



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«Київський політехнічний інститут ім. ІГОРЯ СІКОРСЬКОГО»  
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №3**  
З дисципліни “ДПКС”

Виконав:  
студент групи ІВ-91мн  
Бояршин Ігор

Київ 2020 р.

# Завдання

У цій лабораторній роботі ми дослідимо, як поетапно відбувається компіляція повноцінної linux-системи, з яких частин вона складається, та які інструменти для цього необхідні. В кінці ми зробимо симуляцію роботи зібранної системи за допомогою віртуалізатора qemu.

В якості хостової операційної системи буде використана не Ubuntu, а Arch Linux.

## Збірка

### Встановлюємо ccache

:> sudo pacman -S ccache

```
[1:dev] 2:web 3:tg 4 9:bg | Full | igorek@apol
igorek > -> Stuff > linux > sudo pacman -S ccache
[sudo] password for igorek:
resolving dependencies...
looking for conflicting packages...

Packages (1) ccache-3.7.9-1

Total Download Size: 0.11 MiB
Total Installed Size: 0.27 MiB

:: Proceed with installation? [Y/n]
:: Retrieving packages...
ccache-3.7.9-1-x86_64
(1/1) checking keys in keyring
(1/1) checking package integrity
(1/1) loading package files
(1/1) checking for file conflicts
(1/1) checking available disk space
:: Processing package changes...
(1/1) installing ccache
:: Running post-transaction hooks...
(1/1) Arming ConditionNeedsUpdate...
```

Set the cache size maximum value to 5G and print the status

```
igorek > -> Stuff > linux > ccache -M 5G
Set cache size limit to 5.0 GB
igorek > -> Stuff > linux > ccache -s
cache directory          /home/igorek/.ccache
primary config           /home/igorek/.ccache/ccache.conf
secondary config (readonly) /etc/ccache.conf
cache hit (direct)       0
cache hit (preprocessed) 0
cache miss                0
cache hit rate            0.00 %
cleanups performed        0
files in cache             0
cache size                 0.0 kB
max cache size            5.0 GB
```

### Встановлення необхідного додаткового програмного забезпечення

Як видно зі скріншота, необхідне програмне забезпечення вже було встановлено на системі.

```

igorek ~ > Stuff > linux > sudo pacman -S flex bison bc
[sudo] password for igorek:
warning: flex-2.6.4-3 is up to date -- reinstalling
warning: bison-3.5.3-1 is up to date -- reinstalling
warning: bc-1.07.1-3 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...

Packages (3) bc-1.07.1-3 bison-3.5.3-1 flex-2.6.4-3

Total Installed Size: 3.21 MiB
Net Upgrade Size: 0.00 MiB

:: Proceed with installation? [Y/n]

```

## Завантаження toolchains

```

igorek ~ > Stuff > linux > l
total 611M
drwxr-xr-x 2 igorek users 4.0K Apr  6 20:46 .
drwxr-xr-x 8 igorek users 4.0K Apr  5 12:48 ..
-rw-r--r-- 1 igorek users 367M Apr  5 13:19 gcc-arm-8.3-2019.03-x86_64-arm-eabi.tar.xz
-rw-r--r-- 1 igorek users 245M Apr  5 13:15 gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf.tar.xz

```

Використовуючи команду

:> tar xJvf <NAME> -C /opt

Послідовно розпаковуємо обидва завантажених архіви до папки /opt

```

igorek ~ > Stuff > linux > l /opt | grep "gcc*"
drwxr-xr-x 8 igorek igorek 4.0K Mar 25 2019 gcc-arm-8.3-2019.03-x86_64-arm-eabi
drwxr-xr-x 9 igorek igorek 4.0K Mar 25 2019 gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf

```

## U-Boot

Клонуємо репозиторій:

:> git clone https://gitlab.denx.de/u-boot/u-boot.git

За допомогою наступної команди виведемо теги релізних версій проекту:

:> git tag | grep -v rc | tail -5

Виберемо необхідний (останній тег) та перейдемо на його гілку:

:> git checout v2020.01

```

igorek > > Stuff > linux > u-boot > * ` master git tag | grep -v rc | tail -5
v2019.01
v2019.04
v2019.07
v2019.10
v2020.01
igorek > > Stuff > linux > u-boot > * ` master git checkout v2020.01
Note: switching to 'v2020.01'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable advice.detachedHead to false

```
HEAD is now at 0b0c6af387 Prepare v2020.01
```

```
igorek > > Stuff > linux > u-boot > * ` 0b0c6af3 v2020.01
```

... і застосуємо готову серію патчів до вихідного коду:

```
:> curl https://patchwork.ozlabs.org/series/130450/mbox/ | git am
```

```

igorek > > Stuff > linux > u-boot > * ` 0b0c6af3 v2020.01 curl https://patchwork.ozlabs.org/series/130450/mbox/ | git am
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload Total   Spent    Left  Speed
100 16375  100 16375    0     0  8480      0  0:00:01  0:00:01  --:--:--  8475
Applying: Revert "configs: Remove am335x_boneblack_defconfig"
Applying: configs: am335x_boneblack: Sync with am335x_evm
igorek > > Stuff > linux > u-boot > * ` 1a64c0a365
```

Наступним кроком налаштовуємо середовище компіляції та виконуємо збірку:

```

igorek > > Stuff > linux > u-boot > * ` 1a64c0a365 export PATH=/opt/gcc-arm-8.3-2019.03-x86_64-arm-eabi/bin:$PATH
igorek > > Stuff > linux > u-boot > * ` 1a64c0a365 export CROSS_COMPILE='ccache arm-eabi-'
igorek > > Stuff > linux > u-boot > * ` 1a64c0a365 export ARCH=arm
igorek > > Stuff > linux > u-boot > * ` 1a64c0a365 make am335x_boneblack_defconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
YACC scripts/kconfig/zconf.tab.c
LEX scripts/kconfig/zconf.lex.c
HOSTCC scripts/kconfig/zconf.tab.o
HOSTLD scripts/kconfig/conf
#
# configuration written to .config
#
igorek > > Stuff > linux > u-boot > * ` 1a64c0a365 man make
igorek > > Stuff > linux > u-boot > * ` 1a64c0a365 make -j8
```

Визначаємо максимальну кількість jobs як 8.

## Kernel

Спочатку клонуємо вихідний код:

```
:> git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
```

Вибираємо потрібну версію коду:

```
:> git checkout linux-4.19.y
```

...та налаштовуємо середовище такими самими змінними оточення (тому їх можна не повторювати).

З усіх додаткових налаштувань, оскільки кінцевим завданням є лише емуляція, необхідним здалося лише одне:

```
1 # --- Networking ---
2 CONFIG_BRIDGE=y
```

Кладемо його у файл fragments/bbb.cfg.

Створюємо кінцевий файл конфігурації разом з нашими правками:

```
igorek ~ > Stuff > linux > linux-stable > # v4.19.y * v4.19.114 > ./scripts/kconfig/merge_config.sh arch/arm/configs/multi_v7_defconfig fragments/bbb.cfg
Using arch/arm/configs/multi_v7_defconfig as base
Merging fragments/bbb.cfg
HOSTCC scripts/basic/fixedep
HOSTCC scripts/kconfig/conf.o
YACC scripts/kconfig/zconf.tab.c
LEX scripts/kconfig/zconf.lex.c
HOSTCC scripts/kconfig/zconf.tab.o
HOSTLD scripts/kconfig/conf
scripts/kconfig/conf --alldefconfig Kconfig
#
# configuration written to .config
#
```

Починаємо компіляцію ядра та необхідних модулів:

```
:> make -j8 zImage modules am335x-boneblack.dtb
```

## BusyBox

Скачуємо вихідний код:

```
igorek ~ > Stuff > linux > git clone git://git.busybox.net/busybox
Cloning into 'busybox'...
remote: Enumerating objects: 110424, done.
remote: Counting objects: 100% (110424/110424), done.
remote: Compressing objects: 100% (28074/28074), done.
remote: Total 110424 (delta 88325), reused 102158 (delta 81649)
Receiving objects: 100% (110424/110424), 27.51 MiB | 333.00 KiB/s, done.
Resolving deltas: 100% (88325/88325), done.
igorek ~ > Stuff > linux > d busybox
head: error reading 'busybox': Is a directory
igorek ~ > Stuff > linux > cd busybox
igorek ~ > Stuff > linux > busybox > # master > git branch -a | grep stable | sort -V | tail -1
remotes/origin/1_31_stable
igorek ~ > Stuff > linux > busybox > # master > git checkout 1_31_stable
Branch '1_31_stable' set up to track remote branch '1_31_stable' from 'origin'.
Switched to a new branch '1_31_stable'
```

Оновлюємо дві змінні оточення та генеруємо базову конфігурацію:

```
:> export PATH=/opt/gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf/bin:$PATH
:> export CROSS_COMPILE="ccache arm-linux-gnueabihf-"
:> make defconfig
```

... і запускаємо на компіляцію:

```
:> make -j8
```

Після цього виконуємо інсталяцію:

```
:> make install
```

Після цього створюємо директорії, що необхідні для наповнення rootfs відсутніми віzlами та init-файлами:

```
:> mkdir -p _install/{boot,dev,etcinit.d,lib,proc,root,syskerneldebug,tmp}
```

Створюємо init-скрипт \_install/etc/init.d/rcS і додаємо цому право на запуск:

```
1 #!/bin/sh
2
3 mount -t sysfs none /sys
4 mount -t proc none /proc
5 mount -t debugfs none /sys/kernel/debug
6
7 echo /sbin/mdev > /proc/sys/kernel/hotplug
8 mdev -s
```

В кореневій директорії робимо посилання на ініт-систему:

```
:> ln -s bin/busybox _install/init
```

## Populate /boot

Далі необхідно заповнити /boot директорію файлами ядра:

```
igorek ~ > Stuff > linux > busybox > p ↵ 1_31_stable ↵ 1_31_1 ↵ cd _install/boot
igorek ~ > linux > busybox > _install > boot > p ↵ 1_31_stable ↵ 1_31_1 ↵ cp ../../linux-stable/arch/arm/boot/zImage .
sending incremental file list
zImage

sent 8.73M bytes received 35 bytes 17.46M bytes/sec
total size is 8.73M speedup is 1.00
igorek ~ > linux > busybox > _install > boot > p ↵ 1_31_stable ↵ 1_31_1 ↵ cp ../../linux-stable/arch/arm/boot/dts/am335x-boneblack.dtb .
am335x-boneblack.dtb

sent 34.91K bytes received 35 bytes 69.88K bytes/sec
total size is 34.78K speedup is 1.00
igorek ~ > linux > busybox > _install > boot > p ↵ 1_31_stable ↵ 1_31_1 ↵ cp ../../linux-stable/System.map .
System.map

sent 4.54M bytes received 35 bytes 9.08M bytes/sec
total size is 4.54M speedup is 1.00
igorek ~ > linux > busybox > _install > boot > p ↵ 1_31_stable ↵ 1_31_1 ↵ cp ../../linux-stable/.config .config
.sendin config

sent 193.73K bytes received 35 bytes 387.54K bytes/sec
total size is 193.59K speedup is 1.00
igorek ~ > linux > busybox > _install > boot > p ↵ 1_31_stable ↵ 1_31_1 ↵
```

## Populate /lib

Налаштовуємо середовище:

```
igorek ~ > Stuff > linux > linux-stable # $ linux-4.19.y v4.19.114 # export INSTALL_MOD_PATH=~/Stuff/linux/busybox/_install
igorek ~ > Stuff > linux > linux-stable # $ linux-4.19.y v4.19.114 # export ARCH=arm
igorek ~ > Stuff > linux > linux-stable # $ linux-4.19.y v4.19.114 # make modules_install
```

Визначаємо необхідні бібліотеки:

```
igorek ~ > Stuff > linux > busybox # $ 1_31_stable v1_31_1 # ccache arm-linux-gnueabihf-readelf -d _install/bin/busybox | grep NEEDED
0x00000001 (NEEDED) Shared library: [libm.so.6]
0x00000001 (NEEDED) Shared library: [libresolv.so.2]
0x00000001 (NEEDED) Shared library: [libc.so.6]
```

Копіюємо бібліотеки:

```
igorek ~ > Stuff > linux > busybox # $ 1_31_stable v1_31_1 # cd _install/lib
igorek ~ > linux > busybox > _install > lib # $ 1_31_stable v1_31_1 # libc_dir=$(${CROSS_COMPILE}gcc -print-sysroot)/lib
zsh: command not found: ccache arm-linux-gnueabihf-gcc
igorek ~ > linux > busybox > _install > lib # $ 1_31_stable v1_31_1 # libc_dir=$(ccache arm-linux-gnueabihf-gcc -print-sysroot)/lib
igorek ~ > linux > busybox > _install > lib # $ 1_31_stable v1_31_1 # cp $libc_dir/*.so* .
sending incremental file list
ld-2.28.so
ld-linux-armhf.so.3 -> ld-2.28.so
libBrokenLocale-2.28.so
libBrokenLocale.so.1 -> libBrokenLocale-2.28.so
libSegFault.so
libanl-2.28.so
libanl.so.1 -> libanl-2.28.so
libc-2.28.so
libc.so.6 -> libc-2.28.so
libcrypt-2.28.so
libcrypt.so.1 -> libcrypt-2.28.so
libdl-2.28.so
libdl.so.2 -> libdl-2.28.so
libm-2.28.so
libm.so.6 -> libm-2.28.so
libmemusage.so
libnsl-2.28.so
libnsl.so.1 -> libnsl-2.28.so
libnss_compat-2.28.so
libnss_compat.so.2 -> libnss_compat-2.28.so
libnss_db-2.28.so
libnss_db.so.2 -> libnss_db-2.28.so
libnss_dns-2.28.so
libnss_dns.so.2 -> libnss_dns-2.28.so
libnss_files-2.28.so
libnss_files.so.2 -> libnss_files-2.28.so
libnss_hesiod-2.28.so
libnss_hesiod.so.2 -> libnss_hesiod-2.28.so
libpcprofile.so
libpthread-2.28.so
libpthread.so.0 -> libpthread-2.28.so
libresolv-2.28.so
libresolv.so.2 -> libresolv-2.28.so
librt-2.28.so
librt.so.1 -> librt-2.28.so
libthread_db-1.0.so
libthread_db.so.1 -> libthread_db-1.0.so
libutil-2.28.so
libutil.so.1 -> libutil-2.28.so

sent 31.01M bytes received 469 bytes 62.02M bytes/sec
total size is 31.00M speedup is 1.00
```

## Populate /etc

Підготуємо конфігурацію mdev:

```
:> echo '$MODALIAS=.* root:root 660 @modprobe "$MODALIAS"' > _install/etc/mdev.conf
```

Підготуємо необхідні файли:

```
igorek ~ > Stuff > linux > busybox # $ 1_31_stable v1_31_1 # echo 'root:x:0:' > _install/etc/group
igorek ~ > Stuff > linux > busybox # $ 1_31_stable v1_31_1 # echo 'root:x:0:root:/root:/bin/sh' > _install/etc/passwd
igorek ~ > Stuff > linux > busybox # $ 1_31_stable v1_31_1 # echo 'root::10933:0:99999:7:::' > _install/etc/shadow
igorek ~ > Stuff > linux > busybox # $ 1_31_stable v1_31_1 # echo "nameserver 8.8.8.8" > _install/etc/resolv.conf
```

## QEMU

QEMU уже встановлена на хостовій операційній системі.

Створюємо спрощений архів рутової файлової системи та архівуємо його:

```
igorek [~] > Stuff > linux > busybox > _install > V 1_31_stable > 1_31_1 > find . | cpio -o -H newc | gzip > ../rootfs.cpio.gz  
119382 blocks
```

Запускаємо на виконання:

```
:> qemu-system-arm -kernel _install/boot/zImage -initrd rootfs.cpio.gz -machine virt -nographic -m 512 -append "root=/dev/ram0 rw console=ttyAMA0,115200 mem=512M"
```

Результат виконання базових команд у новоствореній системі:

```
/ # uname -a  
Linux (none) 4.19.114 #1 SMP Mon Apr 6 22:39:36 EEST 2020 armv7l GNU/Linux  
/ # ls -l  
total 0  
drwxr-xr-x  2 1000   985      0 Apr  6 19:58 bin  
drwxr-xr-x  2 1000   985      0 Apr  6 20:09 boot  
drwxr-xr-x  3 1000   985      0 Apr  6 21:42 dev  
drwxr-xr-x  3 1000   985      0 Apr  6 20:38 etc  
lrwxrwxrwx  1 1000   985      11 Apr  6 21:40 init -> bin/busybox  
drwxr-xr-x  3 1000   985      0 Apr  6 21:32 lib  
lrwxrwxrwx  1 1000   985      11 Apr  6 19:58 linuxrc -> bin/busybox  
dr-xr-xr-x  47 root    root     0 Jan  1 1970 proc  
drwxr-xr-x  2 1000   985      0 Apr  6 20:00 root  
drwxr-xr-x  2 1000   985      0 Apr  6 19:58 sbin  
dr-xr-xr-x  12 root    root     0 Apr  6 21:42 sys  
drwxr-xr-x  2 1000   985      0 Apr  6 20:00 tmp  
drwxr-xr-x  4 1000   985      0 Apr  6 19:58 usr  
/ # dmesg | grep init  
[ 0.000000] random: get_random_bytes called from start_kernel+0x9c/0x47c with crng_init=0  
[ 0.000000] Memory: 406440K/524288K available (12288K kernel code, 1601K rwdtata, 4700K rodata, 2048K init, 392K bss, 52312K reserved, 65536K cma-reserved, 0K highmem)  
[ 0.000000]   .init : 0x(ptval) - 0x(ptval)  (2048 kB)  
[ 0.077348] devtmpfs: initialized  
[ 0.104467] pinctrl core: initialized pinctrl subsystem  
[ 0.212256] SCSI subsystem initialized  
[ 0.334971] Trying to unpack rootfs image as initramfs...  
[ 2.154278] Freeing initrd memory: 25100K  
[ 2.283906] SuperH (H)SCI(F) driver initialized  
[ 2.285072] msm_serial: driver initialized  
[ 2.285465] STMicroelectronics ASC driver initialized  
[ 2.286662] STM32 USART driver initialized  
[ 2.525708] Run /init as init process  
/ # busybox --help | head -15  
BusyBox v1.31.1 (2020-04-06 22:55:05 EEST) multi-call binary.  
BusyBox is copyrighted by many authors between 1998-2015.  
Licensed under GPLv2. See source distribution for detailed  
copyright notices.  
  
Usage: busybox [function [arguments]...]  
  or: busybox --list[-full]  
  or: busybox --show SCRIPT  
  or: busybox --install [-s] [DIR]  
  or: function [arguments]...  
  
  BusyBox is a multi-call binary that combines many common Unix  
  utilities into a single executable. Most people will create a  
  link to busybox for each function they wish to use and BusyBox  
  will act like whatever it was invoked as.
```