ЗАЩИЩЕННАЯ СРЕДА РАЗРАБОТКИ И ИСПОЛНЕНИЯ «JAVA AT»

Руководство системного программиста

Листов 18

2026

АННОТАЦИЯ

Данный документ является руководством системного программиста программного обеспечения «Защищенная среда разработки и исполнения «Java AT»» (далее – ПО «Java AT») для платформы X86.

Он описывает состав, процесса сборки, установки и методов проверки ПО «Java AT», предназначенного для запуска Java программ в защищенной среде.

В разделе «Общие сведения о программе» указано назначение программы и краткая характеристика области применения программы.

В разделе «Структура программы» содержится описание ядра и списка инструментов ПО «Java AT».

В разделе «Установка программы» приведено описание установки из пакетов и сборки исходных кодов ПО «Java AT».

В разделе «Настройка программы» приведена инструкция по настройке программного обеспечения.

В разделе «Проверка программы» приведено описание примера процедуры проверки ПО «Java AT» после установки.

В разделе «Сообщения системному программисту» указаны тексты сообщений, выдаваемых в ходе выполнения настройки, проверки и выполнения программы.

СОДЕРЖАНИЕ

1. Общие сведения о программе 4
2. Структура программы....6
3. Установка программы....8
3.1. Установка из пакета 8
3.2. Сборка из исходных кодов под Astra Linux SE....8
4. Настройка программы....10
4.1. Проверка встроенной ЭЦП для файлов формата ELF 10
4.1.1. Подписывание ELF-файлов ПО «Java AT» ключом изготовителя....10
4.2. Проверка присоединенной ЭЦП для всех файлов....12
5. Проверка программы 14
6. Сообщения системному программисту 16
Перечень терминов и сокращений....18

1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

Работы по созданию ПО «Java AT» для платформы x86 под управлением защищенной операционный системы Astra Linux SE ведутся на основании технического задания на выполнение работ «ПО «Защищенная среда разработки и исполнения «Java AT»»» для платформы x86. Необходимость разработки обусловлена требованиями по созданию платформы для выполнения прикладных программ на языке Java с обеспечением соответствия требованиям информационной безопасности при обработке категорийных сведений.

ПО «Java AT» предназначено для разработки и исполнения приложений, написанных на языке программирования Java. Язык Java – это высокоуровневый объектно-ориентированный язык общего назначения, опирающийся на поддержку стандартизованной среды исполнения (виртуальной машины и библиотек классов). Совокупность среды исполнения, библиотек классов и сопутствующих компонентов, а также спецификации языка Java, спецификаций виртуальной машины и интерфейсов к ней также называют платформой Java.

ПО «Java AT» предоставляет собой полную совместимую реализацию международной спецификации Java Standard Edition (Java SE) Platform версии 21 и обладает следующими характеристиками:

- полная поддержка языка Java по стандарту Java Language Specification (JLS);
- полная реализация библиотек языка Java по стандарту Java SE 21 Edition;
- полная реализация виртуальной машины по стандарту Java Virtual Machine Specification (JVMS).
- ПО «Java AT» включает в себя следующий набор программ:
- оптимизирующий компилятор javac;
- виртуальную машину Java;
- отладчик программ jdb;
- дизассемблер java»;
- программу архивации и сжатия файлов jar;
- программу генерации документации для программ на языке Java – javadoc;

- генератор пар «открытый/закрытый ключ» keytool;
- утилиту регистрации и вызова удаленных методов rmiregistry.

На вычислительном комплексе должна быть установлена операционная система Astra Linux OS версии 1.6 или выше.

В случае сборки из исходных кодов под ОС Astra Linux SE требуются следующие пакеты:

- виртуальная машина Java версии 20;
- установленные пакеты *«ant libfreetype6-dev», «libasound2-dev», «libffi-dev», «libx11-dev», «libxext-dev», «libxrender-dev», «libxtst-dev», «libxt-dev», «libcups2-dev», «pkg-config», «libfontconfig1-dev», «libelf-dev», «parsec-dev», «file», «linux-libc-dev»*;
- архиватор zip.

2. СТРУКТУРА ПРОГРАММЫ

Ядром исполнительной системы является виртуальная машина, предназначенная для запуска и исполнения программ, написанных с использованием байт-кода Java.

ПО «Java AT» установлена в директорию *«/usr/lib/jvm/java-21-openjdk-x86»*.

Подкаталог «*bin*» содержит исполняемые файлы виртуальной машины и вспомогательных инструментов разработчика на языке Java. Рекомендуется добавить путь к этой директории в переменную окружения «*PATH*».

Список инструментов, входящих в состав ПО «Java AT», включает в себя:

- «javac» – оптимизирующий компилятор, принимающий на вход исходный код, разработанный в соответствии со спецификацией языка Java и возвращающий байт-код, соответствующий спецификации Java виртуальной машины;
- java – виртуальная машина, осуществляющая исполнение байт-кода Java, полученного на выходе компилятора javac в соответствии со спецификацией JVMS;
- jdb – отладчик программ, выполняющий трассировку кода программ и позволяющий отслеживать, устанавливать или изменять значения переменных в процессе его выполнения;
- javap – дизассемблер, осуществляющий декомпиляцию определений классов Java и отображающий их содержимое;
- jar – программа архивации и сжатия файлов;
- javadoc – программа генерации документации для программ на языке Java, использующую специальный синтаксис для комментирования частей кода и реализующую алгоритмы создания документации из комментариев посредством просмотра исходных текстов программ, выделения фрагментов документации из комментариев и связывания их с именами документируемых классов;

- keytool – инструмент командной строки, позволяющий генерировать пары «открытый/закрытый ключ» и создавать хранилище ключей, который, в частности, работает с хранилищем сертификатов открытых ключей и авторизации, используемых Java-приложениями для шифрования, аутентификации и установки соединений по протоколу HTTPS;
- rmiregistry – утилита регистрации и вызова удаленных методов.

Директория «*lib*» содержит стандартные библиотеки классов, инструментов, а также файлы настроек и ресурсов виртуальной машины.

3. УСТАНОВКА ПРОГРАММЫ

Установка ПО «Java AT» возможна двумя путями: сборка из исходных кодов и установка из пакета, собранного для текущей операционной системы.

3.1. Установка из пакета

ПО «Java AT» поставляется в виде предварительно собранного DEB-пакета «*atc-jdk21+6~astra1.7_amd64.deb*». Установка производится с помощью менеджера пакетов командой представленной ниже, после чего пакет будет размещен в стандартной директории «*/usr/lib/jvm/java-21-atc-amd64*» и автоматически интегрирован в систему:

```
sudo dpkg -I atc-jdk21+6~astra1.7_amd64.deb
```

3.2. Сборка из исходных кодов под Astra Linux SE

Для сборки ПО «Java AT» необходимо установить JDK версии 20. Установка производится с помощью команды ниже:

```
wget https://download.oracle.com/java/20/archive/jdk-20.0.2_linux-x64_bin.tar.gz
```

Путь до виртуальной машины Java версии 20 должен содержаться в переменной среды окружения «*JAVA_HOME*». Переменная среды окружения «*PATH*» должна включать путь «*$JAVA_HOME/bin*».

Далее необходимо установить системные зависимости с помощью команд, представленных ниже:

```
sudo apt-get install -y autoconf make zip build-essential libasound2-dev libcups2-dev libfontconfig1-dev libx11-dev libxext-dev libxrender-dev libxrandr-dev libxtst-dev libxt-dev libgost-dev parsec-dev file linux-libc-dev
```

Для осуществления сборки необходимо развернуть архив с исходными кодами ПО «Java AT» и перейти в папку «*sources*».

Сборка из исходных кодов осуществляется следующими командами:

```
export JAVA_HOME=*Путь до виртуальной машины Java версии 20*
./build.sh
```

Скрипт «build.sh» содержит все необходимые параметры сборки JVM, включая:

1) *«--with-integrity-check=true»* – контроль целостности, на этапе построения происходит вычисление контрольных сумм по алгоритму «SHA-256» для всех файлов, входящих в пакет JDK с помощью системного приложения «sha256sum»;
2) *«--with-protected-java-root=true»* – защищенная программная среда;
3) *«--with-memory-mangling=true»* – зануление памяти;
4) *«--with-parsec-support=true»* – мандатный контроль.

Результатом успешной сборки будет готовая структура директорий и файлов ПО «Java AT» версии 21, расположенная в поддиректориях «*build/linux-x86_64-server-release/images/jdk»*.

4. НАСТРОЙКА ПРОГРАММЫ

В ПО «Java АТ» реализован механизм контроля целостности в режиме замкнутой программной среды, поддерживаемой ОС Astra Linux SE. Средства создания замкнутой программной среды предоставляют возможность внедрения цифровой подписи в исполняемые файлы формата «ELF», входящие в состав устанавливаемого программного обеспечения, а также в расширенные атрибуты всех файлов. Контроль целостности реализован в модуле ядра ОС «*digsig_verif*», который является не выгружаемым модулем ядра Astra Linux SE.

С помощью данных средств обеспечивается «подписывание» исполняемых модулей, библиотек и пакетов Java-классов ПО «Java АТ».

4.1. Проверка встроенной ЭЦП для файлов формата ELF

Замкнутая программная среда позволяет проверять ELF-файлы при запуске на исполнение (загрузке в оперативную память).

Для работы приложения в данном режиме все исполняемые файлы и разделяемые библиотеки должны иметь встроенную подпись, а публичный ключ должен находиться в каталоге `/etc/digsig/keys` и быть зарегистрирован в ядре ОС командой:

```
sudo update-initramfs -uk all
```

Все ELF-файлы поставляемой в DEB-пакете ПО «Java АТ» содержат встроенную ЭЦП, подписанную ключом изготовителя.

Если ПО «Java АТ» собиралась из исходников, ELF-файлы необходимо подписать самостоятельно.

4.1.1. Подписывание ELF-файлов ПО «Java АТ» ключом изготовителя

Для подписания необходимо иметь собственные ключи, подписанные ключом изготовителя.

Импортировать ключи:

```
gpg –import ***.gpg
gpg –import ***.key
```

Узнать идентификатор полученного ключа:

```
gpg –list-keys
```

Подписать все ELF-файлы ПО «Java AT». Например, с помощью следующего bash-скрипта:

```
#!/bin/bash

export GPG_TTY=$(tty)

TARGET_DIR="/home/admin/java-21-atc-amd64"
# ID ключа
GPG_ID="97BFBA1761250261C95CFFFD6E90A9528AA5BF00"

echo "Проверяю утилиту bsign"
tmp_bsign="/tmp/elf-file-4-bsign"
cp /bin/true $tmp_bsign

bsign -s --pgoptions=" --default-key=${GPG_ID}" $tmp_bsign

if [ $? -ne 0 ]; then
   echo "Не удалось подписать '/tmp/bsign-test' с помощью bsign"
   exit $?
fi

rm $tmp_bsign

echo "Подписываю ELF файлы в $TARGET_DIR..."

find "$TARGET_DIR" -type f | while read file; do
   magic_elf_num=$(head -c 4 "$file" 2>/dev/null)
   # Определяем ELF-файл по первым 4 байтам информации о файле
   if [ "$magic_elf_num" = "$(printf '\x7fELF')" ]; then
      echo "Подписываю ELF файл: $file"
      bsign -s -N --pgoptions=" --default-key=${GPG_ID}" "$file"
   fi
```

```
done


echo "Готово!"
```

ПО «Java AT» обладает собственным механизмом контроля целостности. После подписывания ELF-файлов необходимо обновить референсные контрольные суммы "jdk.sha256sum".

```
cd /home/admin/java-21-atc-amd64
find ./ -type f -name "*" -not -name "jdk.sha256sum" -exec sha256sum {} \; > jdk.sha256sum
```

4.2. Проверка присоединенной ЭЦП для всех файлов

Замкнутая программная среда позволяет проверять все файлы при открытии на чтение или запись.

Данный режим замкнутой программной среды не требует наличия ключей изготовителя.

«Подписывание» осуществляется с помощью ключа пользователя и присоединенной ЭЦП.

Необходимо создать дополнительный ключ (тип ключа - ГОСТ Р 34.10-2012):

```
$ gpg –full-generate-key
```

Опубликовать созданный ключ:

```
$ gpg --export <идентификатор ключа>` > key.gpg
$ sudo cp key.gpg /etc/digsig/xattr_keys/
```

Применить изменения:

```
$ sudo update-initramfs -u -k all
$ sudo shutdown -r now
```

Подписывание утилитой «bsign» произвольного файла с внедрением подписи только в расширенные атрибуты:

```
$ bsign --sign --xattr test_elf
```

Для работы замкнутой программной среды в режиме проверки присоединенной ЭЦП в расширенных атрибутах при открытии файла на чтение и запись необходимо указать в «/etc/digsig/xattr_control» список шаблонов имен, определяющих файлы,

в которых проверяется присоединенная ЭЦП, а также включить необходимый режим ЗПС «*DIGSIG_XATTR_MODE=1*».

Начиная с версии ОС Astra Linux 1.7.6, пользователю доступна проверка цифровой подписи исполняемых файлы формата «ELF» с дополнительным ключом.

Необходимо включить ЗПС в режиме проверки цифровой подписи исполняемых файлов и разделяемых библиотек «*DIGSIG_ELF_MODE=1*», а также установить корректный режим проверки цифровой подписи исполняемых файлов «*DIGSIG_ELF_VERIFICATION_MODE=3*». Теперь исполняемые файлы с присоединенной ЭЦП, подписанной собственным ключом пользователя, будут работать в замкнутой программной среде.

При работе с ЗПС в режиме проверки цифровой подписи исполняемых файлов нет необходимости во встроенном контроле целостности, реализованном на уровне JDK. Его можно отключить на этапе сборки передав скрипту *configure* параметр *«--with-integrity-check=false*».

5. ПРОВЕРКА ПРОГРАММЫ

После успешной установки ПО «Java AT» можно компилировать, запускать и отлаживать Java-программы.

Для проверки достаточно написать и запустить простейшую программу на языке Java. Приведем классический пример программы HelloWorld:

```
public class HelloWorld {
        public static void main(String[] args) {
                System.out.println(“Hello!”);
        }
}
```

Текст программы необходимо сохранить как текстовый файл, используя любой текстовый редактор. Имя файла должно точно соответствовать имени класса и иметь расширение «java». Таким образом, для приведенного примера файл должен называться «*HelloWorld.java*».

Далее необходимо скомпилировать программу при помощи установленного компилятора javac. Например:

```
javac HelloWorld.java
```

Компилятор переведет исходный текст программы в Java байт-код, пригодный для непосредственного исполнения Виртуальной машиной Java, и сохранит его в файл, одноименный с программой, но имеющий расширение «*class*». В нашем примере это «*HelloWorld.class*». Для запуска скомпилированной программы следует вызвать программу java из ПО «Java AT» с указанием имени основного класса программы, например:

```
java HelloWorld
```

В результате исполнения программы HelloWorld в консоли должна отобразиться строка «*Hello!»* и должен вернуться код успешного завершения «0».

Расширенное тестирование ПО «Java AT» может быть выполнено с помощью специальных проверочных тестов и запускающего их скрипта «*main.py*», написанного

на языке программирования Python, входящего в архив «*java_tests.tgz*» из архива поставки.

В качестве аргумента скрипту передается номер пункта «Технического Задания», содержащий требование к ПО «Java AT». Скрипт выполняет тест, проверяющий заданное требование.

Если аргумент не задан, последовательно выполняются проверки для всех пунктов «Технического задания», подлежащих проверке.

В случае успешного завершения печатается строка «Пункт_ПМИ: Успешно». В случае неуспешного завершения печатается строка: «Пункт_ПМИ: Неуспешно».

Логи прогонов запускаемых тестов сохраняются в специальной директории логов.

Перед проведением испытаний необходимо перейти в директорию с распакованным архивом тестов и задать путь до тестируемого ПО «Java AT» с помощью команды:

```
«export JAVA_HOME= <путь>»
```

Строка вызова проверочного скрипта «main.py» имеет следующий вид:

```
usage: main.py [-h] [--java JAVA] [--testcase TESTCASE] [--work WORK_DIR] [--loglevel 1|0]
```

Список возможных параметром скрипта:

1) «*-h*», «*--help*» - распечатать список параметров;
2) «*--java JAVA*» - задать путь до ПО «Java AT» (JDK), если данный аргумент не задан, берется путь из переменной среды окружения «JAVA_HOME»;
3) «*--testcase TESTCASE*» - запустить тестирование для проверки заданного пункта ПМИ;
4) «*--work WORK_DIR*» - задать путь до рабочей директории, если данный аргумент не задан, используется стандартная директория – «*workdir*»;
5) «*--loglevel LOG LEVEL*» - регулировать уровень логгирования в консоли, (0 – по умолчанию, стандартные вывод, 1 – вывод всего лога).

6. СООБЩЕНИЯ СИСТЕМНОМУ ПРОГРАММИСТУ

При работе в режиме эксплуатации формируется ряд сообщений системному программисту. Эти сообщения возникают в процессе установки, первоначальной настройки и проверки конфигурации виртуальной машины Java.

Основные типы сообщений:

- сообщения об установке и проверке JDK;
- предупреждения конфигурации виртуальной машины Java;
- ошибки памяти и ресурсов;
- сообщения инициализации приложенного сервера;
- диагностические сообщения.

Данные сообщения генерируются непосредственно виртуальной машиной Java, установщиками JDK или серверами приложений на основе Java при обнаружении проблем конфигурации, нехватке ресурсов или некорректных настройках среды выполнения. Сообщения направлены на помощь системному программисту в диагностике и устранении проблем настройки Java-окружения

Перечень сообщений оператору приведен в документе «Руководство пользователя (оператора)», входящем в комплект программной документации.

Сообщения об установке и проверке JDK:

"Error: could not open '/usr/lib/jvm/jdk-21-oracle-x64/lib/jvm.cfg'" - ошибка отсутствия конфигурационного файла JVM

"JAVA_HOME is not set. Unexpected results may occur. Set JAVA_HOME to the directory of your local JDK to avoid this message." - предупреждение о некорректной настройке переменной окружения

"Java version check successful: openjdk version '21.0.2' 2024-01-16" - подтверждение успешной установки

Предупреждения конфигурации виртуальной машины Java:

"Warning: Printed when an Agent Is Loaded into a Running VM" – предупреждение о динамической загрузке агентов в JVM 21

"Invalid system properties for logging and tracing settings. Degraded mode is set temporarily." – сообщение о некорректных свойствах логирования

"Warning: no leading - on line X of jvm.cfg" и on line X of jvm.cfg"` – предупреждения о синтаксических ошибках в конфигурационном файле JVM.

Ошибки памяти и ресурсов:

"There is insufficient memory for the Java Runtime Environment to continue. Cannot create GC thread. Out of system resources." – критическая ошибка нехватки памяти

"Could not create the Java Virtual Machine" – общая ошибка невозможности создания JVM

"Native memory allocation (mmap) failed to map X bytes for committing reserved memory" – ошибка выделения памяти системой

Сообщения инициализации приложенного сервера:

"JBoss Bootstrap Environment" с детализацией параметров JAVA_HOME, JAVA_OPTS, CLASSPATH

"Server Configuration: Bootstrap, Common Base, Server Library" – информация о конфигурации сервера приложений

"Unable to read the logging configuration from 'file:logging.properties'" – ошибка чтения конфигурации логирования

Диагностические сообщения:

"Using system property jdk.instrument.traceUsage=true" – трассировка использования инструментирования Java

Сообщения об обнаружении конфликтующих версий Java при попытке установки более старой версии поверх новой

ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

ОС	Операционная система
ПО «Java AT»	Программное обеспечение «Защищенная среда разработки и исполнения «Java AT»»
Astra Linux SE	Операционная система Астра Линукс
ELF	от англ. Executable and Linkable Format, формат исполняемых файлов и библиотек
JAR	от англ. Java Archive, формат файлов для хранения классов и ресурсов Java
Java SE 21 Edition	Стандартная спецификация Java версии 21
JDK	от англ. Java Developer’s Kit, набор инструментов разработчика Java
JLS	от англ. Java Language Specification, спецификации языка Java
JVM	от англ. Java Virtual Machine, виртуальная машина Java
PARSEC	Программное обеспечение для управления привилегиями пользовательских процессов
SHA-256	Криптографический алгоритм хеширования для вычисления контрольных сумм