Eltechs ExaGear Desktop

User Guide

Contents

1 Introduction

Eltechs ExaGear Desktop enables ARM-based devices with ARM GNU/Linux operating system to run practically any kind of x86 32-bit GNU/Linux software designed for standard desktops. Moreover, you can run Windows applications using Wine on top of ExaGear.

This guide describes how to install, configure and work with ExaGear along with conceptual view of how it works.

In this document the term **host system** is used to refer to ARM-based GNU/Linux and the term **guest system** is used to refer to the x86 GNU/Linux running on top of the host system.

Note that following formatting conventions are used in examples in this document:

- `monospace` identifies commands to be performed in the host system
- **`monospace bold`** identifies commands to be performed in the guest system
- <version> identifies variable version with different values.

Following guest system images are currently available: Ubuntu 14.04, Ubuntu 15.04, Ubuntu 16.04, Debian-7, Debian 8. Note that the guest system images have to correspond to the host system version. Thus, we strongly recommend following configuration:

- host system is Ubuntu 14 and guest system is Ubuntu 14.04
- host system is Ubuntu 15 and guest system is Ubuntu 15.04
- host system is Ubuntu 16 and guest system is Ubuntu 16.04
- host system is Debian 7 and guest system is Debian 7
- host system is Debian 8 and guest system is Debian 8
- host system is Raspbian Wheezy and guest system is Debian 7
- host system is Raspbian Jessie and guest system is Debian 8.

Support for other GNU/Linux distributions is in development.

2 ExaGear for Ubuntu, Debian, Raspbian

This chapter describes procedures of installation, configuration of ExaGear for Ubuntu, Debian or Raspbian host system, installation and running of x86 applications within it.

2.1 Prerequisites

Note that the host system should meet following requirements:

- CPU:
  - ARMv6 architecture for Raspberry Pi 1 and Raspberry Pi Zero
  - ARMv7 or ARMv8 architecture for other ARM devices
  - support of VFPv3-D32
  - for x86 applications that use MMX/SSE support of NEON is also required
- RAM: 512MB or more

- Disk Space: 1GB (including basic x86 system that is provided as part of ExaGear installation packages). Note that extra space is required for installation of x86 applications.
- OS: ARM Linux Ubuntu 14,15,16, Debian 7,8, Raspbian Wheezy and Jessie. Additionally binfmt_misc kernel module is required.

Note that currently ExaGear only supports 32-bit x86 applications.

2.2 Installing ExaGear

ExaGear distribution consists of two packages that contain a virtualization engine with support utilities and a set of x86 binary files, libs and data, that form minimal x86 environment. Also installation script is provided for automated installation. ExaGear requires activation using a license key.

Prior to the installation check if `/opt/exagear` directory of the host system doesn't exist. In case the directory exists you should follow the instructions in chapter "2.8 Update ExaGear".

Automated installation

In order to install and activate ExaGear run `install-exagear.sh` script in the directory with exagear .deb packages and one license key:

```
$ ls -l
exagear-guest-<package_version>.deb
exagear_<package_version>.deb
install-exagear.sh
pk_<num>.key
$ sudo ./install-exagear.sh
```

This script installs guest x86 image that better fits your host operating system. Sometimes you might need to install guest x86 Ubuntu image, for example Spotify can run with the guest Ubuntu only. To force install the guest x86 Ubuntu run installation script with following option:

```
$ sudo ./install-exagear.sh ubuntu-1404
```

Manual installation

To proceed with installation perform following actions:

1. Ensure that you have the necessary privileges needed for package installation.
2. Ensure ExaGear dependencies:

```
$ sudo apt-get install bash coreutils findutils realpath curl binfmt-support cron
```

3. Identify versions of ExaGear packages best suitable for your host system.
    a. Check the architecture of your ARM device:

```
$ arch
```

    b. Check user/kernel virtual memory split of the host system (execute the hole line in command line terminal and see the output):

```
[ $((`cat /proc/self/maps | grep stack | awk -F'-' '{print "0x"$1}'`)) -gt
$((0x80000000)) ] && echo 3g/1g || echo 2g/2g
```

c. Check the version of the host operating system:

```
$ sudo cat /etc/issue
```

d. Find versions of ExaGear packages to install in the table below:

Arch	User/ kernel split	Operating system	ExaGear packages to install
aarch64	3g/1g	Debian Jessie 8	`exagear-armv8_*.deb` `exagear-guest-debian-8_*.deb`
aarch64	3g/1g	Ubuntu 15.xx	`exagear-armv8_*.deb` `exagear-guest-ubuntu-1504_*.deb`
aarch64	3g/1g	Ubuntu 16.xx	`exagear-armv8_*.deb` `exagear-guest-ubuntu-1604_*.deb`
armv7	2g/2g	Raspbian Wheezy 7	`exagear-mem2g_*.deb` `exagear-guest-debian-7-wine2g_*.deb`
armv7	2g/2g	Raspbian Jessie 8	`exagear-mem2g_*.deb` `exagear-guest-debian-8_*.deb`
armv7	2g/2g	Ubuntu 14.xx	`exagear-mem2g_*.deb` `exagear-guest-ubuntu-1404lts_*.deb`
armv7	2g/2g	Ubuntu 15.xx	`exagear-mem2g_*.deb` `exagear-guest-ubuntu-1504_*.deb`
armv7	2g/2g	Ubuntu 16.xx	`exagear-mem2g_*.deb` `exagear-guest-ubuntu-1604_*.deb`
armv7	3g/1g	Debian Wheezy 7	`exagear_*.deb` `exagear-guest-debian-7_*.deb`
armv7	3g/1g	Debian Jessie 8	`exagear_*.deb` `exagear-guest-debian-8_*.deb`
armv7	3g/1g	Ubuntu 14.xx	`exagear_*.deb` `exagear-guest-ubuntu-1404lts_*.deb`
armv7	3g/1g	Ubuntu 15.xx	`exagear_*.deb` `exagear-guest-ubuntu-1504_*.deb`
armv7	3g/1g	Ubuntu	`exagear_*.deb`

		16.xx	exagear-guest-ubuntu-1604_*.deb
armv6	2g/2g	Raspbian Wheezy 7	exagear-legacy-mem2g_*.deb exagear-guest-debian-7-wine2g_*.deb
armv6	2g/2g	Raspbian Jessie 8	exagear-legacy-mem2g_*.deb exagear-guest-debian-8_*.deb
armv6	3g/1g	Raspbian Wheezy 7	exagear-legacy_*.deb exagear-guest-debian-7_*.deb
armv6	3g/1g	Raspbian Jessie 8	exagear-legacy_*.deb exagear-guest-debian-8_*.deb

4. Use `dpkg` to install ExaGear packages. For example:

```
$ sudo dpkg –i exagear-mem2g_*.deb
$ sudo dpkg –i exagear-guest-debian-8_*.deb
```

5. Use license key to activate ExaGear

```
$ sudo cp pk*.key /opt/exagear/lic
$ sudo /opt/exagear/bin/actool
```

Now the x86 world is created and you can start running the guest system.

2.3 Running the Guest System

You can print the guest system version by using `exagear-ls` command:

```
user@arm-server:$ exagear-ls
Installed guest images:
debian-8
```

To start a guest shell use `exagear` utility. For example:

```
user@arm-server:$ exagear
Starting the shell in the guest image /opt/exagear/images/debian-8
user@arm-server@x86:$ uname -a
Linux arm-server 4.1.19-v7+ #858 SMP Tue Mar 10 10:02:00 GMT 2016 i686
GNU/Linux
user@arm-server@x86:$
```

You are now inside the x86 world. Here your root (“/”) directory is `/opt/exagear/images/<guest_system_version>` (similar to the result of `chroot /opt/exagear/images/<guest_system_version>`). You can use this guest shell as if it were running on an x86 machine.

Note that user accounts are shared between the host and guest systems. When a user is added or removed in the guest system, the modification is made to the host. `/home` directories are

shared between the host and guest systems as well.

To exit from the guest session use `exit`:

```
user@arm-server@x86:$ exit
user@arm-server:$
```

Now you are back in the host system:

```
user@arm-server:$ uname -a
Linux arm-server 4.1.19-v7+ #858 SMP Tue Mar 10 10:02:00 GMT 2016 armv7l
GNU/Linux
```

Note that it is also possible to use your preferred shell in the guest system by setting up SHELL environment variable prior to call to `exagear`.

2.4 Installing Linux x86 Applications

Installation of an x86 application should be performed inside the x86 guest environment. Invoke `exagear` command to begin a guest session. Before proceeding with installation of guest applications ensure that you have the necessary privileges for packages installation. For example:

```
user@arm-server:$ exagear
Starting the shell in the guest image /opt/exagear/images/debian-8
user@arm-server@x86:$ sudo apt-get update
user@arm-server@x86:$ sudo apt-get install nginx
```

The x86 application binary is installed in the guest image `/opt/exagear/images/<guest_system_version>`.

If installed application includes cron jobs or udev-rules you need to reboot your host system to finalize their registration on the host system. Otherwise it is possible to register them manually by executing the following command:

- from the host Ubuntu system

```
user@arm-server:$ sudo initctl start update-exagear-guest-<guest_system_version>
```

- from the host Debian or Raspbian system

```
user@arm-server:$ sudo service exagear-guest-<guest_system_version> start
```

Guest jobs will be placed in `/etc/cron.d` directory on your host system. By default cron-daemon updates its configuration each minute, so jobs will be queued automatically. In case if you are altering your jobs in the guest system you should register them again in the host.

Guest udev-rules will be placed in `/etc/udev/rules.d` directory on your host system. Committed guest system's udev-rules always run after all of the host system's rules have been

run.

2.5 Running Linux x86 Applications

After the installation of an x86 application is complete it can be run both from the guest and host sessions.

You can use the command `exagear` to start the shell of the guest session and then run any x86 application the same way as on an x86 system. For example:

```
user@arm-server:$ exagear
Starting the shell in the guest image /opt/exagear/images/debian-8
user@arm-server@x86:$ which nginx
/usr/sbin/nginx
user@arm-server@x86:$ /usr/sbin/nginx -h
```

Inside the host session you have two options:

1. enter the command `exagear --` and path to the x86 application inside the guest system in one line. For example:

```
user@arm-server:$ exagear -- /usr/sbin/nginx -h
```

2. enter the full path to the binary of x86 application because it is located under a particular directory `/opt/exagear/images/<guest_system_version>` called the x86 world. For example:

```
user@arm-server:$ /opt/exagear/images/debian-8/usr/sbin/nginx -h
```

Only the x86 world is visible for an x86 application. Sometimes it is necessary to access files that are located outside the x86 world on the host system. In this case you can mount a new host directory to a mount point inside the x86 world by using coreutils. For example, if you need to make visible `/newdir` host directory from the guest system perform the following actions.
Create mount point inside the x86 world

```
$ sudo mkdir /opt/exagear/images/debian-8/shareddir
```

and then mount `newdir` to the mountpoint `shareddir`

```
$ sudo mount --bind /newdir /opt/exagear/images/debian-8/shareddir
```

Note that it is prohibited to have x86 and ARM versions of the same service installed simultaneously on the same machine. Also we strongly recommend do not run host services from the guest system as it is not tested and not supported.
Note that doing `apt-get dist-upgrade` in the guest system is not supported.

2.6 Installing and Running Windows x86 Applications

Installing and running of Windows x86 applications can be done using Wine, a compatibility layer capable of running Windows applications on Linux (http://www.winehq.org).

You can instal Wine using `apt-get` in the guest system (except debian-7-wine2g).

```
user@arm-server:$ exagear
Starting the shell in the guest image /opt/exagear/images/debian-8
user@arm-server@x86:$ sudo apt-get install wine
user@arm-server@x86:$ wine --version
```

On debian-7-wine2g guest system Wine is already pre-installed.

```
user@arm-server:$ exagear
Starting the shell in the guest image /opt/exagear/images/debian-7-wine2g
user@arm-server@x86:$ which wine
/usr/bin/wine
```

We strongly recommend you to use pre-installed Wine in debian-7-wine2g guest system as it was enhanced specifically for Raspbian Wheezy 2g/2g split system.

2.7 Administration of ExaGear

Before proceeding with administration ensure that your user can elevate privileges needed.

SUID and SGID binaries under ExaGear

By default support for SUID and SGID guest binaries is enabled in ExaGear. To disable this support switch `ENABLE_SUID` value to `N` in the file `/etc/binfmt-x86.conf on the host system:`

```
user@arm-server:$ sudo cat /etc/binfmt-x86.conf
# ……
ENABLE_SUID=N
```

Then restart `exagear` service:

```
user@arm-server:$ sudo service exagear restart
```

Note that SUID and SGID guest binaries won't run as sudo user if `exagear` service is not enabled. Normally `exagear` service is automatically started right after installation of ExaGear and automatically restarted upon startup of the host machine:

```
user@arm-server:$ sudo service exagear status
x86_guest is enabled
```

As long as `exagear` service is `binfmt_misc` wrapper the lack of `binfmt_misc` kernel module on the host system leads to the impossibility to run SUID and SGID guest binaries as sudo user. In this case you will be able to run SUID and SGID guest binaries only as root.

Directories shared between the host and guest systems

Configuration file `/opt/exagear/images/<guest_system_version>/.exagear/vpaths-list` contains the list of directories and files that are shared between the host and guest systems. For example, user accounts and `/home` directories are shared between the host and guest systems.

Sysctl and network configuration

Sysctl and network configurations are shared between the host and guest systems.

The guest system has access to files that describe the network configuration, like `/etc/resolv.conf`. The guest, however, is not supposed to modify the network settings, so the directory `/etc/network` is purposely empty in the guest system. Configuration of network settings should be done on the host system.

The guest system can access sysctl settings in `/proc/sys/` but can not update `/etc/sysctl.conf` or `/etc/sysctl.d/`. Configuration of sysctl parameters should be done on the host system.

Integration of guest SysV-style init scripts and Upstart jobs into the host

Debian 7, Raspbian uses SysV-style init scripts as the init system. Ubuntu 12.04, Ubuntu 14.04 uses upstart jobs and SysV-style init scripts as the init system. Many software packages provide configuration for these services.

By default, the fully automatic integration mechanism is employed to represent guest services as host ones. Guest services are registered as the host ones under the same name. Thus you can manage guest services both from the guest and host shell (in the last case host service delegates all start/stop/testconfig/etc. requests to the respective guest service). For example:

```
user@arm-server:$ exagear
Starting the shell in the guest image /opt/exagear/images/debian-8
user@arm-server@x86:$ sudo apt-get update
user@arm-server@x86:$ sudo apt-get install nginx
user@arm-server@x86:$ sudo service nginx status
* nginx is running
user@arm-server@x86:$ exit
user@arm-server:$ sudo service nginx status
* nginx is running
```

You can see which services are registered in the guest system by listing files under the `/opt/exagear/images/<guest_system_version>/etc/init.d/` and `/opt/exagear/images/<guest_system_version>/etc/init/` directories of the guest.

After uninstallation of guest applications execute the following command from the host system to delete respective configuration files in the host system:

- from the host Ubuntu system

```
user@arm-server:$ sudo initctl start update-exagear-guest-<guest_system_version>
```

- from the host Debian, Raspbian system:

```
user@arm-server:$ sudo service exagear-guest-<guest_system_version> start
```

Note that it is prohibited to have x86 and ARM versions of the same service installed simultaneously on the same machine. Also we strongly recommend do not run host services from the guest system as it is not tested and not supported.

Integration of guest cron jobs into the host

By default all guest cron jobs are registered to the host upon startup of the host machine. Otherwise it is possible to register them manually by executing the following command:

- from the host Ubuntu system

```
user@arm-server:$ sudo initctl start update-exagear-guest-<guest_system_version>
```

- from the host Debian, Raspbian system:

```
user@arm-server:$ sudo service exagear-guest-<guest_system_version> start
```

`/etc/crontab` configuration file, cron jobs located in `/etc/cron.d` directory and user-specific cron jobs located in `/var/spool/cron/crontabs/` directory of the guest system are converted to host cron jobs into `/etc/cron.d/` directory. Guest jobs names in the host are prefixed with “EG_”. This way you can distinguish host and guest jobs.

Note that cron jobs in `/etc/cron.{hourly,daily,weekly,monthly}` of the guest system are not converted to host cron jobs since they are automatically handled by the converted version of the guest’s `/etc/crontab`. Consider an example:

```
Guest system:                       Host system:
/etc                                /etc
    /crontab                            /crontab
    /cron.d                             /cron.d
        /my-recurrent-job
    /cron.hourly                        /cron.hourly
    /cron.daily                         /cron.daily
    /cron.weekly                        /cron.weekly
        /my-weekly-job
    /cron.monthly                       /cron.weekly

/var/spool/cron/crontabs/joe        /var/spool/cron/crontabs/
```

As a result of guest cron jobs registration the following changes to the host system will be

committed:

```
Guest system:                  Host system:
/etc                           /etc
  /crontab --------------+       /crontab
  /cron.d                |       /cron.d
                         +------> /EG_<guest_system_version>_crontab
                                        (handles guest's
                                         cron.hourly,cron.daily,
                                         cron.weekly,cron.monthly)
     /my-job ------------------> /EG_<guest_system_version>_my-job
                              +--> /EG_<guest_system_version>_peruser_jo
    /cron.hourly              |   /cron.hourly
    /cron.daily               |   /cron.daily
    /cron.weekly              |   /cron.weekly
        /my-weekly-job        |
    /cron.monthly             |   /cron.weekly
                              |
/var/spool/cron/crontabs/jo--+
```

By default cron-daemon updates its configuration each minute, so jobs are queued automatically. In case if you're altering your jobs in a guest system you should register them again in the host.

You can restrict registration of guest cron jobs to the host by setting `DELEGATE_CRONTABS_AUTOMATICALLY=N` in the configuration file `/etc/exagear-guest-<guest_system_version>.conf`.

Integration of guest udev services into the host

To provide user with the ability of handling device events straight from the guest system, ExaGear incorporates integration of udev subsystem. This is required in case if a guest application provides facilities to work with some encrypted USB devices such as HASP keys etc. and uses udev-rules to subscribe on device tree changes. In this case application specific rules should be committed to the host system, otherwise application will remain unnotified on device tree changes.

By default all guest rules are registered to the host upon startup of the host machine. Otherwise it is possible to register them manually by executing the following command:

- from the host Ubuntu system

```
user@arm-server:$ sudo initctl start update-exagear-guest-<guest_system_version>
```

- from the host Debian, Raspbian system:

```
user@arm-server:$ sudo service exagear-guest-<guest_system_version> start
```

All application specific udev rules located in

`/opt/exagear/images/<guest_system_version>/etc/udev/rules.d/` directory of the guest system are converted to host udev rules into `/etc/udev/rules.d/` directory. Guest rules names in the host are prefixed with “EG_”. This way you can distinguish host and guest rules. Registered guest system udev-rules always run after all rules of the host system.

After uninstallation of guest application with registered udev-rule to synchronize guest system udev-rules database with the host one execute:

- from the host Ubuntu system

```
user@arm-server:$ sudo initctl start update-exagear-guest-<guest_system_version>
```

- from the host Debian, Raspbian system:

```
user@arm-server:$ sudo service exagear-guest-<guest_system_version> start
```

2.8 Update ExaGear

To update the version of ExaGear you should perform following actions:

1. Ensure that you have the necessary privileges to perform following actions
2. Ensure that all of the previously installed guest applications are stopped and removed from the guest system
3. Remove previous version packages:

```
$ sudo apt-get purge exagear exagear-mem2g exagear-legacy exagear-legacy-mem2g
exagear-armv8
```

4. Ensure ExaGear dependencies:

```
$ sudo apt-get install bash coreutils findutils realpath curl binfmt-support cron
```

6. Identify versions of ExaGear packages best suitable for your host system.
    a. Check the architecture of your ARM device:

```
$ arch
```

    b. Check user/kernel virtual memory split of the host system (execute the hole line in command line terminal and see the output):

```
[ $((`cat /proc/self/maps | grep stack | awk -F'-' '{print "0x"$1}'`)) -gt
$((0x80000000)) ] && echo 3g/1g || echo 2g/2g
```

    c. Check the version of the host operating system:

```
$ sudo cat /etc/issue
```

    d. Find versions of ExaGear packages to install in the table below:

Arch	User/ kernel split	Operating system	ExaGear packages to install
aarch64	3g/1g	Debian Jessie 8	`exagear-armv8_*.deb` `exagear-guest-debian-8_*.deb`

aarch64	3g/1g	Ubuntu 15.xx	`exagear-armv8_*.deb` `exagear-guest-ubuntu-1504_*.deb`
aarch64	3g/1g	Ubuntu 16.xx	`exagear-armv8_*.deb` `exagear-guest-ubuntu-1604_*.deb`
armv7	2g/2g	Raspbian Wheezy 7	`exagear-mem2g_*.deb` `exagear-guest-debian-7-wine2g_*.deb`
armv7	2g/2g	Raspbian Jessie 8	`exagear-mem2g_*.deb` `exagear-guest-debian-8_*.deb`
armv7	2g/2g	Ubuntu 14.xx	`exagear-mem2g_*.deb` `exagear-guest-ubuntu-1404lts_*.deb`
armv7	2g/2g	Ubuntu 15.xx	`exagear-mem2g_*.deb` `exagear-guest-ubuntu-1504_*.deb`
armv7	2g/2g	Ubuntu 16.xx	`exagear-mem2g_*.deb` `exagear-guest-ubuntu-1604_*.deb`
armv7	3g/1g	Debian Wheezy 7	`exagear_*.deb` `exagear-guest-debian-7_*.deb`
armv7	3g/1g	Debian Jessie 8	`exagear_*.deb` `exagear-guest-debian-8_*.deb`
armv7	3g/1g	Ubuntu 14.xx	`exagear_*.deb` `exagear-guest-ubuntu-1404lts_*.deb`
armv7	3g/1g	Ubuntu 15.xx	`exagear_*.deb` `exagear-guest-ubuntu-1504_*.deb`
armv7	3g/1g	Ubuntu 16.xx	`exagear_*.deb` `exagear-guest-ubuntu-1604_*.deb`
armv6	2g/2g	Raspbian Wheezy 7	`exagear-legacy-mem2g_*.deb` `exagear-guest-debian-7-wine2g_*.deb`
armv6	2g/2g	Raspbian Jessie 8	`exagear-legacy-mem2g_*.deb` `exagear-guest-debian-8_*.deb`
armv6	3g/1g	Raspbian Wheezy 7	`exagear-legacy_*.deb` `exagear-guest-debian-7_*.deb`
armv6	3g/1g	Raspbian Jessie 8	`exagear-legacy_*.deb` `exagear-guest-debian-8_*.deb`

7. Use `dpkg` to install ExaGear packages. For example:

```
$ sudo dpkg -i exagear-mem2g_*.deb
$ sudo dpkg -i exagear-guest-debian-8_*.deb
```

8. Use license key to activate ExaGear

```
$ sudo cp pk*.key /opt/exagear/lic
$ sudo /opt/exagear/bin/actool
```

2.9 Removing ExaGear

Before you proceed with removal of ExaGear you should ensure that all of the previously installed guest applications are stopped and removed from the guest system.

To remove ExaGear use `apt-get purge`:

```
$ sudo apt-get purge exagear exagear-mem2g exagear-legacy exagear-legacy-mem2g
exagear-armv8
```

As a result the virtualization engine and support tools will be removed; directories and files of the guest file system installed within the package will be removed.

Note that `apt-get remove` and `apt-get purge` do not differ for the package with the guest file system. In both cases all files in the guest file system (files installed with ExaGear and those that were later created by user) are deleted.

3 Errata

1. The length of absolute paths that ExaGear can use is limited to 4KiB.
2. x87 (floating point) 80-bit precision support is limited.
    a. 80-bit operands are converted to 64-bit ones.
    b. There are discrepancies in case of NaN operand
    c. There are discrepancies in case of raising floating point exceptions
3. We provide limited support for `/dev/mem` and `/proc`.

4 Support

Report issues to the e-mail address support@eltechs.com. Please provide the log files and detailed description of the problem.

5 ExaGear Concepts

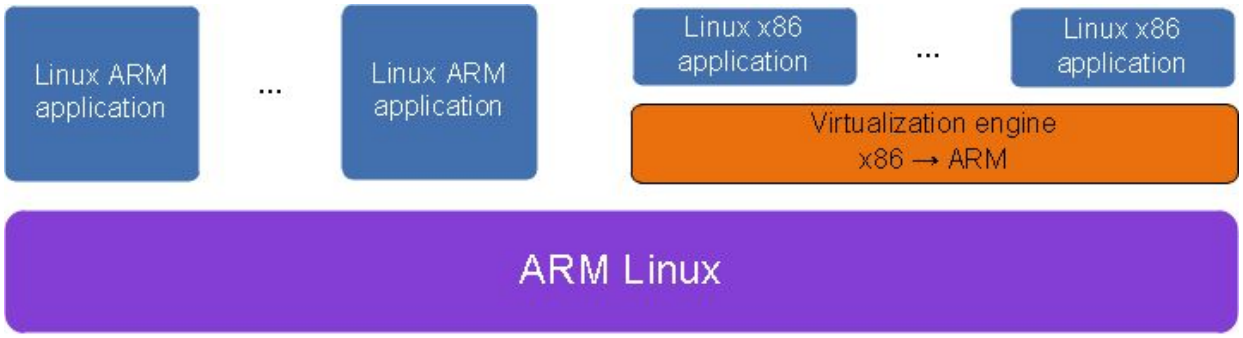
This chapter gives the general view on how ExaGear works.

There are two main components of ExaGear – virtualization engine and x86 world. ExaGear virtualization engine allows to run x86 Linux applications on top of ARM-based devices with ARM Linux operating system without porting or recompiling applications. ExaGear creates an

x86 world that contains all standard libraries, utilities and configurational files. This approach ensures availability of components of infrastructure required by x86 applications to run.

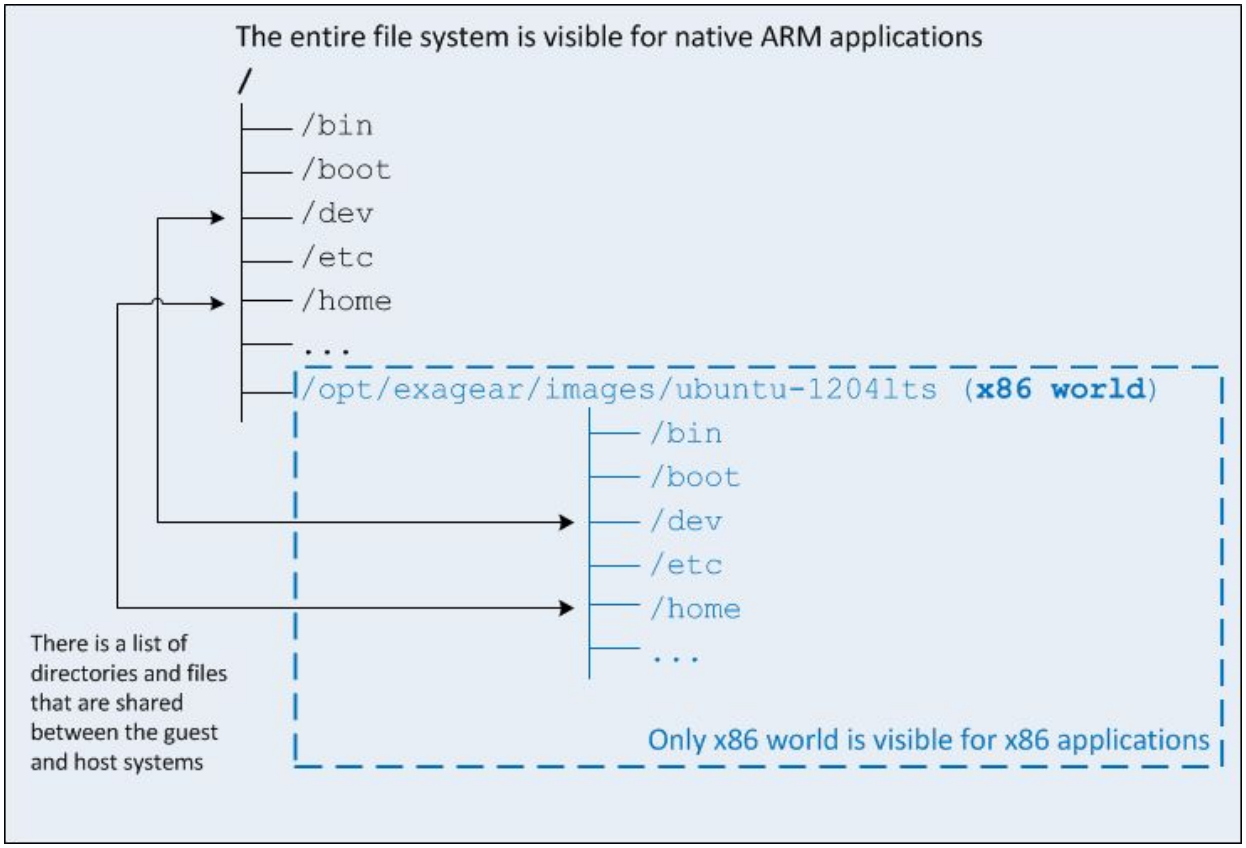
Virtualization engine.
ExaGear virtualization engine is a “middleware” software solution, positioned at the gap between the x86 application and the ARM-based server (see figure 5.1). Within the ExaGear the user sees no difference between running native ARM and x86 applications – ExaGear virtualization engine intercepts x86 applications from the very start, converts them in run-time into ARM-compatible code using binary translation technology, and executes them. The entire process is easy, and transparent to the end-user.

Figure 5.1: The Conceptual Representation of ExaGear



x86 world.
Technically x86 world is a particular file directory that contains the x86 libraries, commands, utilities and other system files (see figure 5.2). x86 applications’ binaries must be located in x86 world as well.

Figure 5.2: The conceptual representation of x86 world



Both virtualization engine and minimal x86 system are provided in ExaGear installation packages.