به نام خدا

امیررضا شعیری [95109467] - دانیال عرفانیان [97110155]

پروژه نهایی (راه اندازی build server برای AOSP)

سیستم های عامل

جناب آقای دکتر جلیلی - جناب آقای تشکری - جناب آقای باجلان

دانشکده مهندسی کامپیوتر - دانشگاه صنعتی شریف

تابستان ۱۴۰۰

1. مقدمه

در این پروژه قصد داریم برای AOSP، یک build server راه اندازی کنیم، و درستی build server راه اندازی شده را تست کنیم. در بخش اول این گزارش به بررسی مفهوم AOSP و build server می پردازیم. این بخش به ما کمک می کند تا با کاری که باید در این پروژه انجام شود به خوبی آشنا شویم، و همین طور اطلاعات جالب دیگری را بدست آوریم. سپس در بخش دوم به راه اندازی build server می پردازیم. این قسمت در واقع بخش اصلی پروژه است. پس از آن در بخش سوم به تست build server راه اندازی شده می پردازیم. در این بخش از صحت build انجام شده مطمئن خواهیم شد. و در نهایت در بخش چهارم پروژه را جمع بندی می کنیم.

2. مفاهیم اولیه

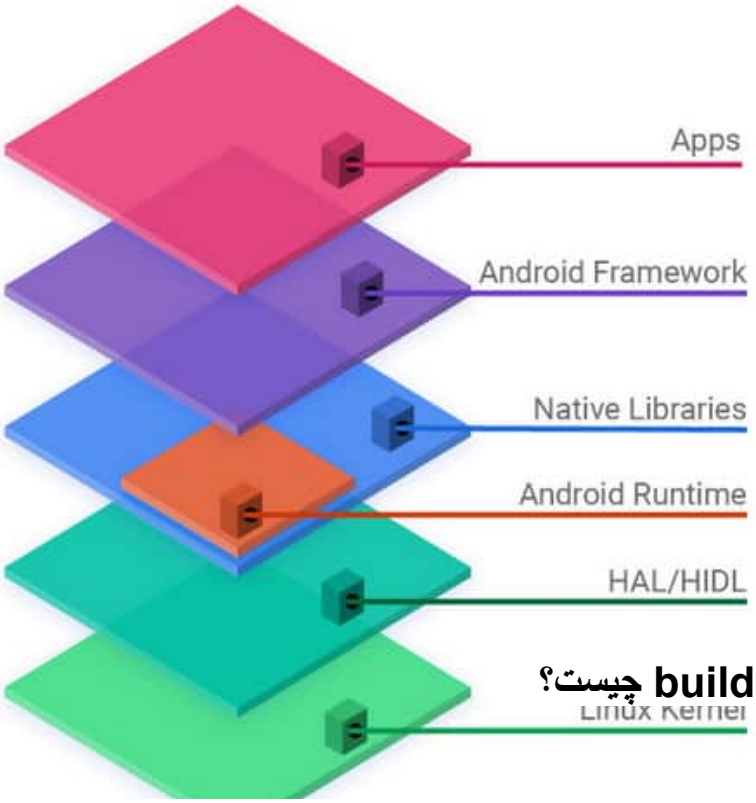
الف) AOSP چیست؟

اندروید (Android) یک سیستم عامل (OS) موبایل است که بر اساس نسخه تغییر یافته هسته لینوکس (linux kernel) و نرم افزارهای متن باز (open source) دیگری توسعه یافته است. به طور خاص از این سیستم عامل بر روی موبایل ها و تبلت ها استفاده می شد. به طور دقیقتر، اندروید توسط تجمیعی (consortium) از توسعه دهنده ها (developers) شناخته شده با عنوان Open Handset Alliance توسعه یافت. این تجمیع توسط گوگل (google) حمایت می شد [1].

اندروید، مجانی (free) و متن باز (open source) است (licensed under the Apache License). کد مرجع source code آن به عنوان Android Open Source Project یا همان AOSP شناخته می شود. این نکته قابل ذکر است که عموم دستگاه های اندرویدی از نسخه های شخصی سازی شده اندروید استفاده می کنند. از نسخه های شخصی سازی شده و سفارشی آن می توان به Fire OS و LineageOS اشاره کرد. نکته دیگری که شاید واضح باشد اما به آن اشاره می کنیم این است که به AOSP نمی توان به عنوان یک نسخه کامل برای یک دستگاه دلخواه نگاه کرد، چرا که بسیاری از driver های مربوط به سخت افزار ها ی آن دستگاه را ندارد. مثلا احتمالا نمی تواند از دوربین آن دستگاه در صورت وجود استفاده کند [1, 2].

سیستم عامل AOSP، مانند خیلی سیستم عامل های امروزی دیگر، از یک طراحی لایه ای استفاده می کند. این موضوع در شکل زیر آمده است. لایه ی ابتدایی آن Linux Kernel هست که مشابه هسته لینوکس است (با کمی تغییر که در بالا اشاره شد). پس از آن لایه Hardware Abstraction Layer یا همان HAL/HIDL را داریم، به طور کلی وظیفه این لایه ارتباط دادن بین app API ها با سخت افزار است. مثلا ارتباط بین API مربوط به sound و

device microphone. از این جا به بعد لایه ها توسط software developer های مختلف استفاده می شوند. لایه بعدی Native Libraries هست، این لایه شامل کتابخانه های (libraries) سطح پایین مانند OpenGL و یا Webkit هست. لایه بعدی یا همان Android Framework امکانات پایه ای اندروید را در اختیار توسعه دهندگان برنامه های اندروید قرار می دهد. مثلا امکانات مرتبط با pushing notifications یا making phone calls. در پایان نیز برنامه ها قرار دارند [2].



ب) build server چیست؟

- در ابتدا مفهوم build را بررسی میکنیم. در این حوزه build به فرآیند (process) تبدیل شدن کد مرجع (source code) به برنامه ای قابل اجرا (executable) بر روی یک کامپیوتر می گویند. این فرآیند عموما به استفاده از از یک ابزار build یا همان build tool صورت می گیرد [3].

- به طور ساده، build server یک server است که از آن برای build یک source code استفاده می شود. انجام شدن این فرآیند بر روی یک server شامل مزایای هست. عموما server ها از توانایی پردازش بالاتری نسبت به کامپیوتر های شخصی برخوردار هستند، در نتیجه می توانند فرآیند build شدن را با سرعت بالاتری انجام دهند. همینطور، عموما سرور ها دارای فضای بیشتری هستند، در نتیجه برای برخی از build ها که به فضای چند صد گیگابایتی احتیاج دارد کار را راحت خواهد کرد. در کنار این دو مورد، اغلب سرور ها دارای سرعت اینترنت بالاتری نسبت به اینترنت های خانگی هستند، در نتیجه دانلود فایل های پر حجم این پروژه را سریع تر انجام خواهند داد. در کنار این مزایای اولیه به موارد شاید مهم تری مانند 1. Controlled, dependable and repeatable و 2. Enforced unit tests و 3. Ensuring code quality اشاره کرد. توضیح جامع این موارد در [4] موجو است [4].

3. راه اندازی build server

- به عنوان پیش نیاز شروع پروژه لازم است با دستورات (commands) اولیه کار کردن با محیط ترمینال (terminal) لینوکس آشنا باشیم. این دستورات شامل دستورات ابتدایی از جمله cd و pwd و ls و du و mv و cp و mkdir و rm و chmod و grep و echo و nano و cat و vim و wget و gzip و screen و reboot و sudo و apt-get و ... هستند. همینطور شامل دستورات مرتبط با اتصال ssh به سرور، و دستورات کار کردن با گیت (git) نیز می شود. آشنایی با فایل های bash script نیز از پیش نیاز ها است.

- برای راه اندازی build server ما منابع مختلفی از جمله وبسایت های مختلف و ویدیو های متفاوتی را بررسی کردیم. از میان این منابع، وبسایت های [5, 7, 8, 9] مطالب بهتری نسبت به بقیه داشتند، همینطور ویدیوهای [10, 11, 14] از بقیه بهتر بودند. این نکته قابل ذکر است که این منابع عموما شامل یک روش تست نیز هستند، ما در بخش بعد به این موضوع خواهیم پرداخت. همینطور نکته مهم تر این است که، عملا همه ی این منابع از یک workflow استفاده می کنند، بین هر دوتای آنها یا تفاوتی وجود ندارد و یا تفاوت های جزئی در دستورات دیده می شود (به غیر از [5] که متعلق به خود android است و به حواشی بیشتری پرداخته است). همچنین، ویدیو [11] نیز در کنار build کردن به بررسی برخی از خطاهای (error) متداول می پردازد.

با این مقدمات حال به بررسی build کردن AOSP بر روی سرور می پردازیم. ابتدایی ترین نکته این است که آیا سرور ما توانایی انجام این build v را دارد؟ برای این کار کافی است تا config سرور را با config پیشنهادی توسط یکی از لینک ها مثلا [7] چک کنیم. سروری که در اختیار ما است به راحتی می تواند این کار را انجام دهد. حال به سراغ مراحل اصلی می رویم:

الف) نصب وابستگی *(dependency)*: در این مرحله با استفاده از دستور apt-get و apt به نصب همه ی وابستگی ها می پردازیم. قبل از نصب نیز می توانیم با استفاده از همین دستور از به روز بودن (update) تمامی موارد نصب شده از قبل اطمینان خاطر پیدا کنیم. دستورات به شرح زیر است:

```
sudo apt-get update
sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386
```

lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z-dev libgl1-mesa-dev libxml2-utils xsltproc unzip

sudo apt install android-tools-adb android-tools-fastboot

ب) نصب و *set* کردن *repo launcher:* در این مرحله با استفاده از دستورات زیر به نصب و set کردن repo launcher می پردازیم. repo یک ابزار است که گوگل از آن برای مدیریت و ... کد AOPS بر روی ماشین ما استفاده می کند. دستورات به شرح زیر است:

```
mkdir ~/bin
PATH=~/bin:$PATH
curl https://storage.googleapis.com/git-repo-downloads/repo >
~/bin/repo
chmod a+x ~/bin/repo
```

ج) دانلود *AOSP:* در این مرحله به ساختن پوشه برای دانلود شدن محتویات AOSP و همینطور دانلود آن می پردازیم. دستورات به شرح زیر است:

```
mkdir ~/aosp
cd ~/aosp
repo init -u https://android.googlesource.com/platform/manifest -b
android-10.0.0_r10 --depth=1
repo sync -qc -j4
```

در مورد این قسمت به چند نکته اشاره می کنیم. اولا، در دستور سوم build tag را با -b مشخص می کنیم (branch). بر اساس نیازمان می توانیم آن را مثلا متناسب با ورژن اندروید یا نوع دستگاه انتخاب کنیم. همینطور می توانیم این بخش را نداشته باشیم و همه را دانلود کنیم. مطلب بعدی --depth=1 در همان دستور سوم هست، این قسمت منجر به دانلود آخرین ورژن (version) خواهد شد. برای دستور چهارم، q به این معنی است که دانلود به صورت ساکت

(silent) انجام شود. یعنی فقط پیشرفت کلی نشان داده شود . در همین دستور، حرف c مطمئن می شود تا فقط branch انتخابی دانلود شود. و در آخر باز هم در همین دستور، j و عدد بعد از آن تعداد پردازنده های درگیر را مشخص می کند. برای یافتن تعداد هسته (core) های ماشین خود می توانیم از دستر ساده nproc --all بهره ببریم.

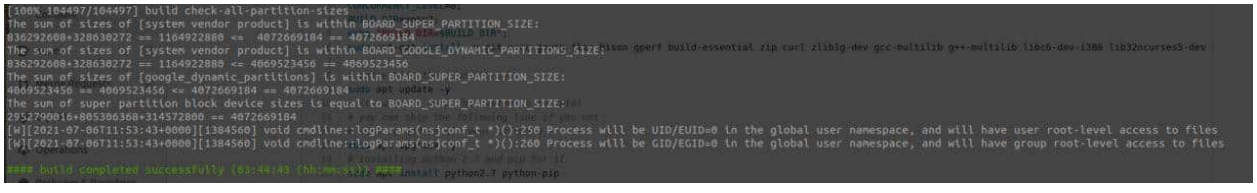
د) ***build*** کردن ***source code*** دانلود شده: در این بخش به build کردن source code دانلود شده در بخش قبل می پردازیم. دستورات به شرح زیر است:

```
source build/envsetup.sh
lunch aosp_blueline-userdebug
m droid -j4
```

در مورد این قسمت به چند نکته اشاره می کنیم. دستور اول یک دستور اولیه و لازم است. بعد از زدن دستور اول می توانیم از hmm برای دیدن دستورات موجود استفاده کنیم. دستور دوم پارامترهای مربوط به مدل دلخواه ما برای build را set می کند، در مورد آن می توانید در [19] بیشتر بخوانید. در بخش بعد نیز یک مدل دیگر را انتخاب خواهیم کرد! دستور سوم نیز شامل یک droid است که یک بخش دلخواه است و برای اشاره به تنظیمات پیش فرض (default) استفاده می شود. این دستور یک بخش j به همراه یک عدد بعد از آن نیز دارد که مانند قبل است. در واقع m دستر اصلی build کردن است!

تصویر زیر مربوط به پیام پایانی build موفق ما است. این پیام به طور کلی با فرمت زیر است که در آن a و b و c و d و e و f پارامترهای مناسب هستند:

[COLOR="SeaGreen"]#### build completed successfully (ab:cd:ef (hh:mm:ss)) ####[/COLOR]



- در پایان این نکته قابل ذکر است که می توانیم از دستور screen به این منظور که قطع شدن ارتباط ssh منجر به قطع شدن اجرا نشود استفاده کنیم. به علاوه [15] راه حل خطایی هست که ما در حین اجرا با آن رو به رو شدیم.

4. تست build server

- برای تست کردن build server ما منابع مختلفی از جمله وبسایت های مختلف و ویدیو های متفاوتی را بررسی کردیم. از میان این منابع، وبسایت های [5, 6, 7, 12 + 1] مطالب بهتری نسبت به بقیه داشتند، همینطور ویدیوهای [10, 12, 14] از بقیه بهتر بودند.

برای تست کردن نسخه های build شده دو راه حل کلی وجود دارد. به طور خلاصه، الف) می توانیم آن را بر روی یک دستگاه واقعی تست کنیم. ب) می توانیم آن را با استفاده از شبیه سازی تست کنیم. مورد ب خود دو حالت دارد: 1. شبیه سازی به صورت مستقیم روی دستگاهی که build بر روی آن انجام شده است صورت گیرد. 2. شبیه سازی به صورت غیر مستقیم و به طور دقیقتر از راه دور (remote) صورت گیرد. لازم به ذکر است که با استفاده از ب امکان تست نسخه های همه ی دستگاه های مختلف وجود ندارد. حال به بررسی دقیق هر کدام از این حالات می پردازیم:

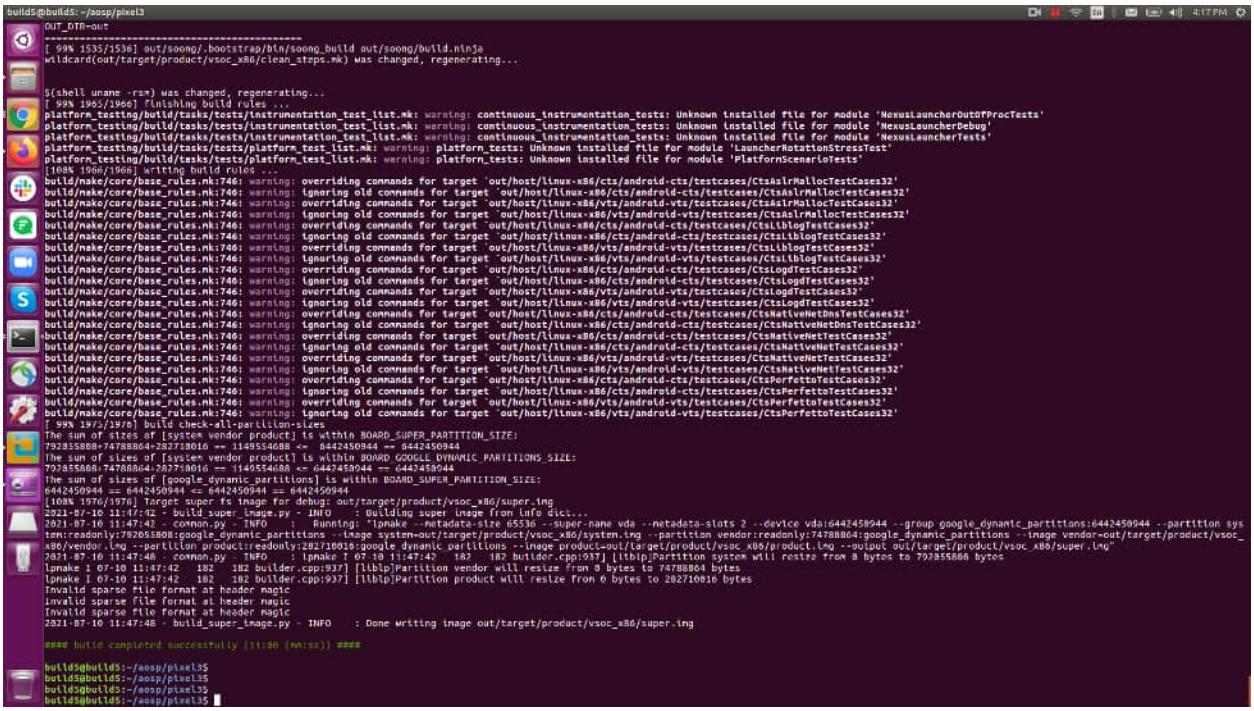
الف) دستگاه واقعی: برای این منظور می توانیم کد های موجود در [5, 7] را مورد استفاده قرار دهیم. همینطور، می توانیم از آموزش ویدیو های [10, 12] استفاده کنیم. در مجموع کار بسیار سر راست است، کافی است تا دستگاهی اندرویدی در دست داشته باشیم (باید root باشد، و این عمل عموما منجر به باطل شدن گارانتی دستگاه خواهد شد)، نسخه AOSP متناسب با آن را

build کنیم، دستگاه اندرویدی را به حالت مناسب ببریم، و با وصل کردن دستگاه به کامپیوتری که build بر روی آن انجام شده است اقدام به نصب AOSP بکنیم. به این صورت می توانیم از درستی build خود مطمئن شویم. ما این روش را به دو دلیل استفاده نکردیم: 1. دستگاهی اندرویدی که شرایط این کار را داشته باشد در اختیار نداشتیم. 2. تا آنجایی که ما جستجو (search) کردیم راهی برای انجام این کار به صورت از راه دور (remote) وجود ندارد و ما دسترسی فیزیکی به سرور نداشتیم.

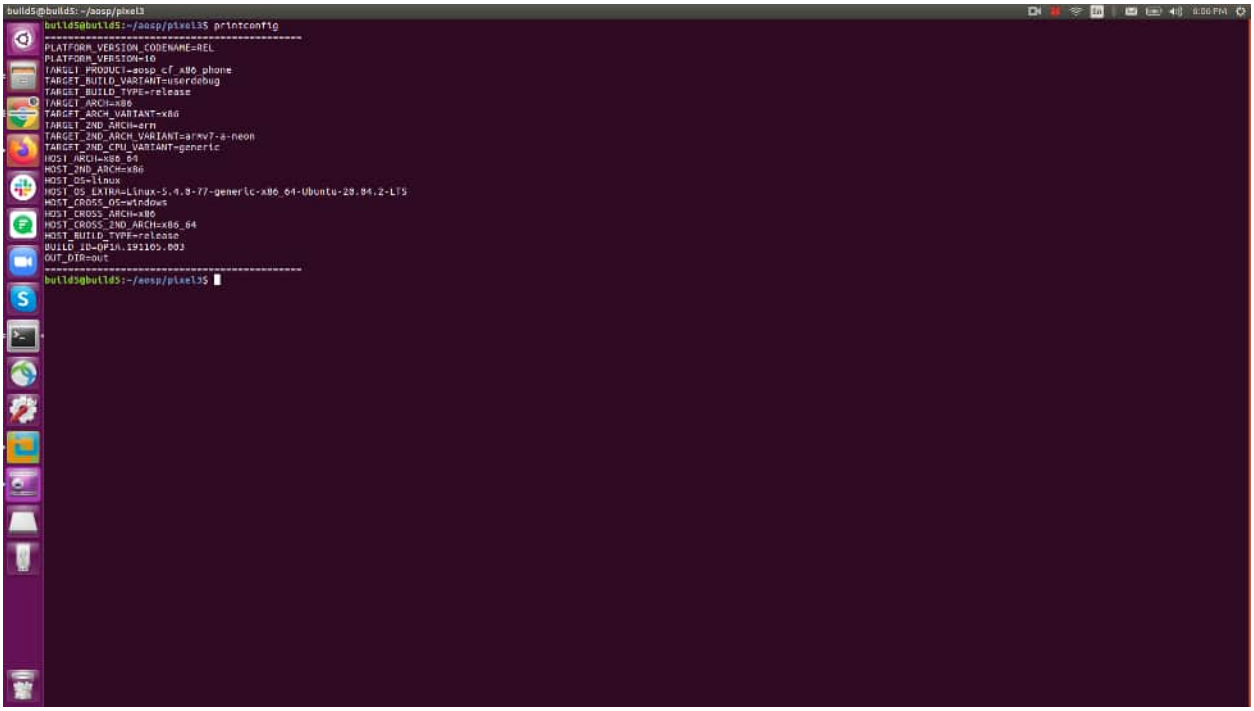
ب) شبیه سازی به صورت مستقیم روی دستگاهی که ***build*** بر روی آن انجام شده است: برای این منظور می توانیم کد موجود در [6] را مورد استفاده قرار دهیم. در مجموع کار بسیار بسیار سر راست است، کافی است تا مثلا نسخه aosp_cf_x86_phone-userdebug را build کنیم، و از دستور emulator استفاده کنیم! به این صورت می توانیم از درستی build خود مطمئن شویم. ما این روش را استفاده نکردیم زیرا این محیط گرافیکی تا جایی که ما جستجو (search) کردیم قابل اجرا به صورت از راه دور (remote) نیست.

ج) شبیه سازی به صورت غیر مستقیم و به طور دقیقتر از راه دور ***(remote):*** برای این منظور می توانیم کد موجود در [12 + 1] را مورد استفاده قرار دهیم. همینطور، می توانیم از آموزش ویدیو [14] استفاده کنیم. ویدیو [14] بسیار زیبا است!

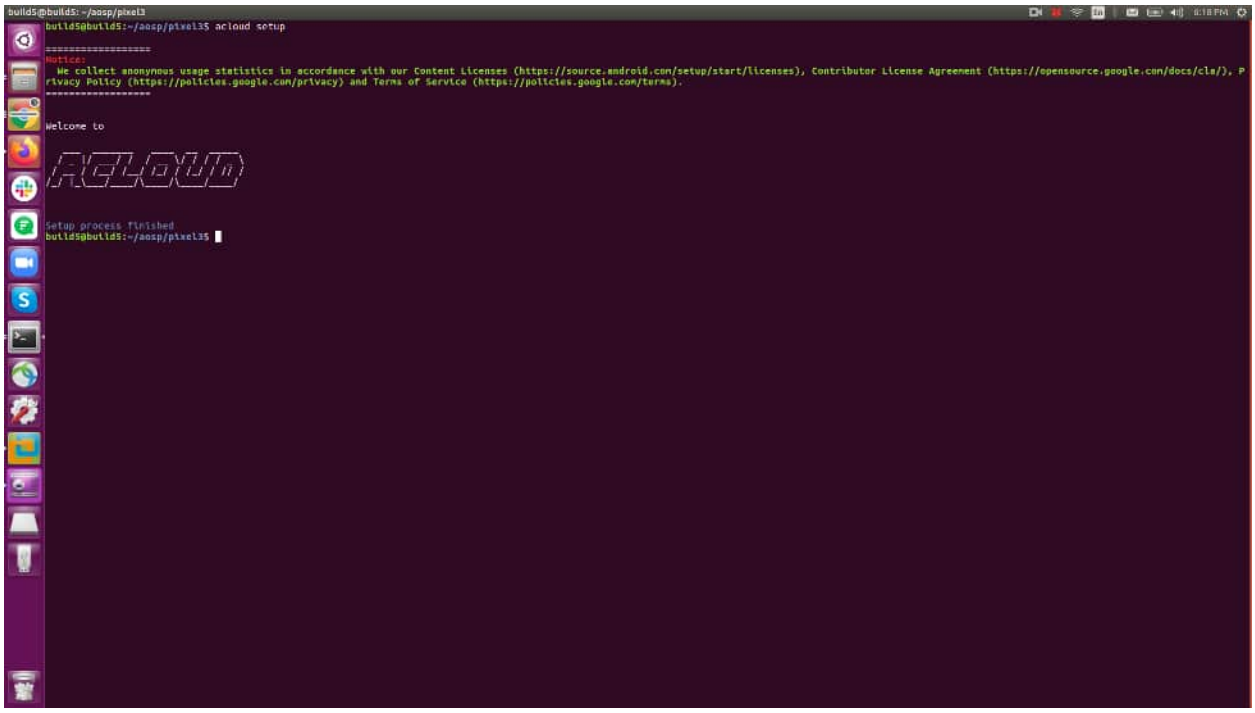
برای این قسمت در ابتدا باید aosp_cf_x86_phone-userdebug را build کنیم. برای این کار از همان دستورات قبلی در زیر بخش د از قسمت سوم این گزارش استفاده می کنیم، فقط کافی است نسخه را به aosp_cf_x86_phone-userdebug تغییر دهیم. نتیجه نهایی build ما در این قسمت در زیر آمده است، فرمت آن مانند قبل است.



همینطور با استفاده از دستور printconfig میتوانیم این موضوع را بررسی کنیم. تصویر مربوط به این قسمت در زیر آمده است.



حال می بایست از acloud استفاده کنیم. از این ابزار برای اجرای شبیه سازی بر روی کامپیوتر خودمان زمانی که build بر روی دستگاه ما نیست می توان استفاده کرد. حالات دیگری که acloud پشتیبانی می کند، و همینطور دستورات مرتبط با آن را می توانید در [20] ببینید. مشکل اساسی استفاده از این امکان تحریم ایران است. برای دور زدن تحریم dns خود را تغییر می دهیم. چگونگی انجام این کار از طریق terminal را می توانید در [17] نگاه کنید. ما از dns های "شکن" استفاده کردیم [18]. به این صورت می توانیم acloud را راه اندازی کنیم. برای این منظور از دستور acloud setup استفاده کردیم. تصویر مربوط به این مرحله در زیر آمده است.



حال باید ایمیج مربوط به نسخه cuttlefish ساخته شود و شبیه سازی در کامپیوتر ما اجرا شود. برای این منظور می توانیم از دستور acloud create --local-instance 1 --local-image استفاده کنیم. اما مشکل اصلی این است که این فرآیند در نهایت به تنظیمات گوناگونی در google cloud account نیاز دارد. چگونگی انجام آنها را می توانید در [21]

نگاه کنید. متاسفانه در مرحله پایانی این تنظیمات و در واقع مرحله پایانی اجرای شبیه سازی بر روی دستگاه خودمان نیاز به تایید billing account داریم. به همین علت توانستیم این مرحله را کامل کنیم. تصاویر مربوط به این قسمت در پایین آمده است.

این نکته قابل ذکر است که ما از حالت توسعه دهنده (developer) ابزار acloud نیز استفاده کردیم، اما باز هم نتیجه مشابه قبل بود [16].

در پایان نیز می توانید ویدیو مربوط به build شدن cuttlefish را در فایل M0.mp4 مشاهده کنید.

5. نتیجه گیری

- در این پروژه ما در ابتدا با مفاهیم AOSP و Build Server آشنا شدیم، و در کنار آنها مطالب جانبی را نیز یاد گرفتیم. سپس به build کردن AOSP بر روی سرور داده شده پرداختیم، و منابع مختلفی را برای انجام این کار بررسی کردیم. در پایان نیز با انواع روش های تست کردن build server راه اندازی شده آشنا شدیم، و تا جای ممکن روش قابل انتخاب را جلو بردیم.

- برای اجرای کاملا اتوماتیک عموم قسمت سوم و چهارم (به جز وارد کردن اکانت گیت (git) و اکانت ابری گوگل (google cloud) بقیه قسمت ها اتوماتیک هستند) می توانید bash script های ما را در پوشه پروژه مشاهده کنید. همینطور، می توانید screenshot های ما را در پوشه پروژه نگاه کنید.

سپاس گزاری

در اینجا می خواهیم از زحمات آقای باجلان، مسئول پروژه ما، تشکر کنیم. همینطور، از شرکتی که سرور را به صورت مجانی در اختیار ما قرار داد تشکر می کنیم.

توضیحات پایانی

- ما برای این پروژه با جناب آقای تشکری یک جلسه اولیه در دو نوبت در ماه اسفند برای set کردن موضوع پروژه داشتیم. پس از با جناب آقای باجلان یک جلسه در عید برای بحث در مورد پروژه داشتیم و سرور را از ایشان تحویل گرفتیم. از آن به بعد با جناب آقای باجلان از طریق پیام رسان تلگرام در ارتباط بودیم.

- در مورد تحول سرور، امیررضا شعیری مسئولیت تحویل گرفتن سرور را از جناب آقای باجلان بر عهده گرفت.

- این پروژه به صورت ۶ نفره تعریف شده بود. اما گروه ۳ نفره ما با مشورت با جناب آقای تشکری این پروژه را انتخاب کرد. متاسفانه تقریبا از بعد از جلسه با جناب آقای باجلان یکی از اعضا (علی اسلامی نژاد) ناپدید شده است و ما از او هیچ گونه خبری نداریم. امیدواریم سلامت باشند.

منابع

[1] https://en.wikipedia.org/wiki/Android_(operating_system)#AOSP
[2] https://www.androidauthority.com/amp/aosp-explained-1093505
[3] https://www.techopedia.com/definition/3759/build

[4] https://searchapparchitecture.techtarget.com/tip/Why-should-I-use-a-build-server
[5] https://source.android.com/setup/build/ (build and emulator)
[6] https://source.android.com/setup/build/running#flashing-a-device (flash)
[7] https://forum.xda-developers.com/t/noobs-guide-to-building-aosp-from-scratch.4012293 (build and flash)
[8] https://www.raywenderlich.com/10197539-building-the-android-open-source-project (build and edit)
[9] https://medium.com/@ryanburnsworth/building-for-aosp-setting-up-a-windows-based-build-environment-912b612616fc (build)
[10] https://youtu.be/-OePyL55rvs (build and flash)
[11] https://m.youtube.com/watch?v=2zVARyqTFtE (build)
[12] https://youtu.be/7Z-uv1N2Pkk (flash)
[12 + 1] https://source.android.com/setup/create/cuttlefish (cuttlefish)
[14] https://youtu.be/eaPUdDkE91g (cuttlefish)

[15] https://askubuntu.com/questions/1252062/how-to-install-libncurses-so-5-in-ubuntu-20-04
[16] https://groups.google.com/g/android-building/c/jfhG1dQ0KRU

[17] https://support.strongvpn.com/hc/en-us/articles/360038982774-How-to-Change-DNS-in-Linux
[18] https://shecan.ir/tutorials

[19] https://source.android.com/setup/build/building.html#choose-a-target
[20] https://android.googlesource.com/platform/tools/acloud/+/refs/heads/master/README.md#create

[21] https://www.youtube.com/watch?v=h3WuYoMzBj0&t=0s (acloud)