Бинарный компилятор уровня приложений

Руководство пользователя

Оглавление

Термины

В данном разделе не приводятся вырванные из контекста определения. Вместо этого дается перечень основных терминов и указания, где пользователь бинарного компилятора может найти соответствующие определения. Термины в предложениях, в которых даются их определения, так же, как и в этом разделе, выделены курсивом.

Исходная платформа, *целевая платформа*, *x86-приложение* – «Назначение бинарного компилятора[1]».

Нативная файловая система (*нативная ФС*), *гостевая файловая система* (*гостевая ФС*), *корень гостевой файловой системы* - «Установка бинарного компилятора».

Менеджер запуска – «Расположение файлов бинарного компилятора».

X86-окружение, *владелец* x86-окружения, *тип* x86-окружения, *имя* x86-окружения – «X86-окружение бинарного компилятора».

Параметр бинарного компилятора, *источники определения* параметра бинарного компилятора, *приоритет* источника определения, *рабочее значение* параметра, *секция* параметров – «Параметры бинарного компилятора».

Конфигурационный файл, *системный конфигурационный файл*, *базовый конфигурационный файл*, *пользовательский конфигурационный файл* — «Конфигурационные файлы».

Образ гостевой файловой системы (*образ гостевой ФС*) - «Создание гостевой файловой системы».

[1] здесь и далее в документе, если это не оговорено явно, словосочетанием бинарный компилятор обозначается исключительно бинарный компилятор уровня приложений (противопоставляется бинарному компилятору уровня системы, известному как Lintel).

Назначение бинарного компилятора

Бинарный компилятор является приложением операционной системы ОС «Эльбрус» и позволяет исполнять linux-приложения[2] в кодах x86/x86_64 (коды так называемой *исходной платформы*), далее называемые *x86-приложениями*, на вычислительных машинах, построенных на базе микропроцессоров семейства «Эльбрус» (далее *целевая платформа*).

В данном руководстве описываются особенности бинарного компилятора версии 5.0. Оно распространяется на бинарные компиляторы, предназначенные для запуска на вычислительных машинах, построенных на базе микропроцессоров Эльбрус-4С, Эльбрус-1С+, Эльбрус-8С, Эльбрус-8СВ, Эльбрус-2С3 и Эльбрус-16С.

Версия 5.0 бинарного компилятора содержит ряд существенных изменений относительно предыдущих релизов. В частности, она предназначена для эксплуатации в среде ОС «Эльбрус» на базе ядра 5.10[3] и не подходит для исполнения x86-приложений на более старых ядрах[4]. Перед началом работы с новой версией рекомендуется внимательно изучить данное руководство.

[2] приложения, работающие под управлением операционной системы семейства Linux.

[3] также поддерживает работу с ядрами 6.1.

[4] для операционных систем с ядрами 3.*, 4.* и 5.4 рекомендуется использовать более ранние версии бинарного компилятора — до rtc-4.4 включительно.

Установка бинарного компилятора

Бинарный компилятор оформлен в виде deb-пакета (rtc_*.deb), входящего в состав общего программного обеспечения для операционной системы ОС «Эльбрус». Установка производится стандартным образом с применением пакетного менеджера.

Компоненты бинарного компилятора при установке размещаются в *нативной файловой системе*[5] (*нативной ФС*), представляющей из себя совокупность файлов и директорий всех файловых систем (ФС), смонтированных в рабочей среде ОС «Эльбрус» в корневую директорию[6]. Сразу после установки бинарный компилятор способен исполнять только статически связанные x86-приложения, не требующие наличия по стандартным путям динамических библиотек. Для возможности запуска динамически связанных x86-приложений потребуется внутри нативной файловой системы[7] разместить так называемую *гостевую файловую систему* (*гостевую ФС*), представляющую из себя совокупность файлов и директорий подмножества[8] файловых систем, монтируемых в рабочей среде x86 операционной системы семейства linux в корневую директорию[9]. Директория нативной ФС, внутри которой располагается гостевая ФС, называется *корнем гостевой файловой системы*. Порядок действий по разворачиванию гостевой файловой системы приводится в разделе «Создание гостевой файловой системы».

При установке бинарного компилятора кроме размещения необходимых файлов в операционной системе регистрируется новая пользовательская

[5] файловой системой в обычном понимании не является; здесь и в случае с гостевой файловой системой, определение которой дается чуть ниже, под этим словосочетанием понимается совокупность файлов и директорий, оформленных в виде некоторой структуры, определяющей их взаимное расположение.

[6] и ее поддиректории.

[7] в одной из ее поддиректорий, путь до которой определяется пользователем.

[8] например, в расчет не берутся виртуальные файловые системы (proc, dev и т.п.).

[9] и ее поддиректории.

группа bincomp[10], для членов которой реализован расширенный набор возможностей по эксплуатации бинарного компилятора.

[10] если группа не была создана ранее.

Расположение файлов бинарного компилятора

При установке бинарного компилятора исполняемые файлы размещаются в директории /opt/mcst/rtc-5.0/bin, основным среди них является *менеджер запуска* — startx86. Именно с него начинается работа бинарного компилятора. Также в указанной директории размещается пара трансляторов[11], отвечающих непосредственно за исполнение x86-приложений, и две символьные ссылки на них — rtc32[12] и rtc64[13].

> *Замечание. Прямой запуск трансляторов, минуя использование менеджера startx86, является некорректным действием со стороны пользователя.*

Рядом с исполняемыми файлами, в директории /opt/mcst/rtc-5.0/doc, размещается пользовательская документация, а в /opt/mcst/rtc-5.0/tools — вспомогательные скрипты pack_guestfs.sh и unpack_guestfs.sh для создания гостевой ФС.

Кроме того, при установке бинарного компилятора регистрируется системная служба — в директории /etc/init.d/ размещается скрипт bincomp. Для уровней запуска операционной системы (system runlevel) 3 и 5 настраивается автоматический старт службы.

> *Замечание. Настоятельно рекомендуется использовать бинарный компилятор в среде операционной системы, запущенной на уровне 3 или 5, и не производить отключения[14] системной службы bincomp.*

Также при установке бинарного компилятора по путям /etc/bincomp/system.conf, /etc/bincomp/default.conf размещаются

[11] файлы с именами вида rtc_opt_*.

[12] указывает на транслятор, отвечающий за исполнение 32-битных x86-приложений.

[13] указывает на транслятор, отвечающий за исполнение 64-битных x86-приложений.

[14] в противном случае прямой запуск x86-приложений окажется невозможным, кроме того, могут возникнуть сложности при работе с x86-окружениями.

системный и базовый конфигурационные файлы и создаются директории /var/log/bincomp, /var/lib/bincomp/enabled. Директория /var/log/bincomp используется бинарным компилятором как штатное место[15] хранения лог-файлов, а директория /var/lib/bincomp/enabled предназначена для размещения пользовательских конфигурационных файлов, на базе которых при старте операционной системы служба bincomp создает x86-окружения.

Помимо файлов и директорий, создаваемых при установке бинарного компилятора, стоит также отметить файл /proc/bincomp/search_path. В нем хранится путь до менеджера запуска startx86. Именно по пути, указанному в файле /proc/bincomp/search_path, ядро пытается найти основной исполняемый файл бинарного компилятора при прямых запусках x86-приложений. Рекомендуется не изменять содержимое данного файла самостоятельно.

[15] пользователь при помощи конфигурационных файлов и опции менеджера запуска может указать другое место для размещения логов (см. описание секции Params и ее параметра LogPath в разделе «Параметры бинарного компилятора»).

X86-окружение бинарного компилятора

Исполнение бинарным компилятором гостевого приложения производится в специальном *x86-окружении*, создаваемом на базе пространства имен файловой системы (mount namespace) ядра Linux. В свою очередь, пространство имен определяется значениями особой группы параметров бинарного компилятора[16], которые фиксируют расположение нового корня ФС и перечень файлов и директорий, отображаемых из нативной ФС в гостевую. У каждого x86-окружения есть *владелец* — пользователь операционной системы, который самостоятельно создал или от имени которого было создано[17] это x86-окружение. X86-окружения бывают двух *типов*: для личного и общего использования. Первые доступны только их владельцу[18], вторые — всем пользователям.

Каждому x86-окружению при создании присваивается *имя*, его либо явно указывает пользователь[19], либо оно назначается менеджером запуска. К имени x86-окружения предъявляются следующие требования:

- все x86-окружения для общего использования должны обладать уникальным именем;
- все x86-окружения для личного использования, имеющие одного владельца, должны обладать уникальным именем;
- совпадение имен x86-окружений для личного использования, имеющих разных владельцев, допускается;
- совпадение имен двух x86-окружений разного типа допускается;

[16] см. описание секций BaseEnv и AdvEnv в разделе «Параметры бинарного компилятора».

[17] пользователем root (см. описание секции BaseEnv и ее параметра Uid в разделе «Параметры бинарного компилятора»).

[18] и пользователю root.

[19] задается при помощи необязательного параметра менеджера запуска startx86 или при помощи имени пользовательского конфигурационного файла, размещенного в специальной директории /var/lib/bincop/enabled.

- имя x86-окружения, заданное пользователем, может состоять из латинских букв, цифр и трех специальных символов (-_@), его длина не должна превышать 256 символов.

Имя x86-окружения, формируемое менеджером запуска, определяется исходя из пути до корня гостевой ФС[20]. Поэтому при создании нескольких x86-окружений одного типа с одним и тем же корнем гостевой ФС понадобятся пользовательские имена. Стоит отметить, что в подавляющем большинстве случаев для каждого корня гостевой ФС будет достаточно создать одно x86-окружение и все запуски бинарного компилятора проводить в нем[21].

Поскольку x86-окружения реализованы на базе пространства имен файловой системы, то их время жизни ограничено временем работы операционной системы. При необходимости автоматическое создание x86-окружений на этапе запуска операционной системы можно возложить на службу bincomp[22]. Для этого достаточно поместить[23] пользовательский конфигурационный файл с расширением conf[24] в директорию /var/lib/bincomp/enabled. В качестве имени создаваемого x86-окружения будет использовано имя пользовательского конфигурационного файла. Параметр Shared, указанный в конфигурационном файле, определит тип x86-окружения — для личного (0) или для общего (1) использования, а параметр Uid — его создателя[25].

[20] является значением хеш-функции, аргументом которой выступает строка с путем до корня гостевой ФС.

[21] о том, как использовать одно и то же x86-окружение для запуска разных задач, подробно написано в разделе «Запуск бинарного компилятора».

[22] см. раздел «Расположение файлов бинарного компилятора».

[23] данное действие доступно системному администратору (пользователю root) и пользователям, состоящим в группе bincomp (см. раздел «Установка бинарного компилятора»).

[24] обязательно установив в нем корректные значения для параметров Uid, Shared (см. описание секции BaseEnv в разделе «Параметры бинарного компилятора»).

[25] обязательно при создании рабочего окружения для личного использования.

Параметры бинарного компилятора

Пользователю доступен позволяющий управлять работой бинарного компилятора интерфейс, реализованный на базе набора *параметров*. Значения параметров устанавливаются с помощью конфигурационных файлов и опций менеджера запуска. Большинство параметров имеет значение по умолчанию, но есть и такие, что требуют явного определения в момент старта бинарного компилятора. При этом системный и базовый конфигурационные файлы, создаваемые при установке бинарного компилятора, уже включают в себя набор определений параметров, достаточный для запуска статически связанных x86-приложений.

Конфигурационные файлы, опции менеджера запуска startx86, а также исходные коды бинарного компилятора с заданными в них значениями по умолчанию могут быть рассмотрены как *источники определения* параметров бинарного компилятора. На множестве источников вводится порядок в соответствии с их *приоритетом*[26]:

- опции менеджера запуска startx86;
- системный конфигурационный файл;
- пользовательский конфигурационный файл;
- базовый конфигурационный файл;
- значения по умолчанию, заданные в исходных кодах бинарного компилятора.

Если один и тот же параметр имеет несколько определений в разных источниках, то в большинстве случаев[27] в качестве *рабочего* будет выбираться то значение, что указано в источнике с наивысшим приоритетом. Наличие

[26] от наиболее приоритетного к наименее.
[27] кроме особого параметра BindPath.

нескольких определений параметра в одном источнике ошибкой не считается, при этом среди них выбирается первое[28], а остальные игнорируются.

Все параметры, исходя из особенностей их влияния на работу бинарного компилятора, разбиты на группы, далее называемые *секциями*. Всего существует 5 секций: System, BaseEnv, AdvEnv, Params и Debug. Описания секций и параметров, входящих в их состав, приводятся ниже. В конце раздела приведена сводная таблица с информацией о параметрах бинарного компилятора.

Секция System

Секция System включает в себя пять параметров: SuidGroupId, GlobalDisableL4, GlobalDisableParallel, L3Threshold, L4Threshold. Описание данной секции производится исключительно в системном конфигурационном файле, который формируется при установке бинарного компилятора. Прописанные в нем значения параметров учитывают особенности операционной системы и аппаратуры, изменять их может только системный администратор.

Значение параметра SuidGroupId определяет группу пользователей[29], которым разрешен запуск suid-приложений, расположенных внутри гостевой ФС. По умолчанию такой группой назначается группа с нулевым идентификатором[30], однако значение параметра SuidGroupId может быть изменено. В качестве подходящего варианта можно рассматривать идентификатор группы bincomp, создаваемой при установке бинарного компилятора. В таком случае системный администратор должен не забыть включить в группу подходящих пользователей.

[28] кроме параметра BindPath, для которого важны все определения, указанные в конфигурационном файле.

[29] числовой идентификатор группы — guid.

[30] даже если значение для данного параметра не указано в системном конфигурационном файле.

Остальные четыре параметра влияют на процесс трансляции[31] кодов исполняемого бинарным компилятором x86-приложения. Для лучшего понимания назначения данных параметров рекомендуется ознакомиться с приведенным ниже замечанием.

> *Замечание. В процессе своей работы бинарный компилятор транслирует исходные коды в функционально эквивалентные им коды процессора семейства Эльбрус, впоследствии исполняя их на вычислительной машине. Существует четыре уровня (режима) трансляции: два шаблонных и два регионных. Любой новый исходный код (исполняемый впервые) транслируется на младшем шаблонном уровне; код, исполнение которого время от времени повторяется, может быть оттранслирован на более высоком уровне — сначала на старшем шаблонном, затем на младшем регионном и, наконец, на старшем регионном уровне по мере увеличения количества исполнений этого кода. На всех четырех уровнях трансляции создаются коды для процессора семейства Эльбрус, которые размещаются в специальных программных кешах и могут быть использованы неоднократно[32]. То есть если для некоторого исходного кода трансляция однажды уже была произведена, то при необходимости его повторного исполнения будет использоваться результат ранее проведенной трансляции, хранимый в одном из программных кешей. Уровни трансляции различаются временем, затрачиваемым на создание e2k-кода, а также качеством[33] получаемого кода. Так, на шаблонных уровнях создается не*

[31] трансляция — процесс создания нативного e2k-кода (исполняемого на процессоре семейства «Эльбрус») функционально эквивалентного некоторому фрагменту исходного x86-кода.

[32] в рамках исполнения текущего x86-приложения.

[33] в данном случае под качеством понимается скорость работы целевого кода (усредненное количество e2k-команд, необходимое для исполнения одной x86-инструкции).

очень качественный код, но делается это быстро, в то же время на регионных уровнях, хоть и требуется значительно больше времени на проведение трансляции, но на выходе получается более быстрый код. Чтобы избежать эффекта «замирания» при исполнении x86-приложения, регионные трансляции по умолчанию проводятся в отдельных процессах, специализирующихся исключительно на создании e2k-кода. Механизм, реализующий такой подход, называется параллельной трансляцией.

Параметр GlobalDisableL4 определяет задействован ли во время работы бинарного компилятора старший регионный уровень трансляции. Если параметр имеет значение 0[34], то используется полноценная 4-х уровневая схема ускорения исходного кода, если же параметр имеет значение 1, то используется 3-х уровневая схема ускорения исходного кода, при которой отключается старший регионный уровень. Значения параметра, отличные от 0 и 1, являются некорректными.

Параметр GlobalDisableParallel управляет механизмом параллельной трансляции. Если параметр имеет значение 0[35], то механизм параллельной трансляции задействован, если же параметр имеет значение 1, то регионные трансляции производятся в том же процессе, что отвечает и за исполнение x86-приложения. В таком случае проведение регионной трансляции «останавливает» исполнение гостевого приложения. Значения параметра, отличные от 0 и 1, являются некорректными.

Параметр L3Threshold определяет сколько раз должен исполниться какой-либо фрагмент исходного кода, чтобы на базе него была произведена трансляция на младшем регионном уровне. В системном конфигурационном файле определение данного параметра отсутствует. Предполагается, что

[34] значение по умолчанию.
[35] значение по умолчанию.

бинарный компилятор будет использовать значение по умолчанию — 10000. Явное указание другого значения для параметра L3Threshold без рекомендации со стороны разработчиков бинарного компилятора может иметь непредсказуемые последствия. Настоятельно рекомендуется отказаться от управления параметром L3Threshold.

Параметр L4Threshold определяет сколько раз должен исполниться какой-либо фрагмент исходного кода, чтобы на базе него была произведена трансляция на старшем регионном уровне. Значение данного параметра учитывается бинарным компилятором только, если параметр GlobalDisableL4 имеет значение 0. Аналогично параметру L3Threshold значение для L4Threshold не указывается в системном конфигурационном файле. Предполагается, что бинарный компилятор будет использовать значение по умолчанию — 1000000. Настоятельно рекомендуется отказаться от управления параметром L4Threshold.

Параметры секции System, будучи по сути глобальными, учитываются при всех запусках бинарного компилятора, производимых на вычислительной машине, и не могут быть изменены опциями менеджера запуска[36].

Секция BaseEnv

Секция BaseEnv включает в себя четыре параметра: PathPrefix, Uid, Shared и IgnoreDefaultBindings. Описание секции BaseEnv может быть размещено в базовом и пользовательском конфигурационных файлах. Кроме того, каждый параметр секции BaseEnv может быть определен и с помощью опции менеджера запуска. В качестве рабочего значения параметра, относящегося к секции BaseEnv, выбирается то, что указано в источнике с наивысшим приоритетом. Данная группа параметров задает базовые

[36] опции --disable_parallel и --disable_l4 хоть и выполняют действия, подобные действиям параметров GlobalDisableParallel и GlobalDisableL4, но оказывают влияние лишь на конкретный запуск бинарного компилятора, поэтому их правильнее связывать с параметрами DisableParallel и DisableL4 секции Debug.

характеристики x86-окружения, используемого бинарным компилятором при исполнении x86-приложения.

> *Замечание. Запуск бинарного компилятора с указанием имени существующего x86-окружения, для которого значения параметров секции BaseEnv отличаются от рабочих значений запуска, расценивается как ошибка.*

Параметр PathPrefix определяет путь до корня гостевой ФС. Данный параметр не имеет значения по умолчанию. Поэтому в базовом конфигурационном файле, создаваемом при установке бинарного компилятора, для параметра PathPrefix указывается значение "/". Такое определение параметра позволяет запускать лишь статически связанные x86-приложения. Системный администратор может развернуть гостевую ФС и, с учетом расположения ее корня, настроить параметр PathPrefix в базовом конфигурационном файле. Тогда появится возможность запуска динамически связанных x86-приложений. Непривилегированный пользователь в своей работе может использовать x86-окружения с гостевой ФС, корень которой размещен по пути, отличному от указанного в базовом конфигурационном файле. Для этого значение параметра PathPrefix необходимо задавать в пользовательском конфигурационном файле или при помощи опции –path_prefix менеджера запуска.

Параметр Uid используется для определения владельца x86-окружения. Значением по умолчанию для данного параметра является идентификатор пользователя, осуществляющего запуск бинарного компилятора. Другое значение для данного параметра рекомендуется указывать только лишь в пользовательском конфигурационном файле, расположенном в специальной директории /var/lib/bincomp/enabled, или с помощью опции —uid[37] менеджера запуска. В противном случае запуск бинарного компилятора завершится с ошибкой.

[37] использование этой опции непривилегированным пользователем имеет ряд ограничений.

Параметр Shared определяет тип x86-окружения. Если параметр имеет значение 0[38], то соответствующее x86-окружение предполагает использование только владельцем, если же параметр имеет значение 1, то x86-окружение допускает использование любым пользователем. Значения параметра, отличные от 0 и 1, являются некорректными. По опции менеджера запуска –shared параметр Shared принимает значение 1.

Параметр IgnoreDefaultBindings определяет необходимость учета определений параметров секции AdvEnv, указанных в базовом[39] конфигурационном файле. Если параметр имеет значение 0[40], то определения BindPath и BindHomeDir, приведенные в базовом конфигурационном файле, учитываются при создании x86-окружения, если же параметр имеет значение 1 – не учитываются. Значения параметра, отличные от 0 и 1, являются некорректными. По опции -B параметр IgnoreDefaultBindings принимает значение 1.

> *Замечание. Определение параметра IgnoreDefaultBindings в базовом конфигурационном файле допускается, но вряд ли имеет хоть какой-то смысл.*

Параметры секции BaseEnv влияют на множество процессов бинарного компилятора, исполняющих x86-приложения в одном и том же x86-окружении.

Секция AdvEnv

Секция AdvEnv включает в себя два параметра: BindHomeDir и BindPath. Описание секции AdvEnv может быть размещено в базовом и пользовательском конфигурационном файлах. Кроме того, оба параметра

[38] значение по умолчанию.

[39] данный параметр не влияет на определения параметров BindPath и BindHomeDir, указанные в пользовательском конфигурационном файле, или при помощи опций –b менеджера запуска startx86.

[40] значение по умолчанию.

секции могут быть определены с помощью опции –b менеджера запуска. Данная группа параметров задает расширенные характеристики x86-окружения, используемого бинарным компилятором при исполнении x86-приложений.

> *Замечание. Запуск бинарного компилятора с явным указанием имени существующего x86-окружения, для которого значения параметров секции AdvEnv отличаются от рабочих значений запуска, расценивается как ошибка.*

Параметр BindHomeDir управляет доступностью содержимого домашней директории пользователя внутри x86-окружения во время исполнения x86-приложения бинарным компилятором. В качестве рабочего значения параметра BindHomeDir выбирается то, что указано в источнике с наивысшим приоритетом. При помощи опции менеджера запуска данный параметр может быть задан так: -b $HOME[41]. Если параметр имеет значение 0[42], то содержимое пользовательской домашней директории будет не доступно внутри x86-окружения. Если же параметр имеет значение 1, то содержимое домашней директории будет доступно во всех случаях кроме одного, когда источником определения параметра BindHomeDir выступает базовый конфигурационный файл и при этом параметр IgnoreDefaultBindings имеет значение 1. Значения параметра BindHomeDir, отличные от 0 и 1, являются некорректными. Важно, чтобы внутри корня гостевой ФС существовал путь, аналогичный пути до домашней директории внутри нативной ФС.

> *Замечание. Параметр BindHomeDir не рекомендуется использовать в случае создания x86-окружений для общего использования. Поскольку у каждого пользователя своя*

[41] эквивалентно определению параметра BindHomeDir в конфигурационном файле с указанием значения 1.

[42] значение по умолчанию.

домашняя директория, то будет использоваться не одно общее x86-окружение, а много разных.

Параметр BindPath управляет доступностью внутри x86-окружения файлов и директорий, расположенных вне корня гостевой ФС. В качестве значения для параметра BindPath указывается полный путь внутри нативной ФС до файла или директории, которые необходимо отобразить внутрь x86-окружения. Важно, чтобы внутри корня гостевой ФС существовал такой же путь[43], иначе бинарный компилятор завершит свою работу с ошибкой. Данный параметр может иметь большое количество определений, как в рамках разных источников, так и в рамках одного. Если параметр IgnoreDefaultBindings[44] имеет значение 0, то все определения параметра BindPath учитываются во время работы бинарного компилятора. Таким образом, все файлы и директории, что отмечены для отображения внутрь x86-окружения будут доступны x86-приложению. Если параметр IgnoreDefaultBindings имеет значение 1, то учитываются все определения параметра BindPath кроме тех, что указаны в базовом конфигурационном файле.

Базовый конфигурационный файл, создаваемый при установке бинарного компилятора, для отображения внутрь x86-окружения определяет директории /proc, /dev и /sys.

Параметры секции AdvEnv влияют на множество процессов бинарного компилятора, исполняющих x86-приложения в одном и том же x86-окружении.

[43] при этом также учитывается и тип файла: можно отображать только директории на директории или обыкновенные файлы на обыкновенные файлы, нельзя отображать символьные ссылки.

[44] см. его описание в секции BaseEnv.

Секция Params

Секция Params включает в себя два параметра: DefaultApp и LogPath. Описание секции Params может быть размещено в базовом и пользовательском конфигурационных файлах. Кроме того, оба параметра секции могут быть определены с помощью опций менеджера запуска. В качестве рабочего значения параметра, относящегося к секции Params, выбирается то, что указано в источнике с наивысшим приоритетом. Параметры секции Params, в отличие от параметров секций BaseEnv и AdvEnv, не обязаны распространять свое влияние на все множество процессов бинарного компилятора, исполняющих x86-приложения в одном и том же x86-окружении. Запуск бинарного компилятора с явным указанием имени существующего x86-окружения, для которого значения параметров секции Params отличаются от рабочих значений запуска, не расценивается как ошибка. Параметры, заданные через опцию, оказывают влияние лишь на запускаемое x86-приложение, а пользовательский конфигурационный файл, в котором описана только секция Params, позволит оказывать влияние на множество процессов бинарного компилятора, среди опций запуска которых указан этот конфигурационный файл.

Параметр DefaultApp используется бинарным компилятором, когда в строке его запуска не указано x86-приложение для исполнения. Данный параметр не имеет значения по умолчанию, при этом, в отличие от прочих подобных параметров, в конфигурационных файлах, создаваемых при установке бинарного компилятора, определение DefaultApp отсутствует.

> *Замечание. Отсутствие определения параметра DefaultApp во всех источниках не является ошибкой. В таком случае в строке запуска бинарного компилятора лишь необходимо явно указывать какое x86-приложение следует исполнить.*

В качестве значения параметра DefaultApp необходимо указывать путь внутри гостевой ФС от ее корня. Наиболее подходящим значением является путь до командного интерпретатора. Определение параметра DefaultApp имеет смысл делать при создании x86-окружения. В таком случае установленное значения будет влиять на все запуски бинарного компилятора в рамках этого x86-окружения[45]. Также определение параметра DefaultApp можно разместить в пользовательском конфигурационном файле. В таком случае установленное значение будет влиять на все запуски бинарного компилятора, использующие этот конфигурационный файл в качестве источника определений[46]. Значение параметра DefaultApp может быть задано при помощи опции –default_app.

Параметр LogPath определяет путь до места расположения лог-файлов бинарного компилятора. Данный параметр не имеет значения по умолчанию, поэтому в базовом конфигурационном файле для него явно устанавливается значение /var/log/bincomp – путь до директории, создаваемой при установке бинарного компилятора. Лог-файл бинарного компилятора формируется в случае нештатного завершения его работы или во время отладочного запуска и может быть полезен при разборе ошибок. Рекомендуется не изменять значение параметра LogPath, установленное в базовом конфигурационном файле. Альтернативное значение параметра LogPath может быть полезно лишь при получении в отладочных целях расширенных лог-файлов бинарного компилятора[47]. В таком случае значение параметра LogPath удобнее задавать при помощи опции –log.

Замечание. При использовании значения параметра LogPath, отличного от заданного в базовом конфигурационном файле, рекомендуется указывать путь до существующей директории с правами, разрешающими запись в эту директорию всем

[45] если не будет переопределено в рамках отдельных запусков.
[46] это могут быть запуски в рамках разных x86-окружений.
[47] см. описание секции Debug.

пользователям бинарного компилятора, для которых нестандартное значение параметра LogPath будет актуально. При нарушении этой рекомендации у бинарного компилятора могут возникнуть проблемы со сбросом лог-файла в случае нештатного завершения его работы.

Параметры группы Params не подразумевают столь массового влияния на запуски бинарного компилятора, как параметры секций BaseEnv, AdvEnv и, уж тем более, System.

Секция Debug

Секция Debug включает в себя семь параметров: DebugMode, X86Strace, X86CoreDump, PrintSignals, DisableL4, DisableParallel и Verbose. Описание секции может быть размещено в базовом и пользовательском конфигурационных файлах. Кроме того, каждый параметр секции Debug может быть определен и с помощью опции[48] менеджера запуска. В качестве рабочего значения параметра, относящегося к секции Debug, выбирается то, что указано в источнике с наивысшим приоритетом. Параметры данной секции предназначены для отладки бинарного компилятора, поэтому не рекомендуется ими пользоваться без острой необходимости.

Параметр DebugMode определяет, включен ли отладочный режим работы бинарного компилятора: при значении 1 – включен, а при значении 0[49] – выключен. Возможности отладочного режима указаны ниже при описании следующих шести параметров. Значения, отличные от 0 и 1, являются некорректными. При подаче хотя бы одной из отладочных опций

[48] параметр DebugMode не имеет собственной опции, но неявно определяется исходя из опций, соответствующих прочим параметрам секции Debug.

[49] значение по умолчанию.

–strace, –enable_coredump, –print_signals, –disable_l4, –disable_parallel, –verbose параметр DebugMode принимает значение 1.

Параметр X86Strace определяет необходимость выдачи в лог-файл бинарного компилятора отладочной информации о выполнении системных вызовов, источником которых выступает x86-приложение[50]. Значение параметра X86Strace учитывается бинарным компилятором лишь тогда, когда параметр DebugMode имеет значение 1. Отладочная информация о системных вызовах направляется в лог-файл, если и X86Strace, и DebugMode имеют значение 1. По умолчанию параметр X86Strace имеет значение 0. Значения, отличные от 0 и 1, являются некорректными. По опции –strace параметр X86Strace, а заодно и DebugMode, принимают значение 1.

Параметр X86CoreDump определяет необходимость сброса core-файла при завершении работы x86-приложения из-за сигнала, требующего формирования дампа регистров и памяти[51]. Структура core-файла соответствует исходной платформе и может быть, например, проанализирован с помощью x86-версии отладчика gdb. Значение параметра X86CoreDump учитывается бинарным компилятором лишь тогда, когда параметр DebugMode имеет значение 1. Сброс core-файла производится, если и X86CoreDump, и DebugMode имеют значение 1. По умолчанию параметр X86CoreDump имеет значение 0. Значения, отличные от 0 и 1, являются некорректными. По опции –enable_coredump параметр X86CoreDump, а заодно и DebugMode, принимают значение 1.

Параметр PrintSignals определяет необходимость выдачи в лог-файл бинарного компилятора отладочной информации о сигналах, доставленных исполняемому x86-приложению. Значение параметра PrintSignals учитывается бинарным компилятором лишь тогда, когда параметр DebugMode имеет значение 1. Отладочная информация о сигналах

[50] информация, подобная выдаче утилиты strace; не включает данные о системных вызовах, источником которых был сам бинарный компилятор.

[51] см. man 7 signal.

направляется в лог-файл, если и PrintSignals, и DebugMode имеют значение 1. По умолчанию параметр PrintSignals имеет значение 0. Значения, отличные от 0 и 1, являются некорректными. По опции –print_signals параметр PrintSignals, а заодно и DebugMode, принимают значение 1.

Параметр DisableL4 определяет, стоит ли бинарному компилятору во время своей работы отказаться от использования старшего регионного уровня трансляции[52]. Значение параметра DisableL4 учитывается бинарным компилятором лишь тогда, когда параметр DebugMode имеет значение 1. Старший регионный уровень трансляции отключается, если и DisableL4, и DebugMode имеют значение 1. По умолчанию параметр DisableL4 имеет значение 0. Значения, отличные от 0 и 1, являются некорректными. По опции –disable_l4 параметр DisableL4, а заодно и DebugMode, принимают значение 1.

Параметр DisableParallel определяет, стоит ли бинарному компилятору во время своей работы отказаться от использования механизма параллельной трансляции[53]. Значение параметра DisableParallel учитывается бинарным компилятором лишь тогда, когда параметр DebugMode имеет значение 1. Механизм параллельной трансляции отключается, если и DisableParallel, и DebugMode имеют значение 1. По умолчанию параметр DisableParallel имеет значение 0. Значения, отличные от 0 и 1, являются некорректными. По опции –disable_parallel параметр DisableParallel, а заодно и DebugMode, принимают значение 1.

Параметр Verbose определяет необходимость выдачи в лог-файл бинарного компилятора полезной информации об исполняемом x86-приложении, а также о подозрительных ситуациях, возникающих во время исполнения x86-приложения. Значение параметра Verbose учитывается бинарным компилятором лишь тогда, когда параметр DebugMode имеет значение 1. Дополнительная информация направляет в лог-файл, если и

[52] подробная информация приводится в описании параметра GlocalDisableL4 секции System.
[53] подробная информация приводится в описании параметра GlobalDisableParallel секции System.

Verbose, и DebugMode имеют значение 1. По умолчанию параметр Verbose имеет значение 0. Значения, отличные от 0 и 1, являются некорректными. По опции –verbose параметр Verbose, а заодно и DebugMode, принимают значение 1.

Пользователю предлагается включать отладочный режим исключительно при помощи опций менеджера запуска startx86. В таком случае в лог-файл будет сбрасываться дополнительная информация, касающаяся только того запуска бинарного компилятора, на котором проявляется требующая анализа проблема.

Таблица параметров

Параметр	**Секция**	**Связанная опция**	**Значение по умолчанию**
SuidGroupId	System	нет	0
GlobalDisableL4	System	нет	0
GlobalDisableParallel	System	нет	0
L3Threshold	System	нет	10000
L4Threshold	System	нет	1000000
Uid	BaseEnv	--uid <id>	RUID[54]
PathPrefix	BaseEnv	--path_prefix <e2k-path[55]>	нет[56]
Shared	BaseEnv	--shared	0
IgnoreDefaultBindings	BaseEnv	-B	0
BindHomeDir	AdvEnv	-b $HOME	0
BindPath	AdvEnv	-b <e2k-path>	нет[57]
DefaultApp	Params	--default_app <x86-path[58]>	нет
LogPath	Params	--log <e2k-path>	нет[59]
DebugMode	Debug	все отладочные опции[60]	0
X86Strace	Debug	--strace	0
X86CoreDump	Debug	--enable_coredump	0
PrintSignals	Debug	--print_signals	0
DisableL4	Debug	--disable_l4	0
DisableParallel	Debug	--disable_parallel	0
Verbose	Debug	--verbose	0

[54] реальный идентификатор пользователя.
[55] путь внутри нативной ФС.
[56] в базовом конфигурационном файле указывается значение /.
[57] в базовом конфигурационном файле указываются значения /proc, /sys, /dev.
[58] путь внутри гостевой ФС.
[59] в базовом конфигурационном файле указывается значение /var/log/bincomp.
[60] опции –strace, --enable_coredump, --print_signals, --disable_l4, --disable_parallel, --verboase.

Конфигурационные файлы

Конфигурационный файл — текстовый файл, содержащий определения параметров бинарного компилятора и имеющий определенную структуру. Существует три типа конфигурационных файлов: системный, базовый и пользовательский.

Системный конфигурационный файл создается во время установки бинарного компилятора и помещается по пути /etc/bincomp/system.conf. В нем допускается определение параметров, относящихся к секции System. Изменение содержимого данного конфигурационного файла по умолчанию доступно только системному администратору, но даже привилегированному пользователю рекомендуется отказаться от правок файла /etc/bincomp/system.conf.

Базовый конфигурационный файл создается во время установки бинарного компилятора и помещается по пути /etc/bincomp/default.conf. В нем допускается определение параметров, относящихся к секциям BaseEnv, AdvEnv, Params и Debug[61]. Изменение содержимого данного конфигурационного файла по умолчанию доступно только системному администратору. Привилегированный пользователь может модифицировать определения параметров, относящихся к секциям BaseEnv, AdvEnv и Params, чтобы сформировать более подходящее базовое x86-окружение. Описание секции Debug изменять не рекомендуется.

Сразу после установки бинарного компилятора данные, указанные в системном и базовом конфигурационных файлах, позволяют производить запуск статически связанных x86-приложений. Системный администратор может развернуть гостевую файловую систему[62] и указать в базовом конфигурационном файле в качестве значения для параметра PathPrefix

[61] по умолчанию параметры секции Debug имеют значение 0, рекомендуется изменять их при помощи опций менеджера запуска startx86.

[62] см. раздел «Создание гостевой файловой системы».

строку, содержащую путь до корня гостевой ФС. В таком случае появится возможность запуска динамически связанных x86-приложений.

Пользовательский конфигурационный файл создается пользователем операционной системы по любому доступному ему пути. В таком конфигурационном файле можно размещать определения параметров, относящихся к секциям BaseEnv, AdvEnv, Params и Debug[63]. Пользовательский конфигурационный файл предназначается для создания собственных x86-окружений[64], либо x86-окружений, отличающихся незначительно[65] от описанного в базовом конфигурационном файле.

> *Замечание. Важно помнить, что параметр бинарного компилятора, относящийся к секции BaseEnv, AdvEnv, Params или Debug и не имеющий определений через опцию менеджера запуска и в пользовательском конфигурационном файле, учитывает определение, указанное в базовом конфигурационном файле[66].*

Во время установки бинарного компилятора создается специальная директория /var/lib/bincomp/enabled. По размещенным в ней пользовательским конфигурационным файлам служба bincomp[67] при старте создает x86-окружения, в рамках которых можно производить запуски x86-приложений[68].

Все конфигурационные файлы состоят из одной или нескольких секций. Описание секции начинается с указания ее имени в квадратных скобках[69] и заканчивается либо началом новой секции, либо концом конфигурационного

---

[63] при описании секции Debug в пользовательском конфигурационном файле рекомендуется для параметров указывать значения 0, а для целей отладки пользоваться опциями менеджера запуска.

[64] x86-окружений с собственным корнем гостевой ФС; с собственным набором файлов, отображенных из нативной ФС в гостевую; с подходящим типом.

[65] значениями параметров секций Params и Debug.

[66] если оно там есть, в противном случае учитывается значение, определенное в исходных кодах бинарного компилятора.

[67] см. раздел «Установка бинарного компилятора».

[68] подробнее об этом написано в разделе «Запуск бинарного компилятора».

[69] [System], [BaseEnv] и т. п.

файла. Не обязательно включать в конфигурационный файл описание всех доступных ему секций[70], при этом описание секций в неподходящем конфигурационном файле[71] является ошибкой. Также ошибкой является описание пустой секции — секции без определения хотя бы одного из ее параметров. Каждый параметр может быть определен только в рамках своей секции[72]. Определение параметра в другой секции или вне какой-либо секции является ошибкой конфигурационного файла. Строка с определением параметра включает в себя его имя и значение, разделенные пробелами или табуляциями. Помимо секций и параметров конфигурационные файлы могут содержать строки с комментариями, начинающиеся с символа #. Создание пользовательских конфигурационных файлов рекомендуется проводить на базе /usr/share/bincomp/examples/default.conf.example.

[70] например, можно не описывать секцию Params в пользовательском конфигурационном файле.
[71] например, описание секции System в базовом конфигурационном файле.
[72] см. таблицу параметров в конце раздела «Параметры бинарного компилятора».

Создание гостевой файловой системы

Полноценный режим работы бинарного компилятора требует наличия гостевой ФС, которую можно сформировать несколькими способами. Во-первых[73], можно подключить к вычислительной машине с ОС «Эльбрус» носитель с установленной на нем x86 операционной системой семейства linux и смонтировать необходимые разделы[74], присутствующие на носителе, в корень гостевой ФС[75]. Во-вторых, можно на базе файлов и директорий, расположенных в корневой директории[76] запущенной x86 операционной системы семейства linux, создать так называемый *образ гостевой файловой системы* (*образ гостевой ФС*) и перенести его в нативную ФС. Второй вариант не предполагает монтирования в среде ОС «Эльбрус» и потому подходит для непривилегированных пользователей, однако он требует наличия вычислительной машины, на которой может быть запущена x86 операционная система семейства linux с возможностью работы в ней от имени системного администратора. В состав бинарного компилятора включены скрипты pack_guestfs.sh и unpack_guestfs.sh[77], позволяющие соответственно создать и развернуть образ[78] гостевой ФС.

> *Замечание. Использование инструмента debootstrap и его аналогов так же может быть отмечено в качестве удобного способа создания гостевых файловых систем. В данном руководстве не дается инструкций по использованию debootstrap, т. к. предполагается, что пользователь сможет самостоятельно найти всю необходимую информацию в сети Интернет.*

[73] для этого способа, вероятно, потребуются права системного администратора.

[74] пользователь самостоятельно должен определить перечень разделов и точки их монтирования внутри корня гостевой ФС; как минимум необходимо смонтировать основной (корневой) раздел.

[75] расположение корня гостевой ФС выбирает пользователь.

[76] и ее поддиректориях.

[77] см. раздел «Расположение файлов бинарного компилятора».

[78] образ гостевой файловой системы представляет из себя tgz-архив.

Создание образа гостевой файловой

Создание образа гостевой ФС скриптом pack_guestfs.sh[79] выполняется системным администратором в среде работающей x86 операционной системы семейства linux[80], запущенной либо на вычислительной машине, построенной на базе процессоров фирм Intel/AMD, либо на вычислительной машине, построенной на базе процессора семейства «Эльбрус», с привлечением бинарного компилятора уровня системы (Lintel). По умолчанию скрипт включает в формируемый образ гостевой ФС лишь содержимое файловой системы, смонтированной в корневую директорию, добавить содержимое файловых систем с другими точками монтирования можно, используя опцию –include[81]. Директории /dev, /proc и /sys являются особенными и в образ не включаются никогда[82]. При помощи опции –exclude можно указать те файлы и директории, которые не стоит включать в образ гостевой ФС. По опции –verbose скрипт pack_guestfs.sh выдаст более детальную информацию о своей работе.

> *Замечание. Перед запуском скрипта pack_guestfs.sh необходимо убедиться в наличии свободного дискового пространства в количестве достаточном для создания образа гостевой ФС. Данное действие возлагается на запускающего скрипт pack_guestfs.sh.*

В случае успешного формирования образа гостевой файловой системы скрипт pack_guestfs.sh укажет имя созданного архива.

---

[79] пользователь должен самостоятельно обеспечить доступность скрипта в среде x86 операционной системы.

[80] чтобы узнать имя созданного образа гостевой файловой системы, запуск скрипта pack_guestfs.sh рекомендуется выполнять в командной строке.

[81] допускается подача нескольких опций –include.

[82] даже по опции –include.

Распаковка образа гостевой файловой системы

Распаковка образа гостевой ФС скриптом unpack_guestfs.sh с формированием гостевой ФС выполняется пользователем в среде ОС «Эльбрус»[83]. Путь до образа гостевой ФС передается скрипту в качестве обязательного аргумента. По умолчанию гостевая ФС размещается в текущей директории. Чтобы указать другое место, необходимо воспользоваться опцией скрипта –root[84]. Если скрипту unpack_guestfs.sh при запуске подать опцию –delete, то он удалит образ гостевой ФС, после его успешной распаковки. По опции –verbose скрипт выдаст более детальную информацию о своей работе.

> *Замечание. Распаковку архива гостевой файловой системы допускается производить из-под непривилегированного пользователя. В таком случае рекомендуется размещать корень гостевой файловой системы внутри домашней директории.*

[83] запуск скрипта рекомендуется выполнять в командной строке.
[84] указав после –root путь до корня гостевой ФС.

Менеджер запуска startx86[85]

Приложение startx86 используется для запуска бинарного компилятора и работы с x86-окружениями. Действие, выполняемое менеджером startx86, определяется наличием соответствующей специальной опции в строке его запуска. Полный список возможностей включает в себя следующие действия:

- создание x86-окружения (опция –create);
- удаление x86-окружения (опция –destroy);
- вывод списка существующих x86-окружений (опции –show, –show-detailed);
- запуск бинарного компилятор с использованием существующего x86-окружения (опция –join);
- запуск бинарного компилятора с созданием при необходимости нового или использованием существующего x86-окружения (опция –run)[86].

Помимо шести приведенных выше опций, в строке запуска startx86 также могут быть указаны:

- имя x86-окружения (обязательно при использовании –destroy/–join);
- путь до пользовательского конфигурационного файла (при помощи опции –conf, допустимо только при использовании –create/–run);
- опции, определяющие значения параметров бинарного компилятора[87];
- x86-приложение для исполнения с его аргументами, обязательно следующее за специальным разделителем –[88] (допустимо только при использовании –join/–run).

[85] см. раздел «Расположение файлов бинарного компилятора».

[86] является действием по умолчанию, если в строке запуска start не указана ни одна из опций –create/–destroy/–show/–show-detailed/–join/–run.

[87] см. таблицу параметров в конце раздела «Параметры бинарного компилятора».

[88] два знака минус, записанных подряд.

Не все комбинации опций и параметров являются корректными или осмысленными, поэтому в данном разделе не приводится строка запуска startx86 в общем виде. Рекомендуется ознакомиться с особенностями запуска startx86 с учетом запрашиваемого действия. Далее в рамках этого раздела описывается работа с x86-окружениями, а особенности запуска бинарного компилятора рассматриваются в следующем.

> *Замечание. Указание в строке запуска startx86 нескольких опций, соответствующих разным действиям, недопустимо и является ошибкой.*

Создание x86-окружения

В общем виде строка запуска startx86 при создании x86-окружения выглядит следующим образом:

$ startx86 --create [<name>] [--conf <user.conf>] [env-opts] [params]

Ключевой в этой строке является опция –create, определяющая действие, выполняемое менеджером запуска. Значения параметров создаваемого x86-окружения[89] определяются на основании опций [env-opts][90], [params][91] и конфигурационных файлов. Учитывается и пользовательский <user.conf>, если он задан. Созданному x86-окружению присваивается имя, указанное в строке запуска startx86 явно (<name>), либо вычисленное исходя из пути до корня гостевой ФС. Если x86-окружение с указанным именем, типом и владельцем уже существует, то запуск startx86 завершится с ошибкой. Не допускается создание x86-окружения, для которого параметр PathPrefix имеет значение /[92].

[89] в том числе его тип и владельца.
[90] опции, соответствующие параметрам бинарного компилятора секций BaseEnv и AdvEnv.
[91] опции, соответствующие параметрам бинарного компилятора секции Params.
[92] означает отсутствие гостевой ФС.

Созданное однажды x86-окружение может быть впоследствии использовано многократно при исполнении различных x86-приложений. Для этого запуск бинарного компилятора необходимо производить при помощи startx86 с опцией –join[93] и указанием имени этого x86-окружения. Если пользователь планирует при помощи бинарного компилятора запускать несколько x86-приложений, которые допускают взаимодействие между собой, то их правильнее исполнять в рамках одного x86-окружения[94]. Таким образом, например, можно гарантировать, что список файлов и директорий нативной ФС, доступных внутри гостевой, для взаимодействующих x86-приложений будет одинаковым.

> *Замечание. Системная служба bincomp при старте осуществляет создание x86-окружений, пользовательские конфигурационные файлы для которых размещаются в специальной директории /var/lib/bincomp/enabled[95], доступной членам группы bincomp. Параметры создаваемых x86-окружений определяются исключительно на основании конфигурационных файлов и значений, указанных в исходных кодах бинарного компилятора. В качестве имени создаваемого x86-окружения используется имя пользовательского конфигурационного файла[96].*

Если предполагается использовать одно фиксированное[97] x86-окружение, то от его явного создания[98] можно отказаться в пользу запусков startx86 с опцией –run[99]. В таком случае при первом запуске x86-окружение будет создано[100], а при последующих оно будет использоваться автоматически.

---

[93] см. раздел «Запуск бинарного компилятора».

[94] не распространяется на случай, когда параметр PathPrefix имеет значение /.

[95] конфигурационные файлы должны иметь расширение conf.

[96] без расширения conf.

[97] не предполагается от запуска к запуску изменять значения параметров, относящихся к секциям BaseEnv и AdvEnv.

[98] при помощи запуска startx86 с опцией –create.

[99] см. раздел «Запуск бинарного компилятор».

[100] и запускаемое x86-приложение сразу после создания x86-окружения начнет исполняться бинарным компилятором.

В рамках одной гостевой ФС рекомендуется создавать не более одного x86-окружения для личного пользования, это позволит избежать ненужной путаницы с особенностями отображения файлов из нативной ФС в гостевую. Если существующее x86-окружение требует модификации[101], то для начала его рекомендуется удалить[102], а уж потом создавать новое.

> *Замечание. Удаление x86-окружения не окажет негативного влияния на запущенные ранее в его рамках процессы. С их точки зрения x86-окружение продолжит существовать в старом виде. Тем не менее, перед удалением x86-окружения все же рекомендуется завершить связанные с ним запуски бинарного компилятора.*

Удаление x86-окружения

В общем виде строка запуска startx86 при удалении x86-окружения выглядит следующим образом:

$ startx86 --destroy <name> [--shared] [--uid]

Ключевой является опция –destroy, определяющая действие, выполняемое менеджером запуска. При удалении обязательно указывается имя x86-окружения (<name>). Опцию –shared необходимо подавать при удалении x86-окружения, предназначенного для общего использования, а опция –uid позволяет удалять либо созданные другим пользователем x86-окружения[103], либо все свои x86-окружения. Существует два специальных имени: :user и :all. Первое позволяет удалить все x86-окружения пользователя, запускающего startx86[104], второе — удалить x86-окружения всех

[101] расширения списка файлов нативной ФС, отображаемых в гостевую ФС.
[102] см. «Удаление x86-окружения» ниже.
[103] доступно только системному администратору.
[104] если опция –shared подана, то будут удалены все x86-окружения, предназначенные для общего использования, в противном случае вообще все x86-окружения.

пользователей[105]. Использовать специальное имя :all может только системный администратор.

Удаление x86-окружения не окажет негативного влияния на запущенные ранее в его рамках процессы. С их точки зрения x86-окружение продолжит существовать, но использовать его для новых запусков бинарного компилятора более не получится. Тем не менее, перед удалением x86-окружения все же рекомендуется завершить связанные с ним запуски бинарного компилятора[106].

Явное удаление x86-окружения имеет смысл только при формировании нового x86-окружения в рамках той же гостевой ФС. Это гарантирует исполнение всех x86-приложений в едином x86-окружении.

Замечание. Системная служба bincomp при завершении работы осуществляет удаление всех x86-окружений.

Вывод списка существующих x86-окружений

В общем виде строка запуска startx86 для получения списка существующих x86-окружений выглядит следующим образом:

$ startx86 --show [<name>] [--shared] [--uid]

Ключевой является опция –show, определяющая действие, выполняемое менеджером запуска. Параметр <name> и опции –shared, –uid, если они заданы, выступают в качестве фильтров для выводимой информации. Из всех существующих x86-окружений менеджер запуска приведет данные лишь для тех, что имеют указанные имя, тип[107] и владельца.

[105] если опция –shared подана, то будут удалены все x86-окружения, предназначенные для общего использования, в противном случае вообще все x86-окружения.

[106] пользователь должен самостоятельно определить эти процессы.

[107] если опция –shared подана, то выводится информация только по x86-окружениям, созданным для общего использования, если опция не подана - то по всем x86-окружениям.

Для получения расширенной информации о существующих x86-окружениях можно воспользоваться опцией –show-detailed:

```
$ startx86 --show-detailed [name] [--shared] [--uid]
```

Опция –show-detailed работает схожим с опцией –show образом, но в ее выдаче также присутствует информация о списке файлов и директорий, отображенных из нативной ФС в гостевую, и путь до директории, в которой бинарный компилятор располагает свои лог-файлы. Эти данные могут пригодиться при возникновении подозрений на некорректную работу бинарного компилятора.

Запуск бинарного компилятора

Запуск бинарного компилятора пользователем всегда осуществляется при помощи менеджера startx86[108], что гарантирует исполнение любого x86-приложения в некотором x86-окружении. Даже в случае прямого запуска x86-приложения ядро ОС «Эльбрус» благодаря информации, содержащейся в файле /proc/bincomp/search_path, находит менеджер запуска и использует его неявным образом.

Если исполнение x86-приложения необходимо осуществить в рамках ранее созданного x86-окружения, то запуск менеджера startx86 следует производить с опцией –join. В таком случае строка запуска имеет следующий общий вид:

$ startx86 --join <name> [--shared] [--uid] [--log <path>] [dbg-opts] [-- x86 args]

Ключевой в этой строке является опция –join, определяющая действие, выполняемое менеджером запуска. После этой опции обязательно указывается имя существующего x86-окружения, в рамках которого должен быть произведен запуск бинарного компилятора. Среди x86-окружений, имеющих одно и то же имя[109], выбрать конкретное можно при помощи опций –shared и –uid[110]. Использовать опции –log и [dbg-opts][111] рекомендуется только в отладочных запусках[112]. После специального разделителя –[113] можно указать путь до x86-приложения[114], которое необходимо исполнить, с его параметрами. Если в строке запуска startx86 не указать x86-приложение явно, то бинарный компилятор запустит x86-приложение по умолчанию, связанное с указанным x86-окружением, если оно задано[115], в противном случае завершит свою работу с ошибкой.

[108] см. раздел «Расположение файлов бинарного компилятора».
[109] а такое допускается, см. раздел «X86-окружение бинарного компилятора».
[110] опция –uid доступна только системному администратору.
[111] опции, соответствующие параметрам бинарного компилятора секции Debug.
[112] см. раздел «Ошибки бинарного компилятора».
[113] два знака минус, идущие подряд.
[114] относительно корня гостевой ФС.
[115] при помощи параметра DefaultApp.

Замечание. Использование опции –join может быть полезно в тех случаях, когда ответственность за создание x86-окружений возлагается на системную службу bincomp[116].

Если при старте x86-приложения достоверно не известно, существует ли подходящее x86-окружение, то запуск менеджера startx86 рекомендуется проводить, используя опцию –run[117]. В таком случае строка запуска имеет следующий общий вид:

$ startx86 --run [<name>] [--conf <user.conf>] [env-opts] [params] [dbg-opts] [-- x86 args]

Ключевой в этой строке является опция –run, определяющая действие, выполняемое менеджером запуска. После этой опции могут быть указаны имя x86-окружения (<name>)[118], путь до пользовательского конфигурационного файла (–conf <user.conf>), опции [env-opts][119], [params][120], [dbg-opts][121], а также запускаемое x86-приложение с его параметрами.

При запуске startx86 с опцией –run бинарный компилятор выполняет определенную последовательность действий. Сначала он пытается найти существующее x86-окружение, основываясь на имени[122], учитывая также наличие опций –shared и –uid[123]. Если поиск оказывается успешным, то проверяется, что найденное x86-окружение совпадает с x86-окружением, которое было бы создано, исходя из опций запуска[124], в том числе учитывается и пользовательский конфигурационный файл, если он задан по опции –conf. В случае несовпадения x86-окружений бинарный компилятор завершает работу с ошибкой. Если подходящего x86-окружения найти не

[116] см. раздел «Расположение файлов бинарного компилятора».

[117] либо не использовать никакую из опций –create/–destroy/–show/–show-detailed/–join/–run, что эквивалентно использованию опции –run; такое допущение делает возможным прямой запуск x86-приложений.

[118] существующего или того, которое будет создано в рамках этого запуска бинарного компилятора.

[119] опции, соответствующие параметрам бинарного компилятора секций BaseEnv и AdvEnv.

[120] опции, соответствующие параметрам бинарного компилятора секции Params.

[121] опции, соответствующие параметрам бинарного компилятора секции Debug.

[122] имя может быть указано явно (<name>), либо сформировано менеджером startx86 на основании пути до корня гостевой ФС (совпадает с рабочим значением параметра PathPrefix с учетом указанных в строке запуска опций и конфигурационных файлов).

[123] в строке запуска они входят в группу [env-opts].

[124] как будто в строке запуска вместо опции –run была указана опция –create.

удается, бинарный компилятор создает новое[125]. После того как подходящее x86-окружение было обнаружено или создано, осуществляется непосредственный запуск бинарного компилятора[126].

Замечание. Указание в строке запуска бинарного компилятора опций –run и –shared[127], явного имени для x86-окружения, а также пользовательского конфигурационного файла, если не достаточно базового, позволяет гарантировать исполнение x86-приложения в нужном x86-окружении (новом или ранее созданном). Такой вариант запуска бинарного компилятора является наиболее предпочтительным.

[125] как будто в строке запуска вместо опции –run была указана опция –create.
[126] как будто в строке запуска вместо опции –run была указана опция –join.
[127] указывается только в случае работы с x86-окружениями для общего использования.

Известные особенности работы бинарного компилятора

Особенность #1. Запускаемое x86-приложение должно использовать библиотеку libc[128], совместимую с версией ядра Linux, работающего на целевой платформе. Это требование связано с тем, что бинарный компилятор при работе адаптирует системные вызовы x86-приложения в системные вызовы ядра ОС «Эльбрус», то есть он закладывается на определенные программные интерфейсы.

Особенность #2. Бинарный компилятор не поддерживает системный вызов ptrace. Таким образом, под бинарным компилятором могут не работать программы-отладчики.

Особенность #3. При работе с аппаратурой не допускается прямых (через порты ввода-вывода или память) обращений к устройствам, допускается работа только через системный вызов ioctl. Данная особенность связана с принципом работы бинарного компилятора – адаптацией лишь системных вызовов исходной платформы. Задача по работе с аппаратурой напрямую, которая в Linux возлагается на ядро (модули ядра), бинарным компилятором не решается.

Особенность #4. В бинарном компиляторе имеется поддержка для ограниченного набора ioctl-запросов. Дело в том, что для большинства запросов в качестве одного из аргументов передается указатель на некоторую специфическую структуру, которую зачастую бинарному компилятору необходимо преобразовывать в вид, понятный ядру целевой платформы. Подобное преобразование требует тщательной проверки, но она не возможна без подходящей аппаратуры. Список поддерживаемых ioctl-запросов может быть расширен исходя из пожеланий пользователя.

[128] если она ему нужна.

Особенность #5. При исполнении x86-приложения бинарным компилятором запускается три[129] дополнительных процесса, которые не видны внутри x86-окружения. В нативном окружении эта тройка выглядит как два дочерних процесса с именами rtc compiler #0, rtc compiler #1 и их родитель[130]. Основной задачей этих процессов является обслуживание механизма параллельной трансляции[131].

Особенность #6. Текущая реализация бинарного компилятора не поддерживает исполнение x86-приложений, которые во время своей работы устанавливают флаг ловушки (флаг TF регистра флагов исходной платформы).

[129] если при запуске бинарного компилятора хотя бы один из параметров GlobalDisableParallel или DisableParallel имеет значение 1, то дополнительные процессы не создаются.

[130] не имеет специфического имени; совпадает с именем исполняемого x86-приложения.

[131] подробнее об этом см. замечание в описании секции System в разделе «Параметры бинарного компилятора».

Ошибки бинарного компилятора

Ошибки бинарного компилятора способны проявляться по-разному. Это может быть как падение самого бинарного компилятора, так и нарушение логики работы x86-приложения, вплоть до его аварийного завершения.

Не всегда неправильная работа x86-приложения является следствием ошибок бинарного компилятора. Иногда это связано с не полностью выполненной настройкой гостевой файловой системы. В таком случае может потребоваться коррекция списка файлов и директорий, отображаемых из нативной ФС в гостевую[132]. Иногда причина кроется в тех ограничениях, которые приведены в разделе «Известные особенности работы бинарного компилятора». Наконец, нельзя исключать и ошибку в самом приложении, такие ситуации также наблюдались.

В случае нештатного завершения работы бинарный компилятор создает аварийный лог-файл с информацией, которой в большинстве случаев достаточно для установления причин сбоя. Данный лог-файл размещается в директории[133], путь до которой передается бинарному компилятору через параметр LogPath. Наличие в этой директории файлов с ненулевым размером может свидетельствовать о случаях нештатного завершения работы бинарного компилятора или о запусках бинарного компилятора в отладочном режиме.

> *Замечание. Чтобы было проще понять причину существования непустого лог-файла, рекомендуется для параметра LogPath использовать определение из базового конфигурационного файла, подменяя его при помощи опции –log менеджера запуска лишь при отладке. При таком подходе*

[132] например, содержимого директорий /etc, /var или /lib/modules.

[133] если пользователь в соответствии с рекомендациями, озвученными в замечании при описании секции Params в разделе «Параметры бинарного компилятора», использовал в качестве значения для параметра LogPath путь до существующей директории.

наличие непустого лог-файла в директории /var/log/bincomp будет однозначно свидетельствовать об ошибках бинарного компилятора.

Лог-файл также может быть получен и для запуска, завершающегося с точки зрения бинарного компилятора штатным образом. Это полезно в тех случаях, когда исполнение x86-приложения со стороны пользователя выглядит некорректным[134], а бинарный компилятор в своей работе никаких ошибок не обнаруживает. В большинстве случаев для получения лог-файла с полезной информацией будет достаточно исполнить x86-приложение с использованием менеджера запуска, которому дополнительно поданы следующие опции: –verbose –strace –print_signals[135]. Полученный таким образом диагностический лог-файл может быть полезен при разборе предполагаемой ошибки бинарного компилятора его разработчиками.

При возникновении подозрения на ошибку в работе бинарного компилятора пользователю, прежде всего, следует проверить содержимое директории /var/log/bincomp[136], используемой для размещения лог-файлов. Если в ней присутствуют недавно созданные непустые файлы, то в работе бинарного компилятора действительно проявилась какая-то ошибка. Если подходящих файлов нет, то стоит произвести запуск x86-приложения, исполнение которого кажется неправильным, в отладочном режиме[137] с формированием диагностического лог-файла.

Сообщения об ошибках следует направлять на электронный адрес службы поддержки support@mcst.ru вместе с лог-файлами для анализа проблемы. В том случае, если по переданным материалам установить причину проблемы не удастся, может возникнуть необходимость[138] в

[134] ситуация должна быть воспроизводимой, при этом она не обязана проявляться на каждом запуске x86-приложения.
[135] также рекомендуется использовать опцию –log (см. замечание этого раздела).
[136] или той, что задавалась пр помощи нестандартного значения параметра LogPath.
[137] см. предыдущий абзац.
[138] актуально только в случае воспроизводимых проблем.

предоставлении в службу поддержки программно-аппаратной конфигурации вычислительной машины для более глубокого анализа ситуации разработчиками бинарного компилятора.