre-veyron since it doesn't work OOTB and linux-libre-cros will use that name in [libre]

So, the only thing I need you to test is linux-libre-chromebook 5.4.12 and libre-testing/linux-libre-cros 5.4.11 (this update is important since I removed a blob I didn't saw before, here are its modified deblobbing scripts). Do not change anything from linux-libre-chromebook if it fails. Just want to know if it boots without changing it. If it has the same problem, you can try to use the same fix (add 'gpt' to cmdline) just to confirm.

This is very good in terms of maintenance. The LTS CrOS kernel required more deblobbing mainly because of backports from the vanilla kernel, but stable Linux versions don't have anything to be backported (only very few things from mainline releases).

#35 - 2020-01-17 05:24 PM - Megver83

- % Done changed from 70 to 80

#36 - 2020-01-17 05:43 PM - bill-auger

On Fri, 17 Jan 2020 17:24:11 +0000 labs@parabola.nu wrote:

> The LTS CrOS kernel
> required more deblobbing
> but stable Linux versions don't

and i presume that lnux-libre would be more interested in
whatever changes you need to make

#37 - 2020-01-18 06:09 PM - mai

I was eager to try the change to linux-libre-chromebook, so i started a fresh native build last night with no changes, just ran makepkg. it took approximately 6 hours to build. when i tested it today, it worked. i didnt change anything. this is from the installation to the emmc. i will test out micro-sd and usb, etc. and start collecting notes for a wiki article

#38 - 2020-01-19 04:01 AM - Megver83

That's good to know. I'll leave it like that then, anyways, just to make sure, try linux-libre-chromebook 5.4.12 from [libre]

For the last, I'd like to know if you noticed any important difference between linux-libre-chromebook and linux-libre-cros. I'm in the doubt if it's really necessary to maintain linux-libre-cros since linux-libre-chromebook is working fine.

#39 - 2020-01-19 04:02 AM - Megver83

- % Done changed from 80 to 90

#40 - 2020-02-03 07:06 PM - Megver83

any news, mai? I'll update linux-libre to 5.5, so please tell me if it works so we can mark this issue as fixed

#41 - 2020-02-03 07:19 PM - Megver83

fyi, I removed linux-libre-veyron and linux-libre-cros from [libre-testing]. If they are needed again, I'll upload them to [libre], but it doesn't look to be the case

#42 - 2020-02-04 03:39 AM - mai

in regards to "[linux-libre-chromebook] not booting on C201", this problem is solved, it boots. the c201 has other problems that are unrelated to it booting though. these are things like the popular ath9k usb wifi adapter only works sometimes and when it fails, the usb port it was connected to doesnt work until the computer is rebooted. as an alternative i have tested a few different ways to get network access: 'smc ez connect usb 2202usb/eth' it uses the pegasus module, and it works great. i also tried a generic usbhub+ethernet adapter that uses the r8152 module. this seemed ok, but im not 100% sure if the usb speeds take a hit... i also tried a netgear wg111v2 usb wifi adapter that uses the carl9170 module and that seemed fine without extensive testing. also it would be easier to stress test my normal routine if icecat / iceweasel got updated... but thats not a boot issue.

as far as other stress testing i tried:

running gnome with wayland: it worked, but was a bit laggy and froze if you werent ultra nice to it.

xorg was more forgiving, but i didnt test gnome as i couldnt get it to launch.

then i went with jwm and that seemed to work ok. i tried out minetest as it seemed to be the most process heavy game in the arm repos. it worked fine, i played it for an hour.

one final note about the c201: (as far as ive figured) in order to install parabola to it without using any proprietary software, you have to use the armrootfs, which wont boot with the newest kernel unless you chroot into it and do pacman -Syu (if you can manage to get past the keyring trouble).

so basically this means anybody wanting to install parabola on the c201 has to jump through a lot of hoops. i can write a wiki article about updating the arm rootfs image, but maybe its time to make an updated one...

also in order to add a wiki page it seems i need a wiki account, and i cant seem create an account on my own.

#43 - 2020-02-04 04:01 AM - Megver83

- % Done changed from 90 to 100

- Status changed from in progress to fixed

Right, I'll mark this as fixed

Regarding the wiki page, I'll request the creation for your account. Anyways, you can write a wiki page in your Wikipedia profile page, so I can then copy-paste, but that's the last option though.

#44 - 2020-02-04 04:59 PM - bill-auger

Issue #2372 has been updated by mai.
i can write a

> wiki article about updating the arm rootfs image, but maybe
> its time to make an updated one...
>
> also in order to add a wiki page it seems i need a wiki
> account, and i cant seem create an account one on my own.

registrations were closed last summer due to spam - for now, you
just need to ask someone to create an account for you

we definitely would not want an article such as you described
though - that would be very unprofessional - if the tarball
is not working properly, then it needs to be replaced - i did
the work a few weeks ago to get it up to date - i could put
out a new one any time; but i can only test it with QEMU - would
you be willing to test it out?

#45 - 2020-02-04 07:55 PM - mai

yes i will test it. for the c201 there will still need to be instructions on how to make a bootable usb drive / micro sd with the updated rootfs. this is basically just partition the drive, manually download the linux-libre-chromebook package and signature, manually verify signature, extract the package to a temporary directory, and dd vmlinux.kpart to the kernel partition. this may break though as the arm rootfs ages, or rather, linux-libre-chromebook gets updated. you can work around this by having linux-libre-chromebook installed to this rootfs, so the matching vmlinux.kpart is available to flash later in time. then they can boot up and do a system upgrade. if you do that, anybody can create a booting parabola-c201 drive that is distro and architecture agnostic and will not require messing with qemu if they do not have access to another already set up armv7h system. the downside to this is that non-c201 parabola-arm users will now have the linux-libre-chromebook package installed to their system. alternatively if you do not want to include linux-libre-chromebook in the rootfs, qemu and a udev rule can be used with ' -hda /dev/sdX ' to do a system upgrade and get linux-libre-chromebook. there is also proot, but i couldnt get that to work.

anyway, yes, i am willing to test and document problems and try to find solutions to those problems.

on a side note, hopefully all this work will also allow the Samsung Chromebook Plus XE513C24 also run parabola :)

#46 - 2020-02-04 08:08 PM - bill-auger

we probably dont want any new wiki pages for this - we have
been working to consolidate the install guides; which were
spread across multiple articles - ideally, the ARM install guide
should include everything relevant for installing onto any ARM
machine, without referring to other articles

https://wiki.parabola.nu/ARM_Installation_Guide

that is the one most in need of attention though - if any important
information is missing, please do make a new section for it

#47 - 2020-02-04 09:35 PM - mai

ok, i just re-read the arm installation guide page and noticed the sections "10 Enable executing arm code on x86" and "11.2 With pacstrap"

with these two approaches one should be able to make a booting c201 parabola drive from an existing parabola (non-arm) machine. the only thing missing is partitioning instructions, which probably anybody wanting to install parabola to the c201 will already have looked at the archlinuxarm wiki page for the c100p. this however requires cgpt which is provided by the vboot-utils package, which is only in the arm repository. sgdisk can be used instead as i outlined in my first post, but its a bit confusing to look at compared to the cgpt commands that do basically the same thing. i am thinking

that the only real important part is that its gpt, the first partition is the kernel partition, and the typecode and attributes are set for that partition. ill experiment to see if i can make partitioning instructions wiki friendly. depending on pacstrap would ultimately mean that anybody wanting to try parabola on the c201 would require an existing machine running parabola.

these machines can be purchased for less than $50 used on ebay, and i think would be cool for people to be able to jump right into using parabola on it with the least amount of prerequisites. this to me seems like partitioning a usb drive / microsd, download/verify the rootfs, extract the rootfs onto the usbdrive, and flash the vmlinux.kpart included in the rootfs. at that point they can boot it and use pacstrap to install to the emmc.

#48 - 2021-01-04 04:53 AM - bill-auger

if anyone receiving this messgae, who owns a C201, and is stil using the parabola 'linux-libre-chromebook' kernel, or would like to use it, please let us know - we would like to support the C201, but currently none of the parabola devs own one; so we need someone to volunteer to use it regularly ,and report any problems

#49 - 2022-05-18 02:33 AM - Megver83

Just for the record, ChromiumOS deprecated the veyron patch justifying that it is only for veyron devices and that most of them reached their EOL.

I might implement the 'gpt' kernel parameter workaround in cmdline in the future.

Files

.config	124 KB	2019-09-12	danielp3344
log.txt	34.9 KB	2019-10-15	CBotulinum
linux-libre-cros.install	790 Bytes	2020-01-13	mai
linux-libre-chromebook-5.4.8-1-armv7h.bootlog	38.3 KB	2020-01-13	mai
linux-libre-cros-5.4.2-1-armv7h.bootlog	53.7 KB	2020-01-13	mai
0001-CHROMIUM-block-partitions-efi-Add-support-for-IGNORE.patch	5.14 KB	2020-01-16	Megver83
veyronlog	16.6 KB	2020-01-17	mai