Preface

As the need to have a general purpose Base Filesystem emerged, the need to have multiple frontends for the installation process was crucial.

Therefore, the horus project was divided into five components:

1- **The Backend:** The script that performs the installation process.

2- **The Handler:** A set of functions that are used by the controller to validate and insert values into the parameters file or returns an error code if the input was invalid.

3- **The Parser:** A set of functions that are used by the controller to translate error codes returned by the handler and passes it to the controller.

4- **The Controller:** A set of functions that are used by the frontend to perform a specific operation, it takes the user input, sends it to the handler, then receives the return value and send it to the parser to get a printable text and returns that text to the frontend.

5- **The Frontend:** The part that interacts with the user, and since the StdBaseFS comes with no Xserver, the frontend should use a proper console frontend engine, supported engines are dialog, readline or a build an engine to provide a single command with parameters parsed as arguments.

We expect the following documentation to be a guide to understand the entire installation process and be a good reference for other developers planning to contribute to the project and/or distribute it separately.

The entire codebase is written in *“Bash scripting language”* in order to guarantee portability and the ease of use for the system administrator.

If you are not familiar with the unix environment or you are don't know how to write bash scripts we recommend a range of resources for you to study then return to this document.

Resources for beginners:

1- The Art of Unix Programming.
[Eric S. Raymond]

2- TLDP.ORG's Bash Guide for Beginners.
[Machtelt Garrels.]

3- UNIX and Linux System Administration Handbook (4th Edition).
[Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley]

4- Advanced Programming in the UNIX Environment (2nd Edition).
[W. Richard Stevens, Stephen A. Rago]

5- TLDP.ORG's Advanced Bash-Scripting Guide.
[Mendel Cooper.]

Section 1: [Parameters File]

a. Parameters file definition:

The Parameters file is a where you should pass arguments to the Backend, currently it has 13 parameters to set.

The default location of the Parameters file is "/var/cache/horus/backend.parm"

This file is copied from the "iso file" filesystem, it resides under "install/backend.parm" where applications such as RedistWizard can modify before the build is ready to be used.

And since the "Debian Live scripts" mount the iso filesystem under "/lib/live/mount/medium/" as read-only file system. The parameters file is copied automatically during live boot time from "/lib/live/mount/medium/install/" to "/var/cache/horus/".

---

b. Parameters file fields:

NOTICE: You should enter the values after the ":" sign and before the ";" sign as the ";" sign is used as an END-OF-LINE Delimiter.

Required fields:

1- rootpart
Root partition that will be used by the installed BaseFS

2- swappart
Swap partition that will be used by the installed BaseFS

3- uname
Default username "UID #1000"

4- upass
Default user password "UID #1000"

5- rpass
Root "Administrator" password

6- timezone
Timezone to be used by the system

Optional fields:

7- hostname
Installed system hostname
Default value: '$uname'-PC

8- homepart
Home partition that will be used to store users' data and files
Default value: root

9- homefsformat
Home partition format, values can be ext4, ext3 ,ext2 ,noformat
Default value: noformat

10- grubloc
Where to install GRUB "Bootloader"
value can either be MBR in case you want GRUB to be installed to the MBR or root if you have an already installed bootloader on the MBR
Default value: MBR

11- ufullname
Default full username "UID #1000"
Leave blank if not sure

12- removepacks
Packages to remove from the installed StdBaseFS
Notice: Packages should be preinstalled in the StdBaseFS

13- installpacks
Packages to install into the installed StdBaseFS
Notice: Packages and their dependencies should be present in the iso filesystem

---

c. Parsing a custom Parameters file to the Horus Backend:

The horus-backend script takes the first argument as the custom Parameters file.

If no Parameters file is assigned, then it assumes that you wish to use the default Parameters file under "/var/cache/horus/backend.parm".

example: Supposing that you made another parameters file under "/usr/share/myparameters", you can easily inform the backend of that by the following command:

```
$ horus-backend /usr/share/myparameters
```

Section 2: [Horus Backend overrides]

a. The overrides definition:

In order to maximize the level of extensibility, we implemented a few more overrides to the StdBaseFS, so that it could be redistributed easily.

These overrides reside in the "install/" directory of the iso filesystem, and they cover a wide range of modifications to installed StdBaseFS.

The distributor will be able to set these overrides during the build time, and the generated iso filesystem will have these modifications made when it's installed through Horus Backend.

---

b. The Overrides:

1- Custom GRUB background:
This override is the only one outside the "install/" directory of the iso filesystem, simply it's a JPEG image that is named grub.jpg

It resides under "isolinux/grub.jpg", simply overwrite it and the horus-backend will copy that to "/etc/grub.jpg" where GRUB expects to find the bootloader background.

2- Custom "sources.list" file:
As the StdBaseFS uses "apt" as the default package manager, it needs a repository list that resides under "/etc/apt/sources.list".

This is also customizable, as the backend will automatically copy a sources.list file from it's location "install/sources.list" from the iso filesystem to the newly installed system under "/etc/apt/sources.list" and when you reboot to the newly installed system and update the repositories list with the command "apt-get update", the custom "sources.list" file will be used to supply it with the repositories.

3- Custom repositories authentication public keys:
When using a custom "sources.list" file, newly added repositories will trigger the following warning message:

```
+-----------------------------------------------------------------+
|   W: GPG error: http://ftp.myrepo.org stable Release:           |
|   The following signatures couldn't be verified because         |
|   the public key is not available: NO_PUBKEY XXXXXXXXXXXXXXXX   |
+-----------------------------------------------------------------+
```

And this will bother the user when installing new packages from this repository with the following warning message:

```
+-------------------------------------------------------------+
|   WARNING: The following packages cannot be authenticated!  |
|   userspackage1 userspackage2                               |
|   Install these packages without verification [y/N]?        |
+-------------------------------------------------------------+
```


This can be resolved when redistributing the StdBaseFS by adding the repositories' keys files to the iso filesystem under "install/apt-keys/", and Horus Backend script will add it to the newly installed system.

4- Custom system-wide configurations:
In case the distributor wishes to use custom system-wide configurations for specific packages this can be accomplished by adding a squashfs "containing filestructure of configuration files" in the iso filesystem under "install/sysconf/etc.squashfs".

Permissions and file structure inside the squashfs should be relative to the /etc directory as shown below:

```
|+package1/
|_________|-file2.conf
|_________|+conf.d/
|_________|_________|-file.conf
|
|+package2/
|_________|-file3.conf
```

The Horus Backend will automatically unsquashfs it and place it to the newly installed system if it finds it.

5- Custom user-specific configurations:
In case the distributor wishes to use user-specific configurations that will be placed in the newly installed system under "/etc/skel" this can be accomplished by adding a squashfs "containing the filestructure of configuration files" in the iso filesystem under "install/userconf/skel.squashfs".

Permissions and filestructure inside the squashfs should be relative to /etc/skel directory as shown in (section: 2.b-4)

6- Custom files to be added to "/":
In case the distributor wishes to add extra files to the root directory of the installed system this can be accomplished by adding a squashfs "containing the filestructure of extra files" in the iso filesystem under "install/rootfs/root.squashfs", permissions and filestructure inside the squashfs should be relative to the / directory as shown below:

```
|+/
|_________|+usr/
|_________|_________|+bin/
|_________|_________|_________|-myapp.sh
|_________|_________|+share/
|_________|_________|_________|+doc/
|_________|_________|_________|_________|+myapp/
|_________|_________|_________|_________|_________|-changelog.gz
|_________|_________|_________|_________|_________|-copyright
|_________|+var/
|_________|_________|+lib/
|_________|_________|_________|+myapp/
|_________|_________|_________|_________|-myapp.cache
```

7- Custom post-installation scripts:
In case the distributor wishes to execute scripts after everything is installed and right before GRUB is installed.
This can be accomplished easily depending on where the script should run:

a- The scripts will runs inside the live environment:
    The distributor might need to run scripts inside the live session to perform a specific task.
    This can be accomplished by adding the script to the iso filesystem under "install/scripts/live/".

    The script filename should have the ".exec" extension (example: myscript.exec), and the Horus Backend will automatically run it inside the live system.

b- The scripts runs inside the installed system "chrooted environment":
    The distributor might need to run scripts inside the newly installed system in a chrooted environment "with network access through the live system if available".

    This can be accomplished by adding the scripts to the iso filesystem under "install/scripts/host".

    The script filename should have the ".exec" extension (example: myscript.exec), and the Horus Backend will automatically run it inside the newly installed system.

8- Add packages to install to the StdBaseFS:
In case the distributor needs to add one or more packages to the iso filesystem in order to use the "installpacks" field in the Parameters file see (section: 1.b-13).
which requires all ".deb packages" to be locally present and accessible where Horus Backend expects to find under "install/packages" in the iso filesystem.

Note that also all dependencies of these packages down to the bottom should be present under the same directory, this is hard to accomplish without knowing which packages were pre-installed inside the StdBaseFS, so a chrooted environment might give you an idea about how to reslove these dependencies.

Tools such as RedistWizard can accomplish that easily without the need to such a manual operation which might generate a broken build.

Also a list of available packages inside the iso filesystem should be present under the same directory where packages reside "install/packages".

You might use the following commands when building the iso filesystem:

```
# $WORKDIR is where the iso filesystem is being built
$ cd $WORKDIR/install/packages
$ apt-ftparchive packages . > Packages
$ bzip2 -kf Packages
```


Section 3: [Horus Backend debugging information]

a. Options:

Horus Backend has only one option which is the enable verbosity option.
This option turns on verbosity for running operations and prints it to the stdout as well as the log file described in (section: 3.b).

By default this option is not enabled, and only error messages will be printed to stdout.

This option can be turned on by passing the "-v" option to "horus-backend" command followed by your custom Parameters file if you need to specify one.

example [No custom Parameters file specified]:
```
$ horus-backend -v
```

example [Custom Parameters file specified]:
```
$ horus-backend -v /usr/share/myparameters
```

---

b. Log file:

The horus-backend script writes out all of it's operations output to a log file located at "/var/log/horus/backend.log".

the log file is copied to the installed system if installed successfully.

---

c. Exit codes and corresponding error messages:

horus-backend script has the following exit codes:

0: Installation finished, no errors reported.

1: Insufficient privileges, must run the script as root.

2: Cannot find Parameters file.

3: Another instance of the installer is already running.

4: Invalid Identifier in parameters file.

5: Bad or missing target partition.

6: Not enough space in specified root partition.

7: Bad or missing swap partition.

8: Cannot find specified home partition.

9: Bad home partition filesystem type.

10: Missing username.

11: Bad username, only English characters, numbers, dashes and underscores are allowed.

12: Bad username, specified username is system reserved.

13: Bad Full username, only English characters, numbers, dashes, underscores, spaces, commas and dots are allowed.

14: Bad hostname, only English characters, numbers, dashes and underscores are allowed.

15: Missing password.

16: Bad password, non-ASCII characters are not allowed.

17: Missing root password.

18: Bad root password, non-ASCII characters are not allowed.

19: Unrecognized specified option for grub location, values can either be 'MBR' or 'root'.

20: Unrecognized timezone.

21: Package you are trying to remove is not installed.

22: Cannot find any packages to install on installation medium.

23: Specified Package is not installable.

---

d. PID file:

The horus-backend script writes the process ID "PID" in a file to make sure that only one instance of the script is running.

The PID file is located at "/var/run/horus.pid" and is deleted automatically once the script exits.

---

e. Status file:

The horus-backend script reports if any errors has occurred whether verbosity is enabled or not,

This is achieved by writing the exitcode to "/var/run/horus.status" and checking if the file exists or not and trigger the error message if the file exists and exits with the exitcode value stored in the file.

Section 4: [The horus handler]

a. Definition:

A set of functions written entirely in bash that provide a simple standard method to validate and pass user's input to the parameters file.

It's located under “/usr/share/horus/scripts/horus-handler”
where you can include them in any script by issuing the following line:

```
$ . /usr/share/horus/scripts/horus-handler
```

Functions take at one argument or more and returns a non-zero value if an error was detected.

---

b. Functions:

NOTICE:
- You must use functions marked with (*) before using any other function.
- Return value (255) means invalid number of arguments detected.
- You should quote passed arguments to avoid accidental 255 return values.

Function	Argunments			Error Return Value	
Name / Desc.	Arg.	Status	Desc.	value	Desc.
*chkRoot() “Checks for root privileges”	None			1	Not enough permissions.
*setParmFile(string *$PARMFILE*) “Sets the parameters file to read from and write into.”	*$PARMFILE*	Optional	The parameters file to be used, if no arguments were supplied, will set value to default parameters file path “/var/cache/horus/backend.parm”.	2	File doesn't exist or not enough permissions.
*chkPid() “Makes sure no second instance of the backend is running"	None			3	Another instance is running.
*getParms() “Reads previously set values from parameters file”	None			4	Unrecognized parameter detected.
SetRTPart(string *$NEWTARGETPART*) “Sets root partition to install the system into”	*$NEWTARGETPART*	Mandatory	Target root partition to install the system into.	5	Not a valid block device.
				6	Not enough space.
ChkPartSize(string *$NEWTARGETPART*) “Checks if the root partition size is enough for the StdBaseFS to be installed – called automatically by SetRTPart()”	*$NEWTARGETPART*	Mandatory	Target root partition to install the system into.	6	Not enough space.
SetSwap(string *$NEWSWAP*) “Sets SWAP partition for the installed system”	*$NEWSWAP*	Mandatory	SWAP partition for the installed system.	7	Not a valid block device.

Function	Argunments			Error Return Value	
setHome(string *$NEWHOME*, string *$NEWHFSTYPE*) “Sets Home partition and it's format for the installed system”	*$NEWHOME*	Optional	Home partition for the installed system – a value should be a block device or “root” for the default value.	8	Not a valid block device.
	$NEWHFSTYPE	Optional	Home partition format – a value should be one of “ext4, ext3, ext2 or noformat for the default value”.	9	Unsupported partition format.
setUser(string *$NEWTUSER*) “Sets username for UID:#1000”	*$NEWTUSER*	Mandatory	username for UID:#1000.	10	Username not passed.
				11	Unsupported characters in username.
				12	System reserved username.
setUFName(string *$NEWUFNAME*) “Sets Full username for UID:#1000”	*$NEWUFNAME*	Optional	Full username for UID:#1000. default value is “$NEWTUSER”.	13	Unsupported characters in full username.
setHostname(string *$NEWHSTNM*) “Sets hostname for installed system”	*$NEWHSTNM*	Optional	hostname for newly installed system. Default value is “$NEWTUSER-PC”.	14	Unsupported characters in hostname.
setUPass(string *$NEWUPASS*, string *$CNFRMUPASS*) “Sets password for UID:#1000”	*$NEWUPASS*	Mandatory	Password for UID:#1000.	15	Password not passed.
	$CNFRMUPASS	Mandatory	Password confirmation for UID:#1000.	16	Passwords don't match.
				17	Unsupported characters in password.
setRootPass(string *$NEWRTPASS*, string *$CNFRMRTPASS*) “Sets root user password for installed system”	*$NEWRTPASS*	Mandatory	root user password.	18	Password not passed.
	$CNFRMRTPASS	Mandatory	root user password confirmation.	19	Passwords don't match.
				20	Unsupported characters in password.
setGrubLoc(string *$NEWGRUBLOC*) “Sets The location of the Bootloader's installation”	*$NEWGRUBLOC*	Optional	Location of bootloader's installation, a value should either be “root” or “MBR” for default value.	21	Unsupported value for GRUB location.
setTZ(string *$NEWTZ*) “Sets default timezone of the installed system”	*$NEWTZ*	Mandatory	Timezone value.	22	Unrecognized timezone.
SetRemovePacks(string *$NEWRMPACKLST*) “Sets packages to remove from StdBaseFS”	*$NEWRMPACKLST*	Mandatory	Quoted packages names separated by spaces	23	Package not installed in StdBaseFS.
setInstallPacks(string *$NEWINPACKLST*) “Sets packages to install to system”	*$NEWINPACKLST*	Mandatory	Quoted packages names separated by spaces	24	No packages detected on installation medium.
				25	Package cannot be installed.
ChkComplete() “Retruns backend script's exitcode”	None			any	Returns the exitcode of the backend script.