Busybox Integration on Android

Botao Sun (botao.sun@linaro.org)

Android Builders Summit

Why we do this?

1. Arming your board:
   - More features, more possibilities.

2. Re-use existing open source software:
   - DO NOT reinvent the wheel.

3. Meet customer requirements:
   - Make a highly customized system to finish a specific job, such like: automation, computation...
   - The control is in your hands, not the manufactures.

Preparation

1. Hardware:
- 64-bit Dual-Core CPU
- Hard disk with enough free space - 100GB+
- 4GB+ RAM
- Linaro development board, for example: Samsung Origen board

2. Software:
- 64-bit Linux distribution, Ubuntu or Debian recommended

Preparation

3. Get busybox source code:

- Download source code package:
  - `http://busybox.net/downloads/`

- Git clone (install git first if you don't have):
  - `http://busybox.net/source.html`
    - `git clone git://busybox.net/busybox.git`
  - For git install: `http://git-scm.com/`

Preparation

4. Get repo (install curl first if you don't have):
   - For curl install: `http://curl.haxx.se/`
   - `$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo`
   - `$ chmod a+x ~/bin/repo`
5. Get Linaro tool chain:
   - `https://android-build.linaro.org/builds/~linaro-android/toolchain-4.6-bzr/`

Preparation

6. Get Linaro Android platform source code:

- `$ repo init -u git://android.git.linaro.org/platform/manifest.git -b linaro_android_4.0.3 -m staging-origen.xml`
- `$ repo sync`

It will require user name & email address during the `repo init`, just follow the instructions.

Preparation

7. Initializing a Build Environment:

- Refer to these 2 web pages to install the packages which you don't have:
  - `http://source.android.com/source/initializing.html`
  - `http://bazaar.launchpad.net/~linaro-infrastructure/linaro-android-build-tools/trunk/view/head:/node/setup-build-android`

A full build

1. Enter Android root directory, then run:
   - `$ . build/envsetup.sh`
2. Extract tool chain package then run:

   -
     ```
     $ make -j4 HOST_CC=gcc-4.5
     HOST_CXX=g++-4.5 HOST_CPP=cpp-4.5
     TARGET_PRODUCT=origen
     TARGET_SIMULATOR=false
     TARGET_TOOLS_PREFIX=/your_toolchain_
     path/bin/arm-linux-androideabi-
     boottarball systemtarball
     userdatatarball showcommands
     ```

A full build

3. A better build command:

-
  ```
  $ make -j4 HOST_CC=gcc-4.5
  HOST_CXX=g++-4.5 HOST_CPP=cpp-4.5
  TARGET_PRODUCT=origen
  TARGET_SIMULATOR=false
  TARGET_TOOLS_PREFIX=/your_toolchain_
  path/bin/arm-linux-androideabi-
  boottarball systemtarball
  userdatatarball showcommands >
  build_log_YYMMDD.txt 2>&1 &
  ```

A full build

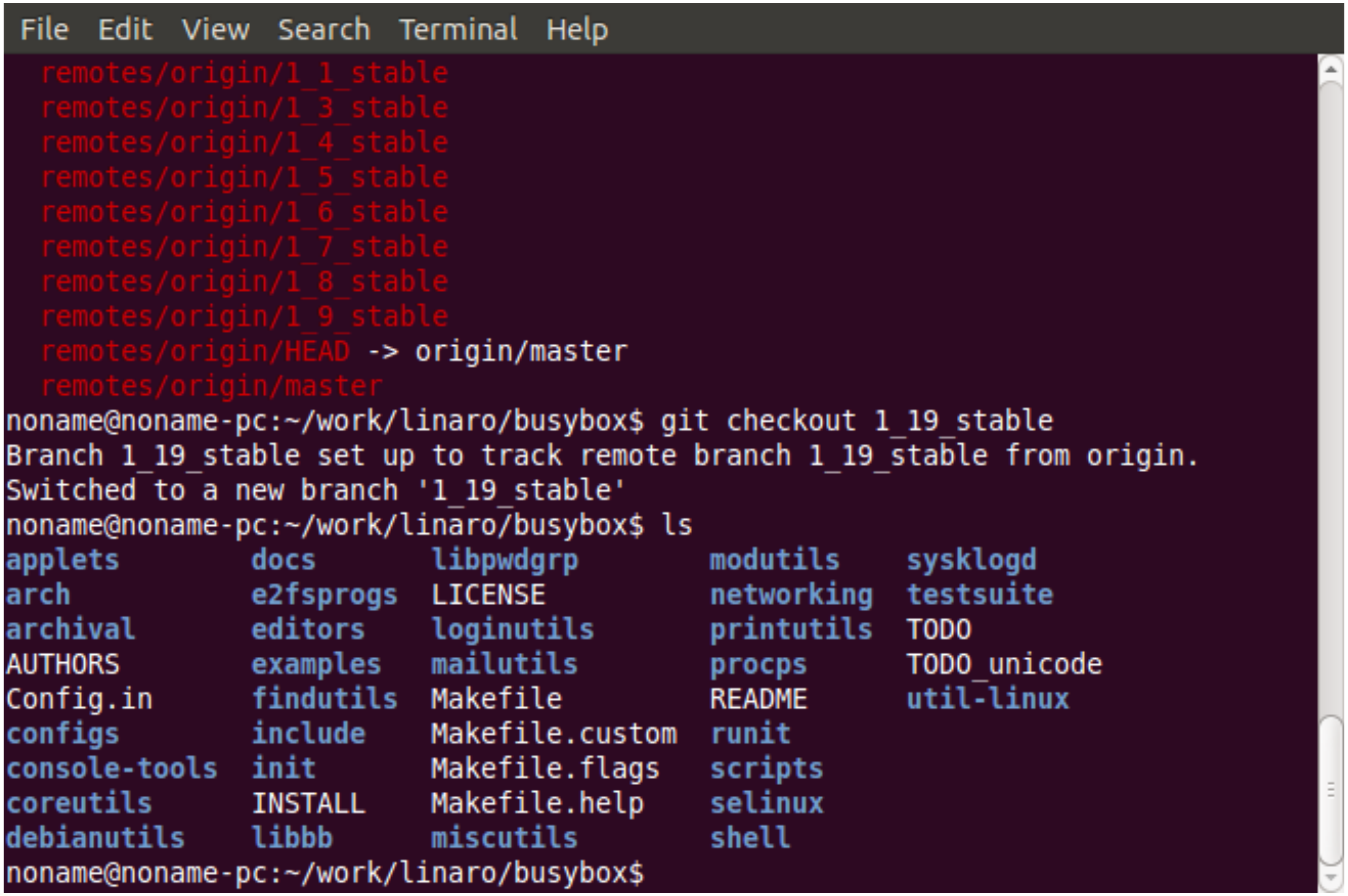
- On my Lenovo ThinkPad T420 laptop, with Intel i5-2410M CPU (2.30GHz), 4GB RAM, 500GB 7200RPM hard disk, a full build will take 90+ minutes.

- Once the build is done, you will find 3 target packages: "`boot.tar.bz2`", "`system.tar.bz2`" and "`userdata.tar.bz2`" under the path: `/your_Android_root/out/target/product/origen`

Build Busybox out of platform

- Before building Busybox against to the whole Linaro Android platform, we'll build it with the Linaro tool chain individually.
- In this step we will familiarize ourselves with the build architecture of Busybox.
- Your Busybox source code should look like this:

Build Busybox out of platform

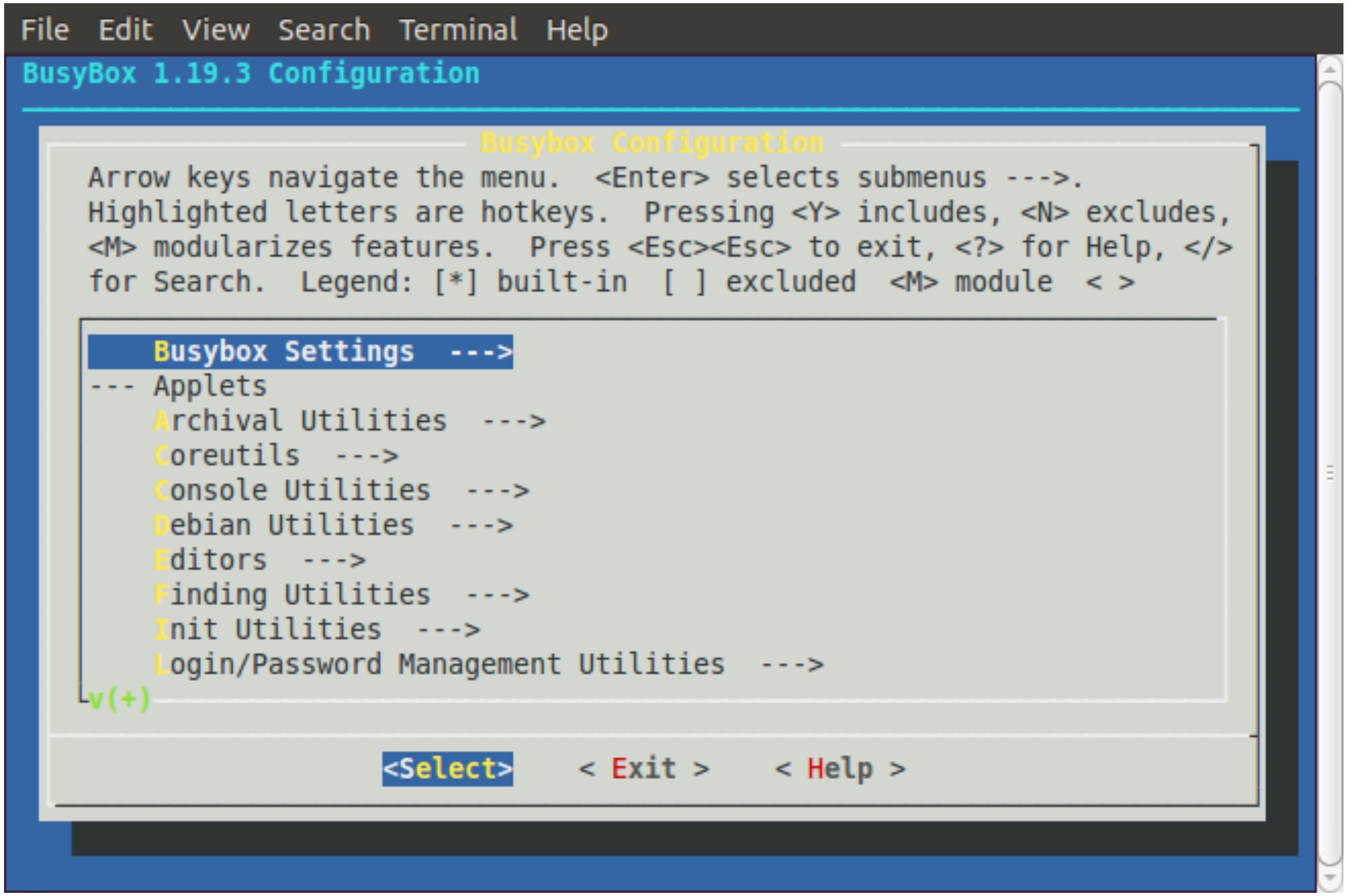


```
  remotes/origin/1_1_stable
  remotes/origin/1_3_stable
  remotes/origin/1_4_stable
  remotes/origin/1_5_stable
  remotes/origin/1_6_stable
  remotes/origin/1_7_stable
  remotes/origin/1_8_stable
  remotes/origin/1_9_stable
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
noname@noname-pc:~/work/linaro/busybox$ git checkout 1_19_stable
Branch 1_19_stable set up to track remote branch 1_19_stable from origin.
Switched to a new branch '1_19_stable'
noname@noname-pc:~/work/linaro/busybox$ ls
applets        docs       libpwdgrp        modutils    sysklogd
arch           e2fsprogs  LICENSE          networking  testsuite
archival       editors    loginutils       printutils  TODO
AUTHORS        examples   mailutils        procps      TODO_unicode
Config.in      findutils  Makefile         README      util-linux
configs        include    Makefile.custom  runit
console-tools  init       Makefile.flags   scripts
coreutils      INSTALL    Makefile.help    selinux
debianutils    libbb      miscutils        shell
noname@noname-pc:~/work/linaro/busybox$
```

botao_sun
Linaro Android Platform Team

Build Busybox out of platform

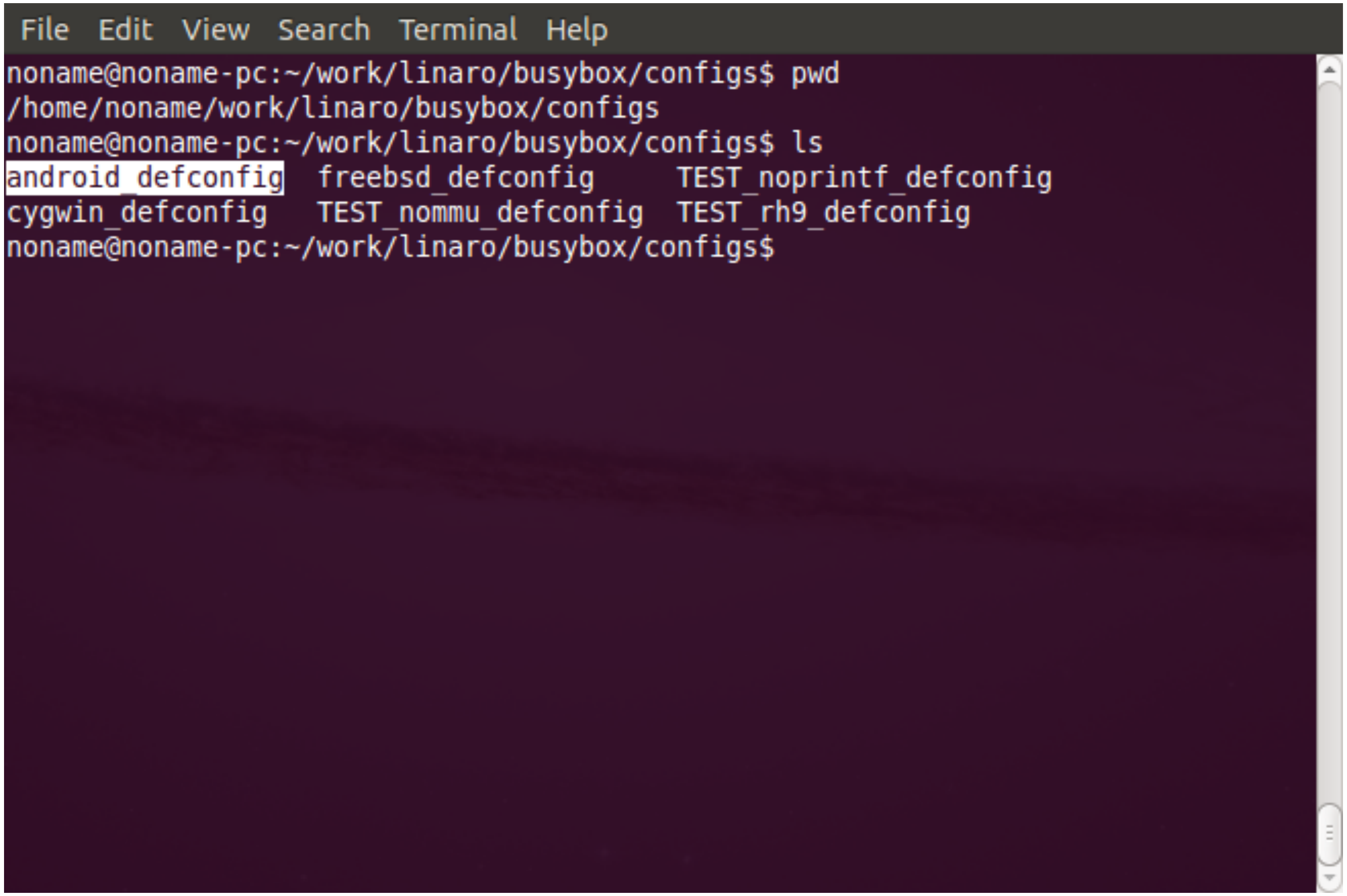
- Run: `$ make menuconfig`



```
File  Edit  View  Search  Terminal  Help
BusyBox 1.19.3 Configuration
─────────────────────────────────────────────
            Busybox Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

      Busybox Settings  --->
  --- Applets
      Archival Utilities  --->
      Coreutils  --->
      Console Utilities  --->
      Debian Utilities  --->
      Editors  --->
      Finding Utilities  --->
      Init Utilities  --->
      Login/Password Management Utilities  --->
  v(+)

           <Select>    < Exit >    < Help >
```

Build Busybox out of platform

Configuration file for Android:

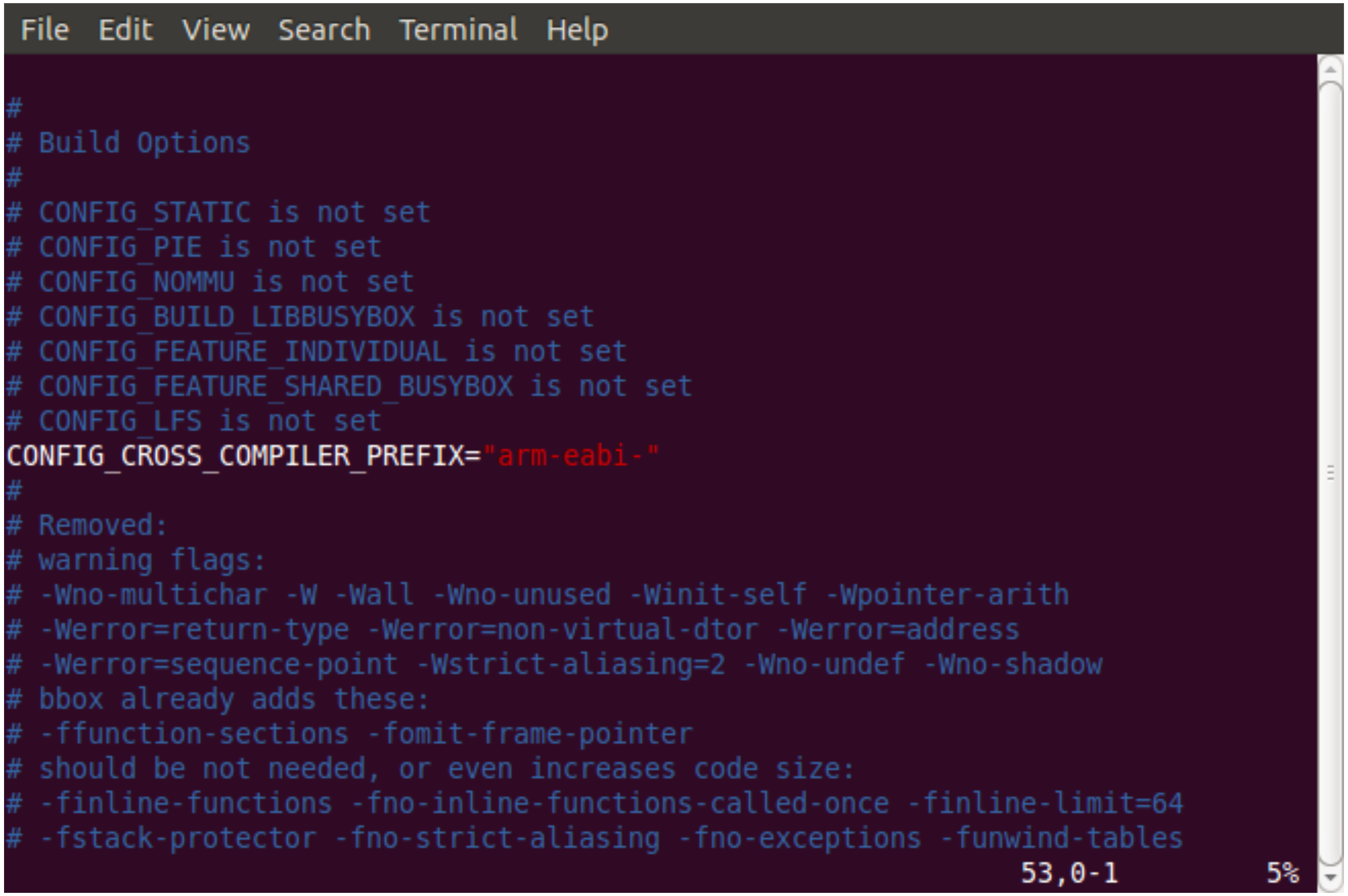


```
noname@noname-pc:~/work/linaro/busybox/configs$ pwd
/home/noname/work/linaro/busybox/configs
noname@noname-pc:~/work/linaro/busybox/configs$ ls
android_defconfig  freebsd_defconfig     TEST_noprintf_defconfig
cygwin_defconfig   TEST_nommu_defconfig  TEST_rh9_defconfig
noname@noname-pc:~/work/linaro/busybox/configs$
```

botao_sun
Linaro Android Platform Team

Build Busybox out of platform

Build options in configuration file for Android:



```
#
# Build Options
#
# CONFIG_STATIC is not set
# CONFIG_PIE is not set
# CONFIG_NOMMU is not set
# CONFIG_BUILD_LIBBUSYBOX is not set
# CONFIG_FEATURE_INDIVIDUAL is not set
# CONFIG_FEATURE_SHARED_BUSYBOX is not set
# CONFIG_LFS is not set
CONFIG_CROSS_COMPILER_PREFIX="arm-eabi-"
#
# Removed:
# warning flags:
# -Wno-multichar -W -Wall -Wno-unused -Winit-self -Wpointer-arith
# -Werror=return-type -Werror=non-virtual-dtor -Werror=address
# -Werror=sequence-point -Wstrict-aliasing=2 -Wno-undef -Wno-shadow
# bbox already adds these:
# -ffunction-sections -fomit-frame-pointer
# should be not needed, or even increases code size:
# -finline-functions -fno-inline-functions-called-once -finline-limit=64
# -fstack-protector -fno-strict-aliasing -fno-exceptions -funwind-tables
                                                               53,0-1          5%
```

Build Busybox out of platform

Customize the configuration file:

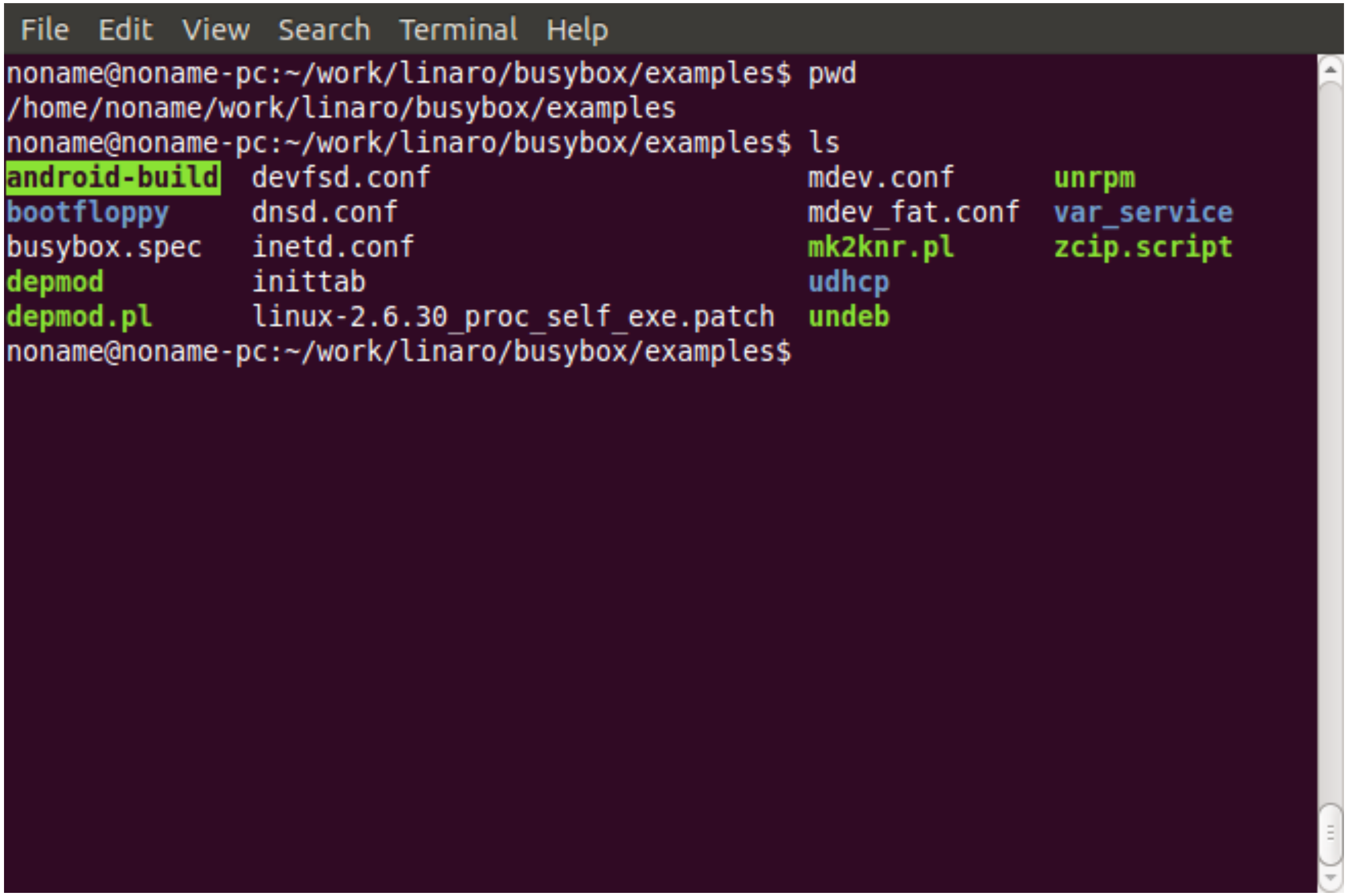
- Set your own tool chain path and prefix;
- Enable more applets;
- Modify other configuration items you want.

Tips for configuration file customization:

- Usually we build Busybox in dynamic mode;
- Enable applet one-by-one;
- Backup your configuration file.

Build Busybox out of platform

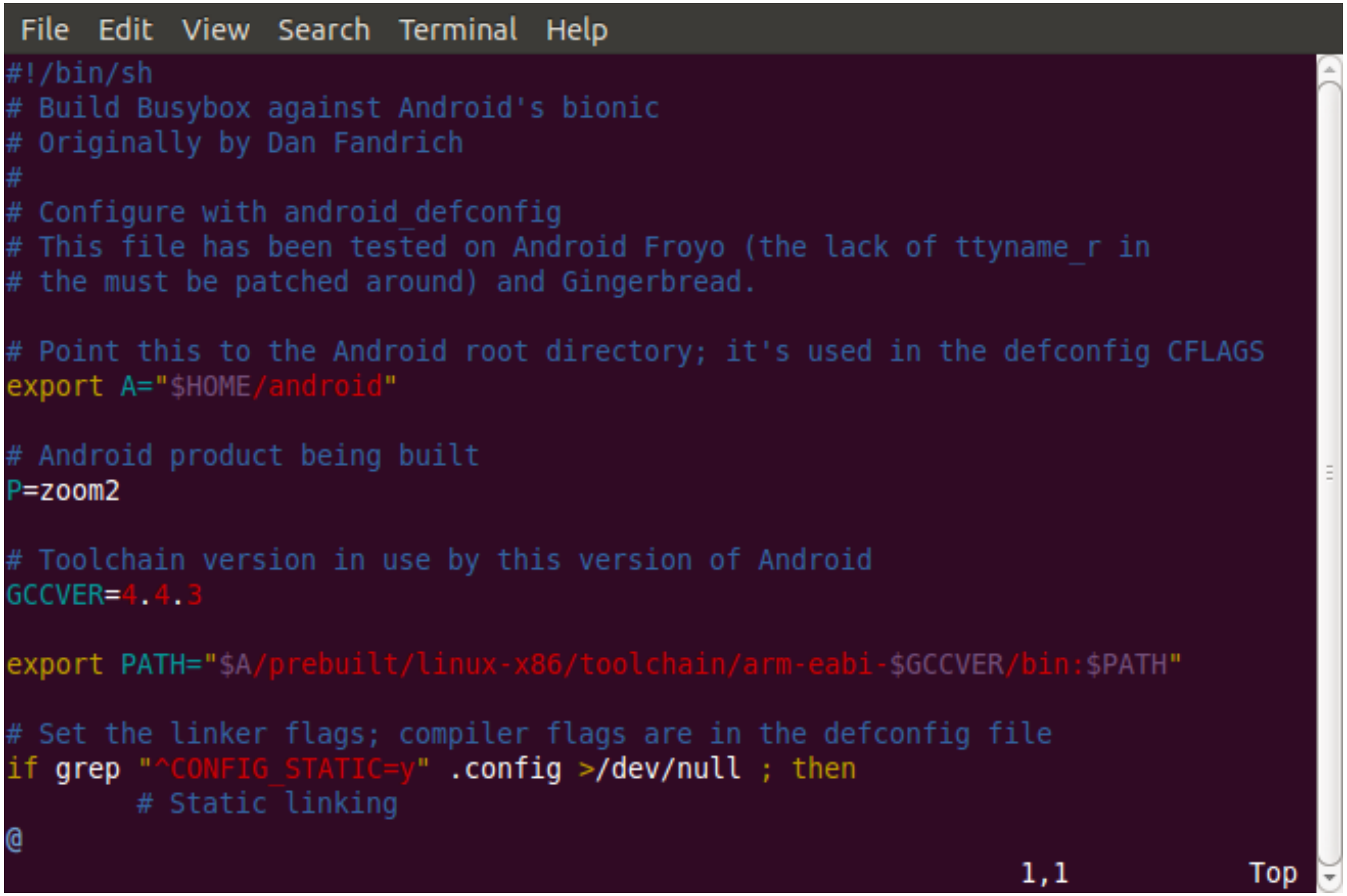
Use "`android-build`" script:



```
noname@noname-pc:~/work/linaro/busybox/examples$ pwd
/home/noname/work/linaro/busybox/examples
noname@noname-pc:~/work/linaro/busybox/examples$ ls
android-build  devfsd.conf                      mdev.conf      unrpm
bootfloppy     dnsd.conf                        mdev_fat.conf  var_service
busybox.spec   inetd.conf                       mk2knr.pl      zcip.script
depmod         inittab                          udhcp
depmod.pl      linux-2.6.30_proc_self_exe.patch undeb
noname@noname-pc:~/work/linaro/busybox/examples$
```

Build Busybox out of platform

Look into "`android-build`" script:



```
#!/bin/sh
# Build Busybox against Android's bionic
# Originally by Dan Fandrich
#
# Configure with android_defconfig
# This file has been tested on Android Froyo (the lack of ttyname_r in
# the must be patched around) and Gingerbread.

# Point this to the Android root directory; it's used in the defconfig CFLAGS
export A="$HOME/android"

# Android product being built
P=zoom2

# Toolchain version in use by this version of Android
GCCVER=4.4.3

export PATH="$A/prebuilt/linux-x86/toolchain/arm-eabi-$GCCVER/bin:$PATH"

# Set the linker flags; compiler flags are in the defconfig file
if grep "^CONFIG_STATIC=y" .config >/dev/null ; then
        # Static linking
@
                                                                      1,1           Top
```

Build Busybox out of platform

Compile the source:

- Enter your Busybox source code directory then run:
  - `$ make android_defconfig`
- Run:
  - `$ ./examples/android-build`
- If everything is OK, you will get the binary "`busybox`" file under your source code directory.

Build Busybox out of platform

Tips:

- While Busybox compiles, it will link to the libraries which have been generated from the whole platform build. Therefore, if you already made a platform build, you will find that Busybox compilies quickly - it take less than `1` minute.
- Because our compiling mode is dynamic, in order to run busybox, you have to transfer it to the directory "`/system/bin`" on your board.

Build Busybox in platform

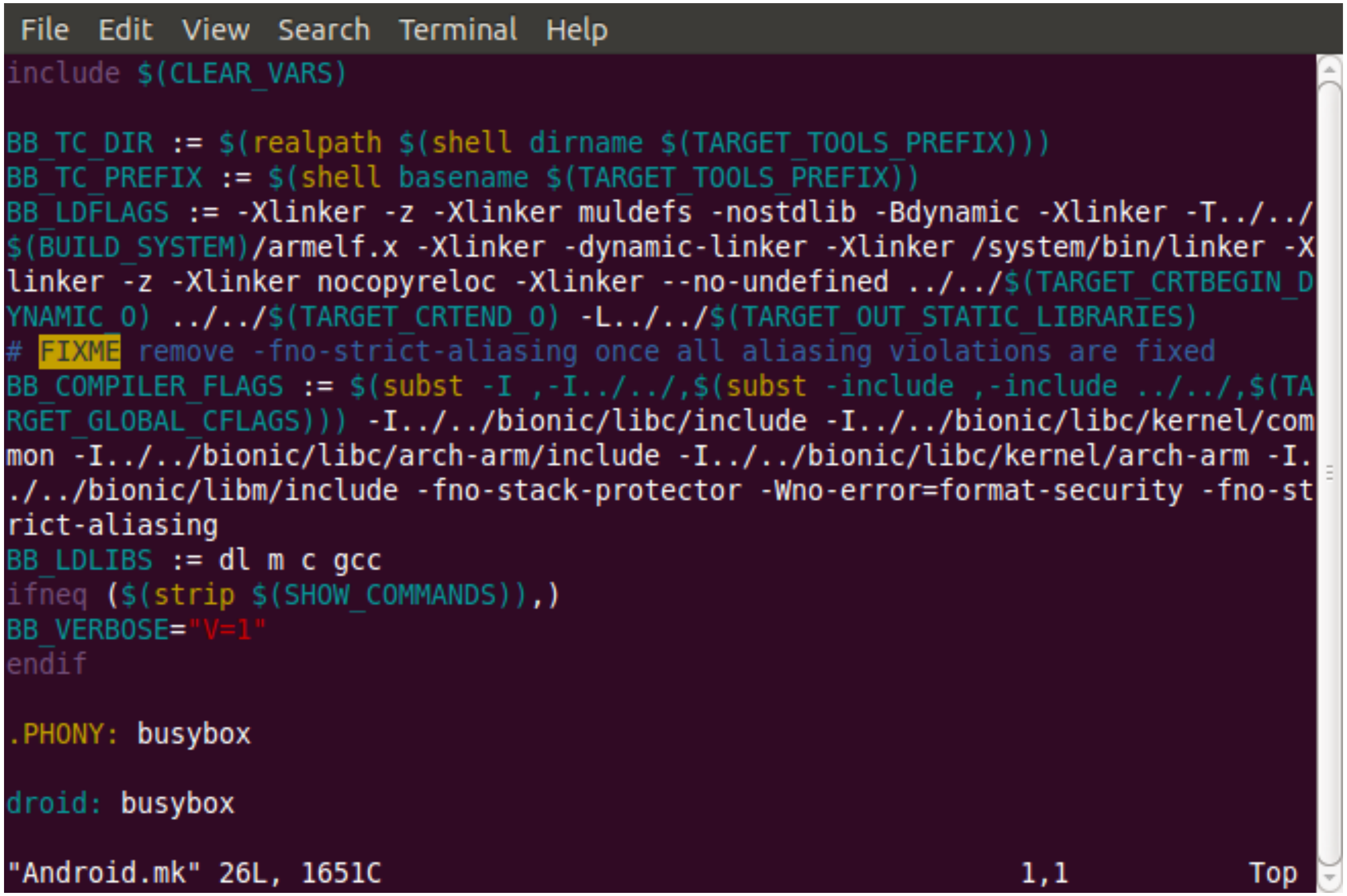
Put Busybox source code here:

- `/your_Android_root_directory/external/busybox`

Write a "`Android.mk`" file then put it in Busybox source code directory.

Build Busybox out of platform

Write "Android.mk" file for Busybox - Part I:

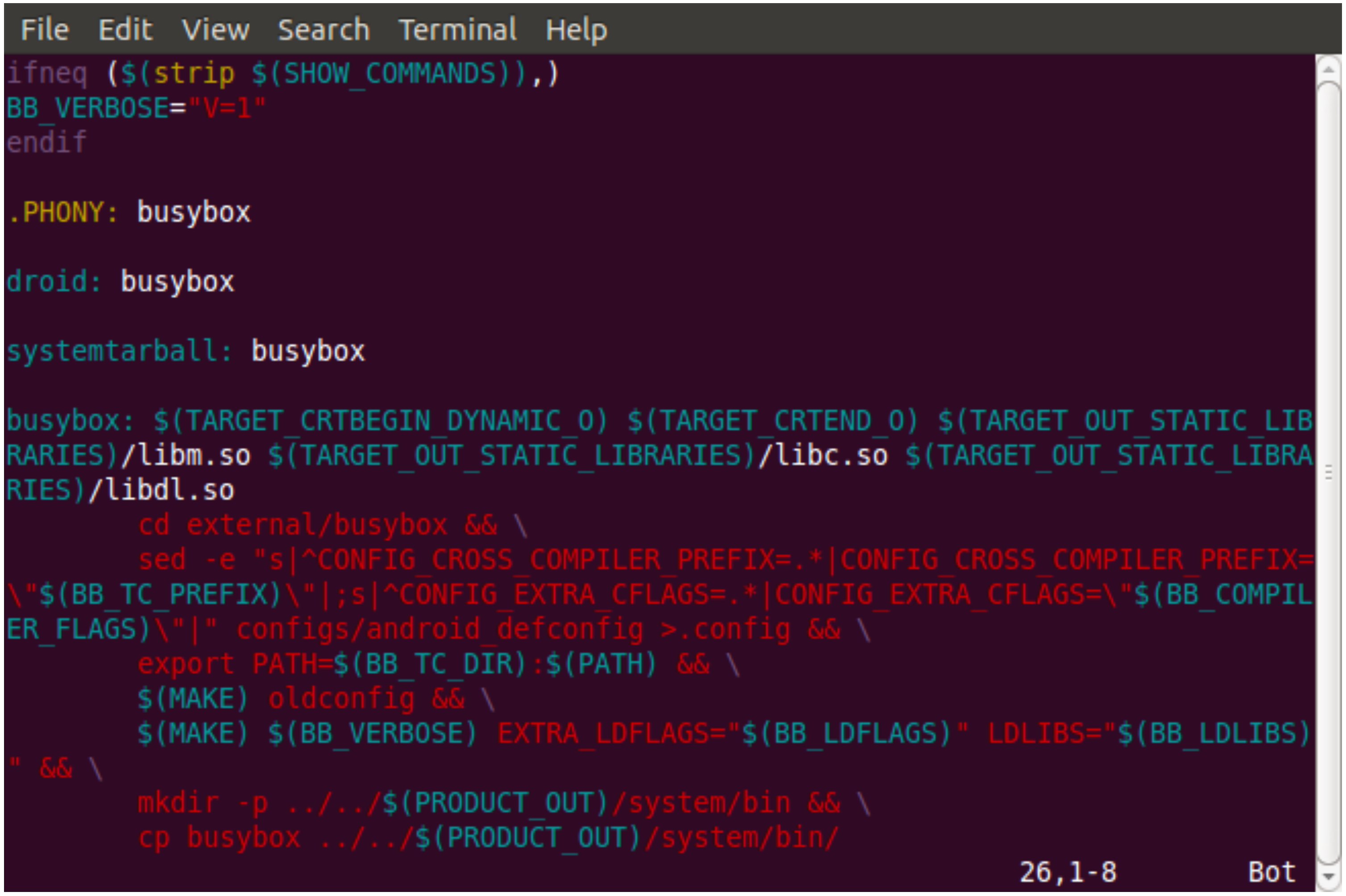


```
include $(CLEAR_VARS)

BB_TC_DIR := $(realpath $(shell dirname $(TARGET_TOOLS_PREFIX)))
BB_TC_PREFIX := $(shell basename $(TARGET_TOOLS_PREFIX))
BB_LDFLAGS := -Xlinker -z -Xlinker muldefs -nostdlib -Bdynamic -Xlinker -T../../
$(BUILD_SYSTEM)/armelf.x -Xlinker -dynamic-linker -Xlinker /system/bin/linker -X
linker -z -Xlinker nocopyreloc -Xlinker --no-undefined ../../$(TARGET_CRTBEGIN_D
YNAMIC_O) ../../$(TARGET_CRTEND_O) -L../../$(TARGET_OUT_STATIC_LIBRARIES)
# FIXME remove -fno-strict-aliasing once all aliasing violations are fixed
BB_COMPILER_FLAGS := $(subst -I ,-I../../,$(subst -include ,-include ../../,$(TA
RGET_GLOBAL_CFLAGS))) -I../../bionic/libc/include -I../../bionic/libc/kernel/com
mon -I../../bionic/libc/arch-arm/include -I../../bionic/libc/kernel/arch-arm -I.
./../bionic/libm/include -fno-stack-protector -Wno-error=format-security -fno-st
rict-aliasing
BB_LDLIBS := dl m c gcc
ifneq ($(strip $(SHOW_COMMANDS)),)
BB_VERBOSE="V=1"
endif

.PHONY: busybox

droid: busybox

"Android.mk" 26L, 1651C                                       1,1          Top
```

botao_sun
Linaro Android Platform Team

Build Busybox out of platform

Write "`Android.mk`" file for Busybox - Part II:



```
ifneq ($(strip $(SHOW_COMMANDS)),)
BB_VERBOSE="V=1"
endif

.PHONY: busybox

droid: busybox

systemtarball: busybox

busybox: $(TARGET_CRTBEGIN_DYNAMIC_O) $(TARGET_CRTEND_O) $(TARGET_OUT_STATIC_LIB
RARIES)/libm.so $(TARGET_OUT_STATIC_LIBRARIES)/libc.so $(TARGET_OUT_STATIC_LIBRA
RIES)/libdl.so
        cd external/busybox && \
        sed -e "s|^CONFIG_CROSS_COMPILER_PREFIX=.*|CONFIG_CROSS_COMPILER_PREFIX=
\"$(BB_TC_PREFIX)\"|;s|^CONFIG_EXTRA_CFLAGS=.*|CONFIG_EXTRA_CFLAGS=\"$(BB_COMPIL
ER_FLAGS)\"|" configs/android_defconfig >.config && \
        export PATH=$(BB_TC_DIR):$(PATH) && \
        $(MAKE) oldconfig && \
        $(MAKE) $(BB_VERBOSE) EXTRA_LDFLAGS="$(BB_LDFLAGS)" LDLIBS="$(BB_LDLIBS)
" && \
        mkdir -p ../../$(PRODUCT_OUT)/system/bin && \
        cp busybox ../../$(PRODUCT_OUT)/system/bin/
                                                            26,1-8        Bot
```

Android.mk

`Android.mk`: Android.mk files are merged into one giant Makefile during the Android build process.
A typical `Android.mk` is its own build system, e.g.

```
LOCAL_SRC_FILES := file1.c file2.c
LOCAL_MODULE := libmylibrary
include $(BUILD_STATIC_LIBRARY)
```

If something already has a build system and you don't want to reinvent it, you have to "abuse" the fact that it's a Makefile at heart:

Android.mk

```
droid: busybox

systemtarball: busybox
```

"`droid`" and "`systemtarball`" are the targets we're building when building the OS - so make sure they depend on the target we're introducing

```
busybox: $(TARGET_CRTBEGIN_DYNAMIC_O) $(TARGET_CRTEND_O)
$(TARGET_OUT_STATIC_LIBRARIES)/libm.so $(TARGET_OUT_STATIC_LIBRARIES)/libc.so
$(TARGET_OUT_STATIC_LIBRARIES)/libdl.so
```

We need bionic (libc) and friends to be built first - and Android.mk has no way of knowing this automatically - so we have to list the deps manually.

```
    cd external/busybox && \
    sed -e "s¦^CONFIG_CROSS_COMPILER_PREFIX=.
*¦CONFIG_CROSS_COMPILER_PREFIX=\"$(shell basename $(TARGET_TOOLS_PREFIX))\"
configs/android_defconfig >.config && \
```

Inject parameters from Android's build system into busybox's build system...

```
    $(MAKE) oldconfig && \
    $(MAKE) && \
    mkdir -p ../../$(PRODUCT_OUT)/system/bin && \
    cp busybox ../../$(PRODUCT_OUT)/system/bin/
```

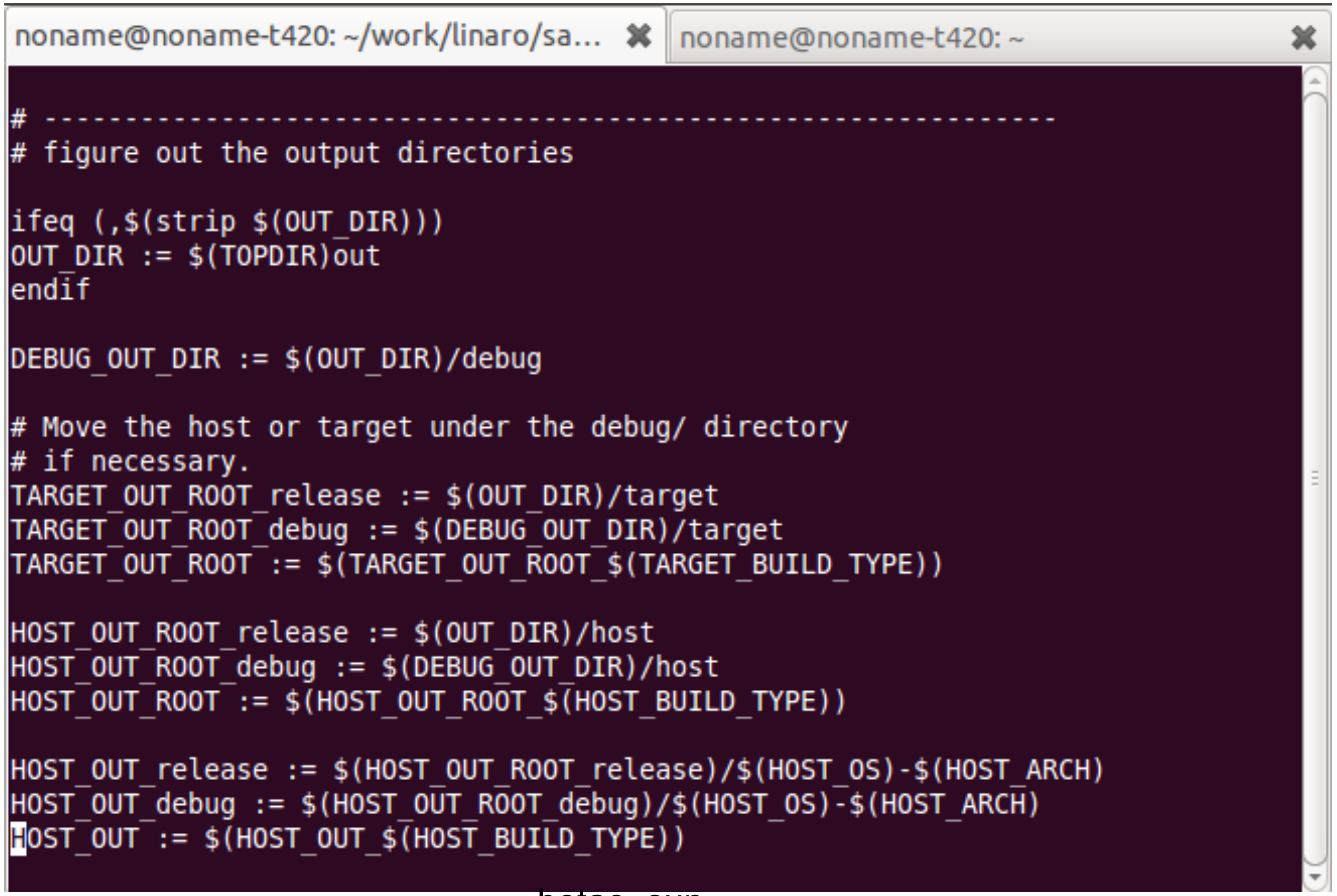
And call into busybox's build system to do its job

botao_sun
Linaro Android Platform Team

Build Busybox in platform

Output directories definition:

- /Android_root/build/core/envsetup.mk



noname@noname-t420: ~/work/linaro/sa...   noname@noname-t420: ~

```
# ------------------------------------------------------------------
# figure out the output directories

ifeq (,$(strip $(OUT_DIR)))
OUT_DIR := $(TOPDIR)out
endif

DEBUG_OUT_DIR := $(OUT_DIR)/debug

# Move the host or target under the debug/ directory
# if necessary.
TARGET_OUT_ROOT_release := $(OUT_DIR)/target
TARGET_OUT_ROOT_debug := $(DEBUG_OUT_DIR)/target
TARGET_OUT_ROOT := $(TARGET_OUT_ROOT_$(TARGET_BUILD_TYPE))

HOST_OUT_ROOT_release := $(OUT_DIR)/host
HOST_OUT_ROOT_debug := $(DEBUG_OUT_DIR)/host
HOST_OUT_ROOT := $(HOST_OUT_ROOT_$(HOST_BUILD_TYPE))

HOST_OUT_release := $(HOST_OUT_ROOT_release)/$(HOST_OS)-$(HOST_ARCH)
HOST_OUT_debug := $(HOST_OUT_ROOT_debug)/$(HOST_OS)-$(HOST_ARCH)
HOST_OUT := $(HOST_OUT_$(HOST_BUILD_TYPE))
```

Build Busybox in platform

Use "`mm`" to rebuild just Busybox:

- After you've finished your changes in, cd back to your Android root directory and run:

  -
    ```
    $ . build/envsetup.sh
    ```

  -
    ```
    $ HOST_CC=gcc-4.5 HOST_CXX=g++-4.5
    HOST_CPP=cpp-4.5
    TARGET_PRODUCT=origen
    TARGET_SIMULATOR=false
    TARGET_TOOLS_PREFIX=/your_toolchai
    n_path/bin/arm-linux-androideabi-
    mm
    ```

Build Busybox in platform

If everything works, you will find your "`busybox`" binary file in this directory:

- `/your_Android_root/out/target/product/origen/system/bin`

Transfer it to the directory "`/system/bin`" on your board then run this command to test it:

- `$ busybox top`

Build Busybox in platform

Make a final full build:

- After add the item likes this in your product manifest;
  - `<project path="external/busybox" name="platform/external/busybox" revision="linaro_android_4.0.3"/>`
- You can launch a full platform build now (Page `8`);
- If the build can be done successfully, after flash it to your product, you can launch `busybox` in the serial or `ADB` shell.

Go through the dots

- Prepare the hardware & software environment
- Launch a full platform build without busybox
- Build busybox out of platform with Linaro tool chain
    - Build error may happen here
- Build busybox in platform with Linaro tool chain and "`Android.mk`"
    - Build error may happen here
- Launch a full platform build with busybox

Thank you!

Feel free to send your questions and comments to
Botao Sun (botao.sun@linaro.org)
or
botao_sun in `#linaro-android` on irc.freenode.net