Санкт-Петербургский Государственный Университет

## Курсовая работа

# Изучение и тестирование CLDC HI JVM

Выполнила: Сарманова С.Г.
Студент 361 группы кафедры
Системного программирования
Математико-механического
факультета СПбГУ
Научный руководитель: Полозов В.С.
Старший преподаватель кафедры
Системного программирования
Математико-механического
факультета СПбГУ

г. Санкт-Петербург, 2013г.

# Содержание

# Введение

Представим ситуацию: разрабатывается какое-то приложения на языке Java. Спустя некоторое время приложению понадобилась мобильная версия, которая будет работать на всех мобильных платформах с учетом архитектуры каждого устройства. Этого можно добиться, написав множество мобильных версий с небольшими изменениями в каждой, так как необходимо подстраиваться под платформы и архитектуры соответствующих мобильных устройств, причем изменения могут быть небольшие.

Как одним из выходов в данной ситуации может быть использование CLDC HI JVM на устройствах, где нет Java.

*Connected Limited Device Configuration (CLDC) HotSpot™ Implementation Virtual Machine* — высокопроизводительная виртуальная Java машина для устройств, ограниченных в ресурсах. Это одна из виртуальных машин для «малых» устройств, позволяющая запускать J2ME-приложения на устройствах с ограниченным объемом памяти и вычислительной мощностью, например, на мобильных телефонах, КПК, платежных терминалах. В ней улучшены алгоритмы оптимизации выполняемого кода в целом и часто выполняемых кусков кода в частности. Это проект с открытым исходным кодом компании Sun Microsystems, также известный как *PhoneME Feature*.

## Постановка цели, задачи

CLDC HI JVM позволяет J2ME-приложению работать на всех устройствах. Если использовать CLDC HI JVM для запуска приложения, где не поддерживается Java платформа, то это значительно сократит время разработки приложения, т.к. не нужно писать множество мобильных версий приложения, и позволит перенести приложение на устройство без Java платформы. Разработчику необходимо будет только собрать CLDC HI JVM для конкретной целевой платформы, учитывая её особенности (процессор, ядро и т.п.).

*Гипотеза:* использование CLDC HI JVM позволит запускать J2ME-приложения на устройствах, не имеющих Java платформу.

*Целью* данной работы является изучение CLDC HI JVM, создание методики по её сборки из исходного кода для разработчиков мобильных приложений и тестирование её производительности.

*Основные задачи:*

- Изучение CLDC HI JVM
- Создание методики сборки CLDC HI JVM
- Описание ограничений применимости
- Тестирование производительности

# Описание предметной области

CLDC HI JVM — оптимизированная виртуальная машина, предназначенная для устройств с ограниченными ресурсами — частотой процессора 50МГц, оперативной памятью более 600 Кбайт и доступным дисковым пространством более 1,5 Мбайт. CLDC HI обеспечивает более быстрое исполнение байт-кода и более эффективное использование ресурсов по сравнению с другими доступными JVM, такими как Squawk, KVM, Maxine, CVM и пр.

Ключевые моменты CLDC HI JVM :

- объем памяти, занимаемый виртуальной машины, и CLDC библиотеки сведены к минимуму;
- минимизация потребления энергии аккумулятора;
- переносимость на широкий круг устройств с разными возможностями.

Устройства, на которых может работать J2ME-приложение, определяются поддерживаемой *конфигурацией* и *профилем* платформы.

Конфигурация описывает только низкоуровневую часть платформы: возможности языка Java, его виртуальной машины, и базовые классы. Конфигурация призвана объединять все устройства со сходными вычислительными возможностями, независимо от их назначения.

В настоящее время существует две конфигурации J2ME(см. рис.1):

1. CDC - *Connected Device Configuration* — используется с виртуальной машиной на устройствах с 32-разрядной архитектурой и требует более 2 Мб памяти. Она предусматривает больше возможностей для приложений по сравнению с CLDC, но и более жесткие требования к аппаратуре.
2. CLDC - *Connected Limited Device Configuration* — используется для 16- и 32-разрядных устройств с ограниченным объемом памяти.
   Приложения J2ME, созданные с учетом конфигурации CLDC, ориентированы на устройства со следующими характеристиками:
   - Минимальный объём ПЗУ: 128 килобайт для CLDC 1.0, 160 килобайт для CLDC 1.1
   - Минимальный объём ОЗУ: 32 килобайта

- Процессор: 16- или 32-битный
- Низкое энергопотребление
- Подключение к сети, в том числе беспроводной, с нестабильным соединением и ограниченной скоростью передачи данных

Базовые классы, которые включает в себя конфигурация CLDC:

- java.io
- java.lang
- java.lang.ref
- java.util
- javax.microedition

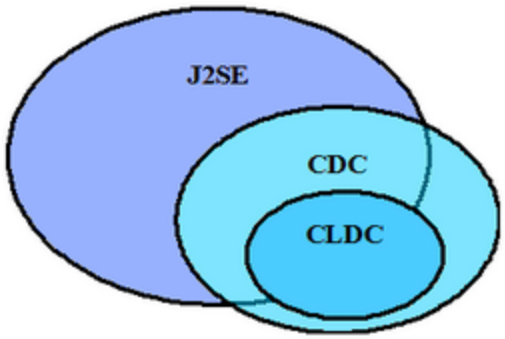



рис.1 Соотношение функциональности конфигураций

Более высокоуровневой частью платформы является профиль. Предполагается, что профиль будет задаваться для каждого крупного класса устройств (мобильные телефоны, игровые автоматы, бытовые приборы). Т.е. профиль определяет тип устройств, поддерживаемых приложением. Профиль дополняет конфигурацию специфическими классами, определяющими область применения устройств.

В J2ME определено два профиля, построенных на основе CLDC: *KJava* и *Mobile Information Device Profile (MIDP)*. Для CDC доступен шаблонный профиль, на котором можно строить свои собственные, Foundation Profile.

Платформа, состоящая из CLDC и MIDP, достаточно распространена на рынке мобильных телефонов. Стоит отметить, что CLDC HI JVM также основывается на конфигурации CLDC 1.1(JSR 118) и профиле MIDP 2.1 (JSR 139).

Java Community Process (JCP) — процесс внесения изменений в спецификации платформ языка Java. Основой JCP является Java Specification Request (JSR) - формальные документы, описывающие спецификации и технологии, которые предлагается добавить к Java платформе. CLDC HI JVM обеспечивает полную или частичную реализации нескольких JSRs. Подробнее с JSR-спецификациями, реализованными в CLDC HI JVM, можно ознакомиться в Приложении 1.

# Практическая часть

Основными целями данной работы являются:

1. Создание методологии сборки CLDC HI JVM
2. Описание ограничений применимости
3. Запуск на Embedded system
4. Тестирование производительности

Рассмотрим каждый пункт в отдельности.

## 1. Методика сборки CLDC HI JVM

Как говорилось ранее, данная виртуальная машина необходима в случае, когда разработчику нужно перенести мобильную версию приложения на устройство, не поддерживающее Java платформу. Удобно, когда под рукой такого разработчика будет находиться методика с подробным описанием сборки виртуальной машины, начиная от исходного кода и заканчивая работающей JVM.

В рамках данной работы была создана методика, в которой подробно описана инструкция сборки CLDC HI JVM, а также указаны доработки исходного кода. Подробнее ознакомиться с ней можно в приложенных документах (см. Приложение 2).

## 2. Ограничения применимости

CLDC HI JVM является оптимизированной виртуальной машиной, которая обеспечивает более быстрое исполнение байт-кода и более эффективное использование ресурсов по сравнению с другими доступными виртуальными машинами, такими как Squawk, KVM, Maxine, CVM и пр.

На таблице 1 представлены минимальные и стандартные требования к процессору и памяти целевого устройства, необходимые для работы виртуальной машины.

| Item | Minimum | Typical |
|---|---|---|
| CPU Type | Mostly ARM | Mostly ARM |
| CPU Speed | 50 MHz | 50 to 200 MHz |
| RAM | 300 KB (including MIDP) | > 600 KB (including MIDP) |
| ROM/Flash | 1 MB | > 1.5 MB |

Таблица 1. Требования к целевому устройству

ARM процессоры представляют значительную долю рынка мобильных телефонов (см. Литература, [1]), поэтому CLDC HI JVM включает в себя множество оптимизаций для этой платформы.

Но CLDC HI JVM также поддерживает следующие целевые платформы:

- Linux on ARM, x86 ;
- JavaCall API on x86 ;
- Windows on x86 и др. (см. Литература, [2])

Стоит отметить, что кроме мобильных телефонов, данная виртуальная машина может использоваться устройствами, которые потенциально принадлежат CLDC и MIDP категории и удовлетворяют требованиям процессора и памяти, например, КПК, платежные терминалы и пр.

Следует также не забывать, что разработчики устройств могут ограничивать круг используемых приложений на их устройствах. Так, например, всем известный iPhone поддерживает архитектуру ARM процессора, но портировать на него свое приложение просто так нельзя, т.к. все приложения должны проходить через AppStore.

CLDC HI JVM можно применять не только в мобильных устройствах. За счет интеграции данной виртуальной машины в NetBeans IDE, CLDC HI JVM можно использовать для запуска и отладки мидлетов, о чем в свое время побеспокоилась компания Sun Microsystems.

## 3. Запуск на Embedded system

В качестве целевого устройства использовался одноплатный компьютер Raspberry Pi. Подробнее с характеристиками устройства можно ознакомиться в Табл. 2 и Изобр. 1.

| | |
|---|---|
| Разработчик | Raspberry Pi Foundation |
| Тип | Одноплатный компьютер |
| Операционная система | Raspbian |
| Оперативная память | 512 Мб |
| Разработчик CPU | Broadcom |
| CPU | BCM2835, ARM1176JZF-S (armv6k) |
| Тактовая частота CPU | 700 MHz |

Таблица 2. Характеристики Raspberry Pi

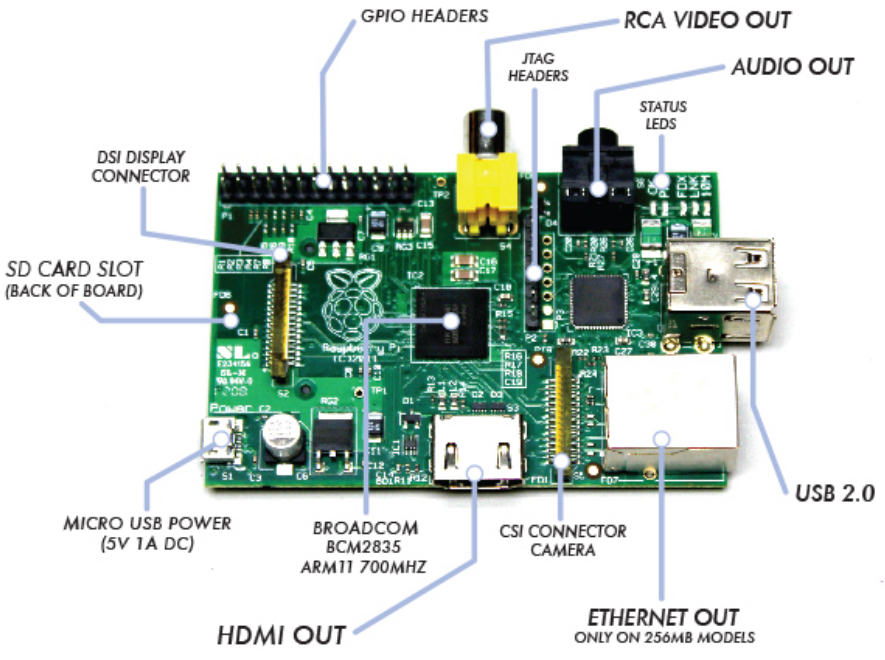



Рис.1 Raspberry Pi

На вход JVM подается jar-файл, который в свою очередь обрабатывает установщик

Midlet'ов и выдает индивидуальный номер для каждого приложения (подробнее см. Приложение 2). После чего запускается CLDC HI JVM с номером приложения и создается frameBuffer, в который выводится изображение экрана устройства. Работа проиллюстрирована на Рис.2 и Рис.3

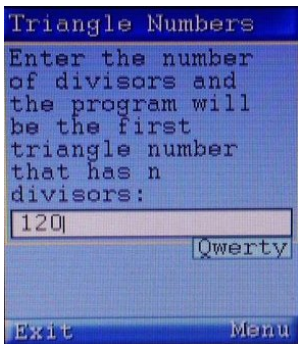



Рис. 2

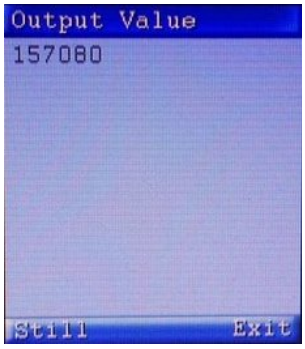



Рис. 3

## 4. Тестирование производительности

В качестве тестирования CLDC HI JVM было выбрано 2 теста, каждый из которых выдает количество проделанных итераций в секунду.

a. *Тест Kfl*. Создает реальную рабочую нагрузку для встраиваемой системы. Приложение является частью распределенной системы управления двигателем и адаптировано для тестирования на встраиваемых системах.
b. *Тест UDP/IP*. Это адаптированный крошечный TCP/IP стек для встраиваемых систем.

Для сравнительного анализа производительности было протестировано 3 устройства с различными JVM:

- Raspberry Pi/CLDC HI JVM
- Raspberry Pi/Hotspot
- Intel Core 2 Duo/Hotspot
- NIOS II/KVM

В Таблице 3 и на Диаграмме 4 можно увидеть результаты тестирования.

| [Iterations/s] | R-Pi/CLDC HI | R-Pi/ Hotspot | Intel Core 2 Duo/ Hotspot | NIOS II/KVM |
|---|---|---|---|---|
| Kfl | 272 000 | 30 000 | 7 476 000 | - |
| UDP/IP | 124 000 | 13 000 | 4 917 000 | - |
| Geom. mean | 182 000 | 20 000 | 6 063 000 | 78 000 |

Таблица 3. Тестирование производительности

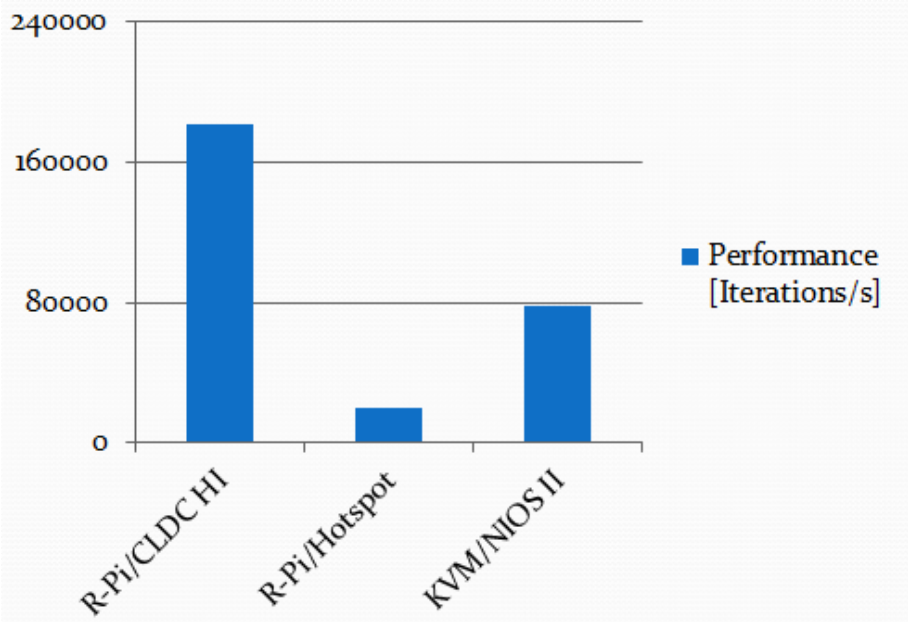



Диаграмма 4. Тестирование производительности

Как видно, CLDC HI показывает высокую производительность по сравнению с аналогами. А также дает неплохой результат по сравнению с более мощными процессорами.

# Результаты

В рамках данной работе была изучена виртуальная java-машина CLDC Hotspot Implementation, а также область её применения.

В ходе исследования был доработан исходный код из репозитория проекта PhoneME, в следствие чего удалось собрать и запустить данную JVM на устройстве, ограниченном в ресурсах, — Raspberry Pi на процессоре ARM.

Создана и описана методика процесса сборки виртуальной машины и запуска с её помощью приложений на устройствах.

Подробно описаны ограничения применимости данной виртуальной машины для различных встраиваемых систем с технической точки зрения, а также описаны области её применения.

Проведено тестирование на работоспособность и производительность собранной виртуальной машины. CLDC HI показала хорошие результаты по сравнению с более мощными JVM, предназначенными для больших устройств, а также превзошла по производительности существующие аналоги.

# Литература

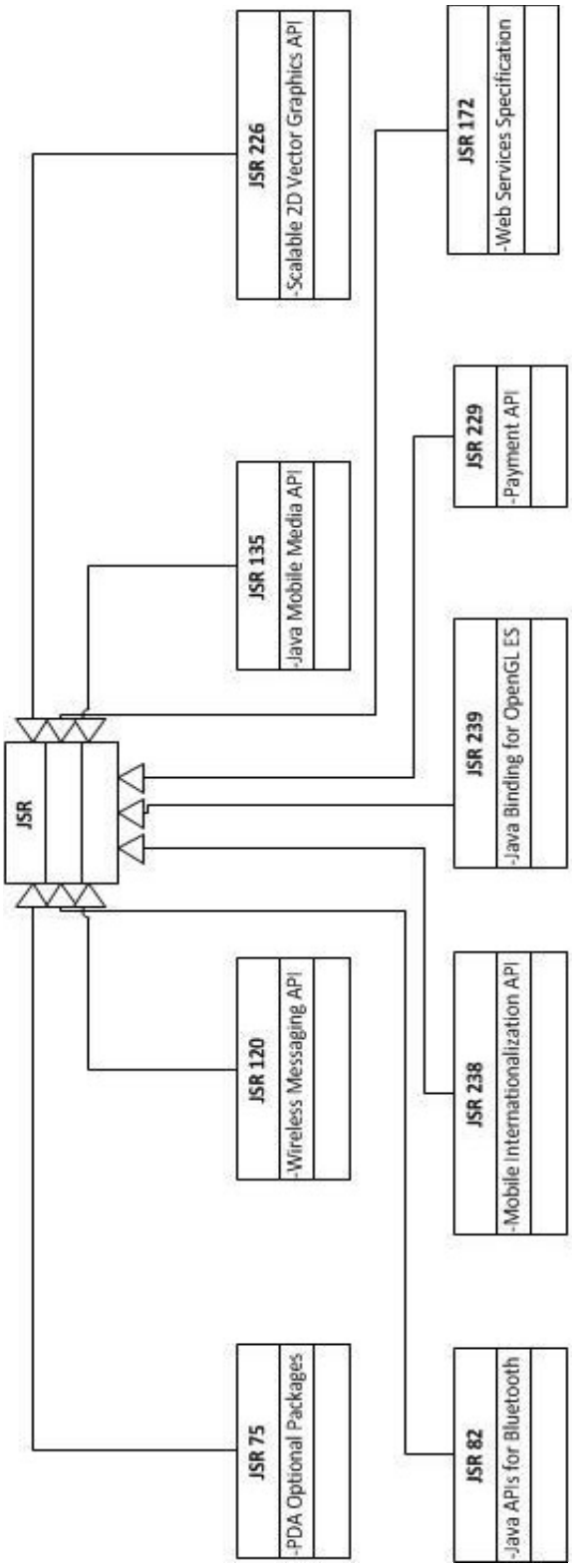

1. Список устройств, имеющих ARM процессор (http://dom-androida.ru/faq/item/23-spisok-ustroystv-armv7-i-armv6-armv5)
2. White Paper CLDC HotSpot™ Implementation Virtual Machine, Java™ 2 Platform, Micro Edition (J2ME™) Technology, February 2005
3. Oracle Corporation (www.oracle.com)
4. Репозиторий проекта PhoneME Feature (https://java.net/projects/phoneme)
5. Инструкция запуска CLDC HI JVM на целевом устройстве (http://meapplicationdevelopers.java.net/how_to_run.html)
6. JavaBenchEmbedded (http://www.jopdesign.com/perf.jsp)
7. Обзор возможных ошибок работы CLDC HI JVM (http://midpath4dingoo.googlecode.com/svn/tags/v0.1/components/core/src/com/sun/midp/installer/InvalidJadException.java)
8. CLDC HotSpot™ Implementation Porting Guide (http://elastos.org/elorg_files/FreeBooks/java/thesis/CLDC-Hotspot-Port.pdf)

# Глоссарий

**J2ME** – Java 2 Micro Edition – подмножество платформы Java для устройств, ограниченных в ресурсах;
**JCP** – Java Community Process – формальный процесс, который позволяет заинтересованным лицам участвовать в формировании будущих версий спецификаций платформ языка Java;
**JSR** – Java Specification Request – запрос на спецификацию Java;
**CLDC HI JVM** – Connected Limited Device Configuration Hotspot Implementation Java Virtual Machine – виртуальная Java машина;
**CDC** – Connected Device Configuration – стандарт конфигурации Java2ME для построения Java-приложений на КПК и различных цифровых устройствах;
**CLDC** – Connected Limited Device Configuration – стандарт конфигурации Java2ME для подключаемых к сети устройств с ограниченными вычислительными возможностями;
**MIDlet** – от англ. MID «Mobile Information Device» и англ. -let «нечто маленькое» – приложение для профиля MIDP платформы Java2ME;
**MIDP** – Mobile Information Device Profile – профиль для мобильного устройства с информационными функциями (платформа J2ME).

# Приложение 1

JSR, реализованные в CLDC HI JVM



# Приложение 2

## Методика сборки CLDC HI JVM

1. Создаем главную папку для JVM: *mkdir <yourDir>/JVM*
   Создаем папку output для собранных компонент: *mkdir <yourDir>/JVM/output*
   Создаем папку для компонент: *mkdir <yourDir>/JVM/components*

2. Компоненты.

Структура репозитория CLDC HI JVM  отображена на рис. 1

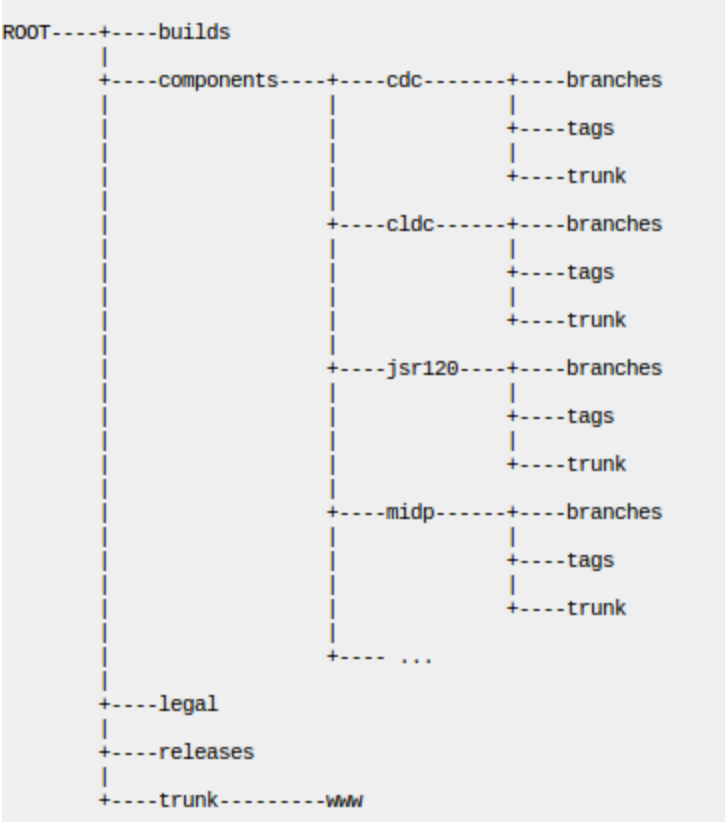



Рис.1 Структура репозитория

Берем из репозитория необходимые компоненты и кладем в <yourDir>/JVM/components:

*svn co https://svn.java.net/svn/phoneme~svn/components/pcsl/trunk pcsl*
*svn co https://svn.java.net/svn/phoneme~svn/components/cldc/trunk cldc*
*svn co https://svn.java.net/svn/phoneme~svn/components/midp/trunk midp*
*svn co https://svn.java.net/svn/phoneme~svn/components/tools/trunk tools*

3. Перед началом сборки устанавливаем инструменты.

- Кросс-компилятор. Берем и устанавливаем buildroot по ссылке : https://code.google.com/p/buildroot/wiki/BuildingonUbuntu
  + в make menuconfig не забыть включить в toolchain C++
  Также обязательно должны быть ar, as, cpp, g++, gcc, gcov, ld, nm, ranlib, strip.

  В <*yourDir*>/*buildroot*/*output*/*host*/*opt*/*ext-toolchain*/ (либо <*yourDir*>/*buildroot*/*output*/*host*/*usr*/*bin*) сделать еще соответствующие ссылки, в духе: g++ → arm-none-linux-gnueabi-g++
  for i in arm-none-linux-*; do ln -s $i `echo $i | cut -d'-' -f 3`; done

- JDK: apt-get install openjdk-7-jdk
- gawk

4. Устанавливаем переменные окружения.

*export MEHOME=<yourDir>*
*export JDK_DIR=/usr/jdk1.7.0_17*
*export GNU_TOOLS_DIR=<yourDir>/buildroot/output/host/usr/*
*export PATH=$PATH:$JDK_DIR/bin*
*export PCSL_OUTPUT_DIR=$MEHOME/output/pcsl*
*export PCSL_PLATFORM=linux_arm_gcc*
*export PCSL_OS=linux*
*export PCSL_CPU=arm*
*export NETWORK_MODULE=bsd/generic*
*export ENABLE_PCSL=true*
*export ENABLE_ISOLATES=true*
*export JVMWorkSpace=$MEHOME/cldc*
*export JVMBuildSpace=$MEHOME/output/cldc*
*export MIDP_OUTPUT_DIR=$MEHOME/output/midp*
*export CLDC_DIST_DIR=$MEHOME/output/cldc/linux_arm/dist*
*export TOOLS_DIR=$MEHOME/tools*
*export TARGET_CPU=arm*
*export CPU=arm*
*export USE_MIDP=true*
*export USE_MULTIPLE_ISOLATES=true*
*export ENABLE_COMPILATION_WARNINGS=true*
*export ENABLE_JAVA_DEBUGGER=true*
*export ENABLE_ROM_DEBUG_SYMBOLS=true*

*export ENABLE_ROM_JAVA_DEBUGGER=true*
*export ENABLE_SYSTEM_CLASSES_DEBUG=true*

5. **PCSL**-библиотеки

Используются для построения CLDC. Содержит несколько доп. опций, включающих файловую систему, сеть, память и печать. Используются переменные окружения FILE_MODULE, NETWORK_MODULE, MEMORY_MODULE, PRINT_MODULE. Возможные значения:

File system: ram posix win32 stubs
Memory: malloc heap stubs
network: bsd/qte bsd/generic sos winsock stubs
print: stdout file stubs

По умолчанию используются значения: posix, malloc, socket/bsd/qte, stdout
Идем в *$MEHOME/components/pcsl* и собираем make'ом.

6. Конфигурация **CLDC**

В общих чертах, сборка JVM (т.е. сборка профиля CLDC, который включает в себя JVM) состоит из следующих основных фаз:

1. Построение interpreter loop generator (loopgen)
2. Построение ROM image generator (romizer, romgen)
3. Execute loopgen to generate interpreter loop
4. Execute romgen to generate ROM image
5. Построение JVM, связывание interpreter loop и ROM image.

Loopgen и romgen составляющие части JVM и они должны быть собраны на хосте с помощью native toolchain хоста (в нашем случае это buildroot).
Необходимые переменные окружения:

JVMWorkSpace - директория CLDC
JVMBuildSpace - путь для output/cldc
ENABLE_PCSL=true - использование уже постороенных PCSL-библиотек
PCSL_OUTPUT_DIR - путь до PCSL-библиотек
ENABLE_ISOLATES=true - многозадачность

Построение:

Заменить в файле *$MEHOME/components/cldc/src/vm/cpu/c/AsmStubs_i386.s* строку 27: i486 на i686.

MAKE сделать в *$MEHOME/components/cldc/build/linux_arm/*

7. Профиль **MIDP**

Построение:

На текущем шаге в CLDC HI JVM можно добавить дополнительные пакеты (JSR, см. Приложение 1), которые могут понадобиться для работы Вашего Midlet. Чтобы добавить их, редактируем файл:

*$MEHOME/components/midp/build/linux_fb_gcc/Options.gmk*, где *linux_fb_gcc* - сборка для ядра linux с *CPU=arm* и использованием *frameBuffer* для отображения изображения (но в целом — директория на ваш выбор).

В *$MEHOME/components/midp/src/ams/example/ams_common_port/default/native/command LineUtil_md.c* условный оператор на 80 строке изменить на:

```
if(midp_home != NULL){

    strncpy(dirBuffer, midp_home, MAX_FILENAME_LENGTH);

    return midp_home;

}
```

В *$MEHOME/components/midp/build/common/makefiles/gcc.gmk* в строке 128 стоит опция -Werror (воспринимать варнинги за ошибки). Её надо убрать.

Также в этом файле заменить строку 105 на:

```
LD_END_GROUP ?= -Xlinker --end-group
```

После чего сделать MAKE сделать в *$MEHOME/components/midp/build/linux_fb_gcc/*

## Как запускать CLDC HI JVM?

***Для Linux:***

Установка MIDlet:

```
export MIDP_HOME=<yourDir>/output/midp
```

*cd $MIDP_HOME/bin/arm/*

Далее:

*./installMidlet file:///Application.jar*

где Application.jar - ваше java-приложение

Запуск MIDlet:

*./runMIDlet <suit number>*

где <suit number> - номер приложения, который выдаст установщик Midlet, либо можно узнать, выполнив команду: *./listMidlets.sh*

Запуск исключительно cldc_vm в консольном режиме:

*./cldc_vm -cp <путь>/jbe.jar jbe.DoAll*

Для системы ***Windows*** используются аналогичные команды.