

ingmar-k / **Gnublin_Emdebian**

Watch

1

Star

2

Fork

0

<> Code

Issues 0

Pull requests 0

Projects 0

Security

Insights

Join GitHub today

Dismiss

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Scripts to automatically create a Emdebian rootfs image and/or a complete, bootable SD-card.

2 commits

1 branch

0 releases


0 contributors





Branch: master ▾


New pull request

Find file

Clone or download ▾

 Ingmar Klein First release of the Emdebian scripts via github. Latest commit 1f9b5c0 on 8 Aug 2012

| | | |
|--|---|-------------|
|  README.md | First release of the Emdebian scripts via github. | 7 years ago |
|  build_emdebian_system.sh | First release of the Emdebian scripts via github. | 7 years ago |
|  build_functions.sh | First release of the Emdebian scripts via github. | 7 years ago |
|  general_settings.sh | First release of the Emdebian scripts via github. | 7 years ago |

 README.md

Gnublin_Emdebian

Scripts to automatically create a Emdebian rootfs image and/or a complete, bootable SD-card.

Author: Ingmar Klein (ingmar.klein@hs-augsburg.de) Created in scope of the "Embedded Linux" lecture, held by Professor Hubert Hoegl, at the University of Applied Sciences Augsburg, 2012

This program (including documentation) is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License version 3 (GPLv3; <http://www.gnu.org/licenses/gpl-3.0.html>) for more details.

README: Gnublin Emdebian scripts

This Readme should explain how to use the Emdebian build-scripts. It also explains the settings and their meaning.

Overview:

1. Host System Requirements

2. Files

3. Settings

4. Running the script

5. Log-Files

6. Logging In

1. Host System Requirements

You need a machine with at least Debian Squeeze, or Ubuntu 10.04. . Enough disk space for the build process (~2GB free space) and maybe 512MB RAM. Internet connection is mandatory for this script (on the host machine, not on the gnublin). No internet connection equals no build process!

2. Files

- build_emdebian_system.sh : Main script file, running the functions defined in the "build_functions.sh". This is what you run in shell.
- build_funtions.sh : As already mentioned, this includes the main build and some helper funtions that are used in the "build_emdebian_system.sh"
- general_settings.sh: File for all settings and configurations. Here you define what the output of the scripts will look like.

3. Settings

ATTENTION: You must (!) edit the "general_settings.sh" to fit your host system and your needs, before running the main script!

Most of the settings are either self-explanatory, or already commented in the file itself. I'll now list the settings that you absolutely must change/edit/check:

- "host_os": Either Emdebian or Ubuntu
- "output_dir_base": This sets the local path for all the files generated by the script
- "nameserver_addr": Without the correct setting for your network environment, the build process will most likely fail.

All other settings are optional and CAN, but don't necessarily need to be changed!

Note: The variable "additional_packages" just is a list of packages that should be installed. So, just browse the Emdebian package list (<http://www.emdebian.org/grip/search.php>) and add the ones you need/want, to the list. These will then be added to the resulting rootfs.

4. Running the script

Running the script is as easy as doing the following:

- Make sure that you downloaded all 3 script files (or the complete package) and edited the "general_settings.sh".
- Run "chmod +x build_emdebian_system.sh" if needed, to make the file executable.
- Run the script itself by typing "sudo ./build_emdebian_system.sh" in a terminal window (NOT in a virtual console!).
- Then follow the instructions on the screen! If nothing goes wrong, no user-input should be required until choosing the SD-Card device is necessary (if you did set that option!).
- The output should be in form of a compressed archive file (.tar.gz or .tar.bz2) containing the rootfs and kernel.
- If you set the option to build a SD-Card, you will be asked to specify the correct device, which then will be subsequently partitioned, formatted and filled with bootloader, rootfs and kernel.

5. Log-Files

There are a number of log files that get created at run time. The main log file is located in the "output_dir"-folder. This log file includes all main points in the script. Some detail information gets logged elsewhere, but that can be seen in the script code.

6. Logging In

The system gets built with a standard root account with password. The initial login data is the following:

User: root Password: root


If you want to change the root password, just run the command "passwd root" and follow the instructions. If you want to add a normal user account, run "adduser 'username' ", obviously with the name you want your user to have.

ATTENTION:

There is a Bug that causes the system to ask you to change the password on each system start, if the system time wasn't set when invoking passwd! Make sure that the system time is set to something else than 01/01/1970, if you intend to change the password!

HAVE FUN!

© 2019 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)



[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)