Relax & Recover – rear

The Ultimate Disaster Recovery Framework
version 1.1

By

Schlomo Schapiro
Gratien D'haese

Introduction

In the area of Linux Disaster Recovery Planning there are no many Open Source programs available to help you building a decent plan. The two most know so fare were "Mondo Rescue" and "make CD-ROM Recovery (mkCDrec)" which both have their strengths and weaknesses. Most probably the biggest weakness of both programs was the monolithic design which scared lots of potential new developers.

The author of mkCDrec, Gratien D'haese, decided together with Schlomo Schapiro to rewrite mkCDrec from scratch in a complete modular way in June 2006. In less then a month time we released rear version 1.0 (July 2006). This was only possible because of the modular design and separation of duties in the way of writing modules (also called workflows).

In this document we will explain the basics of rear and how the modular process does its job.

The main web-site of rear is located at SourceForge:
http://rear.sourceforge.net/

Purpose and Key Features

The main purpose of rear is to have a complete disaster recovery image from your Linux server without too much knowledge. It is designed so it can executed by operators and not only by system administrators. Automation is the key concept and keep it simple as possible.

The key features of rear are:

- Focus on Disaster Recovery
- Modular concept
- For Linux and other Unix-like operations systems
- No external dependencies – use only standard software supplied with the distribution
- Linux: kernel > 2.6 supported (no kernel 2.2/2.4 support !)
- User friendly – minimal output, use log file for error messages and details (/tmp/rear.log)

- Supports lots of file systems such as ext[2,3,4], reiserfs, xfs, jfs, lvm2, software raid
- Backup (and restore) methods are not limited to the built-in tar, but any choice (commercial or open source) are possible as long as a proper workflow is made for this
- Output image can be stored on a local file system, remote file system or disk (NFS, CIFS, or USB), PXE server, tape device or ISO file system
- Licensed under the GNU GPL v2 meaning real open source not pseudo open

The aim is to make rear as least demanding as possible, it will require only the applications necessary to fulfill the job rear is configured for. All other applications will be copied to the rescue system if they are present.

Furthermore, rear should be platform independent and ideally install just as a set of scripts that utilizes everything that the Linux Operating System provides.

Rear follows the Linux Standard Base rules meaning that the configuration files are stored under */etc/rear*. The workflow scripts of rear are stored under */usr/share/rear* directory structure and the documentation under */usr/share/doc/rear-<version>*. There is only one main script of rear, */usr/sbin/rear,* that steers the complete rescue or restore phases.

The Workflow System

To understand how rear works internally we need to explain the workflow system built into rear. The following is the output of the "*rear help*" command:

```
Relax & Recover Version 1.7.9 / 2008-12-02
rear [Options] <command> [command options ...]
Relax & Recover Version 1.7.9 / 2008-12-02
Build: 85c49e8c94ac8055342c53e044c22929
Copyright (C) 2006-2008
        Schlomo Schapiro, probusiness Berlin AG
        Gratien D'haese, IT3 Consultants
Relax & Recover comes with ABSOLUTELY NO WARRANTY; for details
see the GNU General Public License at http://www.gnu.org/licenses/gpl.html

Available Options:
-V                      version information
-d                      debug mode
-D                      debug script mode
-S                      Step-by-step mode
-s                      Simulation mode (shows the scripts included)
-r a.b.c-xx-yy          kernel version to use (current: 2.6.27.7-53.fc9.i686)

List of commands:
dump            Dump configuration and system information
help            Print out usage
mkbackup        Create rescue media and backup system.
```

mkbackuponly	Backup system without creating a (new) rescue media.
mkdeb	Create DEB packages with this rear version
mkdist	Create distribution tar archive with this rear version
mkrescue	Create rescue media only
mkrpm	Create RPM packages with this rear version
mktar	Create tar archive with this rear installation
recover	Recover the system
validate	Submit validation information

Rear is built as a modular framework. Each call to a certain <command> like in "**rear <command>"** will invoke the following general workflow:

1. Configuration:
   Collect system information to assemble a correct configuration (default, arch, OS, OS_ARCH, OS_VER, site, local). See the output of **rear dump** for an example.
   Read config files for the combination of system attributes. Always read **default.conf** first and **site.conf, local.conf** last.

2. Create work area in **/tmp/rear.$$** and start logging to **/tmp/rear.log**

3. Run the workflow script for the specified command:
   /usr/share/rear/lib/<command>-workflow.sh

4. Cleanup work area

Workflow – dump

When we execute the command "*rear dump*" we go into the "dump" workflow of rear. The description of such workflow is defined in a script called */usr/share/rear/lib/dump-workflow.sh*.

The only purpose of the dump workflow is to show the current rear environment – see below:

```
Relax & Recover Version 1.7.9 / 2008-12-02
Dumping out configuration and system information
System definition:
                        ARCH = Linux-i386
                        OS = GNU/Linux
                        OS_VENDOR = Fedora
                        OS_VERSION = 9
                        OS_VENDOR_ARCH = Fedora/i386
                        OS_VENDOR_VERSION = Fedora/9
                        OS_VENDOR_VERSION_ARCH = Fedora/9/i386
Configuration tree:
                        Linux-i386.conf : OK
                        GNU/Linux.conf : OK
                        Fedora.conf : missing/empty
                        Fedora/i386.conf : missing/empty
                        Fedora/9.conf : missing/empty
                        Fedora/9/i386.conf : missing/empty
```

site.conf : OK
local.conf : OK
Backup with REQUESTRESTORE
REQUESTRESTORE_TEXT = “Please start the restore process on your backup host.

Make sure that you restore the data into '/mnt/local' instead of '/' because the hard disks of the recovered system are mounted there.”

Output to ISO
ISO_DIR = /tmp
ISO_FILES =
ISO_IMAGES =
ISO_ISOLINUX_BIN =
ISO_MKISOFS_BIN = /usr/bin/mkisofs
ISO_PREFIX = ReaR
ISO_VOLID = RELAXRECOVER
RESULT_MAILTO =

Your system is not yet validated. Please carefully check all functions and create a validation record with '/usr/sbin/rear validate'. This will help others to know about the validation status of Relax & Recover on this system.
Finished in 1 seconds.

Workflow – Make Rescue Media

The “mkrescue” workflow will generate no backup, but it will make a rescue ISO image (if output to ISO was selected). We strongly advise novice users to start with the mkrescue workflow to try to understand how rear works.

The “mkbackup” workflow includes the “mkrescue” workflow with an additional backup.

The application will have the following general workflow which is represented by appropriately named scripts in various subdirectories:

1. Prep:
   Prepare the build area by copying a skeleton file system layout. This can also come from various sources (FS layout for arch, OS, OS_VER, Backup-SW, Output, ...)

2. Analyze (DR):
   Analyze the system to create the /etc/recovery data

3. Analyze (Rescue):
   Analyze the system to create the rescue system (network, binary dependencies, ...)

4. Build:
   Build the rescue image by copying together everything required

5. Pack:

Package the kernel and initrd image together

6. Backup:
   (Optionally) run the backup software to create a current backup

7. Output:
   Copy / Install the rescue system (kernel+initrd+(optionally) backups) into the target environment (e.g. PXE boot, write on tape, write on CD/DVD)

8. Cleanup:
   Cleanup the build area from temporary files

The configuration must define the BACKUP and OUTPUT methods. The definition of these variables can be found in */etc/rear/default.conf*. However, if an end-user wants to overrule these he/she can redefine these in the */etc/rear/site.conf* or */etc/rear/local.conf* files.

Valid choices are:

NAME	**TYPE**	**Description**	**Implementation Status**
REQUESTRESTORE	BACKUP	Depends on an external backup program (rear does nothing)	√
NETFS	BACKUP	Copy files to NFS/CIFS share	√
EXTERNAL	BACKUP	Copy files to an external system	√
TAPE	BACKUP	Copy files to tape(s)	Sponsor?
CDROM	BACKUP	Copy files to CD/DVD	Sponsor?
NSR	BACKUP	Use Legato Networker	Sponsor?
TSM	BACKUP	Use Tivoli Storage Manager	√
NBU	BACKUP	Use Symantec Netbackup	√ (Sponsored by J&J)
DP	BACKUP	Use HP Data Protector	√ (Sponsored by HP)
XXX	BACKUP	Use XXX Backup SW	my customer next week :-))
CDROM	OUTPUT	Write result to CD/DVD	Sponsor?
OBDR	OUTPUT	Create OBDR Tape	Sponsor?
PXE	OUTPUT	Create PXE bootable files on TFTP server	√
USB	OUTPUT	Create bootable USB device	√
ISO	OUTPUT	Create an ISO image (default)	√

To show an example of how the "mkrescue" follows the different phases as described above we ran the command "*rear -s mkrescue*":

```
Relax & Recover Version 1.7.9 / 2008-12-02
```

```
Simulation mode activated, ReaR base directory: /usr/share/rear
Source prep/default/01_progress_start.sh
Source prep/ISO/default/30_check_iso_dir.sh
Source prep/ISO/default/32_check_cdrom_size.sh
Source prep/ISO/GNU/Linux/32_verify_mkisofs.sh
Source prep/ISO/Linux-i386/33_find_isolinux.sh
Source prep/default/99_progress_stop.sh
Source dr/default/01_mk_config_dir_recovery.sh
Source dr/GNU/Linux/10_describe_physical_devices.sh
Source dr/GNU/Linux/11_describe_mountpoint_device.sh
Source dr/GNU/Linux/12_describe_filesystems.sh
Source dr/GNU/Linux/13_describe_swap.sh
Source dr/GNU/Linux/15_copy_proc_partitions.sh
Source dr/GNU/Linux/21_describe_md.sh
Source dr/GNU/Linux/23_describe_lvm2.sh
Source dr/GNU/Linux/29_find_required_devices.sh
Source dr/Linux-i386/31_describe_device_properties.sh
Source dr/GNU/Linux/80_copy_fstab_file.sh
Source dr/GNU/Linux/95_cfg2html.sh
Source dr/GNU/Linux/96_collect_MC_serviceguard_infos.sh
Source rescue/default/00_remove_workflow_conf.sh
Source rescue/default/01_merge_skeletons.sh
Source rescue/default/10_hostname.sh
Source rescue/default/20_etc_issue.sh
Source rescue/GNU/Linux/30_dns.sh
Source rescue/GNU/Linux/31_network_devices.sh
Source rescue/GNU/Linux/35_routing.sh
Source rescue/GNU/Linux/39_check_usb_modules.sh
Source rescue/GNU/Linux/40_kernel_modules.sh
Source rescue/default/43_prepare_timesync.sh
Source rescue/GNU/Linux/50_clone_keyboard_mappings.sh
Source rescue/default/50_ssh.sh
Source build/GNU/Linux/20_copy_as_is.sh
Source build/GNU/Linux/39_copy_binaries_libraries.sh
Source build/GNU/Linux/40_copy_modules.sh
Source build/default/50_patch_sshd_config.sh
Source build/default/99_update_os_conf.sh
Source pack/GNU/Linux/00_create_symlinks.sh
Source pack/GNU/Linux/10_touch_empty_files.sh
Source pack/GNU/Linux/20_create_dotfiles.sh
Source pack/Linux-i386/30_copy_kernel.sh
Source pack/GNU/Linux/90_create_initramfs.sh
Source output/default/05_add_cfg2html.sh
Source output/ISO/Linux-i386/30_create_isolinux.sh
Source output/ISO/Linux-i386/80_create_isofs.sh
Source output/default/95_email_result_files.sh
Source cleanup/default/01_progress_start.sh
Source cleanup/default/99_progress_stop.sh
Finished in 1 seconds.
```

Workflow – Recovery

The result of the analysis is written into configuration files under /etc/rear/recovery. This directory is copied together with the other rear directories onto the rescue system where the same framework runs a different workflow – the recovery workflow.

The recovery workflow is triggered by the fact that the root file system is mounted in a ram disk or

tmpfs. Alternatively a „demo“ recovery workflow can be started manually. This will simply recover all data into a subdirectory and not touch the hard disks (looking for a sponsor).

The recovery workflow consists of these parts (identically named modules are indeed the same):

Config:
By utilizing the same configuration module, the same configuration variable are available for the recovery, too. This makes writing pairs of backup/restore modules much easier.

1. Verify:
   Verify the integrity and sanity of the recovery data and check the hardware found to determine, whether a recovery will be likely to succeed. If not, then we abort the workflow so as not to touch the hard disks if we don't believe that we would manage to successfully recover the system on this hardware.

2. Recreate:
   Recreate the FS layout (partitioning, LVM, raid, file systems, ...) and mount it under /mnt/local

3. Restore:
   Restore files and directories from the backup to /mnt/local. This module is the analog to the Backup module

4. Finalize:
   Install boot loader, finalize system, dump recovery log onto /tmp in the recovered system.

FS layout

rear tries to be as much LSB compliant as possible. Therefore, rear will be installed into the usual locations:

/etc/rear/*	Configuration files
/usr/sbin/rear	Main program
/usr/share/rear/*	Internal scripts
/var/rear/*	Recovery data
/tmp/rear.$$	Build area

Layout of /etc/rear

default.conf	Default configuration – will define EVERY variable with a sane default setting. Serves also as a reference for the available variables
site.conf	site wide configuration (optional)
local.conf	local machine configuration (optional)
$(uname -s)-$(uname -i).conf	architecture specific configuration (optional)

default.conf	Default configuration – will define EVERY variable with a sane default setting. Serves also as a reference for the available variables
$(uname -o).conf	OS system (e.g. GNU/Linux.conf) (optional)
$OS/$OS_VER.conf	OS and OS Version specific configuration (optional)
templates/	Directory to keep user-changeable templates for various files used or generated
templates/PXE_per_node_config	template for pxelinux.cfg per-node configurations
templates/CDROM_isolinux.cfg	isolinux.cfg template
templates/...	other templates as the need arises

Note: See also the output of the dump workflow as an example of these settings.

Layout of /usr/share/rear

skel/default/	default rescue FS skeleton
skel/$(uname -i)/	arch specific rescue FS skeleton (optional)
skel/$OS_$OS_VER/	OS-specific rescue FS skeleton (optional)
skel/$BACKUP/	Backup-SW specific rescue FS skeleton (optional)
skel/$OUTPUT/	Output-Method specific rescue FS skeleton (optional)
lib/*.sh	function definitions, split into files by their topic
prep/default/*.sh	Prep scripts: The scripts get merged from the applicable directories and executed in their alphabetical order. Naming conventions are: ##_name.sh where 00 < ## < 99
prep/$(uname -i)/*.sh	
prep/$OS_$OS_VER/*.sh	
prep/$BACKUP/*.sh	
prep/$OUTPUT/*.sh	
dr/default/*.sh	Analyze-DR scripts: The scripts get merged from the applicable directories and executed in their alphabetical order. Naming conventions are: ##_name.sh where 00 < ## < 99
dr/$(uname -i)/*.sh	
dr/$OS_OS_VER/*.sh	
dr/$BACKUP/*.sh	
dr/$OUTPUT/*.sh	
rescue/...	Analyze-Rescue scripts: ...

build/...	Build scripts: ...
pack/...	Pack scripts: ...
backup/$BACKUP/*.sh	Backup scripts: ...
output/$OUTPUT/*.sh	Output scripts: ...
cleanup/...	Cleanup scripts
verify/...	Verify the recovery data against the hardware found, whether we can successfully recover the system
recreate/...	Recreate file systems and their dependencies
restore/$BACKUP/...	Restore data from backup media
finalize/...	Finalization scripts

Note: See the output of e.g. "*rear -s mkrescue (or recover)*" to list the scripts that will be executed by rear.

Layout of /var/rear

/var/rear	Data directory where recovery information is kept
/var/rear/recovery	Recovery data of system itself

Inter-module communication

The various stages and modules communicate via standardized environment variables:

NAME	**TYPE**	**Descriptions**	**Example**
CONFIG_DIR	STRING (RO)	Configuration dir	/etc/rear
SHARE_DIR	STRING (RO)	Shared data dir	/usr/share/rear
BUILD_DIR	STRING (RO)	Build directory	/tmp/rear.$$
ROOTFS_DIR	STRING (RO)	Root FS directory	/tmp/rear.$$/initrd

NAME	TYPE	Descriptions	Example
		for rescue system	
REQUIRED_PROGS	LIST	program files to copy	bash ip route grep ls ...
MODULES MODULES_LOAD	LIST	modules to copy	af_unix e1000 ide-cd ...
COPY_AS_IS	LIST	files (with path) to copy as-is	/etc/localtime ...
COPY_AS_IS_EXCLUDED	LIST	Specific files to be excluded from image	...
...			

RO means that the framework manages this variable and modules and methods shouldn't change it.